ST758: Computation for Statistical Research

Tue/Thu 10:15am-11:30am, SAS Hall 1108

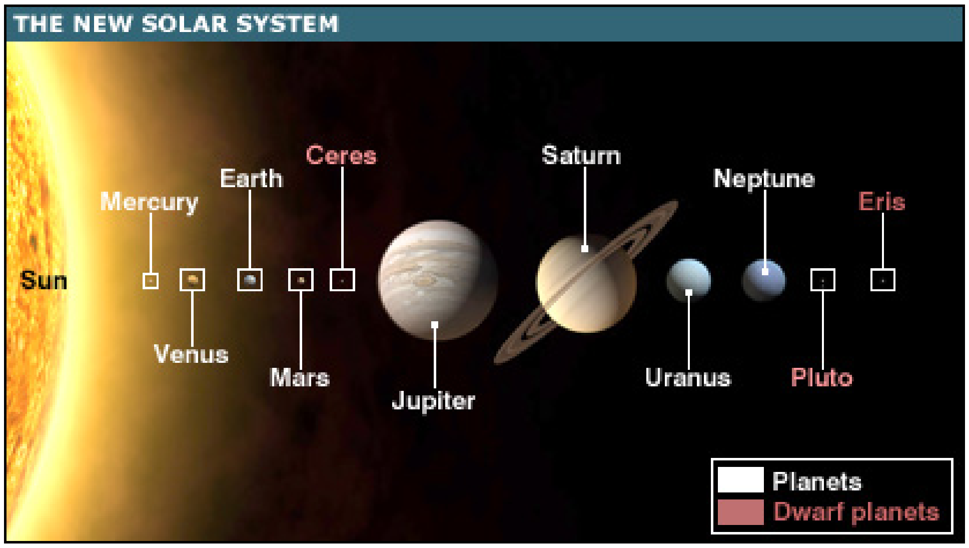
Instructor: Dr Hua Zhou, `hua_zhou@ncsu.edu`

1 Lecture 1: Aug 21

Today

- Introduction and course logistics
- Computer storage and arithmetic
- If you never used R before, go through Appendix A "A Sample Session" of the R manual on your computer

How Gauss became famous?

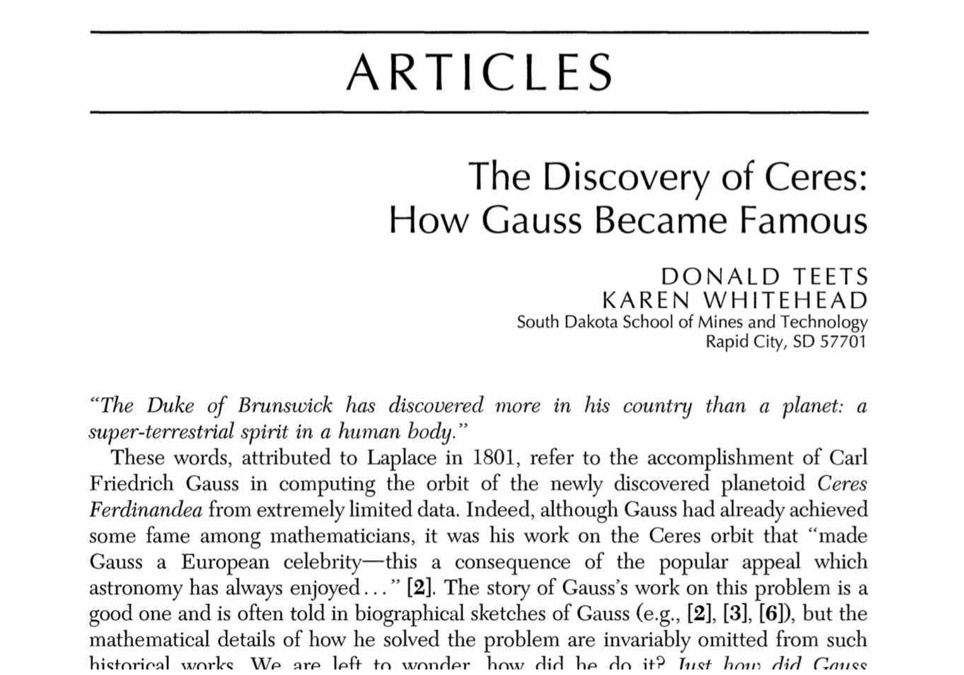


- 1801, *Dr Carl Friedrich Gauss*, 24; proved Fundamental Theorem of Algebra; wrote the book *Disquisitiones Arithmetic*, which is still being studied today
- 1801, Jan 1 - Feb 11 (41 days), astronomer Piazzi observed Ceres (a dwarf planet), which was then lost behind sun
- 1801, Aug – Sep, futile search by top astronomers; Laplace claimed it unsolvable
- 1801, Oct – Nov, Gauss did calculations by *method of least squares*
- 1801, Dec 31, astronomer von Zach relocated Ceres according to Gauss' calculation
- 1802, *Summarische Übersicht der Bestimmung der Bahnen der beiden neuen Hauptplaneten angewandten Methoden*, considered the origin of linear algebra
- 1807, Professor of Astronomy and Director of Göttingen Observatory in remainder of his life
- 1809, *Theoria motus corporum coelestium in sectionibus conicis solem ambientum* (Theory of motion of the celestial bodies moving in conic sections around the Sun); birth of the Gaussian (normal) distribution, as an attempt to rationalize the method of least squares
- 1810, Laplace consolidated importance of Gaussian distribution by proving the central limit theorem

- 1829, Gauss-Markov Theorem. Under Gaussian error assumption (actually only uncorrelated and homoscedastic needed), least square solution is the best linear unbiased estimate (BLUE), i.e., it has the smallest variance and thus MSE among all linear unbiased estimators. Note other estimators such as the James-Stein estimator may have smaller MSE, but they are *nonlinear*.

For more details

- `http://www.keplersdiscovery.com/Asteroid.html`
- Teets and Whitehead (1999)



ARTICLES

The Discovery of Ceres: How Gauss Became Famous

DONALD TEETS
KAREN WHITEHEAD
South Dakota School of Mines and Technology
Rapid City, SD 57701

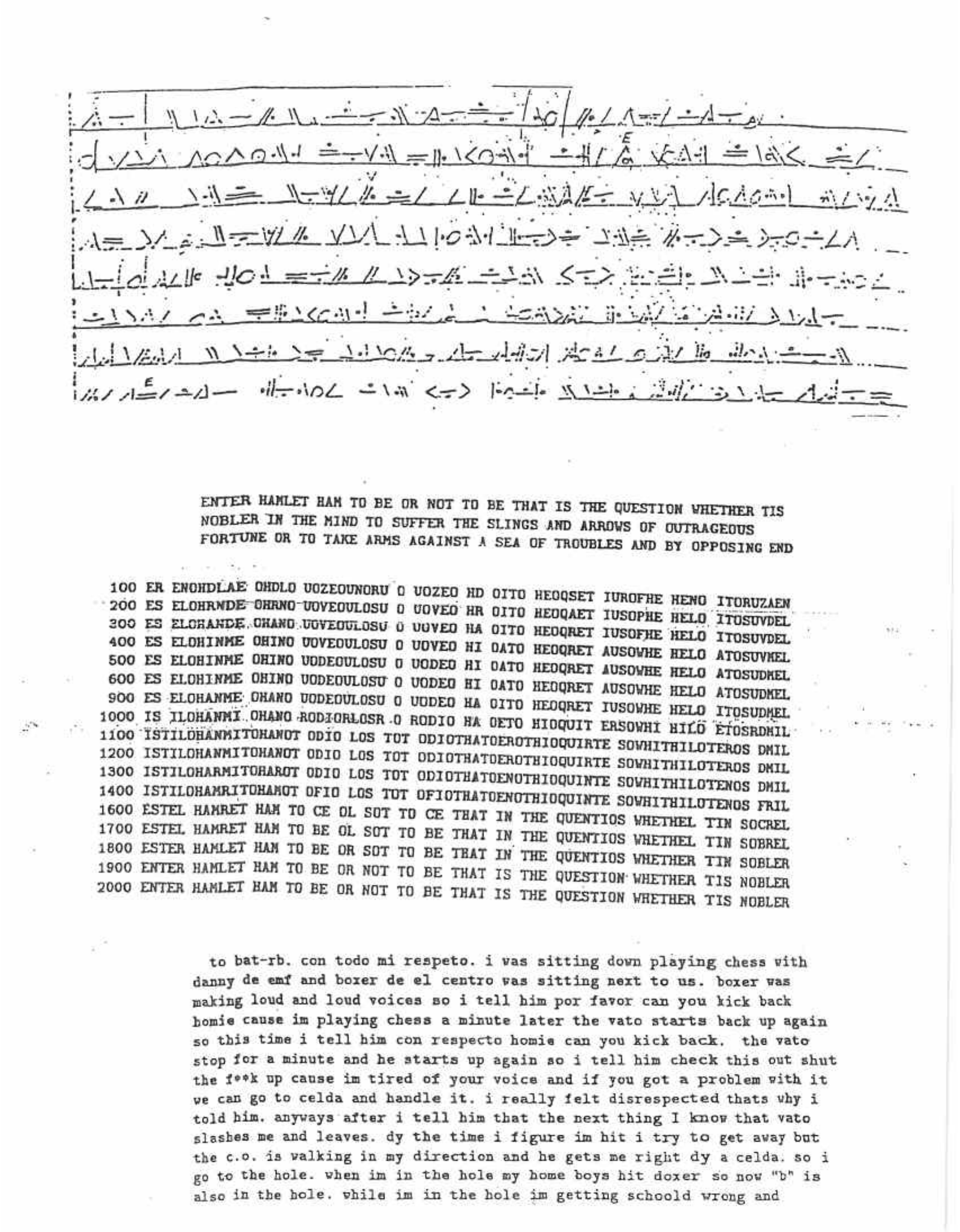
"The Duke of Brunswick has discovered more in his country than a planet: a super-terrestrial spirit in a human body."

These words, attributed to Laplace in 1801, refer to the accomplishment of Carl Friedrich Gauss in computing the orbit of the newly discovered planetoid *Ceres Ferdinandea* from extremely limited data. Indeed, although Gauss had already achieved some fame among mathematicians, it was his work on the Ceres orbit that "made Gauss a European celebrity—this a consequence of the popular appeal which astronomy has always enjoyed..." **[2]**. The story of Gauss's work on this problem is a good one and is often told in biographical sketches of Gauss (e.g., **[2]**, **[3]**, **[6]**), but the mathematical details of how he solved the problem are invariably omitted from such historical works. We are left to wonder, how did he do it? *Just how did Gauss*

Gauss' story

- Motivated by a real problem.
- Heuristic solution: method of least squares.
- Solution readily verifiable: Ceres was re-discovered!
- *Algorithmic development*: linear algebra, Gaussian elimination, FFT (fast Fourier transform).
- Theoretical justification: Gaussian distribution, Gauss-Markov theorem.

A sampler by Marc Coram



```
ENTER HAMLET HAM TO BE OR NOT TO BE THAT IS THE QUESTION WHETHER TIS
NOBLER IN THE MIND TO SUFFER THE SLINGS AND ARROWS OF OUTRAGEOUS
FORTUNE OR TO TAKE ARMS AGAINST A SEA OF TROUBLES AND BY OPPOSING END

 100 ER ENOHDLAE OHDLO UOZEOUNORU O UOZEO HD OITO HEOQSET IUROFHE HENO ITORUZAEN
 200 ES ELOHRNDE OHRNO UOVEOULOSU O UOVEO HR OITO HEOQAET IUSOPHE HELO ITOSUVDEL
 300 ES ELOHANDE OHANO UOVEOULOSU O UOVEO HA OITO HEOQRET IUSOFHE HELO ITOSUVDEL
 400 ES ELOHINME OHINO UOVEOULOSU O UOVEO HI OATO HEOQRET AUSOWHE HELO ATOSUVMEL
 500 ES ELOHINME OHINO UODEOULOSU O UODEO HI OATO HEOQRET AUSOWHE HELO ATOSUDMEL
 600 ES ELOHINME OHINO UODEOULOSU O UODEO HI OATO HEOQRET AUSOWHE HELO ATOSUDMEL
 900 ES ELOHANME OHANO UODEOULOSU O UODEO HA OITO HEOQRET IUSOWHE HELO ITOSUDMEL
1000 IS ILOHANMI OHANO RODIORLOSR O RODIO HA OETO HIOQUIT ERSOWHI HILO ETOSRDMIL
1100 ISTILOHANMITOHANOT ODIO LOS TOT ODIOTHATOEROTHIOQUIRTE SOWHITHILOTEROS DMIL
1200 ISTILOHANMITOHANOT ODIO LOS TOT ODIOTHATOEROTHIOQUIRTE SOWHITHILOTEROS DMIL
1300 ISTILOHARMITOHAROT ODIO LOS TOT ODIOTHATOENOTHIOQUINTE SOWHITHILOTENOS DMIL
1400 ISTILOHAMRITOHANOT OFIO LOS TOT OFIOTHATOENOTHIOQUINTE SOWHITHILOTENOS FRIL
1600 ESTEL HAMRET HAM TO CE OL SOT TO CE THAT IN THE QUENTIOS WHETHEL TIN SOCREL
1700 ESTEL HAMRET HAM TO BE OL SOT TO BE THAT IN THE QUENTIOS WHETHEL TIN SOBREL
1800 ESTER HAMLET HAM TO BE OR SOT TO BE THAT IN THE QUENTIOS WHETHER TIN SOBLER
1900 ENTER HAMLET HAM TO BE OR NOT TO BE THAT IS THE QUESTION WHETHER TIN SOBLER
2000 ENTER HAMLET HAM TO BE OR NOT TO BE THAT IS THE QUESTION WHETHER TIS NOBLER
```

to bat-rb. con todo mi respeto. i was sitting down playing chess with danny de emf and boxer de el centro was sitting next to us. boxer was making loud and loud voices so i tell him por favor can you kick back homie cause im playing chess a minute later the vato starts back up again so this time i tell him con respecto homie can you kick back. the vato stop for a minute and he starts up again so i tell him check this out shut the f**k up cause im tired of your voice and if you got a problem with it we can go to celda and handle it. i really felt disrespected thats why i told him. anyways after i tell him that the next thing I know that vato slashes me and leaves. dy the time i figure im hit i try to get away but the c.o. is walking in my direction and he gets me right dy a celda. so i go to the hole. when im in the hole my home boys hit doxer so now "b" is also in the hole. while im in the hole im getting schoold wrong and

- Consulting project by Marc Coram (then a graduate student in statistics at Stanford); customer is a professor in political science.
- Marc modeled letter sequence by a Markov chain (26×26 transition matrix) and estimated transition probabilities from *War and Peace*.
- Now each mapping σ yields a likelihood $f(\sigma)$ of the symbol sequence.

- Find the σ that maximizes f. Sample space is at least $26! = 4.0329 \times 10^{26}$. Combinatorial optimization – hard!

- *Metropolis sampling*: At each iteration, generate a new σ' by random transposition of two letters; accept σ' with probability $\min\left\{\frac{f(\sigma')}{f(\sigma)}, 1\right\}$

Marc Coram's story

- Motivated by a real problem.

- Solution readily verifiable: we can read it!

- *Algorithm*: Metropolis is one of top 10 algorithms in the 20th century.

- Read Diaconis (2009) for more details.

What is this course about?

- Not a course on "packages and languages for data analysis". It does not answer questions such as "How to fit a linear mixed model in SAS or R?"

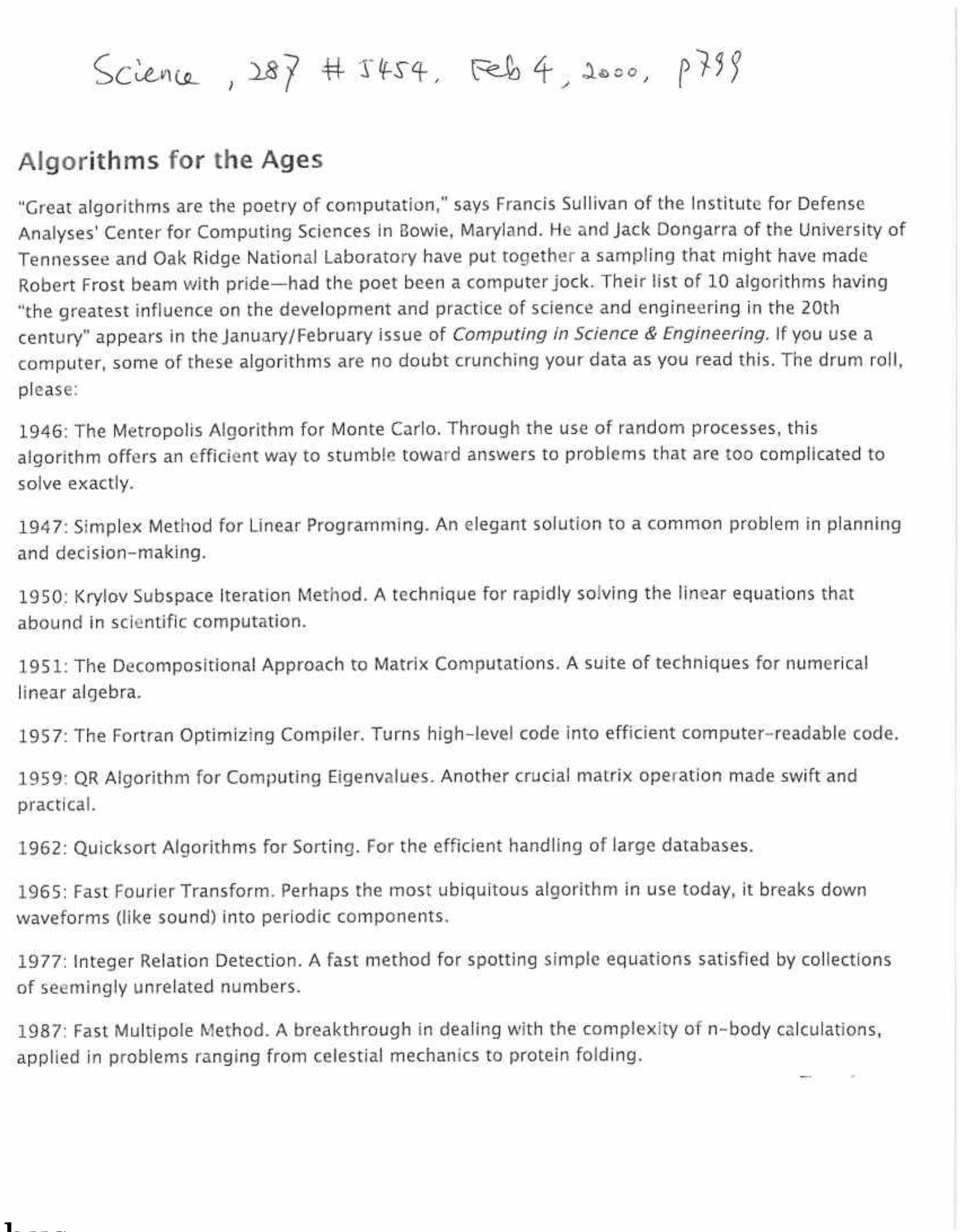
- Not a programming course, although it is extremely important and we do homework and projects in R.

- This course is about "numerical methods in statistics". Our focus is on *algorithms.*

  > The form of a mathematical expression and the way the expression should be evaluated in actual practice may be quite different.

  For a common numerical task in statistics, say solving the normal equation $\boldsymbol{X}^\top \boldsymbol{X}\boldsymbol{\beta} = \boldsymbol{X}^\top \boldsymbol{y}$, we need to know which methods/algorithms are there and what are their advantages and disadvantages. You will automatically fail this course if you use

  ```
  solve(t(X) %*% X) %*% t(X) %*% y
  ```



Science, 287 # 5454, Feb 4, 2000, p799

Algorithms for the Ages

"Great algorithms are the poetry of computation," says Francis Sullivan of the Institute for Defense Analyses' Center for Computing Sciences in Bowie, Maryland. He and Jack Dongarra of the University of Tennessee and Oak Ridge National Laboratory have put together a sampling that might have made Robert Frost beam with pride—had the poet been a computer jock. Their list of 10 algorithms having "the greatest influence on the development and practice of science and engineering in the 20th century" appears in the January/February issue of *Computing in Science & Engineering.* If you use a computer, some of these algorithms are no doubt crunching your data as you read this. The drum roll, please:

1946: The Metropolis Algorithm for Monte Carlo. Through the use of random processes, this algorithm offers an efficient way to stumble toward answers to problems that are too complicated to solve exactly.

1947: Simplex Method for Linear Programming. An elegant solution to a common problem in planning and decision-making.

1950: Krylov Subspace Iteration Method. A technique for rapidly solving the linear equations that abound in scientific computation.

1951: The Decompositional Approach to Matrix Computations. A suite of techniques for numerical linear algebra.

1957: The Fortran Optimizing Compiler. Turns high-level code into efficient computer-readable code.

1959: QR Algorithm for Computing Eigenvalues. Another crucial matrix operation made swift and practical.

1962: Quicksort Algorithms for Sorting. For the efficient handling of large databases.

1965: Fast Fourier Transform. Perhaps the most ubiquitous algorithm in use today, it breaks down waveforms (like sound) into periodic components.

1977: Integer Relation Detection. A fast method for spotting simple equations satisfied by collections of seemingly unrelated numbers.

1987: Fast Multipole Method. A breakthrough in dealing with the complexity of n-body calculations, applied in problems ranging from celestial mechanics to protein folding.

Syllabus

Check course website frequently for updates and announcements.
`http://hua-zhou.github.io/teaching/st758-2014fall/`
Lecture notes will be updated and posted after each lecture.

2 Lecture 2, Sep 2

Announcements

- TA office hours, Wed @ 10A-12P?
- HW1 posted; due Sep 11 in class.

Last time

- Introduction. Gauss (least squares to find Ceres) – optimization, Marc Coram (decipher a note circulating in jail) – sampling. Two major modes of statistical computing.
- Course content, logistics.

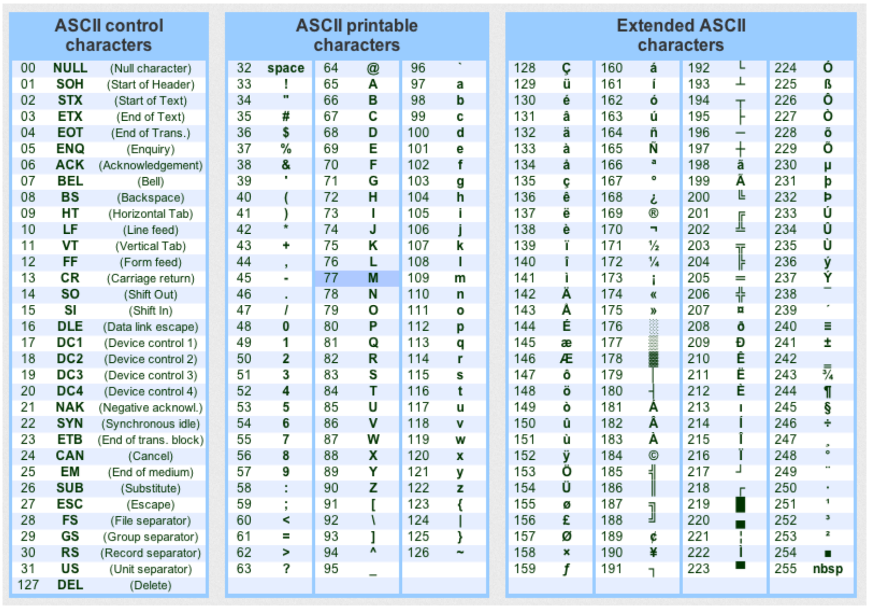
Today

- Computer representation of characters, integers, and real numbers.

Computer storage and arithmetic

Elementary units of computer storage: *bit* = "binary" + "digit" (coined by John Tukey), *byte* = 8 bits, kB = kilobyte = 10^3 bytes, MB = megabytes = 10^6 bytes, GB = gigabytes = 10^9 bytes, TB = terabytes = 10^{12} bytes, PB = petabytes = 10^{15} bytes, ...

