ST790-003: Advanced Statistical Computing

Mon/Wed 10:15am-11:30am, SAS Hall 1216

Instructor: Dr Hua Zhou, `hua_zhou@ncsu.edu`

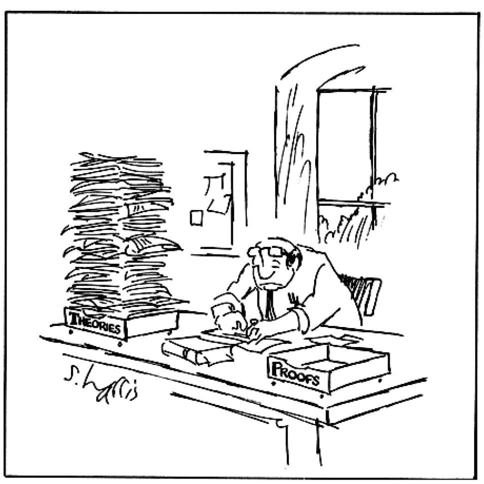
1 Lecture 1: Jan 7

Today

- Introduction and course logistics
- Linux fundamentals

What is this course about?

Statisticians used to ...



Now we spend all time ...

- Statistics, the science of *data analysis*, is the applied mathematics in the 21st century.
- Data is increasing in volume, velocity, and variety. Classification of data sets by Huber (1994, 1996).

Data Size	Bytes	Storage Mode
Tiny	10^2	Piece of paper
Small	10^4	A few pieces of paper
Medium	10^6 (megabyte)	A floppy disk
Large	10^8	Hard disk
Huge	10^9 (gigabytes)	Hard disk(s)
Massive	10^{12} (terabytes)	RAID storage

- Themes of statistics (borrowed from Kenneth Lange's talk)
  - Three pillars: estimation, hypothesis testing, model selection.
  - Two philosophies: frequentist, Bayesian.
  - Mathematical underpinnings: optimization, penalization, asymptotics, integration, Monte Carlo sampling.
  - Statistics is partly empirical and partly mathematical. It is now almost entirely *computational*.
- This course covers some topics on computing I found useful for working statisticians but not covered in ST758 or typical statistics curriculum. *Advanced* does not mean *more difficult* here.
- General topics.
  - Operating systems: Linux and scripting basics
  - Programming languages: R (package development, Rcpp, ...), Matlab, Julia
  - Tools for collaborative and reproducible research: Git, R Markdown, sweave
  - Parallel computing: multi-core, cluster, GPU
  - Convex optimization
  - Integer and mixed integer programming
  - Dynamic programming
  - Advanced topics on EM/MM algorithms
  - Algorithms for sparse regression
  - More advanced optimization methods motivated by modern statistical and machine learning problems, e.g., ALM, ADMM, svm, online algorithms, ...
- Last version (2013 Spring) of this course may give you a rough idea.
  `http://www.stat.ncsu.edu/people/zhou/courses/st810/LectureNotes`
  Of course topics on computing change fast.

Course logistics

- Check course website frequently for updates and announcements.
  `http://hua-zhou.github.io/teaching/st790-2015spr/schedule.html`
  Pre-lecture notes will be posted before each lecture. Cumulative lecture notes will be updated and posted after each lecture.

- My office hours: Mon @ 4P-5P, Wed @ 4P-5P, or by appointment.

- TA office hours: Tue @ 2P-3P, Fri @ 2P-3P, at 1101 SAS Hall.

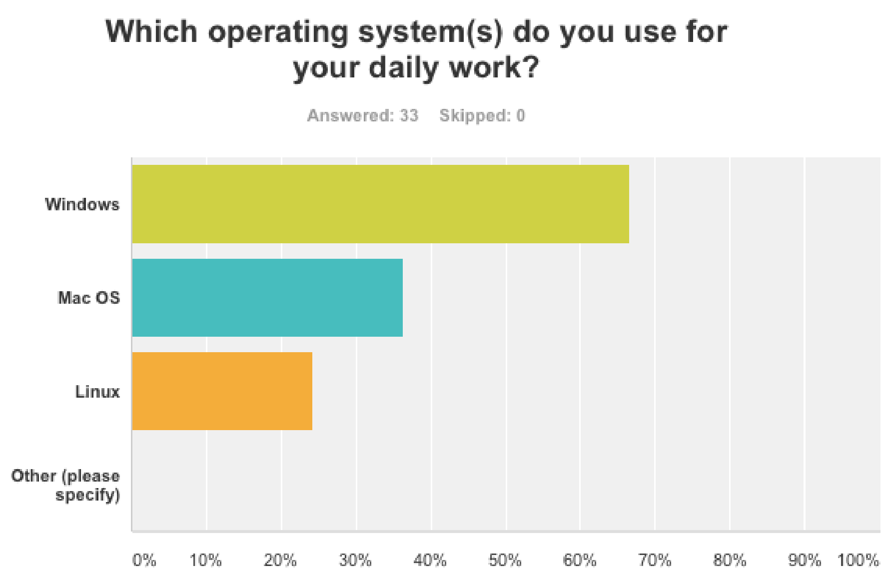
- 5 to 8 homework assignments. Group (20) or solo work (14)?

- A course final project. Survey results: 31 (course project) vs 2 (final exam). Group or solo?

- Final grade: *roughly* 70% HW + 30% final project.

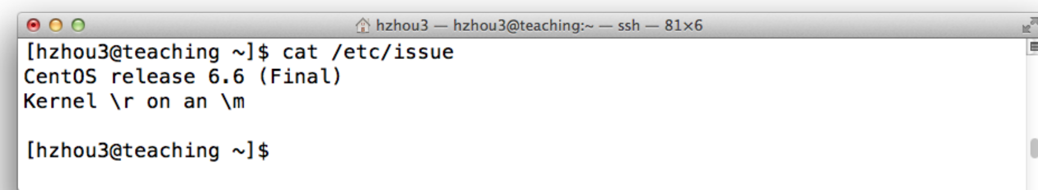
Linux: brief introduction

- Which operating system (OS) are you using? Survey results:

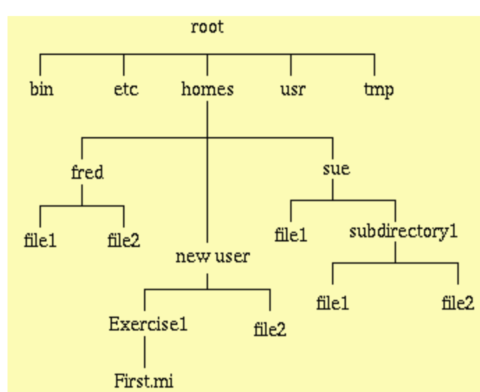


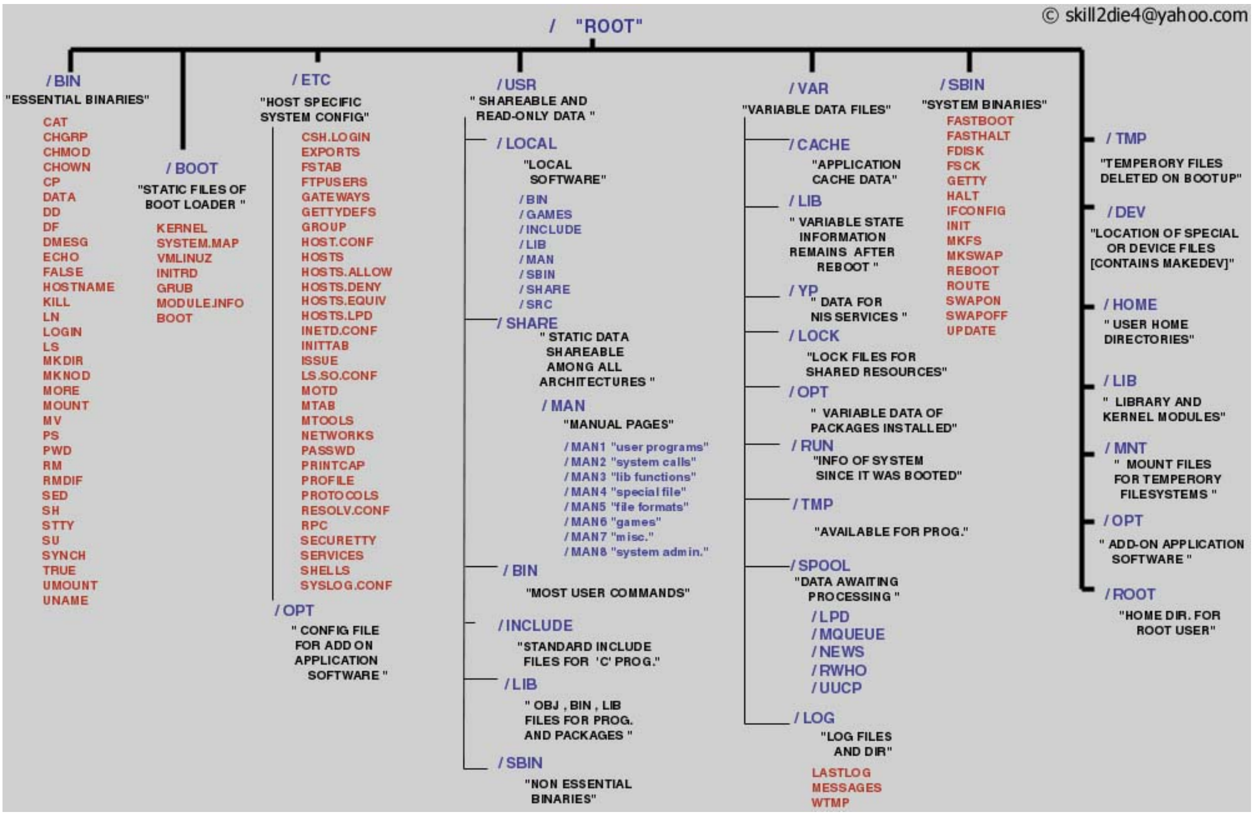
Answer Choices		Responses	
Windows		66.67%	22
Mac OS		36.36%	12
Linux		24.24%	8
Other (please specify)	Responses	0.00%	0
Total Respondents: 33			

- Linux is *the* most common platform for scientific computing.
  - E.g., both department HPC (Beowulf cluster) and campus HPC run on CentOS Linux. It's a lot of computing power sitting there.
  - Open source and community support.
  - Things break, when they break using linux its easy to fix them!
  - Scalability: portable devices (Android, iOS), laptops, servers, and supercomputers.
  - Cost: it's free!
- Distributions of Linux. `http://upload.wikimedia.org/wikipedia/commons/1/1b/Linux_Distribution_Timeline.svg`
  - CentOS is well supported in the department and on campus.
  - Ubuntu is another popular choice for personal computers.
  - `cat /etc/issue` displays the distribution on Linux command line

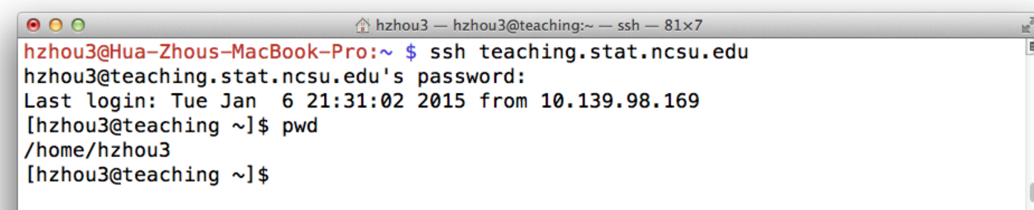


  - ☞ Mac OS was originally derived from Unix/Linux (Darwin kernel). It is POSIX compliant. Most shell commands we review here apply to Mac OS terminal as well. Windows/DOS, unfortunately, is a totally different breed.
- Linux directory structure.





By default, upon log-in user is at his/her home directory



- Linux shells.
  - A shell translates commands to OS instructions.
  - Most commonly used shells: bash, csh, tcsh, ...
  - ☞ Sometimes a script or a command does not run simply because it's written for another shell.
  - Determine the current shell you are working on: `echo $0` or `echo $SHELL`.
  - List available shells: `cat /etc/shells`.

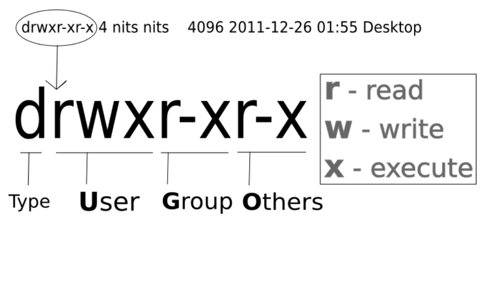
– Change your login shell permanently: `chsh -s /bin/bash userid`. Then log out and log in.

```
[hzhou3@teaching ~]$ echo $SHELL
/bin/tcsh
[hzhou3@teaching ~]$ cat /etc/shells
/bin/sh
/bin/bash
/sbin/nologin
/bin/dash
/bin/tcsh
/bin/csh
[hzhou3@teaching ~]$ chsh -s /bin/bash
Changing shell for hzhou3.
Password:
Shell changed.
[hzhou3@teaching ~]$
```

- Move around the file system.
  - Knowing where you are.
    `pwd` prints the current working directory.
  - `ls` lists contents of a directory.
    `ls -l` lists detailed contents of a directory.
    `ls -a` lists all contents of a directory, including those start with "." (hidden folders).
    ☞ Options for many Linux commands can be combined. E.g., `ls -al`.

```
[hzhou3@teaching ~]$ ls
workspace/
[hzhou3@teaching ~]$ ls -l
total 4
drwxr-xr-x 4 hzhou3 4096 Jan  6 16:39 workspace/
[hzhou3@teaching ~]$ ls -al
total 56
drwx------  6 hzhou3 4096 Jan  6 21:44 ./
drwxr-xr-x 17 root   4096 Jan  6 14:05 ../
-rw-------  1 hzhou3  185 Jan  6 21:50 .bash_history
-rw-r--r--  1 hzhou3   18 Jan  6 14:05 .bash_logout
-rw-r--r--  1 hzhou3  176 Jan  6 14:05 .bash_profile
-rw-r--r--  1 hzhou3  308 Jan  6 14:05 .bashrc
-rw-r--r--  1 hzhou3  319 Jan  6 14:05 .cshrc
-rw-r--r--  1 hzhou3  500 Jan  6 14:05 .emacs
drwxr-xr-x  2 hzhou3 4096 Jan  6 14:05 .gnome2/
-rw-------  1 hzhou3 1698 Jan  6 21:46 .history
drwxr-xr-x  4 hzhou3 4096 Jan  6 14:05 .mozilla/
-rw-------  1 hzhou3    4 Jan  6 16:03 .Rhistory
drwx------  2 hzhou3 4096 Jan  6 16:25 .ssh/
drwxr-xr-x  4 hzhou3 4096 Jan  6 16:39 workspace/
[hzhou3@teaching ~]$
```

– File permissions.



File permissions in Linux

	4	2	1	
0	-	-	-	no permissions
1	-	-	x	only execute
2	-	w	-	only write
3	-	w	x	write and execute
4	r	-	-	only read
5	r	-	x	read and execute
6	r	w	-	read and write
7	r	w	x	read, write and execute

`chmod g+x file` makes a file executable to group members.

`chmod 751 file` sets permission rwxr-x–x to a file.

`groups userid` shows which group(s) a user belongs to.

– ☞ `..` denotes the parent of current working directory.

  ☞ `.` denotes the current working directory.

  ☞ `~` denotes user's home directory.

  `cd ..` changes to parent directory.

  `cd` or `cd ~` changes to home directory.

  `cd /` changes to root directory.

  `pushd` changes the working directory but pushes the current directory into a stack.

  `popd` changes the working directory to the last directory added to the stack.

- Manipulate files and directories.

  – `cp` copies file to a new location.

  – `mv` moves file to a new location.

  – `touch` creates a file, if file already exists it is left unchanged.

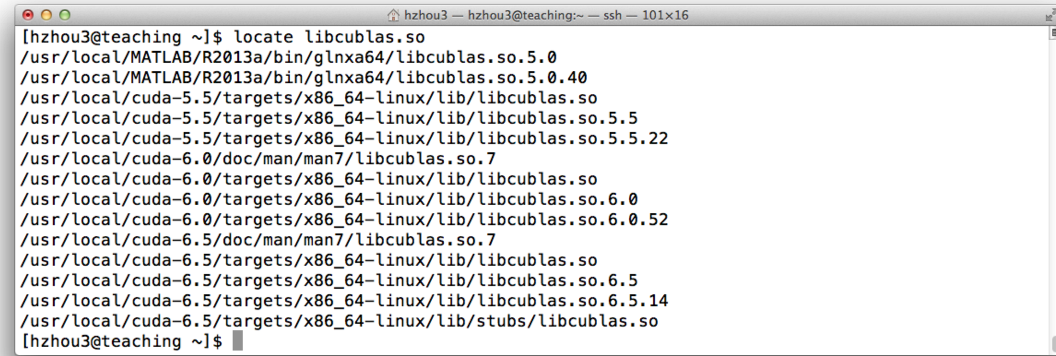
  – `rm` deletes a file.

  – `mkdir` creates a new directory.

  – `rmdir` deletes an *empty* directory.

  – `rm -rf` deletes a directory and all contents in that directory (be cautious using the `-f` option ...)

  – `locate` locates a file by name. E.g., to find files with names containing "libcublas.so"



  - `find` is similar to `locate` but has more functionalities, e.g., select files by age, size, permissions, ... , and is ubiquitous.

- View/peek text files.

  - `cat` prints the contents of a file.
  - `head -l` prints the first l lines of a file
  - `tail -l` prints the last l lines of a file
  - `more` browses a text file screen by screen (only downwards). Scroll down one page (paging) by pressing the spacebar; exit by pressing the `q` key.
  - `less` is also a pager, but has more functionalities, e.g., scroll upwards and downwards through the input.
    ☞ "`less` is more, and `more` is less".
  - `grep` prints lines that match an expression.
  - Wildcard characters:

Wildcard	Matches
? or .	Any single character
*	Any string of characters
+	One or more of preceding pattern
ˆ	beginning of the line
[set]	Any character in set
[!set]	Any character not in the set
[a-z]	Any lowercase letter
[0-9]	Any number (same as [0123456789])

E.g.

```
hzhou3 — hzhou3@teaching:~/workspace/vctest/datasets/GAW18 — ssh — 101×29
[hzhou3@teaching GAW18]$ ls
chr21-geno.bed           chr3-geno-MAP4-849.log                  chr3-geno-MAP4-849.txt
chr21-geno.bim           chr3-geno-MAP4-849-maf5.txt             kinship_all.csv
chr21-geno.fam           chr3-geno-MAP4-849.out                  kinship.csv
chr3-geno-MAP4-849.bed   chr3-geno-MAP4-849-recode12-maf5.map    ped.adj.csv
chr3-geno-MAP4-849.bim   chr3-geno-MAP4-849-recode12-maf5.ped    PED_all.csv
chr3-geno-MAP4-849.fam   chr3-geno-MAP4-849-recode12.map         prepare.R
chr3-geno-MAP4-849.frq   chr3-geno-MAP4-849-recode12.ped
[hzhou3@teaching GAW18]$ head chr21-geno.bim
21      21-9411318      0       9411318 T       C
21      21-9411347      0       9411347 C       G
21      21-9411732      0       9411732 G       T
21      21-9411785      0       9411785 T       G
21      21-9411799      0       9411799 C       T
21      21-9411998      0       9411998 C       T
21      21-9412099      0       9412099 T       C
21      21-9412105      0       9412105 T       A
21      21-9412126      0       9412126 C       T
21      21-9412193      0       9412193 T       C
[hzhou3@teaching GAW18]$ grep 9412105 chr21-geno.bim
21      21-9412105      0       9412105 T       A
[hzhou3@teaching GAW18]$ grep 21-9412[3-7][0-9][0-9] chr21-geno.bim
21      21-9412354      0       9412354 T       C
21      21-9412370      0       9412370 C       A
21      21-9412608      0       9412608 G       A
21      21-9412629      0       9412629 T       C
21      21-9412658      0       9412658 T       C
21      21-9412691      0       9412691 C       A
[hzhou3@teaching GAW18]$
```

– Above wildcards are examples of regular expressions. *Regular expressions* are a powerful tool to efficiently sift through large amounts of text: record linking, data cleaning, scraping data from website or other data-feed. Google 'regular expressions' to learn.

– Piping and redirection.
| sends output from one command as input of another command.
> directs output from one command to a file.
>> appends output from one command to a file.
< reads input from a file.

```
[hzhou3@teaching GAW18]$ wc -l chr21-geno.bim
239352 chr21-geno.bim
[hzhou3@teaching GAW18]$ wc -l < chr21-geno.bim
239352
[hzhou3@teaching GAW18]$ cat chr21-geno.bim | wc -l
239352
[hzhou3@teaching GAW18]$ ls -l /home | grep '^......x'
drwxr-xr-x  5 bsmelton  4096 May 19  2014 bsmelton/
drwxr-xr-x  4 dcoliver  4096 May 16  2014 dcoliver/
drwxr-xr-x  4 laherhol  4096 May 16  2014 laherhol/
drwxr-xr-x  4 mlfurman  4096 May 19  2014 mlfurman/
drwxr-xr-x  5 njmeyer   4096 May 19  2014 njmeyer/
drwxr-xr-x  7 njms      4096 Aug 28 16:26 njms/
drwxr-xr-x  4 npkapur   4096 May 19  2014 npkapur/
drwxr-xr-x  4 rmlaw     4096 May 15  2014 rmlaw/
drwxr-xr-x  5 tawilso3  4096 Dec 27  2013 tawilso3/
drwxr-xr-x  4 wzheng4   4096 May 16  2014 wzheng4/
[hzhou3@teaching GAW18]$
```

– Other useful text editing utilities include
  `sed`, stream editor
  `awk`, filter and report writer
  and so on.

– Combinations of shell commands (`grep`, `sed`, `awk`, ...), piping and redirection, and regular expressions allow us pre-process and reformat huge text files efficiently.

2 Lecture 2, Jan 12

Announcements

- TA office hours changed to Tue @ 1P-2P and Fri @ 2P-3P.
- HW1 posted. Due Mon Jan 19.

Last Time

- Course introduction and logistics.
- Linux introduction: why linux, move around file system, viewing/peeking text files, and simple manipulation of text file.

Today

- Linux introduction (continued).
- Key authentication.
- Version control using Git.

Linux introduction (continued)

- Text editors. “Editor war” `http://en.wikipedia.org/wiki/Editor_war`.

Richard Stallman appearing as St IGNU–cius, a saint in the Church of Emacs

– Emacs is a powerful text editor with extensive support for many languages including R, LaTeX, `python`, and `C/C++`; however it's *not* installed by default on many Linux distributions. Basic survival commands:

  * `emacs filename` to open a file with emacs.
  * `CTRL-x CTRL-f` to open an existing or new file.
  * `CTRL-x CTRX-s` to save.
  * `CTRL-x CTRL-w` to save as.
  * `CTRL-x CTRL-c` to quit.

  Google "emacs cheatsheet" to find something like

GNU Emacs Reference Card

(for version 24)

Starting Emacs

To enter GNU Emacs 24, just type its name: `emacs`

Leaving Emacs

suspend Emacs (or iconify it under X)	`C-z`
exit Emacs permanently	`C-x C-c`

Files

read a file into Emacs	`C-x C-f`
save a file back to disk	`C-x C-s`
save **all** files	`C-x s`
insert contents of another file into this buffer	`C-x i`
replace this file with the file you really want	`C-x C-v`
write buffer to a specified file	`C-x C-w`
toggle read-only status of buffer	`C-x C-q`

Getting Help

The help system is simple. Type `C-h` (or `F1`) and follow the directions. If you are a first-time user, type `C-h t` for a **tutorial**.

remove help window	`C-x 1`
scroll help window	`C-M-v`
apropos: show commands matching a string	`C-h a`
describe the function a key runs	`C-h k`
describe a function	`C-h f`
get mode-specific information	`C-h m`

Error Recovery

abort partially typed or executing command	`C-g`
recover files lost by a system crash	`M-x recover-session`
undo an unwanted change	`C-x u`, `C-_` or `C-/`
restore a buffer to its original contents	`M-x revert-buffer`
redraw garbaged screen	`C-l`

Incremental Search

search forward	`C-s`
search backward	`C-r`
regular expression search	`C-M-s`
reverse regular expression search	`C-M-r`
select previous search string	`M-p`
select next later search string	`M-n`
exit incremental search	`RET`
undo effect of last character	`DEL`
abort current search	`C-g`

Use `C-s` or `C-r` again to repeat the search in either direction. If Emacs is still searching, `C-g` cancels only the part not matched.

Motion

entity to move over	**backward**	**forward**
character	`C-b`	`C-f`
word	`M-b`	`M-f`
line	`C-p`	`C-n`
go to line beginning (or end)	`C-a`	`C-e`
sentence	`M-a`	`M-e`
paragraph	`M-{`	`M-}`
page	`C-x [`	`C-x ]`
sexp	`C-M-b`	`C-M-f`
function	`C-M-a`	`C-M-e`
go to buffer beginning (or end)	`M-<`	`M->`

scroll to next screen	`C-v`
scroll to previous screen	`M-v`
scroll left	`C-x <`
scroll right	`C-x >`
scroll current line to center, top, bottom	`C-l`
goto line	`M-g g`
back to indentation	`M-m`

Killing and Deleting

entity to kill	**backward**	**forward**
character (delete, not kill)	`DEL`	`C-d`
word	`M-DEL`	`M-d`
line (to end of)	`M-0 C-k`	`C-k`
sentence	`C-x DEL`	`M-k`
sexp	`M-- C-M-k`	`C-M-k`

kill **region**	`C-w`
copy region to kill ring	`M-w`
kill through next occurrence of *char*	`M-z` *char*
yank back last thing killed	`C-y`
replace last yank with previous kill	`M-y`

Marking

set mark here	`C-@` or `C-SPC`
exchange point and mark	`C-x C-x`
set mark *arg* **words** away	`M-@`
mark **paragraph**	`M-h`
mark **page**	`C-x C-p`
mark **sexp**	`C-M-@`
mark **function**	`C-M-h`
mark entire **buffer**	`C-x h`

Query Replace

interactively replace a text string	`M-%`
using regular expressions	`M-x query-replace-regexp`

Valid responses in query-replace mode are

replace this one, go on to next	`SPC` or `y`
replace this one, don't move	`,`
skip to next without replacing	`DEL` or `n`
replace all remaining matches	`!`
back up to the previous match	`^`
exit query-replace	`RET`
enter recursive edit (`C-M-c` to exit)	`C-r`

Multiple Windows

When two commands are shown, the second is a similar command for a frame instead of a window.

delete all other windows	`C-x 1`	`C-x 5 1`
split window, above and below	`C-x 2`	`C-x 5 2`
delete this window	`C-x 0`	`C-x 5 0`
split window, side by side	`C-x 3`	
scroll other window	`C-M-v`	
switch cursor to another window	`C-x o`	`C-x 5 o`
select buffer in other window	`C-x 4 b`	`C-x 5 b`
display buffer in other window	`C-x 4 C-o`	`C-x 5 C-o`
find file in other window	`C-x 4 f`	`C-x 5 f`
find file read-only in other window	`C-x 4 r`	`C-x 5 r`
run Dired in other window	`C-x 4 d`	`C-x 5 d`
find tag in other window	`C-x 4 .`	`C-x 5 .`
grow window taller	`C-x ^`	
shrink window narrower	`C-x {`	
grow window wider	`C-x }`	

Formatting

indent current **line** (mode-dependent)	`TAB`
indent **region** (mode-dependent)	`C-M-\`
indent **sexp** (mode-dependent)	`C-M-q`
indent region rigidly *arg* columns	`C-x TAB`
indent for comment	`M-;`
insert newline after point	`C-o`
move rest of line vertically down	`C-M-o`
delete blank lines around point	`C-x C-o`
join line with previous (with arg, next)	`M-^`
delete all white space around point	`M-\`
put exactly one space at point	`M-SPC`
fill paragraph	`M-q`
set fill column to *arg*	`C-x f`
set prefix each line starts with	`C-x .`
set face	`M-o`

Case Change

uppercase word	`M-u`
lowercase word	`M-l`
capitalize word	`M-c`
uppercase region	`C-x C-u`
lowercase region	`C-x C-l`

The Minibuffer

The following keys are defined in the minibuffer.

complete as much as possible	`TAB`
complete up to one word	`SPC`
complete and execute	`RET`
show possible completions	`?`
fetch previous minibuffer input	`M-p`
fetch later minibuffer input or default	`M-n`
regexp search backward through history	`M-r`
regexp search forward through history	`M-s`
abort command	`C-g`

Type `C-x ESC ESC` to edit and repeat the last command that used the minibuffer. Type `F10` to activate menu bar items on text terminals.

  `C-<key>` means hold the control key, and press `<key>`

  `M-<key>` means press the Esc key once, and press `<key>`

– `vi` is ubiquitous (POSIX standard). Learn at least its basics; otherwise you can edit nothing on some clusters. Basic survival commands:

  * `vi filename` to start editing a file.

  * `vi` is a *modal* editor: *insert* mode and *normal* mode. Pressing `i` switches from the normal mode to insert mode. Pressing `ESC` switches from the insert mode to normal mode.
  * `:x<Return>` quit vi and save changes.
  * `:wq<Return>` quit vi and save changes.
  * `:q!<Return>` quit vi without saving latest changes.
  * `:w<Return>` saves changes.

Google "vi cheatsheet" to find something like

Vi Command Cheat Sheet

Quitting

:x	Exit, saving changes
:q	Exit as long as there have been no changes
ZZ	Exit and save changes if any have been made
:q!	Exit and ignore any changes

Inserting Text

i	Insert before cursor
I	Insert before line
a	Append after cursor
A	Append after line
o	Open a new line after current line
O	Open a new line before current line
r	Replace one character
R	Replace many characters

Deleting Text

Almost all deletion commands are performed by typing d followed by a motion.

dw	Delete word
x	Delete character to the right of cursor
X	Delete character to the left of cursor
D	Delete to the end of the line
dd	Delete current line
:d	Delete current line

Yanking Text

Almost all yank commands are performed by typing y followed by a motion.

y$	Yank to the end of the line
yy	Yank the current line
:y	Yank the current line

Changing text

The change command is a deletion command that leaves the editor in insert mode. It is performed by typing c followed by a motion.

cw	Change word
C	Change to the end of the line
cc	Change the whole line

Putting text

p	Put after the position or after the line
P	Put before the position or before the line

Motion

h	Move left
j	Move down
k	Move up
l	Move right
w	Move to next word
W	Move to next blank delimited word
b	Move to the beginning of the word
B	Move to the beginning of blank delimited word
e	Move to the end of the word
E	Move to the end of blank delimited word
(	Move a sentence back
)	Move a sentence forward
{	Move a paragraph back
}	Move a paragraph forward
0	Move to the beginning of the line
$	Move to the end of the line
1G	Move to the first line of the file
G	Move to the last line of the file
nG	Move to nth line of the file
:n	Move to nth line of the file
fc	Move forward to c
Fc	Move back to c
H	Move to top of screen
M	Move to middle of screen
L	Move to button of screen
Ctrl+u	Page up
Ctrl+d	Page down
%	Move to associated ( ), { }, [ ]

Search for strings

/string	Search forward for string
?string	Search back for string
n	Search for next instance of string
N	Search for previous instance of string

Other

~	Toggle capital and lower-case
J	Join lines
.	Repeat last text-changing command
u	Undo last change
U	Undo all changes to line

Based on http://www.lagmonster.org/docs/vi.html

Buffers

Named buffers may be specified before any deletion, change, yank or put command. The general prefix has the form "c where c is any lowercase character. for example, "adw deletes a word into buffer a. It may thereafter be put back into text with an appropriate "ap.

Markers

Named markers may be set on any line in a file. Any lower case letter may be a marker name. Markers may also be used as limits for ranges.

mc	Set marker c on this line
`c	Go to beginning of marker c line.
'c	Go to first non-blank character of marker c line.

Replace

The search and replace function is accomplished with the :s command. It is commonly used in combination with ranges or the :g command (below).

:s/pattern/string/flags	Replace pattern with string according to flags.
g	Flag - Replace all occurrences of pattern
c	Flag - Confirm replaces.
&	Repeat last :s command

Counts

Nearly every command may be preceded by a number that specifies how many times it is to be performed. For example, 5dw will delete 5 words and 3fe will move the cursor forward to the 3rd occurrence of the letter e.

Ranges

Ranges may precede most "colon" commands and cause them to be executed on a line or lines. For example :3,7d would delete lines 3-7.

:n,m	Range - Lines n-m
:.	Range - Current line
:$	Range - Last line
:'c	Range - Marker c
:%	Range - All lines in file
:g/pattern/	Range - All lines that contain pattern

Files

:w file	Write to file
:r file	Read file in after line
:n	Go to next file
:p	Go to previous file
:e file	Edit file
!!program	Replace line with output from program

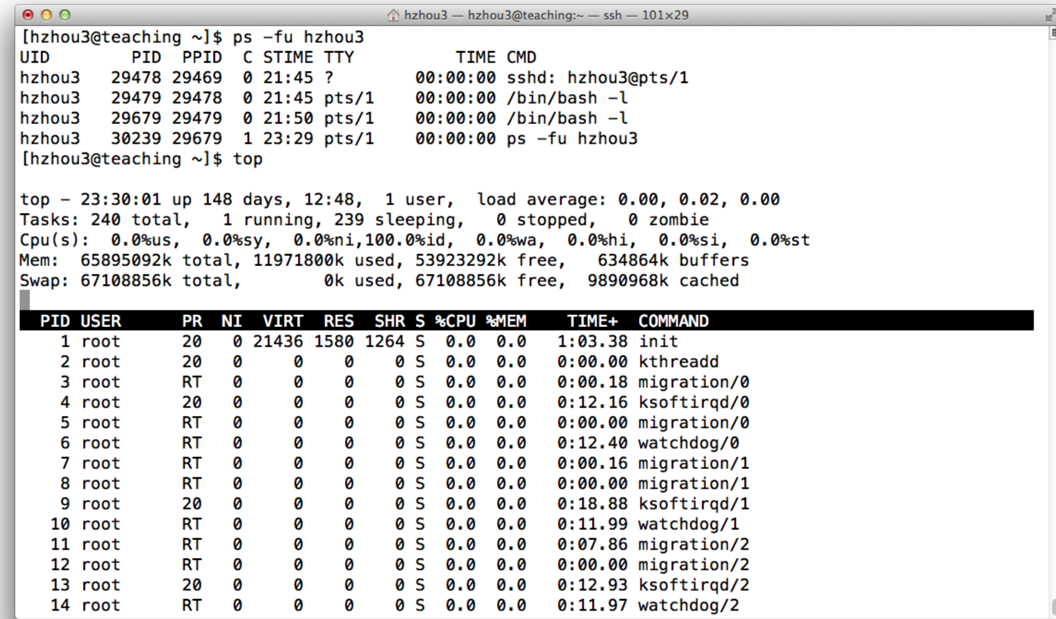
– Statisticians write a lot of code. Critical to adopt a good IDE (integrated development environment) that goes beyond code editing: syntax highlighting, executing code within editor, debugging, profiling, version control, ...
  R Studio, Matlab, Visual Studio, Eclipse, Emacs, ...

- Bash completion. Bash provides the following standard completion for the Linux users by default. Much less typing errors and time!

1. Pathname completion
2. Filename completion
3. Variablename completion
   E.g., `echo $[TAB][TAB]`
4. Username completion
   E.g., `cd ~[TAB][TAB]`
5. Hostname completion
   E.g., `ssh hzhou3@[TAB][TAB]`

  It can also be customized to auto-complete other stuff such as options and command's arguments. Google "bash completion" for more information.

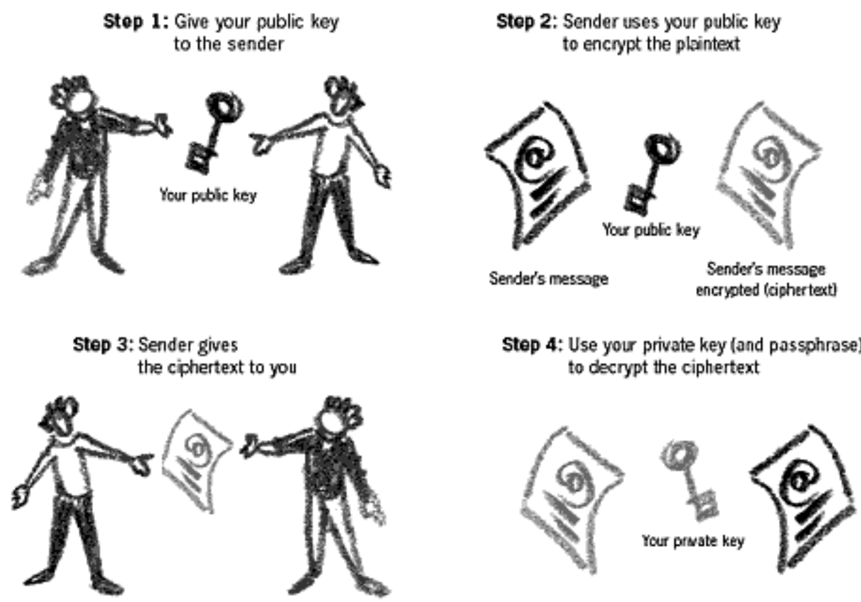
- OS runs processes on behalf of user.
  - Each process has Process ID (PID), Username (UID), Parent process ID (PPID), Time and data process started (STIME), time running (TIME), ...
  - `ps` command provides info on processes.
    `ps -eaf` lists all currently running processes
    `ps -fp 1001` lists process with PID=1001
    `ps -eaf | grep python` lists all python processes
    `ps -fu userid` lists all processes owned by a user.
  - `kill` kills a process. E.g., `kill 1001` kills process with PID=1001.
    `killall` kills a bunch of processes. E.g., `killall -r R` kills all R processes.
  - `top` prints realtime process info (very useful).



```
[hzhou3@teaching ~]$ ps -fu hzhou3
UID        PID  PPID  C STIME TTY          TIME CMD
hzhou3   29478 29469  0 21:45 ?        00:00:00 sshd: hzhou3@pts/1
hzhou3   29479 29478  0 21:45 pts/1    00:00:00 /bin/bash -l
hzhou3   29679 29479  0 21:50 pts/1    00:00:00 /bin/bash -l
hzhou3   30239 29679  1 23:29 pts/1    00:00:00 ps -fu hzhou3
[hzhou3@teaching ~]$ top

top - 23:30:01 up 148 days, 12:48,  1 user,  load average: 0.00, 0.02, 0.00
Tasks: 240 total,   1 running, 239 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:  65895092k total, 11971800k used, 53923292k free,   634864k buffers
Swap: 67108856k total,        0k used, 67108856k free,  9890968k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
    1 root      20   0 21436 1580 1264 S  0.0  0.0   1:03.38 init
    2 root      20   0     0    0    0 S  0.0  0.0   0:00.00 kthreadd
    3 root      RT   0     0    0    0 S  0.0  0.0   0:00.18 migration/0
    4 root      20   0     0    0    0 S  0.0  0.0   0:12.16 ksoftirqd/0
    5 root      RT   0     0    0    0 S  0.0  0.0   0:00.00 migration/0
    6 root      RT   0     0    0    0 S  0.0  0.0   0:12.40 watchdog/0
    7 root      RT   0     0    0    0 S  0.0  0.0   0:00.16 migration/1
    8 root      RT   0     0    0    0 S  0.0  0.0   0:00.00 migration/1
    9 root      20   0     0    0    0 S  0.0  0.0   0:18.88 ksoftirqd/1
   10 root      RT   0     0    0    0 S  0.0  0.0   0:11.99 watchdog/1
   11 root      RT   0     0    0    0 S  0.0  0.0   0:07.86 migration/2
   12 root      RT   0     0    0    0 S  0.0  0.0   0:00.00 migration/2
   13 root      20   0     0    0    0 S  0.0  0.0   0:12.93 ksoftirqd/2
   14 root      RT   0     0    0    0 S  0.0  0.0   0:11.97 watchdog/2
```

(Seamless) remote access to Linux machines

- SSH (secure shell) is the dominant cryptographic network protocol for secure network connection via an insecure network.
  - On Linux or Mac, access the teaching server by `ssh unityid@teaching.stat.ncsu.edu`
  - Windows machines need the PuTTY program (free).
- Forget about passwords. Use keys! Why?
  - Much more secure. Most passwords are weak.
  - Script or a program may need to systematically SSH into other machines.
  - Log into multiple machines using the same key.
  - Seamless use of many other services: Git, svn, Amazon EC2 cloud service, parallel computing on multiple hosts in Julia, ...
  - Many servers only allow key authentication and do not accept password authentication. E.g., NCSU arc cluster.
- Key authentication.



- Public key. Put on the machine(s) you want to log in.
- Private key. Put on your own computer. Consider this as the actual key in your pocket; never give to others.
- Messages from server to your computer is encrypted with your public key. It can only be decrypted using your private key.
- Messages from your computer to server is signed with your private key (digital signatures) and can be verified by anyone who has your public key (authentication).

- Generate keys.

  1. On Linux or Mac, **`ssh-keygen`** generates key pairs. E.g., on the teaching server

```
hzhou3 — hzhou3@teaching:~/.ssh — ssh — 83×32
[hzhou3@teaching ~]$ pwd
/home/hzhou3
[hzhou3@teaching ~]$ mkdir .ssh
[hzhou3@teaching ~]$ cd .ssh
[hzhou3@teaching ~/.ssh]$ ssh-keygen -t dsa -f hzhou_key
Generating public/private dsa key pair.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in hzhou_key.
Your public key has been saved in hzhou_key.pub.
The key fingerprint is:
20:d3:a2:ae:08:82:ea:3e:08:2a:b3:82:f8:d4:f4:81 hzhou3@teaching.stat.ncsu.edu
The key's randomart image is:
+--[ DSA 1024]----+
|                 |
|     .           |
|    + o          |
|   . = .         |
|  . E . S        |
|o. o . .         |
|X o . .          |
|&+               |
|X*o              |
+-----------------+
[hzhou3@teaching ~/.ssh]$ ls -al
total 16
drwxr-xr-x 2 hzhou3 4096 Jan  6 14:52 ./
drwxr-xr-x 5 hzhou3 4096 Jan  6 14:52 ../
-rw------- 1 hzhou3  668 Jan  6 14:52 hzhou_key
-rw-r--r-- 1 hzhou3  619 Jan  6 14:52 hzhou_key.pub
[hzhou3@teaching ~/.ssh]$
```

     Use a (optional) paraphrase different form password.

  2. Set right permissions on the `.ssh` folder and key files

```
hzhou3 — hzhou3@teaching:~/.ssh — ssh — 83×10
[hzhou3@teaching ~/.ssh]$ chmod 700 ~/.ssh
[hzhou3@teaching ~/.ssh]$ chmod 600 hzhou_key*
[hzhou3@teaching ~/.ssh]$ ls -al
total 16
drwx------ 2 hzhou3 4096 Jan  6 14:52 ./
drwxr-xr-x 5 hzhou3 4096 Jan  6 14:52 ../
-rw------- 1 hzhou3  668 Jan  6 14:52 hzhou_key
-rw------- 1 hzhou3  619 Jan  6 14:52 hzhou_key.pub
[hzhou3@teaching ~/.ssh]$
```

3. Append the public key to the `~/.ssh/authorized_keys` file of any Linux machine we want to SSH to, e.g., the Beowulf cluster (`hpc.stat.ncsu.edu`).

```
hzhou3 — hzhou3@stat02bw:~/.ssh — ssh — 83×24
[hzhou3@teaching ~/.ssh]$ scp hzhou_key.pub hzhou3@hpc.stat.ncsu.edu:~/.ssh/
The authenticity of host 'hpc.stat.ncsu.edu (152.1.228.92)' can't be established.
RSA key fingerprint is 61:f1:d1:07:a9:14:4f:cb:9c:3c:a2:6f:a7:3e:8c:78.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'hpc.stat.ncsu.edu,152.1.228.92' (RSA) to the list of kn
own hosts.
hzhou3@hpc.stat.ncsu.edu's password:
hzhou_key.pub                                 100%  619     0.6KB/s   00:00
[hzhou3@teaching ~/.ssh]$ ssh hpc.stat.ncsu.edu
hzhou3@hpc.stat.ncsu.edu's password:
Last login: Tue Jan  6 10:30:37 2015 from 10.139.98.169
[hzhou3@hpc ~]$  cd .ssh/
[hzhou3@hpc .ssh]$  ls -al
total 196
drwx------ 2 hzhou3 ncsu          64 Jan  6 15:08 .
drwx------ 2 hzhou3 4294967294 65536 Dec  9 14:00 ..
-rw-r--r-- 1 hzhou3 ncsu         414 Dec  9 14:00 authorized_keys
-rw------- 1 hzhou3 ncsu         619 Jan  6 15:08 hzhou_key.pub
-rw------- 1 hzhou3 ncsu         668 Jul 23 10:45 id_dsa
-rw------- 1 hzhou3 ncsu        1675 Dec  9 14:00 id_rsa
-rw-r--r-- 1 hzhou3 ncsu         414 Dec  9 14:00 id_rsa.pub
-rw-r--r-- 1 hzhou3 ncsu        2042 Jan  6 10:47 known_hosts
[hzhou3@hpc .ssh]$  cat hzhou_key.pub >> authorized_keys
[hzhou3@hpc .ssh]$
```

4. Now you don't need password each time you connect from the teaching server to the Beowulf cluster.

5. If you set paraphrase when generating keys, you'll be prompted for the paraphrase each time the private key is used. Avoid repeatedly entering the paraphrase by using `ssh-agent` on Linux/Mac or Pagent on Windows.

   ☞ Same key pair can be used between any two machines. We don't need to regenerate keys for each new connection.

   ☞ For Windows users, the private key generated by `ssh-keygen` cannot be directly used by PuTTY; use PuTTYgen for conversion. Then let PuTTYgen use the converted private key. Read Sections A and B of the tutorial `http://tipsandtricks.nogoodatcoding.com/2010/02/svnssh-with-tortoisesvn.html`

- Transfer files between machines.
  - `scp` copies files via SSH.
    `scp filehere unityid@teaching.stat.ncsu.edu:~/remotefolder`
    `scp unityid@teaching.stat.ncsu.edu:~/remotefile folderhere`
  - `sftp` is FTP via SSH.

- GUIs for Windows (WinSCP) or Mac (Cyberduck).
- (My preferred way) Use a version control system to sync project files between different machines and systems.

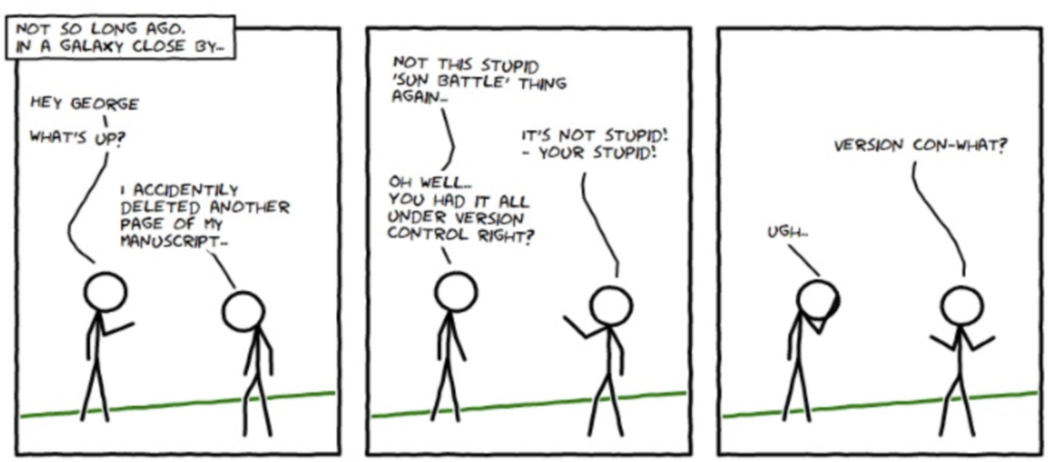
- ☞ Line breaks in text files. Windows uses a pair of `CR` and `LF` for line breaks. Linux/Unix uses an `LF` character only. Mac X also uses a single `LF` character. But old Mac OS used a single `CR` character for line breaks. If transferred in binary mode (bit by bit) between OSs, a text file could look a mess. Most transfer programs automatically switch to text mode when transferring text files and perform conversion of line breaks between different OSs; but I used to run into problems using WinSCP. Sometimes you have to tell WinSCP explicitly a text file is being transferred.

Summary of Linux

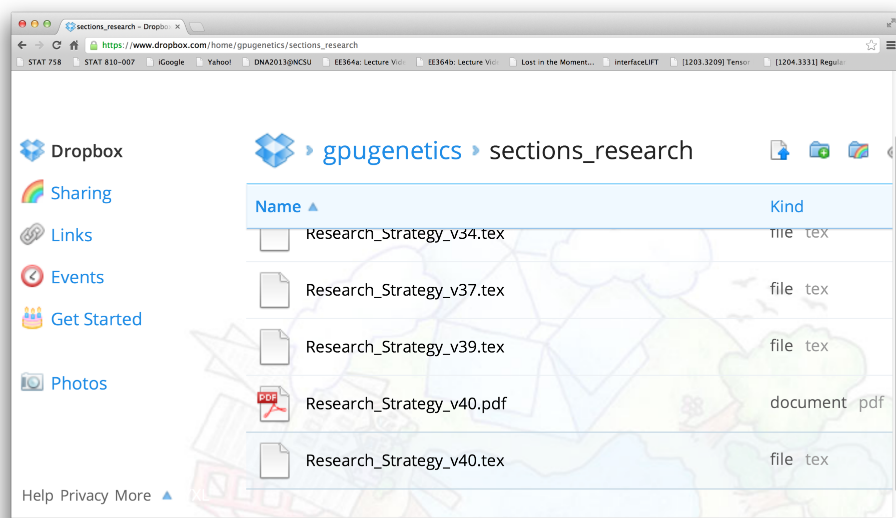
- Practice Linux machine for this class:
  `teaching.stat.ncsu.edu`
  Start using it *right now*.
- Ask for help (order matters): Google (paste the error message to Google often helps), `man` command if no internet access, friends, Terry, ...
- Homework (ungraded): set up keys for connecting your own computer to the teaching server.

Version control by Git

If it's not in source control, it doesn't exist.



- Collaborative research. Statisticians, as opposed to "closet mathematicians", rarely do things in vacuum.
  - We talk to scientists/clients about their data and questions.
  - We write code (a lot!) together with team members or coauthors.
  - We run code/program on different platforms.
  - We write manuscripts/reports with co-authors.
  - ...
- 4 things distinguish professional programmers from amateurs:
  - *Use a version control system.*
  - Automate repetitive taks.
  - Systematic testing.
  - Use debugging aids rather than print statements.
- Why version control?
  - A centralized repository helps coordinate multi-person projects.
  - Synchronize files across multiple computers and platforms.
  - Time machine. Keep track of all the changes and revert back easily (reproducible).
  - Storage efficiency. This is what I often see ...



- Available version control tools.

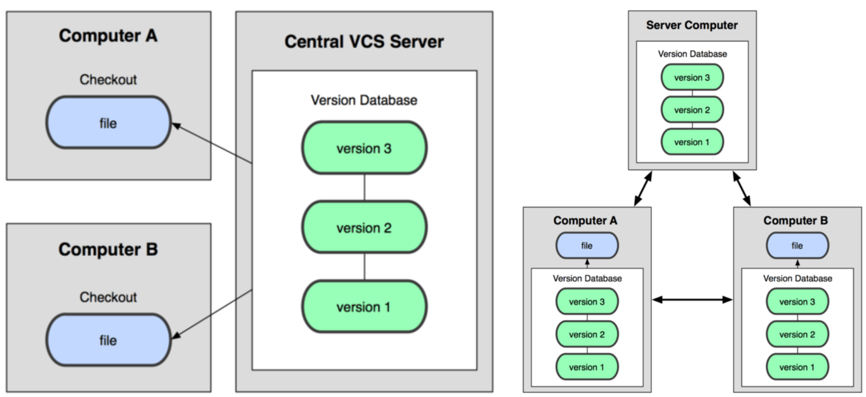
- Open source: cvs, subversion (aka svn), Git, ...
- Proprietary: Visual SourceSafe (VSS), ...
- Dropbox? Mostly for file back and sharing, limited version control (1 month?), ...

We use Git in this course.

- Why Git?
  - The Eclipse Community Survey in 2014 shows Git is the most widely used source code management tool *now*. Git (33.3%) vs svn (30.7%).
  - History: Initially designed and developed by Linus Torvalds in 2005 for Linux kernel development. "`git`" is the British English slang for "unpleasant person".

    > I'm an egotistical bastard, and I name all my projects after myself. First 'Linux', now 'git'.
    >
    > Linus Torvalds

  - A fundamental difference between svn (centralized version control system, left plot) and Git (distributed version control system, right plot):



  - Advantages of Git.
    * Speed and simple (?) design.
    * Strong support for non-linear development (1000s of parallel branches).
    * Fully distributed. Fast, no internet required, disaster recovery,
    * Scalable to large projects like the Linux kernel project.
    * Free and open source.
  - Be aware that svn is still widely used in IT industry (Apache, GCC, SourceForge, Google Code, ...) and R development. E.g., type
    `svn log -v -l 5 https://svn.r-project.org/R`
    on command line to get a glimpse of what R development core team is doing.

- Good to master some basic svn commands.

- What do I need to use Git?
  - A Git server enabling multi-person collaboration through a centralized repository.
    * `github.com`: unlimited public repositories, private repositories costs $, academic user can get 5 private repositories for free.
    * `github.ncsu.edu`: unlimited public or private repositories, but space limitation (300M?), not accessible by non-NCSU collaborators.
    * `bitbucket.org`: unlimited private repositories for academic account (register for free using your NCSU email).

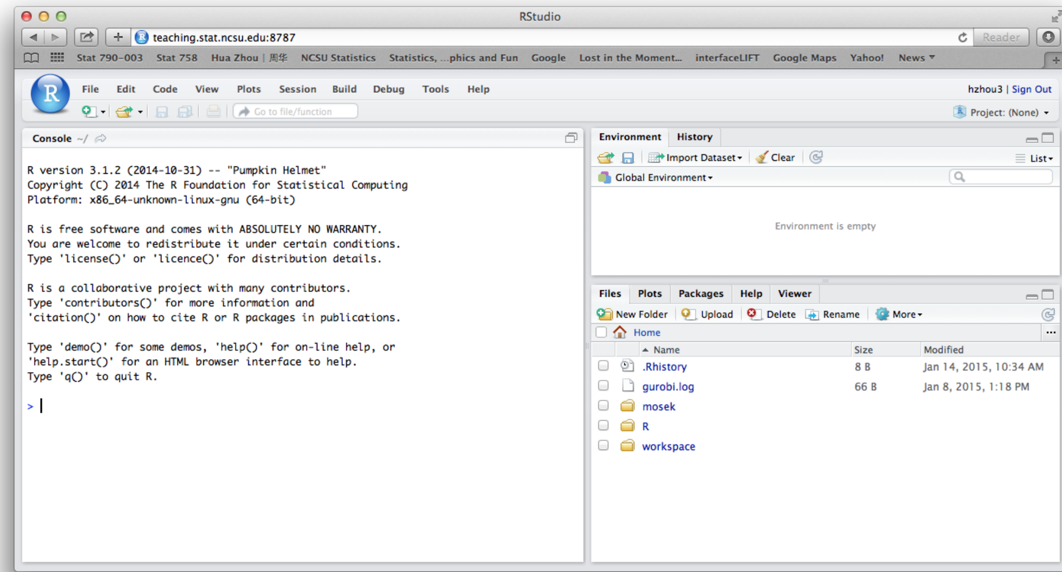
  ☞ For this course, use `github.ncsu.edu` please.
  - Git client.
    * Linux: installed on many servers, including `teaching.stat.ncsu.edu` and `hpc.stat.ncsu.edu`. If not, install on CentOS by `yum install git`.
    * Mac: install by `port install git`.
    * Windows: GitHub for Windows (GUI), TortoiseGIT (is this good?)

  ☞ Don't rely on GUI. Learn to use Git on command line.

3 Lecture 3, Jan 21

Announcements

- *Today*'s office hours change to 5P-6P.
- Install Linux on your personal computer?
- Want to use R Studio on teaching server?



Access via `http://teaching.stat.ncsu.edu:8787`. However you need to change password on command line (`passwd`).

Last Time

- Key authentication.
- Version control using Git.

Today

- Version control using Git (cont'd).
- Reproducible research.
- Next week: languages (R, Matlab, Julia)

Version control using Git (cont'd)

- Life cycle of a project.

  Stage 1:

  - A project (idea) is born on `github.ncsu.edu`, with directories say `codebase`, `datasets`, `manuscripts`, `talks`, ...
  - Advantage of `github.ncsu.edu`: privacy of research ideas (free private repositories).
  - Downside of `github.ncsu.edu`: not accessible by off-campus collaborators; 300M storage limit.
  - `bitbucket.org` is a good alternative. Unlimited private repositories for academic accounts (register with `.edu` email).

  Stage 2:

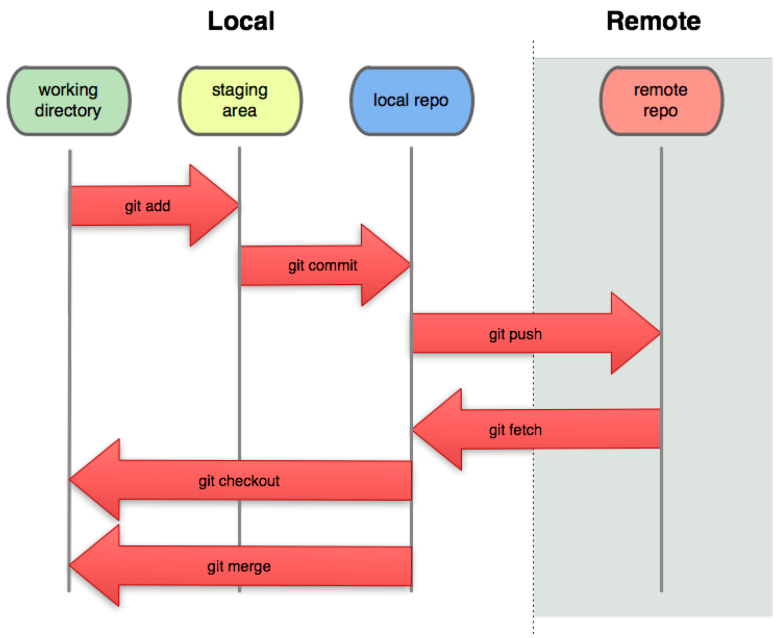
  - Hopefully, research idea pans out and we want to put up a standalone software development repository at `github.com`.
  - This usually inherits from the `codebase` folder and happens when we submit a paper.
  - Challenges: keep all version history. Read Cai Li's slides (`http://hua-zhou.github.io/teaching/st790-2015spr/gitslides-CaiLi.pdf`) for how to migrate part of a project to a new repository while keeping all history.

  Stage 3:

  - Active maintenance of the public software repository.
  - At least three branches: `develop`, `master`, `gh-pages`.
    `develop`: main development area.
    `master`: software release.
    `gh-pages`: software webpage.
  - Maintaining and distributing software on `github.com`.

  ☞ Josh Day will cover how to distribute R package from `github` next week.

- Basic workflow of Git.



- Synchronize local Git directory with remote repository (`git pull`).
- Modify files in local working directory.
- Add snapshots of them to staging area (`git add`).
- Commit: store snapshots permanently to (local) Git repository (`git commit`).
- Push commits to remote repository (`git push`).

• Basic Git usage.

- Register for an account on a Git server, e.g., `github.ncsu.edu`. Fill out your profile, upload your public key to the server, ...
- Identify yourself at local machine:
  `git config --global user.name "Hua Zhou"`
  `git config --global user.email "hua_zhou@ncsu.edu"`
  Name and email appear in each commit you make.
- Initialize a project:
  * Create a repository, e.g., `st790-2015spr`, on the server `github.ncsu.edu`. Then clone to local machine
    `git clone git@github.ncsu.edu:unityID/st790-2015spr.git`
  * Alternatively use following commands to initialize a Git directory from a local folder and then push to the Git server
    `git init`

```
git remote add origin git@github.ncsu.edu:unityID/st790-2015spr.git
git push -u origin master
```

– Edit working directory.

`git pull` update local Git repository with remote repository (fetch + merge).

`git status` displays the current status of working directory.

`git log filename` displays commit logs of a file.

`git diff` shows differences (by default difference from the most recent commit).

`git add ...` adds file(s) to the staging area.

`git commit` commits changes in staging area to Git directory.

`git push` publishes commits in local Git directory to remote repository.

Following demo session is on my local Mac machine.

```
hzhou3@Hua-Zhous-MacBook-Pro:mglm $ pwd
/Users/hzhou3/github.ncsu/mglm
hzhou3@Hua-Zhous-MacBook-Pro:mglm $ ls
.DS_Store    .gitignore    datasets/     manuscripts/
.git/        codebase/     literature/   talks/
hzhou3@Hua-Zhous-MacBook-Pro:mglm $ git pull
remote: Counting objects: 5, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 5 (delta 3), reused 5 (delta 3)
Unpacking objects: 100% (5/5), done.
From github.ncsu.edu:hzhou3/vctest
   80be212..b22d29f  master     -> origin/master
Updating 80be212..b22d29f
Fast-forward
 manuscripts/letter-skat-famskat/Letter_to_the_editor.tex | 4 ++--
 1 file changed, 2 insertions(+), 2 deletions(-)
hzhou3@Hua-Zhous-MacBook-Pro:mglm $ echo "hello st790 class" > gitdemo.txt
hzhou3@Hua-Zhous-MacBook-Pro:mglm $ ls
.DS_Store    .gitignore    datasets/     literature/  talks/
.git/        codebase/     gitdemo.txt   manuscripts/
```

```
hzhou3@Hua-Zhous-MacBook-Pro:mglm $ git add gitdemo.txt
hzhou3@Hua-Zhous-MacBook-Pro:mglm $ git status .
On branch master
Your branch is up-to-date with 'origin/master'.
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

	new file:   gitdemo.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)

	codebase/Example_RNAseq_top100/
	codebase/MGLM/R/.Rhistory
hzhou3@Hua-Zhous-MacBook-Pro:mglm $ git commit -m "git demo for st790 class"
[master ea636ff] git demo for st790 class
 1 file changed, 1 insertion(+)
 create mode 100644 gitdemo.txt
hzhou3@Hua-Zhous-MacBook-Pro:mglm $ git log gitdemo.txt
commit ea636ff5665bc26bf8a79751b75d0e9d67bdb7d1
Author: Hua Zhou <hua_zhou@ncsu.edu>
Date:   Sun Jan 11 17:06:23 2015 -0500

    git demo for st790 class
hzhou3@Hua-Zhous-MacBook-Pro:mglm $ git push
Counting objects: 3, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 301 bytes | 0 bytes/s, done.
Total 3 (delta 1), reused 0 (delta 0)
To git@github.ncsu.edu:hzhou3/mglm.git
   77145d2..ea636ff  master -> master
```

`git reset --soft HEAD 1` undo the last commit.

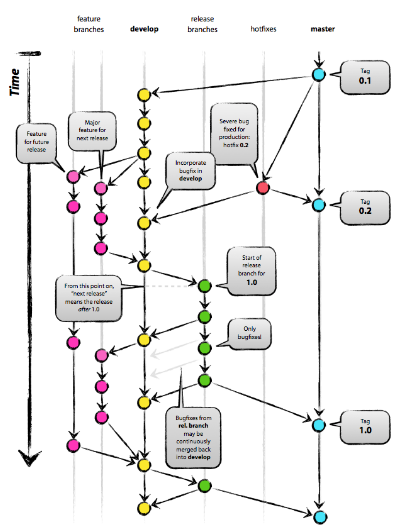
`git checkout filename` go back to the last commit.