Storage of characters



ASCII control characters		
00	NULL	(Null character)
01	SOH	(Start of Header)
02	STX	(Start of Text)
03	ETX	(End of Text)
04	EOT	(End of Trans.)
05	ENQ	(Enquiry)
06	ACK	(Acknowledgement)
07	BEL	(Bell)
08	BS	(Backspace)
09	HT	(Horizontal Tab)
10	LF	(Line feed)
11	VT	(Vertical Tab)
12	FF	(Form feed)
13	CR	(Carriage return)
14	SO	(Shift Out)
15	SI	(Shift In)
16	DLE	(Data link escape)
17	DC1	(Device control 1)
18	DC2	(Device control 2)
19	DC3	(Device control 3)
20	DC4	(Device control 4)
21	NAK	(Negative acknowl.)
22	SYN	(Synchronous idle)
23	ETB	(End of trans. block)
24	CAN	(Cancel)
25	EM	(End of medium)
26	SUB	(Substitute)
27	ESC	(Escape)
28	FS	(File separator)
29	GS	(Group separator)
30	RS	(Record separator)
31	US	(Unit separator)
127	DEL	(Delete)

ASCII printable characters					
32	space	64	@	96	`
33	!	65	A	97	a
34	"	66	B	98	b
35	#	67	C	99	c
36	$	68	D	100	d
37	%	69	E	101	e
38	&	70	F	102	f
39	'	71	G	103	g
40	(	72	H	104	h
41	)	73	I	105	i
42	*	74	J	106	j
43	+	75	K	107	k
44	,	76	L	108	l
45	-	77	M	109	m
46	.	78	N	110	n
47	/	79	O	111	o
48	0	80	P	112	p
49	1	81	Q	113	q
50	2	82	R	114	r
51	3	83	S	115	s
52	4	84	T	116	t
53	5	85	U	117	u
54	6	86	V	118	v
55	7	87	W	119	w
56	8	88	X	120	x
57	9	89	Y	121	y
58	:	90	Z	122	z
59	;	91	[	123	{
60	<	92	\	124	|
61	=	93	]	125	}
62	>	94	^	126	~
63	?	95	_		

Extended ASCII characters							
128	Ç	160	á	192	└	224	Ó
129	ü	161	í	193	┴	225	ß
130	é	162	ó	194	┬	226	Ô
131	â	163	ú	195	├	227	Ò
132	ä	164	ñ	196	─	228	õ
133	à	165	Ñ	197	┼	229	Õ
134	å	166	ª	198	ã	230	µ
135	ç	167	º	199	Ã	231	þ
136	ê	168	¿	200	╚	232	Þ
137	ë	169	®	201	╔	233	Ú
138	è	170	¬	202	╩	234	Û
139	ï	171	½	203	╦	235	Ù
140	î	172	¼	204	╠	236	ý
141	ì	173	¡	205	═	237	Ý
142	Ä	174	«	206	╬	238	¯
143	Å	175	»	207	¤	239	´
144	É	176	░	208	ð	240	≡
145	æ	177	▒	209	Ð	241	±
146	Æ	178	▓	210	Ê	242	‗
147	ô	179	│	211	Ë	243	¾
148	ö	180	┤	212	È	244	¶
149	ò	181	Á	213	ı	245	§
150	û	182	Â	214	Í	246	÷
151	ù	183	À	215	Î	247	¸
152	ÿ	184	©	216	Ï	248	°
153	Ö	185	╣	217	┘	249	¨
154	Ü	186	║	218	┌	250	·
155	ø	187	╗	219	█	251	¹
156	£	188	╝	220	▄	252	³
157	Ø	189	¢	221	¦	253	²
158	×	190	¥	222	Ì	254	■
159	ƒ	191	┐	223	▀	255	nbsp

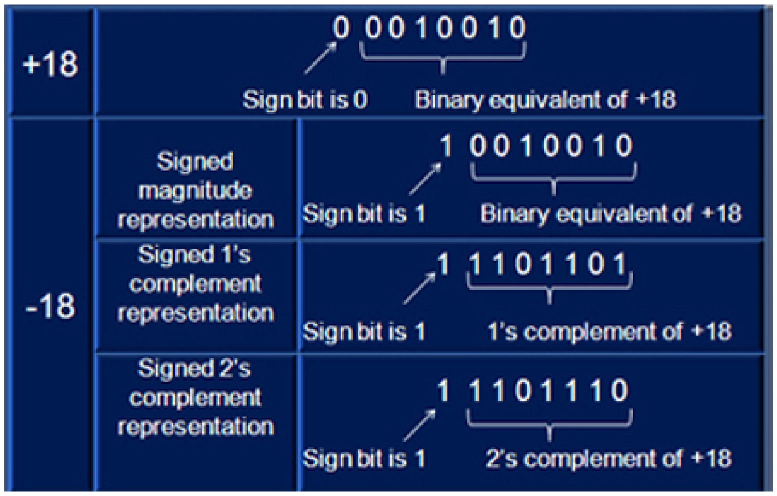
- Plain text files are stored in the form of characters: `.r`, `.c`, `.cpp`, `.tex`, `.html`, ...
- ASCII (American Code for Information Interchange): 7 bits, only $2^7 = 128$ characters, "Hua" corresponds to "48 75 61 (Hex) = 72 117 97 (Dec) = 1001000 1110101 1100001".
- Extended ASCII: 8 bits, $2^8 = 256$ characters.
- Unicode: UTF-8, UTF-16 and UTF-32 support many more characters including foreign characters; last 7 digits conform to ASCII. UTF-8 is the current dominant character encoding on internet.

Fixed-point number system

Fixed-point number system $\mathbb{I}$ is a computer model for integers $\mathbb{Z}$. One storage unit may be $M = 8/16/32/64$ bit.

- The number of bits and method of representing negative numbers vary from system to system. MATLAB has (u)int8, (u)int16, (u)int32, (u)int64. The `integer` type in R has $M = 32$ bits.

- First bit indicates sign: 0 for nonnegative numbers, 1 for negative numbers.
- "two's complement representation" for negative numbers. (i) Sign bit is set to 1, (ii) remaining bits are set to opposite values, (iii) 1 is added to the result.



+18		0 0010010 — Sign bit is 0; Binary equivalent of +18
-18	Signed magnitude representation	1 0010010 — Sign bit is 1; Binary equivalent of +18
	Signed 1's complement representation	1 1101101 — Sign bit is 1; 1's complement of +18
	Signed 2's complement representation	1 1101110 — Sign bit is 1; 2's complement of +18

- Range of representable integers by M-bit storage unit is $[-2^{M-1}, 2^{M-1} - 1]$ (don't need to represent 0 anymore so could have capacity for 2^{M-1} negative numbers).
- For $M = 8$, $[-128, 127]$. For $M = 16$, $[-65536, 65535]$.
  For $M = 32$, $[-2147483648, 2147483647]$.
- Following code snippet shows that the smallest integer in R is $-2^{31} + 1 = -2147483647$.

```
# .Machine variable
.Machine$integer.max
```

```
[1] 2147483647
```

```

# integer type in R uses M=32 bits
M <- 32
big <- 2^(M-1) - 1
small <- - 2^(M-1)
as.integer(big)
```

```
[1] 2147483647
```

```
as.integer(big+1)
```

```
Warning: NAs introduced by coercion
```

```
[1] NA
```

```
as.integer(small+1)
```

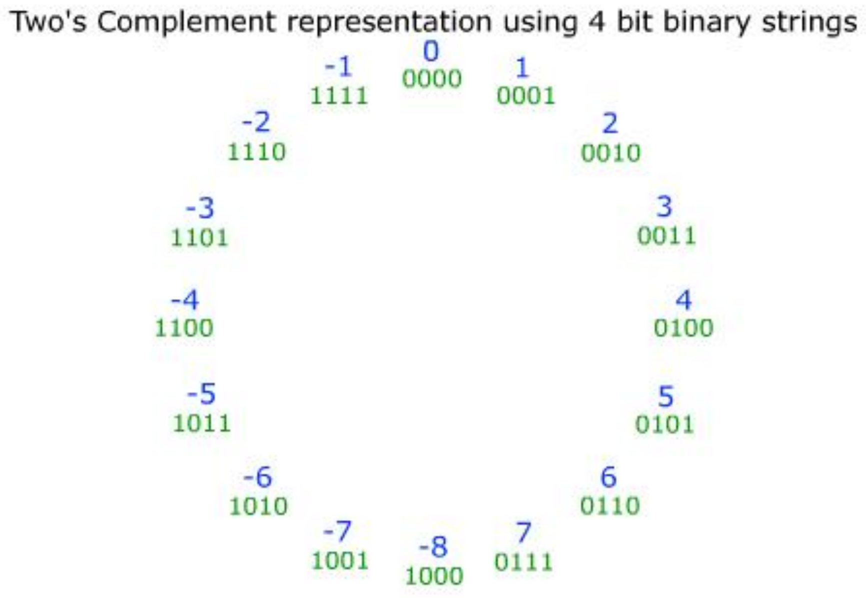
```
[1] -2147483647
```

```
as.integer(small)
```

```
Warning: NAs introduced by coercion
```

```
[1] NA
```

- For unsigned integers such as in MATLAB, the range is $[0, 2^M - 1]$.



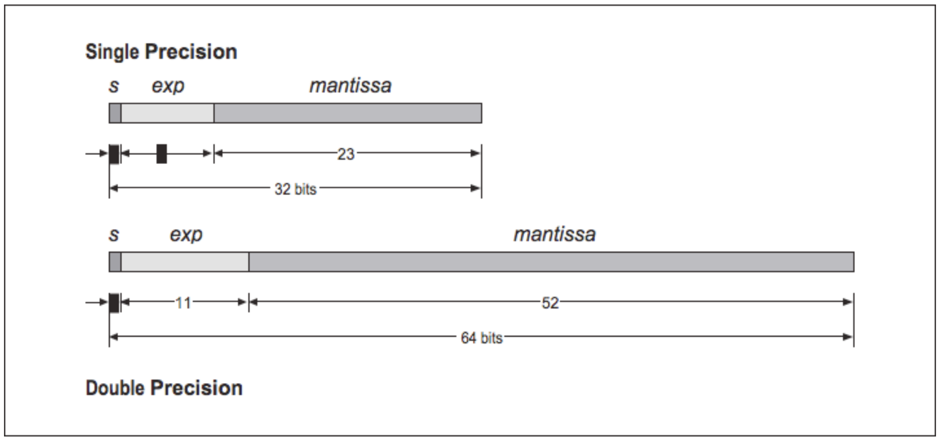
- An integer $-i$ in the interval $[-2^{M-1}, -1]$ would be represented by the same bit pattern by which the nonnegative integer $2^M - i$ is represented, treating the sign bit as a regular numeric bit. For example, (-18) = 11101110, $(2^8) - (18)$=(238)=11101110. Two's complement is subtracting the nonnegative integer i from $1\ldots1 + 1 = 10\ldots0 = (2^M)$.

- Addition and subtraction are much simpler in two's complement representation. Why? it respects modular arithmetic nicely (look at the diagram). E.g., (3)+(4) = 0011 + 0100 = 0111 = (7). Need to keep track of crossing two boundaries 10...0 and 00...0. Cross boundary 10...0 once, we add 2^M, e.g., (6) + (7) = $0110 + 0111 = 1101 = (-3) + (2^4) = (13)$. Cross boundary 00...0, we subtract

2^M, e.g., $(-6)+(-7) = 1010+1001 = 0011 = (3)-(2^4) = (-13)$. Crossing both boundaries is fine, e.g., (-3)+(-4) = 1101 + 1100 = 1001 = (-7).

- Keep track of overflow and underflow. If the result of a summation is R, which must be in the set $[-2^{M-1}, 2^{M-1}+1]$, there are only three possibilities for the true sum: R, $R+2^M$ (overflow), or $R-2^M$ (underflow).
  - When adding two nonnegative integers: 0XX...X + 0YY...Y, where X and Y are arbitrary binary digits, we only need to keep track of overflow in this case. If the resulting binary number has a leading bit of 1, we know overflow occurs since the sum cannot be negative. We should treat that sign bit as a regular numeric bit. In other words, we crossed the upper boundary 100 and should add 2^M to the result. If the resulting binary number has a leading bit of 0, no overflow occurs.
  - When adding two negative integers: 1XX...X + 1YY...Y, where X and Y are arbitrary binary digits, we only need to keep track of underflow in this case. If the resulting binary number has a leading bit of 0, we know underflow occurs since the sum cannot be positive. We crossed the lower boundary 000 and should subtract 2^M from the result. If the resulting binary number has a leading bit of 1, no underflow occurs.
  - When adding a negative integer and a nonnegative integer: 1XX...X + 0YY...Y, the result is always between the two summands. No overflow or underflow could happen.

Floating-point number system



Floating-point number system $\mathbb{F}$ is a computer model for real numbers $\mathbb{R}$.

- A real is represented by $\pm d_0.d_1d_2\cdots d_p \times b^e$ (scientific notation).
- Parameters for a floating-point number system: *base* (or *radix*), range of *fraction* (or *mantissa*, *significand*), range of *exponent*.
- Non-uniqueness of representation. *normalized/denormalized* significant digits. E.g., $+18 = +1.0010 \times 2^4$(normalied) $= +0.10010 \times 2^5$(denormalized).
- *Bias* (or *excess*): actual exponent is obtained by subtracting bias from the value of exponent evaluated regardless of sign digit.
- IEEE 754-1985 and IEEE 754-2008.
  - *Single precision* (32 bit): base 2, $p = 23$ (23 significant bits), $e_{\max} = 127$, $e_{\min} = -126$ (8 exponent bits), bias=127. $e_{\min} - 1$ and $e_{\max} + 1$ are reserved for special numbers. This implies a maximum magnitude of $\log_{10}(2^{127}) \approx 38$ and precision to $\log_{10}(2^{23}) \approx 7$ decimal point. $\pm 10^{\pm 38}$.
  - *Double precision* (64 bit): base 2, $p = 52$ (52 significant bits), $e_{\max} = 1023$, $e_{\min} = -1022$ (11 exponent bits), bias=1023. This implies a maximum magnitude of $\log_{10}(2^{1023}) \approx 308$ and precision to $\log_{10}(2^{52}) \approx 16$ decimal point. $\pm 10^{\pm 308}$.

- "$(+18) = (2^4 + 2^1) = +1.0010 \times 2^4$ in single precision

$$(0)(10000011)(00100000000000000000000).$$

First is sign bit. Next 8 bits are exponent 131 in ordinary base 2 with a bias of 127. Remaining 23 bits represent the fraction beyond the leading bit, known to be 1. In summary it represents (+18) as $+1.0010 \times 2^4$ in the binary format. (-18) is represented by the same bits except changing the sign bit to 1.

- Special floating-point numbers:
  - Exponent $e_{\text{max}} + 1$ plus a mantissa of 0 means $\pm\infty$.
  - Exponent $e_{\text{max}} + 1$ plus a nonzero mantissa means NaN. NaN could be produced from `0/0`, `0*Inf`, ... In general $NaN \neq NaN$.
  - Exponent $e_{\text{min}} - 1$ with a mantissa of all 0s represents the real number 0.
  - Exponent $e_{\text{min}} - 1$ with a nonzero mantissa are for numbers less than $b^{e_{\text{min}}}$. Numbers are de-normalized in the range $(0, b^{e_{\text{min}}})$ – "graceful underflow".
- $\mathbb{F}$ is not a subset of $\mathbb{R}$, although $\mathbb{I} \subset \mathbb{Z}$.
- R only uses double (64-bit) and 32-bit integer. It can be a downside when dealing with big data.

To summarize

- Single precision: $\pm 10^{\pm 38}$ with precision up to 7 decimal digits.
- Double precision: $\pm 10^{\pm 308}$ with precision up to 16 decimal digits.
- Irrational numbers such as π do not exist in $\mathbb{F}$.
- Exercise: what is the floating point representation of the number 0.1?
- The floating-point numbers do not occur uniformly over the real number line.

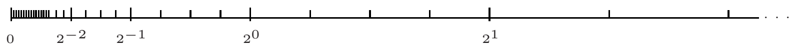


Fig. 2.4. The Floating-Point Number Line, Nonnegative Half

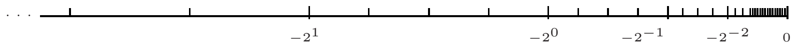


Fig. 2.5. The Floating-Point Number Line, Nonpositive Half

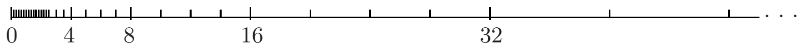


Fig. 2.6. The Floating-Point Number Line, Nonnegative Half; Another View

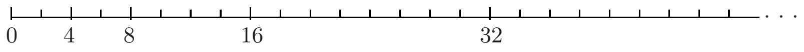


Fig. 2.7. The Fixed-Point Number Line, Nonnegative Half

- *Machine epsilons* are the spacings of numbers around 1. $\epsilon_{\min} = b^{-p}$ and $\epsilon_{\max} = b^{1-p}$.

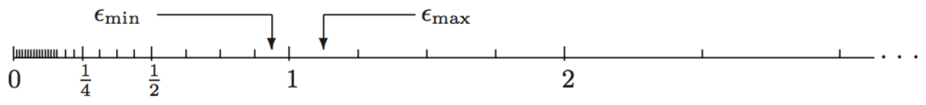


Fig. 2.8. Relative Spacings at 1: "Machine Epsilons"

- The variable `.Machine` in R contains numerical characteristics of the machine.
- How to test `inf` and `nan`? In R, `is.nan()`, `is.finite()`, `is.infinite()`. In MATLAB, `isinf()`, `isnan()`.

3 Lecture 3, Sep 4

Announcements

- TA office hours: Wed @ 1P-3P or Wed @ 10A-12P?

Last time

- Computer representation of characters
- Fixed-point number system for integers
- Floating-point number system (IEEE 754 standards for single/double precision)

Today

- Consequences of compute storage/arithmetic
- Algorithms
- Computer languages

Consequences of computer storage/arithmetic

- Be memory conscious when dealing with big data. E.g., human genome has about 3×10^9 bases, each of which belongs to $\{A, C, T, G\}$. How much storage if we store 10^6 SNPs (single nucleotide polymorphisms) of 1000 individuals (1000 Genome Project) as characters (1GB), single (4GB), double (8GB), int32 (4GB), int16 (2GB), int8 (1GB), PLINK binary format 2bit/SNP (250MB)?

- Know the limit. *Overflow* and *Underflow*. For double precision, $\pm 10^{\pm 308}$. In most situations, underflow is preferred over overflow. Overflow often causes crashes. Underflow yields zeros. E.g., in logistic regression, $p_i = \frac{\exp(\boldsymbol{x}_i^\mathsf{T}\boldsymbol{\beta})}{1+\exp(\boldsymbol{x}_i^\mathsf{T}\boldsymbol{\beta})} = \frac{1}{1+\exp(-\boldsymbol{x}_i^\mathsf{T}\boldsymbol{\beta})}$. The former expression can easily lead to $\infty/\infty = NaN$, while the latter expression leads to graceful underflow.

- Be aware of non-uniform distribution of floating-point numbers, in contrast to fixed-point numbers. There are the same number of floating-point numbers in

$[b^i, b^{i+1}]$ and $[b^{i+1}, b^{i+2}]$ for $e_{\min} \le i \le e_{\max} - 2$. It is more dense when closer to zero.

- "Catastrophic cancellation 1". Addition or subtraction of two numbers of widely different magnitudes: $a + b$ or $a - b$ where $a \gg b$ or $a \ll b$. We loose the precision in the number of smaller magnitude. Consider $a = x.xxx... \times 2^0$ and $b = y.yyy... \times 2^{-53}$. What happens when computer calculates $a + b$? We get $a + b = a$!
- Another example: What happens when compute $\sum_{x=1}^{\infty} x$ in order? Will the partial sum reach `Inf`? "A divergent series converges."
- Always try to add numbers of similar magnitude. Rule 1: add small numbers together before adding larger ones. Rule 2: add numbers of like magnitude together (paring). When all numbers are of same sign and similar magnitude, add in pairs so each stage the summands are of similar magnitude.

$$\begin{tikzcd}
s_1^{(1)} = x_1 + x_2 \arrow[d] & s_2^{(1)} = x_3 + x_4 \arrow[dl] & \cdots & s_{2m-1}^{(1)} = x_{4m-3} + x_{4m-2} \arrow[d] & s_{2m}^{(1)} = \arrow[dl] & \cdots \\
s_1^{(2)} = s_1^{(1)} + s_2^{(1)} \arrow[d] & & \cdots & s_m^{(2)} = s_{2m-1}^{(1)} + s_{2m}^{(1)} \arrow[d] & & \cdots \\
s_1^{(3)} = s_1^{(2)} + s_2^{(2)} & & \cdots & \cdots & & \cdots
\end{tikzcd}$$

- "Catastrophic cancellation 2". Subtraction of two nearly equal numbers eliminates significant digits. $a - b$ where $a \approx b$. Consider $a = x.xxxxxxxxxx1ssss$, $b = x.xxxxxxxxxx0tttt$. The result is $1.vvvvu...u$ where u are unassigned digits.
- Violation of associative law and distributive law.

More about cancellation

- Evaluating e^{-20} by Taylor series

$$e^{-x} = 1 - x + x^2/2! - x^3/3! + \cdots$$

gets `6.138e-09`, while the true value is about `2.061e-09`. Many cancellations accumulate. Anyway, it's *not* the way to evaluate e^x!

- Sometimes catastrophic cancellation can be avoided. Roots of the quadratic function $ax^2 + bx + c$ are

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$$

When one root is close to 0, cancellation can happen. We may evaluate one of the root (away from 0) by the formula and then compute the other by relationship $x_1 x_2 = c/a$.

Algorithms

- *Algorithm* is loosely defined as a set of instructions for doing something. Input $\to$ Output.

- Knuth (2005): (1) finiteness, (2) definiteness, (3) input, (4) output, (5) effectiveness

- Basic unit for measuring efficiency is flop. A *flop* (floating point operation) consists of a floating point multiply (or divide) and the usually accompanying addition, fetch and store. Some books such as Lange (2010) and Golub and Van Loan (2013) consider addition as a separate flop.

- How to measure efficiency of an algorithm? Big O notation. If n is the size of a problem, an algorithm has order $O(f(n))$ if, as $n \to \infty$, the number of computations $\to cf(n)$, where c is some constant that does not depend on n.

- E.g., matrix-vector multiplication `A%*%b`, where $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ and $\boldsymbol{b} \in \mathbb{R}^n$, takes $O(mn)$ flops. Matrix-matrix multiplication `A%*%B`, where $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ and $\boldsymbol{B} \in \mathbb{R}^{n \times p}$, takes $O(mnp)$ flops.

- *Exponential* order $O(b^n)$ (NP-hard="horrible"), *polynomial* order $O(n^q)$ (doable), $O(n \log n)$ (fast), linear order $O(n)$ (fast), *log* order $O(\log n)$ (fast).

- One goal of this course is to get familiar with the flop counts for common numerical tasks in statistics.

> The form of a mathematical expression and the way the expression should be evaluated in actual practice may be quite different.

- Compare flops of the following two expressions:

```
G %*% Xt %*% y
G %*% (Xt %*% y)
```

where `G` $\in \mathbb{R}^{p\times p}$, `Xt` $\in \mathbb{R}^{p\times n}$, and `y` $\in \mathbb{R}^{n}$. "Matrix multiplication is expensive."

- Hardware advancement, e.g., CPU clock rate, only affects constant c. Unfortunately, data size n is increasing too and often at a faster rate.

- Classification of data sets by Huber (1994, 1996).

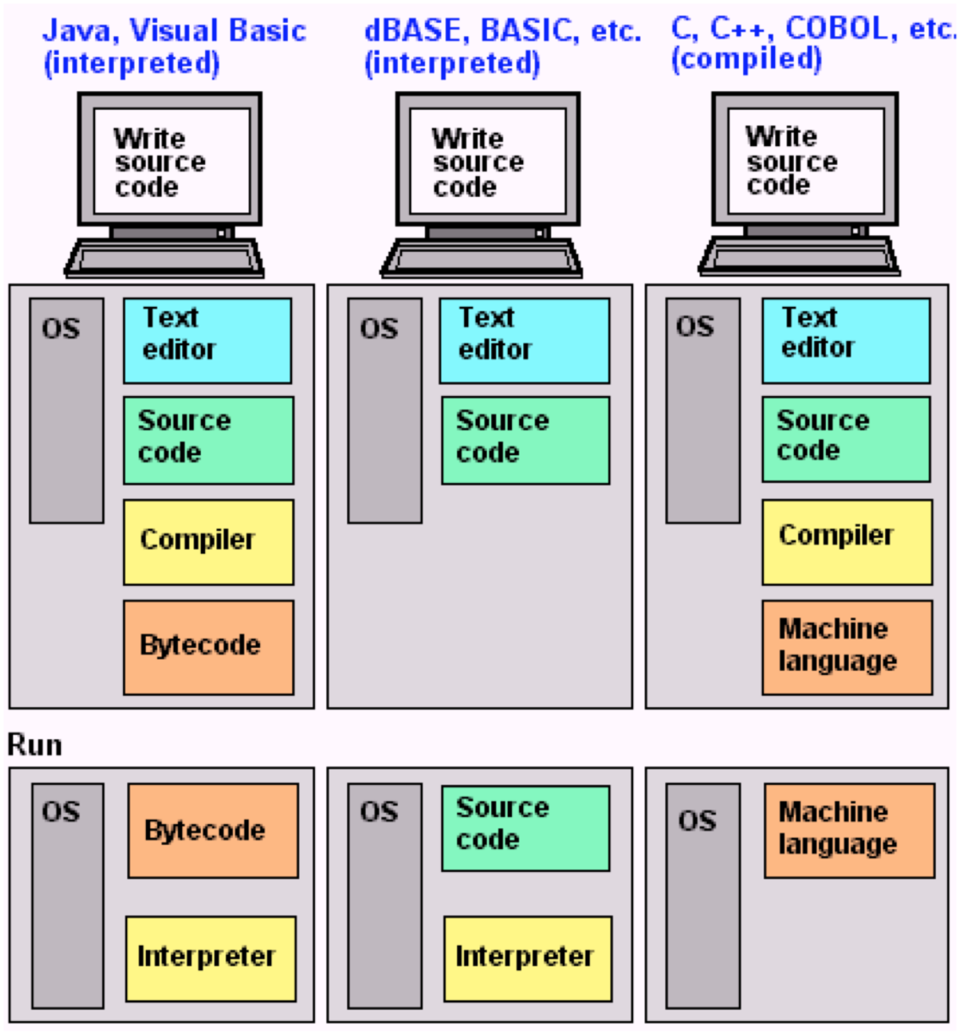
Data Size	Bytes	Storage Mode
Tiny	10^2	Piece of paper
Small	10^4	A few pieces of paper
Medium	10^6 (megabyte)	A floppy disk
Large	10^8	Hard disk
Huge	10^9 (gigabytes)	Hard disk(s)
Massive	10^{12} (terabytes)	RAID storage

- It is an era of "big data": wiki, WSJ, white house, McKinsey report, ..., meaning great opportunities for statisticians. However we should be aware of the gap between classical statistics curriculum and reality.

- Difference of $O(n^2)$ and $O(n \log n)$ on massive data. Suppose we have a teraflop supercomputer – capable of doing 10^{12} flops per second. For a problem of size $n = 10^{12}$, $O(n \log n)$ algorithm takes about $10^{12} \log(10^{12})/10^{12} \approx 27$ seconds. $O(n^2)$ algorithm takes about 10^{12} seconds, which is approximately 31710 years!

- QuickSort and FFT are celebrated algorithms that turn $O(n^2)$ operations into $O(n \log n)$. "Divide-and-conquer" is a powerful technique. Another example is the Strassen's method, which turns $O(n^3)$ matrix multiplication into $O(n^{\log_2 7})$.

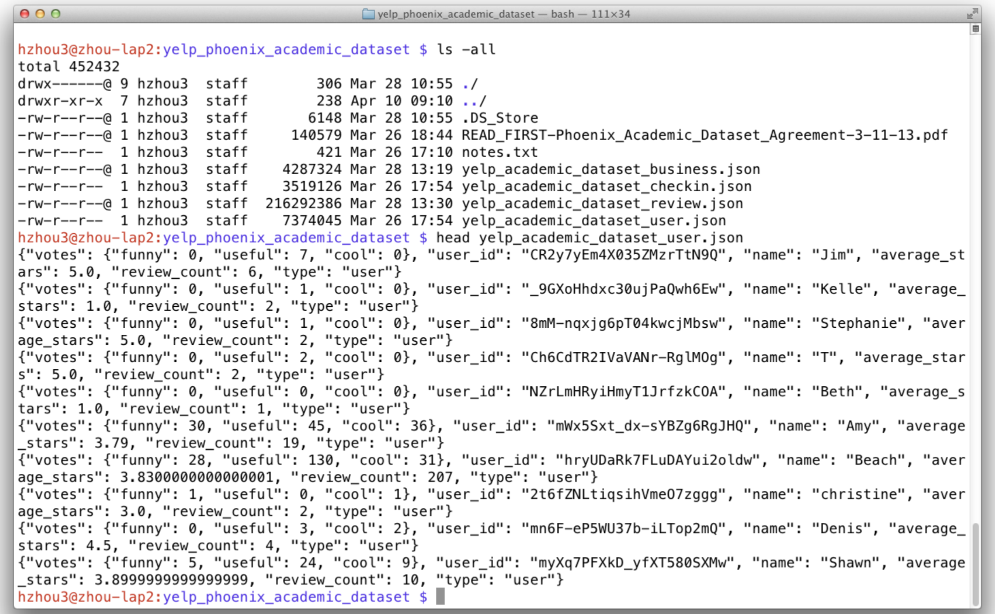
Computer languages



Compiled versus interpreted languages.

- Compiled languages: C/C++, Fortran, ... Directly compiled to machine code that is executed by CPU. Advantage: fast, take less memory. Disadvantage: relatively longer development time, hard to debug.
- Interpreted language: R, Matlab, SAS IML, ... Interpreted by interpreter. Advantage: fast for prototyping. Disadvantage: excruciatingly slow for loops.
- Mixed (compiled and then interpreted by virtual machine): Python, JAVA. Advantage: extremely convenient for data preprocessing and manipulation; relatively short development time. Disadvantage: not as fast as compiled language.

- Scripting: Unix/Linux scripts, Perl, Python. Extremely useful for data pre-processing and manipulation. E.g., massage the Yelp (`http://www.yelp.com/dataset_challenge`) data before analysis



yelp_phoenix_academic_dataset — bash — 111×34

```
hzhou3@zhou-lap2:yelp_phoenix_academic_dataset $ ls -all
total 452432
drwx------@ 9 hzhou3  staff        306 Mar 28 10:55 ./
drwxr-xr-x  7 hzhou3  staff        238 Apr 10 09:10 ../
-rw-r--r--@ 1 hzhou3  staff       6148 Mar 28 10:55 .DS_Store
-rw-r--r--@ 1 hzhou3  staff     140579 Mar 26 18:44 READ_FIRST-Phoenix_Academic_Dataset_Agreement-3-11-13.pdf
-rw-r--r--  1 hzhou3  staff        421 Mar 26 17:10 notes.txt
-rw-r--r--@ 1 hzhou3  staff    4287324 Mar 28 13:19 yelp_academic_dataset_business.json
-rw-r--r--  1 hzhou3  staff    3519126 Mar 26 17:54 yelp_academic_dataset_checkin.json
-rw-r--r--@ 1 hzhou3  staff  216292386 Mar 28 13:30 yelp_academic_dataset_review.json
-rw-r--r--  1 hzhou3  staff    7374045 Mar 26 17:54 yelp_academic_dataset_user.json
hzhou3@zhou-lap2:yelp_phoenix_academic_dataset $ head yelp_academic_dataset_user.json
{"votes": {"funny": 0, "useful": 7, "cool": 0}, "user_id": "CR2y7yEm4X035ZMzrTtN9Q", "name": "Jim", "average_st
ars": 5.0, "review_count": 6, "type": "user"}
{"votes": {"funny": 0, "useful": 1, "cool": 0}, "user_id": "_9GXoHhdxc30ujPaQwh6Ew", "name": "Kelle", "average_
stars": 1.0, "review_count": 2, "type": "user"}
{"votes": {"funny": 0, "useful": 1, "cool": 0}, "user_id": "8mM-nqxjg6pT04kwcjMbsw", "name": "Stephanie", "aver
age_stars": 5.0, "review_count": 2, "type": "user"}
{"votes": {"funny": 0, "useful": 2, "cool": 0}, "user_id": "Ch6CdTR2IVaVANr-RglMOg", "name": "T", "average_star
s": 5.0, "review_count": 2, "type": "user"}
{"votes": {"funny": 0, "useful": 0, "cool": 0}, "user_id": "NZrLmHRyiHmyT1JrfzkCOA", "name": "Beth", "average_s
tars": 1.0, "review_count": 1, "type": "user"}
{"votes": {"funny": 30, "useful": 45, "cool": 36}, "user_id": "mWx5Sxt_dx-sYBZg6RgJHQ", "name": "Amy", "average
_stars": 3.79, "review_count": 19, "type": "user"}
{"votes": {"funny": 28, "useful": 130, "cool": 31}, "user_id": "hryUDaRk7FLuDAYui2oldw", "name": "Beach", "aver
age_stars": 3.8300000000000001, "review_count": 207, "type": "user"}
{"votes": {"funny": 1, "useful": 0, "cool": 1}, "user_id": "2t6fZNLtiqsihVmeO7zggg", "name": "christine", "aver
age_stars": 3.0, "review_count": 2, "type": "user"}
{"votes": {"funny": 0, "useful": 3, "cool": 2}, "user_id": "mn6F-eP5WU37b-iLTop2mQ", "name": "Denis", "average_
stars": 4.5, "review_count": 4, "type": "user"}
{"votes": {"funny": 5, "useful": 24, "cool": 9}, "user_id": "myXq7PFXkD_yfXT580SXMw", "name": "Shawn", "average
_stars": 3.8999999999999999, "review_count": 10, "type": "user"}
hzhou3@zhou-lap2:yelp_phoenix_academic_dataset $
```

- Database language: SQL, Hadoop. Data analysis never happens if we do not know how to retrieve data from databases.

4 Lecture 4, Sep 9

Announcements

- TA office hours. Fri @ 10A-12P, Wed @ 1P-3P, Wed @ 10A-12P, or Thu @ 1P-2P?
- Some resources for R programming:
  - Dr. John Monahan's class (2013 fall) on R: `http://www.stat.ncsu.edu/people/monahan/courses/ST610/`
  - Code school: `http://tryr.codeschool.com/levels/1/challenges/1`
  - *Advanced R* by Hadley Wickham: `http://adv-r.had.co.nz/`

Last time

- Consequences of computer storage/arithmetics (overflow, underflow, catastrophic cancellation, ...)
- Algorithms, flops, big O notation
- Computer languages (compiled, interpreted, mixed, scripting)

Today

- Review of linear algebra (self-study and do HW2)
- Numerical linear algebra: preliminaries

More about computer languages

- To improve efficiency of interpreted languages such as R or MATLAB code, avoid loops as much as possible. Aka, vectorize code.
- For some tasks where looping is necessary (cannot vectorize code), consider coding in C/C++ or FORTRAN. It is convenient to incorporate compiled code into R or MATLAB.

- To be versatile in dealing with big data, master at least one language in each category. Take advantage of the resource in the department. E.g., check out Chris's class on Python, John's class on R programming, ...

- Don't reinvent wheels. Make good use of libraries blas, lapack/linpack, boost, SciPy, NumPy, ...

- One example of my recent pedigree GWAS project, which fits large variance components model on $n \sim 10^3$ persons and $\sim 10^6$ SNPs. Mendel implements the method using Fortran, takes advantage of problem structure, and is flop frugal. GWAF (an R package) uses the canned routine in R for fitting variance component, which does not scale up.

# SNPs	MENDEL	GWAF-GEE	GWAF-LMM
100	4.81	0.71	8.83
1,000	4.80	7.71	87.06
10,000	5.40	207.60	894.82
100,000	10.78	26,486.92	11,703.88

Table 1: Comparison of run times (in seconds). Run times are based on testing the first 100, 1000, 10000 and 100000 SNPs on chromosome 19. Trait values are from the same simulation replicate in Figure 3.

  Many aspects could contribute to this dramatic difference in efficiency: looping, compiled vs interpreted language, problem structure, choice of data types (remember R only does double precision), memory management, ...

- Distinction between compiled language and interpreted language is getting blurred. JIT (just-in-time) compilation technology in Matlab since 2002 (v6.5). compiler package in R for JIT compiling since 2012.

Numerical linear algebra

- The first big chunk of this course is numerical linear algebra.

- Topics in numerical algebra: BLAS, solve linear equations $\boldsymbol{Ax} = \boldsymbol{b}$, regression computations $\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{\beta} = \boldsymbol{X}^T\boldsymbol{y}$, eigen-problems $\boldsymbol{Ax} = \lambda\boldsymbol{x}$ and generalized eigen-problems $\boldsymbol{Ax} = \lambda\boldsymbol{Bx}$, singular value decompositions $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$, ...

- We start with review of some linear algebra facts (review by yourself and do HW2).

Vector and matrix norms

KL chapter 6. Norm measures the "size".

- Vector norm $\|\cdot\| : \mathbb{R}^n \to \mathbb{R}$. (a) $\|\boldsymbol{x}\| \geq 0$, (b) $\|\boldsymbol{x}\| = 0$ if and only if $\boldsymbol{x} = \boldsymbol{0}$, (c) $\|c\boldsymbol{x}\| = c\|\boldsymbol{x}\|$ (homogeneity), (d) $\|\boldsymbol{x} + \boldsymbol{y}\| \leq \|\boldsymbol{x}\| + \|\boldsymbol{y}\|$ (triangle inequality).

- ℓ_p norm. $\|\boldsymbol{x}\|_p = (\sum_i |x_i|^p)^{1/p}$, $p \in [1, \infty]$. ℓ_1 is the Manhattan norm. ℓ_2 is the Euclidean norm. ℓ_∞ is the sup norm.

- For matrix norm $\|\cdot\| : \mathbb{R}^{m \times n} \to \mathbb{R}$, we further require (e) $\|\boldsymbol{AB}\| \leq \|\boldsymbol{A}\|\|\boldsymbol{B}\|$.

- Frobenius norm $\|\boldsymbol{A}\|_{\mathrm{F}} = (\sum_i \sum_j a_{ij}^2)^{1/2}$. Properties of (e) is checked by Cauchy-Schwartz inequality.

- Induced matrix norm (or operator norm): $\|\boldsymbol{A}\| = \sup_{\boldsymbol{x} \neq \boldsymbol{0}} \frac{\|\boldsymbol{Ax}\|}{\|\boldsymbol{x}\|} = \sup_{\|\boldsymbol{x}\|=1} \|\boldsymbol{Ax}\|$ for any fixed vector norm. To check property (e), let $\boldsymbol{y} = \boldsymbol{Bx}$, then $\|\boldsymbol{AB}\| = \sup_{\boldsymbol{x} \neq \boldsymbol{0}} \frac{\|\boldsymbol{Ay}\|}{\|\boldsymbol{y}\|} \frac{\|\boldsymbol{Bx}\|}{\|\boldsymbol{x}\|} \leq \|\boldsymbol{A}\| \sup_{\boldsymbol{x} \neq \boldsymbol{0}} \frac{\|\boldsymbol{Bx}\|}{\|\boldsymbol{x}\|} = \|\boldsymbol{A}\|\|\boldsymbol{B}\|$.

- Matrix-1 norm, $\|\boldsymbol{A}\|_1 = \max_j \sum_i |a_{ij}|$.
  Matrix-2 norm, $\|\boldsymbol{A}\|_2 = \sqrt{\rho(\boldsymbol{A}^{\mathsf{T}}\boldsymbol{A})} = \max_{\|\boldsymbol{u}\|_2=1, \|\boldsymbol{v}\|_2=1} \boldsymbol{u}^{\mathsf{T}}\boldsymbol{Av}$, which reduces to $\rho(\boldsymbol{A})$ if $\boldsymbol{A}$ is symmetric. ρ is the spectral radius of a matrix, the absolute value of the dominant eigenvalue.
  Matrix-∞ norm, $\|\boldsymbol{A}\|_\infty = \max_i \sum_j |a_{ij}|$.

- When $\boldsymbol{A}$ is a column vector, these matrix induced norms reduce to the original vector norm.

- $\rho(\boldsymbol{A}) \leq \|\boldsymbol{A}\|$ for any induced matrix norm. For any $\boldsymbol{A}$ and $\epsilon > 0$, there exists an induced matrix norm such that $\|\boldsymbol{A}\| \leq \rho(\boldsymbol{A}) + \epsilon$.

Rank

Assume $\boldsymbol{A} \in \mathbb{R}^{m \times n}$.

- rank($\boldsymbol{A}$) is the maximum number of linearly independent rows (or columns) of a matrix.

- $\text{rank}(\boldsymbol{A}) \leq \min\{m, n\}$.
- A matrix is *full rank* if $\text{rank}(\boldsymbol{A}) = \min\{m, n\}$. It is *full row rank* if $\text{rank}(\boldsymbol{A}) = m$. It is *full column rank* if $\text{rank}(\boldsymbol{A}) = n$.
- A square matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ is *singular* if $\text{rank}(\boldsymbol{A}) < n$ and *non-singular* if $\text{rank}(\boldsymbol{A}) = n$.
- $\text{rank}(\boldsymbol{AB}) \leq \min\{\text{rank}(\boldsymbol{A}), \text{rank}(\boldsymbol{B})\}$. "Matrix multiplication cannot increase the rank."
- $\text{rank}(\boldsymbol{A}) = \text{rank}(\boldsymbol{A}^T) = \text{rank}(\boldsymbol{A}^T\boldsymbol{A}) = \text{rank}(\boldsymbol{A}\boldsymbol{A}^T)$.
- $\text{rank}(\boldsymbol{AB}) = \text{rank}(\boldsymbol{A})$ if $\boldsymbol{B}$ has full row rank.
- $\text{rank}(\boldsymbol{AB}) = \text{rank}(\boldsymbol{B})$ if $\boldsymbol{A}$ has full column rank.
- $\text{rank}(\boldsymbol{A} + \boldsymbol{B}) \leq \text{rank}(\boldsymbol{A}) + \text{rank}(\boldsymbol{B})$.

Trace

$\boldsymbol{A} \in \mathbb{R}^{n\times n}$ a square matrix.

- $\text{tr}(\boldsymbol{A}) = \sum_{i=1}^{n} a_{ii}$
- $\text{tr}(\boldsymbol{A} + \boldsymbol{B}) = \text{tr}(\boldsymbol{A}) + \text{tr}(\boldsymbol{B})$
- $\text{tr}(\lambda\boldsymbol{A}) = \lambda\text{tr}(\boldsymbol{A})$ where λ is a scalar
- $\text{tr}(\boldsymbol{A}^\intercal) = \text{tr}(\boldsymbol{A})$
- Invariance under cycle permutation: $\text{tr}(\boldsymbol{AB}) = \text{tr}(\boldsymbol{BA})$. In general, $\text{tr}(\boldsymbol{A}_1 \cdots \boldsymbol{A}_k) = \text{tr}(\boldsymbol{A}_{j+1} \cdots \boldsymbol{A}_k\boldsymbol{A}_1 \cdots \boldsymbol{A}_j)$.

Orthogonality and orthogonalization

- $\boldsymbol{v}_1$ is *orthogonal* to $\boldsymbol{v}_2$, written $\boldsymbol{v}_1 \perp \boldsymbol{v}_2$, if $\langle \boldsymbol{v}_1, \boldsymbol{v}_2\rangle = \boldsymbol{v}_1^\intercal\boldsymbol{v}_2 = 0$. They are *orthonormal* if $\boldsymbol{v}_1 \perp \boldsymbol{v}_2$ and $\|\boldsymbol{v}_i\|_2 = 1$, $i = 1, 2$.

- Gram-Schmidt transformation orthonormalizes two non-zero vectors $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$.

$$\begin{aligned}
\tilde{\boldsymbol{x}}_1 &= \frac{1}{\|\boldsymbol{x}_1\|_2}\boldsymbol{x}_1 \\
\tilde{\boldsymbol{x}}_2 &= \frac{1}{\|\boldsymbol{x}_2 - \langle \tilde{\boldsymbol{x}}_1, \boldsymbol{x}_2 \rangle \tilde{\boldsymbol{x}}_1\|_2}(\boldsymbol{x}_2 - \langle \tilde{\boldsymbol{x}}_1, \boldsymbol{x}_2 \rangle \tilde{\boldsymbol{x}}_1)
\end{aligned}$$

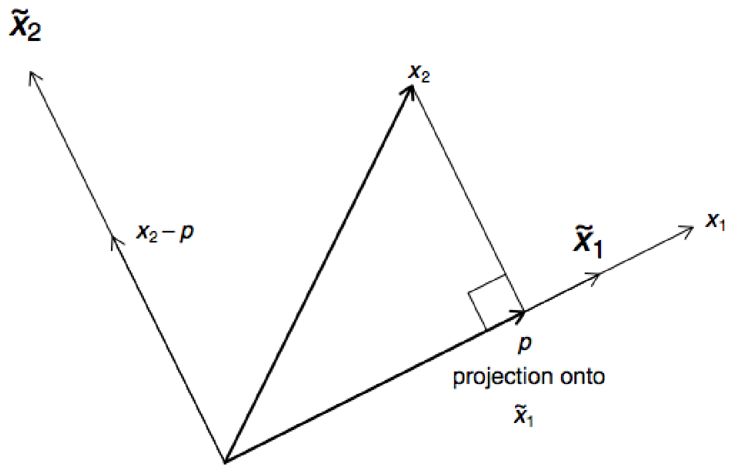


Fig. 2.2. Orthogonalization of x_1 and x_2

- A set of nonzero, mutually orthogonal vectors are linearly independent.

- A real square matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ is orthogonal if $\boldsymbol{A}^\intercal \boldsymbol{A} = \boldsymbol{I}_n$, i.e., its rows/columns are orthonormal. Orthogonal matrix is of full rank, thus $\boldsymbol{A}^\intercal = \boldsymbol{A}^{-1}$ and $\boldsymbol{A}\boldsymbol{A}^\intercal = \boldsymbol{I}_n$.

Positive (semi)definite matrix

Assume $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ is *symmetric*.

- A real symmetric matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ is *positive semidefinite* (or *nonnegative definite*) if $\boldsymbol{x}^\intercal \boldsymbol{A}\boldsymbol{x} \geq 0$ for all $\boldsymbol{x}$. Notation: $\boldsymbol{A} \succeq \boldsymbol{0}_{n\times n}$.

- E.g., the *Gramian matrix* $\boldsymbol{X}^\intercal \boldsymbol{X}$ or $\boldsymbol{X}\boldsymbol{X}^T$.

- If inequality is strict for all $\boldsymbol{x} \neq \boldsymbol{0}$, then $\boldsymbol{A}$ is *positive definite.* Notation: $\boldsymbol{A} \succ \boldsymbol{0}_{n\times n}$.

- $\boldsymbol{A} \succeq \boldsymbol{B}$ means $\boldsymbol{A} - \boldsymbol{B} \succeq \boldsymbol{0}_{n\times n}$.

- If $\boldsymbol{A} \succeq \boldsymbol{B}$, then $\det(\boldsymbol{A}) \geq \det(\boldsymbol{B})$ with equality if and only if $\boldsymbol{A} = \boldsymbol{B}$.
- $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ is positive semidefinite if and only if $\boldsymbol{A}$ is a covariance matrix of a random vector in $\mathbb{R}^n$.

Matrix inverses

Assume $\boldsymbol{A} \in \mathbb{R}^{m \times n}$.

- The *Moore-Penrose inverse* of $\boldsymbol{A}$ is a matrix $\boldsymbol{A}^+ \in \mathbb{R}^{n \times m}$ with following properties
  (a) $\boldsymbol{A}\boldsymbol{A}^+\boldsymbol{A} = \boldsymbol{A}$. (*Generalized inverse*, g_1 *inverse*, or *inner pseudo-inverse*)
  (b) $\boldsymbol{A}^+\boldsymbol{A}\boldsymbol{A}^+ = \boldsymbol{A}^+$. (*Outer pseudo-inverse.* Any g_1 inverse that satisfies this condition is called a g_2 *inverse*, or *reflexive generalized inverse* and is denoted by $\boldsymbol{A}^*$.)
  (c) $\boldsymbol{A}^+\boldsymbol{A}$ is symmetric.
  (d) $\boldsymbol{A}\boldsymbol{A}^+$ is symmetric.
- $\boldsymbol{A}^+$ exists and is unique for any matrix $\boldsymbol{A}$.
- *Generalized inverse* (or g_1 *inverse*, denoted by $\boldsymbol{A}^-$ or $\boldsymbol{A}^g$): property (a).
- g_2 *inverse* (denoted by $\boldsymbol{A}^*$): properties (a)+(b).
- *Moore-Penrose inverse* (denoted by $\boldsymbol{A}^+$): properties (a)+(b)+(c)+(d).
- If $\boldsymbol{A}$ is square and full rank, then the generalized inverse is unique and denoted by $\boldsymbol{A}^{-1}$ (inverse).
- In practice, the Moore-Penrose inverse $\boldsymbol{A}^+$ is easily computed from the singular value decomposition (SVD) of $\boldsymbol{A}$.
- $(\boldsymbol{A}^-)^T$ is a generalized inverse of $\boldsymbol{A}^T$.
- $\mathcal{C}(\boldsymbol{A}) = \mathcal{C}(\boldsymbol{A}\boldsymbol{A}^-)$ and $\mathcal{C}(\boldsymbol{A}^T) = \mathcal{C}((\boldsymbol{A}^-\boldsymbol{A})^T)$.
  $\text{rank}(\boldsymbol{A}) = \text{rank}(\boldsymbol{A}\boldsymbol{A}^-) = \text{rank}(\boldsymbol{A}^-\boldsymbol{A})$.
  "Multiplication by generalized inverse does not change rank."
- $\text{rank}(\boldsymbol{A}^-) \geq \text{rank}(\boldsymbol{A})$. "Generalized inverse has equal or a larger rank than the original matrix."

System of linear equations

$\boldsymbol{Ax} = \boldsymbol{b}$ where $\boldsymbol{A} \in \mathbb{R}^{m \times n}$, $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{b} \in \mathbb{R}^m$.

- When is there a solution? The following statements are equivalent.

    1. The linear system $\boldsymbol{Ax} = \boldsymbol{b}$ has a solution (*consistent*)
    2. $\boldsymbol{b} \in \mathcal{C}(\boldsymbol{A})$.
    3. $\text{rank}((\boldsymbol{A}, \boldsymbol{b})) = \text{rank}(\boldsymbol{A})$.
    4. $\boldsymbol{AA}^- \boldsymbol{b} = \boldsymbol{b}$.

    The last equivalence gives intuition why $\boldsymbol{A}^-$ is called an inverse.

- What are the solutions to a homogeneous system $\boldsymbol{Ax} = \boldsymbol{0}$?
  $\mathcal{N}(\boldsymbol{A}) = \mathcal{C}(\boldsymbol{I}_n - \boldsymbol{A}^- \boldsymbol{A})$.

- If $\boldsymbol{Ax} = \boldsymbol{b}$ is consistent, then $\tilde{\boldsymbol{x}}$ is a solution to $\boldsymbol{Ax} = \boldsymbol{b}$ if and only if

$$\tilde{\boldsymbol{x}} = \boldsymbol{A}^- \boldsymbol{b} + (\boldsymbol{I}_n - \boldsymbol{A}^- \boldsymbol{A})\boldsymbol{q}$$

  for some $\boldsymbol{q} \in \mathbb{R}^n$.
  Interpretation: "a specific solution" + "a vector in the null space of $\boldsymbol{A}$".

- $\boldsymbol{Ax} = \boldsymbol{b}$ is consistent for *all* $\boldsymbol{b}$ if and only if $\boldsymbol{A}$ has full row rank.

- If a system is consistent, its solution is unique if and only if $\boldsymbol{A}$ has full column rank.

- If $\boldsymbol{A}$ has full row and column rank, then $\boldsymbol{A}$ is non-singular and the unique solution is $\boldsymbol{A}^{-1}\boldsymbol{b}$.