`git rm` different from `rm`.

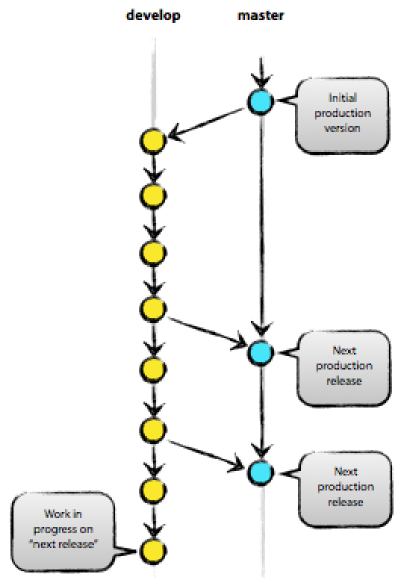
☞ Although `git rm` deletes files from working directory. They are still in Git history and can be retrieved whenever needed. So always be cautious to put large data files or binary files into version control.

```
hzhou3@Hua-Zhous-MacBook-Pro:mglm $ echo "bye st790 class" >> gitdemo.txt
hzhou3@Hua-Zhous-MacBook-Pro:mglm $ cat gitdemo.txt
hello st790 class
bye st790 class
hzhou3@Hua-Zhous-MacBook-Pro:mglm $ git diff gitdemo.txt
diff --git a/gitdemo.txt b/gitdemo.txt
index ece6d4e..2bb77f8 100644
--- a/gitdemo.txt
+++ b/gitdemo.txt
@@ -1 +1,2 @@
 hello st790 class
+bye st790 class
hzhou3@Hua-Zhous-MacBook-Pro:mglm $ git checkout gitdemo.txt
hzhou3@Hua-Zhous-MacBook-Pro:mglm $ cat gitdemo.txt
hello st790 class
hzhou3@Hua-Zhous-MacBook-Pro:mglm $ git rm gitdemo.txt
rm 'gitdemo.txt'
hzhou3@Hua-Zhous-MacBook-Pro:mglm $ git status .
On branch master
Your branch is up-to-date with 'origin/master'.
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

	deleted:    gitdemo.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)

	codebase/Example_RNAseq_top100/
	codebase/MGLM/R/.Rhistory
hzhou3@Hua-Zhous-MacBook-Pro:mglm $ git commit -m "delete the git demo file for st790"
[master 4b4f9c5] delete the git demo file for st790
 1 file changed, 1 deletion(-)
 delete mode 100644 gitdemo.txt
hzhou3@Hua-Zhous-MacBook-Pro:mglm $ git push
Counting objects: 2, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 238 bytes | 0 bytes/s, done.
Total 2 (delta 1), reused 0 (delta 0)
lTo git@github.ncsu.edu:hzhou3/mglm.git
   ea636ff..4b4f9c5  master -> master
hzhou3@Hua-Zhous-MacBook-Pro:mglm $ ls
.DS_Store    .gitignore   datasets/    manuscripts/
.git/        codebase/    literature/  talks/
```

- Branching in Git.
  - Branches in a project:



– For this course, you need to have two branches: `develop` for your own development and `master` for releases (homework submission). Note `master` is the default branch when you initialize the project; create and switch to `develop` branch immediately after project initialization.



– Commonly used commands:

`git branch branchname` creates a branch.

`git branch` shows all project branches.

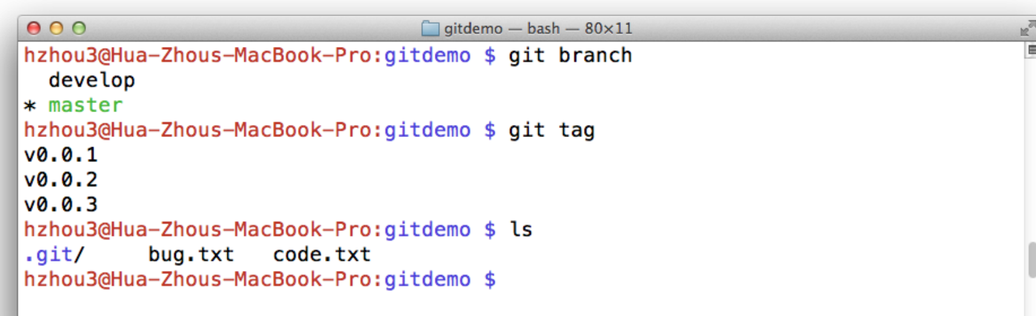
`git checkout branchname` switches to a branch.

`git tag` shows tags (major landmarks).

`git tag tagname` creates a tag.

– Let's look at a typical branching and merging workflow.

  * Now there is a bug in v0.0.3 ...



☞ How to organize version number of your software? Read blog "R Package

Versioning" by Yihui Xie
`http://yihui.name/en/2013/06/r-package-versioning/`

```
gitdemo — bash — 80×31
hzhou3@Hua-Zhous-MacBook-Pro:gitdemo $ git checkout develop
Switched to branch 'develop'
Your branch is up-to-date with 'origin/develop'.
hzhou3@Hua-Zhous-MacBook-Pro:gitdemo $ git branch
* develop
  master
hzhou3@Hua-Zhous-MacBook-Pro:gitdemo $ ls
.git/     code.txt
hzhou3@Hua-Zhous-MacBook-Pro:gitdemo $ git pull origin master
From github.ncsu.edu:hzhou3/gitdemo
 * branch            master     -> FETCH_HEAD
Updating 44dd1d1..da047cf
Fast-forward
 bug.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 bug.txt
hzhou3@Hua-Zhous-MacBook-Pro:gitdemo $ ls
.git/     bug.txt   code.txt
hzhou3@Hua-Zhous-MacBook-Pro:gitdemo $ git rm bug.txt
rm 'bug.txt'
hzhou3@Hua-Zhous-MacBook-Pro:gitdemo $ git commit -m "debug"
[develop 1085b4c] debug
 1 file changed, 1 deletion(-)
 delete mode 100644 bug.txt
hzhou3@Hua-Zhous-MacBook-Pro:gitdemo $ git push
Counting objects: 1, done.
Writing objects: 100% (1/1), 180 bytes | 0 bytes/s, done.
Total 1 (delta 0), reused 0 (delta 0)
To git@github.ncsu.edu:hzhou3/gitdemo.git
   44dd1d1..1085b4c  develop -> develop
hzhou3@Hua-Zhous-MacBook-Pro:gitdemo $
```

Now 'debug' in develop branch is ahead of master branch.

* Merge bug fix to the `master` branch.

```
hzhou3@Hua-Zhous-MacBook-Pro:gitdemo $ git checkout master
Switched to branch 'master'
Your branch is up-to-date with 'origin/master'.
hzhou3@Hua-Zhous-MacBook-Pro:gitdemo $ git branch
  develop
* master
hzhou3@Hua-Zhous-MacBook-Pro:gitdemo $ git pull origin develop
From github.ncsu.edu:hzhou3/gitdemo
 * branch            develop    -> FETCH_HEAD
Updating da047cf..1085b4c
Fast-forward
 bug.txt | 1 -
 1 file changed, 1 deletion(-)
 delete mode 100644 bug.txt
hzhou3@Hua-Zhous-MacBook-Pro:gitdemo $ ls
.git/     code.txt
hzhou3@Hua-Zhous-MacBook-Pro:gitdemo $ git status .
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)
nothing to commit, working directory clean
hzhou3@Hua-Zhous-MacBook-Pro:gitdemo $ git push origin master
Total 0 (delta 0), reused 0 (delta 0)
To git@github.ncsu.edu:hzhou3/gitdemo.git
   da047cf..1085b4c  master -> master
hzhou3@Hua-Zhous-MacBook-Pro:gitdemo $
```

* Tag a new release v0.0.4.

```
hzhou3@Hua-Zhous-MacBook-Pro:gitdemo $ git tag v0.0.4
hzhou3@Hua-Zhous-MacBook-Pro:gitdemo $ git tag
v0.0.1
v0.0.2
v0.0.3
v0.0.4
hzhou3@Hua-Zhous-MacBook-Pro:gitdemo $ git show v0.0.4
commit 1085b4c97ed29fc847442bd0640db2b6fed4d0af
Author: Hua Zhou <hua_zhou@ncsu.edu>
Date:   Tue Jan 13 11:24:19 2015 -0500

    debug

diff --git a/bug.txt b/bug.txt
deleted file mode 100644
index 0a1d6ac..0000000
--- a/bug.txt
+++ /dev/null
@@ -1 +0,0 @@
-There is a bug
hzhou3@Hua-Zhous-MacBook-Pro:gitdemo $ git push origin v0.0.4
Total 0 (delta 0), reused 0 (delta 0)
To git@github.ncsu.edu:hzhou3/gitdemo.git
 * [new tag]         v0.0.4 -> v0.0.4
hzhou3@Hua-Zhous-MacBook-Pro:gitdemo $
```

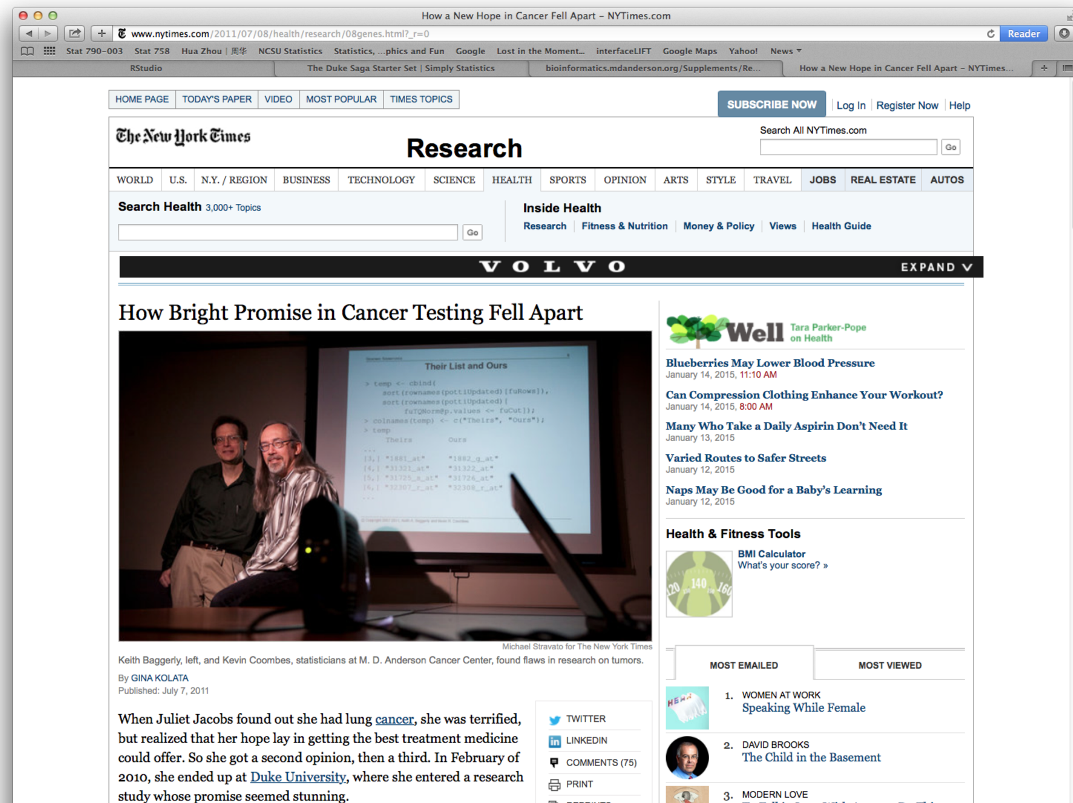
- Further resources:
  - Book *Pro Git*, `http://git-scm.com/book/en/v2`
  - Google
  - Cai Li's slides `http://hua-zhou.github.io/teaching/st790-2015spr/gitslides-CaiLi.pdf` (migrate repositories or folders of a repository, how branching and merging work)
- Some etiquettes of using Git and version control systems in general.
  - Be judicious what to put in repository
    - Not too less: Make sure collaborators or yourself can reproduce everything on other machines.
    - Not too much: No need to put all intermediate files in repository.

    Strictly version control system is for source files only. E.g. only `xxx.tex`, `xxx.bib`, and figure files are necessary to produce a pdf file. Pdf file doesn't need to be version controlled or frequently committed.
  - ☞ "Commit early, commit often and don't spare the horses"
  - Adding an informative message when you commit is *not* optional. Spending one minute now saves hours later for your collaborators and yourself. Read the following sentence to yourself 3 times:
    ☞ "Write every commit message like the next person who reads it is an axe-wielding maniac who knows where you live."
- Acknowledgement: some material in this lecture are taken from Cai Li's group meeting presentation.

Reproducible research (in computational science)

> *An article about computational result is* **advertising**, **not** *scholarship. The actual scholarship is the full software environment, code and data, that produced the result.*
>
> Buckheit and Donoho (1995)
> also see Claerbout and Karrenbach (1992)

- 3 stories of *not* being reproducible.
  - Duke Potti Scandal.


How a New Hope in Cancer Fell Apart – NYTimes.com

HOME PAGE | TODAY'S PAPER | VIDEO | MOST POPULAR | TIMES TOPICS

SUBSCRIBE NOW | Log In | Register Now | Help

The New York Times

Research

WORLD | U.S. | N.Y. / REGION | BUSINESS | TECHNOLOGY | SCIENCE | HEALTH | SPORTS | OPINION | ARTS | STYLE | TRAVEL | JOBS | REAL ESTATE | AUTOS

Search Health 3,000+ Topics

Inside Health

Research | Fitness & Nutrition | Money & Policy | Views | Health Guide

VOLVO EXPAND

How Bright Promise in Cancer Testing Fell Apart

Michael Stravato for The New York Times

Keith Baggerly, left, and Kevin Coombes, statisticians at M. D. Anderson Cancer Center, found flaws in research on tumors.

By GINA KOLATA

Published: July 7, 2011

When Juliet Jacobs found out she had lung cancer, she was terrified, but realized that her hope lay in getting the best treatment medicine could offer. So she got a second opinion, then a third. In February of 2010, she ended up at Duke University, where she entered a research study whose promise seemed stunning.

TWITTER | LINKEDIN | COMMENTS (75) | PRINT | REPRINTS

Well. Tara Parker-Pope on Health

Blueberries May Lower Blood Pressure
January 14, 2015, 11:10 AM

Can Compression Clothing Enhance Your Workout?
January 14, 2015, 8:00 AM

Many Who Take a Daily Aspirin Don't Need It
January 13, 2015

Varied Routes to Safer Streets
January 12, 2015

Naps May Be Good for a Baby's Learning
January 12, 2015

Health & Fitness Tools

BMI Calculator
What's your score? »

MOST EMAILED | MOST VIEWED

1. WOMEN AT WORK
Speaking While Female

2. DAVID BROOKS
The Child in the Basement

3. MODERN LOVE
To Fall in Love With Anyone, Do This

Potti et al. (2006) Genomic signatures to guide the use of chemotherapeutics, *Nature Medicine*, 12(11):1294–1300.

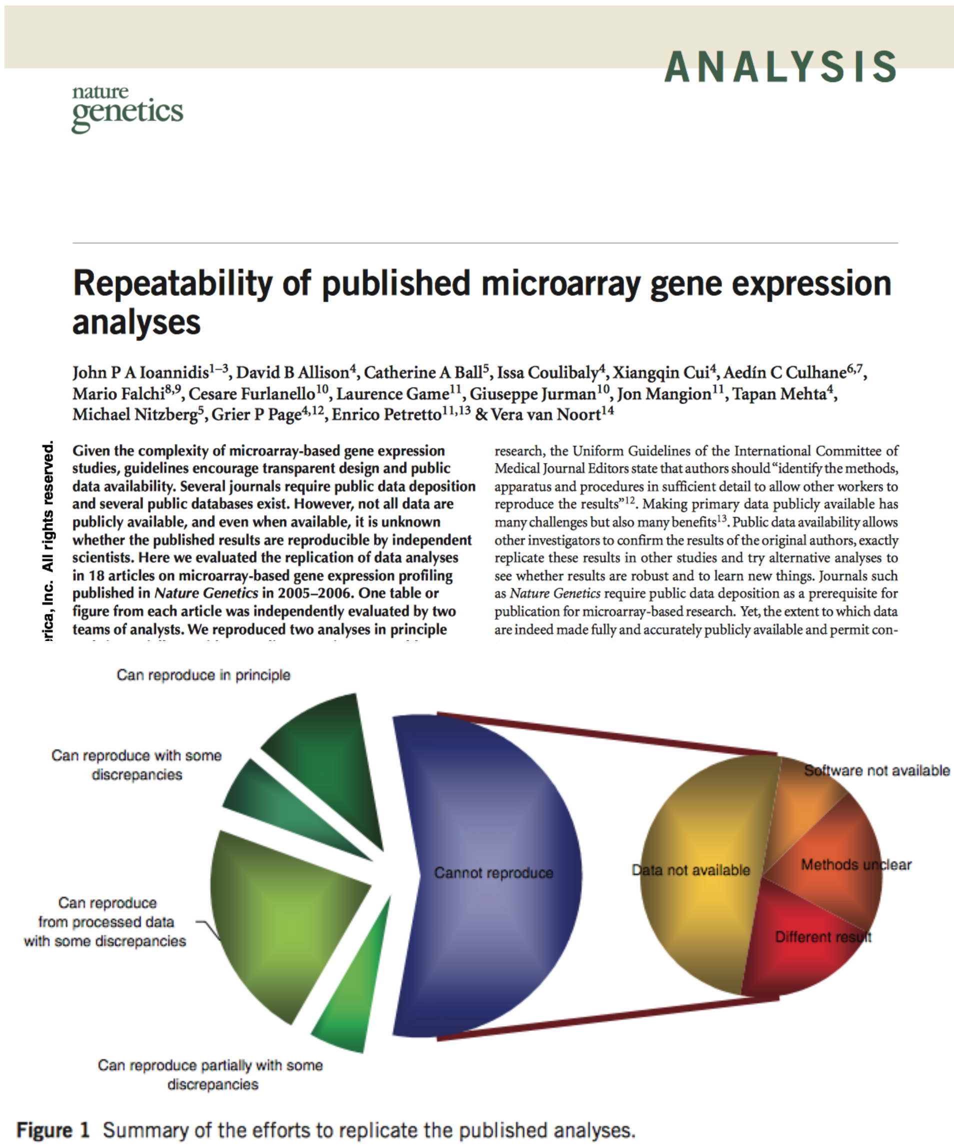
Baggerly and Coombes (2009) Deriving chemosensitivity from cell lines: Forensic bioinformatics and reproducible research in high-throughput biology, *Ann. Appl. Stat.*, 3(4):1309–1334. `http://projecteuclid.org/euclid.aoas/1267453942`

More information is available at

http://en.wikipedia.org/wiki/Anil_Potti

http://simplystatistics.org/2012/02/27/the-duke-saga-starter-set/

– Nature Genetics (2013 Impact Factor: 29.648). 20 articles about microarray profiling published in *Nature Genetics* between Jan 2005 and Dec 2006.



ANALYSIS

nature genetics

Repeatability of published microarray gene expression analyses

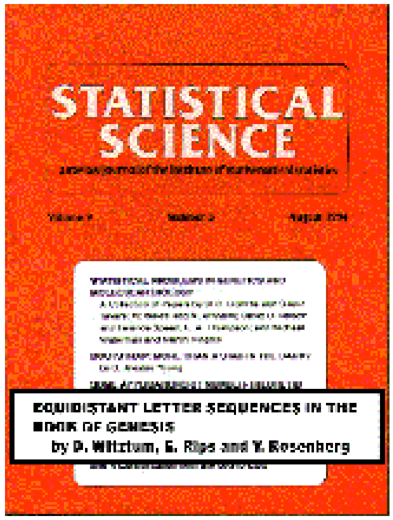
John P A Ioannidis[1–3], David B Allison[4], Catherine A Ball[5], Issa Coulibaly[4], Xiangqin Cui[4], Aedín C Culhane[6,7], Mario Falchi[8,9], Cesare Furlanello[10], Laurence Game[11], Giuseppe Jurman[10], Jon Mangion[11], Tapan Mehta[4], Michael Nitzberg[5], Grier P Page[4,12], Enrico Petretto[11,13] & Vera van Noort[14]

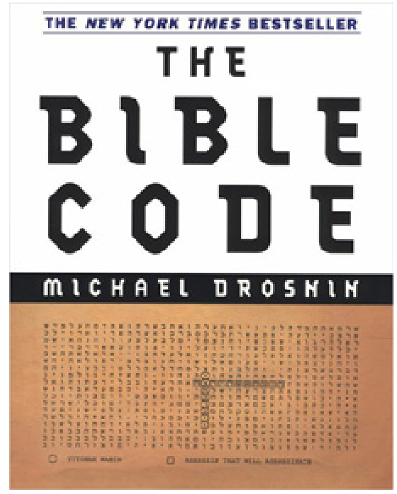
Given the complexity of microarray-based gene expression studies, guidelines encourage transparent design and public data availability. Several journals require public data deposition and several public databases exist. However, not all data are publicly available, and even when available, it is unknown whether the published results are reproducible by independent scientists. Here we evaluated the replication of data analyses in 18 articles on microarray-based gene expression profiling published in *Nature Genetics* in 2005–2006. One table or figure from each article was independently evaluated by two teams of analysts. We reproduced two analyses in principle

research, the Uniform Guidelines of the International Committee of Medical Journal Editors state that authors should "identify the methods, apparatus and procedures in sufficient detail to allow other workers to reproduce the results"[12]. Making primary data publicly available has many challenges but also many benefits[13]. Public data availability allows other investigators to confirm the results of the original authors, exactly replicate these results in other studies and try alternative analyses to see whether results are robust and to learn new things. Journals such as *Nature Genetics* require public data deposition as a prerequisite for publication for microarray-based research. Yet, the extent to which data are indeed made fully and accurately publicly available and permit con-

Figure 1 Summary of the efforts to replicate the published analyses.

– Bible code.





Witztum et al. (1994) Equidistant letter sequences in the book of genesis. *Statist. Sci.*, 9(3):429438. `http://projecteuclid.org/euclid.ss/1177010393`

McKay et al. (1999) Solving the Bible code puzzle, *Statist. Sci.*, 14(2):150–173. `http://cs.anu.edu.au/~bdm/dilugim/StatSci/`

- Why reproducible research?
  - Replicability has been a foundation of science. It helps accumulate scientific knowledge.
  - Better work habit boosts quality of research.
  - Greater research impact.
  - Better teamwork. For you, it probably means better communication with your advisor (Buckheit and Donoho, 1995).
  - ...
- Readings.
  - Buckheit and Donoho (1995) Wavelab and reproducible research, in *Wavelets and Statistics*, volume 103 of *Lecture Notes in Statistics*, page 55–81. Springer Newt York. `http://statweb.stanford.edu/~donoho/Reports/1995/wavelab.pdf`

    Donoho (2010) An invitation to reproducible computational research, *Biostatistics*, 11(3):385-388.
  - Peng (2009) Reproducible research and biostatistics, *Biostatistics*, 10(3):405–408.

    Peng (2011) Reproducible research in computational science, *Science*, 334(6060):1226–1227.

Roger Peng's blogs *Treading a New Path for Reproducible Research*.
http://simplystatistics.org/2013/08/21/treading-a-new-path-for-reproducible-res
http://simplystatistics.org/2013/08/28/evidence-based-data-analysis-treading-a-
http://simplystatistics.org/2013/09/05/implementing-evidence-based-data-analysi

- *Reproducible research with R and RStudio* by Christopher Gandrud. It covers many useful tools: R, RStudio, LaTeX, Markdown, *knitr*, Github, Linux shell, ...

  ☞ This book is nicely reproducible. Git clone the source from https://github.com/christophergandrud/Rep-Res-Book and you should be able to compile into a pdf.

- *Reproducibility in Science* at
  http://ropensci.github.io/reproducibility-guide/

• How to be reproducible in statistics?

> *When we publish articles containing figures which were generated by computer, we also publish the complete software environment which generates the figures.*
>
> Buckheit and Donoho (1995)

- For theoretical results, include all detailed proofs.
- For data analysis or simulation study
  * Describe your computational results with painstaking details.
  * Put your code on your website or in an online supplement (required by many journals, e.g., *Biostatistics*, *JCGS*, ...) that allow replication of entire analysis or simulation study. A good example:
    http://stanford.edu/~boyd/papers/admm_distr_stats.html
  * Create a dynamic version of your simulation study/data analysis.

• What can we do now? At least make your homework reproducible!

- Document everything!
- Everything is a text file (.csv, .tex, .bib, .Rmd, .R, ...) They aid future proof and are subject to version control.

  ☞ Word/Excel are not text files.
- All files should be human readable. Abundant comments and adopt a good style.
- Tie your files together.

- Use a dynamic document generation tool (weaving/knitting text, code, and output together) for documentation. For example
  `http://hua-zhou.github.io/teaching/st758-2014fall/hw01sol.html`
  `http://hua-zhou.github.io/teaching/st758-2014fall/hw02sol.html`
  ...
  `http://hua-zhou.github.io/teaching/st758-2014fall/hw07sol.html`
  `http://hua-zhou.github.io/teaching/st758-2014fall/hw08sol.html`
- Use a version control system proactively.
- Print `sessionInfo()` in R.

☞ For your homework, submit (put in the `master` branch) a final pdf report and all files and instructions necessary to reproduce all results.

- Tools for dynamic document/report generation.
  - R: RMarkdown, knitr, Sweave.
  - Matlab: automatic report generator.
  - Python: IPython, Pweave.
  - Julia: IJulia.

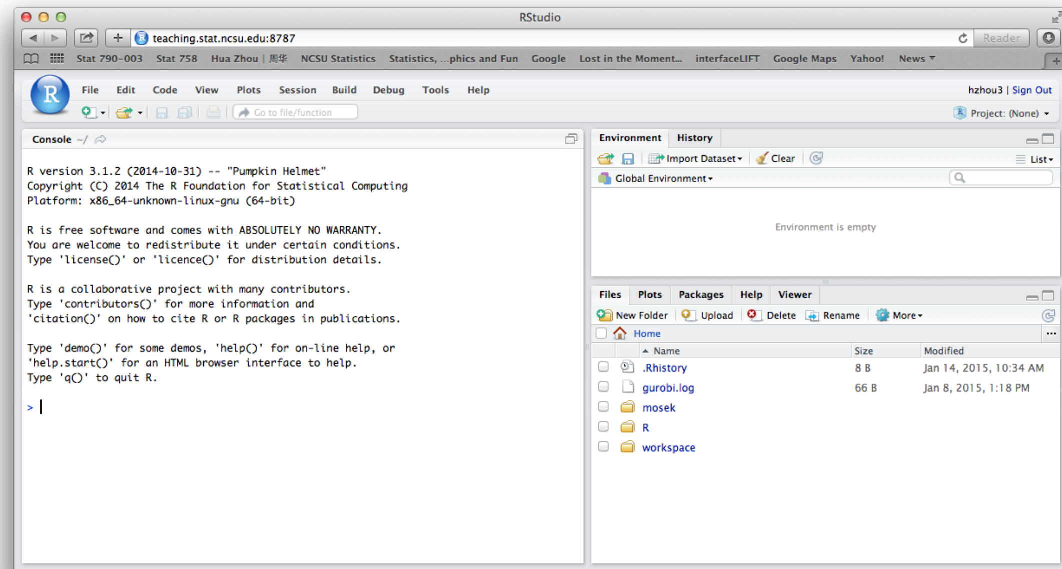
  We will briefly talk about these features when discussing specific languages.

4 Lecture 4, Jan 26

Announcements

- Helpful tutorial about Git branching
  `http://pcottle.github.io/learnGitBranching/`
  shared by Bo Ning.

- Want to use R Studio on teaching server?



Access via `http://teaching.stat.ncsu.edu:8787`. However you need to change password on command line (`passwd`).

Last Time

- Version control using Git (cont'd).

- Reproducible research.

Today

- This week: languages (R, Matlab, Julia)

Computer Languages

> **工欲善其事，必先利其器**
>
> *To do a good job, an artisan needs the best tools.*
>
> *The Analects* by Confucius (about 500 BC)

- What features are we looking for in a language?
  - Efficiency (in both run time and memory) for handling big data.
  - IDE support (debugging, profiling).
  - Open source.
  - Legacy code.
  - Tools for generating dynamic report.
  - Adaptivity to hardware evolution (parallel and distributed computing).
- Types of languages
  1. Compiled languages: C/C++, FORTRAN, ...
     - Directly compiled to machine code that is executed by CPU
     - Pros: fast, memory efficient
     - Cons: longer development time, hard to debug
  2. Interpreted language: R, MATLAB, SAS IML, ...
     - Interpreted by interpreter
     - Pros: fast prototyping
     - Cons: excruciatingly slow for loops
  3. Mixed languages: Julia, Python, JAVA, Matlab (JIT), R (JIT), ...
     - Compiled to bytecode and then interpreted by virtual machine
     - Pros: relatively short development time, cross-platform, good at data preprocessing and manipulation, rich libraries and modules
     - Cons: not as fast as compiled language
  4. Script languages: shell scripts, Perl, ...
     - Extremely useful for data preprocessing and manipulation

- Messages
  - To be versatile in the big data era, be proficient in at least one language in each category.
  - To improve efficiency of interpreted languages such as R or MATLAB, avoid loops as much as possible. Aka, vectorize code
    "The only loop you are allowed to have is that for an iterative algorithm."
  - For some tasks where looping is necessary, consider coding in C or FORTRAN. It is convenient to incorporate compiled code into R or MATLAB. But do this **only after profiling!**
    Success stories: `glmnet` and `lars` packages in R are based on FORTRAN.
  - When coding using C, C++, FORTRAN, make use of libraries for numerical linear algebra: BLAS, LAPACK, ATLAS, ...

☞ Julia seems to combine the strengths of all these languages. That is to achieve efficiency without vectorizing code.

5 Lecture 5, Jan 28

Announcements

- HW2 (NNMF, GPU computing) posted. Due Feb 11.

Last Time

- Computer languages.
- Productivity tools (Rcpp, Boost, Armadillo, R markdown) of R (Josh Day).

Today

- Matlab and Julia.

Computer languages (cont'd)

> *"As some of you may know, I have had a (rather late) mid-life crisis and run off with another language called Julia.* ***http://julialang.org*** *"*
>
> Doug Bates (on the `knitr` Google Group)

- Language features of R, Matlab, and Julia

Features	R	Matlab	julia
Open source	☺	☹	☺
IDE	R Studio ☺☺	☺☺☺	☹
Dynamic document	☺☺☺(RMarkdown)	☺☺☺	☺☺☺(IJulia)
Multi-threading	`parallel` pkg	☺	☺
JIT	`compiler` pkg	☺	☺
Call C/Fortran	wrapper	wrapper	no glue code
Call shared library	wrapper	wrapper	no glue code
Typing	☹	☺☺	☺☺☺
Pass by reference	☹	☹	☺☺☺
Linear algebra	☹	MKL, Arpack	OpenBLAS, Eigpack
Distributed computing	☹	☺	☺☺☺
Sparse linear algebra	☹ (`Matrix` package)	☺☺☺	☺☺☺
Documentation	☹	☺☺☺	☺☺

- Benchmark code `R-benchmark-25.R` from `http://r.research.att.com/benchmarks/R-benchmark-25.R` covers many commonly used numerical operations used in statistics. We ported to MATLAB and Julia and report the run times (averaged over 5 runs) here.

Machine specs: Intel i7 @ 2.6GHz (4 physical cores, 8 threads), 16G RAM, Mac OS 10.9.5.

Test	R 3.1.1	MATLAB R2014a	JULIA 0.3.5
Matrix creation, trans, deformation (2500×2500)	0.80	**0.17**	0.16
Power of matrix (2500×2500, $A.^{1000}$)	0.22	**0.11**	0.23
Quick sort ($n = 7 \times 10^6$)	0.65	**0.24**	0.64
Cross product (2800×2800, $A^T A$)	9.95	**0.35**	0.38
LS solution ($n = p = 2000$)	1.21	**0.07**	0.10
FFT ($n = 2,400,000$)	0.34	**0.04**	0.14
Eigen-decomposition (600×600)	0.78	**0.31**	0.56
Determinant (2500×2500)	3.52	**0.18**	0.23
Cholesky (3000×3000)	4.03	**0.15**	0.23
Matrix inverse (1600×1600)	3.05	**0.16**	0.22
Fibonacci (vector)	0.28	**0.17**	0.66
Hilbert (matrix)	0.22	**0.07**	0.18
GCD (recursion)	0.47	**0.14**	0.20
Toeplitz matrix (loops)	0.34	**0.0014**	0.03
Escoufiers (mixed)	0.38	0.40	**0.17**

- A slightly more complicated (or realistic) example taken from Doug Bates's slides `http://www.stat.wisc.edu/~bates/JuliaForRProgrammers.pdf`. The task is to use Gibbs sampler to sample from bivariate density

$$f(x, y) = kx^2 \exp(-xy^2 - y^2 + 2y - 4x), x > 0,$$

using the conditional distributions

$$\begin{aligned} X|Y &\sim \Gamma\left(3, \frac{1}{y^2 + 4}\right) \\ Y|X &\sim \mathcal{N}\left(\frac{1}{1+x}, \frac{1}{2(1+x)}\right). \end{aligned}$$

Let's sample 10,000 points from this distribution with a thinning of 500.

  - How long does R take? `http://hua-zhou.github.io/teaching/st790-2015spr/gibbs_r.html`
  - How long does Julia take? `http://hua-zhou.github.io/teaching/st790-2015spr/gibbs_julia.html`

- With similar coding efforts, Julia offers ~ 100 fold speed-up! Somehow JIT in R didn't kick in. (Neither does Matlab, which took about 20 seconds.)
- Julia offers the capability of strong typing of variables. This facilitates the optimization by compiler.
- With little efforts, we can do parallel and distributed computing using Julia.

☞ Benchmark of the same example in other languages including Rcpp is available in the blogs by Darren Wilkinson (`http://bit.ly/IWhJ52`) and Dirk Eddelbuettel's (`http://dirk.eddelbuettel.com/blog/2011/07/14/`).

Julia

- IDE in Julia.
  - Juno (`http://junolab.org`) is the currently recommended IDE for Julia. It has limited capabilities (syntax highlighting, tab completion, executing lines from editor, ...) compared to R Studio or Matlab IDE.
  - No easy-to-use debugging tool yet (set breakpoint, inspect variables at breakpoint, break at error, ...) ☹
  - Profiling. The language itself provides many very useful profiling tools.
    `http://julia.readthedocs.org/en/latest/stdlib/profile`
    `@profile` macro shows line-by-line analysis how many times each line is sampled by the profiler.

```
julia> Profile.clear()

julia> @profile Bhat, Σhat = varcomp(Y, X, V);

julia> Profile.print(format=:flat)
 Count File                              Function                                     Line
   130 ...ces/julia/lib/julia/sys.dylib Array                                          -1
     1 ...ces/julia/lib/julia/sys.dylib map                                            -1
   130 ...ces/julia/lib/julia/sys.dylib print_to_string                                -1
     3 ...org/vcmm/codebase/julia/vc.jl kronaxpy!                                     312
     5 ...org/vcmm/codebase/julia/vc.jl kronaxpy!                                     314
     3 ...org/vcmm/codebase/julia/vc.jl kronreduction!                                339
    55 ...org/vcmm/codebase/julia/vc.jl varcomp                                        78
     1 ...org/vcmm/codebase/julia/vc.jl varcomp                                        80
    35 ...org/vcmm/codebase/julia/vc.jl varcomp                                        82
    54 ...org/vcmm/codebase/julia/vc.jl varcomp                                        83
    25 ...org/vcmm/codebase/julia/vc.jl varcomp                                        88
    13 ...org/vcmm/codebase/julia/vc.jl varcomp                                        93
   141 ...org/vcmm/codebase/julia/vc.jl varcomp                                       100
     3 ...org/vcmm/codebase/julia/vc.jl varcomp                                       105
    35 ...org/vcmm/codebase/julia/vc.jl varcomp                                       108
     8 ...org/vcmm/codebase/julia/vc.jl varcomp                                       114
   542 ...org/vcmm/codebase/julia/vc.jl varcomp                                       117
   519 ...org/vcmm/codebase/julia/vc.jl varcomp                                       118
   168 ...org/vcmm/codebase/julia/vc.jl varcomp                                       119
     1 ...org/vcmm/codebase/julia/vc.jl varcomp                                       120
     2 ...org/vcmm/codebase/julia/vc.jl varcomp                                       121
     2 ...org/vcmm/codebase/julia/vc.jl varcomp                                       123
     2 ...org/vcmm/codebase/julia/vc.jl varcomp                                       127
   265 ...org/vcmm/codebase/julia/vc.jl varcomp                                       128
   133 ...org/vcmm/codebase/julia/vc.jl varcomp                                       130
  2044 REPL.jl                           eval_user_input                               53
     1 abstractarray.jl                  cat                                          616
   196 array.jl                          fill!                                        151
    31 linalg/blas.jl                    gemm!                                        527
   376 linalg/blas.jl                    trmm!                                        813
     3 linalg/blas.jl                    trmv!                                        404
     2 linalg/dense.jl                   \                                            417
```

`@time` macro displays memory footprint and significant gc (garbage collection) along with run time.

```
julia> @time Bhat, Σhat = varcomp(Y, X, V);
elapsed time: 37.295446594 seconds (5518642216 bytes allocated, 2.24% gc time)

julia>
```

Finer analysis of line-by-line memory allocation is also available.
`http://docs.julialang.org/en/release-0.3/manual/profile/#memory-allocation-anal`

- Work flow in Julia.
  - Tim Holy:
    "quickly write the simple version first (which is really pleasant thanks to Julia's

design and the nice library that everyone has been contributing to)” → “run it” → “ugh, too slow” → “profile it” → “fix a few problems in places where it actually matters” → “ah, that’s much nicer!”

  - Stefan Karpinski:

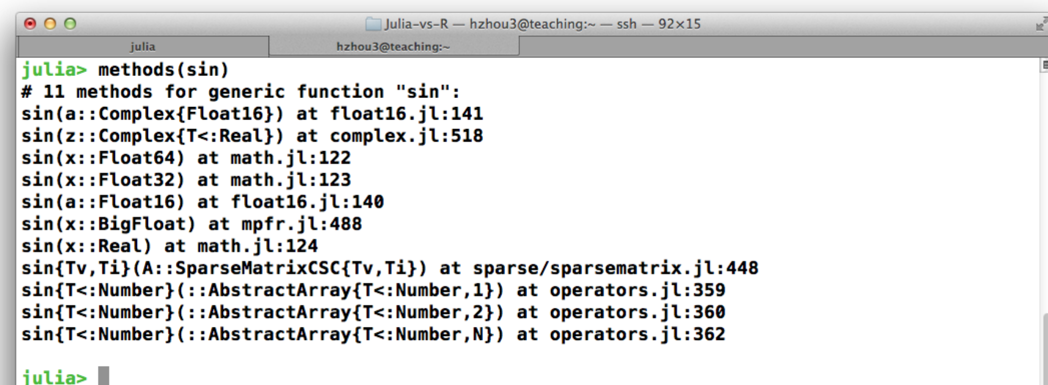
    1. You can write the easy version that works, and
    2. You can usually make it fast with a bit more work.

- Types (data structures) in Julia.

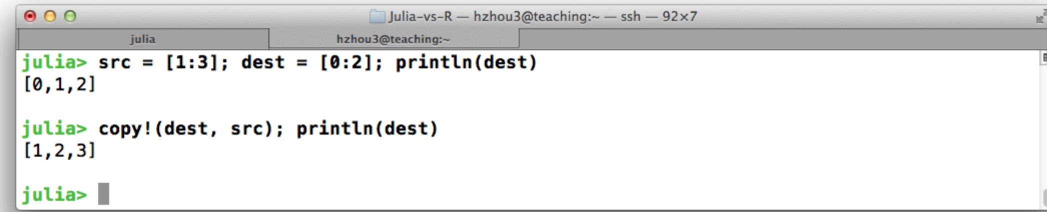
  - Julia provides a very rich collection of static data types, abstract types, and user-defined types.
    http://julia.readthedocs.org/en/latest/manual/integers-and-floating-point-numbe
    http://julia.readthedocs.org/en/latest/manual/types/.

- Functions (methods, algorithms) in Julia.

  - Functions in Julia are really methods. All functions in Julia are generic so any function definition is actually a method definition.
  - Same function (method) names can be applied to different argument *signatures*.
  - Templated methods and data types. Sometimes you want to define algorithms on the abstract type with minor variations for, say, the element type.



  - In Julia, all arguments to functions are passed by reference. A Julia function can modify its arguments. Such *mutating* functions should have names ending in “!”.

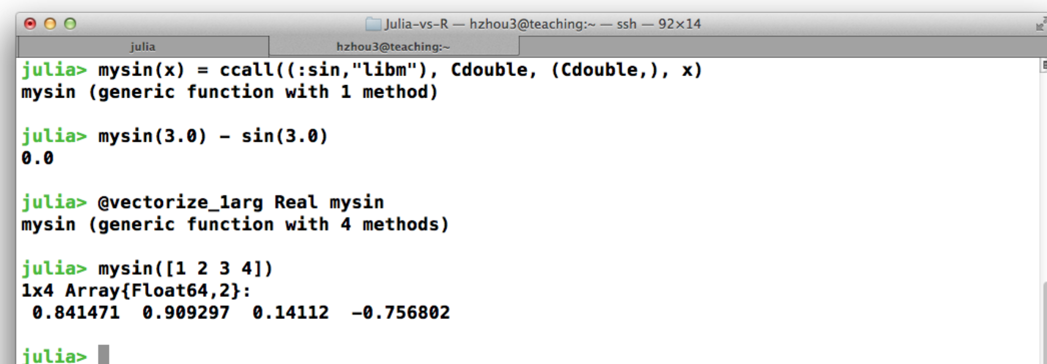


- Call compiled code.
  - In Julia, usually it's *unnecessary* to write C/C++ or Fortran code for performance. Just write loops in Julia and leave the work to its compiler.
  - Still in many situations, we'd like to call functions in some compiled libraries (developed in C or Fortran). Use the `ccall()` function in Julia; no glue code is needed. For example, Mac OS has the math library `libm.dylib`, from which we can call the `sin` function

    ```
    mysin(x) = ccall((:sin,"libm"), Cdouble, (Cdouble,), x)
    ```

    We can vectorize the single argument function `myin` by

    ```
    @vectorize_1arg Real mysin
    ```



  - They must be shared libraries available in the load path, and if necessary a direct path may be specified.
- Call Julia function from other languages like C.
  `http://docs.julialang.org/en/release-0.3/manual/embedding/`

- Documentation.
  - Online help: `? funname`.
  - Online documentation is mostly clear, but seems to lack plenty of examples.
  - I haven't found a good in-source documentation system like `roxygen` for R ☹.
  - Julia provides tab completion, like bash completion.
- Package management in Julia all centers around Github. No manual censorship on CRAN anymore ☺
- Julia summary.

> *"In my opinion Julia provides the best of both worlds and is the technical programming language of the future."*
>
> Doug Bates

6 Lecture 6, Feb 2

Announcements

- HW1 graded. Feedback
  - Solution sketch: `http://hua-zhou.github.io/teaching/st790-2015spr/hw01sol`.
  - `grade_unityID.md` committed to your `master` branch.
  - Don't forget `git tag`: Tagging time will be used as your homework submission time.
  - "Commit early, commit often and don't spare the horses"
  - Reproducibility (source code for reproducing results and instructions). Dynamic document (Rmd, IPython, ...) is worth learning.
- HW2:
  - Think more carefully about algorithmic updates.
  - Use `V0.txt` and `W0.txt` as starting points for timing.
- HW3 (Convex or Not?) posted. Due Mon, Feb 23.

Last Time

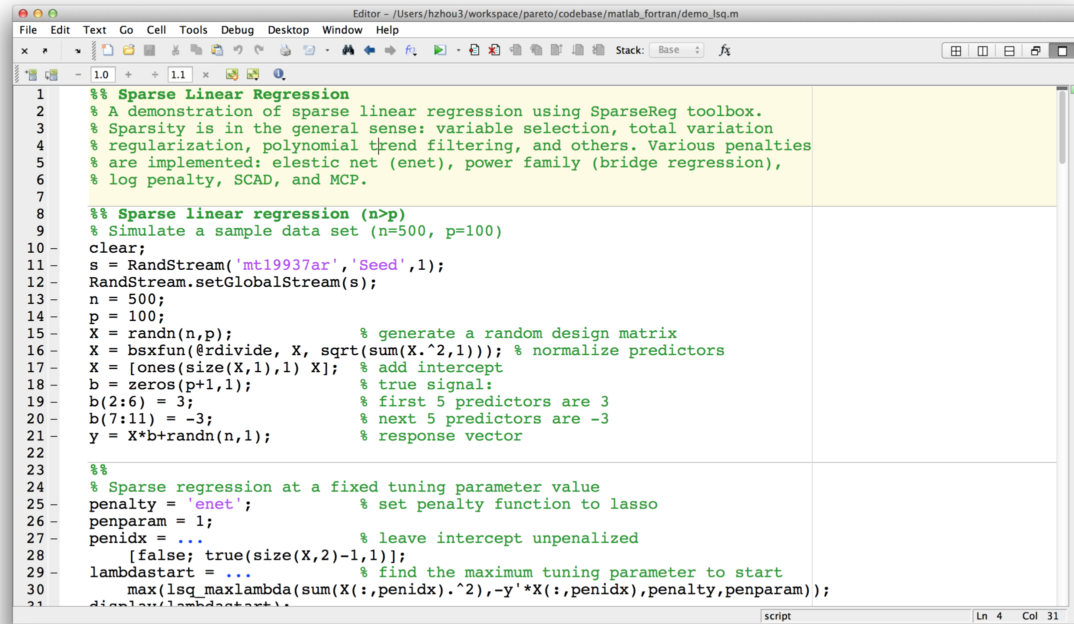
- Julia: a promising language to know about.

Today

- Matlab.
- Parallel computing.

Matlab

- Matlab IDE. A powerful IDE comes with MATLAB. Familiarity with it prevents tons of pain.

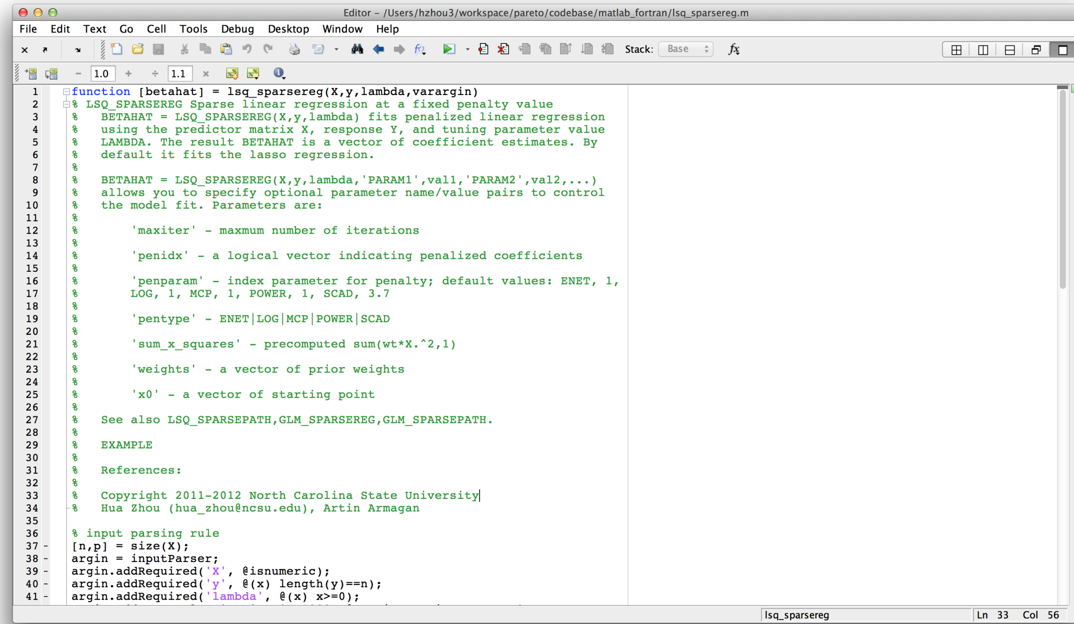


```
%% Sparse Linear Regression
% A demonstration of sparse linear regression using SparseReg toolbox.
% Sparsity is in the general sense: variable selection, total variation
% regularization, polynomial trend filtering, and others. Various penalties
% are implemented: elestic net (enet), power family (bridge regression),
% log penalty, SCAD, and MCP.

%% Sparse linear regression (n>p)
% Simulate a sample data set (n=500, p=100)
clear;
s = RandStream('mt19937ar','Seed',1);
RandStream.setGlobalStream(s);
n = 500;
p = 100;
X = randn(n,p);                 % generate a random design matrix
X = bsxfun(@rdivide, X, sqrt(sum(X.^2,1))); % normalize predictors
X = [ones(size(X,1),1) X];     % add intercept
b = zeros(p+1,1);               % true signal:
b(2:6) = 3;                     % first 5 predictors are 3
b(7:11) = -3;                   % next 5 predictors are -3
y = X*b+randn(n,1);             % response vector

%%
% Sparse regression at a fixed tuning parameter value
penalty = 'enet';               % set penalty function to lasso
penparam = 1;
penidx = ...                    % leave intercept unpenalized
    [false; true(size(X,2)-1,1)];
lambdastart = ...               % find the maximum tuning parameter to start
    max(lsq_maxlambda(sum(X(:,penidx).^2),-y'*X(:,penidx),penalty,penparam));
```

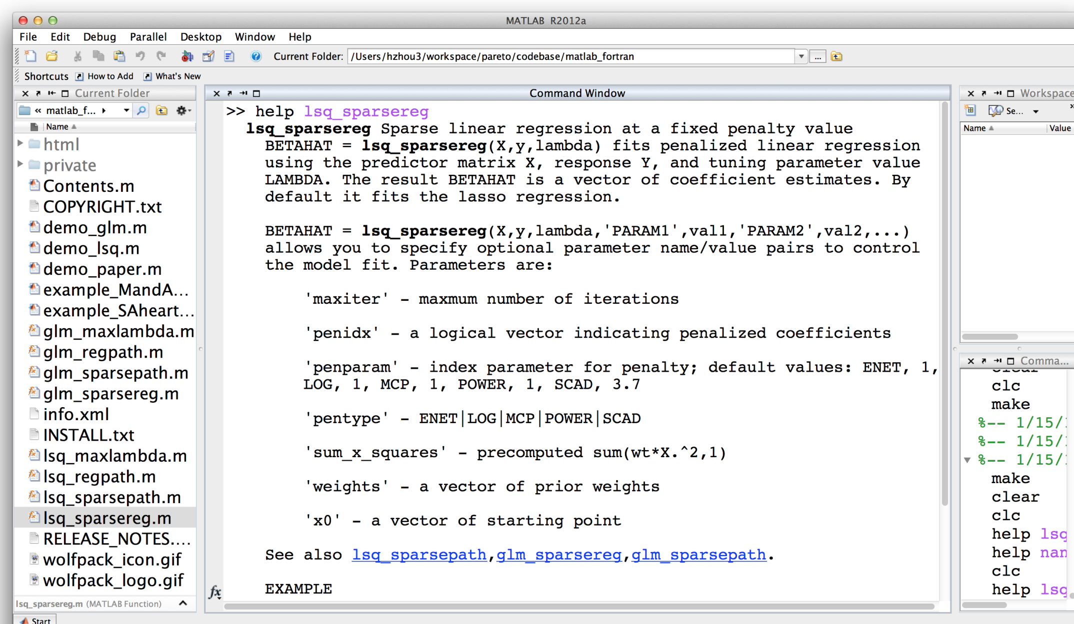
- Essentials: syntax highlighting, code indenting/wrapping/folding, text width (default = 72 characters), ...
- Code cells delimited by `%%`. Cells break script into logical segments and facilitate automatically generating documentation.
- Code analyzer. Are you greened? Check upper-right corner.

• Matlab functions.

- MATLAB development revolves around functions.
- Each function is a separate file: `fun1.m`, `fun2.m`, ...
  ☞ R and Julia can have multiple functions in one file.
- Add help/documentation immediately below the function definition. It facilitates the `help` command and automatically generating documentation.
- If there are more than one functions in a file, only the first one is callable. Others are *local functions*, equivalent of subroutines/subfunctions in other languages.
- Nested function. It has access to the variables in its parent function. Memory saver!
- Function help follows a fixed format: declaration, calling convention, see also, example, copyright, ...



– Help command

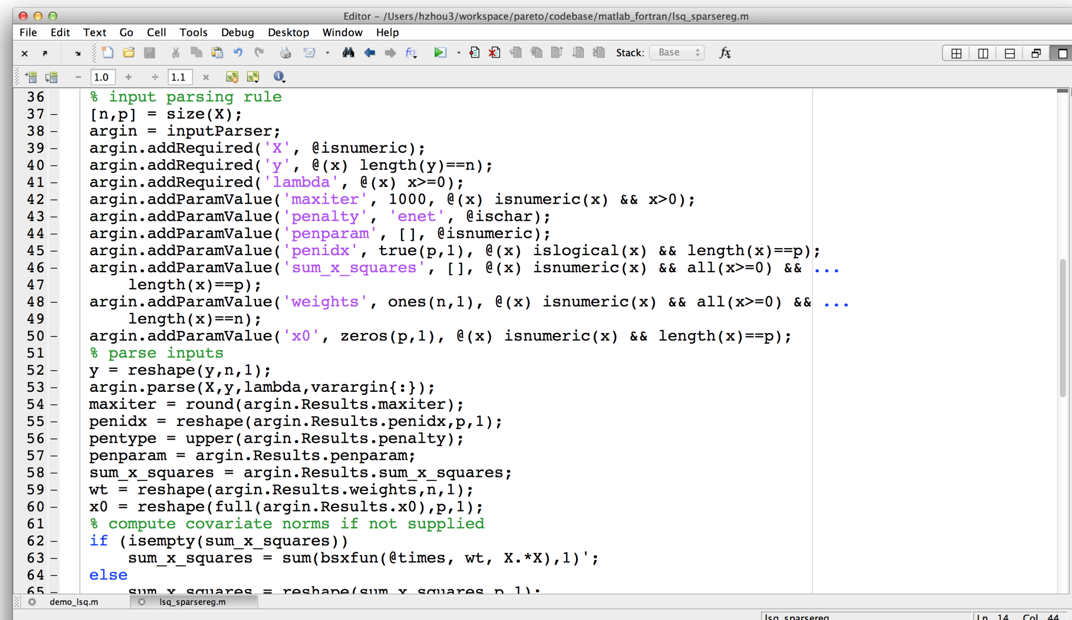


– Support variable number of input/output arguments

```
% linear regression
```

```
b = glmfit(x,y);
% logistic regression
b = glmfit(x,y,'binomial');
% probit regression
b = glmfit(x,y,'binomial','link','probit');
% probit regression with observation weights
b = glmfit(x,y,'binomial','link','probit','weights',wts);
```

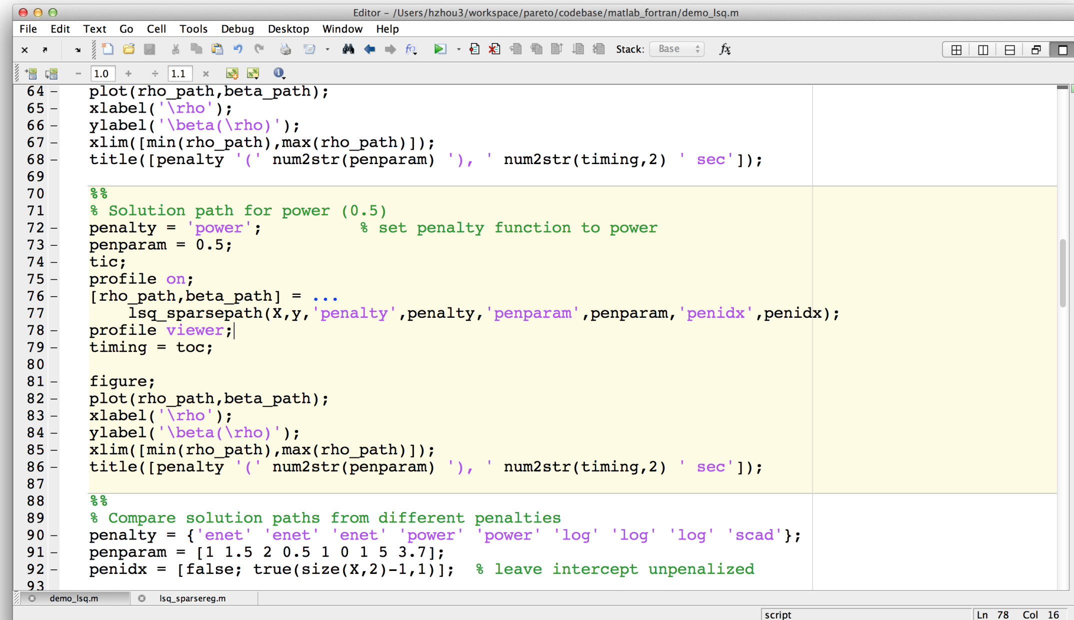
☞ inputParser for parsing name/value pairs



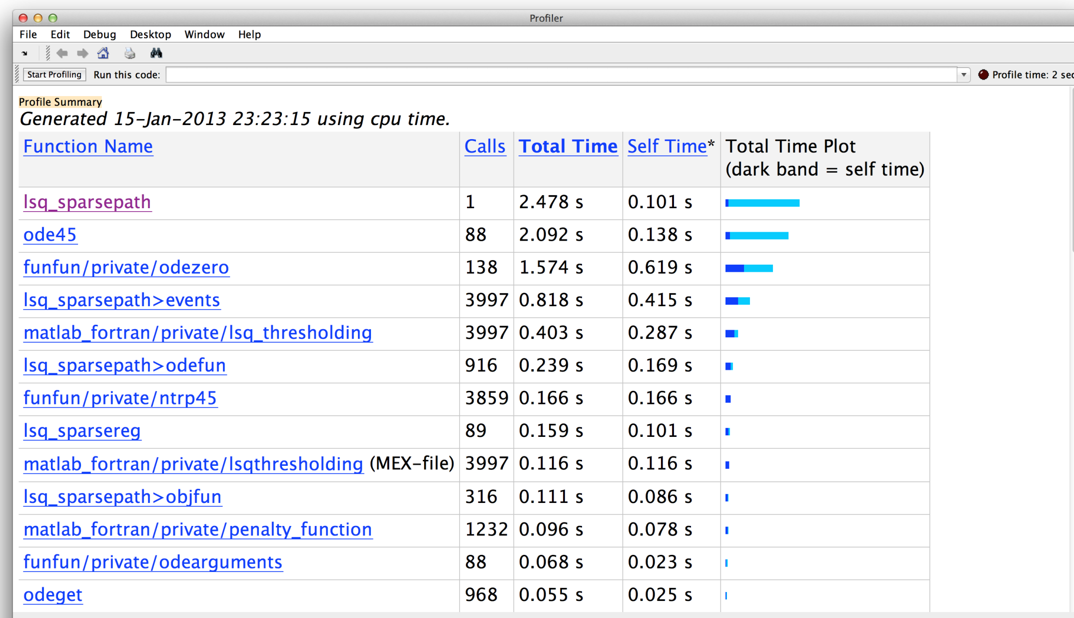
```
% input parsing rule
[n,p] = size(X);
argin = inputParser;
argin.addRequired('X', @isnumeric);
argin.addRequired('y', @(x) length(y)==n);
argin.addRequired('lambda', @(x) x>=0);
argin.addParamValue('maxiter', 1000, @(x) isnumeric(x) && x>0);
argin.addParamValue('penalty', 'enet', @ischar);
argin.addParamValue('penparam', [], @isnumeric);
argin.addParamValue('penidx', true(p,1), @(x) islogical(x) && length(x)==p);
argin.addParamValue('sum_x_squares', [], @(x) isnumeric(x) && all(x>=0) && ...
    length(x)==p);
argin.addParamValue('weights', ones(n,1), @(x) isnumeric(x) && all(x>=0) && ...
    length(x)==n);
argin.addParamValue('x0', zeros(p,1), @(x) isnumeric(x) && length(x)==p);
% parse inputs
y = reshape(y,n,1);
argin.parse(X,y,lambda,varargin{:});
maxiter = round(argin.Results.maxiter);
penidx = reshape(argin.Results.penidx,p,1);
pentype = upper(argin.Results.penalty);
penparam = argin.Results.penparam;
sum_x_squares = argin.Results.sum_x_squares;
wt = reshape(argin.Results.weights,n,1);
x0 = reshape(full(argin.Results.x0),p,1);
% compute covariate norms if not supplied
if (isempty(sum_x_squares))
    sum_x_squares = sum(bsxfun(@times, wt, X.*X),1)';
else
    sum_x_squares = reshape(sum_x_squares,p,1);
```

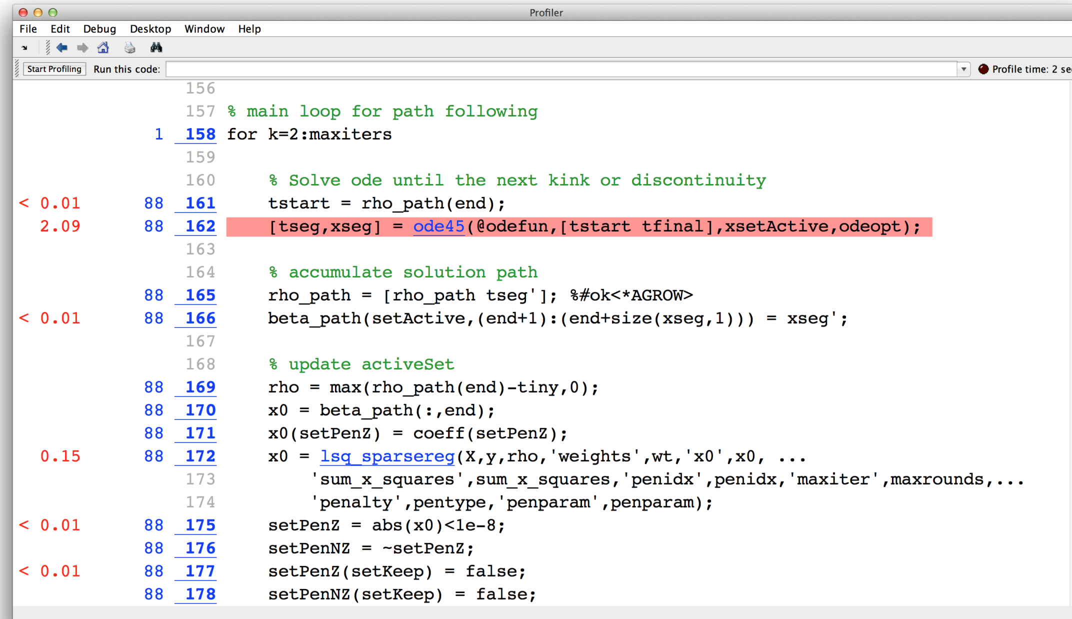
- Debugging in Matlab.
  - Execute code cell-by-cell, line-by-line, ...
  - Breakpoints.
  - Examine intermediate values:
    data tips in editor, command window, workspace browser
  - Error breakpoints.
- Profiling in Matlab.
  - Timing: `tic/toc` (wall time).
  - Profiling: `profile on/viewer`.

– Let's profile the `lsq_sparsepath()` function.