Gramian matrix $\boldsymbol{A}^\intercal \boldsymbol{A}$

- $\boldsymbol{A}^\intercal \boldsymbol{A}$ is symmetric and positive semidefinite.

- $\text{rank}(\boldsymbol{A}) = \text{rank}(\boldsymbol{A}^\intercal) = \text{rank}(\boldsymbol{A}^\intercal \boldsymbol{A}) = \text{rank}(\boldsymbol{A}\boldsymbol{A}^\intercal)$.

- $\boldsymbol{A}^\intercal \boldsymbol{A} = \boldsymbol{0}$ if and only if $\boldsymbol{A} = \boldsymbol{0}$.

- $\boldsymbol{BA}^\intercal \boldsymbol{A} = \boldsymbol{CA}^\intercal \boldsymbol{A}$ if and only if $\boldsymbol{BA}^\intercal = \boldsymbol{CA}^\intercal$.

- $\boldsymbol{A}^\intercal \boldsymbol{A}\boldsymbol{B} = \boldsymbol{A}^\intercal \boldsymbol{A}\boldsymbol{C}$ if and only if $\boldsymbol{A}\boldsymbol{B} = \boldsymbol{A}\boldsymbol{C}$.
- For any generalized inverse $(\boldsymbol{A}^\intercal \boldsymbol{A})^-$, $[(\boldsymbol{A}^\intercal \boldsymbol{A})^-]^\intercal$ is also a generalized inverse of $\boldsymbol{A}^\intercal \boldsymbol{A}$. Note $(\boldsymbol{A}^\intercal \boldsymbol{A})^-$ is not necessarily symmetric.
- $(\boldsymbol{A}^\intercal \boldsymbol{A})^- \boldsymbol{A}^\intercal$ is a generalized inverse of $\boldsymbol{A}$.
- $\boldsymbol{A}\boldsymbol{A}^+ = \boldsymbol{A}(\boldsymbol{A}^\intercal \boldsymbol{A})^- \boldsymbol{A}^\intercal$, where $\boldsymbol{A}^+$ is the Moore-Penrose inverse of $\boldsymbol{A}$.
- $\boldsymbol{P}_{\boldsymbol{A}} = \boldsymbol{A}(\boldsymbol{A}^\intercal \boldsymbol{A})^- \boldsymbol{A}^\intercal$ is symmetric, idempotent, invariant to the choice of generalized inverse $(\boldsymbol{A}^\intercal \boldsymbol{A})^-$, and projects onto $\mathcal{C}(\boldsymbol{A})$.

Idempotent matrix and projection

Assume $\boldsymbol{P} \in \mathbb{R}^{n \times n}$.

- A matrix $\boldsymbol{P} \in \mathbb{R}^{n \times n}$ is *idempotent* if and only if $\boldsymbol{P}^2 = \boldsymbol{P}$.
- A matrix $\boldsymbol{P}$ is a *projection* on a vector space $\mathcal{V}$ if (a) $\boldsymbol{P}$ is idempotent, (b) $\boldsymbol{P}\boldsymbol{x} \in \mathcal{V}$ for all $\boldsymbol{x}$, and (c) $\boldsymbol{P}\boldsymbol{z} = \boldsymbol{z}$ for all $\boldsymbol{z} \in \mathcal{V}$.
- An idempotent matrix $\boldsymbol{P}$ is a projection onto $\mathcal{C}(\boldsymbol{P})$.
- For a *general* matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$, the matrices $\boldsymbol{A}\boldsymbol{A}^-$ are projections onto $\mathcal{C}(\boldsymbol{A})$ and $\boldsymbol{I}_n - \boldsymbol{A}^-\boldsymbol{A}$ are projections onto $\mathcal{N}(\boldsymbol{A})$.

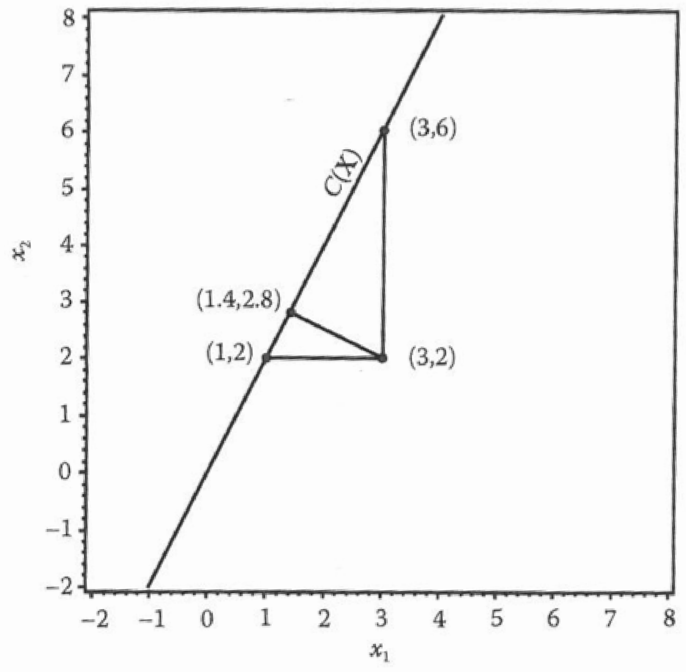


Figure A.1: Three projections onto a column space.

Symmetric idempotent matrix and orthogonal projection

Assume $\boldsymbol{A} \in \mathbb{R}^{n\times n}$.

- A symmetric, idempotent matrix is called an *orthogonal projection*.
- An orthogonal projection $\boldsymbol{P}$ satisfies $\boldsymbol{y} - \boldsymbol{P}\boldsymbol{y} \perp \boldsymbol{v}$ for all $\boldsymbol{v} \in \mathcal{C}(\boldsymbol{P})$.
- The orthogonal projection onto a vector space is unique.
- If a symmetric, idempotent matrix $\boldsymbol{P}$ projects onto $\mathcal{V}$, then $\boldsymbol{I} - \boldsymbol{P}$ projects onto the orthogonal complement $\mathcal{V}^{\perp}$.
- Pythagorean theorem: For $\boldsymbol{P}$ an orthogonal projection,

$$\|\boldsymbol{y}\|_2^2 = \|\boldsymbol{P}\boldsymbol{y}\|_2^2 + \|(\boldsymbol{I} - \boldsymbol{P})\boldsymbol{y}\|_2^2.$$

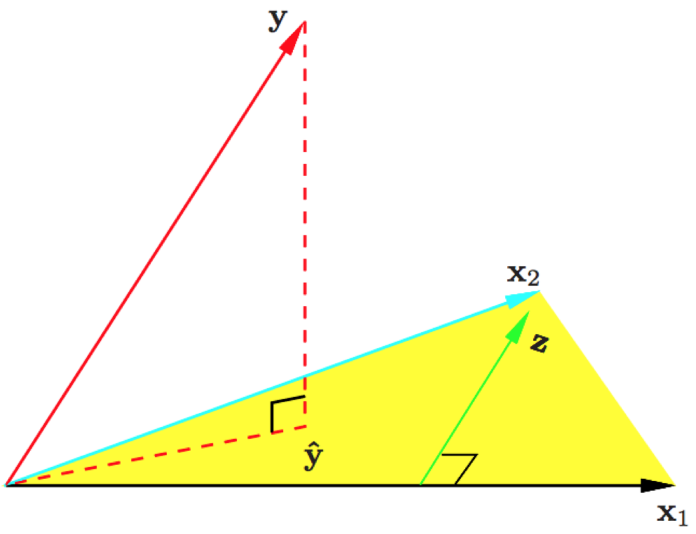
- Many books use the term "projection" in the sense of of orthogonal projection.

Method of least squares

- Goal: Approximate $\boldsymbol{y} \in \mathbb{R}^n$ by a linear combination of columns of $\boldsymbol{X} \in \mathbb{R}^{n \times p}$.
- Least squares criterion: $\min Q(\boldsymbol{b}) = \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{b}\|_2^2$.
- Any solution to the normal equation $\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{b} = \boldsymbol{X}^T\boldsymbol{y}$ (always consistent) is a minimizer of the least squares criterion $Q(\boldsymbol{b})$.
- Solutions to the normal equation:

$$\hat{\boldsymbol{b}} = (\boldsymbol{X}^T\boldsymbol{X})^-\boldsymbol{X}^T\boldsymbol{y} + (\boldsymbol{I}_p - (\boldsymbol{X}^T\boldsymbol{X})^-\boldsymbol{X}^T\boldsymbol{X})\boldsymbol{q},$$

  where $\boldsymbol{q} \in \mathbb{R}^q$ is arbitrary.
- $(\boldsymbol{X}^T\boldsymbol{X})^-\boldsymbol{X}^T$ is a generalized inverse of $\boldsymbol{X}$. Therefore the least squares solution applies even when the system $\boldsymbol{X}\boldsymbol{b} = \boldsymbol{y}$ is consistent.
- Least squares solution is unique if and only if $\boldsymbol{X}$ has full column rank.
- $\boldsymbol{P_X} = \boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^-\boldsymbol{X}^T$ is *the* orthogonal projection onto $\mathcal{C}(\boldsymbol{X})$.
- Geometry: The fitted value from the least squares solution $\hat{\boldsymbol{y}} = \boldsymbol{P_X}\boldsymbol{y}$ is the orthogonal projection of the response vector $\boldsymbol{y}$ onto the column space $\mathcal{C}(\boldsymbol{X})$.



- $\boldsymbol{I}_n - \boldsymbol{P_X}$ is the orthogonal projection onto $\mathcal{N}(\boldsymbol{X}^T)$.

- Decomposition of $\boldsymbol{y}$:

$$\boldsymbol{y} = \boldsymbol{P_X y} + (\boldsymbol{I}_n - \boldsymbol{P_X})\boldsymbol{y} = \hat{\boldsymbol{y}} + \hat{\boldsymbol{e}},$$

  where $\hat{\boldsymbol{y}} \perp \hat{\boldsymbol{e}}$ and

$$\|\boldsymbol{y}\|_2^2 = \|\hat{\boldsymbol{y}}\|_2^2 + \|\hat{\boldsymbol{e}}\|_2^2.$$

Eigenvalues and eigenvectors

KL chapter 8. Assume $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ a square matrix.

- *Eigenvalues* are defined as roots of the characteristic equation $\det(\lambda \boldsymbol{I}_n - \boldsymbol{A}) = 0$.
- If λ is an eigenvalue of $\boldsymbol{A}$, then there exist non-zero $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$ such that $\boldsymbol{Ax} = \lambda \boldsymbol{x}$ and $\boldsymbol{y}^T \boldsymbol{A} = \lambda \boldsymbol{y}^T$. $\boldsymbol{x}$ and $\boldsymbol{y}$ are called the (column) *eigenvector* and *row eigenvector* of $\boldsymbol{A}$ associated with the eigenvalue λ.
- $\boldsymbol{A}$ is singular if and only if it has at least one 0 eigenvalue.
- Eigenvectors associated with distinct eigenvalues are linearly independent.
- Eigenvalues of an upper or lower triangular matrix are its diagonal entries: $\lambda_i = a_{ii}$.
- Eigenvalues of an idempotent matrix are either 0 or 1.
- Eigenvalues of an orthogonal matrix have complex modulus 1.
- In most statistical applications, we deal with eigenvalues/eigenvectors of symmetric matrices.
  The eigenvalues and eigenvectors of a real *symmetric* matrix are real.
- Eigenvectors associated with distinct eigenvalues of a symmetry matrix are orthogonal.
- Eigen-decompostion of a symmetric matrix: $\boldsymbol{A} = \boldsymbol{U \Lambda U}^{\intercal}$, where
  - $\boldsymbol{\Lambda} = \text{diag}(\lambda_1, \ldots, \lambda_n)$
  - columns of $\boldsymbol{U}$ are the eigenvectors which are (or can be chosen to be) mutually orthonormal

- A real symmetric matrix is positive semidefinite (positive definite) if and only if all eigenvalues are nonnegative (positive).
- *Spectral radius* $\rho(\boldsymbol{A}) = \max_i |\lambda_i|$.
- $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ a square matrix (not required to be symmetric), then $\text{tr}(\boldsymbol{A}) = \sum_i \lambda_i$ and $|\boldsymbol{A}| = \prod_i \lambda_i$.

Singular value decomposition

KL chapter 9. Assume $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ and $p = \min\{m, n\}$.

- Singular value decomposition (SVD): $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\intercal$, where
  - $\boldsymbol{U} = (\boldsymbol{u}_1, \ldots, \boldsymbol{u}_m) \in \mathbb{R}^{m \times m}$ is orthogonal
  - $\boldsymbol{V} = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n) \in \mathbb{R}^{n \times n}$ is orthgonal
  - $\boldsymbol{\Sigma} = \text{diag}(\sigma_1, \ldots, \sigma_p) \in \mathbb{R}^{m \times n}$, $\sigma_1 \ge \sigma_2 \ge \cdots \ge \sigma_p \ge 0$.

  σ_i are called the *singular values*, $\boldsymbol{u}_i$ are the *left singular vectors*, and $\boldsymbol{v}_i$ are the *right singular vectors*.
- $\boldsymbol{A}\boldsymbol{v}_i = \sigma_i \boldsymbol{u}_i$ and $\boldsymbol{A}^T \boldsymbol{u}_i = \sigma_i \boldsymbol{v}_i$ for $i = 1, \ldots, p$.
- Thin SVD. Assume $m \ge n$. $\boldsymbol{A}$ can be factored as $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\intercal$, where
  - $\boldsymbol{U} \in \mathbb{R}^{m \times n}$, $\boldsymbol{U}^\intercal \boldsymbol{U} = \boldsymbol{I}_n$
  - $\boldsymbol{V} \in \mathbb{R}^{n \times n}$, $\boldsymbol{V}^\intercal \boldsymbol{V} = \boldsymbol{I}_n$
  - $\boldsymbol{\Sigma} = \text{diag}(\sigma_1, \ldots, \sigma_n)$
- Relation to eigen-decomposition. Using thin SVD,

$$\begin{aligned} \boldsymbol{A}^\intercal \boldsymbol{A} &= \boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{U}^\intercal \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\intercal = \boldsymbol{V}\boldsymbol{\Sigma}^2\boldsymbol{V}^\intercal \\ \boldsymbol{A}\boldsymbol{A}^\intercal &= \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\intercal \boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{U}^\intercal = \boldsymbol{U}\boldsymbol{\Sigma}^2\boldsymbol{U}^\intercal. \end{aligned}$$

- Another relation to eigen-decomposition. Using thin SVD,

$$\begin{pmatrix} \boldsymbol{0}_{n \times n} & \boldsymbol{A}^\intercal \\ \boldsymbol{A} & \boldsymbol{0}_{m \times m} \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} \boldsymbol{V} & \boldsymbol{V} \\ \boldsymbol{U} & -\boldsymbol{U} \end{pmatrix} \begin{pmatrix} \boldsymbol{\Sigma} & \boldsymbol{0}_{n \times n} \\ \boldsymbol{0}_{n \times n} & -\boldsymbol{\Sigma} \end{pmatrix} \frac{1}{\sqrt{2}} \begin{pmatrix} \boldsymbol{V}^\intercal & \boldsymbol{U}^\intercal \\ \boldsymbol{V}^\intercal & -\boldsymbol{U}^\intercal \end{pmatrix}.$$

  Hence any symmetric eigen-solver can produce the SVD of a matrix $\boldsymbol{A}$ without forming $\boldsymbol{A}\boldsymbol{A}^\intercal$ or $\boldsymbol{A}^\intercal \boldsymbol{A}$.

- Yet another relation to eigen-decomposition: If the eigendecomposition of a real symmetric matrix is $\boldsymbol{A} = \boldsymbol{W}\boldsymbol{\Lambda}\boldsymbol{W}^T = \boldsymbol{W}\text{diag}(\lambda_1, \ldots, \lambda_n)\boldsymbol{W}^T$, then

$$\boldsymbol{A} = \boldsymbol{W}\boldsymbol{\Lambda}\boldsymbol{W}^T = \boldsymbol{W}\begin{pmatrix} |\lambda_1| & & \\ & \ddots & \\ & & |\lambda_n| \end{pmatrix}\begin{pmatrix} \text{sgn}(\lambda_1) & & \\ & \ddots & \\ & & \text{sgn}(\lambda_n) \end{pmatrix}\boldsymbol{W}^T$$

  is the SVD of $\boldsymbol{A}$.

- Relation to the Moore-Penrose (MP) inverse: Using thin SVD,

$$\boldsymbol{A}^+ = \boldsymbol{V}\boldsymbol{\Sigma}^+\boldsymbol{U}^\intercal,$$

  where $\boldsymbol{\Sigma}^+ = \text{diag}(\sigma_1^{-1}, \ldots, \sigma_r^{-1}, 0, \ldots, 0)$, $r = \text{rank}(\boldsymbol{A})$.

- Denote $\sigma(\boldsymbol{A}) = (\sigma_1, \ldots, \sigma_p)$. Then
  - $\text{rank}(\boldsymbol{A}) = \#$ nonzero singular values $= \|\sigma(\boldsymbol{A})\|_0$
  - $\boldsymbol{A} = \boldsymbol{U}_r\boldsymbol{\Sigma}_r\boldsymbol{V}_r^T = \sum_{i=1}^r \sigma_i \boldsymbol{u}_i\boldsymbol{v}_i^T$
  - $\|\boldsymbol{A}\|_\text{F} = (\sum_{i=1}^p \sigma_i^2)^{1/2} = \|\sigma(\boldsymbol{A})\|_2$
  - $\|\boldsymbol{A}\|_2 = \sigma_1 = \|\sigma(\boldsymbol{A})\|_\infty$

- Assume $\text{rank}(\boldsymbol{A}) = r$ and partition $\boldsymbol{U} = (\boldsymbol{U}_r, \tilde{\boldsymbol{U}}_r) \in \mathbb{R}^{m\times m}$ and $\boldsymbol{V} = (\boldsymbol{V}_r, \tilde{\boldsymbol{V}}_r) \in \mathbb{R}^{n\times n}$, then
  - $\mathcal{C}(\boldsymbol{A}) = \text{span}\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_r\}$, $\mathcal{N}(\boldsymbol{A}^T) = \text{span}\{\boldsymbol{u}_{r+1}, \ldots, \boldsymbol{u}_m\}$
  - $\mathcal{N}(\boldsymbol{A}) = \text{span}\{\boldsymbol{v}_{r+1}, \ldots, \boldsymbol{v}_n\}$, $\mathcal{C}(\boldsymbol{A}^T) = \text{span}\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_r\}$
  - $\boldsymbol{U}_r\boldsymbol{U}_r^T$ is the orthogonal projection onto $\mathcal{C}(\boldsymbol{A})$
  - $\tilde{\boldsymbol{U}}_r\tilde{\boldsymbol{U}}_r^T$ is the orthogonal projection onto $\mathcal{N}(\boldsymbol{A}^T)$
  - $\boldsymbol{V}_r\boldsymbol{V}_r^T$ is the orthogonal projection onto $\mathcal{C}(\boldsymbol{A}^T)$
  - $\tilde{\boldsymbol{V}}_r\tilde{\boldsymbol{V}}_r^T$ is the orthogonal projection onto $\mathcal{N}(\boldsymbol{A})$

Preliminaries of numerical linear algebra

Numerical linear algebra concerns how matrix/vector computations are done in computer. We first look at some basic linear algebra operations.

Flop counts of some basic linear algebra subroutines (BLAS)

See `http://www.netlib.org/blas/` for a complete listing of BLAS functions.

Level	Operation	Name	Dimension	Flops
1	$\alpha \leftarrow \boldsymbol{x}^T\boldsymbol{y}$	dot product	$\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$	n
	$\boldsymbol{y} \leftarrow \boldsymbol{y} + a\boldsymbol{x}$	saxpy	$a \in \mathbb{R}$, $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$	n
2	$\boldsymbol{y} \leftarrow \boldsymbol{y} + \boldsymbol{A}\boldsymbol{x}$	gaxpy	$\boldsymbol{A} \in \mathbb{R}^{m\times n}$, $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{y} \in \mathbb{R}^m$	mn
	$\boldsymbol{A} \leftarrow \boldsymbol{A} + \boldsymbol{y}\boldsymbol{x}^T$	rank one update	$\boldsymbol{A} \in \mathbb{R}^{m\times n}$, $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{y} \in \mathbb{R}^m$	mn
3	$\boldsymbol{C} \leftarrow \boldsymbol{C} + \boldsymbol{A}\boldsymbol{B}$	matrix multiplication	$\boldsymbol{A} \in \mathbb{R}^{m\times p}$, $\boldsymbol{B} \in \mathbb{R}^{p\times n}$, $\boldsymbol{C} \in \mathbb{R}^{m\times n}$	mnp
	$\boldsymbol{A} \leftarrow \boldsymbol{A}\boldsymbol{D}$	column scaling	$\boldsymbol{A} \in \mathbb{R}^{m\times n}$, $\boldsymbol{D} = \text{diag}(d_1, \ldots, d_n)$	mn
	$\boldsymbol{A} \leftarrow \boldsymbol{D}\boldsymbol{A}$	row scaling	$\boldsymbol{A} \in \mathbb{R}^{m\times n}$, $\boldsymbol{D} = \text{diag}(d_1, \ldots, d_m)$	mn

- Go over the "mxmult" session in R. Also read JM 3.8.
  `http://hua-zhou.github.io/teaching/st758-2014fall/mxmult.html`
  Different ways to compute $\boldsymbol{X}^T\boldsymbol{W}^{-1}\boldsymbol{X}$, such as in the weighted least squares. $\boldsymbol{X} \in \mathbb{R}^{n\times p}$, $\boldsymbol{W} = \text{diag}(w_1, \ldots, w_n) \in \mathbb{R}^{n\times n}$.

  1. `t(X) %*% solve(W) %*% X`: $O(n^3 + n^2p + np^2)$ flops, why need to do the expensive matrix inversion?
  2. `t(X) %*% diag(1 / w) %*% X`: $O(n^2p + np^2)$ flops, why need to save a diagonal matrix and do an extra matrix multiplication?
  3. `(t(X) / w)  %*% X`: wrong! $\boldsymbol{w}$ recycled incorrectly
  4. `t(X) %*% (X / w)`: $O(np^2 + np) = O(np^2)$ flops
  5. `crossprod(X, X / w)`: same as 4, skip the transpose operation

- Another example: Fisher information matrix of a generalized linear model (GLM): $\boldsymbol{X}^T\boldsymbol{W}\boldsymbol{X}$, where $\boldsymbol{X} \in \mathbb{R}^{n\times p}$ and $\boldsymbol{W} = \text{diag}(w_1, \ldots, w_n)$ are the observation weights.

- Bottom line: Always be flop-aware when writing code.

  The form of a mathematical expression and the way the expression should be evaluated in actual practice may be quite different.

- But, for high-performance matrix commutations, it is *not* enough to minimize flops. Pipelining, effective use of memory hierarchy, data layout in memory, ... play important role too.

Vector computer

- Most modern computers are vector machines, which perform vector calculations (saxpy, inner product) fast.
- Vector processing by *pipelining.* E.g., vector addition $\boldsymbol{z} = \boldsymbol{x} + \boldsymbol{y}$

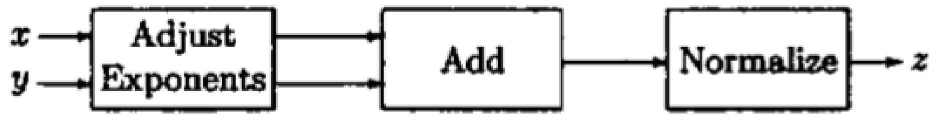


FIG. 1.4.1 *A 3-Cycle Adder*

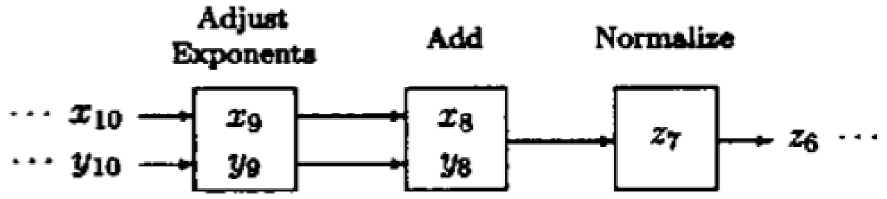
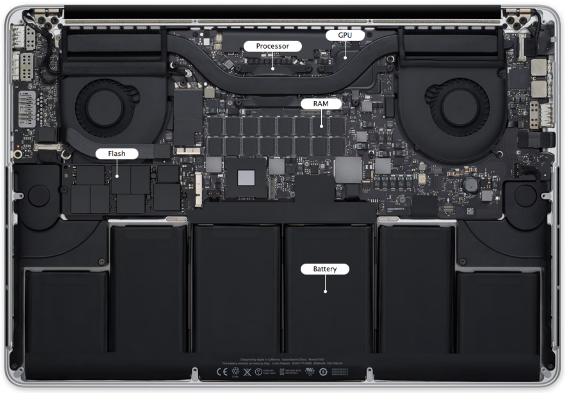
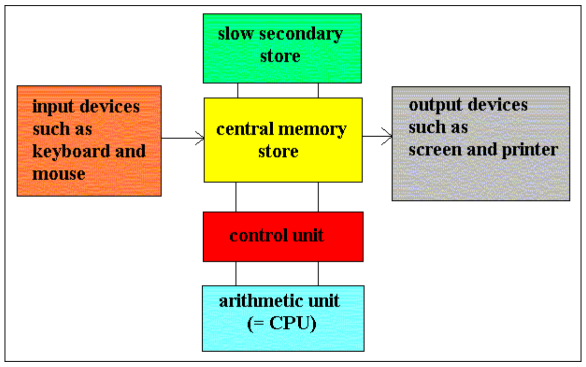


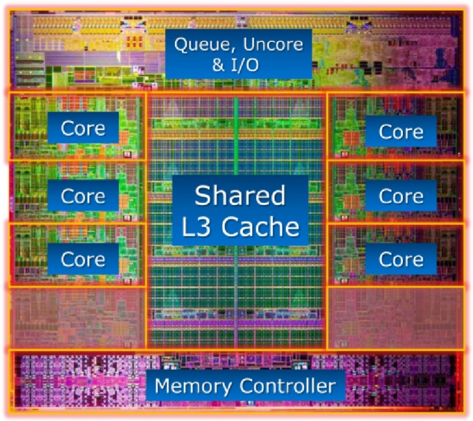
FIG. 1.4.2 *Pipelined Addition*

- One implication of the pipelining technology is that we need to ship vectors to the pipeline fast enough to keep the arithmetic units (ALU) busy and maintain high throughput.