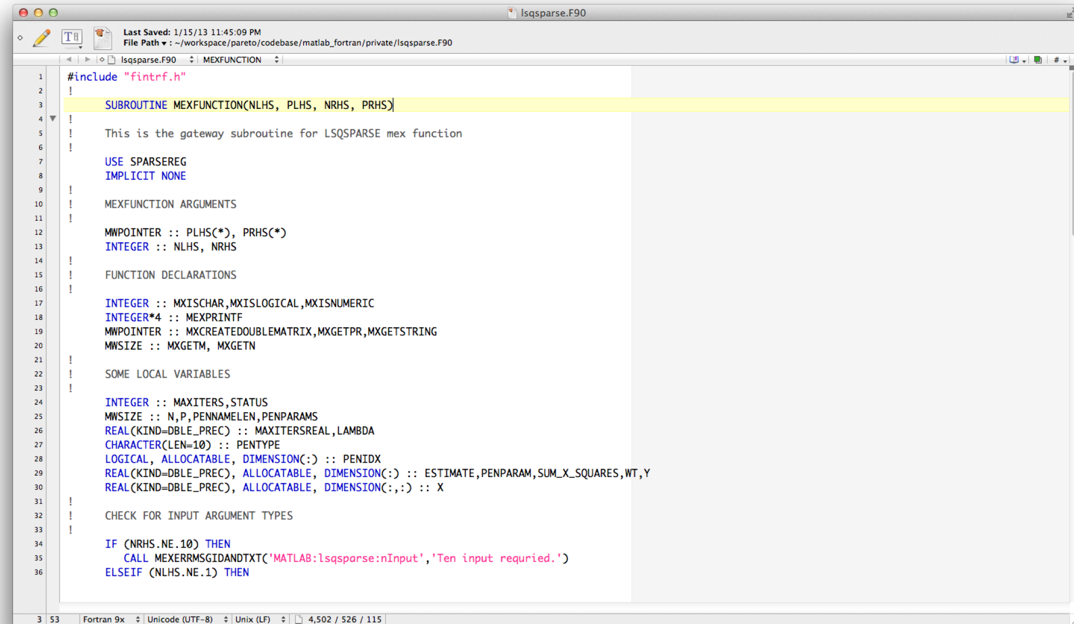


– `profile viewer` produces a summary in html that includes line by line analysis.



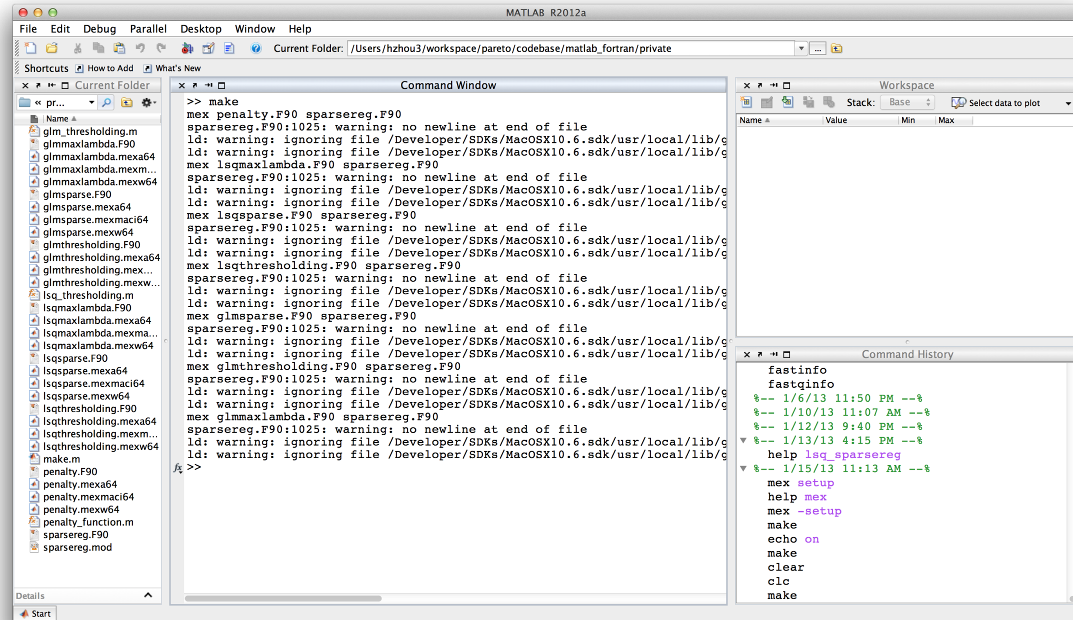


- Call compiled code in MATLAB.
  - Step 0: Are you sure you want to do this? Profile first!
  - Step 1: Check compiler compatibility.
    * What compilers are supported by MATLAB 2014a (Linux)? Check `http://www.mathworks.com/support/compilers/R2014a/index.html`
    * Compilers not supported? Tweak the `mexopts.sh` file
  - Step 2: Write C or FORTRAN code.
    * Develop C or FORTRAN code as usual
    * If you obtain source code from open source projects, internet, book (e.g. *Numerical Recipes*), ..., follow license and give credit
  - Step 3: Write `mex` function wrapper.



  * The name of the **mex** function file is the name of your MATLAB function
  * Purpose: match data types between MATLAB and C/FORTRAN, transfer input/output (pass by value!), ...
  * Format for **mex** function: Google for "matlab mex function"

– Step 4: Use **mex** command to compile source.

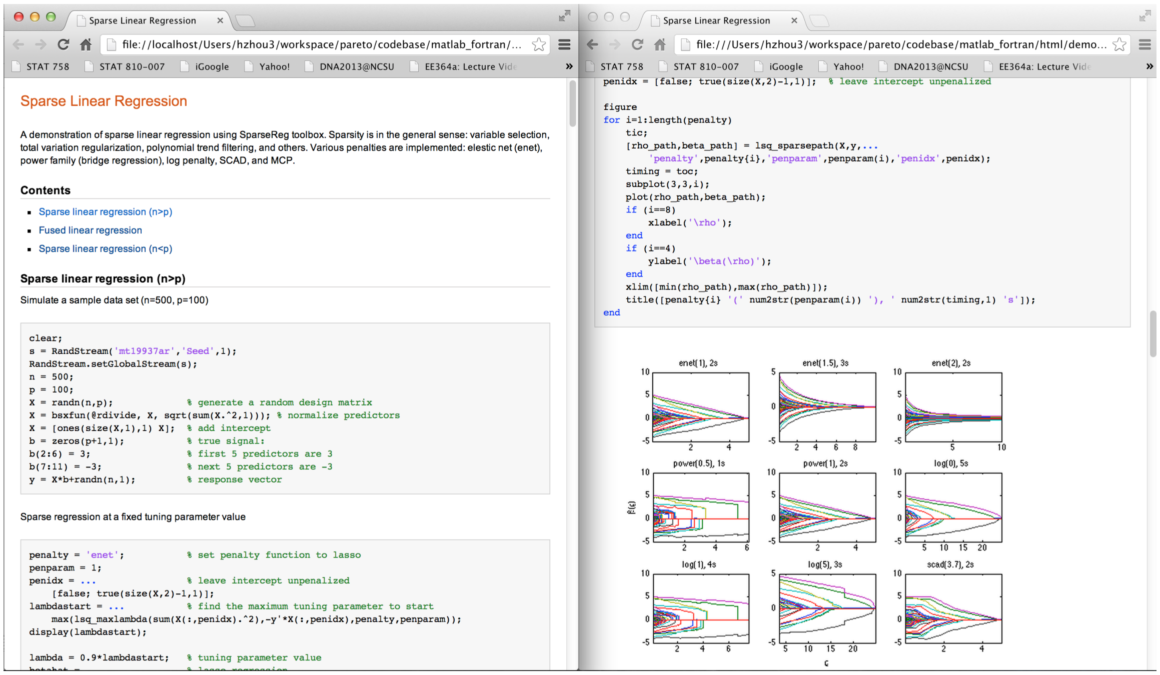


∗ This produces binary code: `funname.mexmaci64` (Mac), `funname.mexw64` (Windows), or `funname.mexa64` (Linux)

∗ These binaries are what you need to run program. Just use as native MATLAB functions

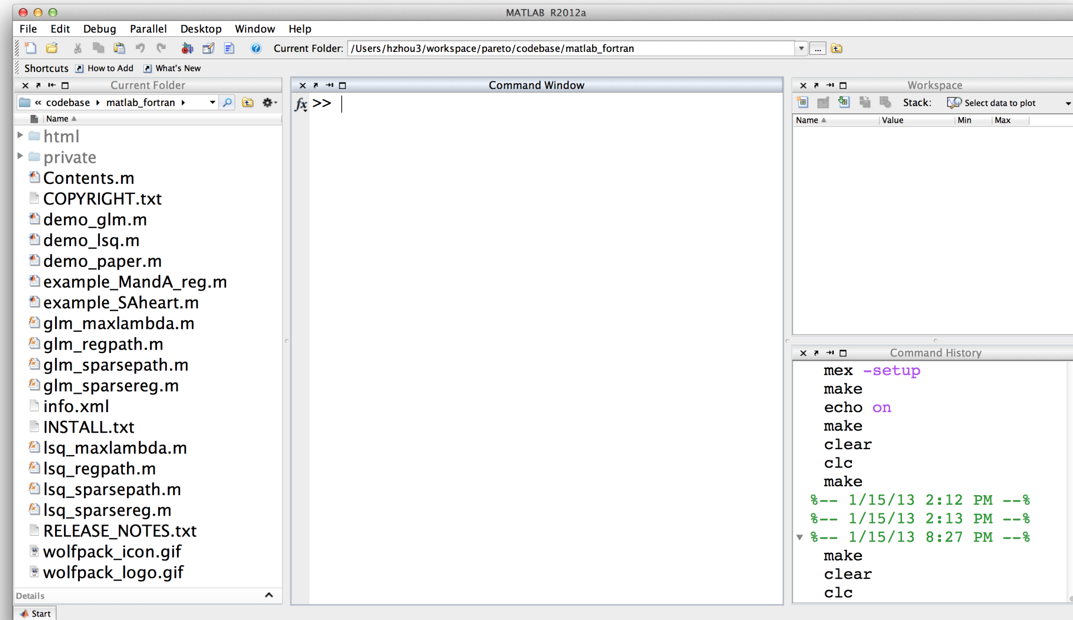
- Toolbox development in MATLAB.
  - Toolbox in MATLAB is equivalent to the packages in R. You can submit to MATLAB CENTRAL (equivalent of CRAN) or simply publish on your github or website.
  - Basic steps to create a toolbox.
    1. Write functions and demo scripts
    2. Debug, test, profile, document
       debug, test, profile, document
       ...
    3. "Publish" your demo scripts as `html` by clicking the `Publish` button. It works just like `knit`.
       `http://www.mathworks.com/help/matlab/matlab_prog/marking-up-matlab-comments-for-publishing.html`
    4. Edit the `info.xml` and `helptoc.xml` files. They help automatically generate the help documentation and put the toolbox to the MATLAB start menu
    5. Write the `COPYRIGHT.txt`, `INSTALL.txt` and `RELEASE_NOTES.txt` documents

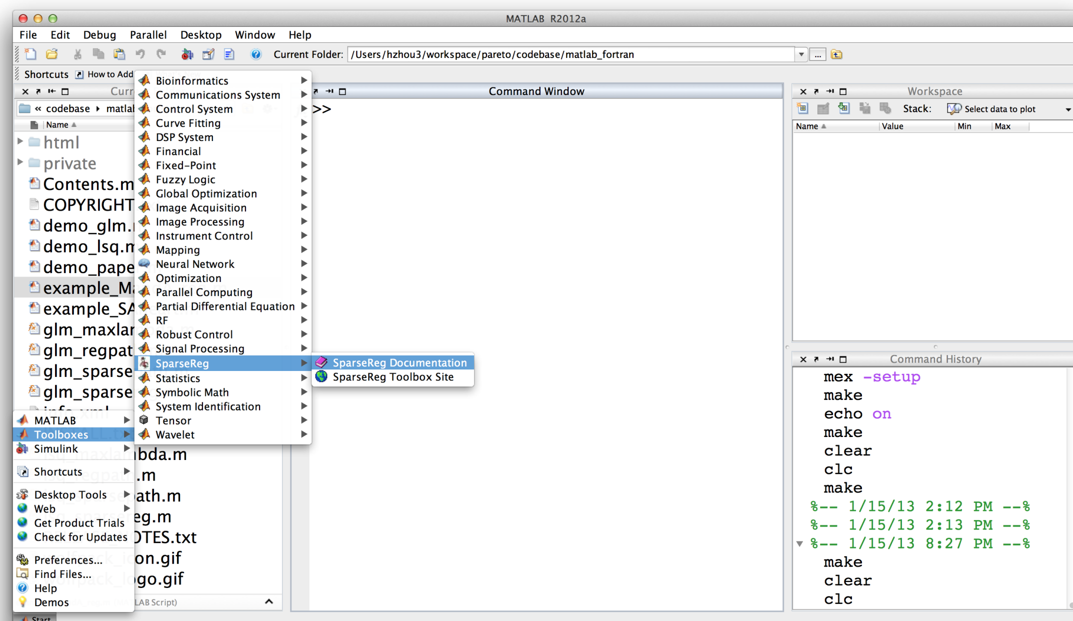
6. Zip the toolbox folder and publish on your website or to MATLAB CENTRAL

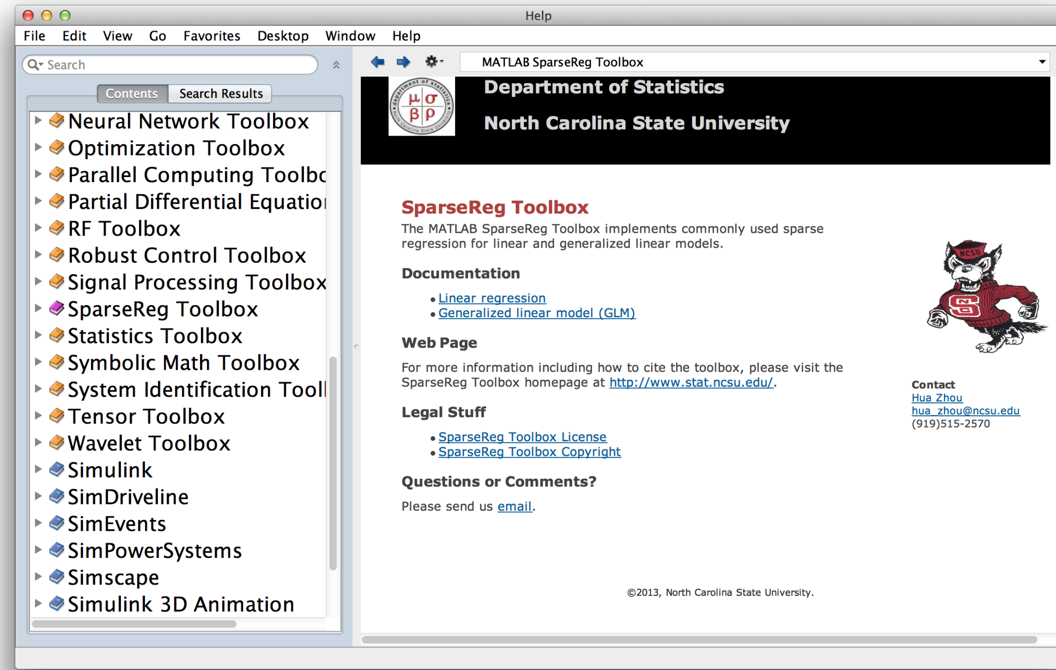


– Contents of a toolbox.

  * Function files. The `private` folder "hides" functions and compiled binaries not directly accessible by user
  * Demo scripts
  * The `html` (or any other name) folder holds the documentation generated by "publishing" demo scripts
  * The `info.xml` file contains essential information about the toolbox. It puts the toolbox to the start menu of MATLAB and links to the help documentation. See screenshots in next two slides for an example


MATLAB R2012a
File Edit Debug Parallel Desktop Window Help
Current Folder: /Users/hzhou3/workspace/pareto/codebase/matlab_fortran
Shortcuts How to Add What's New
Current Folder
Command Window
Workspace
html
private
Contents.m
COPYRIGHT.txt
demo_glm.m
demo_lsq.m
demo_paper.m
example_MandA_reg.m
example_SAheart.m
glm_maxlambda.m
glm_regpath.m
glm_sparsepath.m
glm_sparsereg.m
info.xml
INSTALL.txt
lsq_maxlambda.m
lsq_regpath.m
lsq_sparsepath.m
lsq_sparsereg.m
RELEASE_NOTES.txt
wolfpack_icon.gif
wolfpack_logo.gif
Command History
mex -setup
make
echo on
make
clear
clc
make
%-- 1/15/13 2:12 PM --%
%-- 1/15/13 2:13 PM --%
%-- 1/15/13 8:27 PM --%
make
clear
clc


MATLAB R2012a
Bioinformatics
Communications System
Control System
Curve Fitting
DSP System
Financial
Fixed-Point
Fuzzy Logic
Global Optimization
Image Acquisition
Image Processing
Instrument Control
Mapping
Neural Network
Optimization
Parallel Computing
Partial Differential Equation
RF
Robust Control
Signal Processing
SparseReg
Statistics
Symbolic Math
System Identification
Tensor
Wavelet
SparseReg Documentation
SparseReg Toolbox Site
MATLAB
Toolboxes
Simulink
Shortcuts
Desktop Tools
Web
Get Product Trials
Check for Updates
Preferences...
Find Files...
Help
Demos
Start



- More features of MATLAB.
  - Object-oriented programming (OOP)
  - GUI development.
  - More productivity tools: help report, TODO/FIXME report, code analyzer report, dependency report, ...
- Matlab summary.
  - Good points.
    * Highly efficient, esp. for numerical linear algebra.
    * Good IDE. Debugging and profiling is a breeze.
    * The language of choice for some technical computing areas. E.g., my research requires a lot solving ODE (ordinary differential equations) and tensor (multi-dimensional array) computing, which are not available (or not good enough) in R.
    * Existence of Matlab sets a high standard for other competing technical computing languages. Examples are R Studio and Julia.

    * Reasonably update with hardware technology. For example, > 200 native functions in Matlab supports GPU and distributed computing toolbox enables cluster computing of large scale problems.

  – Pitfalls.

    * Not open source! $$$
    * Limited statistical functionalities compared to R packages.

Summary of languages

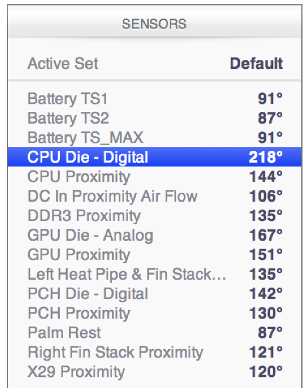
- Choosing language(s) for your project mostly depends your specific tasks, legacy code, and which "church you happen to frequent".
- Trade-off between development time and run time.
- Never believe others' benchmark results. Do your own profiling and benchmark.
- Don't be afraid to learn new languages. Having more tools in your toolbox is always a plus.

Parallel computing – what and why?

- *Parallel computing*, in contrast to serial computing, means carrying out computation simultaneously.
- Recent change in the landscape of parallel computing due to end of *frequency scaling* game in 2004.
- Run time = # instructions × avg. time per instruction.
- Cranking up clock frequency (frequency scaling) obviously reduces avg. time per instruction, but unfortunately ... increases *power consumption* and worsens *cooling problem* too.

$$P_{\text{ower}} = C_{\text{apacitance}} \times V_{\text{oltage}}^2 \times F_{\text{requency}}.$$

- This is what I see when running Matlab benchmark code on a MacBook Pro with a 2.6 GHz Intel Core i7 CPU.

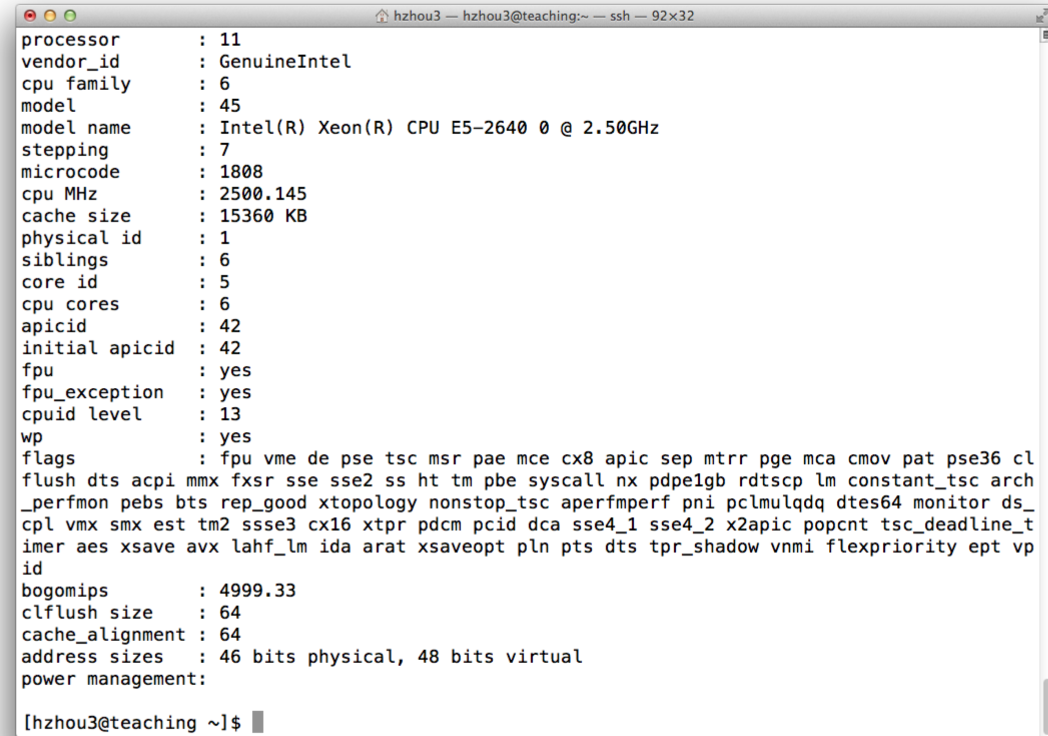


You can cook eggs on that CPU ...

- Intel canceled its Tejas and Jayhawk lines in 2004 due to power consumption constraint, which declared the end of *frequency scaling* and start of *parallel scaling*.
- This paradigm shift changes the way we do computation. Running the serial code written for single-core CPU on a multi-core CPU will not make it faster.
- There are many modes of parallel computing: multi-core, cluster, GPU, ...

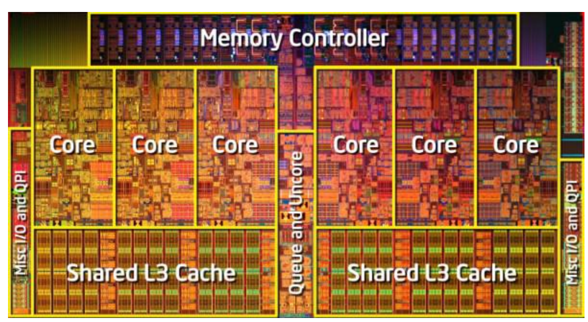
Multi-core parallel computing

- A typical CPU on a server.
  - Issue `cat /proc/cpuinfo` on `teaching.stat.ncsu.edu`.

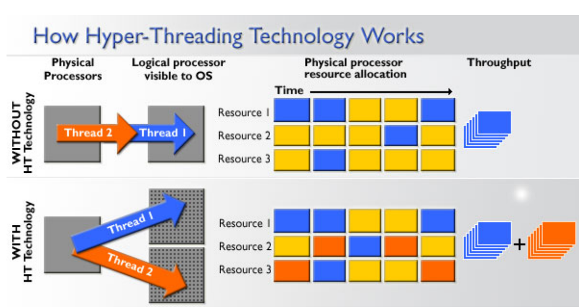


```
processor       : 11
vendor_id       : GenuineIntel
cpu family      : 6
model           : 45
model name      : Intel(R) Xeon(R) CPU E5-2640 0 @ 2.50GHz
stepping        : 7
microcode       : 1808
cpu MHz         : 2500.145
cache size      : 15360 KB
physical id     : 1
siblings        : 6
core id         : 5
cpu cores       : 6
apicid          : 42
initial apicid  : 42
fpu             : yes
fpu_exception   : yes
cpuid level     : 13
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 cl
flush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc arch
_perfmon pebs bts rep_good xtopology nonstop_tsc aperfmperf pni pclmulqdq dtes64 monitor ds_
cpl vmx smx est tm2 ssse3 cx16 xtpr pdcm pcid dca sse4_1 sse4_2 x2apic popcnt tsc_deadline_t
imer aes xsave avx lahf_lm ida arat xsaveopt pln pts dts tpr_shadow vnmi flexpriority ept vp
id
bogomips        : 4999.33
clflush size    : 64
cache_alignment : 64
address sizes   : 46 bits physical, 48 bits virtual
power management:

[hzhou3@teaching ~]$
```

– Intel© Xeon© E5-2640 chip, with 6 physical cores



– `Intel`'s hyperthreading "interleaves" two threads on one core



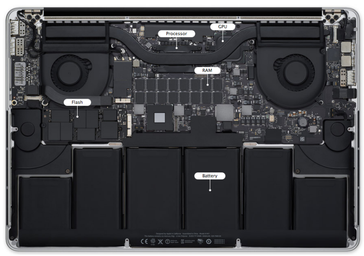
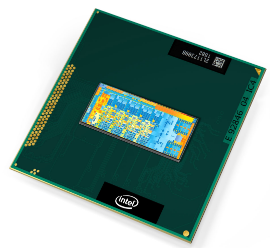
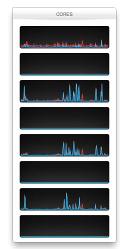
– In total it appears as 12 "processors" (logical processors, virtual cores, logical cores, siblings) to the OS on the `teaching` server.

– *Theoretical* throughput of the machine is
120 DP GFLOPS ≈ 4 DP FLOPs/cycle x 2.5 GHz x 12 logical processors
It's almost impossible to achieve this theoretical throughput.

- A typical CPU on current *PCs* and *laptops*.

  – For example, MacBook Pro has an Intel® Core™ i7-3720QM CPU @ 2.60GHz.

  – 4 physical cores and 8 threads. It appears as 8 virtual cores to the OS.

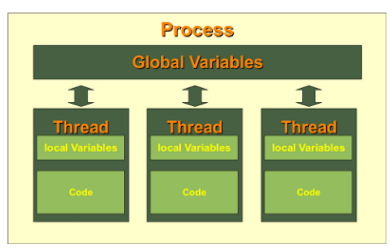




- Some terminology.

  – A *thread* is a (serial) sequence of instructions.

  – A *process* is a collection of threads, which share resources such as memory. Different processes run in separate memory spaces.

  – An *application* may have multiple processes



  – Example: Web browser (Google Chrome) is an application, each tab is a process, threads for each tab control text, music and so on.

- Multi-core or multi-thread computation relies on communication between processes/threads

  – Message passing libraries (MPI, PVM)

    * very powerful

    - designed for C/C++, Fortran
    - not easy to use from `R` (`rpvm`, `Rmpi` packages), `Matlab`, ...
  - Forking.
  - Sockets.

Ideally we need a transparent R interface that hides these communication details.

7 Lecture 7, Feb 4

Last Time

- Matlab.
- Parallel computing: what and why.

Today

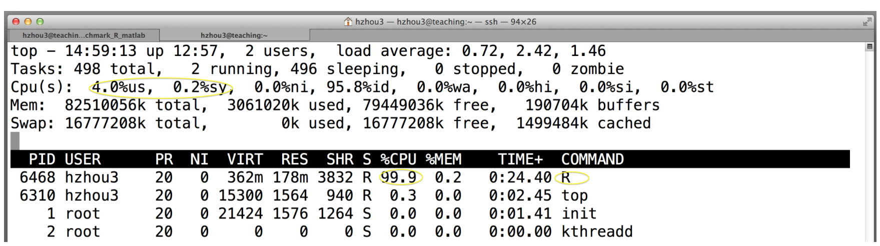
- A debug-profile-optimize session on NNMF (HW2)?
- Parallel computing: multi-core in R, Matlab, Julia.

A debug-profile-optimize exercise on NNMF (HW2)

I was preparing for the solution to HW2, and thought it might be a good example simple enough that we can go through a debug-profile-optimize exercise in class.

Multi-core computing in `R`

- Fact: base `R` is single-threaded.
- Running a benchmark script (random number generation, numerical linear algebra) on the `teaching` server occupies only 1 out of the 12 logical processors.



- To perform multi-core computation in `R`
  - Develop multi-threaded code or libraries in C/C++, Fortran, ... and call from `R`.
  - For *embarrassingly parallel* single-threaded tasks.
    * Option 1: Manually run multiple `R` sessions
    * Option 2: Make multiple `system ("Rscript")` calls. Typically automated by a scripting language (`Python`, `Perl`, shell script) or within `R`.
    * Option 3: Use package `parallel`

- `parallel` package in R.
  - Included in R since 2.14.0 (2011).
  - Based on the `snow` (Luke Tierney) and `multicore` (Urbanek) packages.
  - Authors: Brian Ripley, Luke Tieney, Simon Urbanek.
  - To find number of cores in the `teaching` server.

```
> library(parallel)
> detectCores()
[1] 12
```

  - How to utilize these cores to speed up computation?

- Case study: One common embarrassingly parallel task in statistics is Monte carlo simulation study.
  - E.g., in ST758 (2012, 2013), students are asked to carry out a simulation study to compare three procedures (LRT, eLRT, eRLRT) for testing $H_0 : \sigma_a^2 = 0$ vs $H_a : \sigma_a^2 > 0$ in a linear mixed model (variance component model)

$$\boldsymbol{y} \sim N(\mu \mathbf{1}, \sigma_2^2 \boldsymbol{V}_1 + \sigma_e^2 \boldsymbol{I}).$$

    Want to compare the size of power of three methods across 16 $\boldsymbol{V}_1$ pattern (stored in `n.pattern.list`) and 7 σ_a^2/σ_e^2 ratios (stored in `sigma2.ratio.list`).
  - Monte carlo estimate of size/power and its standard deviation for each method/pattern/σ_a^2 combination can be summarized in a table

		$\sigma_\alpha^2 = 0$	$\sigma_\alpha^2 = 0.1$	$\sigma_\alpha^2 = 0.2$	$\sigma_\alpha^2 = 0.5$	$\sigma_\alpha^2 = 1$	$\sigma_\alpha^2 = 2$	$\sigma_\alpha^2 = 5$
P_1	LRT	0.0171	0.0500	0.0893	0.2166	0.3892	0.5809	0.7851
		(0.0013)	(0.0022)	(0.0029)	(0.0041)	(0.0049)	(0.0049)	(0.0041)
	ELRT	0.0485	0.1175	0.1833	0.3387	0.5243	0.6968	0.8470
		(0.0022)	(0.0032)	(0.0039)	(0.0047)	(0.0050)	(0.0046)	(0.0036)
	RLRT	0.0487	0.1193	0.1833	0.3392	0.5241	0.6969	0.8479
		(0.0022)	(0.0032)	(0.0039)	(0.0047)	(0.0050)	(0.0046)	(0.0036)
P_2	LRT	0.0126	0.0433	0.0782	0.1822	0.3190	0.4905	0.6869
		(0.0011)	(0.0020)	(0.0027)	(0.0039)	(0.0047)	(0.0050)	(0.0046)
	ELRT	0.0517	0.1195	0.1778	0.3220	0.4618	0.6155	0.7789
		(0.0022)	(0.0032)	(0.0038)	(0.0047)	(0.0050)	(0.0049)	(0.0041)
	RLRT	0.0502	0.1140	0.1675	0.3164	0.4655	0.6224	0.7869
		(0.0022)	(0.0032)	(0.0037)	(0.0047)	(0.0050)	(0.0048)	(0.0041)
P_3	LRT	0.0144	0.0460	0.0825	0.2081	0.3386	0.5107	0.7100
		(0.0012)	(0.0021)	(0.0028)	(0.0041)	(0.0047)	(0.0050)	(0.0045)
	ELRT	0.0489	0.1230	0.1860	0.3403	0.4785	0.6343	0.7965

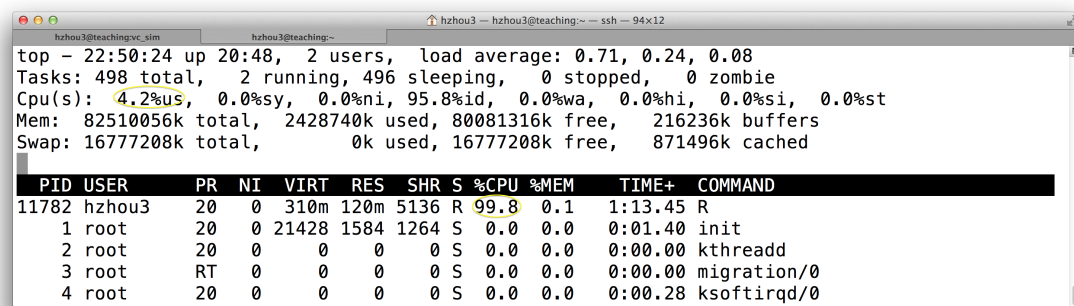
  - Suppose we have a function `compare.tests` that compares the methods at a fixed pattern and signal-to-noise ratio on a large number of Monte carlo replicates

```
compare.tests <- function( n.pattern, sigma2.ratio,
                           mc.size = 10000, ... )
```

- Need to loop over `n.pattern.list` and `sigma2.ratio.list`.
- 112 embarrassingly parallel tasks. Each might take long with many Monte carlo replicates.
- Let's try to parallelize the serial code (HW submission by Yichi Zhang) using the `parallel` package.
- Run the double loop (encapsulated in the `compare.tests.all` function) on the `teaching` server. Monte carlo sample size (`mc.size`) is set at (ridiculously small) 10

```
> # perform simulations -- serial code
> set.seed (123, "L'Ecuyer")
> system.time (result.serial <- compare.tests.all (
+   n.pattern.list, sigma2.ratio.list, mc.size = 10))
   user  system elapsed
238.410   0.148 239.409
```

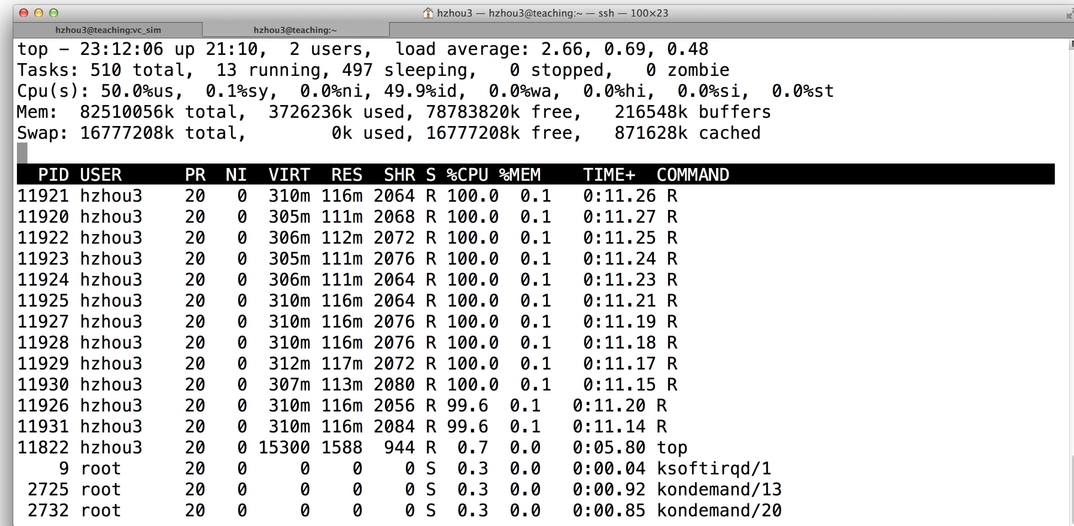
- Only 1 out of the 12 logical processors being used



- Run the same task using `mcmapply()` function (parallel analogue of `mapply`) in the package `parallel`

```
> # parallel simulations using mcmapply w/o load balancing
> set.seed (123, "L'Ecuyer")
> system.time (result.mcmapply <- mcmapply ( compare.tests,
+   rep (n.pattern.list, each = length (sigma2.ratio.list), times = 1),
+   rep (sigma2.ratio.list, each = 1, times = length (n.pattern.list)),
+   MoreArgs = list (mc.size = 10), mc.cores = 12))
   user  system elapsed
218.226   0.840  22.378
```

– `mc.cores=12` instructs using 12 cores



8 Lecture 8, Feb 9

Announcements

- *No* class and office hours this Wednesday. Instructor out of town.
- HW2 due this Wed @ 11:59PM. Tagging time will be your submission time. No tagging time = no hw submission.
- HW2 progress.
  - Matlab (CPU+GPU). Easy.
  - Julia (CPU+GPU). Use `CUDArt` and `CUBLAS` packages.
  - Python (CPU+GPU). Ask Xiang Zhang.
  - R (GPU?).
- A list of potential course projects.
  `http://hua-zhou.github.io/teaching/st790-2015spr/project.html`
  More topics will be added. Talk to me about your course project.

Last Time

- A debug-profile-optimize session on NNMF (HW2).
- Parallel computing: multi-core in R.

Today

- Multi-core computing in R (cont'd), Matlab, Julia.
- Cluster computing.

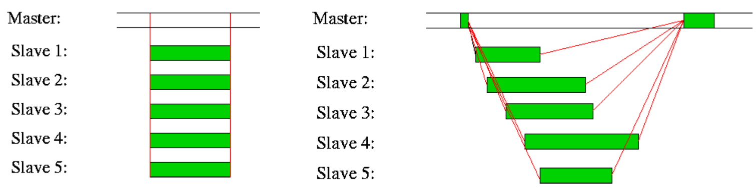
Multi-core computing in `R` (cont'd)

- Last time we demonstrated how to use `parallel` package to do multi-core computing in R on a simulation study. Steps are
  1. Write a function to carry out Monte carlo simulation and method comparison for one combination of levels (one cell in the table).
  2. Use `mcapply` for multi-core parallel computing.

3. Results are automatically collected in the master session. No need for extra scripting to collect results from parallel runs.

☞ Demo code is available on course webpage `http://hua-zhou.github.io/teaching/st790-2015spr/vcsim.r`

- *Load balancing*: Good for small number of parallel tasks with wildly different computation times
  No load balancing: Good for numerous parallel tasks with similar computation times



- Turn on load balancing by setting `mc.preschedule=FALSE`

```
> # parallel simulations using mcmapply with load balancing
> set.seed (123, "L'Ecuyer")
> system.time (result.mcmapplylb <- mcmapply ( compare.tests,
+   rep (n.pattern.list, each = length(sigma2.ratio.list), times = 1),
+   rep (sigma2.ratio.list, each = 1, times = length(n.pattern.list)),
+   MoreArgs = list (mc.size = 10), mc.cores = 12, mc.preschedule = FALSE))
   user  system elapsed
263.397   5.486  21.792
```

- *Forking* creates a new `R` process by taking a complete copy of the master process, including the workspace and *random number stream*. The copy will share memory with the master until modified so forking is very fast.

  ☞ `mcmapply`, `mclapply` and related functions rely on the *forking* capability of POSIX operating systems (e.g. Linux, MacOS) and is *not* available in Windows

- `parLapply`, `parApply`, `parCapply`, `parRapply`, `clusterApply`, `clusterMap`, and related functions create a cluster of workers based on either *socket* (default) or forking

```
cl <- makeCluster(<size of pool>)
# one or more parLapply calls
stopCluster(cl)
```

☞ Socket is available on all platforms: Linux, Mac OS, Windows.

- Same simulation example using `clusterMap` with load balancing

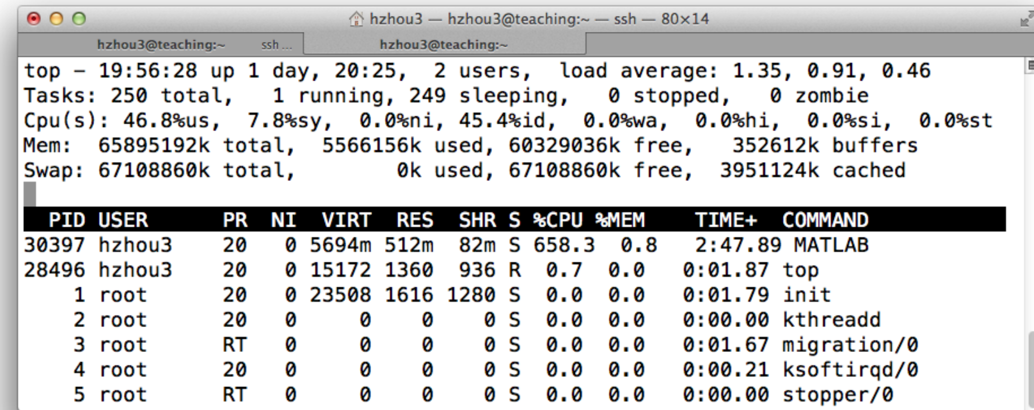
```
> # parallel simulations using clusterMap with load balancing
> cl <- makeCluster (getOption ("cl.cores", 12))
> clusterSetRNGStream(cl, 123)
> clusterExport (cl, c ("generate.design", "generate.response",
+                       "simulate.null.samples") )
> clusterEvalQ (cl, library(nlme) )
> clusterEvalQ (cl, library(RLRsim) )
> system.time (result.clusterMaplb <- clusterMap (cl, compare.tests,
+   rep (n.pattern.list, each = length (sigma2.ratio.list), times = 1),
+   rep (sigma2.ratio.list, each = 1, times = length (n.pattern.list)),
+   MoreArgs = list (mc.size = 10), .scheduling = "dynamic") )
   user  system elapsed
  0.115   0.011  22.310
> stopCluster (cl)
```

- `clusterSetRNGStream` control random number streams.
- `clusterExport` and `clusterEvalQ` copy environment of the master to slaves.
- Many *embarrassingly parallel* tasks in statistics can be organized in a similar way using `parallel`.
  - simulation across multiple factors (methods, generative models, signal/noise ratios, sparsity levels, ...)
  - bootstrap
  - solution path/surface in regularization methods
  - independent MCMC chains
  - cross validation
  - spatial prediction (kriging)
  - ...
- $5 \sim 15$ fold speed-up, depending on the number of cores on your machine.
- Need to make sure each task is thread-safe.

Multi-core and multi-thread computing in `Matlab`

- Many `Matlab` functions, esp. numerical linear algebra (MKL libraries), are multi-threaded since 2007.

- For example, running a benchmark script on the `teaching` server occupies up to all 7 (virtual) cores.



- `parfor` (parallel for loop) mechanism for embarrassingly parallel tasks.

- Parallel Computing Toolbox has more to offer (distributed array and SPMD, GPU computing, parallel MapReduce, cluster computing, ...)
  `http://www.mathworks.com/help/distcomp/index.html`

Multi-core and multi-thread computing in `Julia`

- Numerical linear algebra (OpenBLAS library) is already multi-threaded.

- Distributed computing capabilities are built in core language.
  `http://docs.julialang.org/en/release-0.3/manual/parallel-computing/`

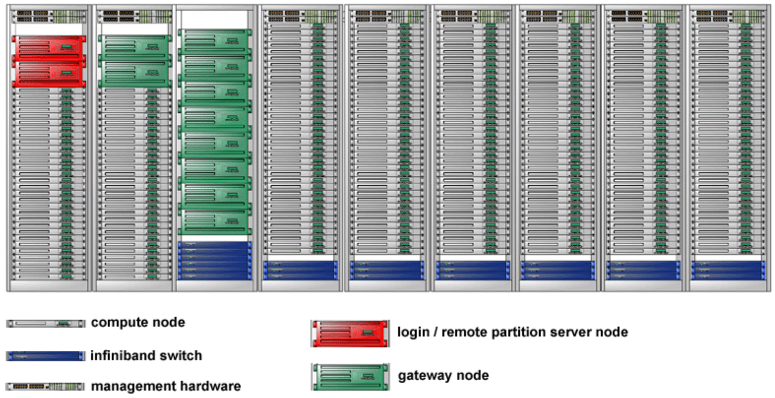
- Perhaps I can say more later this semester ...

Cluster computing

- Architecture of a computer cluster - computing parts.

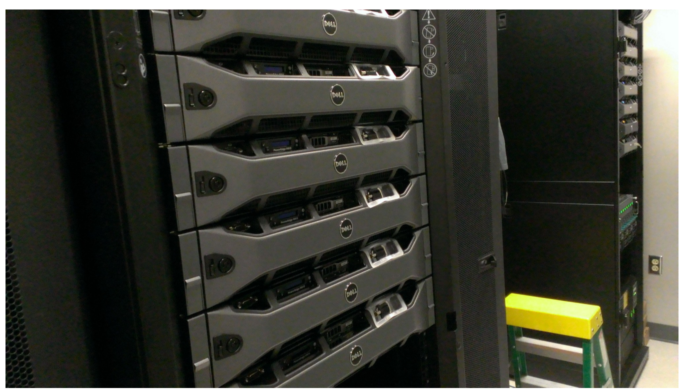
  - Cluster: a network of workstations (nodes).

– Compute nodes, login nodes, gateway (I/O) nodes, management nodes, file servers

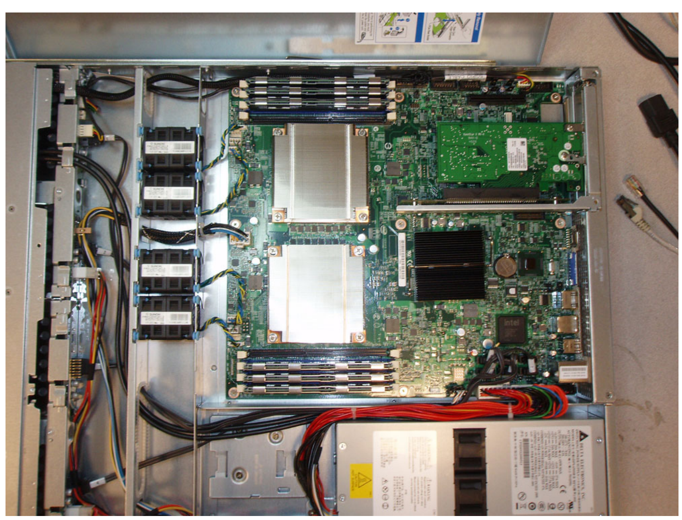


☞ When log into a cluster, always keep in mind you're interacting with the login nodes, not the commute nodes.

– A *chassis* (or *rack*) houses one or more nodes together with power, cooling, connectivity, management, ... This is a rack of our `beowulf` cluster.



– A *node* (or *blade*) contains one or more sockets, memory, a modest size disk drive holding OS, swap space, and a small local scratch space.



  - Each *socket* holds one processor, e.g., Intel `Xeon` or AMD `Opteron`.
  - A *processor* contains one or more cores (logical processors).
  - The *cores* perform FLOPS.

- Architecture of a computer cluster - network.
  - Infiniband: 2.5 Gbits/sec.
  - 4 x Infiniband: 10 Gbits/sec.
  - Hardware: Adapter, switches.
  - Nodes within a single chassis usually communicate faster.

Beowulf HPC cluster in department

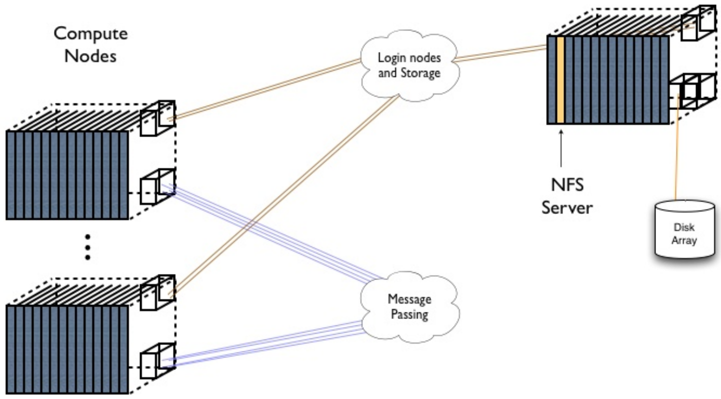
- Access via
  `ssh unityID@hpc.stat.ncsu.edu`
  ☞ Use `git` or `svn` to synchronize project files.

- Read instructions for submitting jobs.
  `http://www.stat.ncsu.edu/computing/beowulf_instructions.php`
  `bwsubmit` submits single-threaded jobs.
  `bwsubmit_mult` submits multi-threaded jobs.
  ☞ Each user can use 20 threads at a time.

- Write one script using `parallel` package and submit by `bwsubmit_mult` seems the easiest way for organizing embarrassingly parallel R jobs.

`henry2` HPC cluster at NCSU

- 1053 nodes (dual Xeon blade servers)



- For more info about `henry2` configuration: `http://www.ncsu.edu/itd/hpc/main.php`

- Ask your advisor for an account.

- Log in: `ssh unityID@login.hpc.ncsu.edu`.

- Users do *not* interact with compute nodes directly. Users submit jobs which are scheduled to be run on compute nodes.

- Some commonly used *job schedulers*

  - Platform LSF (Load Sharing Facility)
  - Altair PBS Pro
  - Sun Grid Engine
  - Microsoft HPC Server 2008
  - TORQUE

- `henry2` uses LSF.

- Some commonly used LSF commands

  - `bsub`: submit a batch job to LSF system
  - `bkill`: kill a running job
  - `bjobs`: status of jobs in the queue
  - `bpeek`: display output and error files
  - `bhist`: history of one or more LSF jobs

- `bqueues`: info about LSF batch queues

- Example: Let's try to run an R script `simulate.fork.r` on `henry2`.
- Have following files ready in the working directory
  - `simulate.fork.r`
    R script file
  - `R.csh`
    R configuration file
  - `henry2_submit_fork`
    Shell script file for LSF job submission
  - `RLRsim_2.0-11.tar.gz`
    necessary files for installing R libraries
- `simulate.fork.r` is the R script to be run on cluster

```
...
# RLRsim package required for LR and RLR test
install.packages ("./RLRsim_2.0-11.tar.gz", repos=NULL, lib="./libs")
library (RLRsim,lib.loc="./libs")
# load libraries
library (compiler)
library (nlme) # requied for lme()
library (parallel)
...
# parallel simulations using mcmapply with load balance
set.seed (123, "L'Ecuyer")
mc = detectCores ()
mc
system.time (result.mcmapplylb <- mcmapply (compare.tests,
  rep (n.pattern.list, each = length(sigma2.ratio.list), times = 1),
  rep (sigma2.ratio.list, each = 1, times = length(n.pattern.list)),
  MoreArgs = list (mc.size = 10), mc.cores = mc, mc.preschedule = FALSE))
...
```

- `henry2_submit_fork` is the shell script for LSF job submission

```
[hzhou3@login05 henry2]$ cat henry2_submit_fork
#!/bin/tcsh
#BSUB -n 12
#BSUB -W 10
#BSUB -R em64t
#BSUB -R span[hosts=1]
source ./R.csh
R CMD BATCH --vanilla simulate.fork.r
#BSUB -o out.%J
#BSUB -e err.%J
[hzhou3@login05 henry2]$
```

– `#BSUB -n 12` requests 12 processors (logical cores, threads).

– `#BSUB -W 10` requests maximum of 10 minutes.

– `#BSUB -R em64t` requests 64-bit machines.

– `#BSUB -R span[hosts=1]` requests all 12 processors to be on the same machine. Note `mcmapply` relies on forking, which is a shared memory model.

– `#BSUB -o out.%J` and `#BSUB -o err.%J` specify output files

- `R.csh` configures the path for R program

```
[hzhou3@login05 henry2]$ cat R.csh
#
# file to set up csh/tcsh shell environment
# to use R
#
setenv R /usr/local/apps/R/em64t/R-2.15.1_gnu_mpich2/bin
set path = ($R $path)

if !($?MANPATH) then
    setenv MANPATH /usr/local/apps/R/em64t/R-2.15.1_gnu_mpich2/share/man/man1:`man -w`
else
    setenv MANPATH /usr/local/apps/R/em64t/R-2.15.1_gnu_mpich2/share/man/man1:$MANPATH
endif

[hzhou3@login05 henry2]$
```

- Submit job to LSF scheduler by `bsub` and check status by `bjobs`

```
[hzhou3@login05 henry2]$ bsub < henry2_submit_fork
Job <111391> is submitted to default queue <short>.
[hzhou3@login05 henry2]$ bjobs
JOBID   USER    STAT  QUEUE      FROM_HOST   EXEC_HOST   JOB_NAME   SUBMIT_TIME
111391  hzhou3  PEND  short      login05                 *-e err.%J Mar 10 22:13
[hzhou3@login05 henry2]$ bjobs
JOBID   USER    STAT  QUEUE      FROM_HOST   EXEC_HOST   JOB_NAME   SUBMIT_TIME
111391  hzhou3  RUN   short      login05     12*bc2c3    *-e err.%J Mar 10 22:13
[hzhou3@login05 henry2]$
```

- Wait for the job to finish. Several files are generated in working directory
  - `out.111391` and `err.111391`: standard and error LSF output files
  - `simulate.fork.r.Rout`: screen display of R session
  - `result.fork.RData`: output data saved by R script
- Portion of `out.111391`

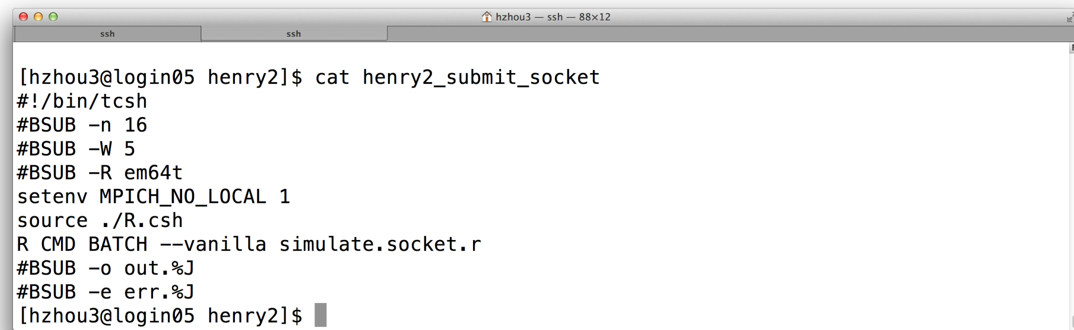
```
[hzhou3@login05 henry2]$ cat out.111391

------------------------------------------------------------
Sender: LSF System <lsfadmin@n2c3-13>
Subject: Job 111391: <#!/bin/tcsh;#BSUB -n 12 ;#BSUB -W 10 ;#BSUB -R em64t ;#BSUB -R span[hosts=1];source ./R.csh;R CMD BATCH
--vanilla simulate.fork.r ;#BSUB -o out.%J;#BSUB -e err.%J> Done

Job <#!/bin/tcsh;#BSUB -n 12 ;#BSUB -W 10 ;#BSUB -R em64t ;#BSUB -R span[hosts=1];source ./R.csh;R CMD BATCH --vanilla simulat
e.fork.r ;#BSUB -o out.%J;#BSUB -e err.%J> was submitted from host <login05> by user <hzhou3> in cluster <henry2>.
Job was executed on host(s) <12*n2c3-13>, in queue <short>, as user <hzhou3> in cluster <henry2>.
</home/hzhou3> was used as the home directory.
</home/hzhou3/workspace/ST810-2013-Spring/slides/Lec08_parallel/material/vc_sim/henry2> was used as the working directory.
Started at Sun Mar 10 22:13:24 2013
Results reported at Sun Mar 10 22:13:53 2013

Your job looked like:

------------------------------------------------------------
# LSBATCH: User input
#!/bin/tcsh
#BSUB -n 12
#BSUB -W 10
#BSUB -R em64t
#BSUB -R span[hosts=1]
source ./R.csh
R CMD BATCH --vanilla simulate.fork.r
#BSUB -o out.%J
#BSUB -e err.%J

------------------------------------------------------------


Successfully completed.
```

- Portion of `simulate.fork.r.Rout`

```
...
> # parallel simulations using mcmapply with load balance
> set.seed (123, "L'Ecuyer")
> mc = detectCores ()
> mc
[1] 12
> system.time (result.mcmapplylb <- mcmapply (
+   compare.tests,
+   rep (n.pattern.list, each = length(sigma2.ratio.list), times = 1),
+   rep (sigma2.ratio.list, each = 1, times = length(n.pattern.list)),
+   MoreArgs = list (mc.size = 10), mc.cores = mc, mc.preschedule = FALSE))
   user  system elapsed
284.954   8.231  26.172
>
```

```
> # save results
> save(n.pattern.list, sigma2.ratio.list,
+   result.mcmapplylb, file = "result.fork.RData")
>
>
> proc.time()
   user  system elapsed
287.951   8.374  29.419
```

- Shell script for submitting `simulate.socket.r` which uses `clusterMap`




```
[hzhou3@login05 henry2]$ cat henry2_submit_socket
#!/bin/tcsh
#BSUB -n 16
#BSUB -W 5
#BSUB -R em64t
setenv MPICH_NO_LOCAL 1
source ./R.csh
R CMD BATCH --vanilla simulate.socket.r
#BSUB -o out.%J
#BSUB -e err.%J
[hzhou3@login05 henry2]$
```

  - Note `clusterMap` relies on socket and in principle works with any number of processors
  - `setenv MPICH_NO_LOCAL 1` specifies that all MPI messages will be passed through sockets, not using shared memory available on a node

Other HPC resources on campus

- BRC cluster. R/Matlab and GPUs available. Ask Tao Hu.
  `http://scarlatti.statgen.ncsu.edu/cluster_workshop/doku.php`
- ARC cluster. Ask your advisor for an account. R/Matlab not available. Only compiled code. GPUs available.
  `http://moss.csc.ncsu.edu/~mueller/cluster/arc/`

9 Lecture 9, Feb 16

Announcements

- HW2 graded. `grade_unityID.md` committed to your `master` branch.

Last Time

- Cluster computing.

Today

- HW2 feedback.
- GPU computing.

HW2 feedback

- Solution sketch in Matlab and Julia:
  `http://hua-zhou.github.io/teaching/st790-2015spr/hw02sol.html`

- Languages (Matlab, Julia, R, Python).
  - For CPU code, Julia offers more low-level memory management capabilities, leading to more efficient computation.
  - For GPU programming, Matlab wins hands down in ease of use. Julia GPU computing relies on the `CUDArt.jl` and `CUBLAS.jl` packages. Currently `CUBLAS.jl` implements approximately half of BLAS functions, including `gemm`. For non-BLAS computations such as elementwise multiplication and division, users need to write their own CUDA kernel functions.

    For using GPU in Python, ask Xiang Zhang and Zhen Han. For using `gputools` package in R, ask Brian Naughton.

- Effects of starting points. Non-convexity implies possible existence of multiple local minima. Identifiability issue: $\boldsymbol{V}\boldsymbol{W} = \boldsymbol{V}\boldsymbol{O}\boldsymbol{O}^{-1}\boldsymbol{W}$ for any non-singular $r \times r$ matrices. What happens when starting from $v_{ij}^{(0)} = w_{jk}^{(0)} \equiv 1$?

- Interpretability of basis images from NNMF. The following figure (Hastie et al., 2009, p55) contrasts the different basis images obtained by NNMF, VQ (vector quantization), and PCA. For a mathematical explanation of what NNMF does, see Donoho and Stodden (2004).

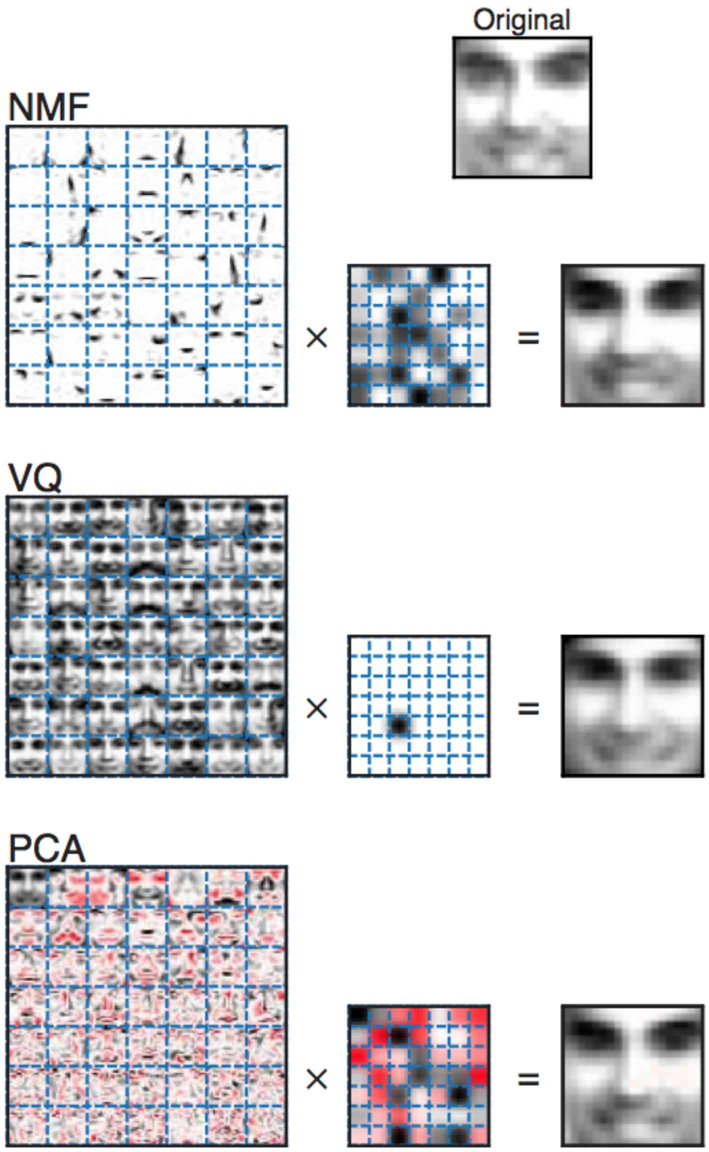


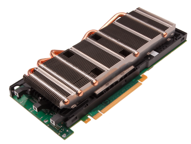
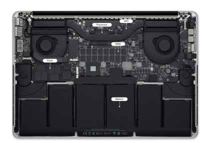
FIGURE 14.33. *Non-negative matrix factorization (NMF), vector quantization (VQ, equivalent to k-means clustering) and principal components analysis (PCA) applied to a database of facial images. Details are given in the text. Unlike VQ and PCA, NMF learns to represent faces with a set of basis images resembling parts of faces.*

- Different kinds of GPUs. I ran the same Matlab and Julia code on the teaching server, a desktop, and a laptop. They represent common GPUs we see everyday. Note these models are a couple years old and stand for technology around 2011.
- CPU vs GPU.
  - Gain of GPU over CPU depends on specific cards and precision. Baby GPUs on laptops show no gain on DP computations.

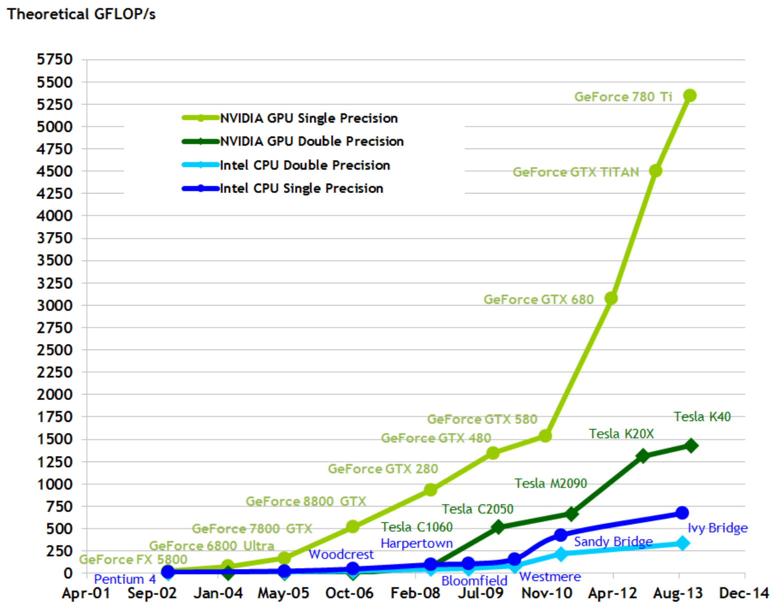
- GPU SP (single precision) vs GPU DP (double precision).
  - Do they get same objective values? Do we have to use double precision? For example, in MCMC, Monte Carlo errors often far exceed numerical roundoff errors.
  - How's the timing using SP vs DP? Tesla card has similar SP and DP performance. GTX card has higher SP performance than DP. Baby GPUs on laptops show no gain on DP computations.

Introduction to GPU computing

- GPUs are ubiquitous: servers, desktops, and laptops.

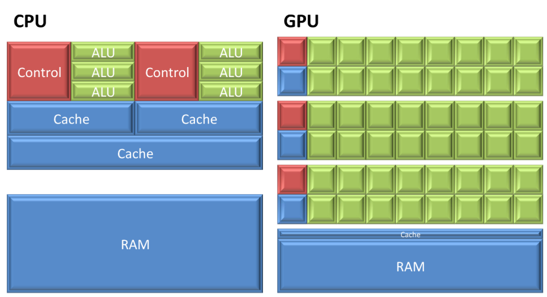
NVIDIA GPUs	Tesla M2090	GTX 580	GT 650M
			
Computers	servers, cluster	desktop	laptop
			
Main usage	scientific computing	gaming	gaming
Current version	K40	GTX 980	GTX 900M
Memory	6GB	1.5GB	1GB
Memory bandwidth	177GB/sec	192GB/sec	80GB/sec
Number of cores	512	512	384
Processor clock	1.3GHz	1.5GHz	0.9GHz
Peak DP performance	666Gflops		
Peak SP performance	1332Gflops	1581Gflops	691Gflops
Release price	\$2500	\$500	OEM

- Cost effective for high performance computing.

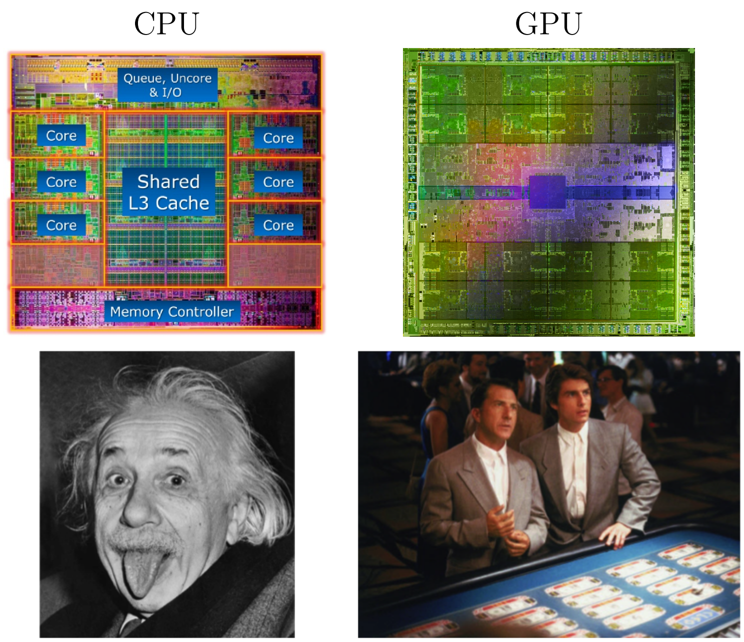


Floating-Point Operations per Second - Nvidia CUDA C Programming Guide
Version 6.5 - 24/9/2014 - copyright Nvidia Corporation 2014

- GPU architecture vs CPU architecture.



  - GPUs contain 100s of processing cores on a single card; several cards can fit in a desktop PC
  - Each core carries out the same operations in parallel on different input data – single program, multiple data (SPMD) paradigm.
  - Extremely high arithmetic intensity *if* one can transfer the data onto and results off of the processors quickly.



An analogy taken from Andrew Beam's presentation in ST790. Also see `https://www.youtube.com/watch?v=-P28LKWTzrI`.

- Which cards to use?
  - Three major manufacturers of GPUs: AMD, NVIDIA, and Intel.
  - So far NVIDIA cards are more widely adopted for GPGPU.
    E.g., GPU servers in our department and NCSU `henry2` cluster all have NVIDIA.
  - NVIDIA has a much richer set of GPU math libraries

	AMD	NVIDIA	Intel
Cards	ATI Radeon	GTX, Tesla	Xeon Phi copressor
Language	OpenCL	CUDA C/C++, PGI CUDA Fortran	C/C++, Fortran, OpenCL
GPU math libraries	clMath (BLAS,FFT)	cuBLAS, cuFFT, cuSPARSE, cuSolver cuRAND, CUDA MATH, Thrust, ...	MKL
Platforms	Linux, Windows	Linux, Windows, MacOS	Linux, Windows

☞ On the other hand, cross-platform feature of OpenCL, adopted by Intel and AMD, is attractive.

- My experience with GPGPU (general purpose GPU computing).
  - Almost always involve (new) algorithm development and/or revamping CPU code.

- Research before going for GPGPU.
- Easier to develop in `C/C++` (free compiler), `Fortran` (compiler $), and `Matlab`.
- Do not reinvent the wheel – use libraries.

- Before using GPUs, do following.

0. Frustrated by slow code ...
1. Am I using the right algorithm(s)?
   Go to your ST758 notes or a numerical analysis book.
2. Repeat: *Profile* and optimize original code
3. Can a compiled language or optimized library (MKL, ATLAS) help?
4. Identify the bottleneck routine and research the potential gain on GPU. Do your own benchmark specific to your own problem and data size
5. Can my data fit into GPU memory?
6. Can other steps besides the bottleneck be easily implemented on GPU? Will any of them become the new bottleneck?
7. Decide the toolchain: `Matlab`, `Julia`, `CUDA`, PGI toolchain, ...

- GPGPU development toolchains.
  - Use a higher level language such as `Matlab`, `Julia` or `Python`, if they happen to provide all functions we need.
  - CUDA® toolchain provided by NVIDIA®
    `https://developer.nvidia.com/cuda-zone`
    * `C/C++`
    * free
    * only for NIVIDA cards
  - PGI® toolchain (CUDA Fortran)
    `https://www.pgroup.com/resources/cudafortran.htm`
    * `C/C++`, `Fortran`
    * $$$
    * only for NVIDIA cards
  - OpenCL™ (Open Computing Language)
    * open source

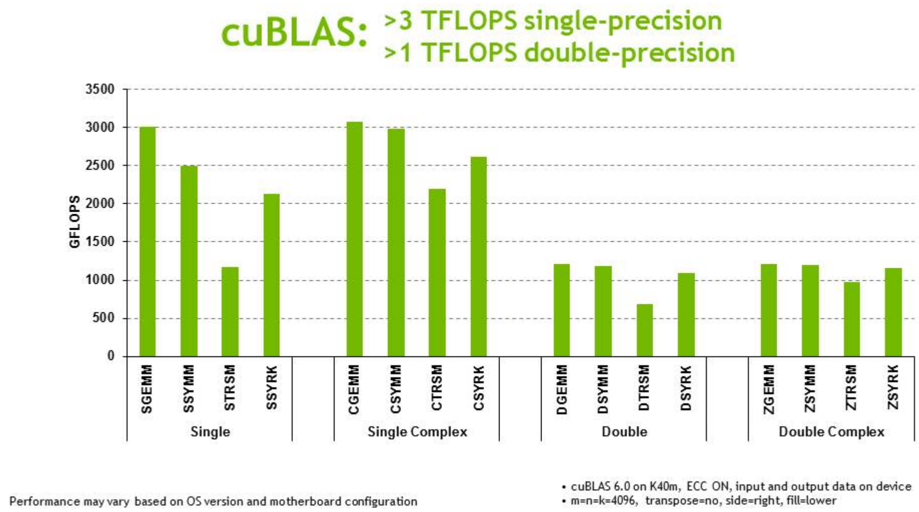
    * Specs for cross-platform, parallel programming of modern processors (PCs, servers, handheld/embedded devices)
    * Adopted by Intel, AMD, NVIDIA, Qualcomm, ...

Mathematical libraries on GPUs

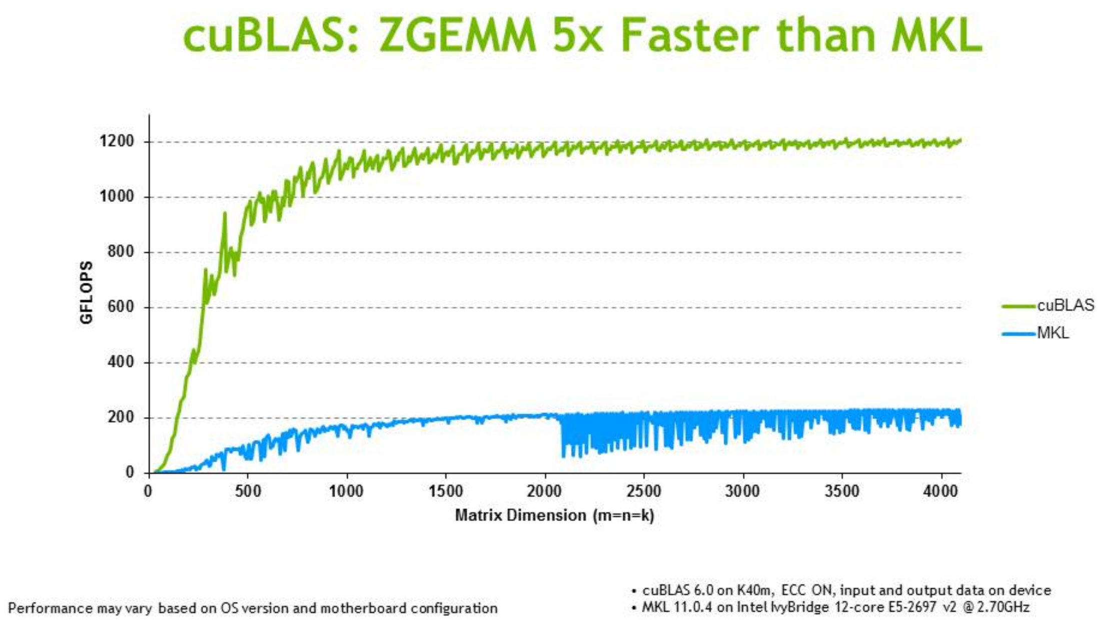
- Many statistical computing subroutiens are covered by the BLAS, LAPACK, sparse linear algebra, random number generation, and other standard libraries.
- Availability of mathematical libraries on GPUs.
  - NVIDIA© CUDA© math libraries.
    * Optimized for NVIDIA GPUs
    * cuBLAS, cuSPARSE, cuRAND, cuFFT, CUDA Math Library, Thrust (data structures and algorithms), cuSolver (CUDA v7.0).
    * Platforms: Linux (free), MacOS (free) and Windows (free)
  - Intel© MKL library.
    * Support both Intel© CPUs and Xeon Phi coprocessors since v11.0 (2013)
    * BLAS, LAPACK, FFT, sparse linear algebra, random number generation, ...
    * Platforms: Linux (free) and Windows ($), no MacOS support ☹
  - AMD© clMath© library.
    * For AMD GPUs
    * BLAS, FFT
    * Platforms: Linux (free) and Windows (free)
  - Third-party libraries
    * CULA ($): CUDA LAPACK
    * MAGMA (free): OpenCL LAPACK

  ☞ NVIDIA's rich collection of math libraries is very attractive.

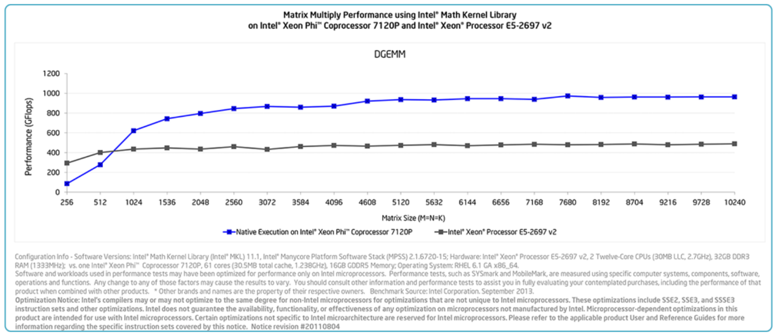
- Some dense linear algebra benchmark results.
  - cuBLAS on NVIDIA K40m.



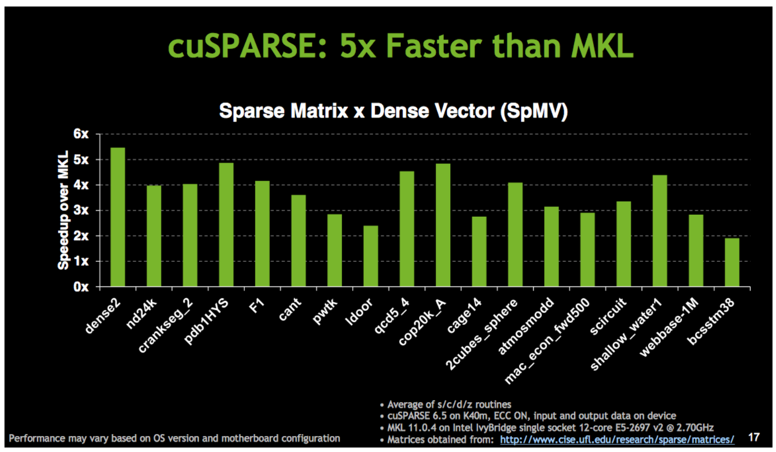
– `zgemm` cuBLAS on NVIDIA K40m vs MKL on Xeon E5-2697 v2 @ 2.7GHz.



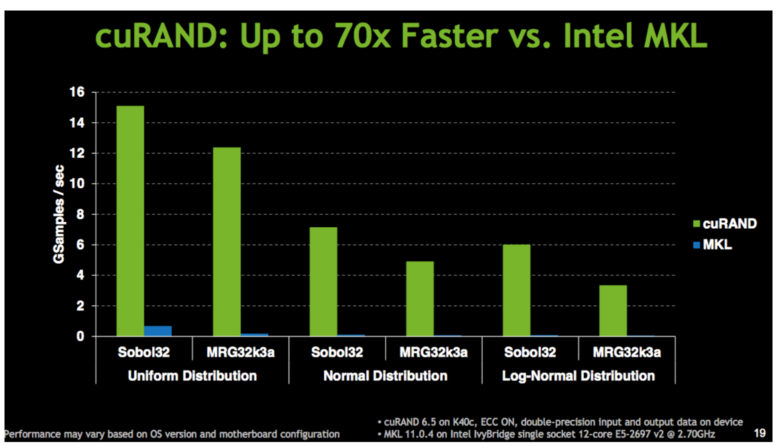
– `dgemm` MKL on Xeon Phi© 7120P vs MKL on Xeon 12-core E5-2697 v2 @ 2.7GHz.



- Sparse linear algebra. cuSPARSE on K40m vs MKL on Xeon E5-2697 v2 @ 2.7GHz.



- Random number generation. cuRAND on K40m vs MKL on Xeon E5-2697 v2 @ 2.7GHz.



10 Lecture 10, Feb 18

Announcements

- TA's Friday office hour changes to Thu Feb 19 @ 2P-3P.
- HW3 deadline extended to Feb 25 @ 11:59PM.

Last Time

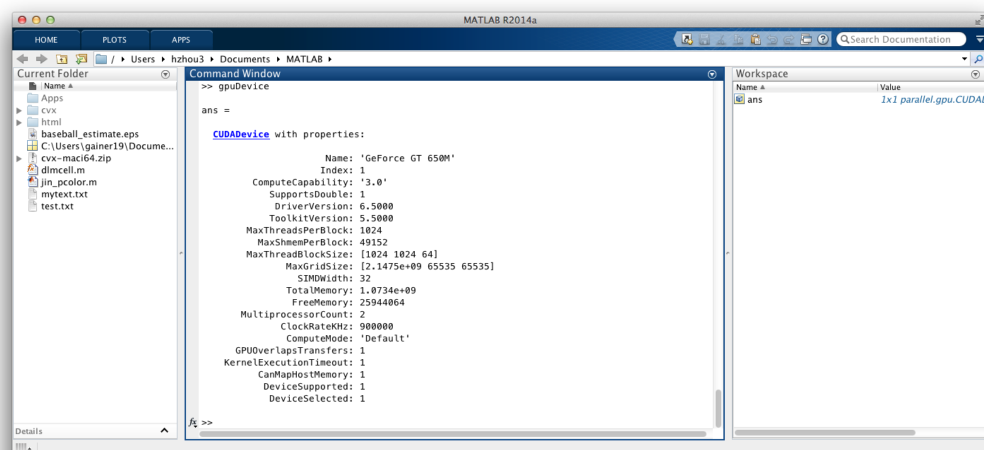
- GPU computing: introduction.

Today

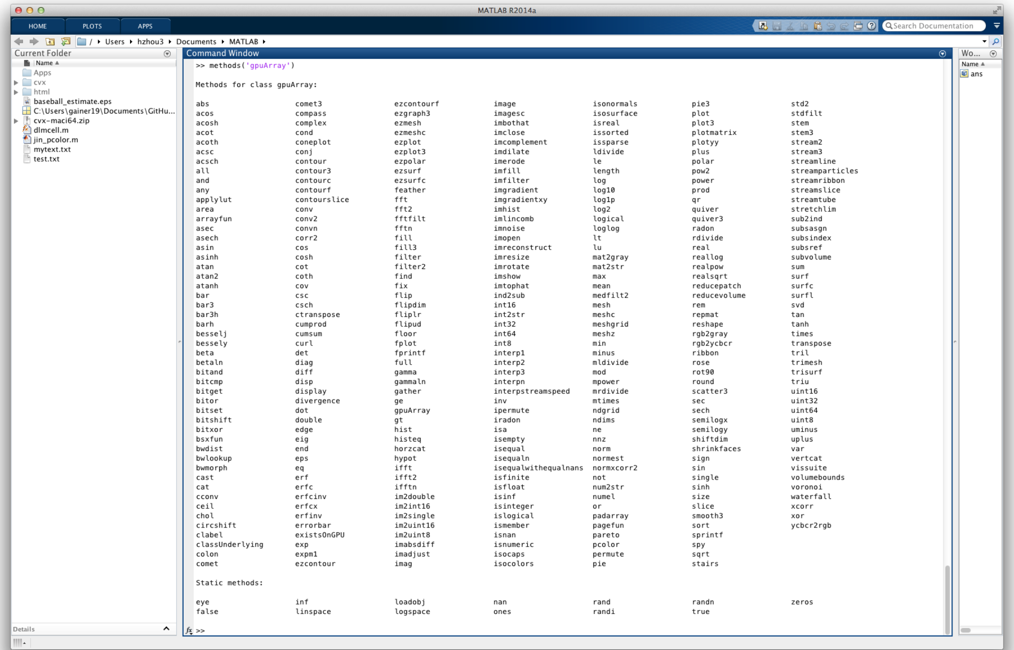
- GPU computing: Matlab, Julia, R.
- GPU computing: case studies.
- Convex programming.

GPU computing in Matlab

- Getting started.
  - Query available GPU devices: `gpuDevice()`.



  - List built-in functions that support GPU: `methods('gpuArray')`. Nearly 300 built-in functions in `Matlab 2014a` support GPU.

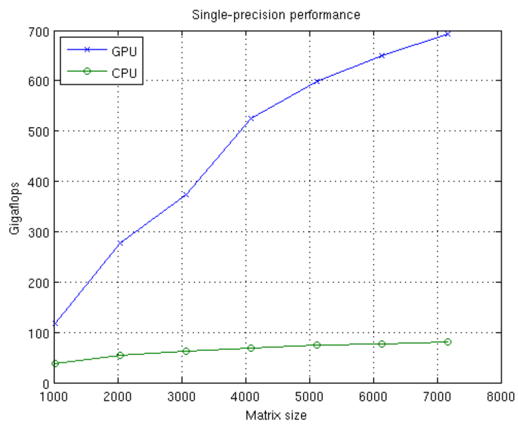


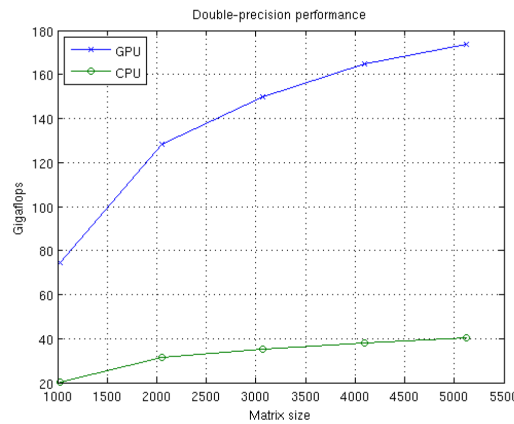
- Scheme for GPU algorithm development on `Matlab`.

```
% transfer data to GPU and initialize variables
gX = gpuArray (X);
gY = gpuArray (Y);
gBetahat = gpuArray.randn (5, 1);
...

% computation on GPU
...

% transfer result off GPU
betahat = gather (gBetahat);
```

☞ Key: minimize memory transfer between host memory and GPU memory

- Always benchmark the specific bottleneck routine in CPU. If the bottleneck routine does not enjoy GPU acceleration, there is no point embarking on GPU computing. E.g., to benchmark `A\b` (solve linear equations) on my desktop: `paralleldemo_gpu_backslash()` in `Matlab 2014a`





Intel i7 960 CPU vs NVIDIA GTX 580 GPU

GPU computing in R

- Not supported in base `R` (opportunity? `HiPLARM` package).
- A few contributed packages in specific application areas: `gputools` (some data-mining algorithms), `cudaBayesreg` (fMRI analysis), ...
- Develop in `C/C++` or `Fortran` and call compiled code from `R`.

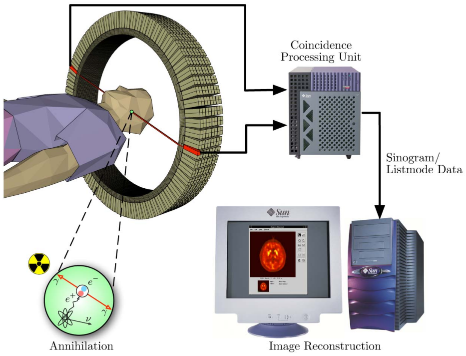
GPU computing in Julia

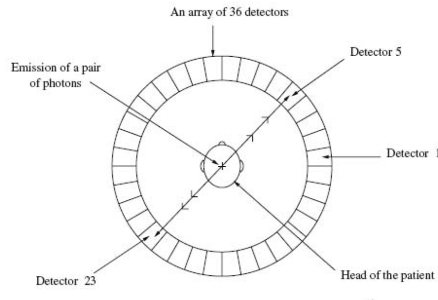
Various packages are being developed at `https://github.com/JuliaGPU`. See HW2 solution for the NNMF example using `CUDArt.jl` and `CUBLAS.jl` packages.

GPU case study 1: NNMF

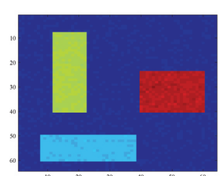
If your language (Matlab or Julia) happens to provide interface to all GPU libraries you need, then the job can be easily done. In the NNMF example, we only need matrix multiplication and elementwise matrix multiplication and division. See HW2 solution for sample code.

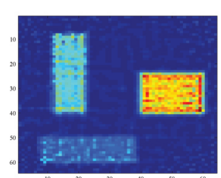
GPU case study 2: PET imaging

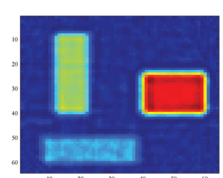




- Data: tube readings $\boldsymbol{y} = (y_1, \ldots, y_d)$.
- Estimate: photon emission intensities (pixels) $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_p)$.
- Poisson Model:
$$Y_i \sim \text{Poisson}\left(\sum_{j=1}^{p} c_{ij}\lambda_j\right),$$
where c_{ij} is the (pre-calculated) cond. prob. that a photon emitted by j-th pixel is detected by i-th tube.
- Log-likelihood:
$$L(\boldsymbol{\lambda}|\boldsymbol{y}) = \sum_i \left[y_i \ln\left(\sum_j c_{ij}\lambda_j\right) - \sum_j c_{ij}\lambda_j \right] + \text{const.}$$
Essentially a Poisson regression with constraint $\lambda_j \geq 0$.
- Issues: *grainy image* and *slow convergence*







- Regularized log-likelihood for smoother image:

$$
\begin{aligned}
& L(\boldsymbol{\lambda}|\boldsymbol{y}) - \frac{\mu}{2} \sum_{\{j,k\} \in \mathcal{N}} (\lambda_j - \lambda_k)^2 \\
= & \sum_i \left[y_i \ln \left(\sum_j c_{ij} \lambda_j \right) - \sum_j c_{ij} \lambda_j \right] - \frac{\mu}{2} \sum_{\{j,k\} \in \mathcal{N}} (\lambda_j - \lambda_k)^2,
\end{aligned}
$$

  where $\mu \geq 0$ is a tuning constant.

- Which algorithm?

  - Newton algorithm needs to solve a large linear system at each iteration ☹
  - In ST758 (2014, notes p145-p149), we derived an MM algorithm for minimizing the regularized log-likelihood.

- MM algorithm for PET:

  Initialize: $\lambda_j^{(0)} = 1$
  repeat
  $z_{ij}^{(t)} = (y_i c_{ij} \lambda_j^{(t)}) / (\sum_k c_{ik} \lambda_k^{(t)})$
  for $j = 1$ to p **do**
  $a = -2\mu|\mathcal{N}_j|$, $b = \mu(|\mathcal{N}_j|\lambda_j^{(t)} + \sum_{k \in \mathcal{N}_j} \lambda_k^{(t)}) - 1$, $c = \sum_i z_{ij}^{(t)}$
  $\lambda_j^{(t+1)} = (-b - \sqrt{b^2 - 4ac})/(2a)$
  end for
  until convergence occurs

- Parameter constraints $\lambda_j \geq 0$ are satisfied when start from positive initial values.

- Update of $z_{ij}^{(t)}$ succumbs to BLAS (matrix-vector multiplication) and elementwise multiplication and division.

- The loop for updating pixels can be carried out independently – *massive parallelism.*

- A simulation example with $n = 2016$ and $p = 4096$ (provided by Ravi Varadhan). CPU code implemented using BLAS in the GNU Scientific Library (GSL). GPU code implemented using cuBLAS.

  - Runtime on a typical computer in 2009:
    CPU: Xeon E5450 @ 3GHZ (1 thread)
    GPU: NVIDIA GeForce GTX 280

	CPU			GPU			
Penalty μ	Iters	Time	Function	Iters	Time	Function	Speedup
0	100000	14790	-7337.152765	100000	282	-7337.153387	52
10^{-7}	24457	3682	-8500.083033	24457	70	-8508.112249	53
10^{-6}	6294	919	-15432.45496	6294	18	-15432.45586	51
10^{-5}	589	86	-55767.32966	589	2	-55767.32970	43

– Runtime on a typical computer in 2011:
CPU: i7 @ 3.20GHZ (1 thread)
GPU: NVIDIA GeForce GTX 580

	CPU			GPU			
Penalty μ	Iters	Time	Function	Iters	Time	Function	Speedup
0	100000	11250	-7337.152765	100000	140	-7337.153387	80
10^{-7}	24506	2573	-8500.082605	24506	35	-8508.112249	74
10^{-6}	6294	710	-15432.45496	6294	9	-15432.45586	79
10^{-5}	589	67	-55767.32966	589	0.8	-55767.32970	84

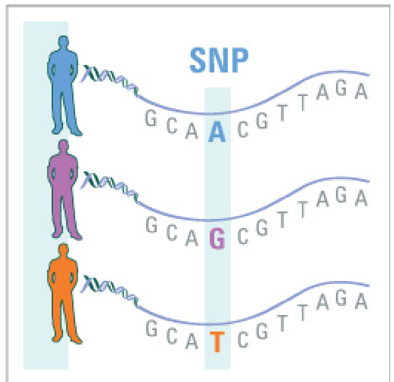
☞ Performance of CPU increases by about *30%*, while GPU increases by *100%*

- Lessons learnt.
  - Algorithm development. *EM/MM*
    * separate variables. Break a complex optimization into numerous independent simple optimizations (massive parallelism)
    * avoid solving large linear systems; only BLAS routines involved
    * exploit high throughput of BLAS routines on GPU
  - *cuBLAS* library eases the GPU implementation
- C++ source code is available at `http://hua-zhou.github.io/teaching/st790-2015spr/pet.tar.gz`.

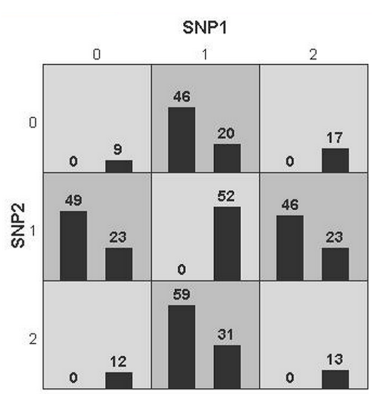
GPU case study 3: MDR for GWAS

- SNP and GWAS.
  - Human genome consists of 3 billion pairs of letters (A,C,G,T)

- Two people's genome sequences are 99.9% identical
- SNP (single nucleotide polymorphism) is a single-letter change in DNA
- About 1 in 1000 DNA letters vary in the form of a SNP
- Genome-wide association study (GWAS) tries to find association of the trait of interest (disease or not, blood pressure, height, ...) and each SNP



- MDR for detecting SNP interactions.
  - Multifactor dimensionality reduction (MDR) is a method for detecting association of a binary trait (0/1,control/disease) and SNP pairs
  - For each SNP pair
    - count number of 0s and 1s for each genotype combination
    - declare that genotype combination as causal ($n_1 > n_0$) or protective ($n_1 < n_0$)
    - predict disease status using the declared causal/protective status of genotype combinations
  - Rank SNP pairs according to their predictive power
  - Alternatively we can do Pearson's χ^2 test for contingency table



- Computation challenge and parallelism.
  - For either MDR or Pearson, we need to construct tables for $\binom{p}{2}$ SNP pairs
  - For $p = 10^6$, $\binom{p}{2} \approx 5 \times 10^{11}$
  - *Massive parallelism*: tables for SNP pairs $(1, 2), \ldots, (p - 1, p)$ obviously can be constructed in parallel
  - How to organize? Merry-go-round.

    Golub and Van Loan (1996, Section 8.4)

432 CHAPTER 8. THE SYMMETRIC EIGENVALUE PROBLEM

lelism of the latter algorithm. To illustrate this, suppose $n = 4$ and group the six subproblems into three *rotation sets* as follows:

$$\begin{aligned} rot.set(1) &= \{(1,2),(3,4)\} \\ rot.set(2) &= \{(1,3),(2,4)\} \\ rot.set(3) &= \{(1,4),(2,3)\} \end{aligned}$$

Note that all the rotations within each of the three rotation sets are "nonconflicting." That is, subproblems (1,2) and (3,4) can be carried out in parallel. Likewise the (1,3) and (2,4) subproblems can be executed in parallel as can subproblems (1,4) and (2,3). In general, we say that

$$(i_1, j_1), (i_2, j_2), \ldots, (i_N, j_N) \qquad N = (n-1)n/2$$

is a *parallel ordering* of the set $\{(i,j) \mid 1 \le i < j \le n\}$ if for $s = 1{:}n-1$ the rotation set $rot.set(s) = \{ (i_r, j_r) : r = 1 + n(s-1)/2{:}ns/2 \}$ consists of nonconflicting rotations. This requires n to be even, which we assume throughout this section. (The odd n case can be handled by bordering A with a row and column of zeros and being careful when solving the subproblems that involve these augmented zeros.)

A good way to generate a parallel ordering is to visualize a chess tournament with n players in which everybody must play everybody else exactly once. In the $n = 8$ case this entails 7 "rounds." During round one we have the following four games:

1	3	5	7
2	4	6	8

$$rot.set(1) = \{ (1,2),(3,4),(5,6),(7,8) \}$$

i.e., 1 plays 2, 3 plays 4, etc. To set up rounds 2 through 7, player 1 stays put and players 2 through 8 embark on a merry-go-round:

1	2	3	5
4	6	8	7

$$rot.set(2) = \{(1,4),(2,6),(3,8),(5,7)\}$$

1	4	2	3
6	8	7	5

$$rot.set(3) = \{(1,6),(4,8),(2,7),(3,5)\}$$

1	6	4	2
8	7	5	3

$$rot.set(4) = \{(1,8),(6,7),(4,5),(2,3)\}$$

1	8	6	4
7	5	3	2

$$rot.set(5) = \{(1,7),(5,8),(3,6),(2,4)\}$$

1	7	8	6
5	3	2	4

$$rot.set(6) = \{(1,5),(3,7),(2,8),(4,6)\}$$

8.4. JACOBI METHODS 433

1	5	7	8
3	2	4	6

$$rot.set(7) = \{(1,3),(2,5),(4,7),(6,8)\}$$

We can encode these operations in a pair of integer vectors $top(1{:}n/2)$ and $bot(1{:}n/2)$. During a given round $top(k)$ plays $bot(k)$, $k = 1{:}n/2$. The pairings for the next round is obtained by updating top and bot as follows:

```
function: [new.top, new.bot] = music(top, bot, n)
    m = n/2
    for k = 1:m
        if k = 1
            new.top(1) = 1
        else if k = 2
            new.top(k) = bot(1)
        elseif k > 2
            new.top(k) = top(k - 1)
        end
        if k = m
            new.bot(k) = top(k)
        else
            new.bot(k) = bot(k + 1)
        end
    end
```

Using **music** we obtain the following parallel order Jacobi procedure.

Algorithm 8.4.4 (Parallel Order Jacobi) Given a symmetric $A \in \mathbb{R}^{n \times n}$ and a tolerance $tol > 0$, this algorithm overwrites A with $V^T AV$ where V is orthogonal and $\mathrm{off}(V^T AV) \le tol \| A \|_F$. It is assumed that n is even.

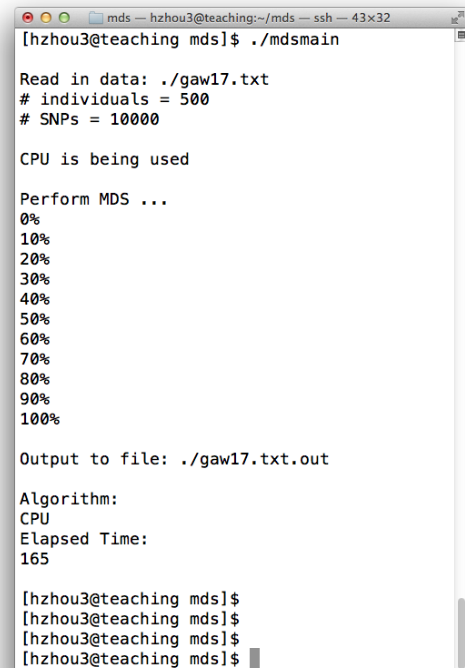
```
V = I_n
eps = tol||A||_F
top = 1:2:n; bot = 2:2:n
while off(A) > eps
    for set = 1:n - 1
        for k = 1:n/2
            p = min(top(k), bot(k))
            q = max(top(k), bot(k))
            (c, s) = sym.schur2(A, p, q)
            A = J(p, q, θ)^T A J(p, q, θ)
            V = V J(p, q, θ)
        end
        [top, bot] = music(top, bot, n)
    end
end
```

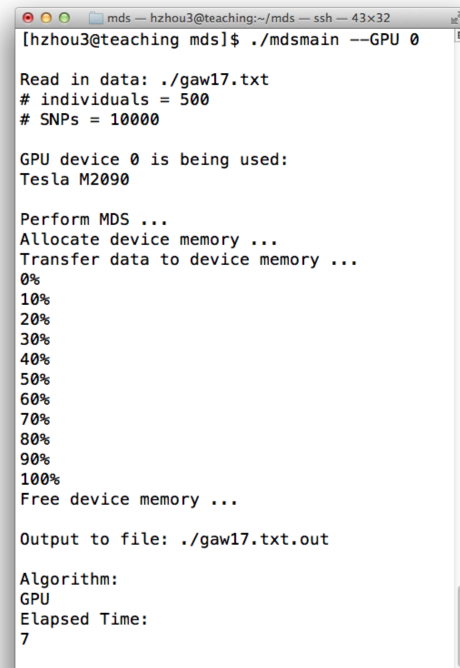
- Try it.
  - Download the source code
    `wget http://hua-zhou.github.io/teaching/st790-2015spr/mds.tar.gz`
  - Extract files `tar -zxvf mds.tar.gz`
  - Browse the contents of the `mds` folder
    * source: `main.cpp`, `mds.cpp`, `mds.h`, `mds_kernel.cu`
    * make file: `Makefile`
    * test data: `gaw17.txt` (500 individuals, 10000 SNPs)
  - Compiling on the `teaching` server

```
g++ -c -O2 -I/usr/local/cuda-6.5/include *.cpp
nvcc -O2 -c *.cu
g++ -o mdsmain -L/usr/local/cuda-6.5/lib64 -lcudart *.o
```

or use the make file. It yields the executable `mdsmain`.

– Run it on `teaching` server.
CPU: Xeon E5-2640 @ 2.5GHZ (1 thread)
GPU: NVIDIA Tesla M2090.
We see > 20 fold speed up.





- CPU host code.

mds.cpp

mds.cpp 〉 No Selection

```cpp
void MDS::SolveMDS_CPU (void) {
    int snp1, snp2, idx=0;
    int* classifyInt = new int[9];
    int* gidx = new int[N];
    int* genotypeRowIdx;
    for (int r=0; r<S-1; r++) {
        if (r%((S-1)/10)==0) cout << 10*(r/((S-1)/10)) << "%\n";
        for (int pidx=0; pidx<S/2; pidx++) {
            snp1 = ((pidx-r)>=0)? (pidx-r):(S-1+pidx-r);
            snp2 = ((S-r-2-pidx)>=0)? (S-r-2-pidx):(2*S-pidx-r-3);
            if (pidx==(S/2-1)) snp2=S-1;
            for (int i=0; i<9; i++) classifyInt[i]=0;
            for (int i=0; i<N; i++) {
                genotypeRowIdx = hGenotypeInt+i*S;
                gidx[i] = 3*(*(genotypeRowIdx+snp1))+(*(genotypeRowIdx+snp2));
                classifyInt[gidx[i]] += hPhenotypeInt[i];
            }
            for (int i=0; i<9; i++)
                classifyInt[i] = (classifyInt[i]>=0)? 1:-1;
            hPredictInt[idx] = 0;
            for (int i=0; i<N; i++) {
                hPredictInt[idx] += (classifyInt[gidx[i]]==hPhenotypeInt[i]);
            }
            idx++;
        }
    }
    delete [] classifyInt;
    delete [] gidx;
}
```

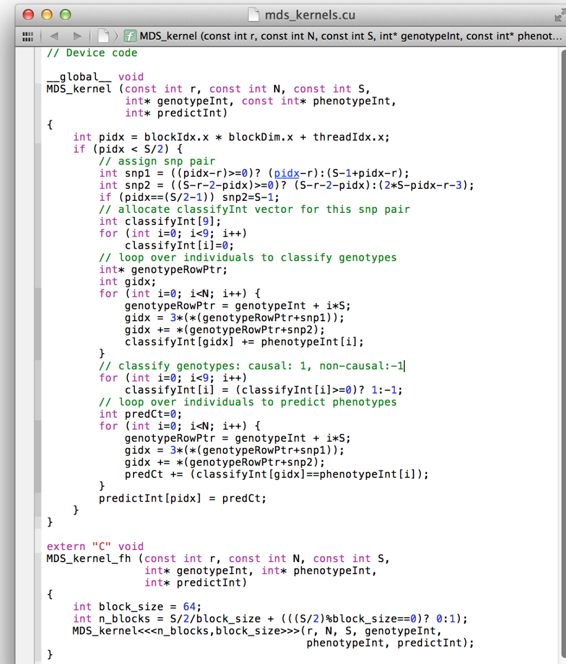
- GPU host code.

mds.cpp

mds.cpp 〉 MDS::SolveMDS_CUDA (void)

```cpp
void MDS::Setup_CUDA (void) {
    // allocate vectors/arrays in device memory
    cout << "Allocate device memory ... \n";
    //size_t pitch;
    //cudaMallocPitch((void**)&dGenotypeInt, &pitch, S*sizeof(int), N);
    cudaMalloc((void**)&dGenotypeInt, N*S*sizeof(int));
    cudaMalloc((void**)&dPhenotypeInt, N*sizeof(int));
    cudaMalloc((void**)&dPredictInt, (S/2)*sizeof(int));

    // transfer data to device
    cout << "Transfer data to device memory ... \n";
    //cudaMemcpy2D(dGenotypeInt, pitch, hGenotypeInt, 2*S*sizeof(int),
    //  2*S*sizeof(int), N, cudaMemcpyHostToDevice);
    cudaMemcpy(dGenotypeInt, hGenotypeInt, N*S*sizeof(int),
        cudaMemcpyHostToDevice);
    cudaMemcpy(dPhenotypeInt, hPhenotypeInt, N*sizeof(int),
        cudaMemcpyHostToDevice);
}

void MDS::Cleanup_CUDA (void) {
    // free vectors/arrays in device memory
    cout << "Free device memory ... \n";
    if(dGenotypeInt) cudaFree(dGenotypeInt);
    if(dPhenotypeInt) cudaFree(dPhenotypeInt);
    if(dPredictInt) cudaFree(dPredictInt);
}

void MDS::SolveMDS_CUDA (void) {
    // initialize
    Setup_CUDA();
    // (S-1) sweeps to classify genotypes
    for (int round=0; round<S-1; round++) {
        // cout << "round " << round << endl;
        if (round%((S-1)/10)==0) cout << 10*(round/((S-1)/10)) << "%\n";
        MDS_kernel_fh (round, N, S, dGenotypeInt, dPhenotypeInt, dPredictInt);
        cudaMemcpy(hPredictInt+round*(S/2), dPredictInt, S/2*sizeof(int),
            cudaMemcpyDeviceToHost);
    }
    // clean up
    Cleanup_CUDA();
}
```

- GPU device code.




```
// Device code

__global__ void
MDS_kernel (const int r, const int N, const int S,
            int* genotypeInt, const int* phenotypeInt,
            int* predictInt)
{
    int pidx = blockIdx.x * blockDim.x + threadIdx.x;
    if (pidx < S/2) {
        // assign snp pair
        int snp1 = ((pidx-r)>=0)? (pidx-r):(S-1+pidx-r);
        int snp2 = ((S-r-2-pidx)>=0)? (S-r-2-pidx):(2*S-pidx-r-3);
        if (pidx==(S/2-1)) snp2=S-1;
        // allocate classifyInt vector for this snp pair
        int classifyInt[9];
        for (int i=0; i<9; i++)
            classifyInt[i]=0;
        // loop over individuals to classify genotypes
        int* genotypeRowPtr;
        int gidx;
        for (int i=0; i<N; i++) {
            genotypeRowPtr = genotypeInt + i*S;
            gidx = 3*(*(genotypeRowPtr+snp1));
            gidx += *(genotypeRowPtr+snp2);
            classifyInt[gidx] += phenotypeInt[i];
        }
        // classify genotypes: causal: 1, non-causal:-1
        for (int i=0; i<9; i++)
            classifyInt[i] = (classifyInt[i]>=0)? 1:-1;
        // loop over individuals to predict phenotypes
        int predCt=0;
        for (int i=0; i<N; i++) {
            genotypeRowPtr = genotypeInt + i*S;
            gidx = 3*(*(genotypeRowPtr+snp1));
            gidx += *(genotypeRowPtr+snp2);
            predCt += (classifyInt[gidx]==phenotypeInt[i]);
        }
        predictInt[pidx] = predCt;
    }
}

extern "C" void
MDS_kernel_fh (const int r, const int N, const int S,
               int* genotypeInt, int* phenotypeInt,
               int* predictInt)
{
    int block_size = 64;
    int n_blocks = S/2/block_size + (((S/2)%block_size==0)? 0:1);
    MDS_kernel<<<n_blocks,block_size>>>(r, N, S, genotypeInt,
                                        phenotypeInt, predictInt);
}
```

- Lessons learnt.
  - Recognize massive parallelism. Common in genomics and statistics
  - Algorithm development. Merry-go-round for organizing parallel pairs
- C++ source code is available at `http://hua-zhou.github.io/teaching/st790-2015spr/mds.tar.gz`.

Convex optimization problems

- A *mathematical optimization problem*, or just *optimization problem*, has the form

$$\begin{array}{ll} \text{minimize} & f_0(\boldsymbol{x}) \\ \text{subject to} & f_i(\boldsymbol{x}) \le b_i, \quad i = 1, \ldots, m. \end{array}$$

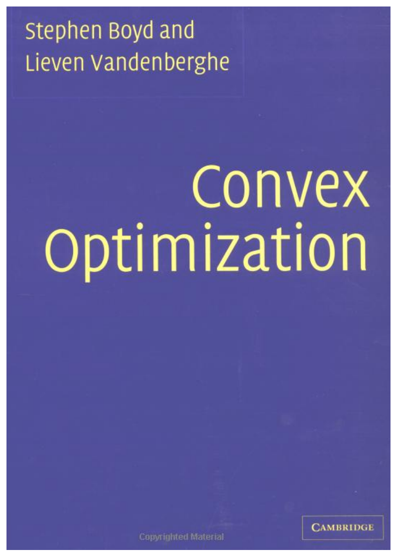
  Here $f_0 : \mathbf{R}^n \mapsto \mathbf{R}$ is the *objective* function and $f_i : \mathbf{R}^n \mapsto \mathbf{R}$, $i = 1, \ldots, m$, are the *constraint* functions.

  ☞ An equality constraint $f_i(\boldsymbol{x}) = b_i$ can be absorbed into inequality constraints $f_i(\boldsymbol{x}) \le b_i$ and $-f_i(\boldsymbol{x}) \le -b_i$.

- If the objective and constraint functions are convex, then it is called a *convex optimization problem*.

☞ In a convex optimization problem, only linear equality constraint of form $\boldsymbol{Ax} = \boldsymbol{b}$ is allowed.

- Convex optimization is becoming a *technology*. Therefore it is important to recognize, formulate, and solve convex optimization problems.
- A definite resource is the book *Convex Optimization* by Boyd and Vandenberghe, which is freely available at `http://stanford.edu/~boyd/cvxbook/`. Same website has links to slides, code, and lecture videos.



- In this course, we learn basic terminology and how to recognize and solve some standard convex programming problems.

11 Lecture 11, Feb 23

Announcements

- HW4 posted (Linear Programming). Due next Friday Mar 6 @ 11:59PM.

Last Time

- GPU computing: Matlab, Julia, R.
- GPU computing: case studies.
- Convex optimization: introduction

Today

- Convex sets and convex functions.

Convex sets

- The *line segment* (interval) connecting points $\boldsymbol{x}$ and $\boldsymbol{y}$ is the set

$$\{\boldsymbol{z} : \alpha \boldsymbol{x} + (1-\alpha)\boldsymbol{y} \text{ for all } \alpha \in [0,1]\}.$$

- A set C is *convex* if for every pair of points $\boldsymbol{x}$ and $\boldsymbol{y}$ lying in C the entire line segment connecting them also lies in C.

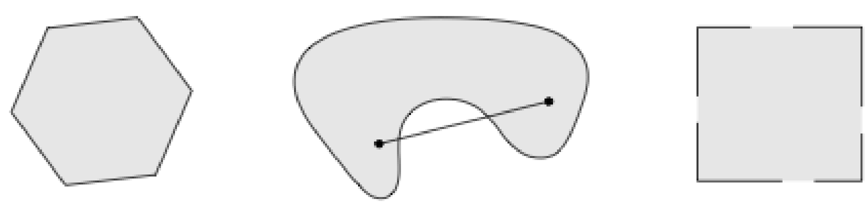


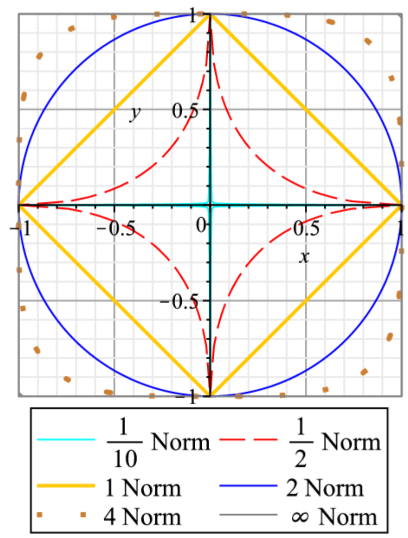
Figure 2.2 Some simple convex and nonconvex sets. *Left.* The hexagon, which includes its boundary (shown darker), is convex. *Middle.* The kidney shaped set is not convex, since the line segment between the two points in the set shown as dots is not contained in the set. *Right.* The square contains some boundary points but not others, and is not convex.

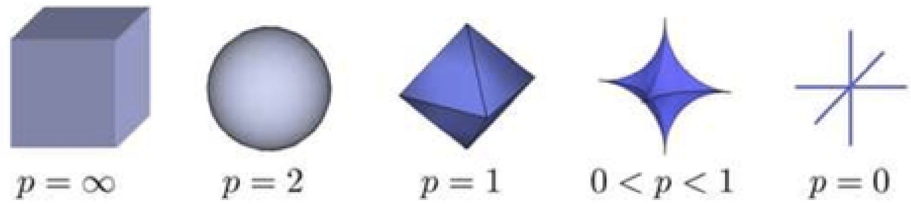
- Examples of convex sets.

  1. Any singleton.

2. $\mathbf{R}^n$.

3. Any *normed ball* $B_r(\boldsymbol{c}) = \{\boldsymbol{x} : \|\boldsymbol{x} - \boldsymbol{c}\| \leq r\}$, open or closed, of radius r centered at $\boldsymbol{c}$.





☞ $l_p(\boldsymbol{x}) = \left(\sum_{i=1}^n |x_i|^p\right)^{1/p}$ is *not* a proper norm for $0 < p < 1$.

4. Any *hyperplane* $\{\boldsymbol{x} : \boldsymbol{x}^T\boldsymbol{v} = c\}$.

5. Any closed *half space* $\{\boldsymbol{x} : \boldsymbol{x}^T\boldsymbol{v} \leq c\}$ or open half space $\{\boldsymbol{x} : \boldsymbol{x}^T\boldsymbol{v} < c\}$.

6. Any polyhedron

$$\begin{aligned}
\mathcal{P} &= \{\boldsymbol{x} : \boldsymbol{a}_j^T\boldsymbol{x} \leq b_j, j = 1, \ldots, m, \boldsymbol{c}_j^T\boldsymbol{x} = d_j, j = 1, \ldots, p\} \\
&= \{\boldsymbol{x} : \boldsymbol{A}\boldsymbol{x} \preceq \boldsymbol{b}, \boldsymbol{C}\boldsymbol{x} = \boldsymbol{d}\}.
\end{aligned}$$

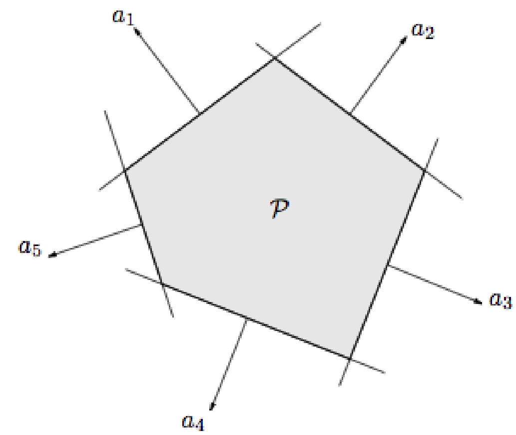


Figure 2.11 The polyhedron $\mathcal{P}$ (shown shaded) is the intersection of five halfspaces, with outward normal vectors $a_1, \ldots, a_5$.

7. The set $\mathbf{S}^n_{++}$ of $n \times n$ pd matrices and the set $\mathbf{S}^n_{+}$ of $n \times n$ psd matrices.

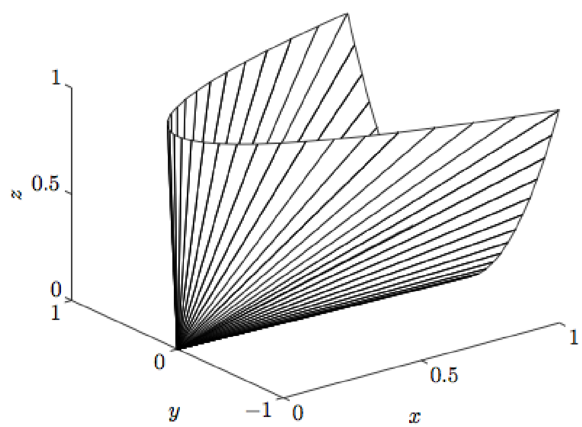


Figure 2.12 Boundary of positive semidefinite cone in $\mathbf{S}^2$.

8. The translate $C + \boldsymbol{w}$ of a convex set C.
9. The image $\boldsymbol{A}(C)$ of a convex set C under a linear map $\boldsymbol{A}$.
10. The inverse image $\boldsymbol{A}^{-1}(C)$ of a convex set C under a linear map $\boldsymbol{A}$.
11. The Cartesian product of two convex sets.

- A set C is a *cone* if for each $\boldsymbol{x} \in C$ the set $\{\theta \boldsymbol{x} : \theta \geq 0\}$ is also in C (closed by multiplication by nonnegative scalars). A cone that is convex is called a *convex cone*.
- Examples of cone:

1. The set $\mathbf{S}^n_{+}$ of psd matrices is a convex cone.

2. Is the set $\mathbf{S}^n_{++}$ of pd matrices a cone?
3. The set $\{(\boldsymbol{x}, t) : \|\boldsymbol{x}\|_2 \le t\}$ is called an *ice cream* (or *Lorentz*, or *second order*, or *quadratic*) *cone*.

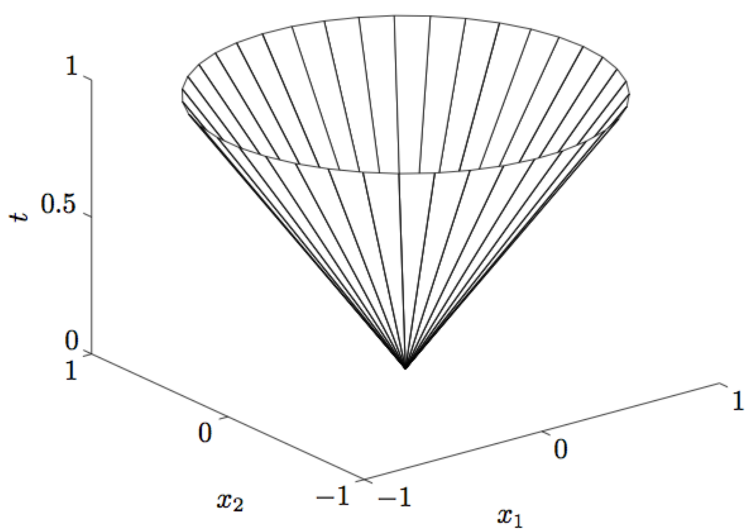


Figure 2.10 Boundary of second-order cone in $\mathbf{R}^3$, $\{(x_1, x_2, t) \mid (x_1^2 + x_2^2)^{1/2} \le t\}$.

4. Any *norm cone* $\{(\boldsymbol{x}, t) : \|\boldsymbol{x}\| \le t\}$ is a convex cone.
5. Can you give a non-convex cone?

- A set C is said to be *affine* if

$$\{\boldsymbol{z} : \theta\boldsymbol{x} + (1-\theta)\boldsymbol{y} \text{ for all } \theta \in \mathbf{R}\} \subset C$$

for all $\boldsymbol{x}, \boldsymbol{y} \in C$. Note θ is *not* restricted to the unit interval. An affine set is convex but not conversely. Every affine set A can be represented as a translate $\boldsymbol{v} + S$ of a vector subspace S.

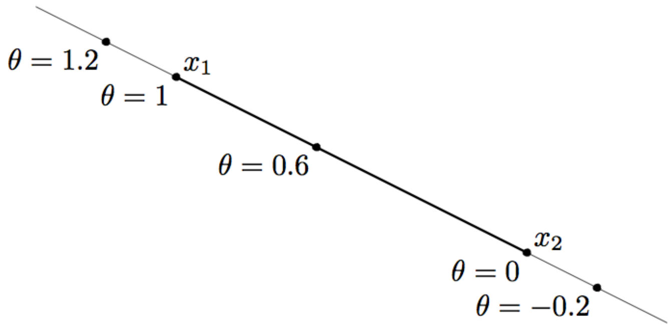


Figure 2.1 The line passing through x_1 and x_2 is described parametrically by $\theta x_1 + (1-\theta)x_2$, where θ varies over $\mathbf{R}$. The line segment between x_1 and x_2, which corresponds to θ between 0 and 1, is shown darker.

- Example: The solution set of linear equations $C = \{\boldsymbol{x} : \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}\}$ is affine. The converse is also true. Every affine set can be expressed as the solution set of a system of linear equations.

- The intersection of an arbitrary collection of convex, affine, or conical sets is convex, affine, conical, respectively.

- Any convex combination $\sum_{i=1}^{m} \alpha_i \boldsymbol{x}_i$ of points from a convex set C belongs to C. By *convex combination* we mean each $\alpha_i \geq 0$ and $\sum_{i=1}^{m} \alpha_i = 1$.

  Similar closure properties apply to convex cones and affine sets if either the restriction $\sum_{i=1}^{m} \alpha_i = 1$ or the constraints $\alpha_i \geq 0$, respectively, are lifted.

- The *convex hull* conv C of a nonempty set C is the smallest convex set containing C. Equivalently, conv C is the set generated by taking all convex combinations $\sum_{i=1}^{m} \alpha_i \boldsymbol{x}_i$ of elements of C.

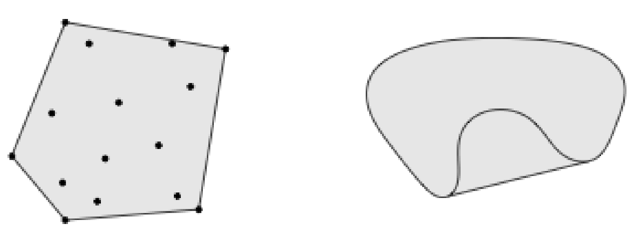


Figure 2.3 The convex hulls of two sets in $\mathbf{R}^2$. *Left.* The convex hull of a set of fifteen points (shown as dots) is the pentagon (shown shaded). *Right.* The convex hull of the kidney shaped set in figure 2.2 is the shaded set.

The *convex conical hull* and *affine hull* of C are generated in a similar manner.

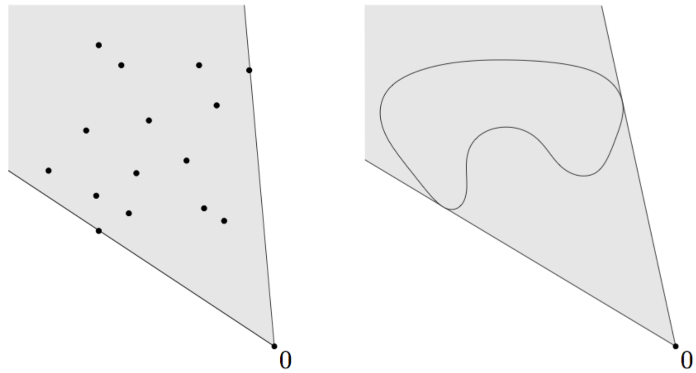


Figure 2.5 The conic hulls (shown shaded) of the two sets of figure 2.3.

What is the affine hull of circle $C = \{\boldsymbol{x} \in \mathbf{R}^2 : \|\boldsymbol{x}\|_2^2 = 1\}$?

- (Carathéodory) For a nonempty set $S \subset \mathbf{R}^n$, every point in conv S can be written as a convex combination of at most $n + 1$ points from S. Furthermore, when S is compact, conv S is also compact.

Convex functions

- A function $f(\boldsymbol{x})$ on $\mathbf{R}^n$ is *convex* if

$$f(\alpha \boldsymbol{x} + (1 - \alpha)\boldsymbol{y}) \leq \alpha f(\boldsymbol{x}) + (1 - \alpha) f(\boldsymbol{y})$$

for all $\boldsymbol{x}, \boldsymbol{y}$ and all $\alpha \in [0, 1]$.

☞ To define a convex function $f(\boldsymbol{x})$ on $\mathbf{R}^n$, it is convenient to allow the value ∞ and disallow the value $-\infty$.

The set $\{\boldsymbol{x} : f(\boldsymbol{x}) < \infty\}$ is a convex set called the *essential domain* of f and written dom f. A convex function is proper if dom $f \neq \emptyset$ and $f(\boldsymbol{x}) > -\infty$ for all $\boldsymbol{x}$.

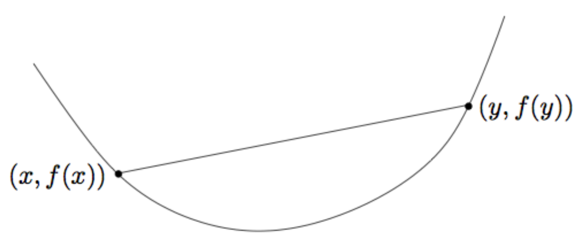


Figure 3.1 Graph of a convex function. The chord (*i.e.*, line segment) between any two points on the graph lies above the graph.

- If the inequality in the definition is strict on dom f when $\alpha > 0$, $\beta > 0$, and $\boldsymbol{x} \neq \boldsymbol{y}$, then the function is said to be *strictly convex.*

- A function $f(\boldsymbol{x})$ is *concave* if its negative $-f(\boldsymbol{x})$ is convex.

  ☞ For concave functions we allow the value $-\infty$ and disallow the value ∞.

- Examples of convex functions on $\mathbf{R}^n$.

  1. *Affine function.* Any affine function $f(\boldsymbol{x}) = \boldsymbol{a}^T\boldsymbol{x} + b$ is both convex and concave.
  2. *Norm.* Any norm (scalar homogeneity, triangle inequality and separates points) on $\mathbf{R}^n$ is convex.
  3. *Indicator function.* The indicator function
  $$\delta_C(\boldsymbol{x}) = \begin{cases} 0 & \boldsymbol{x} \in C \\ \infty & \boldsymbol{x} \notin C \end{cases}$$
  of a nonempty set C is convex if an only if the set itself is convex.
  4. *Quadratic-over-linear function.* The function $f(x, y) = x^2/y$, with dom $f = \mathbf{R} \times \mathbf{R}_{++}$ is convex.

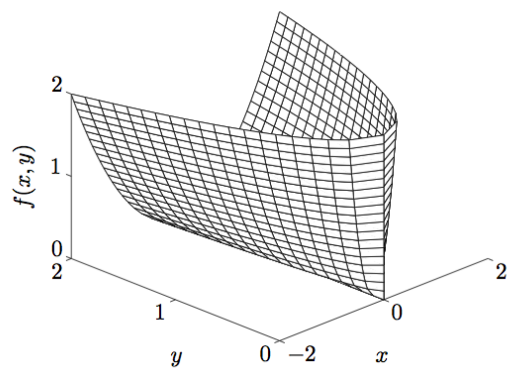


Figure 3.3 Graph of $f(x, y) = x^2/y$.

  5. *log-sum-exp.* The function $f(\boldsymbol{x}) = \ln(e^{x_1} + \cdots + e^{x_n})$ is convex.
  6. *Geometric mean.* The geometric mean $f(\boldsymbol{x}) = \prod_{i=1}^{n} x_i^{1/n}$ is concave.
  7. *Log-det.* The function $f(\boldsymbol{X}) = \ln \det \boldsymbol{X}$ is concave on $\mathbf{S}_{++}^n$. (Two proofs below.)