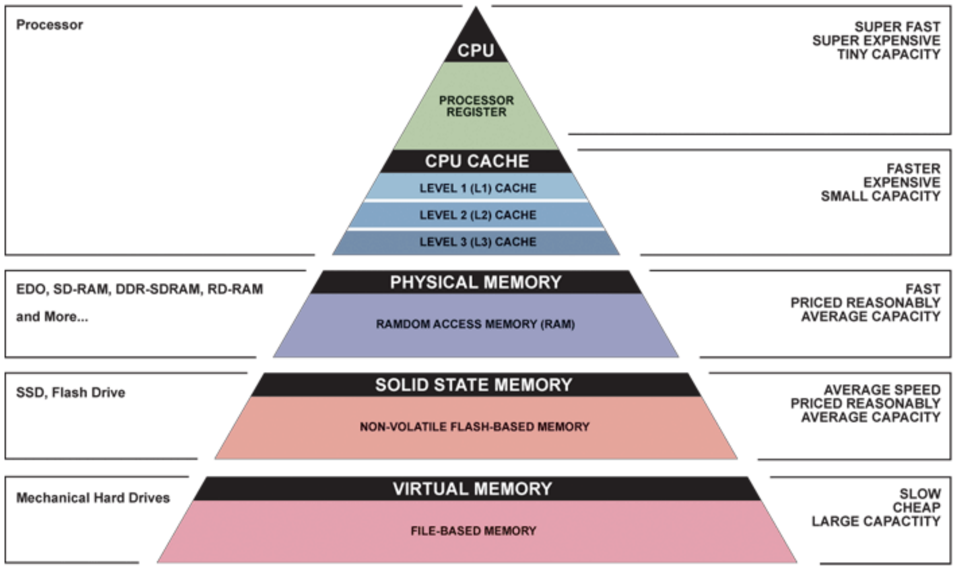
Computer architecture




slow secondary store
input devices such as keyboard and mouse
central memory store
output devices such as screen and printer
control unit
arithmetic unit (= CPU)



- Memory hierarchy:



▲ Simplified Computer Memory Hierarchy
Illustration: Ryan J. Leng

Upper the hierarchy, faster the memory accessing speed, and more expensive the memory units.

> Key to high performance is effective use of memory hierarchy. True on all architectures.

- Can we keep the super fast arithmetic units busy with enough deliveries of matrix data and ship the results to memory fast enough to avoid backlog? Answer: use high-level BLAS as much as possible

- Why high-level BLAS?

BLAS	Dimension	Mem Refs	Flops	Ratio
Level 1: $\boldsymbol{y} \leftarrow \boldsymbol{y} + a\boldsymbol{x}$	$\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$	$3n$	n	3:1
Level 2: $\boldsymbol{y} \leftarrow \boldsymbol{y} + \boldsymbol{A}\boldsymbol{x}$	$\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n, \boldsymbol{A} \in \mathbb{R}^{n \times n}$	n^2	n^2	1:1
Level 3: $\boldsymbol{C} \leftarrow \boldsymbol{C} + \boldsymbol{A}\boldsymbol{B}$	$\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C} \in \mathbb{R}^{n \times n}$	$4n^2$	n^3	4:n

- BLAS 1 tend to be *memory bandwidth-limited*. E.g., Xeon X5650 CPU has a theoretical throughput of 128 DP GFLOPS but a max memory bandwidth of 32GB/s.

- Higher level BLAS (3 or 2) make more effective use of ALUs (keep them busy).

- Message: Although we state many algorithms (solving linear equations, least squares, eigen-decomposition, SVD, ...) in terms of inner product and saxpy, the actual implementation may be quite different.

- A distinction between LAPACK and LINPACK is that LAPACK makes use of higher level BLAS as much as possible (usually by smart partitioning) to increase the so-called *level-3 fraction*.

5 Lecture 5, Sep 11

Announcements

- TA office hours: Fri @ 10A-12P
- HW1 due today. Submit hardcopy and email R or Rmd code
- HW2 posted, due Tue Sep 23

Last time

- Flop counts of some BLAS subroutines
- Memory hierarchy and its implication for implementation of BLAS

Today

- Numerical linear algebra preliminaries: effects of data layout
- Numerical linear algebra: GE, LU

Effect of data layout

- Data layout in memory effects execution speed too. It is much faster to move chunks of data in memory than retrieving/writing scattered data.
- Storage mode: column-major (Fortran, Matlab, R) vs row-major (C/C++).
- Take matrix multiplication as an example. Assume the storage is column-major, such as in Fortran. $\boldsymbol{C} \leftarrow \boldsymbol{C} + \boldsymbol{A}\boldsymbol{B}$, where $\boldsymbol{A} \in \mathbb{R}^{m \times n}$, $\boldsymbol{B} \in \mathbb{R}^{n \times p}$, $\boldsymbol{C} \in \mathbb{R}^{m \times p}$. There are 6 variants of the algorithms according to the order in the triple loops. We pay attention to the innermost loop, where the vector calculation occurs,

```
jki or kji:        for i = 1:m
                       C(i,j) = C(i,j) + A(i,k)B(k,j)
                   end

ikj or kij:        for j = 1:n
                       C(i,j) = C(i,j) + A(i,k)B(k,j)
                   end

ijk or jik:        for k = 1:p
                       C(i,j) = C(i,j) + A(i,k)B(k,j)
                   end
```

and the associated *stride* when accessing the three matrices in memory (assuming column-major storage)

Variant	A Stride	B Stride	C Stride
jki or kji	Unit	0	Unit
ikj or kij	0	Non-Unit	Non-Unit
ijk or jik	Non-Unit	Unit	0

Apparently the variants jki or kji are preferred.

- Message: data storage mode effects algorithm implementation too.

Solving linear equations

We consider algorithms for solving linear equations $\boldsymbol{Ax} = \boldsymbol{b}$, a ubiquitous task in statistics. Idea: turning original problem into an "easy" one, e.g., triangular system.

Triangular system

- *Forward substitution* to solve $\boldsymbol{Lx} = \boldsymbol{b}$, where $\boldsymbol{L} \in \mathbb{R}^{n \times n}$ is lower triangular

$$\begin{bmatrix} a_{11} & 0 & \dots & 0 \\ a_{21} & a_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mm} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$$

$$\begin{aligned} x_1 &= b_1/a_{11} \\ x_2 &= (b_2 - a_{21}x_1)/a_{22} \\ x_3 &= (b_3 - a_{31}x_1 - a_{32}x_2)/a_{33} \\ &\vdots \\ x_m &= b_m - a_{m1}x_1 - a_{m2}x_2 - \dots - a_{m,m-1}x_{m-1})/a_{mm} \end{aligned}$$

$n^2/2$ flops ($n^2/2$ multiplications/divisions and $n^2/2$ additions/substractions) and $\boldsymbol{L}$ is accessed by row.

- *Back substitution* to solve $\boldsymbol{U}\boldsymbol{x} = \boldsymbol{b}$ where $\boldsymbol{U} \in \mathbb{R}^{n \times n}$ is upper triangular

$$\begin{bmatrix} a_{11} & \dots & a_{1,m-1} & a_{1m} \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \dots & a_{m-1,m-1} & a_{m-1,m} \\ 0 & \dots & 0 & a_{mm} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_{m-1} \\ x_m \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_{m-1} \\ b_m \end{bmatrix}$$

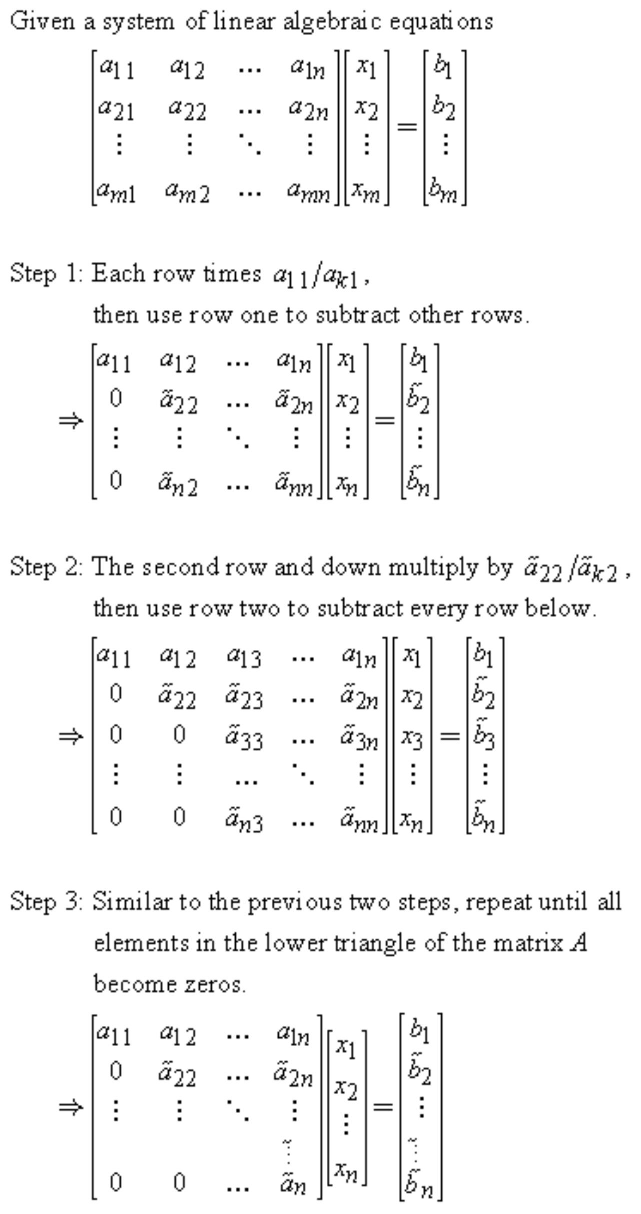
$$\begin{aligned} x_m &= b_m/a_{mm} \\ x_{m-1} &= (b_{m-1} - a_{m-1,m}x_m)/a_{m-1,m-1} \\ x_{m-2} &= (b_{m-2} - a_{m-2,m-1}x_{m-1} - a_{m-2,m}x_m)/a_{m-2,m-2} \\ &\vdots \\ x_1 &= b_1 - a_{12}x_2 - a_{13}x_3 - \dots - a_{1m}x_m)/a_{11} \end{aligned}$$

$n^2/2$ flops ($n^2/2$ multiplications/divisions and $n^2/2$ additions/substractions) and $\boldsymbol{U}$ is accessed by row.

- Column version: reverse the order of looping.
- BLAS level 2 function: `?trsv` (triangular solve with one right hand side)
- BLAS level 3 function: `?trsm` (matrix triangular solve, i.e., multiple right hand sides)

- In R, `forwardsolve()` and `backsolve()` (wrappers of `dtrsm`)
- Eigenvalues of $\boldsymbol{L}$ are diagonal entries $\lambda_i = \ell_{ii}$. $\det(\boldsymbol{L}) = \prod_i \ell_{ii}$.
- A *unit triangular matrix* is a triangular matrix with all diagonal entries being 1.
- The algebra of triangular matrices (HW2)
  - The product of two upper (lower) triangular matrices is upper (lower) triangular.
  - The inverse of an upper (lower) triangular matrix is upper (lower) triangular.
  - The product of two unit upper (lower) triangular matrices is unit upper (lower) triangular.
  - The inverse of a unit upper (lower) triangular matrix is unit upper (lower) triangular.

Gaussian elimination and LU decomposition



Given a system of linear algebraic equations

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$$

Step 1: Each row times a_{11}/a_{k1},
then use row one to subtract other rows.

$$\Rightarrow \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ 0 & \tilde{a}_{22} & \cdots & \tilde{a}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \tilde{a}_{n2} & \cdots & \tilde{a}_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ \tilde{b}_2 \\ \vdots \\ \tilde{b}_n \end{bmatrix}$$

Step 2: The second row and down multiply by $\tilde{a}_{22}/\tilde{a}_{k2}$,
then use row two to subtract every row below.

$$\Rightarrow \begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ 0 & \tilde{a}_{22} & \tilde{a}_{23} & \cdots & \tilde{a}_{2n} \\ 0 & 0 & \tilde{a}_{33} & \cdots & \tilde{a}_{3n} \\ \vdots & \vdots & \cdots & \ddots & \vdots \\ 0 & 0 & \tilde{a}_{n3} & \cdots & \tilde{a}_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ \tilde{b}_2 \\ \tilde{b}_3 \\ \vdots \\ \tilde{b}_n \end{bmatrix}$$

Step 3: Similar to the previous two steps, repeat until all
elements in the lower triangle of the matrix A
become zeros.

$$\Rightarrow \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ 0 & \tilde{a}_{22} & \cdots & \tilde{a}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ & & & \tilde{\vdots} \\ 0 & 0 & \cdots & \tilde{a}_{n} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ \tilde{b}_2 \\ \vdots \\ \tilde{\vdots} \\ \tilde{b}_n \end{bmatrix}$$

- Solve $\boldsymbol{Ax} = \boldsymbol{b}$ for a general matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$.
- An important contribution by Gauss. No linear algebra in 1800!
- Idea: a series of *elementary operations* that turn $\boldsymbol{A}$ into a triangular system.
- We consider the square $\boldsymbol{A}$ case first.

- *Elementary operator matrix* $\boldsymbol{E}_{jk}(c)$ is the identity with the 0 in position (j,k) replaced by c. For any vector $\boldsymbol{x}$,

$$\boldsymbol{E}_{jk}(c)\boldsymbol{x} = (x_1, \ldots, x_{j-1}, x_j + cx_k, x_{j+1}, \ldots, x_n)^\intercal.$$

  Applying $\boldsymbol{E}_{jk}(c)$ to both sides of the system $\boldsymbol{Ax} = \boldsymbol{b}$ replaces the j-th equation $\boldsymbol{a}_{j*}^\intercal \boldsymbol{x} = b_j$ by $\boldsymbol{a}_{j*}^\intercal \boldsymbol{x} + c\boldsymbol{a}_{k*}^\intercal \boldsymbol{x} = b_j + cb_k$. For $j > k$, $\boldsymbol{E}_{jk}(c) = \boldsymbol{I} + c\boldsymbol{e}_j\boldsymbol{e}_k^\intercal$ is unit lower triangular and full rank. $\boldsymbol{E}_{jk}^{-1}(c) = \boldsymbol{E}_{jk}(-c)$.

- Zeroing the first column

$$\begin{aligned}
\boldsymbol{E}_{21}(c_2^{(1)})\boldsymbol{Ax} &= \boldsymbol{E}_{21}(c_2^{(1)})\boldsymbol{b} \\
\boldsymbol{E}_{31}(c_3^{(1)})\boldsymbol{E}_{21}(c_2^{(1)})\boldsymbol{Ax} &= \boldsymbol{E}_{31}(c_3^{(1)})\boldsymbol{E}_{21}(c_2^{(1)})\boldsymbol{b} \\
&\vdots \\
\boldsymbol{E}_{n1}(c_n^{(1)})\cdots\boldsymbol{E}_{31}(c_3^{(1)})\boldsymbol{E}_{21}(c_2^{(1)})\boldsymbol{Ax} &= \boldsymbol{E}_{n1}(c_n^{(1)})\cdots\boldsymbol{E}_{31}(c_3^{(1)})\boldsymbol{E}_{21}(c_2^{(1)})\boldsymbol{b}
\end{aligned}$$

  where $c_i^{(1)} = -a_{i1}/a_{11}$. Denote $\boldsymbol{M}_1 = \boldsymbol{E}_{n1}(c_n^{(1)})\cdots\boldsymbol{E}_{31}(c_3^{(1)})\boldsymbol{E}_{21}(c_2^{(1)})$.

- Then zero the k-th column for $k = 2, \ldots, n-1$ sequentially. This results in a transformed linear system $\boldsymbol{Ux} = \tilde{\boldsymbol{b}}$, where $\boldsymbol{U} = \boldsymbol{M}_{n-1}\cdots\boldsymbol{M}_1\boldsymbol{A}$ is upper triangular and $\tilde{\boldsymbol{b}} = \boldsymbol{M}_{n-1}\cdots\boldsymbol{M}_1\boldsymbol{b}$. $\boldsymbol{M}_k$ has the shape

$$\boldsymbol{M}_k = \boldsymbol{E}_{n,k}^{(k)}\cdots\boldsymbol{E}_{k+1,k}^{(k)} = \begin{pmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & 1 & & & \\ & & c_{k+1}^{(k)} & 1 & & \\ & & \vdots & & \ddots & \\ & & c_n^{(k)} & & & 1 \end{pmatrix},$$

  where $c_i^{(k)} = -\tilde{a}_{ik}^{(k-1)}/\tilde{a}_{kk}^{(k-1)}$. $\boldsymbol{M}_k$ is unit lower triangular and full rank. $\boldsymbol{M}_k$ are called the *Gauss transformations*.

- Let $\boldsymbol{L} = \boldsymbol{M}_1^{-1}\cdots\boldsymbol{M}_{n-1}^{-1}$. We have the decomposition

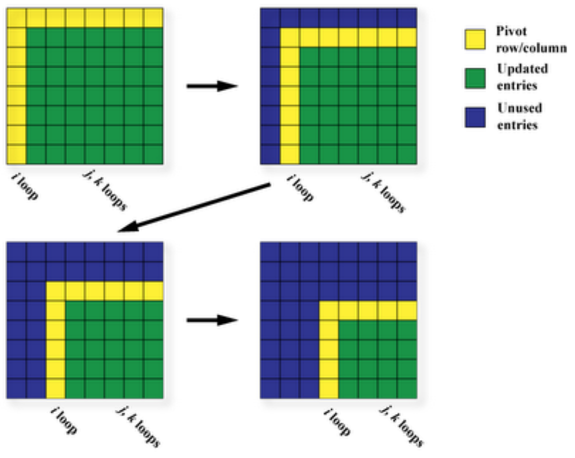
$$\boldsymbol{A} = \boldsymbol{LU}.$$

  $\boldsymbol{M}_k$ is unit upper triangular, so $\boldsymbol{M}_k^{-1}$ and thus $\boldsymbol{L}$ is unit lower triangular.

- Where is $\boldsymbol{L}$? Note $\boldsymbol{M}_k = \boldsymbol{I} + (0, \ldots, 0, c_{k+1}^{(k)}, \ldots, c_n^{(k)})^\mathsf{T} \boldsymbol{e}_k^\mathsf{T}$. By Sherman-Morrison, $\boldsymbol{M}_k^{-1} = \boldsymbol{I} - (0, \ldots, 0, c_{k+1}^{(k)}, \ldots, c_n^{(k)})^\mathsf{T} \boldsymbol{e}_k^\mathsf{T}$. So the entries of $\boldsymbol{L}$ are simply $\ell_{ik} = -c_i^{(k)}$, $i > k$, the negative of multipliers in GE.

- The whole LU procedure is done in place, i.e., $\boldsymbol{A}$ is overwritten by $\boldsymbol{L}$ and $\boldsymbol{U}$.

- Implementation: outer product LU (kij loop), block outer product LU (higher level-3 fraction), Crout's algorithm (jki loop), ...



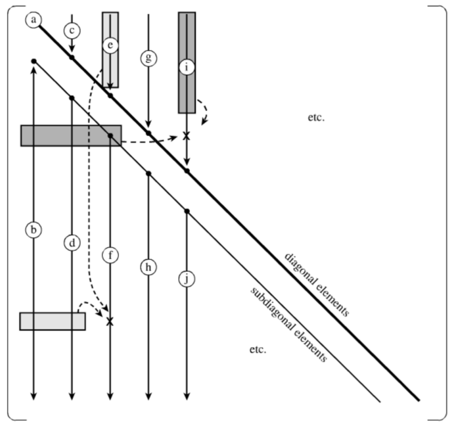


Figure 2.3.1. Crout's algorithm for *LU* decomposition of a matrix. Elements of the original matrix are modified in the order indicated by lower case letters: a, b, c, etc. Shaded boxes show the previously modified elements that are used in modifying two typical elements, each indicated by an "x".

- LU decomposition exists if the principal sub-matrix $A(1:k, 1:k)$ is non-singular for $k = 1, \ldots, n-1$. If the LU decomposition exists and $\boldsymbol{A}$ is non-singular, then the LU decomposition is unique and $\det(\boldsymbol{A}) = \prod_{i=1}^n u_{ii}$.

- This *forward elimination* or *LU decomposition* costs $(n-1)^2 + (n-2)^2 + \cdots + 1^2 \approx \frac{1}{3}n^3$ flops ($n^3/3$ multiplications and $n^3/3$ additions).

- Given LU, one right hand side costs n^2 flops (one backward substitution and then forward substitution).

- For inversion, there are n right hand sides $\boldsymbol{e}_i$. However, taking advantage of zeros reduces n^3 flops to $\frac{2}{3}n^3$ (see JM exercise 3.2). So matrix inversion costs $\frac{1}{3}n^3 + \frac{2}{3}n^3 = n^3$ flops in total.

- We do *not* compute matrix inverse unless (i) it is absolutely necessary to compute s.e., (2) number of right hand sides is much larger than n, (3) n is small.

- LU decomposition of a rectangular matrix $\boldsymbol{A} \in \mathbf{R}^{m\times n}$ exists if $\boldsymbol{A}(1:k, 1:k)$ is non-singular for $k = \min\{m, n\}$. For example,

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 3 & 1 \\ 5 & 2 \end{pmatrix} \begin{pmatrix} 1 & 2 \\ 0 & -2 \end{pmatrix}$$

and

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 4 & 1 \end{pmatrix} \begin{pmatrix} 1 & 2 & 3 \\ 0 & -3 & -6 \end{pmatrix}.$$

Slight modification to the algorithm.

6 Lecture 6, Sep 16

Announcements

- HW1 returned:
  - Change of code naming convention for later homework: `LastFirstHW2.R` or `LastFirstHW2.Rmd`, e.g., `ZhouHuaHW2.Rmd`
  - Identify yourself in your code
  - Submit code on time: late penalty 5pts/day (?)
  - Style: indenting, no more 80 characters per line, space around binary operators, space after comma, ...
  - Q1: work out the computer arithmetic on at least one case, so you're clear about what's happening with rounding, losing significant digits when matching exponents, normalization, ...
  - Q4: `crossprod()`, `tcrossprod()`, `outer()`, `row()`, `col()`, `rowSums()`, `colSums()` functions in R, subsetting in R, ...
  - Q5: vectorize code, determinant from Cholesky decomposition
  - Sketch of solution: `http://hua-zhou.github.io/teaching/st758-2014fall/hw01sol.html`

Last time

- Effect of data layout (column-major vs row-major)
- Triangular system: $n^2/2$ flops for forward substitution or backward substitution
- GE and LU decomposition (JM 3.4)

Today

- GE and LU: pivoting
- Cholesky decomposition (KL 7.7 and JM 3.5)

Pivoting for LU

- What if we encounter a *pivot* $\tilde{a}_{kk}^{(k-1)}$ being 0 or close to 0 due to underflow?
- Think about $\boldsymbol{A} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$. Does it have a solution for arbitrary $\boldsymbol{b}$? Does GE work?
- Work on the example

$$\begin{aligned} 0.0001x_1 + x_2 &= 1 \\ x_1 + x_2 &= 2, \end{aligned}$$

  which has solution $x_1 = 1.0001$ and $x_2 = 0.9999$. Suppose we have 3 digits of precision. After first step of elimination, we have (catastrophic cancellation happens)

$$\begin{aligned} 0.0001x_1 + x_2 &= 1 \\ -10,000x_2 &= -10,000 \end{aligned}$$

  and the solution by back substitution is $x_2 = 1.000$ and $x_1 = 0.000$.

- Message: zero or very small pivots cause trouble.
  Solution: *pivoting*.
- *Partial pivoting*: at the k-th stage the equation with $\max_{i=k}^{n} |\tilde{a}_{ik}^{(k-1)}|$ is moved into the k-th row. Thus we have $\boldsymbol{M}_{n-1}\boldsymbol{P}_{n-1}\cdots\boldsymbol{M}_1\boldsymbol{P}_1\boldsymbol{A} = \boldsymbol{U}$.
- With partial pivoting, it can be shown that

$$\boldsymbol{PA} = \boldsymbol{LU},$$

  where $\boldsymbol{P} = \boldsymbol{P}_{n-1}\cdots\boldsymbol{P}_1$, $\boldsymbol{L}$ is unit lower triangular with $|\ell_{ij}| \le 1$, and $\boldsymbol{U}$ is upper triangular.