- Sublevel sets $\{\boldsymbol{x} : f(\boldsymbol{x}) \leq c\}$ of a convex function $f(\boldsymbol{x})$ are convex. If $f(\boldsymbol{x})$ is continuous as well, then all sublevel sets are also closed.

  The converse is not true. For example, the sublevel set $\{\boldsymbol{x} \in \mathbf{R}_+^2 : 1 - x_1x_2 \leq 0\}$ is closed and convex, but the function $1 - x_1x_2$ is not convex on the domain $\mathbf{R}_+^2 = \{\boldsymbol{x} : x_1 \geq 0, x_2 \geq 0\}$.

- (Jenen's inequality) A function $f(\boldsymbol{x})$ is convex if and only if

$$f\left(\sum_{i=1}^{m} \alpha_i \boldsymbol{x}_i\right) \leq \sum_{i=1}^{m} \alpha_i f(\boldsymbol{x}_i),$$

  for all $\alpha_i \geq 0$ and $\sum_{i=1}^{m} \alpha_i = 1$.

  The probabilistic version states $f[E(X)] \leq E[f(X)]$.

- (First order condition, support hyperplane inequality) If $f(\boldsymbol{x})$ is differentiable on the open convex set C, then a necessary and sufficient condition for $f(\boldsymbol{x})$ to be convex is

$$f(\boldsymbol{y}) \geq f(\boldsymbol{x}) + df(\boldsymbol{x})(\boldsymbol{y} - \boldsymbol{x})$$

  for all $\boldsymbol{x}, \boldsymbol{y} \in C$. Furthermore, $f(\boldsymbol{x})$ is strictly convex if and only if strict inequality holds for all $\boldsymbol{y} \neq \boldsymbol{x}$.

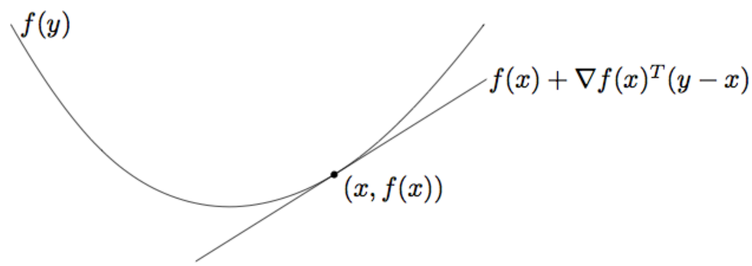


Figure 3.2 If f is convex and differentiable, then $f(x)+\nabla f(x)^T(y-x) \leq f(y)$ for all $x,\ y \in \mathbf{dom}\, f$.

- (Second order condition) Let $f(\boldsymbol{x})$ be a twice differentiable function on the open convex set $C \subset \mathbf{R}^n$. If its Hessian matrix $d^2 f(\boldsymbol{x})$ is psd for all $\boldsymbol{x}$, then $f(\boldsymbol{x})$ is convex. When $d^2 f(\boldsymbol{x})$ is pd for all $\boldsymbol{x}$, $f(\boldsymbol{x})$ is strictly convex.

12 Lecture 12, Feb 25

Announcements

- HW3 due today @ 11:59PM. Commit to your `master` branch and tag.
- HW4 posted (Linear Programming). Due next Friday Mar 6 @ 11:59PM (?).

Last Time

- Convex sets and convex functions.

Today

- Convex functions (cont'd).
- Overview of optimization softwares.

Convex function (cont'd)

- Closure properties of convex functions often offer the easiest way to check convexity.

  1. (Nonnegative weighted sums) If $f(\boldsymbol{x})$ and $g(\boldsymbol{x})$ are convex and α and β are nonnegative constants, then $\alpha f(\boldsymbol{x}) + \beta g(\boldsymbol{x})$ is convex.
  2. (Composition) $h(\boldsymbol{x})$ is convex and increasing, and $g(\boldsymbol{x})$ is convex and finite, then the functional composition $f(\boldsymbol{x}) = h \circ g(\boldsymbol{x})$ is convex.
  3. (Composition with affine mapping) If $f(\boldsymbol{x})$ is convex, then the functional composition $f(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b})$ of $f(\boldsymbol{x})$ with an affine function $\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}$ is convex.
  4. (Pointwise maximum and supremum) If $f_i(\boldsymbol{x})$ is convex for each fixed $i \in I$, then $g(\boldsymbol{x}) = \sup_{i \in I} f_i(\boldsymbol{x})$ is convex provided it is proper. Note the index set I may be infinite.
  5. (Pointwise limit) If $f_m(\boldsymbol{x})$ is a sequence of convex functions, then $\lim_{m \to \infty} f_m(\boldsymbol{x})$ is convex provided it exists and is proper.
  6. (Integration) If $f(\boldsymbol{x}, \boldsymbol{y})$ is convex in $\boldsymbol{x}$ for each fixed $\boldsymbol{y}$ and μ is a measure, then the integral $g(\boldsymbol{x}) = \int f(\boldsymbol{x}, \boldsymbol{y}) d\mu(\boldsymbol{y})$ is convex provided it is proper.

     ☞ It is generalization of the nonnegative weighted sum rule.
  7. (Minimum) If $f(\boldsymbol{x}, \boldsymbol{y})$ is jointly convex in $(\boldsymbol{x}, \boldsymbol{y})$, then $g(\boldsymbol{x}) = \inf_{y \in C} f(\boldsymbol{x}, \boldsymbol{y})$ is convex provided it is proper and C is convex.

☞ Product of two convex functions is not necessarily convex. Counter example: $x^3 = xx^2$. However if both functions are convex, nondecreasing (or nonincreasing), and positive functions on an interval, then the product is convex.

- Example: The function $f(\boldsymbol{x}) = x_{[1]} + \cdots + x_{[k]}$, the sum of the k largest components of $\boldsymbol{x} \in \mathbf{R}^n$, is convex.

  ☞ This is hint for HW3 Q3.

  Proof. Write the function f as

  $$f(\boldsymbol{x}) = \max\{x_{i_1} + \cdots + x_{i_k} : 1 \leq i_1 < i_2 < \cdots i_k \leq n\},$$

  i.e., the maximum of all possible sums of k different components of $\boldsymbol{x}$. Since it is the pointwise maximum of $\binom{n}{k}$ linear functions, it is convex. □

- Example: Dominant eigenvalue of a symmetric matrix

  $$\lambda_{\max}(\boldsymbol{M}) = \max_{\|\boldsymbol{x}\|=1} \boldsymbol{x}^T \boldsymbol{M} \boldsymbol{x}$$

  is convex in $\boldsymbol{M}$ since it is pointwise maximum of linear functions. Similarly the minimum eigenvalue $\lambda_{\min}(\boldsymbol{M})$ is concave in $\boldsymbol{M}$.

  ☞ Sum of k largest eigenvalues is convex on $\mathbf{S}^n$.

- More on composition rule. Scalar composition $f = h \circ g$, where $h : \mathbf{R} \mapsto \mathbf{R}$ and $g : \mathbf{R} \mapsto \mathbf{R}$:

  - f is convex if h is convex and nondecreasing, and g is convex.
  - f is convex if h is convex and nonincreasing, and g is concave.
  - f is concave if h is concave and nondecreasing, and g is concave.
  - f is concave if h is concave and nonincreasing, and g is convex.

  ☞ Remember by $f''(x) = h''(g(x))g'(x)^2 + h'(g(x))g''(x)$. But same results apply to non-differential functions as well.

  Vector composition $f(\boldsymbol{x}) = h \circ g(\boldsymbol{x}) = h(g_1(\boldsymbol{x}), \ldots, g_k(\boldsymbol{x}))$, where $g_i : \mathbf{R}^n \mapsto \mathbf{R}$ and $h : \mathbf{R}^k \mapsto \mathbf{R}$.

  - f is convex if h is convex, h is nondecreasing in each argument, and g_i are convex.
  - f is convex if h is convex, h is nonincreasing in each argument, and g_i are concave.
  - f is concave if h is concave, h is nondecreasing in each argument, and g_i are concave.

☞ Remember by $d^2 f(\boldsymbol{x}) = Dg(\boldsymbol{x})^T d^2 h(g(\boldsymbol{x})) Dg(\boldsymbol{x}) + (Dh(g(\boldsymbol{x})) \otimes \boldsymbol{I}_n) d^2 g(\boldsymbol{x})$. But same results apply to non-differential functions as well.

- The *epigraph* of a function $f(\boldsymbol{x})$ is the set

$$\text{epi} f = \{(\boldsymbol{x}, r) : f(\boldsymbol{x}) \leq r\}.$$

- A function $f(\boldsymbol{x})$ is convex if and only if its epigraph is a convex set.

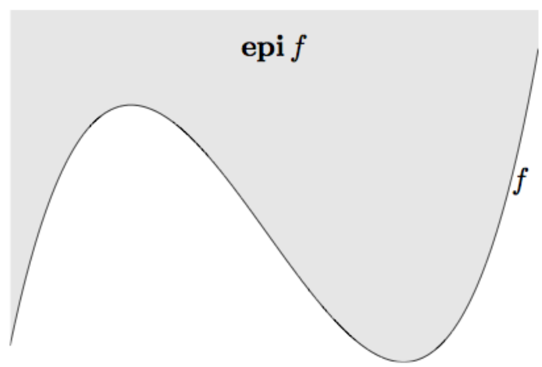


Figure 3.5 Epigraph of a function f, shown shaded. The lower boundary, shown darker, is the graph of f.

- Example: The *matrix fractional function*

$$f(\boldsymbol{x}, \boldsymbol{Y}) = \boldsymbol{x}^T \boldsymbol{Y}^{-1} \boldsymbol{x}$$

is convex on domain $\mathbf{R}^n \times \mathbf{S}^n_{++}$. This generalizes the convexity of quadratic-over-linear function $f(x, y) = x^2/y$ on $\mathbf{R} \times \mathbf{R}_{++}$.

☞ This is hint for HW3 Q5 and Q10.

Proof (by epigraph). The epigraph of matrix fractional function is

$$\begin{aligned} \text{epi} f &= \{(\boldsymbol{x}, \boldsymbol{Y}, t) : \boldsymbol{Y} \succ \mathbf{0}, \boldsymbol{x}^T \boldsymbol{Y}^{-1} \boldsymbol{x} \leq t\} \\ &= \left\{ (\boldsymbol{x}, \boldsymbol{Y}, t) : \begin{pmatrix} \boldsymbol{Y} & \boldsymbol{x} \\ \boldsymbol{x}^T & t \end{pmatrix} \succeq \mathbf{0}, \boldsymbol{Y} \succ \mathbf{0} \right\}, \end{aligned}$$

which is convex. The second equality is from the linear algebra fact that a block matrix

$$\begin{pmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{B}^T & \boldsymbol{C} \end{pmatrix}$$

is psd if and only if $\boldsymbol{A}$ is psd, the Schur complement $\boldsymbol{C} - \boldsymbol{B}^T \boldsymbol{A}^{-1} \boldsymbol{B}$ is psd, and $(\boldsymbol{I} - \boldsymbol{A}\boldsymbol{A}^-)\boldsymbol{B} = \mathbf{0}$ ($\boldsymbol{B} \in \mathcal{C}(\boldsymbol{A})$). □

☞ Same argument yields joint convexity of the matrix function $f(\boldsymbol{X},\boldsymbol{Y}) = \boldsymbol{X}^T\boldsymbol{Y}^{-1}\boldsymbol{X}$ on $\mathbf{R}^{m\times n} \times \mathbf{S}^n_{++}$.

☞ (Singular case) The result can be further extended to show that the function

$$f(\boldsymbol{X},\boldsymbol{Y}) = \begin{cases} \frac{1}{2}\boldsymbol{u}^T\boldsymbol{X}^T\boldsymbol{Y}^+\boldsymbol{X}\boldsymbol{u} & \boldsymbol{X}\boldsymbol{u} \in \mathcal{C}(\boldsymbol{Y}) \\ \infty & \boldsymbol{X}\boldsymbol{u} \notin \mathcal{C}(\boldsymbol{Y}) \end{cases}$$

on $\mathbf{R}^{m\times n} \times \mathbf{S}^n_{+}$ is jointly convex in $\boldsymbol{X}$ and $\boldsymbol{Y}$ for any choice of $\boldsymbol{u}$.

- (Line theorem) A function is convex if and only if it is convex when restricted to a line that intersects its domain. That is $f(\boldsymbol{x})$ is convex if and only if for any $\boldsymbol{x} \in \text{dom}f$ and $\boldsymbol{v} \in \mathbf{R}^n$, then function

$$g(t) = f(\boldsymbol{x} + t\boldsymbol{v})$$

is convex on dom $g = \{t : \boldsymbol{x} + t\boldsymbol{v} \in \text{dom}f\}$.

☞ Not sure if a function is convex? Generate a bunch of lines through the domain and plot. If any of them are not convex, the function is not convex.

- Example: Concavity of $\ln\det\boldsymbol{\Omega}$ on $\mathbf{S}^n_{++}$. This generalizes the concavity of $\ln x$ for $x > 0$.

☞ This is hint for HW3 Q10.

Proof. Let $\boldsymbol{X} \in \mathbf{S}^n_{++}$ and $\boldsymbol{V} \in \mathbf{S}^n$. Then

$$\begin{aligned} g(t) &= \ln\det(\boldsymbol{X} + t\boldsymbol{V}) \\ &= \ln\det\boldsymbol{X}^{1/2}(\boldsymbol{I} + t\boldsymbol{X}^{-1/2}\boldsymbol{V}\boldsymbol{X}^{-1/2})\boldsymbol{X}^{1/2} \\ &= \ln\det\boldsymbol{X} + \ln\det(\boldsymbol{I} + t\boldsymbol{X}^{-1/2}\boldsymbol{V}\boldsymbol{X}^{-1/2}) \\ &= \ln\det\boldsymbol{X} + \sum_{i=1}^n \ln(1 + \lambda_i t), \end{aligned}$$

where λ_i are eigenvalues of $\boldsymbol{X}^{-1/2}\boldsymbol{V}\boldsymbol{X}^{-1/2}$. $g(t)$ is concave in t thus $\ln\det$ function is concave too. □

Log-convexity

- A positive function $f(\boldsymbol{x})$ is said to be *log-convex* if $\ln f(\boldsymbol{x})$ is convex.

- A log-convex function is convex. Why?

- Log-convex functions enjoy the same closure properties 1 through 7. In part 2 (composition rule), g is convex and h is log-convex.

  In addition the collection of log-convex functions is closed under the formation of products and powers.

  ☞ Not all rules apply to log-concave functions! For instance, nonnegative sum of log-concave functions is not necessarily log-concave.

- Examples:

  1. The beta function
  $$B(x, y) = \int_0^1 u^{x-1}(1-u)^{y-1}\, du$$
  is log-convex. Why?

  2. The gamma function
  $$\Gamma(t) = \int_0^\infty x^{t-1} e^{-x}\, dx$$
  is log-convex. Why?

  3. The moment function
  $$M(x) = \int_0^\infty u^x f(u)\, du,$$
  where f is density of a nonnegative random variable, is log-convex. Why?

  4. The Riemmann zeta function
  $$\zeta(s) = \frac{1}{\Gamma(s)} \int_0^\infty \frac{x^{s-1}}{e^x - 1}\, dx$$
  is log-convex. Why?

  5. The Normal cdf
  $$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^x e^{-u^2/2}\, du$$
  is log-concave. See (Boyd and Vandenberghe, 2004, Exercise 3.54).

- Example: Concavity of $\ln \det \mathbf{\Omega}$ on $\mathbf{S}_{++}^n$. This generalizes the concavity of $\ln x$ for $x > 0$.

  ☞ This is hint for HW3 Q10.

Proof by log-concavity. Integration of the multivariate Gaussian density with pd covariance $\boldsymbol{\Sigma}$

$$f(\boldsymbol{x}) = \frac{1}{(2\pi)^{n/2}} |\det \boldsymbol{\Sigma}|^{-1/2} e^{-\boldsymbol{x}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{x}/2}$$

produces

$$|\det \boldsymbol{\Sigma}|^{1/2} = \frac{1}{(2\pi)^{n/2}} \int e^{-\boldsymbol{x}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{x}/2} \, d\boldsymbol{x}.$$

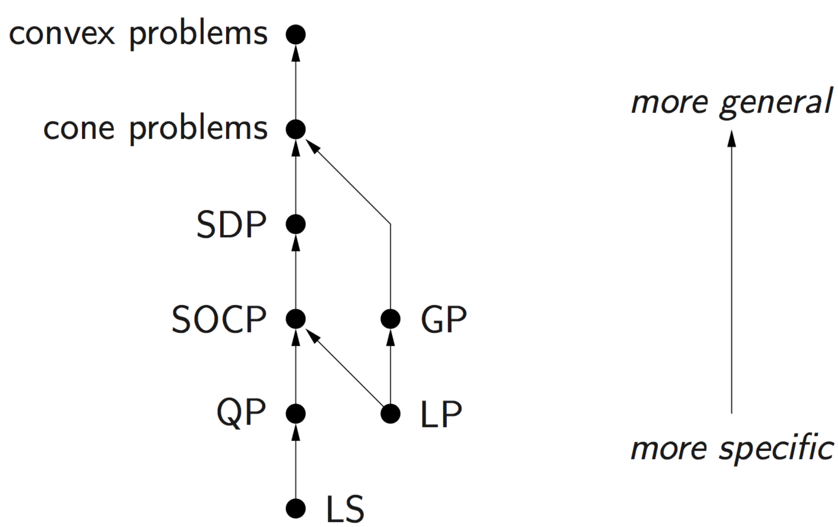
This identity can be restated in terms of the precision matrix $\boldsymbol{\Omega} = \boldsymbol{\Sigma}^{-1}$ as

$$\ln \det \boldsymbol{\Omega} = n \ln(2\pi) - 2 \ln \int e^{-\boldsymbol{x}^T \boldsymbol{\Omega} \boldsymbol{x}/2} \, d\boldsymbol{x}.$$

The integral on the right is log-convex. Why? Is the integral log-concave? Thus $\ln \det \boldsymbol{\Omega}$ is concave. □

Hierarchy of convex optimization problems

In ST758, we spent a fair amount of time on the LS (least squares) problem. In this course, we study LP (linear programming), QP (quadratic programming), SOCP (second-order cone programming), SDP (semidefinite programming), and GP (geometric programming), with an emphasis on statistical applications and software implementation.



Optimization softwares

Like computer languages, getting familiar with *good* optimization softwares broadens the scope and scale of problems we are able to solve in statistics.

- Following table lists some of the best convex optimization softwares. Use of Gurobi and/or Mosek is highly recommended.

  ☞ Gurobi is named after its founders: Zonghao **Gu**, Edward **Ro**thberg, and Robert Bix**by**. Bixby founded the CPLEX at IBM, while Rothberg and Gu led the CPLEX development team for nearly a decade.

- Difference between *modeling tool* and *solvers*.

  - Solvers (`Gurobi`, `Mosek`, ...) are concrete software implementation of optimization algorithms.
  - Modeling tools such as `cvx` and `Convex.jl` (Julia analog of cvx) implement the disciplined convex programming (DCP) paradigm proposed by Grant and Boyd (2008). `http://stanford.edu/~boyd/papers/disc_cvx_prog.html`. DCP prescribes a set of simple rules from which users can construct convex optimization problems easily.

    Modeling tools usually have the capability to use a variety of solvers. But modeling tools are solver agnostic so users do not have to worry about specific solver interface.

	LP	MILP	SOCP	MISOCP	SDP	GP	NLP	MINLP	R	Matlab	Julia	Python	Cost
JuMP.jl	✓	✓	✓	✓			✓	✓			✓		O
Convex.jl	✓	✓	✓	✓	✓						✓		O
cvx	✓	✓	✓	✓	✓	✓				✓		✓	A
Gurobi	✓	✓	✓	✓					✓	✓	✓	✓	A
Mosek	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	A
CPLEX	✓	✓	✓						?	✓	✓	✓	A
SCS	✓		✓		✓					✓	✓	✓	O
SeDuMi	✓		✓		✓	?				✓			O
SDPT3	✓		✓		✓	?				✓			O
KNITRO	✓	✓					✓	✓		✓	✓	✓	$

LP = Linear Programming, MILP = Mixed Integer LP, SOCP = Second-order cone programming (includes QP, QCQP), MISOCP = Mixed Integer SOCP, SDP = Semidefinite Programming, GP = Geometric Programming, NLP = (constrained) Nonlinear Programming (includes general QP, QCQP), MINLP = Mixed Integer NLP, O = Open source, A = Free academic license

Set up Gurobi on the teaching server

1. Gurobi 6.0 has been installed on the teaching server at
   `/use/local/gurobi600`
   But you have to obtain a license (free) first in order to use it.

2. Register for an account on `http://www.gurobi.com/account`. Be sure to use your `edu` email and check `Academic` as your account type.

3. After confirmation of your academic account, log into your account and request a free academic license at `http://www.gurobi.com/download/licenses/free-academic`.

4. Run `grbgetkey` command on the teaching server and enter the key you obtained in step 3. Place the file at `/home/USERID/.gurobi/`

5. Now you should be able to use Gurobi in Matlab, R, and Julia.

Set up Mosek on the teaching server

1. Mosek 7 has been installed on the teaching server at
   `/usr/local/mosek/7/`
   License file is already put into your home directory.
   `/home/unityID/mosek/mosek.lic`

2. You should be able to use Mosek in Matlab or R already.

Set up CVX on the teaching server

1. CVX v2.1 has been installed on the teaching server at
   `/use/local/cvx`
   But you have to obtain a license (free) first in order to use it.

2. Request a free academic (professional) license at `http://cvxr.com/cvx/academic/` using your `edu` email. Your will receive the license file `license.dat` by email. Place the license file at `/home/USERID/.cvx/`

3. Within Matlab, type
   `cvx_setup /home/hzhou3/.cvx/cvx_license.dat`

4. Now you should be able to use CVX in Matlab.

   ☞ The *standard license* comes with free solvers `SeDuMi` and `SDPT3`. The *Academic license* also bundles with `Gurobi` and `Mosek`.

13 Lecture 13, Mar 2

Announcements

- HW4 (LP) deadline extended to Mon, Mar 16 @ 11:59PM.
- HW5 (QP, SOCP) posted. Due Fri, Mar 20 @ 11:59PM. `http://hua-zhou.github.io/teaching/st790-2015spr/ST790-2015-HW5.pdf`

Last Time

- Convex and log-convex functions.
- Overview of optimization softwares.

Today

- LP (linear programming).

Linear programming (LP)

- A general linear program takes the form

$$\begin{array}{ll} \text{minimize} & \boldsymbol{c}^T \boldsymbol{x} \\ \text{subject to} & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \\ & \boldsymbol{G}\boldsymbol{x} \preceq \boldsymbol{h}. \end{array}$$

  Linear program is a convex optimization problem, why?

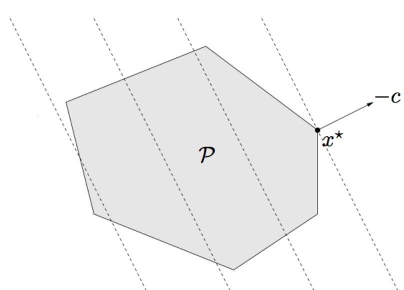


Figure 4.4 Geometric interpretation of an LP. The feasible set $\mathcal{P}$, which is a polyhedron, is shaded. The objective $c^T x$ is linear, so its level curves are hyperplanes orthogonal to c (shown as dashed lines). The point $x^\star$ is optimal; it is the point in $\mathcal{P}$ as far as possible in the direction $-c$.

- The *standard form* of an LP is

$$\begin{array}{ll} \text{minimize} & \boldsymbol{c}^T\boldsymbol{x} \\ \text{subject to} & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \\ & \boldsymbol{x} \succeq \boldsymbol{0}. \end{array}$$

  To transform a general linear program into the standard form, we introduce the *slack variables* $\boldsymbol{s} \succeq \boldsymbol{0}$ such that $\boldsymbol{G}\boldsymbol{x} + \boldsymbol{s} = \boldsymbol{h}$. Then we write $\boldsymbol{x} = \boldsymbol{x}^+ - \boldsymbol{x}^-$, where $\boldsymbol{x}^+ \succeq \boldsymbol{0}$ and $\boldsymbol{x}^- \succeq \boldsymbol{0}$. This yields the problem

$$\begin{array}{ll} \text{minimize} & \boldsymbol{c}^T(\boldsymbol{x}^+ - \boldsymbol{x}^-) \\ \text{subject to} & \boldsymbol{A}(\boldsymbol{x}^+ - \boldsymbol{x}^-) = \boldsymbol{b} \\ & \boldsymbol{G}(\boldsymbol{x}^+ - \boldsymbol{x}^-) + \boldsymbol{s} = \boldsymbol{h} \\ & \boldsymbol{x}^+ \succeq \boldsymbol{0}, \boldsymbol{x}^- \succeq \boldsymbol{0}, \boldsymbol{s} \succeq \boldsymbol{0} \end{array}$$

  in $\boldsymbol{x}^+$, $\boldsymbol{x}^-$, and $\boldsymbol{s}$.

  ☞ Slack variables are often used to transform a complicated inequality constraint to simple non-negativity constraints.

- The *inequality form* of an LP is

$$\begin{array}{ll} \text{minimize} & \boldsymbol{c}^T\boldsymbol{x} \\ \text{subject to} & \boldsymbol{G}\boldsymbol{x} \preceq \boldsymbol{h}. \end{array}$$

  ☞ Some softwares, e.g., `solveLP` in `R`, require an LP be written in either standard or inequality form. However a good software should do this for you!

- A *piecewise-linear minimization* problem

$$\text{minimize} \quad \max_{i=1,\ldots,m} (\boldsymbol{a}_i^T\boldsymbol{x} + b_i)$$

  can be transformed to an LP

$$\begin{array}{ll} \text{minimize} & t \\ \text{subject to} & \boldsymbol{a}_i^T\boldsymbol{x} + b_i \le t, \quad i = 1, \ldots, m, \end{array}$$

  in $\boldsymbol{x}$ and t. Apparently

$$\text{minimize} \max_{i=1,\ldots,m} |\boldsymbol{a}_i^T\boldsymbol{x} + b_i|$$

  and

$$\text{minimize} \max_{i=1,\ldots,m} (\boldsymbol{a}_i^T\boldsymbol{x} + b_i)_+$$

are also LP.

☞ Any *convex optimization problem*

$$\begin{aligned} &\text{minimize} && f_0(\boldsymbol{x}) \\ &\text{subject to} && f_i(\boldsymbol{x}) \le 0, \quad i = 1, \ldots, m \\ &&& \boldsymbol{a}_i^T \boldsymbol{x} = b_i, \quad i = 1, \ldots, p, \end{aligned}$$

where $f_0, \ldots, f_m$ are convex functions, can be transformed to the *epigraph form*

$$\begin{aligned} &\text{minimize} && t \\ &\text{subject to} && f_0(\boldsymbol{x}) - t \le 0 \\ &&& f_i(\boldsymbol{x}) \le 0, \quad i = 1, \ldots, m \\ &&& \boldsymbol{a}_i^T \boldsymbol{x} = b_i, \quad i = 1, \ldots, p \end{aligned}$$

in variables $\boldsymbol{x}$ and t. That is why people often say linear program is universal.

- The *linear fractional programming*

$$\begin{aligned} &\text{minimize} && \frac{\boldsymbol{c}^T \boldsymbol{x} + d}{\boldsymbol{e}^T \boldsymbol{x} + f} \\ &\text{subject to} && \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \\ &&& \boldsymbol{G}\boldsymbol{x} \preceq \boldsymbol{h} \\ &&& \boldsymbol{e}^T \boldsymbol{x} + f > 0 \end{aligned}$$

can be transformed to an LP

$$\begin{aligned} &\text{minimize} && \boldsymbol{c}^T \boldsymbol{y} + dz \\ &\text{subject to} && \boldsymbol{G}\boldsymbol{y} - z\boldsymbol{h} \preceq \boldsymbol{0} \\ &&& \boldsymbol{A}\boldsymbol{y} - z\boldsymbol{b} = \boldsymbol{0} \\ &&& \boldsymbol{e}^T \boldsymbol{y} + fz = 1 \\ &&& z \ge 0 \end{aligned}$$

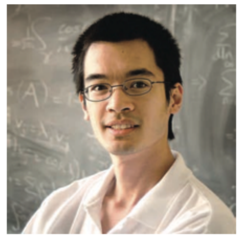
in $\boldsymbol{y}$ and z, via transformation of variables

$$\boldsymbol{y} = \frac{\boldsymbol{x}}{\boldsymbol{e}^T \boldsymbol{x} + f}, \quad z = \frac{d}{\boldsymbol{e}^T \boldsymbol{x} + f}.$$

See Boyd and Vandenberghe (2004, Section 4.3.2) for proof.

- Example. Compressed sensing (Candès and Tao, 2006; Donoho, 2006) tries to address a fundamental question: how to compress and transmit a complex signal (e.g., musical clips, mega-pixel images), which can be decoded to recover the original signal?

Emmanuel Candes. *(Photo courtesy of Emmanuel Candes.)*



Terence Tao. *(Photo courtesy of Reed Hutchinson/UCLA.)*

Suppose a signal $\boldsymbol{x} \in \mathbf{R}^n$ is sparse with s non-zeros. We under-sample the signal by multiplying a measurement matrix $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$, where $\boldsymbol{A} \in \mathbf{R}^{m \times n}$ has iid normal entries. Candès et al. (2006) show that the solution to

$$\begin{aligned} &\text{minimize} \quad \|\boldsymbol{x}\|_1 \\ &\text{subject to} \quad \boldsymbol{A}\boldsymbol{x} = \boldsymbol{y} \end{aligned}$$

exactly recovers the true signal under certain conditions on $\boldsymbol{A}$ when $n \gg s$ and $m \approx s \ln(n/s)$. Why sparsity is a reasonable assumption? *Virtually all real-world images have low information content.*

The ℓ_1 minimization problem apparently is an LP, by writing $\boldsymbol{x} = \boldsymbol{x}^+ - \boldsymbol{x}^-$,

$$\begin{aligned} \text{minimize} \quad & \mathbf{1}^T(\boldsymbol{x}^+ + \boldsymbol{x}^-) \\ \text{subject to} \quad & \boldsymbol{A}(\boldsymbol{x}^+ - \boldsymbol{x}^-) = \boldsymbol{y} \\ & \boldsymbol{x}^+ \succeq \mathbf{0}, \boldsymbol{x}^- \succeq \mathbf{0}. \end{aligned}$$

Let's work on a numerical example. `http://hua-zhou.github.io/teaching/st790-2015spr/demo_cs.html`

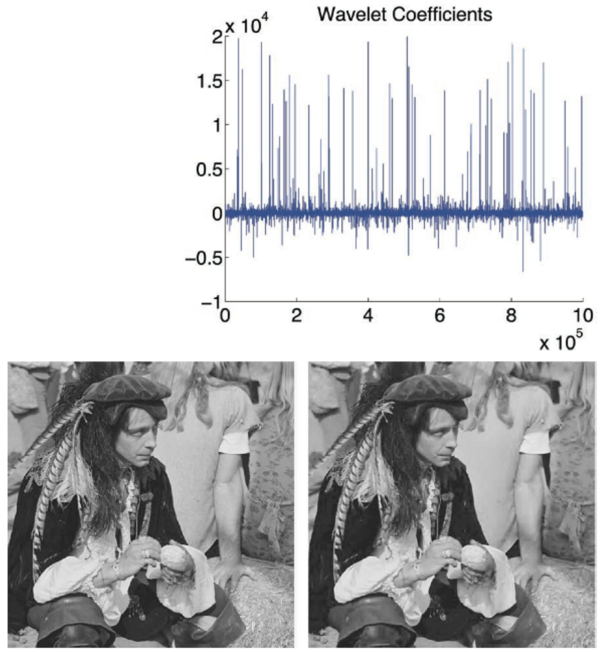
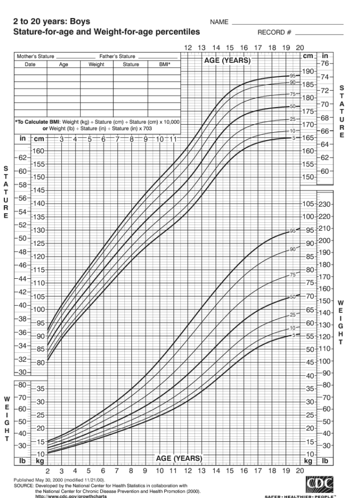
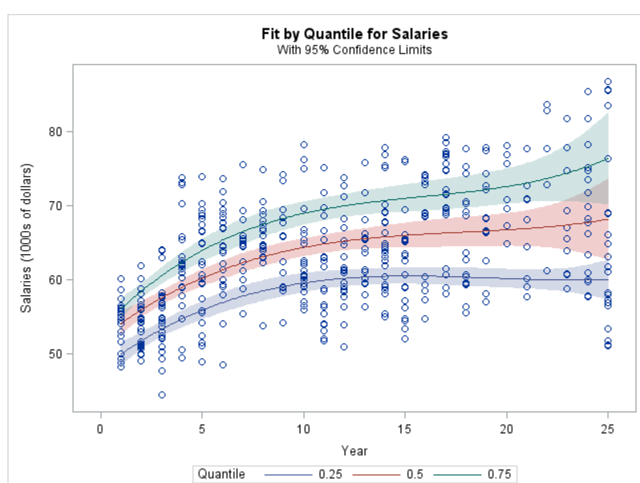


Figure 1. *Normal scenes from everyday life are compressible with respect to a basis of wavelets. (left) A test image. (top) One standard compression procedure is to represent the image as a sum of wavelets. Here, the coefficients of the wavelets are plotted, with large coefficients identifying wavelets that make a significant contribution to the image (such as identifying an edge or a texture). (right) When the wavelets with small coefficients are discarded and the image is reconstructed from only the remaining wavelets, it is nearly indistinguishable from the original. (Photos and figure courtesy of Emmanuel Candes.)*

- Example. Quantile regression (HW4). In linear regression, we model the mean of response variable as a function of covariates. In many situations, the error variance is not constant, the distribution of y may be asymmetric, or we simply care about the quantile(s) of response variable. Quantile regression offers a better modeling tool in these applications.





In τ-quantile regression, we minimize the loss function

$$f(\boldsymbol{\beta}) = \sum_{i=1}^{n} \rho_\tau(y_i - \boldsymbol{x}_i^T \boldsymbol{\beta}),$$

where $\rho_\tau(z) = z(\tau - 1_{\{z<0\}})$. Writing $\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta} = \boldsymbol{r}^+ - \boldsymbol{r}^-$, this is equivalent to the LP

$$\begin{aligned} \text{minimize} \quad & \tau \mathbf{1}^T \boldsymbol{r}^+ + (1-\tau)\mathbf{1}^T \boldsymbol{r}^- \\ \text{subject to} \quad & \boldsymbol{r}^+ - \boldsymbol{r}^- = \boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta} \\ & \boldsymbol{r}^+ \succeq \mathbf{0}, \boldsymbol{r}^- \succeq \mathbf{0} \end{aligned}$$

in $\boldsymbol{r}^+$, $\boldsymbol{r}^-$, and $\boldsymbol{\beta}$.

- Example: ℓ_1 regression (HW4). A popular method in robust statistics is the median absolute deviation (MAD) regression that minimizes the ℓ_1 norm of the residual vector $\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_1$. This apparently is equivalent to the LP

  $$\begin{aligned} \text{minimize} \quad & \mathbf{1}^T (\boldsymbol{r}^+ + \boldsymbol{r}^-) \\ \text{subject to} \quad & \boldsymbol{r}^+ - \boldsymbol{r}^- = \boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta} \\ & \boldsymbol{r}^+ \succeq \mathbf{0}, \boldsymbol{r}^- \succeq \mathbf{0} \end{aligned}$$

  in $\boldsymbol{r}^+$, $\boldsymbol{r}^-$, and $\boldsymbol{\beta}$.

  ☞ ℓ_1 regression = MAD = 1/2-quantile regression.

- Example: ℓ_∞ regression (Chebychev approximation). Minimizing the worst possible residual $\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_\infty$ is equivalent to the LP

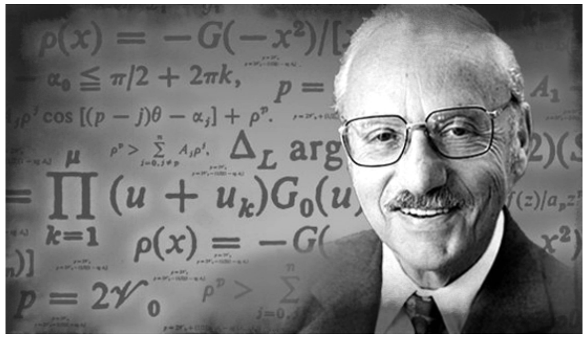
  $$\begin{aligned} \text{minimize} \quad & t \\ \text{subject to} \quad & -t \le y_i - \boldsymbol{x}_i^T \boldsymbol{\beta} \le t, \quad i = 1, \ldots, n \end{aligned}$$

  in variables $\boldsymbol{\beta}$ and t.

- Example: Dantzig selector (HW4). Candès and Tao (2007) propose a variable selection method called the Dantzig selector that solves

  $$\begin{aligned} \text{minimize} \quad & \|\boldsymbol{X}^T (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})\|_\infty \\ \text{subject to} \quad & \sum_{j=2}^{p} |\boldsymbol{\beta}_j| \le t, \end{aligned}$$

  which can be transformed to an LP. Indeed they name the method after George Dantzig, who invented the simplex method for efficiently solving LP in 50s.



☞ Apparently any loss/penalty or loss/constraint combinations of form

$$\{\ell_1, \ell_\infty, \text{quantile}\} \times \{\ell_1, \ell_\infty, \text{quantile}\},$$

possibly with affine (equality and/or inequality) constraints, can be formulated as an LP.

- Example: 1-norm SVM (HW4). In two-class classification problems, we are given training data $(\boldsymbol{x}_i, y_i)$, $i = 1, \ldots, n$, where $\boldsymbol{x}_i \in \mathbf{R}^p$ are feature vectors and $y_i \in \{-1, 1\}$ are class labels. Zhu et al. (2004) propose the 1-norm support vector machine (svm) that achieves the dual purpose of classification and feature selection. Denote the solution of the optimization problem

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^{n} \left[1 - y_i \left(\beta_0 + \sum_{j=1}^{p} x_{ij}\beta_j \right) \right]_+ \\ \text{subject to} & \|\boldsymbol{\beta}\|_1 = \sum_{j=1}^{p} |\beta_j| \le t \end{array}$$

by $\hat{\beta}_0(t)$ and $\hat{\boldsymbol{\beta}}(t)$. 1-norm svm classifies a future feature vector $\boldsymbol{x}$ by the sign of fitted model

$$\hat{f}(\boldsymbol{x}) = \hat{\beta}_0 + \boldsymbol{x}^T \hat{\boldsymbol{\beta}}.$$

- Many more applications: Airport scheduling (Copenhagen airport uses Gurobi), airline flight scheduling, NFL scheduling, `match.com`, LaTeX, ...

14 Lecture 14, Mar 4

Announcements

- HW4 (LP) deadline extended to Mon, Mar 16 @ 11:59PM.
- HW5 (QP, SOCP) posted. Due Fri, Mar 20 @ 11:59PM. `http://hua-zhou.github.io/teaching/st790-2015spr/ST790-2015-HW5.pdf`

Last Time

- LP (linear programming).

Today

- QP (quadratic programming).
- SOCP (second order cone programming).

More LP

- In the worst k error regression (HW3), we minimize $\sum_{i=1}^{k} |r|_{(i)}$ where $|r|_{(1)} \geq |r|_{(2)} \geq \cdots \geq |r|_{(n)}$ are order statistics of the absolute values of residuals $|r_i| = |y_i - \boldsymbol{x}_i^T \boldsymbol{\beta}|$. This can be solved by the LP

$$\begin{array}{rl} \text{minimize} & kt + \mathbf{1}^T \boldsymbol{z} \\ \text{subject to} & -t\mathbf{1} - \boldsymbol{z} \preceq \boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta} \preceq t\mathbf{1} + \boldsymbol{z} \\ & \boldsymbol{z} \succeq \mathbf{0} \end{array}$$

  in variables $\boldsymbol{\beta} \in \mathbf{R}^p$, $t \in \mathbf{R}$, and $\boldsymbol{z} \in \mathbf{R}^n$.

- Our catalogue of linear parts: composition of ℓ_1 (absolute values), ℓ_∞ (max), check loss (quantile), hinge loss (svm), sum of k largest component, ... with affine functions.

Quadratic programming (QP)

- A *quadratic program* (QP) has quadratic objective function and affine constraint functions

$$\begin{array}{rl} \text{minimize} & (1/2)\boldsymbol{x}^T \boldsymbol{P} \boldsymbol{x} + \boldsymbol{q}^T \boldsymbol{x} + r \\ \text{subject to} & \boldsymbol{G}\boldsymbol{x} \preceq \boldsymbol{h} \\ & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \end{array}$$

where we require $\boldsymbol{P} \in \mathbf{S}_+^n$ (why?).

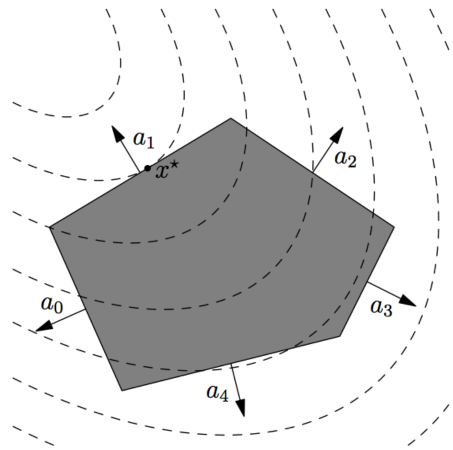


Figure 5.1: Geometric interpretation of quadratic optimization. At the optimal point $x^\star$ the hyperplane $\{x \mid a_1^T x = b\}$ is tangential to an ellipsoidal level curve.

- Example. The *least squares* problem minimizes $\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2$, which obviously is a QP.

- Example. Least squares with linear constraints. For example, *nonnegative least squares* (NNLS)

$$\begin{array}{ll} \text{minimize} & (1/2)\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 \\ \text{subject to} & \boldsymbol{\beta} \succeq \mathbf{0}. \end{array}$$

☞ In NNMF (nonnegative matrix factorization), the objective $\|\boldsymbol{X} - \boldsymbol{V}\boldsymbol{W}\|_F^2$ can be minimized by alternating NNLS.

- Example. Lasso regression (Tibshirani, 1996; Donoho and Johnstone, 1994) minimizes the least squares loss with ℓ_1 (lasso) penalty

$$\text{minimize} \quad \frac{1}{2}\|\boldsymbol{y} - \beta_0\mathbf{1} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_1,$$

where $\lambda \geq 0$ is a tuning parameter. Writing $\boldsymbol{\beta} = \boldsymbol{\beta}^+ - \boldsymbol{\beta}^-$, the equivalent QP is

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}(\boldsymbol{\beta}^+ - \boldsymbol{\beta}^-)^T\boldsymbol{X}^T\left(\boldsymbol{I} - \frac{\mathbf{1}\mathbf{1}^T}{n}\right)\boldsymbol{X}(\boldsymbol{\beta}^+ - \boldsymbol{\beta}^-) + \\ & \boldsymbol{y}^T\left(\boldsymbol{I} - \frac{\mathbf{1}\mathbf{1}^T}{n}\right)\boldsymbol{X}(\boldsymbol{\beta}^+ - \boldsymbol{\beta}^-) + \lambda\mathbf{1}^T(\boldsymbol{\beta}^+ + \boldsymbol{\beta}^-) \\ \text{subject to} & \boldsymbol{\beta}^+ \succeq \mathbf{0}, \, \boldsymbol{\beta}^- \succeq \mathbf{0} \end{array}$$

in $\boldsymbol{\beta}^+$ and $\boldsymbol{\beta}^-$.

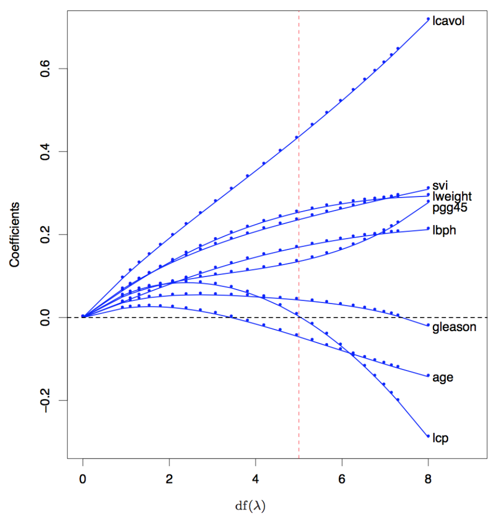


FIGURE 3.8. *Profiles of ridge coefficients for the prostate cancer example, as the tuning parameter λ is varied. Coefficients are plotted versus $\mathrm{df}(\lambda)$, the effective degrees of freedom. A vertical line is drawn at $\mathrm{df} = 5.0$, the value chosen by cross-validation.*

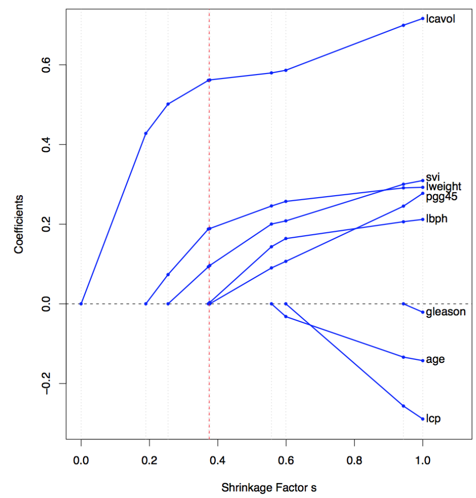


FIGURE 3.10. *Profiles of lasso coefficients, as the tuning parameter t is varied. Coefficients are plotted versus $s = t/\sum_1^p |\hat{\beta}_j|$. A vertical line is drawn at $s = 0.36$, the value chosen by cross-validation. Compare Figure 3.8 on page 65; the lasso profiles hit zero, while those for ridge do not. The profiles are piece-wise linear, and so are computed only at the points displayed; see Section 3.4.4 for details.*

- Example: Elastic net (Zou and Hastie, 2005)

$$\text{minimize} \quad \frac{1}{2}\|\boldsymbol{y} - \beta_0\mathbf{1} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 + \lambda(\alpha\|\boldsymbol{\beta}\|_1 + (1-\alpha)\|\boldsymbol{\beta}\|_2^2),$$

  where $\lambda \geq 0$ and $\alpha \in [0, 1]$ are tuning parameters.

- Example: Generalized lasso

$$\text{minimize} \quad \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{D}\boldsymbol{\beta}\|_1,$$

  where $\lambda \geq 0$ is a tuning parameter $\boldsymbol{D}$ is a fixed regularization matrix. This generates numerous applications (Tibshirani and Taylor, 2011).

- Example: Image denoising by anisotropic penalty. See HW5.

- Example: (Linearly) constrained lasso

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\|\boldsymbol{y} - \beta_0\mathbf{1} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_1 \\ \text{subject to} \quad & \boldsymbol{G}\boldsymbol{\beta} \preceq \boldsymbol{h} \\ & \boldsymbol{A}\boldsymbol{\beta} = \boldsymbol{b}, \end{aligned}$$

  where $\lambda \geq 0$ is a tuning parameter.

- Example: The Huber loss function

$$\phi(r) = \begin{cases} r^2 & |r| \le M \\ M(2|r| - M) & |r| > M \end{cases}$$

is commonly used in robust statistics. The robust regression problem

$$\text{minimize} \quad \sum_{i=1}^{n} \phi(y_i - \beta_0 - \boldsymbol{x}_i^T \boldsymbol{\beta})$$

can be transformed to a QP

$$\begin{array}{ll} \text{minimize} & \boldsymbol{u}^T\boldsymbol{u} + 2M\mathbf{1}^T\boldsymbol{v} \\ \text{subject to} & -\boldsymbol{u} - \boldsymbol{v} \preceq \boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta} \preceq \boldsymbol{u} + \boldsymbol{v} \\ & \mathbf{0} \preceq \boldsymbol{u} \preceq M\mathbf{1}, \boldsymbol{v} \succeq \mathbf{0} \end{array}$$

in $\boldsymbol{u}, \boldsymbol{v} \in \mathbf{R}^n$ and $\boldsymbol{\beta} \in \mathbf{R}^p$. Hint: write $|r_i| = (|r_i| \wedge M) + (|r_i| - M)_+ = u_i + v_i$.

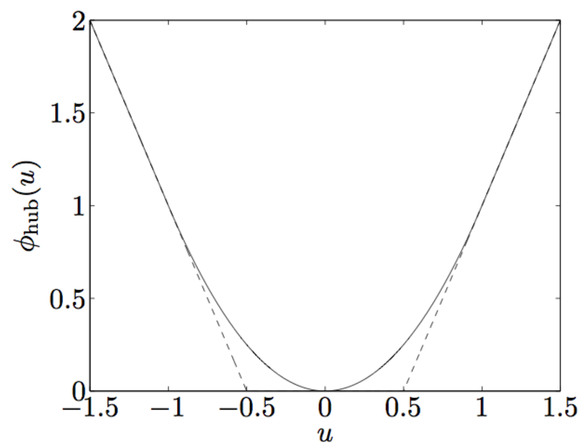


Figure 6.4 The solid line is the robust least-squares or Huber penalty function ϕ_{hub}, with $M = 1$. For $|u| \le M$ it is quadratic, and for $|u| > M$ it grows linearly.

- Example: Support vector machines (SVM, HW5). In two-class classification problems, we are given training data $(\boldsymbol{x}_i, y_i)$, $i = 1, \ldots, n$, where $\boldsymbol{x}_i \in \mathbf{R}^n$ are feature vector and $y_i \in \{-1, 1\}$ are class labels. Support vector machine solves the optimization problem

$$\text{minimize} \quad \sum_{i=1}^{n} \left[1 - y_i \left(\beta_0 + \sum_{j=1}^{p} x_{ij}\beta_j \right) \right]_+ + \lambda \|\boldsymbol{\beta}\|_2^2,$$

where $\lambda \ge 0$ is a tuning parameters. This is a QP.

Second-order cone programming (SOCP)

- A *second-order cone program* (SOCP)

$$\begin{array}{ll} \text{minimize} & \boldsymbol{f}^T\boldsymbol{x} \\ \text{subject to} & \|\boldsymbol{A}_i\boldsymbol{x} + \boldsymbol{b}_i\|_2 \le \boldsymbol{c}_i^T\boldsymbol{x} + d_i, \quad i = 1,\ldots,m \\ & \boldsymbol{F}\boldsymbol{x} = \boldsymbol{g} \end{array}$$

  over $\boldsymbol{x} \in \mathbf{R}^n$. This says the points $(\boldsymbol{A}_i\boldsymbol{x} + \boldsymbol{b}_i, \boldsymbol{c}_i^T\boldsymbol{x} + d_i)$ live in the second order cone (ice cream cone, Lorentz cone, quadratic cone)

$$\mathbf{Q}^{n+1} = \{(\boldsymbol{x}, t) : \|\boldsymbol{x}\|_2 \le t\}$$

  in $\mathbf{R}^{n+1}$.

  ☞ QP is a special case of SOCP. Why?

- When $\boldsymbol{c}_i = \mathbf{0}$ for $i = 1,\ldots,m$, SOCP is equivalent to a *quadratically constrained quadratic program* (QCQP)

$$\begin{array}{ll} \text{minimize} & (1/2)\boldsymbol{x}^T\boldsymbol{P}_0\boldsymbol{x} + \boldsymbol{q}_0^T\boldsymbol{x} \\ \text{subject to} & (1/2)\boldsymbol{x}^T\boldsymbol{P}_i\boldsymbol{x} + \boldsymbol{q}_i^T\boldsymbol{x} + r_i \le 0, \quad i = 1,\ldots,m \\ & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \end{array}$$

  where $\boldsymbol{P}_i \in \mathbf{S}_+^n$, $i = 0, 1, \ldots, m$. Why?

- Example: Group lasso (HW5). In many applications, we need to perform variable selection at group level. For instance, in factorial analysis, we want to select or de-select the group of regression coefficients for a factor simultaneously. Yuan and Lin (2006) propose the group lasso that

$$\text{minimize} \quad \frac{1}{2}\|\boldsymbol{y} - \beta_0\mathbf{1} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 + \lambda\sum_{g=1}^{G} w_g\|\boldsymbol{\beta}_g\|_2,$$

  where $\boldsymbol{\beta}_g$ is the subvector of regression coefficients for group g, and w_g are fixed group weights. This is equivalent to the SOCP

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\boldsymbol{\beta}^T\boldsymbol{X}^T\left(\boldsymbol{I} - \frac{\mathbf{1}\mathbf{1}^T}{n}\right)\boldsymbol{X}\boldsymbol{\beta} + \\ & \quad \boldsymbol{y}^T\left(\boldsymbol{I} - \frac{\mathbf{1}\mathbf{1}^T}{n}\right)\boldsymbol{X}\boldsymbol{\beta} + \lambda\sum_{g=1}^{G} w_g t_g \\ \text{subject to} & \|\boldsymbol{\beta}_g\|_2 \le t_g, \quad g = 1,\ldots,G, \end{array}$$

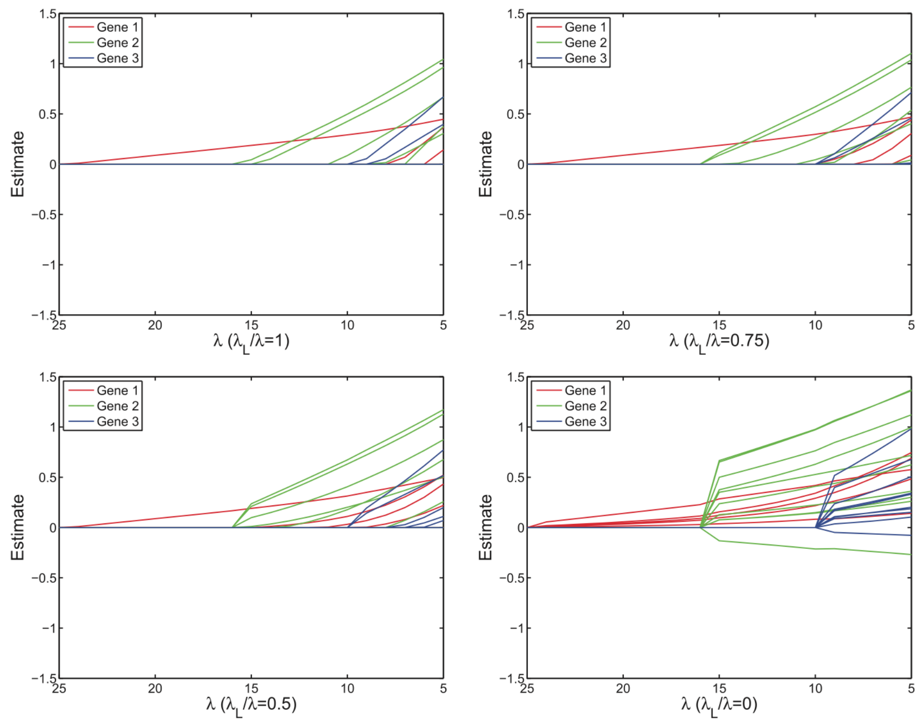
  in variables $\boldsymbol{\beta}$ and $t_1, \ldots, t_G$.

  ☞ Overlapping groups are allowed here.

- Example. Sparse group lasso

$$\text{minimize} \quad \frac{1}{2}\|\boldsymbol{y} - \beta_0 \mathbf{1} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 + \lambda_1 \|\boldsymbol{\beta}\|_1 + \lambda_2 \sum_{g=1}^{G} w_g \|\boldsymbol{\beta}_g\|_2$$

achieves sparsity at both group and individual coefficient level and can be solved by SOCP as well.



☞ Apparently we can solve any previous loss functions (quantile, ℓ_1, composite quantile, Huber, multi-response model) plus group or sparse group penalty by SOCP.

15 Lecture 15, Mar 16

Announcements

- HW4 (LP) due today 11:59PM.
- HW5 (QP, SOCP) due this Fri, Mar 20 @ 11:59PM.

Last Time

- QP (quadratic programming).
- SOCP (second order cone programming).

Today

- SOCP (cont'd).

SOCP (cont'd)

- Example. Square-root lasso (Belloni et al., 2011) minimizes

$$\|\boldsymbol{y} - \beta_0 \mathbf{1} - \boldsymbol{X}\boldsymbol{\beta}\|_2 + \lambda \|\boldsymbol{\beta}\|_1$$

  by SOCP. This variant generates the same solution path as lasso (why?) but simplifies the choice of λ.

  A demo example: `http://hua-zhou.github.io/teaching/st790-2015spr/demo_lasso.html`

- Example: Image denoising by ROF model. See HW5 Q4.
- A *rotated quadratic cone* in $\mathbf{R}^{n+2}$ is

$$\mathbf{Q}_r^{n+2} = \{(\boldsymbol{x}, t_1, t_2) : \|\boldsymbol{x}\|_2^2 \le 2t_1 t_2, t_1 \ge 0, t_2 \ge 0\}.$$

  A point $\boldsymbol{x} \in \mathbf{R}^{n+1}$ belongs to the second order cone $\mathbf{Q}^{n+1}$ if and only if

$$\begin{pmatrix} \boldsymbol{I}_{n-2} & 0 & 0 \\ 0 & -1/\sqrt{2} & 1/\sqrt{2} \\ 0 & 1/\sqrt{2} & 1/\sqrt{2} \end{pmatrix} \boldsymbol{x}$$

  belongs to the rotated quadratic cone $\mathbf{Q}_r^{n+1}$.

☞ Gurobi allows users to input second order cone constraint and quadratic constraints directly.

☞ Mosek allows users to input second order cone constraint, quadratic constraints, and rotated quadratic cone constraint directly.

- Following sets are *(rotated) quadratic cone representable sets*:
  - (Absolute values) $|x| \le t \Leftrightarrow (x, t) \in \mathbf{Q}^2$.
  - (Euclidean norms) $\|\boldsymbol{x}\|_2 \le t \Leftrightarrow (\boldsymbol{x}, t) \in \mathbf{Q}^{n+1}$.
  - (Squared Euclidean norms) $\|\boldsymbol{x}\|_2^2 \le t \Leftrightarrow (\boldsymbol{x}, t, 1/2) \in \mathbf{Q}_r^{n+2}$.
  - (Ellipsoid) For $\boldsymbol{P} \in \mathbf{S}_+^n$ and if $\boldsymbol{P} = \boldsymbol{F}^T\boldsymbol{F}$, where $\boldsymbol{F} \in \mathbf{R}^{n\times k}$, then

$$\begin{aligned} & (1/2)\boldsymbol{x}^T\boldsymbol{P}\boldsymbol{x} + \boldsymbol{c}^T\boldsymbol{x} + r \le 0 \\ \Leftrightarrow\ & \boldsymbol{x}^T\boldsymbol{P}\boldsymbol{x} \le 2t, t + \boldsymbol{c}^T\boldsymbol{x} + r = 0 \\ \Leftrightarrow\ & (\boldsymbol{F}\boldsymbol{x}, t, 1) \in \mathbf{Q}_r^{k+2}, t + \boldsymbol{c}^T\boldsymbol{x} + r = 0. \end{aligned}$$

    Similarly,

$$\|\boldsymbol{F}(\boldsymbol{x} - \boldsymbol{c})\|_2 \le t \Leftrightarrow (\boldsymbol{y}, t) \in \mathbf{Q}^{n+1}, \boldsymbol{y} = \boldsymbol{F}(\boldsymbol{x} - \boldsymbol{c}).$$

    ☞ This fact shows that QP and QCQP are instances of SOCP.
  - (Second order cones) $\|\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}\|_2 \le \boldsymbol{c}^T\boldsymbol{x} + d \Leftrightarrow (\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}, \boldsymbol{c}^T\boldsymbol{x} + d) \in \mathbf{Q}^{m+1}$.
  - (Simple polynomial sets)

$$\begin{aligned} \{(t,x) : |t| \le \sqrt{x}, x \ge 0\} &= \{(t,x) : (t, x, 1/2) \in \mathbf{Q}_r^3\} \\ \{(t,x) : t \ge x^{-1}, x \ge 0\} &= \{(t,x) : (\sqrt{2}, x, t) \in \mathbf{Q}_r^3\} \\ \{(t,x) : t \ge x^{3/2}, x \ge 0\} &= \{(t,x) : (x, s, t), (s, x, 1/8) \in \mathbf{Q}_r^3\} \\ \{(t,x) : t \ge x^{5/3}, x \ge 0\} &= \{(t,x) : (x, s, t), (s, 1/8, z), (z, s, x) \in \mathbf{Q}_r^3\} \\ \{(t,x) : t \ge x^{(2k-1)/k}, x \ge 0\}, k \ge 2, &\ \text{can be represented similarly} \\ \{(t,x) : t \ge x^{-2}, x \ge 0\} &= \{(t,x) : (s, t, 1/2), (\sqrt{2}, x, s) \in \mathbf{Q}_r^3\} \\ \{(t,x,y) : t \ge |x|^3/y^2, y \ge 0\} &= \{(t,x,y) : (x, z) \in \mathbf{Q}^2, (z, y/2, s), (s, t/2, z) \in \mathbf{Q}_r^3\} \end{aligned}$$

  - (Geometric mean) The hypograph of the (concave) geometric mean function

$$\mathbf{K}_{\text{gm}}^n = \{(\boldsymbol{x}, t) \in \mathbf{R}^{n+1} : (x_1 x_2 \cdots x_n)^{1/n} \ge t, \boldsymbol{x} \succeq \mathbf{0}\}$$

    can be represented by rotated quadratic cones. See (Lobo et al., 1998) for derivation. For example,

$$\begin{aligned} \mathbf{K}_{\text{gm}}^2 &= \{(x_1, x_2, t) : \sqrt{x_1 x_2} \ge t, x_1, x_2 \ge 0\} \\ &= \{(x_1, x_2, t) : (\sqrt{2}t, x_1, x_2) \in \mathbf{Q}_r^3\}. \end{aligned}$$

– (Harmonic mean) The hypograph of the harmonic mean function $\left(n^{-1}\sum_{i=1}^n x_i^{-1}\right)^{-1}$ can be represented by rotated quadratic cones

$$
\begin{aligned}
& \left(n^{-1}\sum_{i=1}^n x_i^{-1}\right)^{-1} \geq t, \boldsymbol{x} \succeq \mathbf{0} \\
\Leftrightarrow \quad & n^{-1}\sum_{i=1}^n x_i^{-1} \leq y, \boldsymbol{x} \succeq \mathbf{0} \\
\Leftrightarrow \quad & x_i z_i \geq 1, \sum_{i=1}^n z_i = ny, \boldsymbol{x} \succeq \mathbf{0} \\
\Leftrightarrow \quad & 2x_i z_i \geq 2, \sum_{i=1}^n z_i = ny, \boldsymbol{x} \succeq \mathbf{0}, \boldsymbol{z} \succeq \mathbf{0} \\
\Leftrightarrow \quad & (\sqrt{2}, x_i, z_i) \in \mathbf{Q}_r^3, \mathbf{1}^T \boldsymbol{z} = ny, \boldsymbol{x} \succeq \mathbf{0}, \boldsymbol{z} \succeq \mathbf{0}.
\end{aligned}
$$

– (Convex increasing rational powers) For $p, q \in \mathbf{Z}_+$ and $p/q \geq 1$,

$$\mathbf{K}^{p/q} = \{(x,t) : x^{p/q} \leq t, x \geq 0\} = \{(x,t) : (t\mathbf{1}_q, \mathbf{1}_{p-q}, x) \in \mathbf{K}_{\text{gm}}^p\}.$$

– (Convex decreasing rational powers) For any $p, q \in \mathbf{Z}_+$,

$$\mathbf{K}^{-p/q} = \{(x,t) : x^{-p/q} \leq t, x \geq 0\} = \{(x,t) : (x\mathbf{1}_p, t\mathbf{1}_q, 1) \in \mathbf{K}_{\text{gm}}^{p+q}\}.$$

– (Power cones) The *power cone* with rational powers is

$$\mathbf{K}_{\boldsymbol{\alpha}}^{n+1} = \left\{(\boldsymbol{x}, y) \in \mathbf{R}_+^n \times \mathbf{R} : |y| \leq \prod_{j=1}^n x_j^{p_j/q_j}\right\},$$

where p_j, q_j are integers satisfying $0 < p_j \leq q_j$ and $\sum_{j=1}^n p_j/q_j = 1$. Let $\beta = \text{lcm}(q_1, \ldots, q_n)$ and

$$s_j = \beta \sum_{k=1}^j \frac{p_k}{q_k}, \quad j = 1, \ldots, n-1.$$

Then it can be represented as

$$
\begin{aligned}
& |y| \leq (z_1 z_2 \cdots z_\beta)^{1/q} \\
& z_1 = \cdots = z_{s_1} = x_1, \quad z_{s_1+1} = \cdots = z_{s_2} = x_2, \quad z_{s_{n-1}+1} = \cdots = z_\beta = x_n.
\end{aligned}
$$

☞ References for above examples: Papers(Lobo et al., 1998; Alizadeh and Goldfarb, 2003) and book (Ben-Tal and Nemirovski, 2001, Lecture 3). Now our catalogue of SOCP terms includes all above terms.