- $\det(\boldsymbol{P})\det(\boldsymbol{A}) = \det(\boldsymbol{U}) = \prod_{i=1}^{n} u_{ii}$.
- To solve $\boldsymbol{Ax} = \boldsymbol{b}$, we solve two triangular systems

$$\boldsymbol{Ly} = \boldsymbol{Pb} \quad \text{and} \quad \boldsymbol{Ux} = \boldsymbol{y},$$

  costing n^2 flops.

- *Complete pivoting*: Do both row and column interchanges so that the largest entry in the sub matrix $\boldsymbol{A}(k:n,k:n)$ is permuted to the (k,k)-th entry. This yields the decomposition $\boldsymbol{PAQ} = \boldsymbol{LU}$, where $|\ell_{ij}| \le 1$.

- Warning: In actual implementation, we do not really need to interchange rows/columns for pivoting. Just keep track of the indices we have interchanged.

- Gaussian elimination with partial pivoting is one of the most commonly used methods for solving general linear systems. Complete pivoting is stable but costs more computation. Partial pivoting is stable most of times.

- LAPACK: `?GETRF` does $\boldsymbol{PA} = \boldsymbol{LU}$ (LU decomposition with partial pivoting).

- In R, `solve()` implicitly performs LU decomposition (wrapper of LAPACK routine `DGESV`). `solve()` allows specifying a single or multiple right hand sides. If none, it computes the matrix inverse. The **`matrix`** package contains `lu()` function.

Cholesky decomposition (symmetric LU)

JM 3.5 and KL 7.7

Archives de l'Ecole polytechnique (Fonds A. Cholesky)

- A basic tenet in numerical analysis:

  The structure should be exploited whenever solving a problem.

  Common structures include: symmetry, definiteness, sparsity, Kronecker product, low rank, ...

- Consider solving the normal equation $\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{\beta} = \boldsymbol{X}^T\boldsymbol{y}$ for linear regression. The coefficient matrix $\boldsymbol{X}^T\boldsymbol{X}$ is symmetric and positive semidefinite, how to exploit this structure?

- Theorem: Let $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ be symmetric and positive definite. Then $\boldsymbol{A} = \boldsymbol{L}\boldsymbol{L}^\intercal$ where $\boldsymbol{L}$ is lower triangular with positive diagonal entries and is unique.

  Proof by induction. If $n = 1$, then $\ell = \sqrt{a}$. For $n > 1$, the block equation

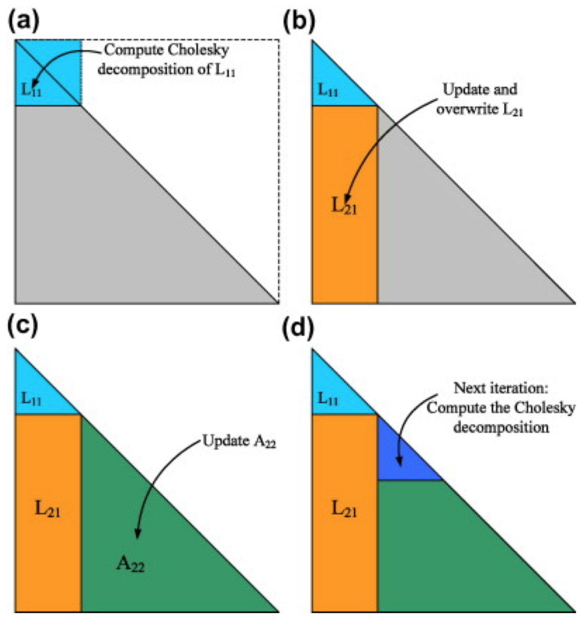
$$\begin{pmatrix} a_{11} & \boldsymbol{a}^\intercal \\ \boldsymbol{a} & \boldsymbol{A}_{22} \end{pmatrix} = \begin{pmatrix} \ell_{11} & \boldsymbol{0}_{n-1}^\intercal \\ \boldsymbol{l} & \boldsymbol{L}_{22} \end{pmatrix} \begin{pmatrix} \ell_{11} & \boldsymbol{l}^\intercal \\ \boldsymbol{0}_{n-1} & \boldsymbol{L}_{22}^\intercal \end{pmatrix}.$$

  has solution

$$\begin{aligned} \ell_{11} &= \sqrt{a_{11}} \\ \boldsymbol{l} &= \ell_{11}^{-1}\boldsymbol{a} \\ \boldsymbol{L}_{22}\boldsymbol{L}_{22}^\intercal &= \boldsymbol{A}_{22} - \boldsymbol{l}\boldsymbol{l}^\intercal = \boldsymbol{A}_{22} - a_{11}^{-1}\boldsymbol{a}\boldsymbol{a}^\intercal. \end{aligned}$$

  Now $a_{11} > 0$ (why?), so ℓ_{11} and $\boldsymbol{l}$ are uniquely determined. $\boldsymbol{A}_{22} - a_{11}^{-1}\boldsymbol{a}\boldsymbol{a}^\intercal$ is positive definite because $\boldsymbol{A}$ is positive definite (why?). By induction hypothesis, $\boldsymbol{L}_{22}$ exists and is unique. □

- The constructive proof completely specifies the algorithm.


(a)
Compute Cholesky decomposition of L_{11}
L_{11}
(b)
L_{11}
Update and overwrite L_{21}
L_{21}
(c)
L_{11}
L_{21}
Update A_{22}
A_{22}
(d)
L_{11}
Next iteration: Compute the Cholesky decomposition
L_{21}

- Computational cost: $\frac{1}{2}[(n-1)^2 + (n-2)^2 + \cdots + 1^2] \approx \frac{1}{6}n^3$ (half the cost of LU decomposition due to symmetry) plus n square roots.

- Avoid square roots: $\boldsymbol{LDL}^{\mathsf{T}}$ decomposition.

- Pivoting? In general Cholesky decomposition is very stable. Failure of the decomposition simply means $\boldsymbol{A}$ is not positive definite. It is an efficient way to test positive definiteness.

  If zero pivots $\tilde{a}_{ii} = 0$ are encountered, we can still continue the algorithm by setting $\ell_{ii} = 0$ and $\boldsymbol{l} = \mathbf{0}$. A better alternative is to use Cholesky decomposition with symmetric pivoting.

- Any matrix $\boldsymbol{X} \in \mathbb{R}^{n \times n}$ is a *square root* of $\boldsymbol{A} \succeq \mathbf{0}_{n \times n}$ if $\boldsymbol{A} = \boldsymbol{XX}$. Note that Cholesky factor is *not* a square root of $\boldsymbol{A}$ unless $\boldsymbol{A}$ is diagonal.

7 Lecture 7, Sep 18

Announcement

- HW2 hints: `http://hua-zhou.github.io/teaching/st758-2014fall/st758fall2014/2014/09/16/hw2-hints.html`
- HW3 will be posted today. Due Tue Sep 30.

Last time

- LU with partial pivoting: $\boldsymbol{PA} = \boldsymbol{LU}$ (still $n^3/3$ flops)
- Cholesky decomposition $\boldsymbol{A} = \boldsymbol{LL}^T$, $\boldsymbol{A}$ p.d.: $n^3/6$ flops taking advantage of symmetry

Today

- Cholesky decomposition for $\boldsymbol{A}$ p.s.d.: symmetric pivoting
- Linear regression by Cholesky (JM 3.5, KL 7.7)
- QR decomposition (JM 5.4-5.8, KL 7.8-7.9)

Cholesky with symmetric pivoting

$\boldsymbol{A} \succeq \mathbf{0}_{n \times n}$ (p.s.d.)

- When $\boldsymbol{A}$ does not have full rank, e.g., $\boldsymbol{X}^T\boldsymbol{X}$ with a non-full column rank $\boldsymbol{X}$, we encounter $\tilde{a}_{kk} = 0$ during the procedure.
- *Symmetric pivoting.* At each stage k, we permute both row and column such that $\max_{i=k}^n \tilde{a}_{kk}$ becomes the pivot. If we encounter $\max_{i=k}^n \tilde{a}_{kk} = 0$, then $A(k:n, k:n) = \mathbf{0}$ (why?) and the algorithm terminates.
- With symmetric pivoting: $\boldsymbol{PAP}^T = \boldsymbol{LL}^T$, where $\boldsymbol{P}$ is a permutation matrix and $\boldsymbol{L} \in \mathbb{R}^{n \times r}$, $r = \text{rank}(\boldsymbol{A})$.
- In R, `chol()` is a wrapper function for LAPACK routines `DPOTRF` (p.d.) and `DPSTRF` (p.s.d. with pivoting).

- Option `pivot = FALSE` calls `DPOTRF`. It does $\boldsymbol{A} = \boldsymbol{R}^T\boldsymbol{R}$ and gives error message if $\boldsymbol{A}$ is not full rank.
- Option `pivot = TRUE` calls `DPSTRF`. It does symmetric pivoting $\boldsymbol{P}\boldsymbol{A}\boldsymbol{P}^T = \boldsymbol{R}^T\boldsymbol{R}$ and yields `rank` and `pivot`.
- Option `tol` passes the tolerance to LAPACK for deciding zero pivots. Default is $n \cdot$ machine epsilon $\cdot \max(\text{diag}(\boldsymbol{A}))$.

Applications of Cholesky decomposition

There are numerous applications of Cholesky decomposition.

- *No inversion* mentality: Whenever we see matrix inverse, we should think in terms of solving linear equations. If the matrix is positive semidefinite, Cholesky decomposition applies.
- Example: multivariate normal density $N_n(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, $\boldsymbol{\Sigma}$ is p.d.

$$-\frac{n}{2}\ln(2\pi) - \frac{1}{2}\ln\det\boldsymbol{\Sigma} - \frac{1}{2}(\boldsymbol{y}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{y}-\boldsymbol{\mu}).$$

  - Method 1: (a) compute explicit inverse $\boldsymbol{\Sigma}^{-1}$ (n^3 flops), (b) compute quadratic form ($n^2 + n$ flops), (c) compute determinant ($n^3/3$ flops).
  - Method 2: (a) Cholesky decomposition $\boldsymbol{\Sigma} = \boldsymbol{L}\boldsymbol{L}^T$ ($n^3/6$ flops), (b) Solve $\boldsymbol{L}\boldsymbol{x} = \boldsymbol{y} - \boldsymbol{\mu}$ by forward substitutions ($n^2/2$ flops), (c) compute quadratic form $\boldsymbol{x}^T\boldsymbol{x}$ (n flops), and (d) compute determinant from Cholesky factor (n flops).

  Which method is better?
- Compute Moore-Penrose inverse $\boldsymbol{A}^+$. (HW3)
- Linear regression.

Linear regression by Cholesky (method of normal equations)

Assume $\boldsymbol{X} \in \mathbb{R}^{n\times p}$ has full column rank. (For rank deficient $\boldsymbol{X}$, use Cholesky with symmetric pivoting.)

- It is easier to work on the augmented matrix

$$\begin{pmatrix} \boldsymbol{X}^\intercal\boldsymbol{X} & \boldsymbol{X}^\intercal\boldsymbol{y} \\ \boldsymbol{y}^\intercal\boldsymbol{X} & \boldsymbol{y}^\intercal\boldsymbol{y} \end{pmatrix} = \begin{pmatrix} \boldsymbol{L} & \boldsymbol{0} \\ \boldsymbol{l}^\intercal & d \end{pmatrix} \begin{pmatrix} \boldsymbol{L}^\intercal & \boldsymbol{l} \\ \boldsymbol{0}^\intercal & d \end{pmatrix} = \begin{pmatrix} \boldsymbol{L}\boldsymbol{L}^\intercal & \boldsymbol{L}\boldsymbol{l} \\ \boldsymbol{l}^\intercal\boldsymbol{L}^\intercal & \|\boldsymbol{l}\|_2^2 + d^2 \end{pmatrix}.$$

  Normal equation implies the equation

$$\boldsymbol{X}^\intercal\boldsymbol{X}\boldsymbol{\beta} = \boldsymbol{L}\boldsymbol{L}^\intercal\boldsymbol{\beta} = \boldsymbol{X}^\intercal\boldsymbol{y} = \boldsymbol{L}\boldsymbol{l} \text{ or } \boldsymbol{L}^\intercal\boldsymbol{\beta} = \boldsymbol{l},$$

  which we can solve for $\boldsymbol{\beta}$ in $p^2/2$ flops. Since $\boldsymbol{l} = \boldsymbol{L}^{-1}\boldsymbol{X}^\intercal\boldsymbol{y}$, we have

$$\boldsymbol{l}^\intercal\boldsymbol{l} = \boldsymbol{y}\boldsymbol{X}(\boldsymbol{L}\boldsymbol{L}^\intercal)^{-1}\boldsymbol{X}^\intercal\boldsymbol{y} = \boldsymbol{y}^\intercal\boldsymbol{X}(\boldsymbol{X}^\intercal\boldsymbol{X})^{-1}\boldsymbol{X}^\intercal\boldsymbol{y} = \boldsymbol{y}^\intercal\boldsymbol{P}_{\boldsymbol{X}}\boldsymbol{y} = \|\hat{\boldsymbol{y}}\|_2^2$$

  and

$$d^2 = \boldsymbol{y}^\intercal\boldsymbol{y} - \boldsymbol{l}^\intercal\boldsymbol{l} = \boldsymbol{y}^\intercal(\boldsymbol{I} - \boldsymbol{P}_{\boldsymbol{X}})\boldsymbol{y} = \|\boldsymbol{y} - \hat{\boldsymbol{y}}\|_2^2 = \text{SSE}.$$

  If standard errors are needed, we do inversion $(\boldsymbol{X}^T\boldsymbol{X})^{-1} = (\boldsymbol{L}\boldsymbol{L}^T)^{-1} = \boldsymbol{L}^{-T}\boldsymbol{L}^{-1}$. Use `chol2inv()` in R function for this purpose.

- In summary, linear regression by Cholesky, aka the method of normal equations:
  - Form the lower triangular part of $(\boldsymbol{X}, \boldsymbol{y})^T(\boldsymbol{X}, \boldsymbol{y})$ ($n(p+1)^2/2$ flops)
  - Cholesky decomposition of the augmented system $\begin{pmatrix} \boldsymbol{X}^\intercal\boldsymbol{X} & \boldsymbol{X}^\intercal\boldsymbol{y} \\ \boldsymbol{y}^\intercal\boldsymbol{X} & \boldsymbol{y}^\intercal\boldsymbol{y} \end{pmatrix}$ ($(p+1)^3/6$ flops)
  - Solve $\boldsymbol{L}^T\boldsymbol{\beta} = \boldsymbol{l}$ for regression coefficients $\hat{\boldsymbol{\beta}}$ ($p^2/2$ flops)
  - If want standard errors, estimate σ^2 by $\hat{\sigma}^2 = d^2/(n-p)$ and compute $\hat{\sigma}^2(\boldsymbol{X}^T\boldsymbol{X})^{-1} = \hat{\sigma}^2(\boldsymbol{L}\boldsymbol{L}^T)^{-1}$ ($2p^3/3$ flops)

  Total cost is $p^3/6 + np^2/2$ flops (without s.e.) or $5p^3/6 + np^2/2$ flops (with s.e.).

QR decomposition and linear regression

Assume $\boldsymbol{X} \in \mathbb{R}^{n\times p}$ has full column rank.

- QR decomposition: $\boldsymbol{X} = \boldsymbol{Q}\boldsymbol{R}$, where $\boldsymbol{Q} \in \mathbb{R}^{n\times n}$, $\boldsymbol{Q}^\intercal\boldsymbol{Q} = \boldsymbol{I}_n$, and $\boldsymbol{R} \in \mathbb{R}^{n\times p}$.
  - first p columns of $\boldsymbol{Q}$ form an orthonormal basis of $\mathcal{C}(\boldsymbol{X})$
  - last $n-p$ columns of $\boldsymbol{Q}$ form an orthonormal basis of $\mathcal{N}(\boldsymbol{X}^T)$

- (Thin/Skinny QR) Then $\boldsymbol{X} = \boldsymbol{Q}_1 \boldsymbol{R}_1$ where $\boldsymbol{Q}_1 \in \mathbb{R}^{n \times p}$ has orthogonal columns, $\boldsymbol{Q}_1^\intercal \boldsymbol{Q}_1 = \boldsymbol{I}_p$, and $\boldsymbol{R}_1 \in \mathbb{R}^{p \times p}$ is an invertible upper triangular matrix with positive diagonal entries.

- For linear regression, we only need skinny QR.
  Note $\boldsymbol{X}^\intercal \boldsymbol{X} = \boldsymbol{R}_1^\intercal \boldsymbol{R}_1$ yields the Cholesky decomposition of $\boldsymbol{X}^\intercal \boldsymbol{X}$.

- Better to perform (skinny) QR on the augmented matrix
$$\begin{pmatrix} \boldsymbol{X} & \boldsymbol{y} \end{pmatrix} = \begin{pmatrix} \boldsymbol{Q} & \boldsymbol{q} \end{pmatrix} \begin{pmatrix} \boldsymbol{R} & \boldsymbol{r} \\ \boldsymbol{0}_p^\intercal & d \end{pmatrix} = \begin{pmatrix} \boldsymbol{QR} & \boldsymbol{Qr} + d\boldsymbol{q} \end{pmatrix}.$$
  Normal equation $\boldsymbol{X}^\intercal \boldsymbol{X} \boldsymbol{\beta} = \boldsymbol{X}^\intercal \boldsymbol{y}$ implies
$$\boldsymbol{R}\boldsymbol{\beta} = \boldsymbol{R}^{-T} \boldsymbol{X}^\intercal \boldsymbol{y} = \boldsymbol{R}^{-T} \boldsymbol{R}^\intercal \boldsymbol{Q}^\intercal \boldsymbol{y} = \boldsymbol{Q}^\intercal \boldsymbol{y} = \boldsymbol{r},$$
  which is easy to solve for $\boldsymbol{\beta}$. The fitted value is $\hat{\boldsymbol{y}} = \boldsymbol{X}\hat{\boldsymbol{\beta}} = \boldsymbol{Q}\boldsymbol{R}\boldsymbol{R}^{-1}\boldsymbol{r} = \boldsymbol{Qr}$. The residual is
$$\hat{\boldsymbol{e}} = \boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{\beta}} = \boldsymbol{y} - \boldsymbol{Qr} = d\boldsymbol{q}$$
  and SSE $= \|\hat{\boldsymbol{e}}\|_2^2 = d^2$. The projection matrix is
$$\boldsymbol{X}(\boldsymbol{X}^\intercal \boldsymbol{X})^{-1} \boldsymbol{X} = \boldsymbol{QR}(\boldsymbol{R}^\intercal \boldsymbol{R})^{-1} \boldsymbol{R}^\intercal \boldsymbol{Q}^\intercal = \boldsymbol{QQ}^\intercal.$$

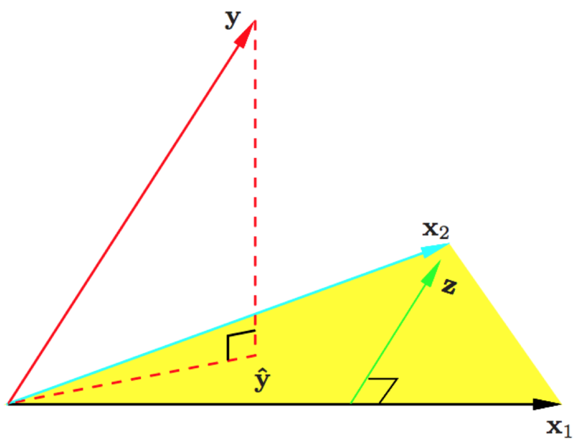
- Three numerical methods to compute QR: (modified) Gram-Schmidt, Householder transform, (fast) Givens transform

QR by (modified) Gram-Schmidt

Jørgen Pedersen Gram in an undated photo

Erhard Schmidt

Erhard Schmidt (courtesy MFO)



Assume $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_p) \in \mathbb{R}^p$ has full column rank.

- Gram-Schmidt algorithm produces the skinny QR.
- Gram-Schmidt algorithm orthonormalizes a set of non-zero, *linearly independent* vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_p$. Initialize $\boldsymbol{q}_1 = \boldsymbol{x}_1 / \|\boldsymbol{x}_1\|_2$; then for $k = 2, \ldots, p$,

$$\begin{aligned} \boldsymbol{v}_k &= \boldsymbol{x}_k - \boldsymbol{P}_{\mathcal{C}\{\boldsymbol{q}_1,\ldots,\boldsymbol{q}_{k-1}\}} \boldsymbol{x}_k = \boldsymbol{x}_k - \sum_{j=1}^{k-1} \langle \boldsymbol{x}_k, \boldsymbol{q}_j \rangle \boldsymbol{q}_j \\ \boldsymbol{q}_k &= \boldsymbol{v}_k / \|\boldsymbol{v}_k\|_2 \end{aligned}$$

- For $j = 1, \ldots, p$, $\mathcal{C}\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_j\} = \mathcal{C}\{\boldsymbol{q}_1, \ldots, \boldsymbol{q}_j\}$ and $\boldsymbol{q}_j \perp \mathcal{C}\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{j-1}\}$.
- Collectively, we have $\boldsymbol{X} = \boldsymbol{Q}\boldsymbol{R}$ (skinny QR), where
  - $\boldsymbol{Q} \in \mathbb{R}^{n \times p}$ has orthonormal columns $\boldsymbol{q}_k$ and thus $\boldsymbol{Q}^T\boldsymbol{Q} = \boldsymbol{I}_p$.
  - What's $\boldsymbol{R}$? $\boldsymbol{R} = \boldsymbol{Q}^T\boldsymbol{X} \in \mathbb{R}^{p \times p}$ has entries $r_{jk} = \langle \boldsymbol{q}_j, \boldsymbol{x}_k \rangle$, which are available from the algorithm. Note $r_{jk} = 0$ for $j > k$. Thus $\boldsymbol{R}$ is upper triangular.
- $\boldsymbol{X}$ is over-written by $\boldsymbol{Q}$ and $\boldsymbol{R}$ is stored in a separate array.
- The regular Gram-Schmidt is *unstable* (we loose orthogonality due to roundoff errors) when columns of $\boldsymbol{X}$ are collinear.
- *Modified Gram-Schmidt* (MGS): after each normalization step of $\boldsymbol{v}_k$, we replace $\tilde{\boldsymbol{x}}_j$, $j > k$, by its residual.

- Why MGS is better than GS? Read `http://cavern.uark.edu/~arnold/4353/CGSMGS.pdf`
- Computational cost of GS and MGS is $\sum_{k=1}^{p} 2n(k-1) \approx np^2$.

8 Lecture 8, Sep 23

Announcements

- HW2 due today. Submit both hardcopy and R code (`LastFirstHW2.R` or `LastFirstHW2.Rmd`)
- HW3 due next Tue.

Last time

- Cholesky decomposition with symmetric pivoting: $\boldsymbol{PAP}^T = \boldsymbol{LL}^T$
- Linear regression by Cholesky
- QR and linear regression
- QR by (modified) Gram-Schmidt: np^2 flops to get $\boldsymbol{X} = \boldsymbol{Q}_1\boldsymbol{R}_1$ (thin QR)

Today

- QR by Householder
- QR by Givens

QR by Householder

Photograph by Paul Halmos

Assume $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_p) \in \mathbb{R}^{n\times p}$ has full column rank.

- Idea: $\boldsymbol{H}_p \cdots \boldsymbol{H}_2 \boldsymbol{H}_1 \boldsymbol{X} = \begin{pmatrix} \boldsymbol{R}_1 \\ \boldsymbol{0} \end{pmatrix}$, where $\boldsymbol{H}_j \in \mathbb{R}^{n\times n}$ are the Householder transformation matrix. It yields the full QR where $\boldsymbol{Q} = \boldsymbol{H}_1 \cdots \boldsymbol{H}_p \in \mathbb{R}^{n\times n}$. Recall GS/MGS only produces the thin QR decomposition.

- For arbitrary $\boldsymbol{v}, \boldsymbol{w} \in \mathbb{R}^n$ with $\|\boldsymbol{v}\|_2 = \|\boldsymbol{w}\|_2$, we can construct a *Householder matrix* $\boldsymbol{H} = \boldsymbol{I}_n - 2\boldsymbol{u}\boldsymbol{u}^\mathsf{T}$, $\boldsymbol{u} = -\frac{1}{\|\boldsymbol{v}-\boldsymbol{w}\|_2}(\boldsymbol{v} - \boldsymbol{w})$, that carries $\boldsymbol{v}$ to $\boldsymbol{w}$:

$$\boldsymbol{H}\boldsymbol{v} = \boldsymbol{w}.$$

  $\boldsymbol{H}$ is symmetric and orthogonal. Calculation of Householder vector $\boldsymbol{u}$ costs $2n$ flops.