☞ Most of these function are implemented as the built-in function in the convex optimization modeling language `cvx`.

- Example. ℓ_p regression with $p \geq 1$ a rational number

$$\text{minimize} \quad \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_p$$

can be formulated as a SOCP. Why? For instance, $\ell_{3/2}$ regression combines advantage of both robust ℓ_1 regression and least squares.

☞ `norm(x, p)` is a built-in function in the convex optimization modeling language `cvx`.

16 Lecture 16, Mar 18

Announcements

- HW5 (QP, SOCP) due this Fri, Mar 20 @ 11:59PM.
- HW6 (SDP, GP, MIP) posted `http://hua-zhou.github.io/teaching/st790-2015spr/ST790-2015-HW6.pdf`. Due Mon, Mar 30 @ 11:59PM.
- HW4 (LP) solution sketch posted. `http://hua-zhou.github.io/teaching/st790-2015spr/hw04sol.html`

Last Time

- SOCP (cont'd).

Today

- SDP (semidefinite programming).
- GP (geometric programming).

Semidefinite programming (SDP)

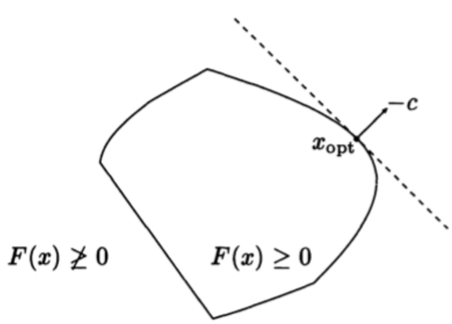


FIG. 1. *A simple semidefinite program with $x \in \mathbf{R}^2$, $F(x) \in \mathbf{R}^{7\times 7}$.*

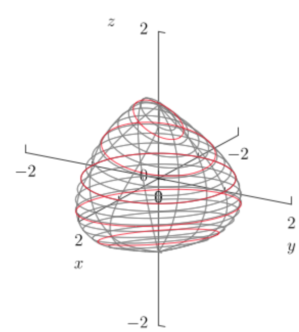


Figure 4.1: Plot of spectrahedron $S = \{(x,y,z) \in \mathbf{R}^3 \mid A(x,y,z) \succeq 0\}$.

- A *semidefinite program* (SDP) has the form

$$
\begin{array}{ll}
\text{minimize} & \boldsymbol{c}^T \boldsymbol{x} \\
\text{subject to} & x_1 \boldsymbol{F}_1 + \cdots + x_n \boldsymbol{F}_n + \boldsymbol{G} \preceq \mathbf{0} \quad \text{(LMI, linear matrix inequality)} \\
& \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b},
\end{array}
$$

where $\boldsymbol{G}, \boldsymbol{F}_1, \ldots, \boldsymbol{F}_n \in \mathbf{S}^k$, $\boldsymbol{A} \in \mathbf{R}^{p \times n}$, and $\boldsymbol{b} \in \mathbf{R}^p$.

☞ When $\boldsymbol{G}, \boldsymbol{F}_1, \ldots, \boldsymbol{F}_n$ are all diagonal, SDP reduces to LP.

- The *standard form SDP* has form

$$\begin{array}{ll} \text{minimize} & \operatorname{tr}(\boldsymbol{C}\boldsymbol{X}) \\ \text{subject to} & \operatorname{tr}(\boldsymbol{A}_i\boldsymbol{X}) = b_i, \quad i = 1, \ldots, p \\ & \boldsymbol{X} \succeq \mathbf{0}, \end{array}$$

where $\boldsymbol{C}, \boldsymbol{A}_1, \ldots, \boldsymbol{A}_p \in \mathbf{S}^n$.

- An *inequality form SDP* has form

$$\begin{array}{ll} \text{minimize} & \boldsymbol{c}^T\boldsymbol{x} \\ \text{subject to} & x_1\boldsymbol{A}_1 + \cdots + x_n\boldsymbol{A}_n \preceq \boldsymbol{B}, \end{array}$$

with variable $\boldsymbol{x} \in \mathbf{R}^n$, and parameters $\boldsymbol{B}, \boldsymbol{A}_1, \ldots, \boldsymbol{A}_n \in \mathbf{S}^n$, $\boldsymbol{c} \in \mathbf{R}^n$.

- Exercise. Write LP, QP, QCQP, and SOCP in form of SDP.

- Example. Nearest correlation matrix. Let $\mathbf{C}^n$ be the convex set of $n \times n$ correlation matrices

$$\mathbf{C} = \{\boldsymbol{X} \in \mathbf{S}_+^n : x_{ii} = 1, i = 1, \ldots, n\}.$$

Given $\boldsymbol{A} \in \mathbf{S}^n$, often we need to find the closest correlation matrix to $\boldsymbol{A}$

$$\begin{array}{ll} \text{minimize} & \|\boldsymbol{A} - \boldsymbol{X}\|_{\mathrm{F}} \\ \text{subject to} & \boldsymbol{X} \in \mathbf{C}. \end{array}$$

This projection problem can be solved via an SDP

$$\begin{array}{ll} \text{minimize} & t \\ \text{subject to} & \|\boldsymbol{A} - \boldsymbol{X}\|_{\mathrm{F}} \le t \\ & \boldsymbol{X} = \boldsymbol{X}^T, \ \operatorname{diag}(\boldsymbol{X}) = \mathbf{1} \\ & \boldsymbol{X} \succeq \mathbf{0} \end{array}$$

in variables $\boldsymbol{X} \in \mathbf{R}^{n \times n}$ and $t \in \mathbf{R}$. The SOC constraint can be written as an LMI

$$\begin{pmatrix} t\boldsymbol{I} & \operatorname{vec}(\boldsymbol{A} - \boldsymbol{X}) \\ \operatorname{vec}(\boldsymbol{A} - \boldsymbol{X})^T & t \end{pmatrix} \succeq \mathbf{0}$$

by the Schur complement lemma.

- Eigenvalue problems. Suppose

$$\boldsymbol{A}(\boldsymbol{x}) = \boldsymbol{A}_0 + x_1\boldsymbol{A}_1 + \cdots x_n\boldsymbol{A}_n,$$

  where $\boldsymbol{A}_i \in \mathbf{S}^m$, $i = 0, \ldots, n$. Let $\lambda_1(\boldsymbol{x}) \geq \lambda_2(\boldsymbol{x}) \geq \cdots \geq \lambda_m(\boldsymbol{x})$ be the ordered eigenvalues of $\boldsymbol{A}(\boldsymbol{x})$.

  - Minimize the maximal eigenvalue is equivalent to the SDP

$$\begin{array}{ll} \text{minimize} & t \\ \text{subject to} & \boldsymbol{A}(\boldsymbol{x}) \preceq t\boldsymbol{I} \end{array}$$

    in variables $\boldsymbol{x} \in \mathbf{R}^n$ and $t \in \mathbf{R}$.

    ☞ Minimizing the sum of k largest eigenvalues is an SDP too. How about minimizing the sum of all eigenvalues?

    ☞ Maximize the minimum eigenvalue is an SDP as well.

  - Minimize the spread of the eigenvalues $\lambda_1(\boldsymbol{x}) - \lambda_m(\boldsymbol{x})$ is equivalent to the SDP

$$\begin{array}{ll} \text{minimize} & t_1 - t_m \\ \text{subject to} & t_m\boldsymbol{I} \preceq \boldsymbol{A}(\boldsymbol{x}) \preceq t_1\boldsymbol{I} \end{array}$$

    in variables $\boldsymbol{x} \in \mathbf{R}^n$ and $t_1, t_m \in \mathbf{R}$.

  - Minimize the *spectral radius* (or *spectral norm*) $\rho(\boldsymbol{x}) = \max_{i=1,\ldots,m} |\lambda_i(\boldsymbol{x})|$ is equivalent to the SDP

$$\begin{array}{ll} \text{minimize} & t \\ \text{subject to} & -t\boldsymbol{I} \preceq \boldsymbol{A}(\boldsymbol{x}) \preceq t\boldsymbol{I} \end{array}$$

    in variables $\boldsymbol{x} \in \mathbf{R}^n$ and $t \in \mathbf{R}$.

  - To minimize the condition number $\kappa(\boldsymbol{x}) = \lambda_1(\boldsymbol{x})/\lambda_m(\boldsymbol{x})$, note $\lambda_1(\boldsymbol{x})/\lambda_m(\boldsymbol{x}) \leq t$ if and only if there exists a $\mu > 0$ such that $\mu\boldsymbol{I} \preceq \boldsymbol{A}(\boldsymbol{x}) \preceq \mu t\boldsymbol{I}$, or equivalently, $\boldsymbol{I} \preceq \mu^{-1}\boldsymbol{A}(\boldsymbol{x}) \preceq t\boldsymbol{I}$. With change of variables $y_i = x_i/\mu$ and $s = 1/\mu$, we can solve the SDP

$$\begin{array}{ll} \text{minimize} & t \\ \text{subject to} & \boldsymbol{I} \preceq s\boldsymbol{A}_0 + y_1\boldsymbol{A}_1 + \cdots y_n\boldsymbol{A}_n \preceq t\boldsymbol{I} \\ & s \geq 0, \end{array}$$

    in variables $\boldsymbol{y} \in \mathbf{R}^n$ and $s, t \geq 0$. In other words, we normalize the spectrum by the smallest eigenvalue and then minimize the largest eigenvalue of the normalized LMI.

- Minimize the ℓ_1 norm of the eigenvalues $|\lambda_1(\boldsymbol{x})| + \cdots + |\lambda_m(\boldsymbol{x})|$ is equivalent to the SDP

$$\begin{array}{ll} \text{minimize} & \operatorname{tr}(\boldsymbol{A}^+) + \operatorname{tr}(\boldsymbol{A}^-) \\ \text{subject to} & \boldsymbol{A}(\boldsymbol{x}) = \boldsymbol{A}^+ - \boldsymbol{A}^- \\ & \boldsymbol{A}^+ \succeq \boldsymbol{0}, \boldsymbol{A}^- \succeq \boldsymbol{0}, \end{array}$$

  in variables $\boldsymbol{x} \in \mathbf{R}^n$ and $\boldsymbol{A}^+, \boldsymbol{A}^- \in \mathbf{S}^n_+$.

- Roots of determinant. The determinant of a semidefinite matrix $\det(\boldsymbol{A}(\boldsymbol{x})) = \prod_{i=1}^m \lambda_i(\boldsymbol{x})$ is neither convex or concave, but rational powers of the determinant can be modeled using linear matrix inequalities. For a rational power $0 \le q \le 1/m$, the function $\det(\boldsymbol{A}(\boldsymbol{x}))^q$ is concave and we have

$$\begin{aligned} & t \le \det(\boldsymbol{A}(\boldsymbol{x}))^q \\ \Leftrightarrow \quad & \begin{pmatrix} \boldsymbol{A}(\boldsymbol{x}) & \boldsymbol{Z} \\ \boldsymbol{Z}^T & \operatorname{diag}(\boldsymbol{Z}) \end{pmatrix} \succeq \boldsymbol{0}, \quad (z_{11} z_{22} \cdots z_{mm})^q \ge t, \end{aligned}$$

  where $\boldsymbol{Z} \in \mathbf{R}^{m \times m}$ is a lower-triangular matrix. Similarly for any rational $q > 0$, we have

$$\begin{aligned} & t \ge \det(\boldsymbol{A}(\boldsymbol{x}))^{-q} \\ \Leftrightarrow \quad & \begin{pmatrix} \boldsymbol{A}(\boldsymbol{x}) & \boldsymbol{Z} \\ \boldsymbol{Z}^T & \operatorname{diag}(\boldsymbol{Z}) \end{pmatrix} \succeq \boldsymbol{0}, \quad (z_{11} z_{22} \cdots z_{mm})^{-q} \le t \end{aligned}$$

  for a lower triangular $\boldsymbol{Z}$.

- Trace of inverse. $\operatorname{tr}\boldsymbol{A}(\boldsymbol{x})^{-1} = \sum_{i=1}^m \lambda_i^{-1}(\boldsymbol{x})$ is a convex function and can be minimized using SDP

$$\begin{array}{ll} \text{minimize} & \operatorname{tr}\boldsymbol{B} \\ \text{subject to} & \begin{pmatrix} \boldsymbol{B} & \boldsymbol{I} \\ \boldsymbol{I} & \boldsymbol{A}(\boldsymbol{x}) \end{pmatrix} \succeq \boldsymbol{0}. \end{array}$$

  Note $\operatorname{tr}\boldsymbol{A}(\boldsymbol{x})^{-1} = \sum_{i=1}^m \boldsymbol{e}_i^T \boldsymbol{A}(\boldsymbol{x})^{-1} \boldsymbol{e}_i$. Therefore another equivalent formulation is

$$\begin{array}{ll} \text{minimize} & \displaystyle\sum_{i=1}^m t_i \\ \text{subject to} & \boldsymbol{e}_i^T \boldsymbol{A}(\boldsymbol{x})^{-1} \boldsymbol{e}_i \le t_i. \end{array}$$

  Now the constraints can be expressed by LMI

$$\boldsymbol{e}_i^T \boldsymbol{A}(\boldsymbol{x})^{-1} \boldsymbol{e}_i \le t_i \Leftrightarrow \begin{pmatrix} \boldsymbol{A}(\boldsymbol{x}) & \boldsymbol{e}_i \\ \boldsymbol{e}_i^T & t_i \end{pmatrix} \succeq \boldsymbol{0}.$$

☞ See (Ben-Tal and Nemirovski, 2001, Lecture 4, p146-p151) for the proof of above facts.

☞ `lambda_max`, `lambda_min`, `lambda_sum_largest`, `lambda_sum_smallest`, `det_rootn`, and `trace_inv` are implemented in `cvx` for Matlab.

☞ `lambda_max`, `lambda_min` are implemented in `Convex.jl` package for Julia.

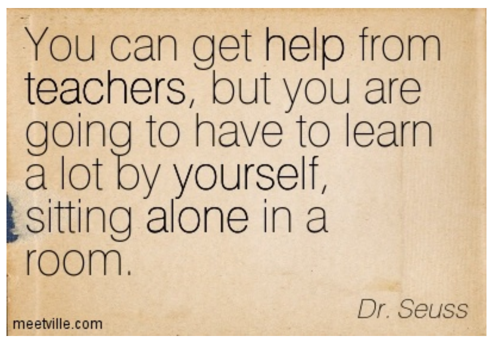
- Example. Experiment design. See HW6 Q1 `http://hua-zhou.github.io/teaching/st790-2015spr/ST790-2015-HW6.pdf`

17 Lecture 17, Mar 23

Announcements

- HW6 (SDP, GP, MIP) due next Mon, Mar 30 @ 11:59PM. `http://hua-zhou.github.io/teaching/st790-2015spr/ST790-2015-HW6.pdf`

- Lecture pace too fast ☹ For this course I put priority on diversity over thoroughness of topics. The goal is to introduce variety of tools that I consider useful but not covered in standard statistics curriculum. That means, given time limitation, many details have to be omitted. On the other hand, I have tried hard to point you to the best resources I know of (text book, lecture video, best software, ...) regarding these topics. It is *your* responsibility to follow up, understand and do homework problems, and internalize the material to become your own tools.

  For the convex optimization part, the most important thing is to keep a catalog of problems that can be solved by each problem class (LP, QP, SOCP, SDP, GP) and get familiar with the *good* convex optimization tools for solving them.

- On course project:

  – Ideally I hope *you* can come up a project that benefits yourself. You've learnt a lot tools from this course. Do something with them, that can turn into a manuscript, a software package, or a blog, and most importantly, something that interests yourself.

    * Re-examine the computational issues in your research projects. Is that slow? What's the bottleneck? Would `Rcpp` or changing to another language like `Julia` help? Is there an optimization problem there? Is that a convex problem? Can I do convex relaxation? Can I formulate the problem as a standard problem class (LP, QP, ...)?

* Create new applications by try different combinations of the terms in each category. Say XXX loss + XXX penalty? Can they solve some problems better (or faster) than current methods?
* Reverse engineering. Go over the examples and exercises in the textbook (Boyd and Vandenberghe, 2004) and ask yourself "this is cool, can I apply this to solve some statistical problems?"
* Do not worry about how to satisfy the instructor. Think about doing something that benefit yourself in the long run. Be creative and do not be afraid your idea dose not work. Even negative results are valuable; I appreciate negative results as far as I see a strong motivation and efforts in them and you provide some hindsights why the method does not work as you thought. And seriously, you should write a blog for whatever negative results you got. I think they have as much intellectual merits as published positive results.

"If your mentor handed you a sure-fire project, then it probably is dull." (Kenneth Lange)

> **授人以鱼不如授人以渔**
>
> *Give a man a fish, he eats for a day. Teach him to fish, he will never go hungry.*

- If you really lack ideas, work on an active competition on `kaggle.com`. Provide your best position in the leaderboard in your final project report.
- The final project report should look like a paper: introduction, motivation, method, algorithm, simulation studies if necessary, real data analysis, conclusion.

• Up to technology? NVIDIA CUDA v7.0 was released last week. A new library `cuSOLVER` provides a collection of dense and sparse direct solvers. `https://developer.nvidia.com/cusolver` This potentially opens up a lot GPU computing opportunities for statistics.

Last Time

• SDP.

Today

• SDP (cont'd).

SDP (cont'd)

- Singular value problems. Let $\boldsymbol{A}(\boldsymbol{x}) = \boldsymbol{A}_0 + x_1\boldsymbol{A}_1 + \cdots x_n\boldsymbol{A}_n$, where $\boldsymbol{A}_i \in \mathbf{R}^{p\times q}$ and $\sigma_1(\boldsymbol{x}) \geq \cdots \sigma_{\min\{p,q\}}(\boldsymbol{x}) \geq 0$ be the ordered singular values.
  - *Spectral norm* (or *operator norm* or *matrix-2 norm*) minimization. Consider minimizing the spectral norm $\|\boldsymbol{A}(\boldsymbol{x})\|_2 = \sigma_1(\boldsymbol{x})$. Note $\|\boldsymbol{A}\|_2 \leq t$ if and only if $\boldsymbol{A}^T\boldsymbol{A} \preceq t^2\boldsymbol{I}$ (and $t \geq 0$) if and only if $\begin{pmatrix} t\boldsymbol{I} & \boldsymbol{A} \\ \boldsymbol{A}^T & t\boldsymbol{I} \end{pmatrix} \succeq \mathbf{0}$. This results in the SDP

    $$\begin{array}{ll} \text{minimize} & t \\ \text{subject to} & \begin{pmatrix} t\boldsymbol{I} & \boldsymbol{A}(\boldsymbol{x}) \\ \boldsymbol{A}(\boldsymbol{x})^T & t\boldsymbol{I} \end{pmatrix} \succeq \mathbf{0} \end{array}$$

    in variables $\boldsymbol{x} \in \mathbf{R}^n$ and $t \in \mathbf{R}$.

    ☞ Minimizing the sum of k largest singular values is an SDP as well.
  - Nuclear norm minimization. Minimization of the *nuclear norm* (or *trace norm*) $\|\boldsymbol{A}(\boldsymbol{x})\|_* = \sum_i \sigma_i(\boldsymbol{x})$ can be formulated as an SDP.

    Argument 1: Singular values of $\boldsymbol{A}$ coincides with the eigenvalues of the symmetric matrix

    $$\begin{pmatrix} \mathbf{0} & \boldsymbol{A} \\ \boldsymbol{A}^T & \mathbf{0} \end{pmatrix},$$

    which has eigenvalues $(\sigma_1, \ldots, \sigma_p, -\sigma_p, \ldots, -\sigma_1)$. Therefore minimizing the nuclear norm of $\boldsymbol{A}$ is same as minimizing the ℓ_1 norm of eigenvalues of the augmented matrix, which we know is an SDP.

    Argument 2: An alternative characterization of nuclear norm is $\|\boldsymbol{A}\|_* = \sup_{\|\boldsymbol{Z}\|_2 \leq 1} \operatorname{tr}(\boldsymbol{A}^T\boldsymbol{Z})$. That is

    $$\begin{array}{ll} \text{maximize} & \operatorname{tr}(\boldsymbol{A}^T\boldsymbol{Z}) \\ \text{subject to} & \begin{pmatrix} \boldsymbol{I} & \boldsymbol{Z}^T \\ \boldsymbol{Z} & \boldsymbol{I} \end{pmatrix} \succeq \mathbf{0}, \end{array}$$

    with the dual problem

    $$\begin{array}{ll} \text{minimize} & \operatorname{tr}(\boldsymbol{U} + \boldsymbol{V})/2 \\ \text{subject to} & \begin{pmatrix} \boldsymbol{U} & \boldsymbol{A}(\boldsymbol{x})^T \\ \boldsymbol{A}(\boldsymbol{x}) & \boldsymbol{V} \end{pmatrix} \succeq \mathbf{0}. \end{array}$$

Therefore the epigraph of nuclear norm can be represented by LMI

$$\begin{aligned} & \|\boldsymbol{A}(\boldsymbol{x})\|_* \le t \\ \Leftrightarrow & \begin{pmatrix} \boldsymbol{U} & \boldsymbol{A}(\boldsymbol{x})^T \\ \boldsymbol{A}(\boldsymbol{x}) & \boldsymbol{V} \end{pmatrix} \succeq \boldsymbol{0}, \quad \operatorname{tr}(\boldsymbol{U}+\boldsymbol{V})/2 \le t. \end{aligned}$$

Argument 3: See (Ben-Tal and Nemirovski, 2001, Proposition 4.2.2, p154).

☞ See (Ben-Tal and Nemirovski, 2001, Lecture 4, p151-p154) for the proof of above facts.

☞ `sigma_max` and `norm_nuc` are implemented in `cvx` for Matlab.

☞ `operator_norm` and `nuclear_norm` are implemented in `Convex.jl` package for Julia.

- Example. Matrix completion. See HW6 Q2 `http://hua-zhou.github.io/teaching/st790-2015spr/ST790-2015-HW6.pdf`

- Quadratic or quadratic-over-linear matrix inequalities. Suppose

$$\begin{aligned} \boldsymbol{A}(\boldsymbol{x}) &= \boldsymbol{A}_0 + x_1\boldsymbol{A}_1 + \cdots + x_n\boldsymbol{A}_n \\ \boldsymbol{B}(\boldsymbol{y}) &= \boldsymbol{B}_0 + y_1\boldsymbol{B}_1 + \cdots + y_r\boldsymbol{B}_r. \end{aligned}$$

Then

$$\begin{aligned} & \boldsymbol{A}(\boldsymbol{x})^T\boldsymbol{B}(\boldsymbol{y})^{-1}\boldsymbol{A}(\boldsymbol{x}) \preceq \boldsymbol{C} \\ \Leftrightarrow & \begin{pmatrix} \boldsymbol{B}(\boldsymbol{y}) & \boldsymbol{A}(\boldsymbol{x})^T \\ \boldsymbol{A}(\boldsymbol{x}) & \boldsymbol{C} \end{pmatrix} \succeq \boldsymbol{0} \end{aligned}$$

by the Schur complement lemma.

☞ `matrix_frac()` is implemented in both `cvx` for Matlab and `Convex.jl` package for Julia.

- General quadratic matrix inequality. Let $\boldsymbol{X} \in \mathbf{R}^{m\times n}$ be a rectangular matrix and

$$F(\boldsymbol{X}) = (\boldsymbol{AXB})(\boldsymbol{AXB})^T + \boldsymbol{CXD} + (\boldsymbol{CXD})^T + \boldsymbol{E}$$

be a quadratic matrix-valued function. Then

$$\begin{aligned} & F(\boldsymbol{X}) \preceq \boldsymbol{Y} \\ \Leftrightarrow & \begin{pmatrix} \boldsymbol{I} & (\boldsymbol{AXB})^T \\ \boldsymbol{AXB} & \boldsymbol{Y} - \boldsymbol{E} - \boldsymbol{CXD} - (\boldsymbol{CXD})^T \end{pmatrix} \preceq \boldsymbol{0} \end{aligned}$$

by the Schur complement lemma.

- Another matrix inequality

$$\begin{aligned} & \boldsymbol{X} \succeq \mathbf{0}, \boldsymbol{Y} \preceq (\boldsymbol{C}^T \boldsymbol{X}^{-1} \boldsymbol{C})^{-1} \\ \Leftrightarrow \quad & \boldsymbol{Y} \preceq \boldsymbol{Z}, \boldsymbol{Z} \succeq \mathbf{0}, \boldsymbol{X} \succeq \boldsymbol{C}\boldsymbol{Z}\boldsymbol{C}^T. \end{aligned}$$

See (Ben-Tal and Nemirovski, 2001, 20.c, p155).

- Cone of nonnegative polynomials. Consider nonnegative polynomial of degree $2n$

$$f(t) = \boldsymbol{x}^T \boldsymbol{v}(t) = x_0 + x_1 t + \cdots x_{2n} t^{2n} \geq 0, \text{ for all } t.$$

The cone

$$\mathbf{K}^n = \{\boldsymbol{x} \in \mathbf{R}^{2n+1} : f(t) = \boldsymbol{x}^T \boldsymbol{v}(t) \geq 0, \text{ for all } t \in \mathbf{R}\}$$

can be characterized by LMI

$$f(t) \geq 0 \text{ for all } t \Leftrightarrow x_i = \langle \boldsymbol{X}, \boldsymbol{H}_i \rangle, i = 0, \ldots, 2n, \boldsymbol{X} \in \mathbf{S}_+^{n+1},$$

where $\boldsymbol{H}_i \in \mathbf{R}^{(n+1)\times(n+1)}$ are Hankel matrices with entries $(\boldsymbol{H}_i)_{kl} = 1$ if $k + l = i$ or 0 otherwise. Here $k, l \in \{0, 1, \ldots, n\}$.

Similarly the cone of nonnegative polynomials on a finite interval

$$\mathbf{K}_{a,b}^n = \{\boldsymbol{x} \in \mathbf{R}^{n+1} : f(t) = \boldsymbol{x}^T \boldsymbol{v}(t) \geq 0, \text{ for all } t \in [a, b]\}$$

can be characterized by LMI as well.

  - (Even degree) Let $n = 2m$. Then

$$\begin{aligned} \mathbf{K}_{a,b}^n = & \{\boldsymbol{x} \in \mathbf{R}^{n+1} : x_i = \langle \boldsymbol{X}_1, \boldsymbol{H}_i^m \rangle + \langle \boldsymbol{X}_2, (a+b)\boldsymbol{H}_{i-1}^{m-1} - ab\boldsymbol{H}_i^{m-1} - \boldsymbol{H}_{i-2}^{m-1} \rangle, \\ & i = 0, \ldots, n, \boldsymbol{X}_1 \in \mathbf{S}_+^m, \boldsymbol{X}_2 \in \mathbf{S}_+^{m-1}\}. \end{aligned}$$

  - (Odd degree) Let $n = 2m + 1$. Then

$$\begin{aligned} \mathbf{K}_{a,b}^n = & \{\boldsymbol{x} \in \mathbf{R}^{n+1} : x_i = \langle \boldsymbol{X}_1, \boldsymbol{H}_{i-1}^m - a\boldsymbol{H}_i^m \rangle + \langle \boldsymbol{X}_2, b\boldsymbol{H}_i^m - \boldsymbol{H}_{i-1}^m \rangle, \\ & i = 0, \ldots, n, \boldsymbol{X}_1, \boldsymbol{X}_2 \in \mathbf{S}_+^m\}. \end{aligned}$$

☞ References: paper (Nesterov, 2000) and the book (Ben-Tal and Nemirovski, 2001, Lecture 4, p157-p159).

- Example. Polynomial curve fitting. We want to fit a univariate polynomial of degree n

$$f(t) = x_0 + x_1 t + x_2 t^2 + \cdots x_n t^n$$

to a set of measurements (t_i, y_i), $i = 1, \ldots, m$, such that $f(t_i) \approx y_i$. Define the Vandermonde matrix

$$\boldsymbol{A} = \begin{pmatrix} 1 & t_1 & t_1^2 & \cdots & t_1^n \\ 1 & t_2 & t_2^2 & \cdots & t_2^n \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & t_m & t_m^2 & \cdots & t_m^n \end{pmatrix},$$

then we wish $\boldsymbol{A}\boldsymbol{x} \approx \boldsymbol{y}$. Using least squares criterion, we obtain the optimal solution $\boldsymbol{x}_{\mathrm{LS}} = (\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{y}$. With various constraints, it is possible to find optimal $\boldsymbol{x}$ by SDP.

1. Nonnegativity. Then we require $\boldsymbol{x} \in \mathbf{K}_{a,b}^n$.
2. Monotonicity. We can ensure monotonicity of $f(t)$ by requiring that $f'(t) \geq 0$ or $f'(t) \leq 0$. That is $(x_1, 2x_2, \ldots, nx_n) \in \mathbf{K}_{a,b}^{n-1}$ or $-(x_1, 2x_2, \ldots, nx_n) \in \mathbf{K}_{a,b}^{n-1}$.
3. Convexity or concavity. Convexity or concavity of $f(t)$ corresponds to $f''(t) \geq 0$ or $f''(t) \leq 0$. That is $(2x_2, 2x_3, \ldots, (n-1)nx_n) \in \mathbf{K}_{a,b}^{n-2}$ or $-(2x_2, 2x_3, \ldots, (n-1)nx_n) \in \mathbf{K}_{a,b}^{n-2}$.

☞ `nonneg_poly_coeffs()` and `convex_poly_coeffs()` are implemented in `cvx`. Not in `Convex.jl` yet.

18 Lecture 18, Mar 25

Announcements

- HW6 (SDP, GP, MIP) due next Mon, Mar 30 @ 11:59PM. `http://hua-zhou.github.io/teaching/st790-2015spr/ST790-2015-HW6.pdf`
- HW5 (QP, SOCP) solution sketch posted `http://hua-zhou.github.io/teaching/st790-2015spr/hw05sol.html`
- The `teaching` server is reserved for teaching purpose. Please do *not* run and store your research stuff on it. Each ST790-003 homework problem should take no longer than a few minutes. Most of them take only a couple seconds.

Last Time

- SDP (cont'd).

Today

- SDP (cont'd).
- GP (geometric programming).

SDP (cont'd)

- Example. Nonparametric density estimation by polynomials. See notes.
- SDP relaxation of combinatorial problems. An effective strategy to solve difficult combinatorial optimization problem (NP hard) is to bound the unknown optimal value. Upper bound is provided by any feasible point, while lower bound is often provided by a convex relaxation of the original problem.
  - SDP relaxation of binary optimization. Consider a binary linear optimization problem

    $$\begin{array}{ll} \text{minimize} & \boldsymbol{c}^T\boldsymbol{x} \\ \text{subject to} & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \quad \boldsymbol{x} \in \{0,1\}^n. \end{array}$$

    Note

    $$x_i \in \{0,1\} \Leftrightarrow x_i^2 = x_i \Leftrightarrow \boldsymbol{X} = \boldsymbol{x}\boldsymbol{x}^T, \text{diag}(\boldsymbol{X}) = \boldsymbol{x}.$$

By relaxing the rank 1 constraint on $\boldsymbol{X}$, we obtain an SDP relaxation

$$\begin{array}{ll} \text{minimize} & \boldsymbol{c}^T\boldsymbol{x} \\ \text{subject to} & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \text{diag}(\boldsymbol{X}) = \boldsymbol{x}, \boldsymbol{X} \succeq \boldsymbol{x}\boldsymbol{x}^T, \end{array}$$

which can be efficiently solved and provides a lower bound to the original problem. If the optimal $\boldsymbol{X}$ has rank 1, then it is a solution to the original binary problem also. Note $\boldsymbol{X} \succeq \boldsymbol{x}\boldsymbol{x}^T$ is equivalent to the LMI

$$\begin{pmatrix} 1 & \boldsymbol{x}^T \\ \boldsymbol{x} & \boldsymbol{X} \end{pmatrix} \succeq \boldsymbol{0}.$$

We can tighten the relaxation by adding other constraints that cut away part of the feasible set, without excluding rank 1 solutions. For instance, $0 \le x_i \le 1$ and $0 \le X_{ij} \le 1$.

– SDP relaxation of boolean optimization. For Boolean constraints $\boldsymbol{x} \in \{-1, 1\}^n$, we note

$$x_i \in \{0, 1\} \Leftrightarrow \boldsymbol{X} = \boldsymbol{x}\boldsymbol{x}^T, \text{diag}(\boldsymbol{X}) = \boldsymbol{1}.$$

☞ References: Paper (Laurent and Rendl, 2005) and book (Ben-Tal and Nemirovski, 2001, Lecture 4.3).

Geometric programming (GP)

- A function $f : \mathbf{R}^n \mapsto \mathbf{R}$ with $\text{dom} f = \mathbf{R}^n_{++}$ defined as

$$f(\boldsymbol{x}) = c x_1^{a_1} x_2^{a_2} \cdots x_n^{a_n},$$

where $c > 0$ and $a_i \in \mathbf{R}$, is called a *monomial*.

- A sum of monomials

$$f(\boldsymbol{x}) = \sum_{k=1}^{K} c_k x_1^{a_{1k}} x_2^{a_{2k}} \cdots x_n^{a_{nk}},$$

where $c_k > 0$, is called a *posynomial*.

- Posynomials are closed under addition, multiplication, and nonnegative scaling.

- A *geometric program* is of form

$$\begin{array}{lll} \text{minimize} & f_0(\boldsymbol{x}) & \\ \text{subject to} & f_i(\boldsymbol{x}) \le 1, & i = 1, \ldots, m \\ & h_i(\boldsymbol{x}) = 1, & i = 1, \ldots, p \end{array}$$

where $f_0, \ldots, f_m$ are posynomials and $h_1, \ldots, h_p$ are monomials. The constraint $\boldsymbol{x} \succ \boldsymbol{0}$ is implicit.

☞ Is GP a convex optimization problem?

- With change of variable $y_i = \ln x_i$, a monomial

$$f(\boldsymbol{x}) = c x_1^{a_1} x_2^{a_2} \cdots x_n^{a_n}$$

can be written as

$$f(\boldsymbol{x}) = f(e^{y_1}, \ldots, e^{y_n}) = c(e^{y_1})^{a_1} \cdots (e^{y_n})^{a_n} = e^{\boldsymbol{a}^T \boldsymbol{y} + b},$$

where $b = \ln c$. Similarly, we can write a posynomial as

$$\begin{aligned} f(\boldsymbol{x}) &= \sum_{k=1}^{K} c_k x_1^{a_{1k}} x_2^{a_{2k}} \cdots x_n^{a_{nk}} \\ &= \sum_{k=1}^{K} e^{\boldsymbol{a}_k^T \boldsymbol{y} + b_k}, \end{aligned}$$

where $\boldsymbol{a}_k = (a_{1k}, \ldots, a_{nk})$ and $b_k = \ln c_k$.

- The original GP can be expressed in terms of the new variable $\boldsymbol{y}$

$$\begin{array}{ll} \text{minimize} & \sum_{k=1}^{K_0} e^{\boldsymbol{a}_{0k}^T \boldsymbol{y} + b_{0k}} \\ \text{subject to} & \sum_{k=1}^{K_i} e^{\boldsymbol{a}_{ik}^T \boldsymbol{y} + b_{ik}} \leq 1, \quad i = 1, \ldots, m \\ & e^{\boldsymbol{g}_i^T \boldsymbol{y} + h_i} = 1, \quad i = 1, \ldots, p, \end{array}$$

where $\boldsymbol{a}_{ik}, \boldsymbol{g}_i \in \mathbf{R}^n$. Taking log of both objective and constraint functions, we obtain the *geometric program in convex form*

$$\begin{array}{ll} \text{minimize} & \ln\left(\sum_{k=1}^{K_0} e^{\boldsymbol{a}_{0k}^T \boldsymbol{y} + b_{0k}}\right) \\ \text{subject to} & \ln\left(\sum_{k=1}^{K_i} e^{\boldsymbol{a}_{ik}^T \boldsymbol{y} + b_{ik}}\right) \leq 0, \quad i = 1, \ldots, m \\ & \boldsymbol{g}_i^T \boldsymbol{y} + h_i = 0, \quad i = 1, \ldots, p. \end{array}$$

☞ Mosek is capable of solving GP. `cvx` has a GP mode that recognizes and transforms GP problems.

- Example. Logistic regression as GP. Given data $(\boldsymbol{x}_i, y_i)$, $i = 1, \ldots, n$, where $y_i \in \{0, 1\}$ and $\boldsymbol{x}_i \in \mathbf{R}^p$, the likelihood of the logistic regression model is

$$
\begin{aligned}
& \prod_{i=1}^{n} p_i^{y_i} (1 - p_i)^{1-y_i} \\
= & \prod_{i=1}^{n} \left(\frac{e^{\boldsymbol{x}_i^T \boldsymbol{\beta}}}{1 + e^{\boldsymbol{x}_i^T \boldsymbol{\beta}}} \right)^{y_i} \left(\frac{1}{1 + e^{\boldsymbol{x}_i^T \boldsymbol{\beta}}} \right)^{1-y_i} \\
= & \prod_{i:y_i=1} e^{\boldsymbol{x}_i^T \boldsymbol{\beta} y_i} \prod_{i=1}^{n} \left(\frac{1}{1 + e^{\boldsymbol{x}_i^T \boldsymbol{\beta}}} \right).
\end{aligned}
$$

The MLE solves

$$
\text{minimize} \quad \prod_{i:y_i=1} e^{-\boldsymbol{x}_i^T \boldsymbol{\beta}} \prod_{i=1}^{n} \left(1 + e^{\boldsymbol{x}_i^T \boldsymbol{\beta}} \right).
$$

Let $z_j = e^{\beta_j}$, $j = 1, \ldots, p$. The objective becomes

$$
\prod_{i:y_i=1} \prod_{j=1}^{p} z_j^{-x_{ij}} \prod_{i=1}^{n} \left(1 + \prod_{j=1}^{p} z_j^{x_{ij}} \right).
$$

This leads to a GP

$$
\begin{aligned}
\text{minimize} \quad & \prod_{i:y_i=1} s_i \prod_{i=1}^{n} t_i \\
\text{subject to} \quad & \prod_{j=1}^{p} z_j^{-x_{ij}} \le s_i, \quad i = 1, \ldots, m \\
& 1 + \prod_{j=1}^{p} z_j^{x_{ij}} \le t_i, \quad i = 1, \ldots, n,
\end{aligned}
$$

in variables $\boldsymbol{s} \in \mathbf{R}^m$, $\boldsymbol{t} \in \mathbf{R}^n$, and $\boldsymbol{z} \in \mathbf{R}^p$. Here m is the number of observations with $y_i = 1$.

☞ How to incorporate lasso penalty? Let $z_j^+ = e^{\beta_j^+}$, $z_j^- = e^{\beta_j^-}$. Lasso penalty takes the form $e^{\lambda|\beta_j|} = (z_j^+ z_j^-)^\lambda$.

- Example. Bradley-Terry model for sports ranking. See ST758 HW8 `http://hua-zhou.github.io/teaching/st758-2014fall/ST758-2014-HW8.pdf`. The likelihood is

$$
\prod_{i,j} \left(\frac{\gamma_i}{\gamma_i + \gamma_j} \right)^{y_{ij}}.
$$

MLE is solved by GP

$$\begin{array}{ll} \text{minimize} & \prod_{i,j} t_{ij}^{y_{ij}} \\ \text{subject to} & 1 + \gamma_i^{-1}\gamma_j \leq t_{ij} \end{array}$$

in $\boldsymbol{\gamma} \in \mathbf{R}^n$ and $\boldsymbol{t} \in \mathbf{R}^{n^2}$.

19 Lecture 19, Mar 30

Announcements

- HW6 (SDP, GP, MIP) deadline extended to this Wed, Apr 1 @ 11:59PM. Some hints if you use `Convex.jl` package in `Julia` for HW6:
  - Q1(a): `Convex.jl` does not implement root determinant function but it implements the `logdet` function that you can use
  - Q1(d): `Convex.jl` does not implement `trace_inv` function but you can easily formulate it as an SDP
  - Q4(a): `Convex.jl` does not model GP (geometric program), but you can use change of variable $y_i = \ln x_i$ and utilize the logsumexp function in Convex.jl
  - Q4(b): `Convex.jl` does not have a `log_normcdf` function but you can learn the quadratic approximation trick from cvx `https://github.com/cvxr/CVX/blob/master/functions/%40cvx/log_normcdf.m`

Last Time

- SDP (cont'd).
- GP (geometric programming).

Today

- Cone programming.
- Separable convex optimization in Mosek.
- Mixed integer programming (MIP).
- Planned topics for remaining of the course: algorithms for sparse and regularized regressions, dynamic programming, EM/MM advanced topics: s.e., convergence and acceleration, and online estimation.

Generalized inequalities and cone programming

- A cone $\mathbf{K} \in \mathbf{R}^n$ is *proper* if it is closed, convex, has non-empty interior, and is pointed, i.e., $\boldsymbol{x} \in \mathbf{K}$ and $-\boldsymbol{x} \in \mathbf{K}$ implies $\boldsymbol{x} = \mathbf{0}$.

A proper cone defines a partial ordering on $\mathbf{R}^n$ via *generalized inequalities*: $\boldsymbol{x} \preceq_{\mathbf{K}} \boldsymbol{y}$ if and only if $\boldsymbol{y} - \boldsymbol{x} \in \mathbf{K}$ and $\boldsymbol{x} \prec \boldsymbol{y}$ if and only if $\boldsymbol{y} - \boldsymbol{x} \in \text{int}(\mathbf{K})$.

E.g., $\boldsymbol{X} \preceq \boldsymbol{Y}$ means $\boldsymbol{Y} - \boldsymbol{X} \in \mathbf{S}_+^n$ and $\boldsymbol{X} \prec \boldsymbol{Y}$ means $\boldsymbol{Y} - \boldsymbol{X} \in \mathbf{S}_{++}^n$.

- A *conic form problem* or *cone program* has the form

$$\begin{aligned} \text{minimize} \quad & \boldsymbol{c}^T \boldsymbol{x} \\ \text{subject to} \quad & \boldsymbol{F}\boldsymbol{x} + \boldsymbol{g} \preceq_K \mathbf{0} \\ & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}. \end{aligned}$$

- The *conic form problem in standard form* is

$$\begin{aligned} \text{minimize} \quad & \boldsymbol{c}^T \boldsymbol{x} \\ \text{subject to} \quad & \boldsymbol{x} \succeq_K \mathbf{0} \\ & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}. \end{aligned}$$

- The *conic form problem in inequality form* is

$$\begin{aligned} \text{minimize} \quad & \boldsymbol{c}^T \boldsymbol{x} \\ \text{subject to} \quad & \boldsymbol{F}\boldsymbol{x} + \boldsymbol{g} \preceq_K \mathbf{0}. \end{aligned}$$

- Special cases of cone programming.
  - Nonnegative orthant $\{\boldsymbol{x} | \boldsymbol{x} \succeq \mathbf{0}\}$: LP
  - Second order cone $\{(\boldsymbol{x}, t) | \|\boldsymbol{x}\|_2 \le t\}$: SOCP
  - Rotated quadratic cone $\{(\boldsymbol{x}, t_1, t_2) | \|\boldsymbol{x}\|_2^2 \le 2t_1 t_2\}$: SOCP
  - Geometric mean cone $\{(\boldsymbol{x}, t) | (\prod x_i)^{1/n} \ge y, \boldsymbol{x} \succeq \mathbf{0}\}$: SOCP
  - Semidefinite cone $\mathbf{S}_+^n$: SDP
  - Nonnegative polynomial cone: SDP
  - Monotone polynomial cone: SDP
  - Convex/concave polynomial cone: SDP
  - Exponential cone $\{(x, y, z) | y e^{x/y} \le z, y > 0\}$. Terms `logsumexp`, `exp`, `log`, `entropy`, `lndet`, ... are exponential cone representable.
- Where is today's technology up to?
  - `Gurobi` implements up to SOCP.

- `Mosek` implements up to SDP.
- `SCS` (free solver accessible from `Convex.jl`) can deal with exponential cone program.
- `cvx` uses a successive approximation strategy to deal with exponential cone representable terms, which only relies on SOCP.
  `http://web.cvxr.com/cvx/doc/advanced.html#successive`
  ☞ `cvx` implements `log_det` and `log_sum_exp`.
- `Convex.jl` accepts exponential cone representable terms, which can solve using SCS.
  ☞ `Convex.jl` implements `logsumexp`, `exp`, `log`, `entropy`, and `logistic_loss`.

- Example. Logistic regression as an exponential cone problem

$$\text{minimize} \quad -\sum_{i:y_i=1} \boldsymbol{x}_i^T \boldsymbol{\beta} + \sum_{i=1}^n \ln\left(1 + e^{\boldsymbol{x}_i^T \boldsymbol{\beta}}\right).$$

  See `cvx` example library for an example for logistic regression. `http://cvxr.com/cvx/examples/`
  See the link for an example using Julia. `http://nbviewer.ipython.org/github/JuliaOpt/Convex.jl/blob/master/examples/logistic_regression.ipynb`

- Example. Gaussian covariance estimation and graphical lasso

$$\ln\det(\boldsymbol{\Sigma}) + \operatorname{tr}(\boldsymbol{S}\boldsymbol{\Sigma}) - \lambda\|\text{vec}\Sigma\|_1$$

  involves exponential cones because of the $\ln\det$ term.

Separable convex optimization in Mosek

- Mosek is posed to solve general convex nonlinear programs (NLP) of form

$$\begin{array}{ll} \text{minimize} & f(\boldsymbol{x}) + \boldsymbol{c}^T\boldsymbol{x} \\ \text{subject to} & l_i \le g_i(\boldsymbol{x}) + \boldsymbol{a}_i^T\boldsymbol{x} \le u_i, \quad i = 1, \ldots, m \\ & \boldsymbol{l}^x \preceq \boldsymbol{x} \preceq \boldsymbol{u}^x. \end{array}$$

  Here functions $f : \mathbf{R}^n \mapsto \mathbf{R}$ and $g_i : \mathbf{R}^n \mapsto \mathbf{R}$, $i = 1, \ldots, m$ must be *separable* in parameters.

- The example

$$\begin{array}{ll} \text{minimize} & x_1 - \ln(x_1 + 2x_2) \\ \text{subject to} & x_1^2 + x_2^2 \le 1 \end{array}$$

is not separable. But the equivalent formulation

$$\begin{array}{ll} \text{minimize} & x_1 - \ln(x_3) \\ \text{subject to} & x_1^2 + x_2^2 \leq 1, x_1 + 2x_2 - x_3 = 0, x_3 \geq 0 \end{array}$$

is.

- It should cover a lot statistical applications. But I have no experience with its performance yet.
- Which modeling tool to use?
  - `cvx` and `Convex.jl` can *not* model general NLP.
  - `JuMP.jl` in `Julia` can model NLP or even MINLP. See `http://jump.readthedocs.org/en/latest/nlp.html`

Other topics in convex optimization

- Duality theory. (Boyd and Vandenberghe, 2004, Chapter 5).
- Algorithms. Interior point method. (Boyd and Vandenberghe, 2004) Part III (Chapters 9-11).
- History:
  1. 1948: Dantzig's simplex algorithm for solving LP.
  2. 1984: first practical polynomial-time algorithm (interior point method) by Karmarkar.
  3. 1984-1990: efficient implementations for large-scale LP.
  4. around 1990: polynomial-time interior-point methods for nonlinear convex programming by Nesterov and Nemirovski.
  5. since 1990: extensions and high-quality software packages.

Mixed integer programming

- Mixed integer program allows certain optimization variables to be integer.
- Current technology can solve small to moderately sized MILP and MIQP.

  ☞ `cvx` allows binary and integer variables.
  `Convex.jl` for Julia does *not* allow integer variables.
  `JuMP.jl` for Julia allows binary and integer variables.

- Modeling using integer variables. References (Nemhauser and Wolsey, 1999; Williams, 2013).
  - (Positivity) If $0 \leq x < M$ for a known upper bound M, then we can model the implication $(x > 0) \rightarrow (z = 1)$ by linear inequality $x \leq Mz$, where $z \in \{0, 1\}$.

    Similarly if $0 < m \leq x$ for a known lower bound m. Then we can model the implication $(z = 1) \rightarrow (x > 0)$ by the linear inequality $x \geq mz$, where $z \in \{0, 1\}$.
  - (Semi-continuity) We can model semi-continuity of a variable $x \in \mathbf{R}$, $x \in 0 \cup [a, b]$ where $0 < a \leq b$ using a double inequality $az \leq x \leq bz$ where $z \in \{0, 1\}$.
  - (Constraint satisfaction) Suppose we know the upper bound M on $\boldsymbol{a}^T\boldsymbol{x} - b$. Then the implication $(z = 1) \rightarrow (\boldsymbol{a}^T\boldsymbol{x} \leq b)$ can be modeled as

    $$\boldsymbol{a}^T\boldsymbol{x} \leq b + M(1 - z),$$

    where $z \in \{0, 1\}$. Equivalently the reverse implication $(\boldsymbol{a}^T\boldsymbol{x} \leq b) \rightarrow (z = 1)$ is modeled as

    $$\boldsymbol{a}^T\boldsymbol{x} \geq b + (m - \epsilon)z + \epsilon.$$

    where $m < \boldsymbol{a}^T\boldsymbol{x} - b$ is a lower bound. Collectively we model $(\boldsymbol{a}^T\boldsymbol{x} \leq b) \leftrightarrow (z = 1)$ as

    $$\boldsymbol{a}^T\boldsymbol{x} \leq b + M(1 - z), \quad \boldsymbol{a}^T\boldsymbol{x} \geq b + (m - \epsilon)z + \epsilon.$$

    In a similar fashion, $(z = 1) \leftrightarrow (\boldsymbol{a}^T\boldsymbol{x} \geq b)$ is modeled as

    $$\boldsymbol{a}^T\boldsymbol{x} \geq b + M(1 - z), \quad \boldsymbol{a}^T\boldsymbol{x} \leq b + (m - \epsilon)z + \epsilon$$

    using the lower bound $m < \boldsymbol{a}^T\boldsymbol{x} - b$ and upper bound $M > \boldsymbol{a}^T\boldsymbol{x} - b$.

    Combining both we can model equality $\boldsymbol{a}^T\boldsymbol{x} = b$ by modeling $(z = 1) \rightarrow (\boldsymbol{a}^T\boldsymbol{x} = b$ by

    $$\boldsymbol{a}^T\boldsymbol{x} \leq b + M(1 - z), \quad \boldsymbol{a}^T\boldsymbol{x} \geq b + m(1 - z)$$

    and $(z = 0) \rightarrow (\boldsymbol{a}^T\boldsymbol{x} \neq b)$ by

    $$\boldsymbol{a}^T\boldsymbol{x} \geq b + (m - \epsilon)z_1 + \epsilon, \quad \boldsymbol{a}^T\boldsymbol{x} \leq b + (M + \epsilon)z_2 - \epsilon, \quad z_1 + z_2 - z \leq 1,$$

    where $z_1 + z_2 - z \leq 1$ is equivalent to $(z = 0) \rightarrow (z_1 = 0) \vee (z_2 = 0)$.

- (Disjunctive constraints) The requirement that at least one out of a set of constraints is satisfied $(z = 1) \to (\boldsymbol{a}_1^T \boldsymbol{x} \le b_1) \vee (\boldsymbol{a}_2^T \boldsymbol{x} \le b_2) \vee \cdots \vee (\boldsymbol{a}_k^T \boldsymbol{x} \le b_k)$ can be modeled by

$$z = z_1 + \cdots z_k \ge 1, \quad \boldsymbol{a}_j^T \boldsymbol{x} \le b_j + M(1 - z_j), \quad \text{for all } j,$$

  where $z_j \in \{0, 1\}$ are binary variables and $M > \boldsymbol{a}_j^T - b_j$ for all j is a collective upper bound.

  The reverse implication $(\boldsymbol{a}_1^T \boldsymbol{x} \le b_1) \vee (\boldsymbol{a}_2^T \boldsymbol{x} \le b_2) \vee \cdots \vee (\boldsymbol{a}_k^T \boldsymbol{x} \le b_k) \to (z = 1)$ is modeld as

$$\boldsymbol{a}_j^T \boldsymbol{x} \ge b + (m - \epsilon) z + \epsilon, \quad j = 1, \dots, k,$$

  with a lower bound $m < \boldsymbol{a}_j^T \boldsymbol{x} - b_j$ for all j and $z \in \{0, 1\}$.
- (Pack constraints) The requirement at most one of the constraints are satisfied is modeled as

$$z_1 + \cdots + z_k \le 1, \quad \boldsymbol{a}_j^T \boldsymbol{x} \le b_j + M(1 - z_j), \quad \text{for all j.}$$

- (Partition constraints) The requirement exactly one of the constraints are satisfied is modeled as

$$z_1 + \cdots + z_k = 1, \quad \boldsymbol{a}_j^T \boldsymbol{x} \le b_j + M(1 - z_j), \quad \text{for all j.}$$

- Boolean primitives.
  * *Complement*

x	$\neg x$
0	1
1	0

    is modeled as $\neg x = 1 - x$.
  * *Conjunction*

x	y	$x \wedge y$
0	0	0
1	0	0
0	1	0
1	1	1

    $z = (x \wedge y)$ is modeled as $z + 1 \ge x + y, x \ge z, y \ge z$.
  * *Disjunction*

x	y	$x \vee y$
0	0	0
1	0	1
0	1	1
1	1	1

is modeled as $x + y \geq 1$.

∗ *Implication*

x	y	$x \to y$
0	0	1
1	0	0
0	1	1
1	1	1

is modeled as $x \leq y$.

– Special ordered set constraint: SOS1 and SOS2. See (Williams, 2013, Section 9.3) or (Bertsimas and Weismantel, 2005).

An *SOS1 constraint* is a set of variables for which at most one variable in the set may take a value other than zero. An *SOS2 constraint* is an ordered set of variables where at most two variables in the set may take non-zero values. If two take non-zeros values, they must be contiguous in the ordered set.

☞ Gurobi solver allows SOS1 and SOS2 constraints. `JuMP.jl` modeling tool for Julia accepts SOS1 and SOS2 constraints and pass them to solvers that support them. `cvx` and `Convex.jl` dose not take SOS constraints.

- Example. Best subset regression. HW6 Q3. Consider

$$\begin{aligned} \text{minimize} \quad & \|\boldsymbol{y} - \beta_0 \mathbf{1} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 \\ \text{subject to} \quad & \|\boldsymbol{\beta}\|_0 \leq k. \end{aligned}$$

Introducing binary variables $z_j \in \{0, 1\}$ such that $(\beta_j \neq 0) \to (z_j = 1)$, then it can be formulated as a MIQP

$$\begin{aligned} \text{minimize} \quad & \|\boldsymbol{y} - \beta_0 \mathbf{1} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 \\ \text{subject to} \quad & -M z_j \leq \beta_j \leq M z_j \\ & \sum_{j=1}^{p} z_j \leq k, \end{aligned}$$

where $M \geq \|\boldsymbol{\beta}\|_\infty$. Alternatively, we may model cardinality constraint by SOS1 constraints $\{\beta_j, z_j\} \in \text{SOS1}$, which does not need a pre-defined M.

☞ We should be able to do best subset XXX for all problems in HW4/5 by formulating a corresponding MILP, MIQP or MISOCP.

- Example. Variable selection in presence of interaction. Consider variable selection for linear regression with p predictors and their pairwise interactions. For better interpretability, we may want to retain interaction terms only when their main effects are in the model as well. We may achieve this by

$$\text{minimize} \quad \frac{1}{2}\sum_{i=1}^{n}\left(y_i - \mu - \sum_{j=1}^{p} x_{ij}\beta_j - \sum_{j,j'} x_{ij}x_{ij'}\beta_{jj'}\right)^2 + \lambda\left(\sum_{j=1}^{p}|\beta_j| + \sum_{j,j'}|\beta_{jj'}|\right)$$

subject to the logical constraints $(\beta_{jj'} \neq 0) \to (\beta_j \neq 0) \wedge (\beta_{j'} \neq 0)$.

- Example. Sudoku. How to solve Sudoku using integer programming?
Define solution as a binary array $\boldsymbol{X} \in \{0,1\}^{9\times 9\times 9}$ with entries $x_{ijk} = 1$ if and only if (i,j)-th entry is integer k. We need constraints

  1. Each square in the 2D grid has exactly one value. So $\sum_{k=1}^{9} x_{ijk} = 1$.
  2. Each row i of the 2D grid has exactly one value out of each of the digits from 1 to 9. So $\sum_{j=1}^{9} x_{ijk} = 1$.
  3. Each column i of the 2D grid has exactly one value out of each of the digits from 1 to 9. So $\sum_{i=1}^{9} x_{ijk} = 1$.
  4. The major 3-by-3 grids have similar property. So $\sum_{i=1}^{3}\sum_{j=1}^{3} x_{i+U,j+V,k} = 1$, where $U, V \in \{0,3,6\}$.
  5. Observed entries prescribe $x_{ijm} = 1$ if (i,j)-th entry is integer m.

  Julia: `http://nbviewer.ipython.org/github/JuliaOpt/juliaopt-notebooks/blob/master/notebooks/JuMP-Sudoku.ipynb`
  Matlab: `http://www.mathworks.com/help/optim/ug/solve-sudoku-puzzles-via-integer-prog html`

- Optimization involving piecewise-linear functions can be formulated as MIP. See (Vielma et al., 2010).

20 Lecture 20, Apr 1

Announcements

- HW6 (SDP, GP, MIP) due today @ 11:59PM. Don't forget git tag your submission.
- A few more course project ideas added on `http://hua-zhou.github.io/teaching/st790-2015spr/project.html`

Last Time

- Cone programming.
- Separable convex optimization in Mosek.
- Mixed integer programming (MIP).

Today

- Sparse regression.

Sparse regression: what and why?

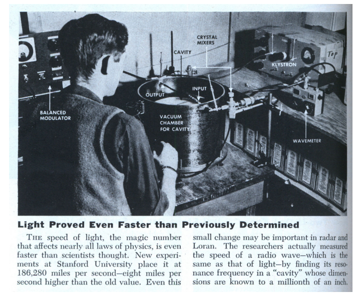
- Famous lasso (Donoho and Johnstone, 1994, 1995; Tibshirani, 1996)

$$\text{minimize} \qquad \frac{1}{2}\|\boldsymbol{y} - \beta_0\mathbf{1} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 + \lambda \sum_{j=1}^{p} |\beta_j|.$$

  Why everyone does this?

  - Shrinkage
  - Model selection
  - Computational efficiency (convex optimization)

- Why shrinkage? Idea of shrinkage dates back to one of the most surprising results in mathematical statistics in the 20th century. Let's consider the simple task of estimating population mean(s).



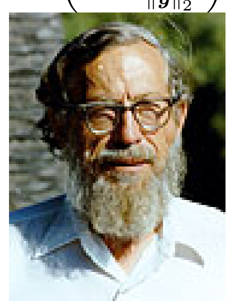
Light Proved Even Faster than Previously Determined

THE speed of light, the magic number that affects nearly all laws of physics, is even faster than scientists thought. New experiments at Stanford University place it at 186,280 miles per second—eight miles per second higher than the old value. Even this small change may be important in radar and Loran. The researchers actually measured the speed of a radio wave—which is the same as that of light—by finding its resonance frequency in a "cavity" whose dimensions are known to a millionth of an inch.

- How to estimate hog weight in Montana?
- How to estimate hog weight in Montana *and* tea consumption in China?
- How to estimate hog weight in Montana, tea consumption in China, *and* speed of light?

- Stein's paradox.

$$\hat{\boldsymbol{\mu}}_{\text{LS}} = \boldsymbol{y} \qquad \text{or} \quad \hat{\boldsymbol{\mu}}_{\text{JS}} = \left(1 - \frac{(m-2)}{\|\boldsymbol{y}\|_2^2}\right) \boldsymbol{y}?$$



The James-Stein shrinkage estimator $\hat{\boldsymbol{\mu}}_{\text{JS}}$ dominates the least squares estimate $\hat{\boldsymbol{\mu}}_{\text{LS}}$ when the number of populations $m \geq 3$!

- Observe independent $y_i \sim N(\mu_i, 1)$, $i = 1, \ldots, m$.

Theorem 1. *For $m \geq 3$, the James-Stein estimator $\hat{\boldsymbol{\mu}}_{JS}$ everywhere dominates the MLE $\hat{\boldsymbol{\mu}}_{LS}$ in terms of the expected total squared error; that is*

$$\mathbf{E}_{\boldsymbol{\mu}} \|\hat{\boldsymbol{\mu}}_{JS} - \boldsymbol{\mu}\|_2^2 < \mathbf{E}_{\boldsymbol{\mu}} \|\hat{\boldsymbol{\mu}}_{LS} - \boldsymbol{\mu}\|_2^2$$

for every choice of $\boldsymbol{\mu}$.

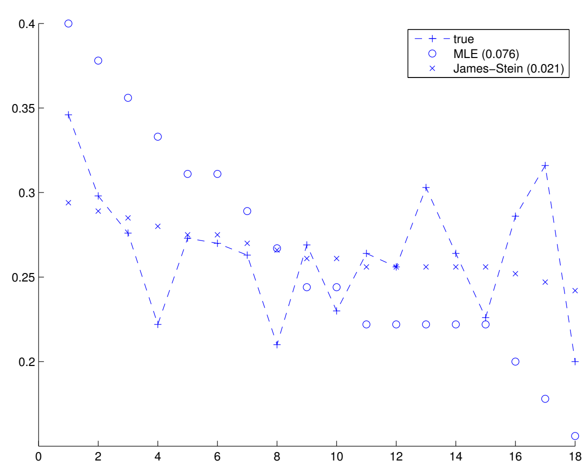
  - Stein (1956) showed the inadmissibility of $\hat{\boldsymbol{\mu}}_{\text{LS}}$; his student James and himself later proposed the specific form of $\hat{\boldsymbol{\mu}}_{\text{JS}}$ in (James and Stein, 1961).
  - Message: when estimating *many* parameters, *shrinkage* helps improve risk property, even when the parameters are totally unrelated to each other.

- Efron's famous baseball example (Efron and Morris, 1977).

Table 1.1: Batting averages $z_i = \hat{\mu}_i^{(\mathrm{MLE})}$ for 18 major league players early in the 1970 season; μ_i values are averages over the remainder of the season. The James–Stein estimates $\hat{\mu}_i^{(\mathrm{JS})}$ (1.35) based on the z_i values provide much more accurate overall predictions for the μ_i values. (By coincidence, $\hat{\mu}_i$ and μ_i both average 0.265; the average of $\hat{\mu}_i^{(\mathrm{JS})}$ must equal that of $\hat{\mu}_i^{(\mathrm{MLE})}$.)

Name	hits/AB	$\hat{\mu}_i^{(\mathrm{MLE})}$	μ_i	$\hat{\mu}_i^{(\mathrm{JS})}$
Clemente	18/45	.400	**.346**	.294
F Robinson	17/45	.378	**.298**	.289
F Howard	16/45	.356	**.276**	.285
Johnstone	15/45	.333	**.222**	.280
Berry	14/45	.311	**.273**	.275
Spencer	14/45	.311	**.270**	.275
Kessinger	13/45	.289	**.263**	.270
L Alvarado	12/45	.267	**.210**	.266
Santo	11/45	.244	**.269**	.261
Swoboda	11/45	.244	**.230**	.261
Unser	10/45	.222	**.264**	.256
Williams	10/45	.222	**.256**	.256
Scott	10/45	.222	**.303**	.256
Petrocelli	10/45	.222	**.264**	.256
E Rodriguez	10/45	.222	**.226**	.256
Campaneris	9/45	.200	**.286**	.252
Munson	8/45	.178	**.316**	.247
Alvis	7/45	.156	**.200**	.242
Grand Average		.265	**.265**	.265

MLE empirical risk: 0.076. James-Stein (shrinkage towards average) empirical risk: 0.021



- Stein's effect is universal and underlies many modern statistical learning methods

- Empirical Bayes connection (Efron and Morris, 1973)
"Learning from the experience of the others" (John Tukey)

- Why $m \geq 3$? Connection with transience/recurrence of Markov chains (Brown, 1971; Eaton, 1992)
  "A drunk man will eventually find his way home but a drunk bird may get lost forever." (Kakutani at a UCLA colloquium talk)

- Now we see the benefits of shrinkage. Lasso has the added benefit of model selection.

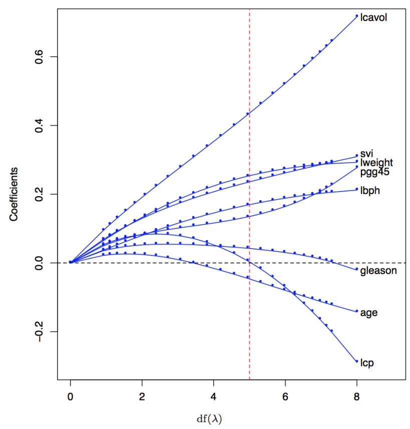


FIGURE 3.8. *Profiles of ridge coefficients for the prostate cancer example, as the tuning parameter λ is varied. Coefficients are plotted versus $\mathrm{df}(\lambda)$, the effective degrees of freedom. A vertical line is drawn at $\mathrm{df} = 5.0$, the value chosen by cross-validation.*

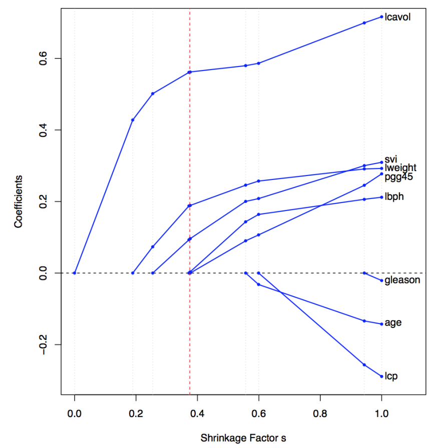


FIGURE 3.10. *Profiles of lasso coefficients, as the tuning parameter t is varied. Coefficients are plotted versus $s = t/\sum_1^p |\hat\beta_j|$. A vertical line is drawn at $s = 0.36$, the value chosen by cross-validation. Compare Figure 3.8 on page 65; the lasso profiles hit zero, while those for ridge do not. The profiles are piece-wise linear, and so are computed only at the points displayed; see Section 3.4.4 for details.*

  The left plot shows the solution path of ridge regression

$$\text{minimize} \quad \|\boldsymbol{y} - \beta_0 \mathbf{1} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 + \lambda \sum_{j=1}^p \beta_j^2$$

  for the prostate cancer data in HW4/5/6. The right plot shows the lasso solution path on the same data set. We see both ridge and lasso shrink $\hat{\boldsymbol{\beta}}$. But lasso has the extra benefit of performing variable selection.

- A general sparse regression minimizes the criterion

$$f(\boldsymbol{\beta}) + \sum_{j=1}^p P_\eta(|\beta_j|, \lambda)$$

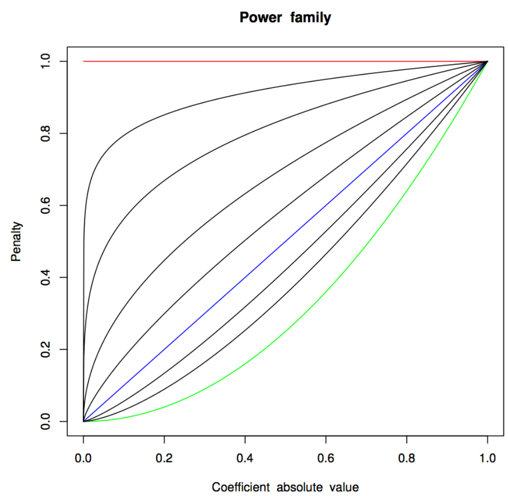
  - f a differentiable loss function
    * $f(\boldsymbol{\beta}) = \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2/2$: linear regression
    * $f(\boldsymbol{\beta}) = -\ell(\boldsymbol{\beta})$: negative log-likelihood (GLM, Cox model, ...)
    * ...

- P: the penalty function
- λ: penalty tuning parameter
- η: index a penalty family

- Power family penalty (bridge regression) (Frank and Friedman, 1993)

$$P_\eta(|w|, \lambda) \quad = \lambda |w|^\eta, \qquad \eta \in [0, 2].$$

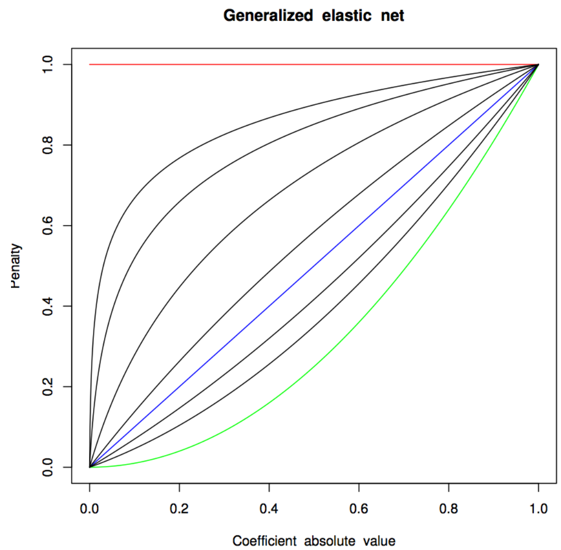
  - $\eta \in (0, 1)$: concave, $\eta \in [1, 2]$: convex
  - $\eta = 2$: ridge, $\eta = 1$: lasso, $\eta = 0$: ℓ_0 (best subset) regression



- Elastic net penalty (Zou and Hastie, 2005)

$$P_\eta(|w|, \lambda) \quad = \lambda \left\{(\eta - 1)w^2/2 + (2 - \eta)|w|\right\}, \qquad \eta \in [1, 2]$$

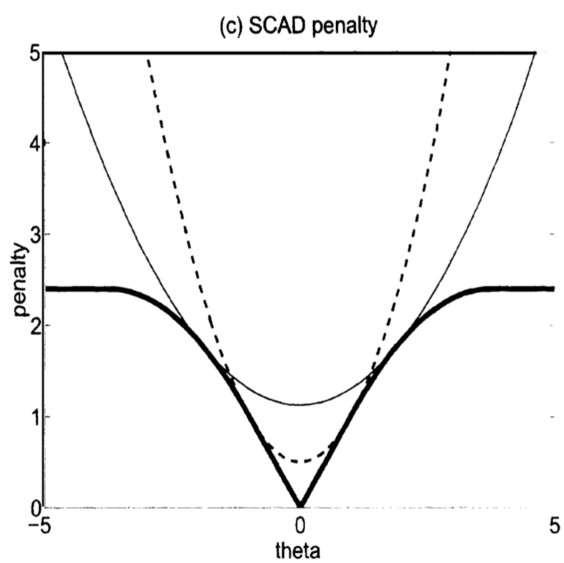
  - Enet tries to combine both lasso and ridge penalty.
  - $\eta = 1$: lasso, $\eta = 2$: ridge.
  - Friedman (2008) calls the (concave) log penalty generalized enet.



- SCAD (Fan and Li, 2001),

$$P_\eta(|w|,\lambda) \;= \begin{cases} \lambda|w| & |w|<\lambda \\ \lambda^2 + \frac{\eta\lambda(|w|-\lambda)}{\eta-1} - \frac{w^2-\lambda^2}{2(\eta-1)} & |w|\in[\lambda,\eta\lambda] \\ \lambda^2(\eta+1)/2 & |w|>\eta\lambda \end{cases}$$

  - for small signals $|w|<\lambda$, it acts as lasso; for large signals $|w|>\eta\lambda$, the penalty flattens and leads to the unbiasedness of the regularized estimate



- Log penalty (Candès et al., 2008; Armagan et al., 2013)

$$P_\eta(|w|,\lambda) \;= \lambda\ln(\eta+|w|), \quad \eta>0$$

- MC+ penalty (Zhang, 2010)

$$P_\eta(|w|, \lambda) = \left(\lambda|w| - \frac{w^2}{2\eta}\right) 1_{\{|w|<\lambda\eta\}} + \frac{\lambda^2\eta}{2} 1_{\{|w|\geq\lambda\eta\}}, \quad \eta > 0,$$

  is quadratic on $[0, \lambda\eta]$ and flattens beyond $\lambda\eta$. Varying η from 0 to ∞ bridges hard thresholding (ℓ_0 regression) to lasso (ℓ_1) shrinkage.

Sparse regression: overview of algorithms

- Difficulties in minimizing

$$f(\boldsymbol{\beta}) + \sum_{j=1}^{p} P_\eta(|\beta_j|, \lambda).$$

  - Non-smooth. Not differentiable at $\beta_j = 0$.
  - Possibility non-convex.
  - Extremely high dimensions in modern applications. E.g., $p \sim 10^6$ in genetics.

- We discuss following algorithms.

  - Convex optimization softwares if applicable.
  - Coordinate descent.
  - Nesterov method (accelerated proximal gradient method).
  - Path following algorithm.

- We have seen many examples where convex optimization softwares apply. For a convex loss f and convex penalty P, write $\beta_j = \beta_j^+ - \beta_j^-$, where $\beta_j^+ = \max\{\beta_j, 0\}$ and $\beta_j^- = -\min\{\beta_j, 0\}$. Then we minimize the objective

$$f(\boldsymbol{\beta}^+ - \boldsymbol{\beta}^-) + \sum_{j=1}^{p} P_\eta(\beta_j^+ + \beta_j^-, \lambda)$$

  subject to nonnegativity constraints $\beta_j^+, \beta_j^- \geq 0$ using a convex optimization solver.

  ☞ To guarantee β_j^+ and β_j^- to be the positive and negative part of β_j, we also need the (non-convex) constraint $\beta_j^+\beta_j^- = 0$. This condition can be dispensed in sparse regression because the penalty function is an increasing function in $(\beta_j^+ + \beta_j^-)$. So the solution will always put β_j^+ or β_j^- to be 0.

- May not be efficient for extremely high dimensional, unstructured problems.

21 Lecture 21, Apr 6

Announcements

- HW6 solution sketch posted: `http://hua-zhou.github.io/teaching/st790-2015spr/hw06sol.html`

Last Time

- Sparse regression: introduction.

Today

- Coordinate descent for sparse regression.
- Proximal gradient method.

Coordinate descent (CD)

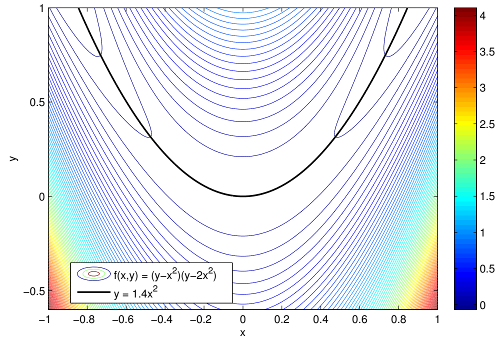
- Idea: coordinate-wise minimization of β_j

$$\begin{aligned} \beta_j^{(t+1)} &\leftarrow \text{argmin}_{\beta_j} f(\beta_1^{(t+1)}, \ldots, \beta_{j-1}^{(t+1)}, \beta_j, \beta_{j+1}^{(t)}, \ldots, \beta_p^{(t)}) + P_\eta(|\beta_j|, \lambda) \\ \text{for} &\quad j = 1, \ldots, p \end{aligned}$$

  until objective value converges. Similar to the Gauss-Seidel method for solving linear equations. Why objective value converges?

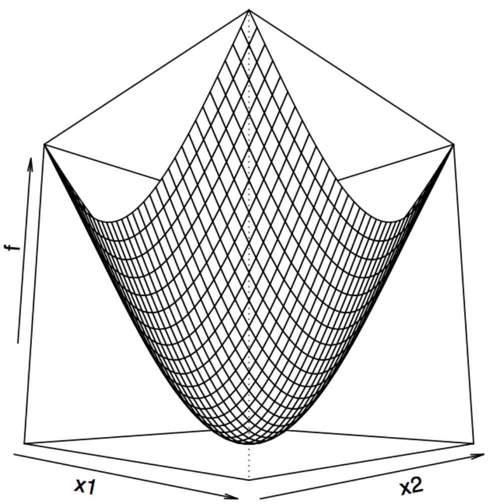
- Success stories
  - Linear regression (Fu, 1998; Daubechies et al., 2004; Friedman et al., 2007; Wu and Lange, 2008): GlmNet in R.
  - GLM (Friedman et al., 2010): GlmNet in R.
  - Non-convex penalties (Mazumder et al., 2011): SparseNet in R.

- Why CD works for sparse regressions?
  - Q1: Given a *non-convex* function f, if we are at a point $\boldsymbol{x}$ such that f is minimized along each coordinate axis, is $\boldsymbol{x}$ a global minimum?
    * Exercise: consider $f(x, y) = (y - x^2)(y - 2x^2)$. Show that all directional derivatives at (0,0) is nonnegative, but it is not a local minimum



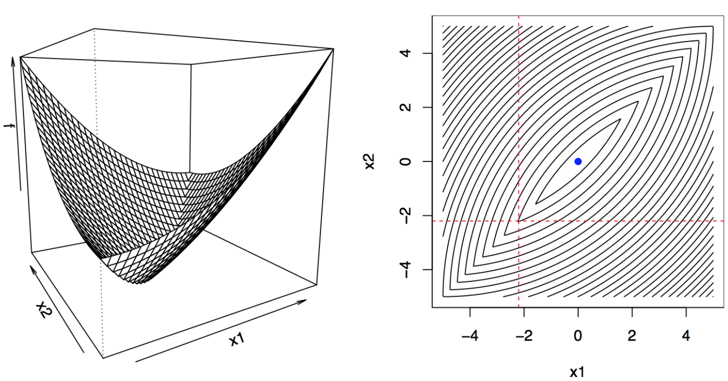
Answer: No.

– Q2: Same question, but for a *convex*, *differentiable* f.



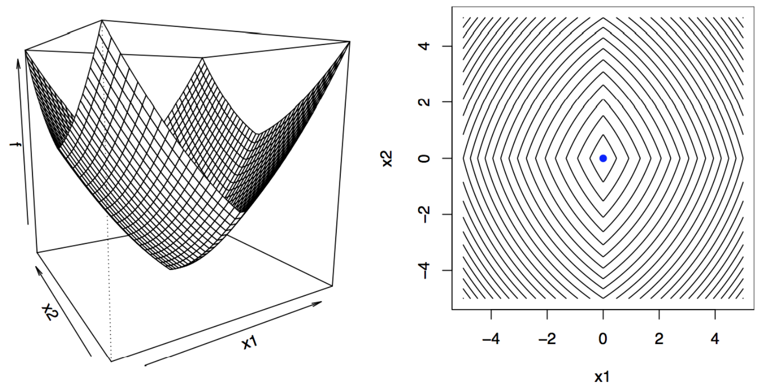
Answer: Yes. Why?

– Q3: Same question, but for a *convex*, *non-differentiable* f.



Answer: No.

– Q4: Same question, but for $h(\boldsymbol{x}) = f(\boldsymbol{x}) + \sum_j g_j(x_j)$, where f is convex and differentiable and g_j are convex but not necessarily differentiable.



Yes. Proof: for any $\boldsymbol{y}$,

$$\begin{aligned} h(\boldsymbol{y}) - h(\boldsymbol{x}) &= f(\boldsymbol{y}) - f(\boldsymbol{x}) + \sum_j [g_j(y_j) - g_j(x_j)] \\ &\geq \nabla f(\boldsymbol{x})^\mathsf{T}(\boldsymbol{y} - \boldsymbol{x}) + \sum_j [g_j(y_j) - g_j(x_j)] \\ &= \sum_j [\nabla_j f(\boldsymbol{x})(y_j - x_j) + g_j(y_j) - g_j(x_j)] \\ &\geq 0. \end{aligned}$$

The first inequality is by supporting hyperplane inequality for f. The second inequality is because h is minimized along x_j coordinate thus by the first order optimality condition

$$0 \in \nabla_j f(\boldsymbol{x}) + \partial g_j(x_j)$$

or equivalently

$$-\nabla_j f(\boldsymbol{x}) \in \partial g_j(x_j).$$

Then, by the definition of subgradient,

$$g_j(y_j) - g_j(x_j) \geq -\nabla_j f(\boldsymbol{x})(y_j - x_j).$$

– This justifies the CD algorithm for sparse regression of form $f(\boldsymbol{\beta}) + \sum_{j=1}^p P_\eta(|\beta_j|, \lambda)$, when both loss and penalty are convex.

– Tseng (2001) rigorously shows the convergence of CD. For f continuous on compact set $\{\boldsymbol{x} : f(\boldsymbol{x}) \leq f(\boldsymbol{x}^{(0)})\}$ and attaining its minimum, any limit point of CD is a minimizer of f.

- Example. Lasso penalized linear regression.

$$\min_{\beta_0,\boldsymbol{\beta}} \quad \frac{1}{2}\sum_{i=1}^{n}(y_i - \beta_0 - \boldsymbol{x}_i^{\mathsf{T}}\boldsymbol{\beta})^2 + \lambda\sum_{j=1}^{p}|\boldsymbol{\beta}_j|.$$

  - Update of intercept β_0

$$\begin{aligned}
\beta_0^{(t+1)} &= \frac{1}{n}\sum_{i=1}^{n}(y_i - \boldsymbol{x}_i^{\mathsf{T}}\boldsymbol{\beta}^{(t)}) \\
&= \frac{1}{n}\sum_{i=1}^{n}(y_i - \beta_0^{(t)} - \boldsymbol{x}_i^{\mathsf{T}}\boldsymbol{\beta}^{(t)} + \beta_0^{(t)}) \\
&= \beta_0^{(t)} + \frac{1}{n}\sum_{i=1}^{n} r_i^{(t)}.
\end{aligned}$$

  - Update of β_j

$$\begin{aligned}
\beta_j^{(t+1)} &= \arg\min_{\beta_j} \frac{1}{2}\sum_{i=1}^{n}\left[y_i - \beta_0^{(t)} - \boldsymbol{x}_i^T\boldsymbol{\beta}^{(t)} - (\beta_j - \beta_j^{(t)})x_{ij}\right]^2 + \lambda|\beta_j| \\
&= \arg\min_{\beta_j} \frac{1}{2}\sum_{i=1}^{n}\left[r_i^{(t)} - (\beta_j - \beta_j^{(t)})x_{ij}\right]^2 + \lambda|\beta_j| \\
&= \arg\min_{\beta_j} \frac{\boldsymbol{x}_{\cdot j}^T\boldsymbol{x}_{\cdot j}}{2}\left(\beta_j - \beta_j^{(t)} - \frac{\boldsymbol{x}_{\cdot j}^{\mathsf{T}}\boldsymbol{r}^{(t)}}{\boldsymbol{x}_{\cdot j}^{\mathsf{T}}\boldsymbol{x}_{\cdot j}}\right)^2 + \lambda|\beta_j| \\
&= \mathrm{ST}\left(\beta_j^{(t)} + \frac{\boldsymbol{x}_{\cdot j}^{\mathsf{T}}\boldsymbol{r}^{(t)}}{\boldsymbol{x}_{\cdot j}^{\mathsf{T}}\boldsymbol{x}_{\cdot j}}, \frac{\lambda}{\boldsymbol{x}_{\cdot j}^{\mathsf{T}}\boldsymbol{x}_{\cdot j}}\right),
\end{aligned}$$

    where

$$\mathrm{ST}(z,\gamma) = \arg\min_{x} \frac{1}{2}(x-z)^2 + \gamma|x| = \mathrm{sgn}(z)(|z|-\gamma)_+$$

    is the *soft-thresholding* operator.

  - Organize computation around residuals $\boldsymbol{r}$. Each coordinate update requires computing $\boldsymbol{x}_{\cdot j}^{\mathsf{T}}\boldsymbol{r}^{(t)}$ and update of $\boldsymbol{r}^{(t+1)} \leftarrow \boldsymbol{r}^{(t)} + (\beta_j^{(t)} - \beta_j^{(t+1)})\boldsymbol{x}_{\cdot j}$, totally $O(n)$ flops or less with structures.

- Example. Lasso penalized generalized linear model (GLM).

$$\text{minimize} \quad f(\boldsymbol{\beta}) + \lambda\sum_{j=1}^{p}|\beta_j|,$$

  where f is the negative log-likelihood of a GLM.

- Method 1: Use Newton method to update coordinate β_j (Wu et al., 2009).
- Method 2 (IWLS): Each coordinate descent sweep is performed on the quadratic approximation

$$\frac{1}{2}\sum_{i=1}^{n} w_i^{(t)}(z_i^{(t)} - \boldsymbol{x}_i^{\mathsf{T}}\boldsymbol{\beta})^2 + \lambda \sum_{j=1}^{p} |\beta_j|,$$

where $w_i^{(t)}$ are the working weights and $z_i^{(t)}$ are the working responses (Friedman et al., 2010).

☞ IWLS becomes more popular because it needs much less exponentiations.

- Remarks on CD.
  - *Scalable* to extremely large p with careful implementation, because most variables keep parked at 0 at large λ. Can be slow at smaller λ, where many β_j are non-zero.
  - *Active set* strategy. Keep updating active predictors until convergence and then check other predictors. See (Tibshirani et al., 2012).
  - *Warm start* from large λ: move from sparser solutions to dense ones, using solution at previous λ as initial value for next λ.
  - Coding in lower level language (C/C++, Fortran, Julia?) is almost necessary for efficiency due to extensive looping.

    ☞ What Trevor Hastie calls the FFT trick: **F**riedman + **F**ortran + some numerical **T**ricks = no waste flops.
  - Wide applicability of CD: ℓ_1 regression (Wu and Lange, 2008), svm (Platt, 1999), group lasso (block CD), graphical lasso (Friedman et al., 2008), ...

22 Lecture 22, Apr 8

Last Time

- Coordinate descent for sparse regression.

Today

- Proximal gradient and accelerated proximal gradient method.

Proximal gradient and accelerated proximal gradient method: why?

- Because of applications in machine learning and statistics, there is a resurgence of interests in first order optimization methods that use only gradient information since 90s.
- The classical gradient descent (steepest descent) method minimizes a differentiable function f by iterating

$$\boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{(t)} - s\nabla f(\boldsymbol{x}^{(t)})$$

  - Step size s can be fixed or determined by line search (backtracking or exact)
  - Advantages
    - Each iteration is inexpensive.
    - No need to derive, compute, store and invert Hessians; attractive in large scale problems.
  - Disadvantages
    - Slow convergence (zigzagging).

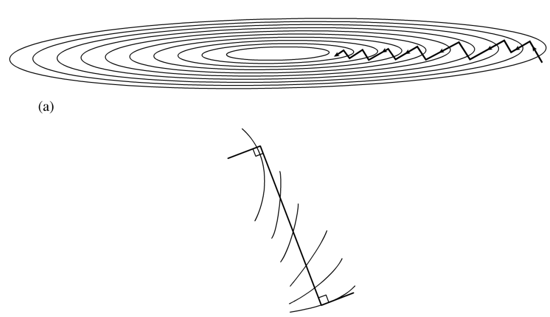
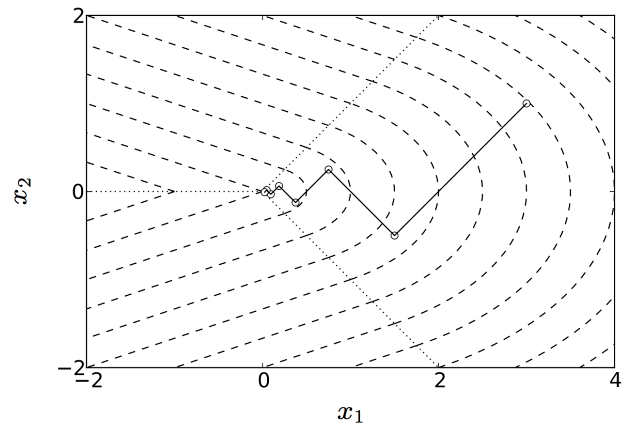


Figure 10.6.1. (a) Steepest descent method in a long, narrow "valley." While more efficient than the strategy of Figure 10.5.1, steepest descent is nonetheless an inefficient strategy, taking many steps to reach the valley floor. (b) Magnified view of one step: A step starts off in the local gradient direction, perpendicular to the contour lines, and traverses a straight line until a local minimum is reached, where the traverse is parallel to the local contour lines.

    * Do not work for non-smooth problems.



- Remedies
    * Slow convergence:
        · conjugate gradient method
        · quasi-Newton
        · **accelerated gradient method**
    * Non-differentiable or constrained problems:
        · subgradient method
        · **proximal gradient method**
        · smoothing method
        · cutting-plane methods

Proximal gradient method

☞ A definite resource for learning about proximal algorithms is (Parikh and Boyd, 2013) `https://web.stanford.edu/~boyd/papers/prox_algs.html`

- "Much like Newton's method is a standard tool for solving unconstrained smooth minimization problems of modest size, *proximal algorithms* can be viewed as an analogous tool for nonsmooth, constrained, large-scale, or distributed versions of these problems."
- The *proximal mapping* (or *prox-operator*) of a convex function g is

$$\text{prox}_g(\boldsymbol{x}) = \text{argmin}_{\boldsymbol{u}} \left(g(\boldsymbol{u}) + \frac{1}{2}\|\boldsymbol{u} - \boldsymbol{x}\|_2^2 \right).$$

☞ Intuitively $\text{prox}_g(\boldsymbol{x})$ moves towards the minimum of g but not far away (proximal) from the point $\boldsymbol{x}$.

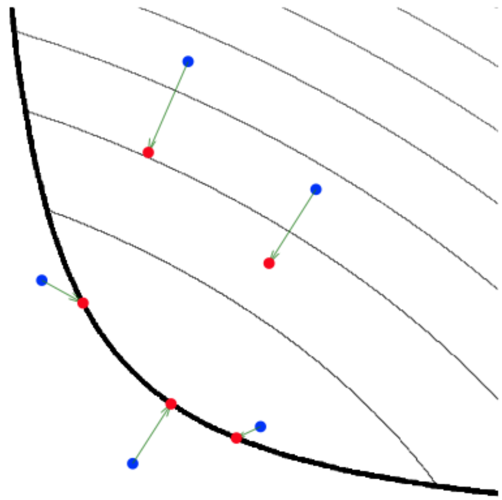


Figure 1.1: Evaluating a proximal operator at various points.

- Fact: For a *closed* convex g, $\text{prox}_g(\boldsymbol{x})$ exists and is unique for all $\boldsymbol{x}$.

  ☞ A function $f(\boldsymbol{x})$ with domain $\mathbf{R}^n$ and range $(-\infty, \infty]$ is said to be *closed* (or *lower semicontinuous*) if every sublevel set $\{\boldsymbol{x} : f(\boldsymbol{x}) \le c\}$ is closed. Alternative definition is $f(\boldsymbol{x}) \le \liminf_m f(\boldsymbol{x}_m)$ whenever $\lim_m \boldsymbol{x}_m = \boldsymbol{x}$. Another definition is the epigraph $\{(\boldsymbol{x}, y) \in \mathbf{R}^n \times \mathbf{R} : f(\boldsymbol{x}) \le y\}$ is closed. Examples of closed functions are all continuous functions, matrix rank, and set indicators.

- Examples of proximal mapping.

  1. (*Constant function*) $g(\boldsymbol{x}) \equiv c$: $\text{prox}_g(\boldsymbol{x}) = \boldsymbol{x}$.
  2. (*Indicator*) $g(\boldsymbol{x}) = \chi_C(\boldsymbol{x})$: projection operator

  $$\text{prox}_g(\boldsymbol{x}) = \text{argmin}_{\boldsymbol{u}} \left(\chi_C(\boldsymbol{u}) + \frac{1}{2}\|\boldsymbol{u} - \boldsymbol{x}\|_2^2 \right) = P_C(\boldsymbol{x}).$$

  ☞ In this sense, proximal operator generalizes the projection operator to a closed convex set.

  3. (*Lasso*) $g(\boldsymbol{x}) = \lambda \|\boldsymbol{x}\|_1$: soft-thresholding (shrinkage) operator

  $$\begin{aligned} \text{prox}_g(\boldsymbol{x})_i &= \text{argmin}_{u_i} \left(\lambda |u_i| + \frac{1}{2}(u_i - x_i)^2 \right) \\ &= \text{sgn}(x_i)(|x_i| - \lambda)_+. \end{aligned}$$

  Proof. If $u_i \ge 0$, then stationarity condition dictates $u_i = (x_i - \lambda)_+$. If $u_i < 0$, then stationarity condition dictates $u_i = x_i + \lambda = -(-x_i - \lambda)_+$. □

4. (*Group lasso*) $g(\boldsymbol{x}) = \lambda\|\boldsymbol{x}\|_2$: group soft-thresholding

$$
\begin{aligned}
\operatorname{prox}_g(\boldsymbol{x}) &= \operatorname{argmin}_{\boldsymbol{u}} \left(\lambda\|\boldsymbol{u}\|_2 + \frac{1}{2}\|\boldsymbol{u}-\boldsymbol{x}\|_2^2\right) \\
&= \begin{cases} (1-\lambda/\|\boldsymbol{x}\|_2)\boldsymbol{x} & \|\boldsymbol{x}\|_2 \ge \lambda \\ \boldsymbol{0} & \text{otherwise} \end{cases}.
\end{aligned}
$$

Proof. Assuming $\|\boldsymbol{u}\|_2 > 0$, stationarity condition says

$$\boldsymbol{u} - \boldsymbol{x} + \frac{\lambda}{\|\boldsymbol{u}\|_2}\boldsymbol{u} = \boldsymbol{0}$$

or equivalently

$$\left(1 + \frac{\lambda}{\|\boldsymbol{u}\|_2}\right)\boldsymbol{u} = \boldsymbol{x}.$$

Taking ℓ_2 norm on both sides shows $\|\boldsymbol{u}\|_2 = \|\boldsymbol{x}\|_2 - \lambda$. Therefore

$$\boldsymbol{u}^* = \left(1 - \frac{\lambda}{\|\boldsymbol{x}\|_2}\right)\boldsymbol{x}$$

is the global minimum, when $\|\boldsymbol{x}\|_2 \ge \lambda$. For $\|\boldsymbol{x}\|_2 < \lambda$, we have

$$
\begin{aligned}
& \frac{1}{2}\|\boldsymbol{u}-\boldsymbol{x}\|_2^2 + \lambda\|\boldsymbol{u}\|_2 \\
= & \frac{1}{2}\|\boldsymbol{x}\|_2^2 + \frac{1}{2}\|\boldsymbol{u}\|_2^2 - \langle \boldsymbol{u}, \boldsymbol{x}\rangle + \lambda\|\boldsymbol{u}\|_2 \\
\ge & \frac{1}{2}\|\boldsymbol{x}\|_2^2 + \frac{1}{2}\|\boldsymbol{u}\|_2^2 - \|\boldsymbol{u}\|_2\|\boldsymbol{x}\|_2 + \lambda\|\boldsymbol{u}\|_2 \\
= & \frac{1}{2}\|\boldsymbol{x}\|_2^2 + \frac{1}{2}\|\boldsymbol{u}\|_2^2 + \|\boldsymbol{u}\|_2(\lambda - \|\boldsymbol{x}\|_2) \\
\ge & \frac{1}{2}\|\boldsymbol{x}\|_2^2.
\end{aligned}
$$

Therefore $\boldsymbol{u}^* = \boldsymbol{0}$ is the global minimum. □

☞ When $\boldsymbol{x}$ is a scalar, it reduces to the soft thresholding operator $\operatorname{sgn}(x)(|x| - \lambda)_+$.

5. (*Nuclear norm*) $h(\boldsymbol{X}) = \lambda\|\boldsymbol{X}\|_*$: matrix soft-thresholding

$$
\begin{aligned}
\operatorname{prox}_g(\boldsymbol{X}) &= \operatorname{argmin}_{\boldsymbol{Y}} \left(\lambda\|\boldsymbol{Y}\|_* + \frac{1}{2}\|\boldsymbol{Y}-\boldsymbol{X}\|_{\mathrm{F}}^2\right) \\
&= \boldsymbol{U}\operatorname{diag}((\boldsymbol{\sigma}-\lambda)_+)\boldsymbol{V}^T \\
&= S_\lambda(\boldsymbol{X}),
\end{aligned}
$$

where $\boldsymbol{X} = \boldsymbol{U}\operatorname{diag}(\boldsymbol{\sigma})\boldsymbol{V}^T$ is the SVD of $\boldsymbol{X}$. See ST758 (2014 fall) lecture notes p159 for the proof.

6. ...

☞ It is worthwhile to maintain a library of projection and proximal operators in Julia because they form the building blocks of many machine learning algorithms.

- *Proximal gradient algorithm* minimizes the composite function

$$h(\boldsymbol{x}) = f(\boldsymbol{x}) + g(\boldsymbol{x}),$$

where f is convex and differentiable and g is a closed convex function with inexpensive prox-operator by iterating

$$\begin{aligned} \boldsymbol{x}^{(t+1)} &= \operatorname{prox}_{sg}\left(\boldsymbol{x}^{(t)} - s\nabla f(\boldsymbol{x}^{(t)})\right) \\ &= \operatorname{argmin}_{\boldsymbol{u}} \left(g(\boldsymbol{u}) + \frac{1}{2s}\|\boldsymbol{u} - \boldsymbol{x}^{(t)} + s\nabla f(\boldsymbol{x}^{(t)})\|_2^2 \right) \\ &= \operatorname{argmin}_{\boldsymbol{u}} \left(g(\boldsymbol{u}) + f(\boldsymbol{x}^{(t)}) + \nabla f(\boldsymbol{x}^{(t)})^\intercal(\boldsymbol{u} - \boldsymbol{x}^{(t)}) + \frac{1}{2s}\|\boldsymbol{u} - \boldsymbol{x}^{(t)}\|_2^2 \right). \end{aligned}$$

Here s is a constant step size or determined by line search.

☞ Interpretation: from the third line, we see $\boldsymbol{x}^{(t+1)}$ minimizes $g(\boldsymbol{x})$ plus a simple quadratic local model of $f(\boldsymbol{x})$ around $\boldsymbol{x}^{(t)}$.

☞ Interpretation: the function on the third line

$$h(\boldsymbol{x}|\boldsymbol{x}^{(t)}) := g(\boldsymbol{x}) + f(\boldsymbol{x}^{(t)}) + \nabla f(\boldsymbol{x}^{(t)})^\intercal(\boldsymbol{x} - \boldsymbol{x}^{(t)}) + \frac{1}{2s}\|\boldsymbol{x} - \boldsymbol{x}^{(t)}\|_2^2$$

majorizes $f(\boldsymbol{x})+g(\boldsymbol{x})$ at current iterate $\boldsymbol{x}^{(t)}$ when $s \le 1/L$ (why?). Therefore proximal gradient is an MM algorithm as well.

☞ The function to be minimized in each iteration is *separated* in parameters ☺

☞ When g is constant, proximal gradient method reduces to the classical *gradient descent* (or *steepest descent*) method. When g is indicator function $\chi_C(\boldsymbol{x})$, proximal gradient method reduces to the *projected gradient* method.

- Example. Lasso regression

$$\text{minimize} \quad \frac{1}{2}\|\boldsymbol{y} - \beta_0\mathbf{1} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_1,$$

where we identify $f(\boldsymbol{\beta}) = \frac{1}{2}\|\boldsymbol{y} - \beta_0\mathbf{1} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2$ and $g(\boldsymbol{\beta}) = \lambda\|\boldsymbol{\beta}\|_1$. Then the proximal gradient method iterates according to

$$\begin{aligned} \boldsymbol{\beta}^{(t+1)} &= \operatorname{prox}_{sg}(\boldsymbol{\beta}^{(t)} + s\boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}^{(t)})) \\ &= \operatorname{ST}(\boldsymbol{\beta}^{(t)} + s\boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}^{(t)}), s\lambda). \end{aligned}$$

That is we do iterative soft-thresholding. Note the intercept is not penalized so we do not apply soft-thresholding to it.

- Convergence of proximal gradient method.
  - Assumptions
    * f is convex and $\nabla f(\boldsymbol{x})$ is Lipschitz continuous with parameter $L > 0$
    * g is a closed convex function (so that prox_{sg} is well-defined)
    * optimal value $h^* = \inf_{\boldsymbol{x}} h(\boldsymbol{x})$ is finite and attained at $\boldsymbol{x}^*$
  - Theorem: With fixed step size $s = 1/L$,

$$h(\boldsymbol{x}^{(t)}) - h^* \leq \frac{L\|\boldsymbol{x}^{(0)} - \boldsymbol{x}^*\|_2^2}{2t}.$$

    Similar result for backtracking line search without knowing L.
  - Same convergence rate as the classical gradient method for smooth functions: $O(1/\epsilon)$ steps to reach $h(\boldsymbol{x}^{(t)}) - h^* \leq \epsilon$.
  - Q: Can the $O(1/t)$ rate be improved?

23 Lecture 23, Apr 15

Announcements

- HW7 posted `http://hua-zhou.github.io/teaching/st790-2015spr/ST790-2015-HW7.pdf`
- Typo in lecture notes p167 (CD for lasso penalized least squares).

Last Time

- Proximal gradient algorithm.

Today

- Accelerated proximal gradient method.

Accelerated proximal gradient method

- Now we have a powerful tool, the proximal gradient method, for dealing with the non-smooth term in sparse regression. But it converges slowly at the $O(1/t)$ rate ☹ Nesterov comes to the rescue ☺
- History:
  - Nesterov:
    - Nesterov (1983): original acceleration method for smooth functions
    - Nesterov (1988): second acceleration method for smooth functions
    - Nesterov (2005): smoothing techniques for nonsmooth functions, coupled with original acceleration method
    - Nesterov (2007): acceleration for composite functions
  - Beck and Teboulle (2009b): extension of Nesterov (1983) to composite functions (FISTA).
  - Tseng (2008): unified analysis of acceleration techniques (all of these, and more).
- FISTA: **F**ast **I**terative **S**hrinkage-**T**hresholding **A**lgorithm (Beck and Teboulle, 2009b).
  - Minimize

$$h(\boldsymbol{x}) = f(\boldsymbol{x}) + g(\boldsymbol{x}),$$

where f is convex and differentiable and g is convex with inexpensive prox-operator.

- FISTA algorithm: choose any $\boldsymbol{x}^{(0)} = \boldsymbol{x}^{(-1)}$; for $t \geq 1$, repeat

$$
\begin{aligned}
\boldsymbol{y} &\leftarrow \boldsymbol{x}^{(t-1)} + \frac{t-2}{t+1}(\boldsymbol{x}^{(t-1)} - \boldsymbol{x}^{(t-2)}) && \text{(extrapolation)} \\
\boldsymbol{x}^{(t)} &\leftarrow \operatorname{prox}_{sg}(\boldsymbol{y} - s\nabla f(\boldsymbol{y})) && \text{(prox. grad. desc.)}
\end{aligned}
$$

  Step size s is fixed or determined by line search.

  ☞ Interpretation: proximal gradient step is performed on the extrapolated point $\boldsymbol{y}$ based on the previous two iterates.

  ☞ Physical interpretation of Nesterov acceleration? (Pointed to me by Xiang Zhang) `http://cs231n.github.io/neural-networks-3/#sgd`

- Convergence of FISTA.
  - Assumptions
    * f is convex and $\nabla f(\boldsymbol{x})$ is Lipschitz continuous with parameter $L > 0$
    * g is closed convex (so that prox_{sg} is well-defined)
    * optimal value $h^* = \inf_{\boldsymbol{x}} h(\boldsymbol{x})$ is finite and attained at $\boldsymbol{x}^*$
  - Theorem: With fixed step size $s = 1/L$,

$$
h(\boldsymbol{x}^{(t)}) - h^* \leq \frac{L\|\boldsymbol{x}^{(0)} - \boldsymbol{x}^*\|_2^2}{2(t+1)^2}.
$$

    Similar result for backtracking line search.

    ☞ Need $O(1/\sqrt{\epsilon})$ iterations to get $h(\boldsymbol{x}^{(t)}) - h^* \leq \epsilon$. To appreciate this acceleration, to get close to optimal value within $\epsilon = 10^{-4}$, proximal gradient method requires up to 10^4 iterations, while accelerated proximal gradient method requires up to 100 iterations.

- Improvement of convergence rate from $O(1/t)$ to $O(1/t^2)$ is remarkable. Can we do better? Nesterov says no. Formally
  - Assumptions (smooth case)
    * f is convex and differentiable
    * $\nabla f(\boldsymbol{x})$ is Lipschitz continuous with parameter $L > 0$
    * optimal value $f^* = \inf_{\boldsymbol{x}} f(\boldsymbol{x})$ is finite and attained at $\boldsymbol{x}^*$

– *First order method*: any iterative algorithm that selects $\boldsymbol{x}^{(k)}$ in

$$\boldsymbol{x}^{(0)} + \text{span}\{\nabla f(\boldsymbol{x}^{(0)}), \ldots, \nabla f(\boldsymbol{x}^{(k-1)})\}$$

is called a first order method.

– *Problem class*: any function that satisfies the above assumptions.

– Theorem (Nesterov, 1983): for every integer $t \le (n-1)/2$ and every $\boldsymbol{x}^{(0)}$, there exist functions in the problem class such that for any first-order method

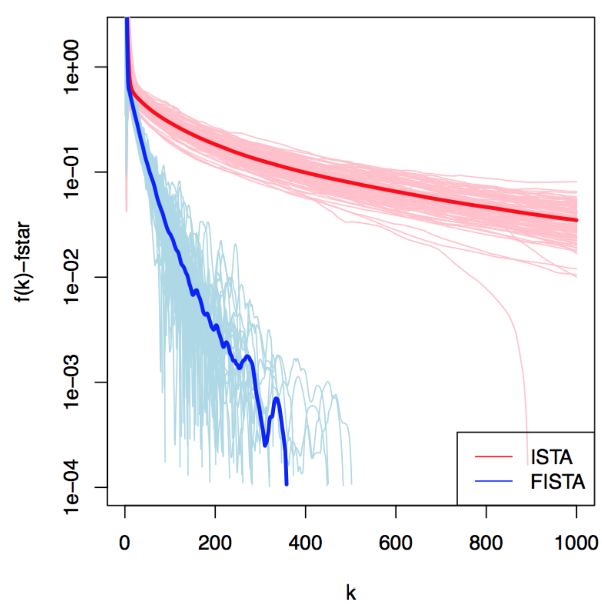
$$f(\boldsymbol{x}^{(t)}) - f^* \ge \frac{3}{32} \frac{L\|\boldsymbol{x}^{(0)} - \boldsymbol{x}^*\|_2^2}{(t+1)^2}$$

☞ This says $O(1/t^2)$ is the best rate first order methods can achieve.

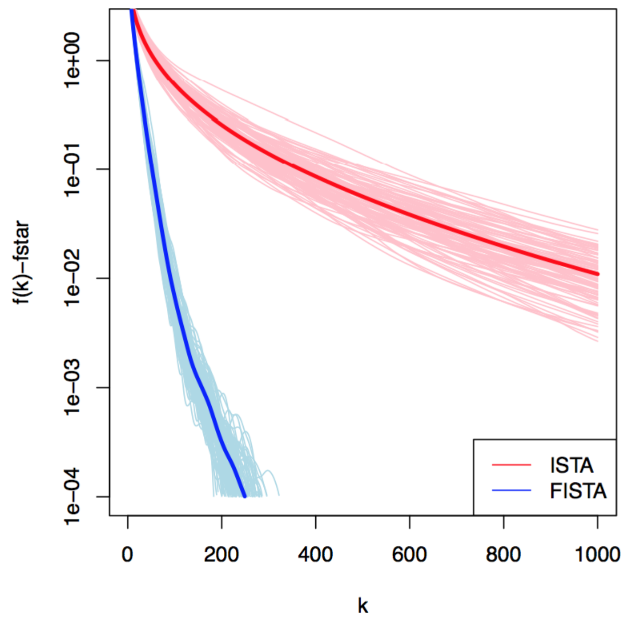
– Nesterov's accelerated gradient method achieves the *optimal* $O(1/t^2)$ *rate* among all first-order methods!

– Similarly FISTA achieves the *optimal* $O(1/t^2)$ *rate* among all first-order methods for minimizing composite function $h(\boldsymbol{x}) = f(\boldsymbol{x}) + g(\boldsymbol{x})$. See (Beck and Teboulle, 2009b) for proof.

- Example. Lasso ($n = 100$, $p = 500$): 100 instances.



- Example. Lasso logistic regression ($n = 100$, $p = 500$): 100 instances.



- Numerous applications of FISTA.
  - Constrained optimization. When the projection to the constraint set C is inexpensive, accelerated projected gradient method applies.
  - Lasso: $f(\boldsymbol{\beta}) + \lambda \sum_{j=1}^{p} |\beta_j|$.
  - Group lasso: $f(\boldsymbol{\beta}) + \lambda \sum_g \|\boldsymbol{\beta}_g\|_2$.
  - Matrix completion: $(1/2)\|P_\Omega(\boldsymbol{A}) - P_\Omega(\boldsymbol{B})\|_{\mathrm{F}}^2 + \lambda \|\boldsymbol{B}\|_*$
    It yields an algorithm different from the MM algorithm we learned in ST758.
  - Regularized matrix regression: $f(\boldsymbol{B}) + \lambda \|\boldsymbol{B}\|_*$ (Zhou and Li, 2014).
  - ...
- Remarks.
  - Whenever we do (proximal) gradient method, use Nesterov's acceleration. It is "free" but makes a big difference in convergence rate.
  - For regularization problems, *warm start* strategy may diminish the need for acceleration.
  - FISTA is *not* a monotone algorithm. See (Beck and Teboulle, 2009a) for a monotone version.
  - In practice the Lipschitz constant L is unknown.
    * Obtain an initial estimate of L using the fact a twice differentiable f has Lipschitz continuous gradient with parameter L iff $L\boldsymbol{I} - d^2 f(\boldsymbol{x})$ is psd for

all $\boldsymbol{x}$ iff the largest eigenvalue of Hessian is bounded above by L. For least squares, we have $L = \lambda_{\max}(\boldsymbol{X}^T\boldsymbol{X})$. For logistic regression, we have $L = 0.25\lambda_{\max}(\boldsymbol{X}^T\boldsymbol{X})$.

* See (Beck and Teboulle, 2009b) for the line search strategy. Same $1/t^2$ convergence rate.

$$
\begin{array}{rcll}
\boldsymbol{y} & \leftarrow & \boldsymbol{x}^{(t-1)} + \dfrac{t-2}{t+1}(\boldsymbol{x}^{(t-1)} - \boldsymbol{x}^{(t-2)}) & \text{(extrapolation)} \\
\text{Repeat} & & & \text{(line search)} \\
\boldsymbol{x}_{\text{temp}} & \leftarrow & \text{prox}_{sg}(\boldsymbol{y} - s\nabla f(\boldsymbol{y})) & \\
s & \leftarrow & s/2 & \\
\text{until} & & h(\boldsymbol{x}_{\text{temp}}) \leq h(\boldsymbol{x}_{\text{temp}}|\boldsymbol{y}) & \\
\boldsymbol{x}^{(t)} & \leftarrow & \boldsymbol{x}_{\text{temp}} &
\end{array}
$$

– For non-convex f, convergence to stationarity point. See (Beck and Teboulle, 2009a, Theorem 1.3).

– Alternative Nesterov acceleration sequence. Original Nesterov acceleration sequence takes the form (starting from $\alpha^{(-2)} = 0, \alpha^{(-1)} = 1$)

$$
\begin{array}{rcll}
\boldsymbol{y} & \leftarrow & \boldsymbol{x}^{(t-1)} + \dfrac{\alpha^{(t-2)} - 1}{\alpha^{(t-1)}}(\boldsymbol{x}^{(t-1)} - \boldsymbol{x}^{(t-2)}) & \text{(extrapolation)} \\
\boldsymbol{x}^{(t)} & \leftarrow & \text{prox}_{sg}(\boldsymbol{y} - s\nabla f(\boldsymbol{y})) & \text{(prox. grad. desc.)} \\
\alpha^{(t)} & \leftarrow & \dfrac{1 + \sqrt{1 + (2\alpha^{(t-1)})^2}}{2}. &
\end{array}
$$

See (Beck and Teboulle, 2009b). Same $O(1/t^2)$ convergence rate.

24 Lecture 24, Apr 20

Announcements

- HW7 due Tue, 4/21 @ 11:59PM.

Last Time

- Accelerated proximal gradient algorithm.

Today

- Path algorithm.
- ALM.

Path algorithm for regularization problems

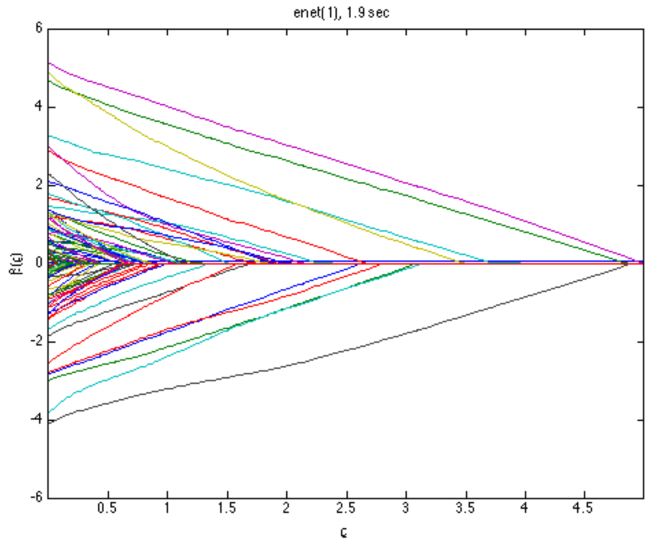
- In statistics and machine learning, regularization problems solve

$$\text{minimize}_{\boldsymbol{\beta}} \quad f(\boldsymbol{\beta}) + \lambda J(\boldsymbol{\beta})$$

  for all $\lambda \geq 0$.

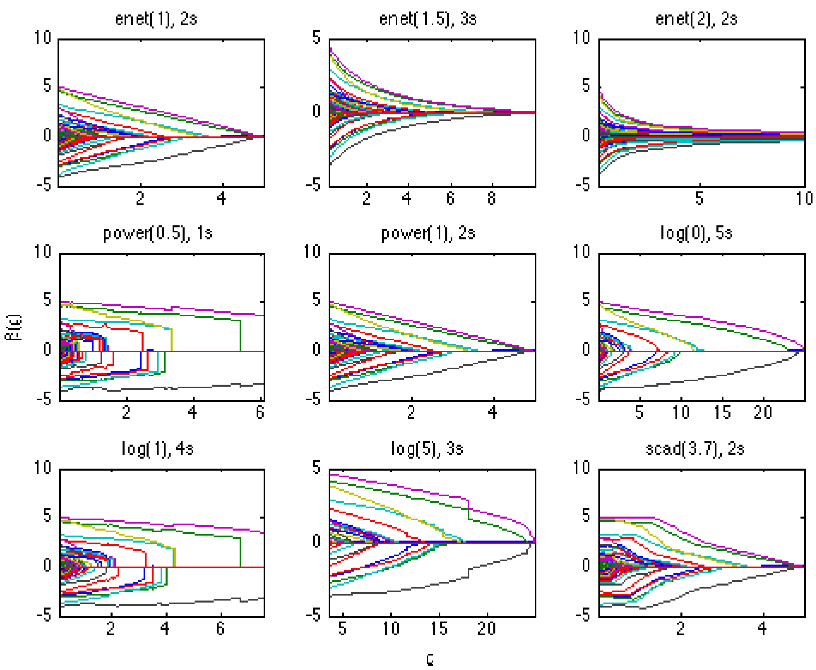
  - λ controls the balance between model fit and model complexity.
  - Most time we seek whole *solution path*, instead of solution at individual λs.
  - Path algorithms trace the solution $\boldsymbol{\beta}(\lambda)$ as a function of λ.
  - Need a principled way to choose λ (model selection).

- Example: Lasso solution path ($n = 500$, $p = 100$)



Observation: the solution path (in terms of λ) is *piece-wise linear*.

- Example: Solution paths with various penalties ($n = 500$, $p = 100$)



Observation: (1) The solution paths are *piece-wise smooth* for convex penalties, (2) but may be *discontinuous* for non-convex penalties.

- How to derive path algorithm? Consider sparse regression $f(\boldsymbol{\beta}) + \sum_{j=1}^{p} P_\eta(|\beta_j|, \lambda)$ with a convex penalty P_η.

1. Write down the Karush-Kuhn-Tucker (KKT) condition for solution $\boldsymbol{\beta}(\lambda)$

$$
\begin{aligned}
0 &= \nabla_j f(\boldsymbol{\beta}) + \nabla_{\beta_j} P_\eta(|\beta_j|, \lambda), \quad \text{for all } \beta_j \neq 0 \\
0 &\in \nabla_j f(\boldsymbol{\beta}) + \partial_{\beta_j} P_\eta(|\beta_j|, \lambda), \quad \text{for all } \beta_j = 0.
\end{aligned}
$$

2. Apply the *implicit function theorem* to the first set of equations to derive the path direction for active β_j and determine when each of them hits zero.
3. Use the second set of equations to determine when a zero coefficient β_j becomes non-zero.

☞ Recall that the *subdifferential* $\partial f(\boldsymbol{x})$ of a convex function $f(\boldsymbol{x})$ is the set of all vectors $\boldsymbol{g}$ satisfying the supporting hyperplane inequality

$$
f(\boldsymbol{y}) \geq f(\boldsymbol{x}) + \boldsymbol{g}^T(\boldsymbol{y} - \boldsymbol{x})
$$

for all $\boldsymbol{y}$. For instance, subdifferential of $f(x) = |x|$ is [-1,1] at $x = 0$. If $f(\boldsymbol{x})$ is differentiable at $\boldsymbol{x}$, then the set $\partial f(\boldsymbol{x})$ reduces to the single vector $\nabla f(\boldsymbol{x})$.

- Example: Lasso (Osborne et al., 2000; Efron et al., 2004)

$$
\hat{\boldsymbol{\beta}}(\lambda) = \arg\min_{\boldsymbol{\beta}} \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 + \lambda \sum_{j=1}^p |\beta_j|.
$$

For simplicity, we assume predictors and responses are centered so omit the intercept. Stationarity condition (necessary and sufficient for global minimum in this case) says

$$
\mathbf{0}_p \in -\boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}) + \lambda \partial \|\boldsymbol{\beta}\|_1.
$$

Let $\mathcal{A} = \{j : \beta_j \neq 0\}$ index the non-zero coefficients. Then we have

$$
\begin{aligned}
\mathbf{0}_{|\mathcal{A}|} &= -\boldsymbol{X}_{\mathcal{A}}^T(\boldsymbol{y} - \boldsymbol{X}_{\mathcal{A}}\boldsymbol{\beta}_{\mathcal{A}}) + \lambda \operatorname{sgn}(\boldsymbol{\beta}_{\mathcal{A}}) \\
-\lambda \mathbf{1}_{|\mathcal{A}^c|} &\preceq -\boldsymbol{X}_{\mathcal{A}^c}^T(\boldsymbol{y} - \boldsymbol{X}_{\mathcal{A}}\boldsymbol{\beta}_{\mathcal{A}}) \preceq \lambda \mathbf{1}_{|\mathcal{A}^c|}.
\end{aligned}
$$

Applying the implicit function theorem to the first set of equations yields the path following direction

$$
\frac{d}{d\lambda}\hat{\boldsymbol{\beta}}_{\mathcal{A}}(\lambda) = -(\boldsymbol{X}_{\mathcal{A}}^T\boldsymbol{X}_{\mathcal{A}})^{-1} \operatorname{sgn}(\boldsymbol{\beta}_{\mathcal{A}}),
$$

which effectively shows that non-zero coefficients $\hat{\boldsymbol{\beta}}_{\mathcal{A}}(\lambda)$ and thus the subgradient vector $-\boldsymbol{X}_{\mathcal{A}^c}^T(\boldsymbol{y} - \boldsymbol{X}_{\mathcal{A}}\hat{\boldsymbol{\beta}}_{\mathcal{A}}(\lambda))$ moves linearly within a segment. The second set of equations monitor the events a zero coefficient becomes non-zero. Therefore for each β_j, $j \in \mathcal{A}$, we calculate when it (ever) hits 0. And for each β_j, $j \in \mathcal{A}^c$, we calculate when it becomes zero. Then the end of current segment (or start of next segment) is determined by the event that happens soonest, where we update $\mathcal{A}$ and then continues.

The computational cost per segment is $O(|\mathcal{A}|^2)$. The number of segments is harder to characterize though (Donoho and Tanner, 2010). Under certain conditions whole (piece-wise linear) solution path is obtained at the cost of a regular least squares fit (Efron et al., 2004).

- Example: Generalized lasso (Tibshirani and Taylor, 2011; Zhou and Lange, 2013)

$$\frac{1}{2}\|\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\beta}\|_2^2+\lambda\|\boldsymbol{V}\boldsymbol{\beta}-\boldsymbol{d}\|_1+\lambda\|\boldsymbol{W}\boldsymbol{\beta}-\boldsymbol{e}\|_+.$$

  Piece-wise linear path. Applications include lasso, fused lasso, polynomial trend filtering, image denoising, ...

- Example: Support vector machine (Hastie et al., 2004)

$$\min_{\beta_0,\boldsymbol{\beta}}\sum_{i=1}^n[1-y_i(\beta_0+\boldsymbol{x}_i^{\mathsf{T}}\boldsymbol{\beta})]_+ + \frac{\lambda}{2}\|\boldsymbol{\beta}\|_2^2.$$

  Piece-wise linear path.

- Example: Quantile regression and many more piece-wise linear solution paths (Rosset and Zhu, 2007).

- Example: GLM lasso

$$f(\boldsymbol{\beta})+\lambda\sum_{j=1}^p|\beta_j|.$$

  Approximate path algorithm (Park and Hastie, 2007) and exact path algorithm (Wu, 2011; Zhou and Wu, 2014) using ODE.

- Example: Convex generalized lasso (Zhou and Wu, 2014)

$$f(\boldsymbol{\beta})+\lambda\|\boldsymbol{V}\boldsymbol{\beta}-\boldsymbol{d}\|_1+\lambda\|\boldsymbol{W}\boldsymbol{\beta}-\boldsymbol{e}\|_+.$$

  Applications include GLM (generalized) lasso, non-parametric density estimation, Gaussian graphical lasso, ...

- A very general path algorithm presented by Friedman (2008) works for a large class of convex/concave penalties, but is mysterious ☹.

- Tuning parameter selection.

  - λ balances the model fit and model complexity.

- Choosing λ is critical in statistical applications.
- Commonly used methods
  * Cross validation
  * Information criteria:

$$
\begin{aligned}
\text{AIC}(\lambda) &= \frac{\|\boldsymbol{y} - \hat{\boldsymbol{y}}(\lambda)\|^2}{\sigma^2} + 2\text{df}(\lambda) \\
\text{BIC}(\lambda) &= \frac{\|\boldsymbol{y} - \hat{\boldsymbol{y}}(\lambda)\|^2}{\sigma^2} + \ln(n)\text{df}(\lambda),
\end{aligned}
$$

    where $\hat{\boldsymbol{y}}(\lambda) = \boldsymbol{X}\hat{\boldsymbol{\beta}}(\lambda)$ and $\text{df}(\lambda)$ is the effective *degrees of freedom* of the selected model at λ
- Using Stein (1981)'s theory of unbiased risk estimation (SURE), Efron (2004) shows

$$
\text{df}(\lambda) = \frac{1}{\sigma^2} \sum_{i=1}^{n} \text{cov}(\hat{y}_i(\lambda), y_i) = \mathbf{E}\left[\text{tr}\left(\frac{\partial \hat{\boldsymbol{y}}(\lambda)}{\partial \boldsymbol{y}}\right)\right]
$$

  under differentiability condition on the mapping $\hat{\boldsymbol{y}}(\lambda)$.
  * least squares estimate: $\text{df} = \text{tr}(\boldsymbol{X}(\boldsymbol{X}^\top \boldsymbol{X})^{-1}\boldsymbol{X}^\top) = p$
  * ridge: $\text{df}(\lambda) = \text{tr}(\boldsymbol{X}(\boldsymbol{X}^\top \boldsymbol{X} + \lambda \boldsymbol{I})^{-1}\boldsymbol{X}^\top) = \sum_{j=1}^{p} d_j^2/(d_j^2 + \lambda)$, where d_j are singular values of $\boldsymbol{X}$
  * lasso (Zou et al., 2007): number of non-zero coefficients
  * generalized lasso (Tibshirani and Taylor, 2011)
  * group lasso (Yuan and Lin, 2006)
  * nuclear norm regularization (Zhou and Li, 2014)
  * ...

Augmented Lagrangian method (ALM)

- ALM is also called the method of multipliers.
- Consider optimization problem

$$
\begin{aligned}
&\text{minimize} \quad f(\boldsymbol{x}) \\
&\text{subject to} \quad g_i(\boldsymbol{x}) = 0, \quad i = 1, \ldots, q.
\end{aligned}
$$

  - Inequality constraints are ignored for simplicity.
  - Assume f and g_i are smooth for simplicity.

– At a constrained minimum, the Lagrange multiplier condition

$$\mathbf{0} = \nabla f(\boldsymbol{x}) + \sum_{i=1}^{q} \lambda_i \nabla g_i(\boldsymbol{x})$$

holds provided $\nabla g_i(\boldsymbol{x})$ are linearly independent.

- *Augmented Lagrangian*:

$$\mathcal{L}_\rho(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + \sum_{i=1}^{q} \lambda_i g_i(\boldsymbol{x}) + \frac{\rho}{2} \sum_{i=1}^{q} g_i(\boldsymbol{x})^2.$$

  – The penalty term $(\rho/2) \sum_{i=1}^{q} g_i(\boldsymbol{x})^2$ punishes violations of the equality constraints $g_i(\boldsymbol{\theta})$.