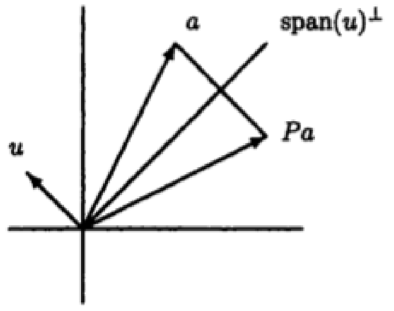


FIG. 2.3.1. ***Reflection of a vector a in a hyperplane with normal u.***

- Now choose $\boldsymbol{H}_1$ to zero the first column of $\boldsymbol{X}$ below diagonal

$$\boldsymbol{H}_1 \boldsymbol{x}_1 = \begin{pmatrix} \|\boldsymbol{x}_1\|_2 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

  Take $\boldsymbol{H}_2$ to zero the second column below diagonal; ...

$$
H_2H_1A = \begin{bmatrix} \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & 0 & \boxtimes & \times & \times \\ 0 & 0 & \boxtimes & \times & \times \\ 0 & 0 & \boxtimes & \times & \times \\ 0 & 0 & \boxtimes & \times & \times \end{bmatrix}
$$

In general, choose the j-th Householder transform $\boldsymbol{H}_j = \boldsymbol{I}_n - 2\boldsymbol{u}_j\boldsymbol{u}_j^{\mathsf{T}}$, where $\boldsymbol{u}_j = \begin{pmatrix} \boldsymbol{0}_{j-1} \\ \tilde{\boldsymbol{u}}_j \end{pmatrix}$, $\tilde{\boldsymbol{u}}_j \in \mathbb{R}^{n-j+1}$, to zero the j-th column below diagonal. $\boldsymbol{H}_j$ takes the form

$$
\boldsymbol{H}_j = \begin{pmatrix} \boldsymbol{I}_{j-1} & \\ & \boldsymbol{I}_{n-j+1} - 2\tilde{\boldsymbol{u}}_j\tilde{\boldsymbol{u}}_j^T \end{pmatrix} = \begin{pmatrix} \boldsymbol{I}_{j-1} & \\ & \tilde{\boldsymbol{H}}_j \end{pmatrix}.
$$

- Applying a Householder transform $\boldsymbol{H} = \boldsymbol{I} - 2\boldsymbol{u}\boldsymbol{u}^T$ to a matrix $\boldsymbol{X} \in \mathbb{R}^{n\times p}$

$$
\boldsymbol{H}\boldsymbol{X} = \boldsymbol{X} - 2\boldsymbol{u}(\boldsymbol{u}^{\mathsf{T}}\boldsymbol{X})
$$

costs $2np$ flops. *We never explicitly form the Householder matrices.*

Note applying $\boldsymbol{H}_j$ to $\boldsymbol{X}$ only needs $2(n-j+1)(p-j+1)$ flops.

- QR by Householder: $\boldsymbol{H}_p \cdots \boldsymbol{H}_1\boldsymbol{X} = \begin{pmatrix} \boldsymbol{R}_1 \\ \boldsymbol{0} \end{pmatrix}$.

- The process is done in place. Upper triangular part of $\boldsymbol{X}$ is overwritten by $\boldsymbol{R}_1$ and the essential Householder vectors ($\tilde{u}_{j1}$ is normalized to 1) are stored in $\boldsymbol{X}(j:n,j)$.

- At j-th stage

  1. computing the Householder vector $\tilde{\boldsymbol{u}}_j$ costs $2(n-j+1)$ flops
  2. applying the Householder transform $\tilde{\boldsymbol{H}}_j$ to the $\boldsymbol{X}(j:n,j:p)$ block costs $2(n-j+1)(p-j+1)$ flops

In total we need $\sum_{j=1}^{p}[2(n-j+1) + 2(n-j+1)(p-j+1)] \approx np^2 - \frac{1}{3}p^3$ flops.

- Where is $\boldsymbol{Q}$? $\boldsymbol{Q} = \boldsymbol{H}_1 \cdots \boldsymbol{H}_p$.

  In some applications, it's necessary to form the orthogonal matrix $\boldsymbol{Q}$.

  Accumulating $\boldsymbol{Q}$ costs another $np^2 - \frac{1}{3}p^3$ flops.

- When computing $\boldsymbol{Q}^\intercal \boldsymbol{v}$ or $\boldsymbol{Q}\boldsymbol{v}$ as in some applications (e.g., solve linear equation using QR), no need to form $\boldsymbol{Q}$. Simply apply Householder transforms successively.
  `qr.qy()` and `qr.qty()` in R do this.

- Computational cost of Householder QR for linear regression: $np^2 - \frac{1}{3}p^3$ (regression coefficients and $\hat{\sigma}^2$) or more (fitted values, s.e., ...).

Rank deficient $\boldsymbol{X}$: Householder QR with column pivoting

$\boldsymbol{X} \in \mathbb{R}^{n \times p}$ may not have full column rank.

- Idea (due to Businger and Golub 1965): at the j-th stage, swap the column in $\boldsymbol{X}(j:n, j:p)$ with maximum ℓ_2 norm to be the pivot column. If the maximum ℓ_2 norm is 0, it stops, ending with

$$\boldsymbol{X}\boldsymbol{\Pi} = \boldsymbol{Q} \begin{pmatrix} \boldsymbol{R}_{11} & \boldsymbol{R}_{12} \\ \mathbf{0}_{(n-r)\times r} & \mathbf{0}_{(n-r)\times(p-r)} \end{pmatrix},$$

  where $\boldsymbol{\Pi} \in \mathbb{R}^{p \times p}$ is a permutation matrix and r is the rank of $\boldsymbol{X}$. QR with column pivoting is rank revealing.

- The overhead of re-computing the column norms can be reduced by the property

$$\boldsymbol{Q}\boldsymbol{z} = \begin{pmatrix} \alpha \\ \boldsymbol{\omega} \end{pmatrix} \Rightarrow \|\boldsymbol{\omega}\|_2^2 = \|\boldsymbol{z}\|_2^2 - \alpha^2$$

  for any orthogonal matrix $\boldsymbol{Q}$.

- In R, the `qr()` function is a wrapper for various LINPACK (default) and LAPACK routines. It performs Householder QR with column pivoting and returns

  - `$qr`: a matrix of same size as input matrix
  - `$rank`: rank of the input matrix

  - `$pivot`: pivot vector
  - `$aux`: normalizing constants of Householder vectors

Auxiliary functions `qr.coef()`, `qr.resid()`, `qr.qy()`, `qr.qty()`, `qr.solve()`, ... are very helpful.

QR by Givens rotation (JM 5.7-5.8)

- Householder transform $\boldsymbol{H}_j$ introduces batch of zeros into a vector.
- Givens transform (aka Givens rotation, Jacobi rotation, plane rotation) selectively zeros one element of a vector.
- Overall QR by Givens rotation is less efficient than the Householder method, but is better suited for matrices with structured patterns of nonzero elements.
- Givens/Jacobi rotations:

$$\boldsymbol{G}(i,k,\theta) = \begin{pmatrix} 1 & & 0 & & 0 & & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & & c & & s & & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & & -s & & c & & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & & 0 & & 0 & & 1 \end{pmatrix},$$

  where $c = \cos(\theta)$ and $s = \sin(\theta)$. $\boldsymbol{G}(i,k,\theta)$ is orthogonal.

- Pre-multiplication by $\boldsymbol{G}(i,k,\theta)^T$ rotates counterclockwise θ radians in the (i,k) coordinate plane. If $\boldsymbol{x} \in \mathbb{R}^n$ and $\boldsymbol{y} = \boldsymbol{G}(i,k,\theta)^T\boldsymbol{x}$, then

$$y_j = \begin{cases} cx_i - sx_k & j = i \\ sx_i + cx_k & j = k \\ x_j & j \neq i,k \end{cases}.$$

  Apparently if we choose $\tan(\theta) = -x_k/x_i$, or equivalently,

$$c = \frac{x_i}{\sqrt{x_i^2 + x_k^2}}, \quad s = \frac{-x_k}{\sqrt{x_i^2 + x_k^2}},$$

  then $y_k = 0$.

- Pre-applying Givens transform $\boldsymbol{G}(i,k,\theta)^T \in \mathbb{R}^{n\times n}$ to a matrix $\boldsymbol{A} \in \mathbb{R}^{n\times m}$ only effects two rows of $\boldsymbol{A}$:

$$\boldsymbol{A}([i,k],:) \leftarrow \begin{pmatrix} c & s \\ -s & c \end{pmatrix}^T \boldsymbol{A}([i,k],:),$$

  costing $2m$ flops.

- Post-applying Givens transform $\boldsymbol{G}(i,k,\theta) \in \mathbb{R}^{m\times m}$ to a matrix $\boldsymbol{A} \in \mathbb{R}^{n\times m}$ only effects two columns of $\boldsymbol{A}$:

$$\boldsymbol{A}(:,[i,k]) \leftarrow \boldsymbol{A}(:,[i,k]) \begin{pmatrix} c & s \\ -s & c \end{pmatrix},$$

  costing $2n$ flops.

- QR by Givens: $\boldsymbol{G}_t^T \cdots \boldsymbol{G}_1^T \boldsymbol{X} = \begin{pmatrix} \boldsymbol{R}_1 \\ \boldsymbol{0} \end{pmatrix}$

$$\begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \end{bmatrix} \xrightarrow{(3,4)} \begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ 0 & \times & \times \end{bmatrix} \xrightarrow{(2,3)} \begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ 0 & \times & \times \\ 0 & \times & \times \end{bmatrix} \xrightarrow{(1,2)}$$

$$\begin{bmatrix} \times & \times & \times \\ 0 & \times & \times \\ 0 & \times & \times \\ 0 & \times & \times \end{bmatrix} \xrightarrow{(3,4)} \begin{bmatrix} \times & \times & \times \\ 0 & \times & \times \\ 0 & \times & \times \\ 0 & 0 & \times \end{bmatrix} \xrightarrow{(2,3)} \begin{bmatrix} \times & \times & \times \\ 0 & \times & \times \\ 0 & 0 & \times \\ 0 & 0 & \times \end{bmatrix} \xrightarrow{(3,4)} \boldsymbol{R}$$

- Zeros in $\boldsymbol{X}$ can also be introduced row-by-row.
- If $\boldsymbol{X} \in \mathbb{R}^{n \times p}$, the total cost is $(3/2)np^2 - p^3/2$ flops and $O(np)$ square roots.
- Note each Givens transform can be summarized by a single number, which is stored in the zeroed entry of $\boldsymbol{X}$.
- Fast Givens transform avoids taking square roots.

9 Lecture 9, Sep 25

Announcements

- HW2 returned. Feedback:
  - Only Q2, Q4, and Q5 are graded. Maximum points 60.
  - Code style penalty doubles.
  - Late penalty: 5 pts/day.
  - Sketch of solution: `http://hua-zhou.github.io/teaching/st758-2014fall/hw02sol.html`. Please compare to your code carefully and understand why.
- HW3 due next Tue. FAQs at `http://hua-zhou.github.io/teaching/st758-2014fall/st758fall2014/2014/09/24/hw3-hints.html`
- HW4 posted. Due Oct 14.
- Answer to Susheela's questions:
  - *Backward accumulation algorithm* for accumulating $\boldsymbol{Q} = \boldsymbol{H}_1 \cdots \boldsymbol{H}_p$ from Householder vectors stored in the Household QR output costs about $2(n^2p - np^2 + p^3/3)$ flops. See Golub and Van Loan (1996, p213).
  - Computing $\boldsymbol{Q}^T\boldsymbol{Y}$, $\boldsymbol{Y} \in \mathbb{R}^{n \times r}$ (r right hand sides), costs about $(2np - p^2)r$ flops. See Golub and Van Loan (1996, p212).
- Answer to Meng's question: Why (how) normalize Householder vector such that the 1st element is 1? See Golub and Van Loan (1996, Algorithm 5.1.1, p212).

Last time

- QR by Householder
- QR by Givens

Today

- Sweep operator (KL 7.4-7.6, JM 5.12)

Sweep operator

Assume $\boldsymbol{A} \succeq \boldsymbol{0}_{n\times n}$.

- KL 7.4-7.6; JM 5.12; Also see "*A tutorial on the SWEEP operator*" by James H. Goodnight. `http://www.jstor.org/stable/2683825`
- Note the (anti-symmetric) version in JM is different from the (symmetric) version in KL. I follow the convention in KL.
- *Sweep* on the k-th diagonal entry $a_{kk} \neq 0$ yields $\hat{\boldsymbol{A}}$ with entries

$$\begin{aligned} \hat{a}_{kk} &= -\frac{1}{a_{kk}} \\ \hat{a}_{ik} &= \frac{a_{ik}}{a_{kk}} \\ \hat{a}_{kj} &= \frac{a_{kj}}{a_{kk}} \\ \hat{a}_{ij} &= a_{ij} - \frac{a_{ik}a_{kj}}{a_{kk}}, \quad i \neq k, j \neq k. \end{aligned}$$

  $n^2/2$ flops (taking into account of symmetry).
- *Inverse sweep* sends $\boldsymbol{A}$ to $\check{\boldsymbol{A}}$ with entries

$$\begin{aligned} \check{a}_{kk} &= -\frac{1}{a_{kk}} \\ \check{a}_{ik} &= -\frac{a_{ik}}{a_{kk}} \\ \check{a}_{kj} &= -\frac{a_{kj}}{a_{kk}} \\ \check{a}_{ij} &= a_{ij} - \frac{a_{ik}a_{kj}}{a_{kk}}, \quad i \neq k, j \neq k. \end{aligned}$$

  $n^2/2$ flops (taking into account of symmetry).
- $\check{\hat{\boldsymbol{A}}} = \boldsymbol{A}$.
- Block form of sweep: Let the symmetric matrix $\boldsymbol{A}$ be partitioned as $\boldsymbol{A} = \begin{pmatrix} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} \\ \boldsymbol{A}_{21} & \boldsymbol{A}_{22} \end{pmatrix}$. If possible, sweep on the diagonal entries of $\boldsymbol{A}_{11}$ yields

$$\hat{\boldsymbol{A}} = \begin{pmatrix} -\boldsymbol{A}_{11}^{-1} & \boldsymbol{A}_{11}^{-1}\boldsymbol{A}_{12} \\ \boldsymbol{A}_{21}\boldsymbol{A}_{11}^{-1} & \boldsymbol{A}_{22} - \boldsymbol{A}_{21}\boldsymbol{A}_{11}^{-1}\boldsymbol{A}_{12} \end{pmatrix}.$$

  Order dose *not* matter.

- Pd and determinant: $\boldsymbol{A}$ is pd if and only if each diagonal entry can be swept in succession and is positive until it is swept. When a diagonal entry of a pd matrix $\boldsymbol{A}$ is swept, it becomes negative and remains negative thereafter. Taking the product of diagonal entries just before each is swept yields the determinant of $\boldsymbol{A}$.

- Linear regression by sweep. Sweep on $\begin{pmatrix} \boldsymbol{X}^\mathsf{T}\boldsymbol{X} & \boldsymbol{X}^\mathsf{T}\boldsymbol{y} \\ \boldsymbol{y}^\mathsf{T}\boldsymbol{X} & \boldsymbol{y}^\mathsf{T}\boldsymbol{y} \end{pmatrix}$ yields

$$\begin{pmatrix} -(\boldsymbol{X}^\mathsf{T}\boldsymbol{X})^{-1} & (\boldsymbol{X}^\mathsf{T}\boldsymbol{X})^{-1}\boldsymbol{X}^\mathsf{T}\boldsymbol{y} \\ \boldsymbol{y}^\mathsf{T}\boldsymbol{X}(\boldsymbol{X}\boldsymbol{X})^{-1} & \boldsymbol{y}^\mathsf{T}\boldsymbol{y} - \boldsymbol{y}^\mathsf{T}\boldsymbol{X}(\boldsymbol{X}^\mathsf{T}\boldsymbol{X})^{-1}\boldsymbol{X}^\mathsf{T}\boldsymbol{y} \end{pmatrix} = \begin{pmatrix} -\frac{1}{\sigma^2}\mathrm{Var}(\hat{\boldsymbol{\beta}}) & \hat{\boldsymbol{\beta}} \\ \hat{\boldsymbol{\beta}}^\mathsf{T} & \|\boldsymbol{y} - \hat{\boldsymbol{y}}\|_2^2 \end{pmatrix}.$$

  In total $np^2/2 + p^3/2$ flops.

- Sweep is useful for stepwise regression, (conditional) multivariate normal density calculation, MANOVA, ...

- Warning: the `sweep()` function in R has nothing do to with the sweep operator here.

- Demo code: `http://hua-zhou.github.io/teaching/st758-2014fall/sweep.html`

10 Lecture 10, Sep 30

Announcements

- HW3 due today. Submit hard copy + code (`LastFirstHW3.R` or `LastFirstHW3.Rmd`)
- HW4 posted. Due Oct 14.
- TA office hours on Oct 8 @ 10AM-12PM?
- Some FAQs:
  - (Maggie) If $\boldsymbol{A} \in \mathbb{R}^{n \times p}$ has full rank, Householder QR gives $\boldsymbol{A} = \boldsymbol{Q}_1 \boldsymbol{R}_1$ and MGS gives $\boldsymbol{A} = \boldsymbol{Q}_2 \boldsymbol{R}_2$, where $\boldsymbol{Q}_1, \boldsymbol{Q}_2 \in \mathbb{R}^{n \times p}$, $\boldsymbol{Q}_1^T \boldsymbol{Q}_1 = \boldsymbol{Q}_2^T \boldsymbol{Q}_2 = \boldsymbol{I}_p$, and $\boldsymbol{R}_1, \boldsymbol{R}_2 \in \mathbb{R}^{p \times p}$ are upper triangular with positive diagonal entries. How do we know $\boldsymbol{Q}_1 = \boldsymbol{Q}_2$?
  - (Meng) Block sweep simply means sweeping diagonal entries in that block sequentially. Order does *not* matter.
  - Check "80 character rule" in R Studio.
  - (Liuyi) What exactly is that `pivot` from the output of `qr()` and `chol()` (with `pivot = TRUE`) functions in R? See hints at `http://hua-zhou.github.io/teaching/st758-2014fall/st758fall2014/2014/09/24/hw3-hints.html`

Last time

- HW2 review
- Sweep operator (KL 7.4-7.6, JM 5.12)

Today

- Summary of numerical methods for linear regression
- Summary of solving linear equations: overdetermined system
- Condition number for solving linear equations

Summary of linear regression: Table on KL p105

Method	Flops	Remarks	Software	Stability
Sweep	$np^2/2 + p^3/2$	$(\boldsymbol{X}^\top \boldsymbol{X})^{-1}$ available	SAS	less stable
Cholesky	$np^2/2 + p^3/6$			less stable
QR by Householder	$np^2 - p^3/3$		R	
QR by MGS	np^2	$\boldsymbol{Q}_1$ available		more stable

Table 1: Numerical methods for linear regression. In order of stability.

Remarks:

- When $n \gg p$, sweep and Cholesky are twice faster than QR and need less space.

  But QR methods are more stable and produce numerically more accurate solution.

- Although sweep is slower than Cholesky, it yields standard errors and so on.

- Sweep is useful for stepwise regression, multivariate normal calculation, and numerous other statistical applications.

- MGS appears slower than Householder, but it yields $\boldsymbol{Q}_1$.

  "There is simply no such thing as a universal 'gold standard' when it comes to algorithms".

Summary of solving linear equations

Consider linear system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$.

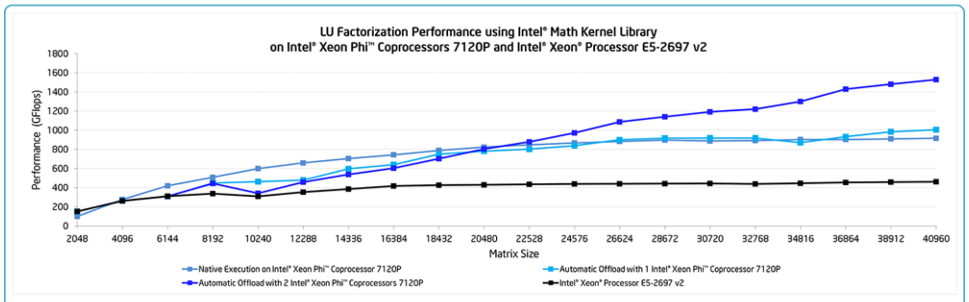
- We now know some good numerical methods for the least squares problem, which is essentially "solving" an overdetermined system (a tall $\boldsymbol{A}$).

- Table 2 compares the flops of some methods (in order of stability) for solving a square (unstructured) $\boldsymbol{A} \in \mathbb{R}^{n\times n}$.

Method	Flops	Stability
Gaussian elimination	$n^3/3$	less stable
QR by Householder	$(2/3)n^3$	
QR by MGS	n^3	
SVD	$6n^3$	most stable

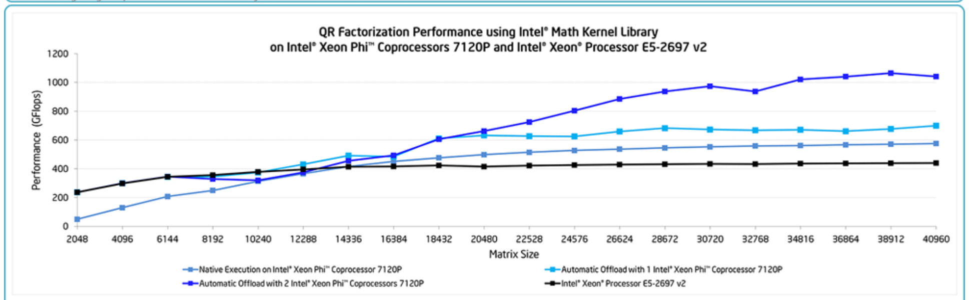
Table 2: Flops of different numerical methods for $n \times n$ square linear systems, assuming availability of the right hand side at time of decomposition.

- Flop count is not everything. GE/LU has a higher memory traffic and vectorization overheads, and QR approach is comparable in efficiency. QR methods are more stable.



Configuration Info - Software Versions: Intel® Math Kernel Library (Intel® MKL) 11.1, Intel® Manycore Platform Software Stack (MPSS) 2.1.6720-15; Hardware: Intel® Xeon® Processor E5-2697 v2, 2 Twelve-Core CPUs (30MB LLC, 2.7GHz), 32GB DDR3 RAM (1333MHz); Intel® Xeon Phi™ Coprocessor 7120P, 61 cores (30.5MB total cache, 1.238GHz), 16GB GDDR5 Memory; Operating System: RHEL 6.1 GA x86_64.

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. * Other brands and names are the property of their respective owners. Benchmark Source: Intel Corporation. September 2013.
Optimization Notice: Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice. Notice revision #20110804



Configuration Info - Software Versions: Intel® Math Kernel Library (Intel® MKL) 11.1, Intel® Manycore Platform Software Stack (MPSS) 2.1.6720-15; Hardware: Intel® Xeon® Processor E5-2697 v2, 2 Twelve-Core CPUs (30MB LLC, 2.7GHz), 32GB DDR3 RAM (1333MHz); Intel® Xeon Phi™ Coprocessor 7120P, 61 cores (30.5MB total cache, 1.238GHz), 16GB GDDR5 Memory; Operating System: RHEL 6.1 GA x86_64.

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. * Other brands and names are the property of their respective owners. Benchmark Source: Intel Corporation. September 2013.
Optimization Notice: Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice. Notice revision #20110804

- Solve an underdetermined system (a flat $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ of full row rank) by QR –

version 1. First compute QR on $\boldsymbol{A}^T$

$$\boldsymbol{A}^T = \boldsymbol{QR} = \boldsymbol{Q} \begin{pmatrix} \boldsymbol{R}_1 \\ \boldsymbol{0}_{(n-m)\times m} \end{pmatrix}.$$

Then $\boldsymbol{Ax} = \boldsymbol{b}$ becomes

$$(\boldsymbol{QR})^T \boldsymbol{x} = \begin{pmatrix} \boldsymbol{R}_1^T & \boldsymbol{0}_{m\times(n-m)} \end{pmatrix} \begin{pmatrix} \boldsymbol{z}_1 \\ \boldsymbol{z}_2 \end{pmatrix} = \boldsymbol{b},$$

where

$$\boldsymbol{Q}^T \boldsymbol{x} = \begin{pmatrix} \boldsymbol{z}_1 \\ \boldsymbol{z}_2 \end{pmatrix}.$$

$\boldsymbol{z}_1$ is determined from $\boldsymbol{R}_1^T \boldsymbol{z}_1 = \boldsymbol{b}$. If we take $\boldsymbol{z}_2 = \boldsymbol{0}_{n-m}$, then we obtain the minimum norm solution (why?).

- Solve an underdetermined system (a flat $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ of full row rank) by QR – version 2. First compute QR with column pivoting on $\boldsymbol{A}$

$$\boldsymbol{A\Pi} = \boldsymbol{Q} \begin{pmatrix} \boldsymbol{R}_1 & \boldsymbol{R}_2 \end{pmatrix},$$

where $\boldsymbol{R}_1 \in \mathbb{R}^{m\times m}$ is upper triangular and $\boldsymbol{R}_2 \in \mathbb{R}^{m\times(n-m)}$. Thus $\boldsymbol{Ax} = \boldsymbol{b}$ transforms to

$$\begin{pmatrix} \boldsymbol{R}_1 & \boldsymbol{R}_2 \end{pmatrix} \begin{pmatrix} \boldsymbol{z}_1 \\ \boldsymbol{z}_2 \end{pmatrix} = \boldsymbol{Q}^T \boldsymbol{b},$$

where

$$\boldsymbol{\Pi}^T \boldsymbol{x} = \begin{pmatrix} \boldsymbol{z}_1 \\ \boldsymbol{z}_2 \end{pmatrix}.$$

One solution is obtained by $\boldsymbol{z}_1 = \boldsymbol{R}_1^{-1} \boldsymbol{Q}^T \boldsymbol{b}$ and $\boldsymbol{z}_2 = \boldsymbol{0}_{n-m}$. It is not guaranteed to be of minimum norm.

Condition number for linear equations (matrix inversion)

- Assume $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ is nonsingular and consider the system of linear equation $\boldsymbol{Ax} = \boldsymbol{b}$. The solution is $\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{b}$. We want to know how the solution changes with a small perturbation of the input $\boldsymbol{b}$ (or $\boldsymbol{A}$).