  – *Idea*: optimize the Augmented Lagrangian and adjust $\boldsymbol{\lambda}$ in the hope of matching the true Lagrange multipliers.

  – For ρ large enough (but finite), the unconstrained minimizer of the augmented Lagrangian coincides with the constrained solution of the original problem.

  – At convergence, the gradient $\rho g_i(\boldsymbol{x}) \nabla g_i(\boldsymbol{x})$ vanishes and we recover the standard multiplier rule.

- *Algorithm*: take ρ initially large or gradually increase it; iterate

  – find the unconstrained minimum

$$\boldsymbol{x}^{(t+1)} \leftarrow \min_{\boldsymbol{x}} \mathcal{L}_\rho(\boldsymbol{x}, \boldsymbol{\lambda}^{(t)})$$

  – update the multiplier vector $\boldsymbol{\lambda}$

$$\lambda_i^{(t+1)} \leftarrow \lambda_i^{(t)} + \rho g_i(\boldsymbol{x}^{(t)}), \quad i = 1, \ldots, q.$$

☞ Intuition for updating $\boldsymbol{\lambda}$: if $\boldsymbol{x}^{(t)}$ is the unconstrained minimum of $\mathcal{L}_\rho(\boldsymbol{x}, \boldsymbol{\lambda})$, then the stationarity condition says

$$\begin{aligned} \mathbf{0} &= \nabla f(\boldsymbol{x}^{(t)}) + \sum_{i=1}^{q} \lambda_i^{(t)} \nabla g_i(\boldsymbol{x}^{(t)}) + \rho \sum_{i=1}^{q} g_i(\boldsymbol{x}^{(t)}) \nabla g_i(\boldsymbol{x}^{(t)}) \\ &= \nabla f(\boldsymbol{x}^{(t)}) + \sum_{i=1}^{q} [\lambda_i^{(t)} + \rho g_i(\boldsymbol{x}^{(t)})] \nabla g_i(\boldsymbol{x}^{(t)}). \end{aligned}$$

☞ For non-smooth f, replace gradient ∇f by subdifferential ∂f.

- Example: *Compressed sensing* (or *basis pursuit*) problem seeks the sparsest solution subject to linear constraints

$$\begin{array}{ll} \text{minimize} & \|\boldsymbol{x}\|_1 \\ \text{subject to} & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}. \end{array}$$

  Take ρ initially large or gradually increase it; iterate according to

$$\begin{aligned} \boldsymbol{x}^{(t+1)} &\leftarrow \min \|\boldsymbol{x}\|_1 + \langle \boldsymbol{\lambda}^{(t)}, \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b} \rangle + \frac{\rho}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 \quad \text{(lasso)} \\ \boldsymbol{\lambda}^{(t+1)} &\leftarrow \boldsymbol{\lambda}^{(t)} + \rho(\boldsymbol{A}\boldsymbol{x}^{(t+1)} - \boldsymbol{b}). \end{aligned}$$

  Converges in a finite (small) number of steps (Yin et al., 2008)

- The matrix completion problem (HW6 Q2)

$$\begin{array}{ll} \text{minimize} & \|\boldsymbol{X}\|_* \\ \text{subject to} & x_{ij} = y_{ij}, \quad (i,j) \in \boldsymbol{\Omega} \end{array}$$

  can be solved by ALM as well. It leads to an iterative singular value thresholding procedure (Cai et al., 2010), which scales to very large problems.

- Remarks on ALM:

  - History: The augmented Lagrangian method dates back to 50s (Hestenes, 1969; Powell, 1969).

    Without the quadratic penalty term $(\rho/2)\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\|_2^2$, it is the classical *dual ascent* algorithm. Dual ascent algorithm works under a set of restrictive assumptions and can be slow. ALM converges under much more relaxed assumptions (f can be non differentiable, takes value ∞, ...)

  - Monograph by Bertsekas (1982) provides a general treatment.

  - Same as the *Bregman iteration* (Yin et al., 2008) for basis pursuit (compressive sensing).

  - Equivalent to proximal point algorithm applied to the dual; can be accelerated (Nesterov).

25 Lecture 25, Apr 22

Announcements

- Course project due Wed, 4/29 @ 11:00AM.

Last Time

- Path algorithm.
- ALM (augmented Lagrangian method) or method of multipliers.

Today

- ADMM (alternating direction method of multipliers). A generic method for solving many regularization problems.
- Dynamic programming: hidden Markov model, some fused lasso problems.
- HW7 solution sketch in Julia. `http://hua-zhou.github.io/teaching/st790-2015spr/hw07sol.html`

ADMM

☞ A definite resource for learning ADMM is (Boyd et al., 2011)
`http://stanford.edu/~boyd/admm.html`

- **A**lternating **d**irection **m**ethod of **m**ultipliers (ADMM).
  - Consider optimization problem

$$\begin{aligned} &\text{minimize} \quad f(\boldsymbol{x}) + g(\boldsymbol{y}) \\ &\text{subject to} \quad \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{y} = \boldsymbol{c}. \end{aligned}$$

  - The augmented Lagrangian

$$\mathcal{L}_\rho(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + g(\boldsymbol{y}) + \langle \boldsymbol{\lambda}, \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{y} - \boldsymbol{c} \rangle + \frac{\rho}{2}\|\boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{y} - \boldsymbol{c}\|_2^2.$$

  - *Idea*: perform block descent on $\boldsymbol{x}$ and $\boldsymbol{y}$ and then update multiplier vector $\boldsymbol{\lambda}$

$$\begin{aligned} \boldsymbol{x}^{(t+1)} &\leftarrow \min_{\boldsymbol{x}} f(\boldsymbol{x}) + \langle \boldsymbol{\lambda}^{(t)}, \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{y}^{(t)} - \boldsymbol{c} \rangle + \frac{\rho}{2}\|\boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{y}^{(t)} - \boldsymbol{c}\|_2^2 \\ \boldsymbol{y}^{(t+1)} &\leftarrow \min_{\boldsymbol{y}} g(\boldsymbol{y}) + \langle \boldsymbol{\lambda}^{(t)}, \boldsymbol{A}\boldsymbol{x}^{(t+1)} + \boldsymbol{B}\boldsymbol{y} - \boldsymbol{c} \rangle + \frac{\rho}{2}\|\boldsymbol{A}\boldsymbol{x}^{(t+1)} + \boldsymbol{B}\boldsymbol{y} - \boldsymbol{c}\|_2^2 \\ \boldsymbol{\lambda}^{(t+1)} &\leftarrow \boldsymbol{\lambda}^{(t)} + \rho(\boldsymbol{A}\boldsymbol{x}^{(t+1)} + \boldsymbol{B}\boldsymbol{y}^{(t+1)} - \boldsymbol{c}) \end{aligned}$$

☞ If we minimize $\boldsymbol{x}$ and $\boldsymbol{y}$ jointly, then it is same as ALM. We gain splitting by blockwise updates.

- ADMM converges under mild conditions: f, g convex, closed, and proper, $\mathcal{L}_0$ has a saddle point.

- Example: *Generalized lasso* problem minimizes

$$\frac{1}{2}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 + \mu\|\boldsymbol{D}\boldsymbol{\beta}\|_1.$$

  - Special case $\boldsymbol{D} = \boldsymbol{I}_p$ corresponds to *lasso*. Special case

$$\boldsymbol{D} = \begin{pmatrix} 1 & -1 & & & \\ & & \cdots & & \\ & & & 1 & -1 \end{pmatrix}$$

  corresponds to *fused lasso*. Numerous applications.

  - Define $\boldsymbol{\gamma} = \boldsymbol{D}\boldsymbol{\beta}$. Then we solve

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 + \mu\|\boldsymbol{\gamma}\|_1 \\ \text{sujbect to} & \boldsymbol{D}\boldsymbol{\beta} = \boldsymbol{\gamma}. \end{array}$$

  - Augmented Lagrangian is

$$\mathcal{L}_\rho(\boldsymbol{\beta}, \boldsymbol{\gamma}, \boldsymbol{\lambda}) = \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 + \mu\|\boldsymbol{\gamma}\|_1 + \boldsymbol{\lambda}^\intercal(\boldsymbol{D}\boldsymbol{\beta} - \boldsymbol{\gamma}) + \frac{\rho}{2}\|\boldsymbol{D}\boldsymbol{\beta} - \boldsymbol{\gamma}\|_2^2.$$

  - ADMM algorithm:

$$\begin{aligned} \boldsymbol{\beta}^{(t+1)} &\leftarrow \min_{\boldsymbol{\beta}} \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 + \boldsymbol{\lambda}^{(t)T}(\boldsymbol{D}\boldsymbol{\beta} - \boldsymbol{\gamma}^{(t)}) + \frac{\rho}{2}\|\boldsymbol{D}\boldsymbol{\beta} - \boldsymbol{\gamma}^{(t)}\|_2^2 \\ \boldsymbol{\gamma}^{(t+1)} &\leftarrow \min_{\boldsymbol{\gamma}} \mu\|\boldsymbol{\gamma}\|_1 + \boldsymbol{\lambda}^\intercal(\boldsymbol{D}\boldsymbol{\beta}^{(t+1)} - \boldsymbol{\gamma}) + \frac{\rho}{2}\|\boldsymbol{D}\boldsymbol{\beta}^{(t+1)} - \boldsymbol{\gamma}\|_2^2 \\ \boldsymbol{\lambda}^{(t+1)} &\leftarrow \boldsymbol{\lambda}^{(t)} + \rho(\boldsymbol{D}\boldsymbol{\beta}^{(t+1)} - \boldsymbol{\gamma}^{(t+1)}) \end{aligned}$$

  ☞ Update $\boldsymbol{\beta}$ is a smooth quadratic problem. Note the Hessian keeps constant between iterations, therefore its inverse (or decomposition) can be calculated just once, cached in memory, and re-used in each iteration.

  ☞ Update $\boldsymbol{\gamma}$ is a separated lasso problem (elementwise soft-thresholding).

- Remarks on ADMM:

  - Related algorithms

    * *split Bregman iteration* (Goldstein and Osher, 2009)
    * Dykstra (1983)'s alternating projection algorithm
    * ...

  Proximal point algorithm applied to the dual.

  - Numerous applications in statistics and machine learning: lasso, generalized lasso, graphical lasso, (overlapping) group lasso, ...
  - Embraces distributed computing for big data (Boyd et al., 2011).

- Distributed computing with ADMM. Consider, for example, solving lasso with a huge training data set $(\boldsymbol{X}, \boldsymbol{y})$, which is distributed on B machines. Denote the distributed data sets by $(\boldsymbol{X}_1, \boldsymbol{y}_1), \ldots, (\boldsymbol{X}_B, \boldsymbol{y}_B)$. Then the lasso criterion is

$$\frac{1}{2}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 + \mu\|\boldsymbol{\beta}\|_1 = \frac{1}{2}\sum_{b=1}^{B} \|\boldsymbol{y}_b - \boldsymbol{X}_b\boldsymbol{\beta}\|_2^2 + \mu\|\boldsymbol{\beta}\|_1.$$

  The ADMM form is

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\sum_{b=1}^{B} \|\boldsymbol{y}_b - \boldsymbol{X}_b\boldsymbol{\beta}_b\|_2^2 + \mu\|\boldsymbol{\beta}\|_1 \\ \text{subject to} \quad & \boldsymbol{\beta}_b = \boldsymbol{\beta}, \quad b = 1, \ldots, B. \end{aligned}$$

  Here $\boldsymbol{\beta}_b$ are local variables and $\boldsymbol{\beta}$ is the global (or consensus) variable. The augmented Lagrangian function is

$$\mathcal{L}_\rho(\boldsymbol{\beta}, \boldsymbol{\gamma}, \boldsymbol{\lambda}) = \frac{1}{2}\sum_{b=1}^{B} \|\boldsymbol{y}_b - \boldsymbol{X}_b\boldsymbol{\beta}_b\|_2^2 + \mu\|\boldsymbol{\beta}\|_1 + \sum_{b=1}^{B} \boldsymbol{\lambda}_b^{\mathsf{T}}(\boldsymbol{\beta}_b - \boldsymbol{\beta}) + \frac{\rho}{2}\sum_{b=1}^{B} \|\boldsymbol{\beta}_b - \boldsymbol{\beta}\|_2^2.$$

  The ADMM algorithm runs as

  - Update local variables $\boldsymbol{\beta}_b$

$$\boldsymbol{\beta}_b^{(t+1)} \quad \leftarrow \quad \min \frac{1}{2}\|\boldsymbol{y}_b - \boldsymbol{X}_b\boldsymbol{\beta}_b\|_2^2 + \boldsymbol{\lambda}_b^{(t)T}(\boldsymbol{\beta}_b - \boldsymbol{\beta}^{(t)}) + \frac{\rho}{2}\|\boldsymbol{\beta}_b - \boldsymbol{\beta}^{(t)}\|_2^2, \; b = 1, \ldots, B,$$

    in parallel on B machines.

  - Collect local variables $\boldsymbol{\beta}_b^{(t)}$, $b = 1, \ldots, B$, and update consensus variable $\boldsymbol{\beta}$

$$\boldsymbol{\beta}^{(t+1)} \quad \leftarrow \quad \min \mu\|\boldsymbol{\beta}\|_1 + \sum_{b=1}^{B} \boldsymbol{\lambda}_b^{(t)T}(\boldsymbol{\beta}_b^{(t+1)} - \boldsymbol{\beta}) + \frac{\rho}{2}\sum_{b=1}^{B} \|\boldsymbol{\beta}_b^{(t+1)} - \boldsymbol{\beta}\|_2^2$$

    by elementwise soft-thresholding.

  - Update multipliers

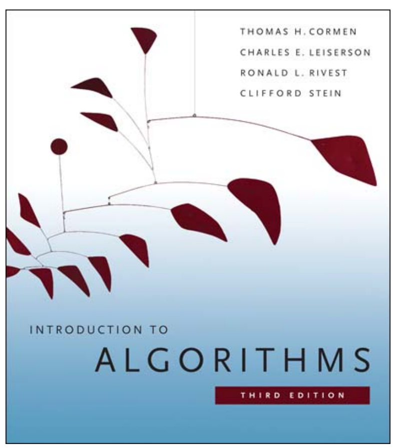
$$\boldsymbol{\lambda}_b^{(t+1)} \leftarrow \boldsymbol{\lambda}_b^{(t)} + \rho(\boldsymbol{\beta}_b^{(t+1)} - \boldsymbol{\beta}^{(t+1)}), \quad b = 1, \ldots, B.$$

☞ The whole procedure is carried out without ever transferring distributed data sets $(\boldsymbol{y}_b, \boldsymbol{X}_b)$ to a central location!

Dynamic programming: introduction

- *Divide-and-conquer*: break the problem into smaller *independent* subproblems
  - fast sorting,
  - FFT,
  - ...
- *Dynamic programming* (DP): subproblems are not independent, that is, subproblems share common subproblems.
- DP solves these subproblems once and store them in a table.
- Use these optimal solutions to construct an optimal solution for the original problem.
- Richard Bellman began the systematic study of DP in 50s.
- Some classical (non-statistical) DP problems:
  - Matrix-chain multiplication,
  - Longest common subsequence,
  - Optimal binary search trees,
  - ...

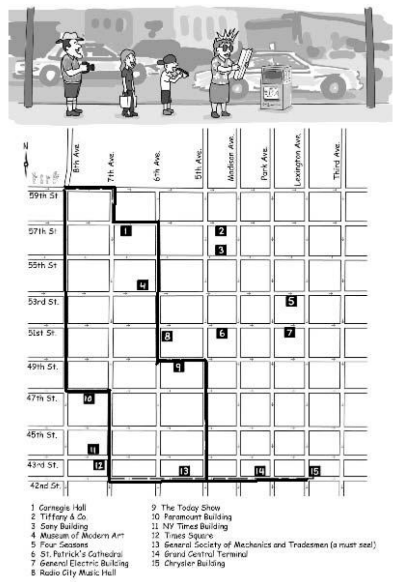
  See (Cormen et al., 2009) for a general introduction



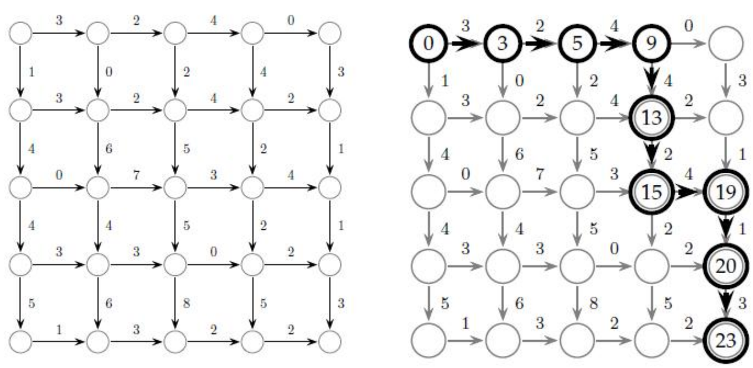
- Some classical DP problems in statistics
  - Hidden Markov model (HMM),
  - Some fused-lasso problems,

- Graphical models (Wainwright and Jordan, 2008),
- Sequence alignment, e.g., discovery of the cystic fibrosis gene in 1989,
- ...

- Let's work on the a DP algorithm for the Manhattan tourist problem (MTP), taken from Jones and Pevzner (2004, Section 6.3).



- MTP: weighted graph

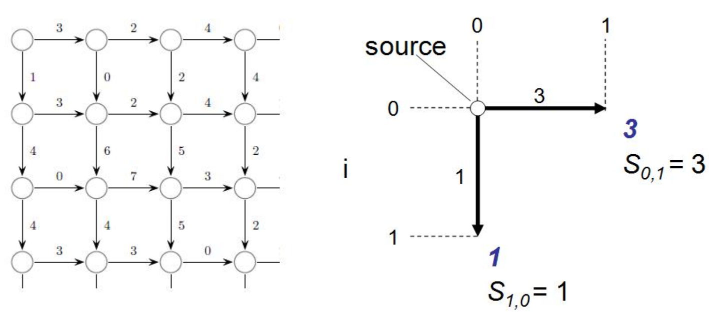


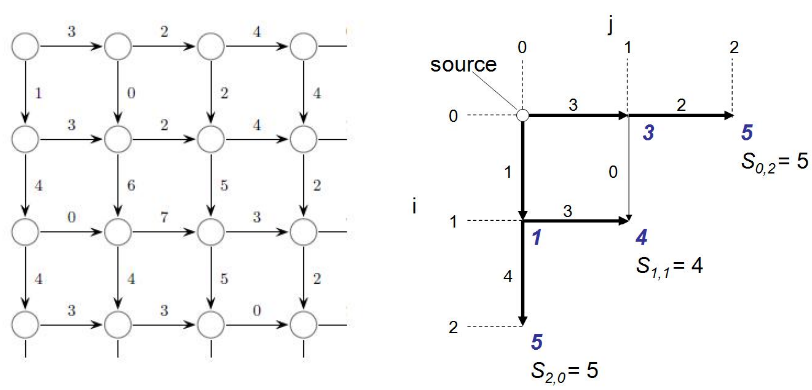
Find a longest path in a weighted grid (only eastward and southward)

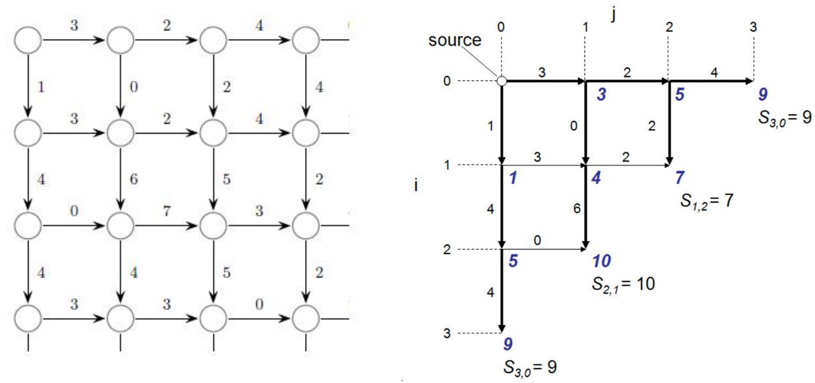
- *Input*: a weighted grid G with two distinguished vertices: a source $(0, 0)$ and a sink (n, m).
- *Output*: a longest path $MT(n, m)$ in G from source to sink.

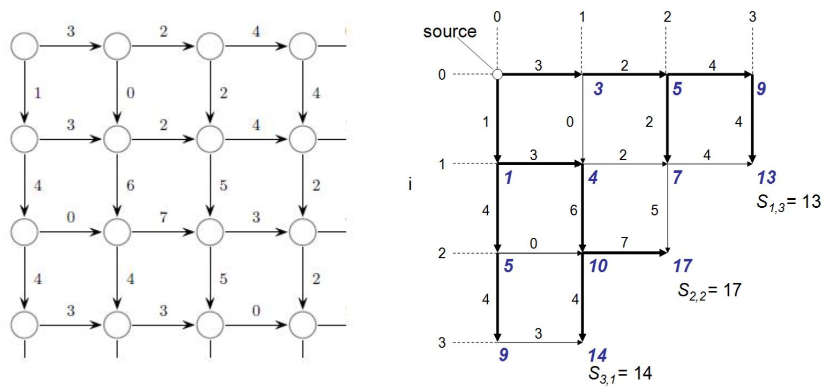
Brute force enumeration is out of the question even for a moderate sized graph.

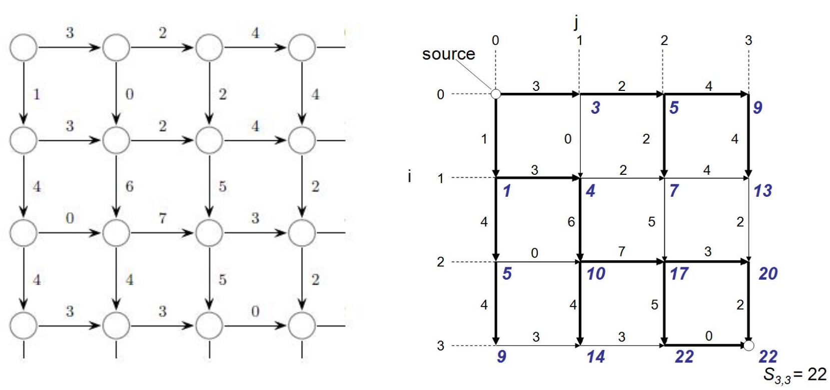
- Simple recursive program.
  $MT(n, m)$:
  - If $n = 0$ or $m = 0$, return $MT(0, 0)$
  - $x \leftarrow MT(n-1, m)+$ weight of the edge from $(n-1, m)$ to (n, m)
    $y \leftarrow MT(n, m-1)+$ weight of the edge from $(n, m-1)$ to (n, m)
  - Return $\max\{x, y\}$
- Something wrong
  - $MT(n, m-1)$ needs $MT(n-1, m-1)$, so as $MT(n-1, m)$.
  - So $MT(n-1, m-1)$ will be computed at least twice.
  - Dynamic programming: the same idea as this recursive algorithm, but keep all intermediate results in a *table* and reuse.
- MTP: dynamic programming
  - Calculate optimal path score for each vertex in the graph
  - Each vertex's score is the maximum of the previous vertices score plus the weight of the respective edge in between



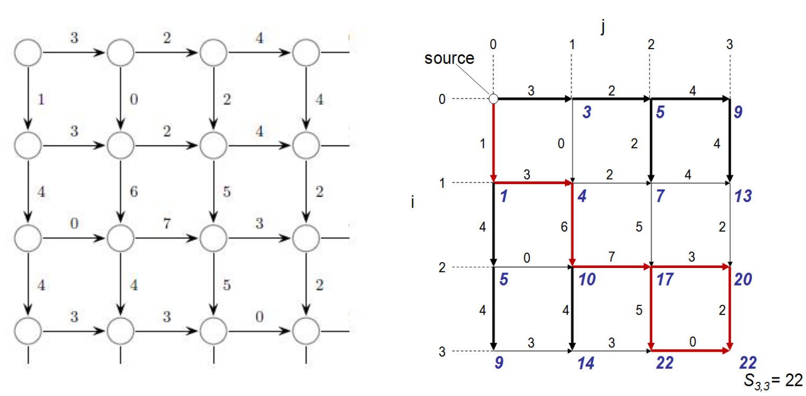








- MTP dynamic programming: *path!*



Showing all back-traces!

- MTP: recurrence
  - Computing the score for a point (i, j) by the recurrence relation:

$$s(i,j) = \max \begin{cases} s(i-1,j) + \text{weight between } (i-1,j) \text{ and } (i,j) \\ s(i,j-1) + \text{weight between } (i,j-1) \text{ and } (i,j) \end{cases}$$

  - The *run time* is mn for a n by m grid.
    (n = number of rows, m = number of columns)

- Remarks on DP:
  - Steps for developing a DP algorithm
    1. Characterize the structure of an optimal solution
    2. Recursively define the value of an optimal solution
    3. Computer the value of an optimal solution in a bottom-up fashion
    4. Construct an optimal solution from computed information
  - "*Programming*" both here and in linear programming refers to the use of a *tabular* solution method.
  - Many problems involve large tables and entries along certain directions may be filled out in parallel – fine scale *parallel computing*.

Application of dynamic programming: HMM

- Hidden Markov model (HMM) (Baum et al., 1970).
  - HMM is a Markov chain that emits symbols:
    Markov chain $(\mu, A = \{a_{kl}\})$ + emission probabilities $e_k(b)$
  - The *state sequence* $\pi = \pi_1 \cdots \pi_L$ is governed by the Markov chain

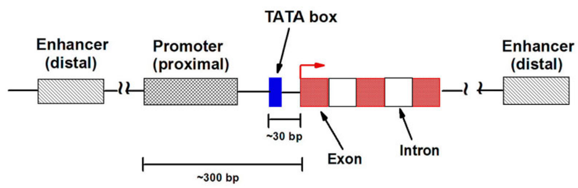
$$\mathbf{P}(\pi_1 = k) = \mu(k), \quad \mathbf{P}(\pi_i = l | \pi_{i-1} = k) = a_{kl}.$$

  - The *symbol sequence* $\mathbf{x} = x_1 \cdots x_L$ is determined by the underlying state sequence π

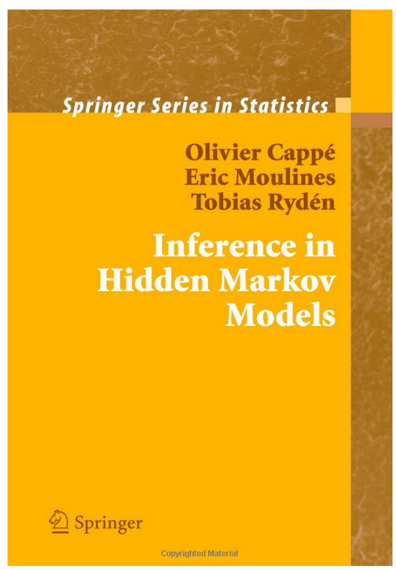
$$\mathbf{P}(\mathbf{x}, \pi) = \prod_{i=1}^{L} e_{\pi_i}(x_i) a_{\pi_{i-1}\pi_i}.$$

  - It is called *hidden* because in applications the state sequence is unobserved.

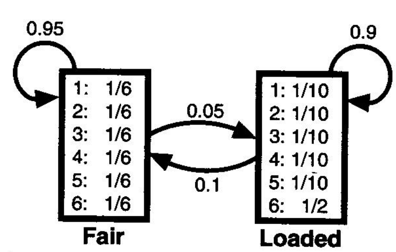
- Wide applications of HMM.
  - Wireless communication: IEEE 802.11 WLAN.
  - Mobile communication: CDMA and GSM.
  - Speech recognition (Rabiner, 1989)
    Hidden states: text, symbols: acoustic signals.
  - Haplotyping and genotype imputation
    Hidden states: haplotypes, symbols: genotypes.
  - Gene prediction (Burge, 1997)



- General reference book on HMM:



- Let's work on a simple HMM example. The Occasionally Dishonest Casino (Durbin et al., 2006)



- Fundamental questions of HMM:

```
Rolls (Observed data)  315423531425413251463612662616 4...
Die (Hidden states)    FFFFFFFFFFFFFFFFFFLLLLLLLLLLLL...
```

  - How to compute the probability of the observed sequence of symbols given known parameters a_{kl} and $e_k(b)$?
    Answer: *Forward algorithm.*
  - How to compute the posterior probability of the state at a given position (posterior decoding) given a_{kl} and $e_k(b)$?
    Answer: *Backward algorithm.*
  - How to estimate the parameters a_{kl} and $e_k(b)$?
    Answer: *Baum-Welch algorithm.*
  - How to find the most likely sequence of hidden states?
    Answer: *Viterbi algorithm* (Viterbi, 1967).

- Forward algorithm:

  - Calculate the *probability of an observed sequence*

$$\mathbf{P}(\mathbf{x}) = \sum_{\pi} \mathbf{P}(\mathbf{x}, \pi).$$

  - Brute force evaluation by enumerating is *impractical*
  - Define the *forward variable*

$$f_k(i) = \mathbf{P}(x_1 \dots x_i, \pi_i = k).$$

  - Recursion formula for forward variables

$$f_l(i+1) = \mathbf{P}(x_1 \dots x_i x_{i+1}, \pi_{i+1} = l) = e_l(x_{i+1}) \sum_k f_k(i) a_{kl}.$$

  - Algorithm:
    * Initialization ($i = 1$): $f_k(1) = a_{0k} e_k(x_1)$.
    * Recursion ($i = 2, \dots, L$): $f_l(i) = e_l(x_i) \sum_k f_k(i-1) a_{kl}$.
    * Termination: $\mathbf{P}(\mathbf{x}) = \sum_k f_k(L)$.

    Time complexity = (# states)^2 $\times$ length of sequence.

- Backward algorithm.

– Calculate the *posterior state probabilities at each position*

$$\mathbf{P}(\pi_i = k|\mathbf{x}) = \frac{\mathbf{P}(\mathbf{x}, \pi_i = k)}{\mathbf{P}(\mathbf{x})}.$$

– Enough to calculate the numerator

$$\begin{aligned}
\mathbf{P}(\mathbf{x}, \pi_i = k) &= \mathbf{P}(x_1 \dots x_i, \pi_i = k)\mathbf{P}(x_{i+1} \dots x_L | x_1 \dots x_i, \pi_i = k) \\
&= \mathbf{P}(x_1 \dots x_i, \pi_i = k)\mathbf{P}(x_{i+1} \dots x_L | \pi_i = k) \\
&= f_k(i) b_k(i).
\end{aligned}$$

– Recursion formula for the *backward variables*

$$b_k(i) = \mathbf{P}(x_{i+1} \dots x_L | \pi_i = k) = \sum_l a_{kl} e_l(x_{i+1}) b_l(i+1).$$

– Algorithm:

  * Initialization ($i = L$): $b_k(L) = 1$ for all k
  * Recursion ($i = L-1, \dots, 1$): $b_k(i) = \sum_l a_{kl} e_l(x_{i+1}) b_l(i+1)$
  * Termination: $\mathbf{P}(\mathbf{x}) = \sum_l a_{0l} e_l(x_1) b_l(1)$

Time complexity = $(\#\text{ states})^2 \times$ length of sequence

– The Occasionally Dishonest Casino.

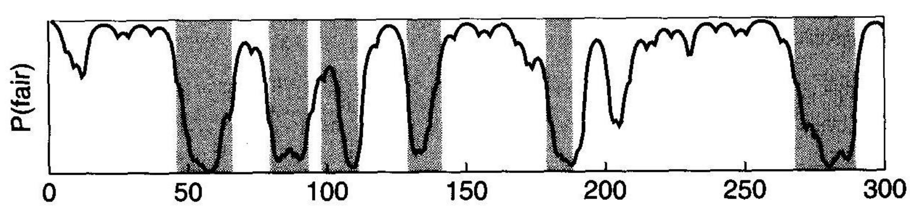


Figure 3.6 ***The posterior probability of being in the state corresponding to the fair die in the casino example. The x axis shows the number of the roll. The shaded areas show when the roll was generated by the loaded die.***

- Parameter estimation for HMM – Baum-Welch algorithm.
- Question: Given n independent training symbol sequences $\mathbf{x}^1, \dots, \mathbf{x}^n$, how to find the parameter value that maximizes the log-likelihood $\log \mathbf{P}(\mathbf{x}^1, \dots, \mathbf{x}^n|\theta) = \sum_{j=1}^n \log \mathbf{P}(\mathbf{x}^j|\theta)$?

  – When the underlying state sequences are *known*: Simple.

  – When the underlying state sequences are *unknown*: Baum-Welch algorithm.

- MLE when state sequences are known.
  - Let $A_{kl} = \#$ transitions from state k to l
    $E_k(b) = \#$ state k emitting symbol b
    The MLEs are

$$a_{kl} = \frac{A_{kl}}{\sum_{l'} A_{kl'}} \text{ and } e_k(b) = \frac{E_k(b)}{\sum_{b'} E_k(b')}. \tag{1}$$

  - To avoid overfitting with insufficient data, add pseudocounts

$$\begin{aligned} A_{kl} &= \# \text{ transitions } k \text{ to } l \text{ in training data } + r_{kl}; \\ E_k(b) &= \# \text{ emissions of } b \text{ from } k \text{ in training data } + r_k(b) \end{aligned}$$

- MLE when state sequences are unknown: Baum-Welch algorithm.
  - *Idea*: Replace the counts A_{kl} and $E_k(b)$ by their expectations conditional on current parameter iterate (*EM algorithm*!)
  - The probability that a_{kl} is used at position i of sequence $\mathbf{x}$:

$$\begin{aligned} & \mathbf{P}(\pi_i = k, \pi_{i+1} = l|\mathbf{x}, \theta) \\ = \ & \mathbf{P}(\mathbf{x}, \pi_i = k, \pi_{i+1} = l)/\mathbf{P}(\mathbf{x}) \\ = \ & \mathbf{P}(x_1 \dots x_i, \pi_i = k) a_{kl} e_l(x_{i+1}) \mathbf{P}(x_{i+2} \dots x_L | \pi_{i+1} = l)/\mathbf{P}(\mathbf{x}) \\ = \ & f_k(i) a_{kl} e_l(x_{i+1}) b_l(i+1)/\mathbf{P}(\mathbf{x}). \end{aligned}$$

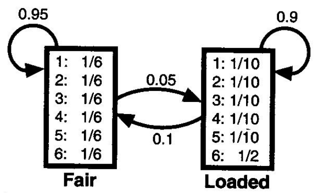
  - So the expected number of times that a_{kl} is used in all training sequences is

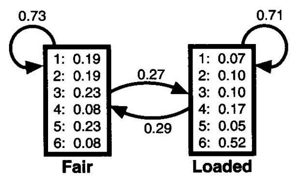
$$A_{kl} = \sum_{j=1}^{n} \frac{1}{\mathbf{P}(\mathbf{x}^j)} \sum_i f_k^j(i) a_{kl} e_l(x_{i+1}^j) b_l^j(i+1). \tag{2}$$

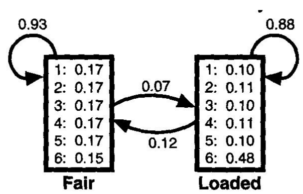
- Baum-Welch Algorithm.
  - Initialization: Pick arbitrary model parameters
  - Recursion
    * Set all the A and E variables to pseudocounts rs (or to zero)
    * For each sequence $j = 1, \dots, n$
      · calculate $f_k(i)$ for sequence j using the forward algorithm
      · calculate $b_k(i)$ for sequence j using the backward algorithm
      · add contribution of sequence j to A (2) and E (**??**)
    * Calculate the new model parameters using (1)
    * Calculate the new log-likelihood of the model

– Termination: Stop if change in log-likelihood is less than a predefined threshold or the maximum number of iteration is exceeded

- Baum-Welch – The Occasionally Dishonest Casino.







- Viterbi Algorithm:

  – Calculate *the most probable state path*

$$\boldsymbol{\pi}^* = \text{argmax}_{\boldsymbol{\pi}} P(\boldsymbol{x}, \boldsymbol{\pi}).$$

  – Define the *Viterbi variable*

$$v_l(i) = P(\text{the most probable path ending in state } k \text{ with observation } x_i).$$

  – Recursion for the Viterbi variables

$$v_l(i+1) = e_l(x_{i+1}) \max_k (v_k(i) a_{kl})$$

  – Algorithm:

    ∗ Initialization ($i = 0$): $v_0(0) = 1$, $v_k(0) = 0$ for all $k > 0$

    ∗ Recursion ($i = 1, \ldots, L$):

$$\begin{aligned} v_l(i) &= e_l(x_i) \max_k (v_k(i-1) a_{kl}) \\ \text{ptr}_i(l) &= \text{argmax}_k (v_k(i-1) a_{kl}) \end{aligned}$$

* Termination:

$$
\begin{aligned}
\mathbf{P}(\mathbf{x}, \boldsymbol{\pi}^*) &= \max_k (v_k(L) a_{k0} \\
\pi_L^* &= \operatorname{argmax}_k (v_k(L) a_{k0})
\end{aligned}
$$

* Traceback ($i = L, \ldots, 1$): $\pi^*_{i=1} = \mathrm{ptr}_i(\pi_i^*)$

Time complexity = (# states)^2 $\times$ length of sequence

– Viterbi decoding - The Occasionally Dishonest Casino.

```
Rolls   315116246446644245311321631164152133625144543631656626566666
Die     FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLLLLLLLLL
Viterbi FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLLLLLL

Rolls   651166453132651245636664631636663162326455236266666625151631
Die     LLLLLLFFFFFFFFFFFFLLLLLLLLLLLLLLLLFFFLLLLLLLLLLLLLLLFFFFFFFFF
Viterbi LLLLLLFFFFFFFFFFFFLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLFFFFFFF

Rolls   222555441666566563564324364131513465146353411126414626253356
Die     FFFFFFFFLLLLLLLLLLLLLFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFLL
Viterbi FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFL

Rolls   366163666466232534413661661163252562462255265252266435353336
Die     LLLLLLLLFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
Viterbi LLLLLLLLLLLLFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF

Rolls   233121625364414432335163243633665562466662632666612355245242
Die     FFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLLLLLLLLLLLLLLLFFFFFFFFFFF
Viterbi FFFFFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLLLLLLLLLLLLLFFFFFFFFFFFF
```

Figure 3.5 *The numbers show 300 rolls of a die as described in the example. Below is shown which die was actually used for that roll (F for fair and L for loaded). Under that the prediction by the Viterbi algorithm is shown.*

Application of dynamic programming: fused-lasso

- Fused lasso (Tibshirani et al., 2005) minimizes

$$
-\ell(\boldsymbol{\beta}) + \lambda_1 \sum_{k=1}^{p-1} |\beta_k - \beta_{k-1}| + \lambda_2 \sum_{k=1}^{p} |\beta_k|
$$

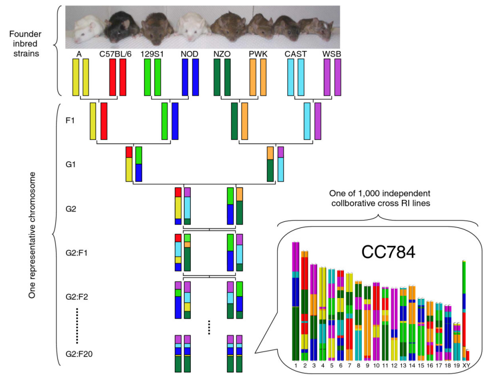
  over $\mathbf{R}^p$ for better recovery of signals that are both sparse and *smooth*

- In many applications, one needs to minimize

$$
O_n(\mathbf{u}) = -\sum_{k=1}^{n} \ell_k(u_k) + \lambda \sum_{k=1}^{n-1} p(u_k, u_{k+1})
$$

  where u_t takes values in a finite space $\mathcal{S}$ and p is a penalty function. A *discrete* (combinatorial) optimization problem.

- A genetic example:



  - Model organism study designs: inbred mice
  - *Goal*: impute the strain origin of inbred mice (Zhou et al., 2012)

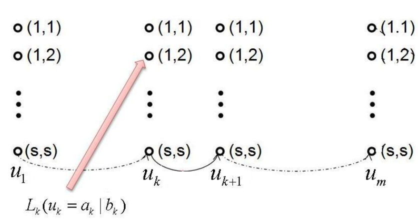
- Combinatorial optimization of penalized likelihood.
  - Minimize objective function

$$O(\mathbf{u}) = -\sum_{k=1}^{n} L_k(u_k) + \sum_{k=1}^{n-1} P_k(u_k, u_{k+1})$$

    by choosing the proper ordered strain origin assignment along the genome
  - $u_k = a_k|b_k$: the ordered strain origin pair
  - L_k: log-likelihood function at marker k - matching imputed genotypes with the observed ones
  - P_k: penalty function for adjacent marker k and $k+1$ - encouraging smoothness of the solution

- Loglikelihood at each marker. At marker k, $u_k = a_k|b_k$: the ordered strain origin pair; r_k/s_k: observed genotype for animal i. Log-penetrance (conditional log-likelihood) is

$$L_k(u_k) = \ln\left[\Pr(r_k/s_k \mid a_k|b_k)\right]$$

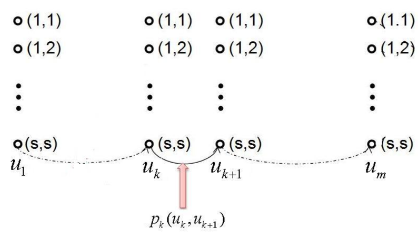


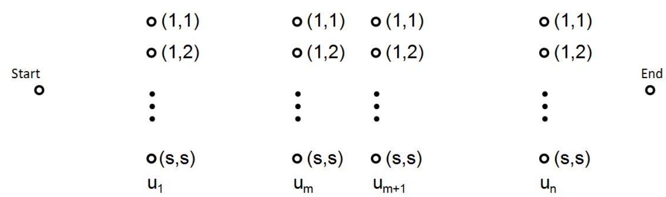
- Penalty for adjacent markers.

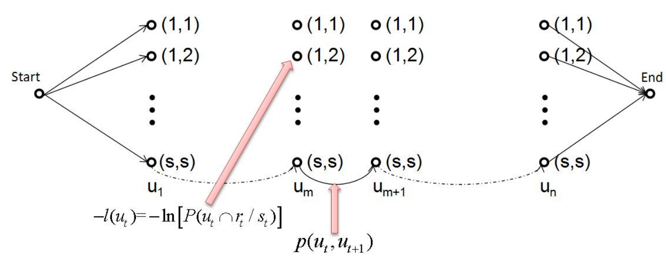
– Penalty $P_k(u_k, u_{k+1})$ for each pair of adjacent markers is

$$P_k(u_k, u_{k+1}) = \begin{cases} 0, & a_k = a_{k+1},\ b_k = b_{k+1} \\ -\ln \gamma_i^p(b_{k+1}) + \lambda, & a_k = a_{k+1},\ b_k \neq b_{k+1} \\ -\ln \gamma_i^m(a_{k+1}) + \lambda, & a_k \neq a_{k+1},\ b_k = b_{k+1} \\ -\ln \psi_{ii}^{mp}(a_{k+1}, b_{k+1}) + 2\lambda, & a_k \neq a_{k+1},\ b_k \neq b_{k+1}. \end{cases}$$

– Penalties suppress jumps between strains and guide jumps, when they occur, toward more likely states.







- For each $m = 1, \ldots, n$,

$$O_m(u_m) = \min_{u_1, \ldots, u_{m-1}} \Big[-\sum_{t=1}^{m} \ell_t(u_t) + \lambda \sum_{t=1}^{m-1} p(u_t, u_{t+1}) \Big]$$

  beginning with $O_1(u_1) = -\ell_1(u_1)$. And to proceed

$$O_{m+1}(u_{m+1}) = \min_{u_m} \Big[O_m(u_m) - \ell_{m+1}(u_{m+1}) + p(u_m, u_{m+1}) \Big]$$

- Computational time is $O(s^4 n)$, where $n = \#$ markers and $s =$ is $\#$ founders.

- More fused-lasso examples.

- Johnson (2013) proposes the dynamic programming algorithm for maximizing the general objective function

$$\sum_{k=1}^{n} e_k(\beta_k) - \lambda \sum_{k=2}^{n} d(\beta_k, \beta_{k-1}),$$

  where e is an exponential family log-likelihood and d is a penalty function, e..g, $d(\beta_k, \beta_{k-1}) = 1_{\{\beta_k \neq \beta_{k-1}\}}$

- Applications: L_0-least squares segmentation, fused lasso signal approximator (FLSA), ...

Take home message from this course

- Statistics, the science of *data analysis*, is the applied mathematics in the 21st century.
- Being good at computing (both *algorithms* and *programming*) is a must for today's working statisticians.
- In this course, we studied and practiced many (overwhelming?) tools that help us deliver results faster and more accurate.
  - Operating systems: Linux and scripting basics
  - Programming languages: R (package development, Rcpp, ...), Matlab, Julia
  - Tools for collaborative and reproducible research: Git, R Markdown, sweave
  - Parallel computing: multi-core, cluster, GPU
  - Convex optimization (LP, QP, SOCP, SDP, GP, cone programming)
  - Integer and mixed integer programming
  - Algorithms for sparse regression
  - More advanced optimization methods motivated by modern statistical and machine learning problems, e.g., ALM, ADMM, online algorithms, ...
  - Dynamic programming
  - Advanced topics on EM/MM algorithms (not really ...)

  Of course there are many tools *not* covered in this course, notably the Bayesian MCMC machinery. Take a Bayesian course!

- Updated benchmark results. R is upgraded to v3.2.0 and Julia to 0.3.7 since beginning of this course. I re-did the benchmark and did not see notable changes.

  Benchmark code `R-benchmark-25.R` from `http://r.research.att.com/benchmarks/R-benchmark-25.R` covers many commonly used numerical operations used in statistics. We ported to Matlab and Julia and report the run times (averaged over 5 runs) here.

  Machine specs: Intel i7 @ 2.6GHz (4 physical cores, 8 threads), 16G RAM, Mac OS 10.9.5.

Test	R 3.2.0	Matlab R2014a	julia 0.3.7
Matrix creation, trans, deformation (2500×2500)	0.80	**0.17**	0.16
Power of matrix (2500×2500, $A.^{1000}$)	0.22	**0.11**	0.22
Quick sort ($n = 7 \times 10^6$)	0.64	**0.24**	0.62
Cross product (2800×2800, $A^T A$)	9.89	**0.35**	0.37
LS solution ($n = p = 2000$)	1.21	**0.07**	0.09
FFT ($n = 2400000$)	0.36	**0.04**	0.14
Eigen-decomposition (600×600)	0.77	**0.31**	0.53
Determinant (2500×2500)	3.52	**0.18**	0.22
Cholesky (3000×3000)	4.08	**0.15**	0.21
Matrix inverse (1600×1600)	2.93	**0.16**	0.19
Fibonacci (vector)	0.29	**0.17**	0.65
Hilbert (matrix)	0.18	**0.07**	0.17
GCD (recursion)	0.28	**0.14**	0.20
Toeplitz matrix (loops)	0.32	**0.0014**	0.03
Escoufiers (mixed)	0.39	0.40	**0.15**

  For the simple Gibbs sampler test, R v3.2.0 takes **38.32**s elapsed time. Julia v0.3.7 takes **0.35**s.

- Do not forget course evaluation: `https://classeval.ncsu.edu/secure/prod/cesurvey/`

References

Alizadeh, F. and Goldfarb, D. (2003). Second-order cone programming. *Math. Program.*, 95(1, Ser. B):3–51. ISMP 2000, Part 3 (Atlanta, GA).

Armagan, A., Dunson, D., and Lee, J. (2013). Generalized double Pareto shrinkage. *Statistica Sinica*, 23:119–143.

Baggerly, K. A. and Coombes, K. R. (2009). Deriving chemosensitivity from cell lines: Forensic bioinformatics and reproducible research in high-throughput biology. *Ann. Appl. Stat.*, 3(4):1309–1334.

Baum, L. E., Petrie, T., Soules, G., and Weiss, N. (1970). A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *Ann. Math. Statist.*, 41:164–171.

Beck, A. and Teboulle, M. (2009a). Fast gradient-based algorithms for constrained total variation image denoising and deblurring problems. *Trans. Img. Proc.*, 18(11):2419–2434.

Beck, A. and Teboulle, M. (2009b). A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. Imaging Sci.*, 2(1):183–202.

Belloni, A., Chernozhukov, V., and Wang, L. (2011). Square-root lasso: pivotal recovery of sparse signals via conic programming. *Biometrika*, 98(4):791–806.

Ben-Tal, A. and Nemirovski, A. (2001). *Lectures on Modern Convex Optimization.* MPS/SIAM Series on Optimization. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA; Mathematical Programming Society (MPS), Philadelphia, PA. Analysis, algorithms, and engineering applications.

Bertsekas, D. P. (1982). *Constrained Optimization and Lagrange Multiplier Methods.* Computer Science and Applied Mathematics. Academic Press Inc. [Harcourt Brace Jovanovich Publishers], New York.

Bertsimas, D. and Weismantel, R. (2005). *Optimization Over Integers.* Athena Scientific.

Boyd, S., Parikh, N., Chu, E., Peleato, B., and Eckstein, J. (2011). Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn.*, 3(1):1–122.

Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization.* Cambridge University Press, Cambridge.

Brown, L. D. (1971). Admissible estimators, recurrent diffusions, and insoluble boundary value problems. *Ann. Math. Statist.*, 42:855–903.

Buckheit, J. and Donoho, D. (1995). Wavelab and reproducible research. In Antoniadis, A. and Oppenheim, G., editors, *Wavelets and Statistics*, volume 103 of *Lecture Notes in Statistics*, pages 55–81. Springer New York.

Burge, C. (1997). Prediction of complete gene structures in human genomic DNA. *Journal of Molecular Biology*, 268(1):78–94.

Cai, J.-F., Candès, E. J., and Shen, Z. (2010). A singular value thresholding algorithm for matrix completion. *SIAM J. Optim.*, 20(4):1956–1982.

Candès, E. and Tao, T. (2007). The Dantzig selector: statistical estimation when p is much larger than n. *Ann. Statist.*, 35(6):2313–2351.

Candès, E. J., Romberg, J. K., and Tao, T. (2006). Stable signal recovery from incomplete and inaccurate measurements. *Communications on Pure and Applied Mathematics*, 59(8):1207–1223.

Candès, E. J. and Tao, T. (2006). Near-optimal signal recovery from random projections: universal encoding strategies? *IEEE Trans. Inform. Theory*, 52(12):5406–5425.

Candès, E. J., Wakin, M. B., and Boyd, S. P. (2008). Enhancing sparsity by reweighted l_1 minimization. *J. Fourier Anal. Appl.*, 14(5-6):877–905.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2009). *Introduction to Algorithms*. MIT Press, Cambridge, MA, third edition.

Daubechies, I., Defrise, M., and De Mol, C. (2004). An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Comm. Pure Appl. Math.*, 57(11):1413–1457.

Donoho, D. and Stodden, V. (2004). When does non-negative matrix factorization give a correct decomposition into parts? In Thrun, S., Saul, L., and Schölkopf, B., editors, *Advances in Neural Information Processing Systems 16*, pages 1141–1148. MIT Press.

Donoho, D. L. (2006). Compressed sensing. *IEEE Trans. Inform. Theory*, 52(4):1289–1306.

Donoho, D. L. (2010). An invitation to reproducible computational research. *Biostatistics*, 11(3):385–388.

Donoho, D. L. and Johnstone, I. M. (1994). Ideal spatial adaptation by wavelet shrinkage. *Biometrika*, 81(3):425–455.

Donoho, D. L. and Johnstone, I. M. (1995). Adapting to unknown smoothness via wavelet shrinkage. *J. Amer. Statist. Assoc.*, 90(432):1200–1224.

Donoho, D. L. and Tanner, J. (2010). Counting the faces of randomly-projected hypercubes and orthants, with applications. *Discrete Comput. Geom.*, 43(3):522–541.

Durbin, R., Eddy, S., Krogh, A., and Mitchison, G. (2006). *Biological Sequence Analysis*. eleventh edition.

Dykstra, R. L. (1983). An algorithm for restricted least squares regression. *J. Amer. Statist. Assoc.*, 78(384):837–842.

Eaton, M. L. (1992). A statistical diptych: admissible inferences—recurrence of symmetric Markov chains. *Ann. Statist.*, 20(3):1147–1179.

Efron, B. (2004). The estimation of prediction error: covariance penalties and cross-validation. *J. Amer. Statist. Assoc.*, 99(467):619–642. With comments and a rejoinder by the author.

Efron, B., Hastie, T., Johnstone, I., and Tibshirani, R. (2004). Least angle regression. *Ann. Statist.*, 32(2):407–499. With discussion, and a rejoinder by the authors.

Efron, B. and Morris, C. (1973). Stein's estimation rule and its competitors—an empirical Bayes approach. *J. Amer. Statist. Assoc.*, 68:117–130.

Efron, B. and Morris, C. (1977). Stein's paradox in statistics. *Scientific American*, 236(5):119–127.

Fan, J. and Li, R. (2001). Variable selection via nonconcave penalized likelihood and its oracle properties. *J. Amer. Statist. Assoc.*, 96(456):1348–1360.

Frank, I. E. and Friedman, J. H. (1993). A statistical view of some chemometrics regression tools. *Technometrics*, 35(2):109–135.

Friedman, J. (2008). Fast sparse regression and classification. *http://www-stat.stanford.edu/ jhf/ftp/GPSpaper.pdf*.

Friedman, J., Hastie, T., Höfling, H., and Tibshirani, R. (2007). Pathwise coordinate optimization. *Ann. Appl. Stat.*, 1(2):302–332.

Friedman, J., Hastie, T., and Tibshirani, R. (2008). Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441.

Friedman, J. H., Hastie, T., and Tibshirani, R. (2010). Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22.

Fu, W. J. (1998). Penalized regressions: the bridge versus the lasso. *J. Comput. Graph. Statist.*, 7(3):397–416.

Goldstein, T. and Osher, S. (2009). The split Bregman method for l_1-regularized problems. *SIAM J. Img. Sci.*, 2:323–343.

Golub, G. H. and Van Loan, C. F. (1996). *Matrix Computations*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, Baltimore, MD, third edition.

Grant, M. and Boyd, S. (2008). Graph implementations for nonsmooth convex programs. In Blondel, V., Boyd, S., and Kimura, H., editors, *Recent Advances in Learning and Control*, Lecture Notes in Control and Information Sciences, pages 95–110. Springer-Verlag Limited. `http://stanford.edu/~boyd/graph_dcp.html`.

Hastie, T., Rosset, S., Tibshirani, R., and Zhu, J. (2004). The entire regularization path for the support vector machine. *J. Mach. Learn. Res.*, 5:1391–1415.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Series in Statistics. Springer, New York, second edition.

Hestenes, M. R. (1969). Multiplier and gradient methods. *J. Optimization Theory Appl.*, 4:303–320.

Huber, P. J. (1994). Huge data sets. In *COMPSTAT 1994 (Vienna)*, pages 3–13. Physica, Heidelberg.

Huber, P. J. (1996). Massive data sets workshop: The morning after. In *Massive Data Sets: Proceedings of a Workshop*, pages 169–184. National Academy Press, Washington.

James, W. and Stein, C. (1961). Estimation with quadratic loss. In *Proc. 4th Berkeley Sympos. Math. Statist. and Prob., Vol. I*, pages 361–379. Univ. California Press, Berkeley, Calif.

Johnson, N. A. (2013). A dynamic programming algorithm for the fused lasso and L_0-segmentation. *Journal of Computational and Graphical Statistics*, to appear.

Jones, N. C. and Pevzner, P. A. (2004). *An Introduction to Bioinformatics Algorithms (Computational Molecular Biology)*. The MIT Press.

Laurent, M. and Rendl, F. (2005). Semidefinite programming and integer programming. In K. Aardal, G. N. and Weismantel, R., editors, *Discrete Optimization*, volume 12 of *Handbooks in Operations Research and Management Science*, pages 393 – 514. Elsevier.

Lobo, M. S., Vandenberghe, L., Boyd, S., and Lebret, H. (1998). Applications of second-order cone programming. *Linear Algebra Appl.*, 284(1-3):193–228. ILAS Symposium on Fast Algorithms for Control, Signals and Image Processing (Winnipeg, MB, 1997).

Mazumder, R., Friedman, J. H., and Hastie, T. (2011). SparseNet: Coordinate descent with nonconvex penalties. *Journal of the American Statistical Association*, 106(495):1125–1138.

McKay, B., Bar-Natan, D., Bar-Hillel, M., and Kalai, G. (1999). Solving the bible code puzzle. *Statist. Sci.*, 14(2):150–173.

Nemhauser, G. and Wolsey, L. (1999). *Integer and Combinatorial Optimization*. Wiley-Interscience Series in Discrete Mathematics and Optimization. John Wiley & Sons, Inc., New York. Reprint of the 1988 original, A Wiley-Interscience Publication.

Nesterov, Y. (1983). A method of solving a convex programming problem with convergence rate $O(1/k^2)$. *Soviet Mathematics Doklady*, 27(2):372–376.

Nesterov, Y. (1988). On an approach to the construction of optimal methods of minimization of smooth convex functions. *Ekonomika i Mateaticheskie Metody*, 24:509–517.

Nesterov, Y. (2000). Squared functional systems and optimization problems. In *High performance optimization*, volume 33 of *Appl. Optim.*, pages 405–440. Kluwer Acad. Publ., Dordrecht.

Nesterov, Y. (2005). Smooth minimization of non-smooth functions. *Math. Program.*, 103(1, Ser. A):127–152.

Nesterov, Y. (2007). Gradient methods for minimizing composite objective function.

Osborne, M. R., Presnell, B., and Turlach, B. A. (2000). A new approach to variable selection in least squares problems. *IMA J. Numer. Anal.*, 20(3):389–403.

Parikh, N. and Boyd, S. (2013). Proximal algorithms. *Found. Trends Mach. Learn.*, 1(3):123–231.

Park, M. Y. and Hastie, T. (2007). L_1-regularization path algorithm for generalized linear models. *J. R. Stat. Soc. Ser. B Stat. Methodol.*, 69(4):659–677.

Peng, R. D. (2009). Reproducible research and biostatistics. *Biostatistics*, 10(3):405–408.

Peng, R. D. (2011). Reproducible research in computational science. *Science*, 334(6060):1226–1227.

Platt, J. C. (1999). Fast training of support vector machines using sequential minimal optimization. In Schölkopf, B., Burges, C. J. C., and Smola, A. J., editors, *Advances in Kernel Methods*, pages 185–208. MIT Press, Cambridge, MA, USA.

Potti, A., Dressman, H. K., Bild, A., and Riedel, R. F. (2006). Genomic signatures to guide the use of chemotherapeutics. *Nature medicine*, 12(11):1294–1300.

Powell, M. J. D. (1969). A method for nonlinear constraints in minimization problems. In *Optimization (Sympos., Univ. Keele, Keele, 1968)*, pages 283–298. Academic Press, London.

Rabiner, L. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

Rosset, S. and Zhu, J. (2007). Piecewise linear regularized solution paths. *Ann. Statist.*, 35(3):1012–1030.

Stein, C. (1956). Inadmissibility of the usual estimator for the mean of a multivariate normal distribution. In *Proceedings of the Third Berkeley Symposium on Mathematical Statistics and Probability, 1954–1955, vol. I*, pages 197–206, Berkeley and Los Angeles. University of California Press.

Stein, C. M. (1981). Estimation of the mean of a multivariate normal distribution. *Ann. Statist.*, 9(6):1135–1151.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *J. Roy. Statist. Soc. Ser. B*, 58(1):267–288.

Tibshirani, R., Bien, J., Friedman, J., Hastie, T., Simon, N., Taylor, J., and Tibshirani, R. J. (2012). Strong rules for discarding predictors in lasso-type problems. *J. R. Stat. Soc. Ser. B. Stat. Methodol.*, 74(2):245–266.

Tibshirani, R., Saunders, M., Rosset, S., Zhu, J., and Knight, K. (2005). Sparsity and smoothness via the fused lasso. *J. R. Stat. Soc. Ser. B Stat. Methodol.*, 67(1):91–108.

Tibshirani, R. J. and Taylor, J. (2011). The solution path of the generalized lasso. *Ann. Statist.*, 39(3):1335–1371.

Tseng, P. (2001). Convergence of a block coordinate descent method for nondifferentiable minimization. *J. Optim. Theory Appl.*, 109(3):475–494.

Tseng, P. (2008). On accelerated proximal gradient methods for convex-concave optimization. *submitted to SIAM Journal on Optimization*.

Vielma, J. P., Ahmed, S., and Nemhauser, G. (2010). Mixed-integer models for nonseparable piecewise-linear optimization: unifying framework and extensions. *Oper. Res.*, 58(2):303–315.

Viterbi, A. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *Information Theory, IEEE Transactions on*, 13(2):260–269.

Wainwright, M. J. and Jordan, M. I. (2008). Graphical models, exponential families, and variational inference. *Found. Trends Mach. Learn.*, 1(1-2):1–305.

Williams, H. P. (2013). *Model Building in Mathematical Programming*. John Wiley & Sons, Ltd., Chichester, fifth edition.

Witztum, D., Rips, E., and Rosenberg, Y. (1994). Equidistant letter sequences in the book of genesis. *Statist. Sci.*, 9(3):429–438.

Wu, T. T., Chen, Y., Hastie, T., Sobel, E., and Lange, K. (2009). Genome-wide association analysis by lasso penalized logistic regression. *Bioinformatics*, 25(6):714–721.

Wu, T. T. and Lange, K. (2008). Coordinate descent algorithms for lasso penalized regression. *Ann. Appl. Stat.*, 2(1):224–244.

Wu, Y. (2011). An ordinary differential equation-based solution path algorithm. *Journal of Nonparametric Statistics*, 23:185–199.

Yin, W., Osher, S., Goldfarb, D., and Darbon, J. (2008). Bregman iterative algorithms for l_1-minimization with applications to compressed sensing. *SIAM J. Imaging Sci.*, 1(1):143–168.

Yuan, M. and Lin, Y. (2006). Model selection and estimation in regression with grouped variables. *J. R. Stat. Soc. Ser. B Stat. Methodol.*, 68(1):49–67.

Zhang, C.-H. (2010). Nearly unbiased variable selection under minimax concave penalty. *Ann. Statist.*, 38(2):894–942.

Zhou, H. and Lange, K. (2013). A path algorithm for constrained estimation. *Journal of Computational and Graphical Statistics*, 22:261–283.

Zhou, H. and Li, L. (2014). Regularized matrix regressions. *Journal of Royal Statistical Society, Series B*, 76(2):463–483.

Zhou, H. and Wu, Y. (2014). A generic path algorithm for regularized statistical estimation. *J. Amer. Statist. Assoc.*, 109(506):686–699.

Zhou, J. J., Ghazalpour, A., Sobel, E. M., Sinsheimer, J. S., and Lange, K. (2012). Quantitative trait loci association mapping by imputation of strain origins in multifounder crosses. *Genetics*, 190(2):459–473.

Zhu, J., Rosset, S., Tibshirani, R., and Hastie, T. J. (2004). 1-norm support vector machines. In Thrun, S., Saul, L., and Schölkopf, B., editors, *Advances in Neural Information Processing Systems 16*, pages 49–56. MIT Press.

Zou, H. and Hastie, T. (2005). Regularization and variable selection via the elastic net. *J. R. Stat. Soc. Ser. B Stat. Methodol.*, 67(2):301–320.

Zou, H., Hastie, T., and Tibshirani, R. (2007). On the "degrees of freedom" of the lasso. *Ann. Statist.*, 35(5):2173–2192.