- Let $\tilde{\boldsymbol{b}} = \boldsymbol{b} + \Delta\boldsymbol{b}$. Then $\tilde{\boldsymbol{x}} = \boldsymbol{A}^{-1}(\boldsymbol{b} + \Delta\boldsymbol{b}) = \boldsymbol{x} + \Delta\boldsymbol{x}$. Thus

$$\|\Delta\boldsymbol{x}\| = \|\boldsymbol{A}^{-1}\Delta\boldsymbol{b}\| \le \|\boldsymbol{A}^{-1}\|\|\Delta\boldsymbol{b}\|.$$

  Because $\boldsymbol{b} = \boldsymbol{A}\boldsymbol{x}$, $\frac{1}{\|\boldsymbol{x}\|} \le \|\boldsymbol{A}\|\frac{1}{\|\boldsymbol{b}\|}$. This results

$$\frac{\|\Delta\boldsymbol{x}\|}{\|\boldsymbol{x}\|} \le \|\boldsymbol{A}\|\|\boldsymbol{A}^{-1}\|\frac{\|\Delta\boldsymbol{b}\|}{\|\boldsymbol{b}\|}.$$

- $\kappa(\boldsymbol{A}) = \|\boldsymbol{A}\|\|\boldsymbol{A}^{-1}\|$ is called the *condition number for inversion*. It depends on the matrix norm being used. κ_p means condition number defined from matrix-p norm.

- Large condition number means "bad".

- Some useful facts

$$\begin{aligned}
\kappa(\boldsymbol{A}) &= \kappa(\boldsymbol{A}^{-1}) \\
\kappa(c\boldsymbol{A}) &= \kappa(\boldsymbol{A}) \\
\kappa(\boldsymbol{A}) &\ge 1 \\
\kappa_1(\boldsymbol{A}) &= \kappa_\infty(\boldsymbol{A}^\intercal) \\
\kappa_2(\boldsymbol{A}) &= \kappa_2(\boldsymbol{A}^\intercal) = \frac{\sigma_1(\boldsymbol{A})}{\sigma_n(\boldsymbol{A})} \\
\kappa_2(\boldsymbol{A}^\intercal\boldsymbol{A}) &= \frac{\lambda_1(\boldsymbol{A}^\intercal\boldsymbol{A})}{\lambda_n(\boldsymbol{A}^\intercal\boldsymbol{A})} = \kappa_2^2(\boldsymbol{A}) \ge \kappa_2(\boldsymbol{A}).
\end{aligned}$$

  The last fact says that the condition number of $\boldsymbol{A}^\intercal\boldsymbol{A}$ can be much larger than that of $\boldsymbol{A}$.

- The smallest singular value σ_n indicates the "distance to the trouble".

- Condition number for the least squares problem is more complicated. Roughly speaking, the method based on normal equation (Cholesky, sweep) has a condition depending on $\kappa_2(\boldsymbol{X})^2$. QR for a "small residuals" least squares problem has a condition depending on $\kappa_2(\boldsymbol{X})$.

- Numerically, consider the simple case

$$\boldsymbol{X} = \begin{pmatrix} 1 & 1 \\ 10^{-3} & 0 \\ 0 & 10^{-3} \end{pmatrix}.$$

Forming normal equation yields a singular Gramian matrix

$$\boldsymbol{X}^{\mathsf{T}}\boldsymbol{X} = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$$

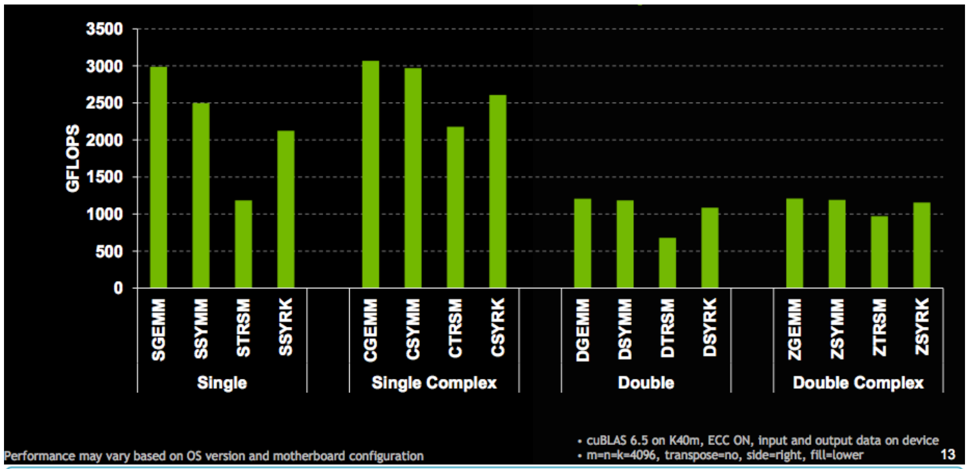
if executed with a precision of 6 digits.

- In `R`, the `kappa()` function (wrapper of `DTRCON` in LAPACK and `DTRCO` in LINPACK) computes or approximates (default) the condition number of a matrix.
- In regression problems, standardizing the predictors could improve the condition. See demo on the Longley data `http://hua-zhou.github.io/teaching/st758-2014fall/longleycond.html`.
- In design of experiments (DoE), people favor orthogonal design. Why?

11 Lecture 11, Oct 2

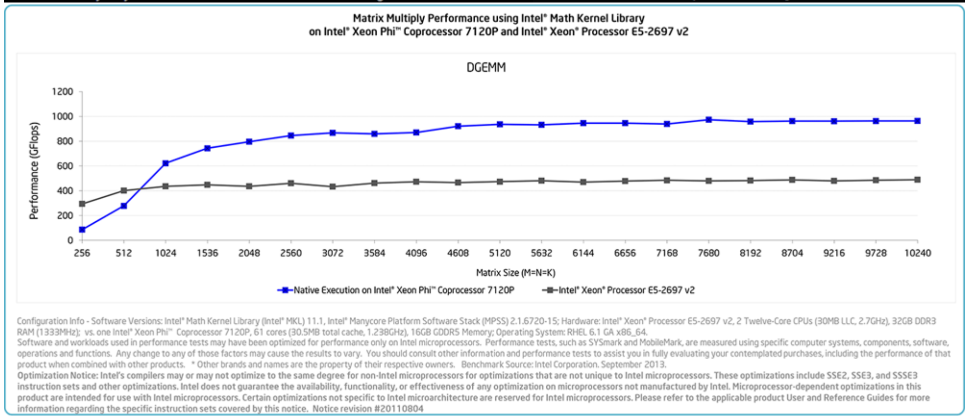
Announcements

- HW3 returned. Feedback:
  - Only Q2(d) and Q3 are graded. Total points is 50.
  - Style issues: penalty doubles.
  - Late penalty: 5 pts/day.
  - Q1, probabilistic proof?
  - By default, `qr()` function does Householder QR *with column pivoting*! Need to use the `pivot` vector to permute results.
  - Make good use of `crossprod()` and `tcrossprod()` functions to compute Grammian matrix, avoid unnecessary matrix transpose, and so on. Check source code:
    `/R-3.1.1/src/main/names.c`
    `/R-3.1.1/src/main/array.c`
  - How to generate rank deficient matrix?
  - Efficient implementation of sweep operator?
  - Sketch of solution: `http://hua-zhou.github.io/teaching/st758-2014fall/hw03sol.html`. Please compare to your code carefully and understand why.
- HW4 due Oct 14.
- TA office hours on Wed Oct 8. No TA office hours on Fri Oct 10 (fall break).
- Caleb's question: Nvidia Tesla vs Intel Xeon Phi



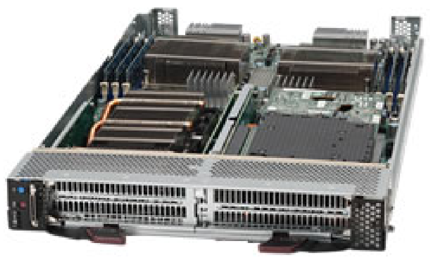
Performance may vary based on OS version and motherboard configuration

- cuBLAS 6.5 on K40m, ECC ON, input and output data on device
- m=n=k=4096, transpose=no, side=right, fill=lower



Configuration Info - Software Versions: Intel® Math Kernel Library (Intel® MKL) 11.1, Intel® Manycore Platform Software Stack (MPSS) 2.1.6720-15; Hardware: Intel® Xeon® Processor E5-2697 v2, 2 Twelve-Core CPUs (30MB LLC, 2.7GHz), 32GB DDR3 RAM (1333MHz); vs. one Intel® Xeon Phi™ Coprocessor 7120P, 61 cores (30.5MB total cache, 1.238GHz), 16GB GDDR5 Memory; Operating System: RHEL 6.1 GA x86_64.
Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. * Other brands and names are the property of their respective owners. Benchmark Source: Intel Corporation. September 2013.
Optimization Notice: Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice. Notice revision #20110804

Our department has at least 2 servers each with 4 Nvidia Tesla M2070Q GPUs. Each Tesla M2070Q has 6G memory (5.25G with ECC), 786K L2 cache, 448 cores @ 1.15GHz, and theoretical throughput of 1288 SP GFLOPS or 512 DP GFLOPS.



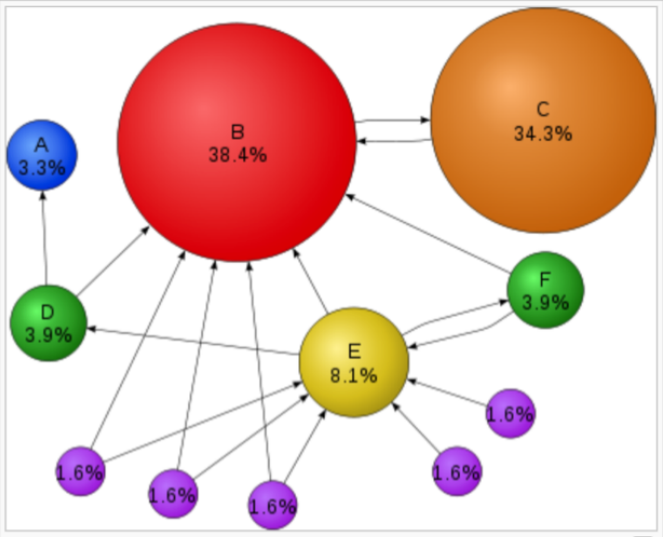
Last time

- Summary of linear regressions
- Summary of numerical methods for solving linear equations: tall $\boldsymbol{A}$ (least squares, over-determined system), square $\boldsymbol{A}$, and flat $\boldsymbol{A}$ (underdetermined system)
- Condition number for solving linear equations

Today

- Iterative solvers for linear systems

Iterative method for solving linear equations: introduction



Mathematical **PageRanks** for a simple network, expressed as percentages. (Google uses a logarithmic scale.) Page C has a higher PageRank than Page E, even though there are fewer links to C; the one link to C comes from an important page and hence is of high value. If web surfers who start on a random page have an 85% likelihood of choosing a random link from the page they are currently visiting, and a 15% likelihood of jumping to a page chosen at random from the entire web, they will reach Page E 8.1% of the time. (The 15% likelihood of jumping to an arbitrary page corresponds to a damping factor of 85%.) Without damping, all web surfers would eventually end up on Pages A, B, or C, and all other pages would have PageRank zero. In the presence of damping, Page A effectively links to all pages in the web, even though it has no outgoing links of its own.

- Direct method (flops fixed *a priori*) vs iterative methods:

- Direct method (GE/LU, Cholesky, QR, SVD): good for dense, small or moderate sized, unstructured $\boldsymbol{A}$
- Iterative methods (Jacobi, Gauss-Seidal, SOR, conjugate-gradient, GMRES): good for large, sparse, or structured linear system, parallel computing, warm start

- PageRank (HW4):
  - $\boldsymbol{A} \in \{0,1\}^{n \times n}$ the connectivity matrix with entries

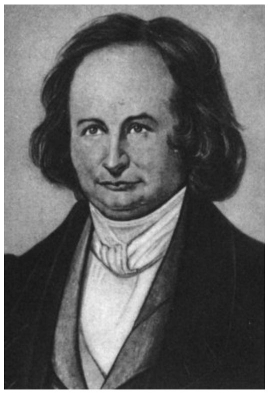
    $$a_{ij} = \begin{cases} 1 & \text{if page } i \text{ links to page } j \\ 0 & \text{otherwise} \end{cases}.$$

    $n \approx 10^9$ in Sep 2014.
  - $r_i = \sum_j a_{ij}$ is the out-degree of page i.
  - Imagine a random surfer wandering on internet according to following rules:
    * From a page i with $r_i > 0$
      - with probability p, (s)he randomly chooses a link on page i (uniformly) and follows that link to the next page
      - with probability $1-p$, (s)he randomly chooses one page from the set of all n pages (uniformly) and proceeds to that page
    * From a page i with $r_i = 0$ (a dangling page), (s)he randomly chooses one page from the set of all n pages (uniformly) and proceeds to that page

    The process defines a Markov chain on the space of n pages.
  - Stationary distribution of this Markov chain gives the ranks (probabilities) of each page.
  - Stationary distribution is the top left eigenvector of the transition matrix $\boldsymbol{P}$ corresponding to eigenvalue 1. Equivalently it can be cast as a linear equation.
  - Largest matrix computation in world (?).

- GE/LU will take $(10^9)^3/3/10^{12} \approx 3.33 \times 10^{14}$ seconds $\approx 1 \times 10^7$ years on a tera-flop supercomputer!
- Iterative methods come to the rescue.

Jacobi method



Solve linear system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$.

- Jacobi iteration:

$$x_i^{(t+1)} = \frac{b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(t)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(t)}}{a_{ii}}.$$

- Requires non-zero diagonal element!
- One round costs n^2 flops with an unstructured $\boldsymbol{A}$. Gain over GE/LU if converges in $o(n)$ iterations. Saving is huge for *sparse* or *structured* $\boldsymbol{A}$. By structured, we mean the matrix-vector multiplication $\boldsymbol{A}\boldsymbol{v}$ is fast.
- Splitting: $\boldsymbol{A} = \boldsymbol{L} + \boldsymbol{D} + \boldsymbol{U}$.
- Jacobi: $\boldsymbol{D}\boldsymbol{x}^{(t+1)} = -(\boldsymbol{L} + \boldsymbol{U})\boldsymbol{x}^{(t)} + \boldsymbol{b}$, i.e.,

$$\boldsymbol{x}^{(t+1)} = -\boldsymbol{D}^{-1}(\boldsymbol{L} + \boldsymbol{U})\boldsymbol{x}^{(t)} + \boldsymbol{D}^{-1}\boldsymbol{b}.$$

Gauss-Seidel

- Gauss-Seidel iteration:

$$x_i^{(t+1)} = \frac{b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(t+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(t)}}{a_{ii}}.$$

- With splitting, $(\boldsymbol{D} + \boldsymbol{L})\boldsymbol{x}^{(t+1)} = -\boldsymbol{U}\boldsymbol{x}^{(t)} + \boldsymbol{b}$, i.e.,

$$\boldsymbol{x}^{(t+1)} = -(\boldsymbol{D} + \boldsymbol{L})^{-1}\boldsymbol{U}\boldsymbol{x}^{(t)} + (\boldsymbol{D} + \boldsymbol{L})^{-1}\boldsymbol{b}.$$

- GS converges for any $\boldsymbol{x}^{(0)}$ for symmetric and pd $\boldsymbol{A}$.
- Convergence rate of Gauss-Seidel is the spectral radius of the $(\boldsymbol{D} + \boldsymbol{L})^{-1}\boldsymbol{U}$.
- Comparing Jacobi and GS, Jacobi is particularly attractive for parallel computing.

Successive over-relaxation (SOR)

- SOR: $x_i^{(t+1)} = \omega(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(t+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(t)})/a_{ii} + (1-\omega)x_i^{(t)}$, i.e.,

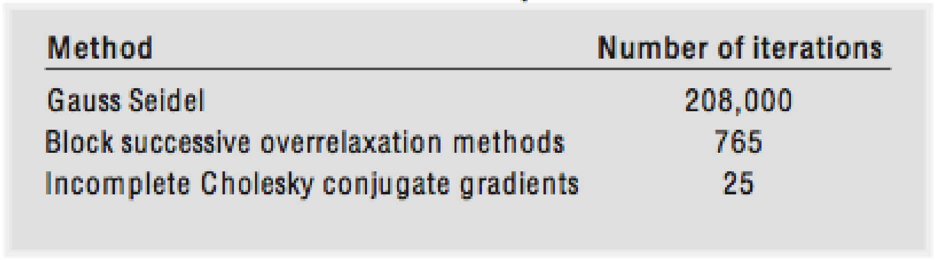
$$\boldsymbol{x}^{(t+1)} = (\boldsymbol{D} + \omega\boldsymbol{L})^{-1}[(1-\omega)\boldsymbol{D} - \omega\boldsymbol{U}]\boldsymbol{x}^{(t)} + (\boldsymbol{D} + \omega\boldsymbol{L})^{-1}(\boldsymbol{D} + \boldsymbol{L})^{-1}\omega\boldsymbol{b}.$$

- Need to pick $\omega \in [0, 1]$ beforehand, with the goal of improving convergence rate.

Conjugate gradient method

Solving $\boldsymbol{Ax} = \boldsymbol{b}$ is equivalent to minimizing the quadratic function $\frac{1}{2}\boldsymbol{x}^T\boldsymbol{Ax} - \boldsymbol{b}^T\boldsymbol{x}$. To do later, when we study optimization. Conjugate gradient and its variants are the top-notch iterative methods for solving huge, structured linear systems.

Table 1. Kershaw's results for a fusion problem.



Method	Number of iterations
Gauss Seidel	208,000
Block successive overrelaxation methods	765
Incomplete Cholesky conjugate gradients	25

12 Lecture 12, Oct 7

Announcements

- HW4 due next Tue Oct 14
- HW5 will be posted this week
- TA office hours this week: Wed Oct 8 @ 10A-12P
- Dr. Zhou office hours this week: Tue Oct 7 @ 4P-5P, Thu Oct 9 by appointment

Last time

- HW3 review
- Iterative methods for solving linear equations

Today

- A catalog of "easy" linear systems – last topic in solving linear equations
- Eigen-decomposition and SVD

A list of "easy" linear systems

Consider $\boldsymbol{Ax} = \boldsymbol{b}$, $\boldsymbol{A} \in \mathbb{R}^{n \times n}$. Or, consider matrix inverse (if you want). $\boldsymbol{A}$ can be huge. Keep massive data in mind: 1000 Genome Project, NetFlix, Google PageRank, finance, spatial statistics, ... We should be alert to many easy linear systems. *Don't waste computing resources by bad choices of algorithms!*

- Diagonal: n flops.
- Tridiagonal/banded: Band LU, band Cholesky, ... roughly $O(n)$ flops
- Triangular: $n^2/2$ flops
- Block diagonal: Suppose $n = \sum_i n_i$. $(\sum_i n_i)^3/3$ vs $\sum_i n_i^3/3$.
- Kronecker product: $(\boldsymbol{A} \otimes \boldsymbol{B})^{-1} = \boldsymbol{A}^{-1} \otimes \boldsymbol{B}^{-1}$, $(\boldsymbol{C}^{\mathsf{T}} \otimes \boldsymbol{A})\text{vec}\boldsymbol{B} = \text{vec}(\boldsymbol{ABC})$ fits iterative method.

- Sparsity: iterative method, or sparse matrix decomposition.
  Remark: Probably the easiest recognizable structure. Familiarize yourself with the sparse matrix computation tools in Matlab, R (`Matrix` package), MKL (sparse BLAS), ... as soon as possible.

- Easy plus low rank: $\boldsymbol{U} \in \mathbb{R}^{n \times m}$, $\boldsymbol{V} \in \mathbb{R}^{n \times m}$, $m \ll n$. Woodbury formula

$$(\boldsymbol{A} + \boldsymbol{U}\boldsymbol{V}^\top)^{-1} = \boldsymbol{A}^{-1} - \boldsymbol{A}^{-1}\boldsymbol{U}(\boldsymbol{I}_m + \boldsymbol{V}^\top \boldsymbol{A}^{-1}\boldsymbol{U})^{-1}\boldsymbol{V}^\top \boldsymbol{A}^{-1}.$$

  Keep HW2 Q5 in mind.

- Easy plus border: For $\boldsymbol{A}$ pd and $\boldsymbol{V}$ full row rank,

$$\begin{pmatrix} \boldsymbol{A} & \boldsymbol{V}^\top \\ \boldsymbol{V} & \boldsymbol{0} \end{pmatrix}^{-1} = \begin{pmatrix} \boldsymbol{A}^{-1} - \boldsymbol{A}^{-1}\boldsymbol{V}^\top(\boldsymbol{V}\boldsymbol{A}^{-1}\boldsymbol{V}^\top)^{-1}\boldsymbol{V}\boldsymbol{A}^{-1} & \boldsymbol{A}^{-1}\boldsymbol{V}^\top(\boldsymbol{V}\boldsymbol{A}^{-1}\boldsymbol{V}^\top)^{-1} \\ (\boldsymbol{V}\boldsymbol{A}^{-1}\boldsymbol{V}^\top)^{-1}\boldsymbol{V}\boldsymbol{A}^{-1} & -(\boldsymbol{V}\boldsymbol{A}^{-1}\boldsymbol{V}^\top)^{-1} \end{pmatrix}.$$

- Orthogonal: n^2 flops *at most*. Permutation matrix, Householder matrix, Jacobi matrix, ... take less.

- Toeplitz systems:

$$\boldsymbol{T} = \begin{pmatrix} r_0 & r_1 & r_2 & r_3 \\ r_{-1} & r_0 & r_1 & r_2 \\ r_{-2} & r_{-1} & r_0 & r_1 \\ r_{-3} & r_{-2} & r_{-1} & r_0 \end{pmatrix}.$$

  $\boldsymbol{T}\boldsymbol{x} = \boldsymbol{b}$, where $\boldsymbol{T}$ is pd and Toeplitz, can be solved in $O(n^2)$ flops. Durbin algorithm (Yule-Walker equation), Levinson algorithm (general $\boldsymbol{b}$), Trench algorithm (inverse). These matrices occur in auto-regressive models and econometrics.

- Circulant systems: Toeplitz matrix with wraparound

$$C(\boldsymbol{z}) = \begin{pmatrix} z_0 & z_4 & z_3 & z_2 & z_1 \\ z_1 & z_0 & z_4 & z_3 & z_2 \\ z_2 & z_1 & z_0 & z_4 & z_3 \\ z_3 & z_2 & z_1 & z_0 & z_4 \\ z_4 & z_3 & z_2 & z_1 & z_0 \end{pmatrix},$$

  DCT (discrete cosine transform) and DST (discrete sine transform).
  FFT type algorithms.

- Vandermonde matrix: such as in interpolation and approximation problems

$$\boldsymbol{V}(x_0, \ldots, x_n) = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ x_0 & x_1 & \cdots & x_n \\ \vdots & \vdots & & \vdots \\ x_0^n & x_1^n & \cdots & x_n^n \end{pmatrix}.$$

  $\boldsymbol{V}\boldsymbol{x} = \boldsymbol{b}$ or $\boldsymbol{V}^T\boldsymbol{x} = \boldsymbol{b}$ can be solved in $O(n^2)$ flops.

- Cauchy-like matrices:

$$\boldsymbol{\Omega}\boldsymbol{A} - \boldsymbol{A}\boldsymbol{\Lambda} = \boldsymbol{R}\boldsymbol{S}^T,$$

  where $\boldsymbol{\Omega} = \text{diag}(\omega_1, \ldots, \omega_n)$ and $\boldsymbol{\Lambda} = (\lambda_1, \ldots, \lambda_n)$. $O(n)$ flops for LU and QR.

- Structured-rank problems: semiseparable matrices (LU and QR takes $O(n)$ flops), quasiseparable matrices, ...

- Fast multiple method (FMM) for kernel matrix.

- ...

Other computations such as matrix-vector multiplication with these "easy" matrices are typically fast too.
Bottom line: Don't blindly use `solve()`.

Linear algebra review: eigen-decomposition

Our last topic on numerical linear algebra is eigen-decomposition and singular value decomposition (SVD). We already saw the wide applications of QR decomposition in least squares problem and solving square and under-determined linear equations. Eigen-decomposition and SVD can be deemed as more thorough orthogonalization of a matrix. We start with a brief review of the related linear algebra.

- *Eigenvalues* are defined as roots of the characteristic equation $\det(\lambda \boldsymbol{I}_n - \boldsymbol{A}) = 0$.

- If λ is an eigenvalue of $\boldsymbol{A}$, then there exist non-zero $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$ such that $\boldsymbol{A}\boldsymbol{x} = \lambda\boldsymbol{x}$ and $\boldsymbol{y}^T\boldsymbol{A} = \lambda\boldsymbol{y}^T$. $\boldsymbol{x}$ and $\boldsymbol{y}$ are called the (column) *eigenvector* and *row eigenvector* of $\boldsymbol{A}$ associated with the eigenvalue λ.

- $\boldsymbol{A}$ is singular if and only if it has at least one 0 eigenvalue.
- Eigenvectors associated with distinct eigenvalues are linearly independent.
- Eigenvalues of an upper or lower triangular matrix are its diagonal entries: $\lambda_i = a_{ii}$.
- Eigenvalues of an idempotent matrix are either 0 or 1.
- Eigenvalues of an orthogonal matrix have complex modulus 1.
- In most statistical applications, we deal with eigenvalues/eigenvectors of symmetric matrices. The eigenvalues and eigenvectors of a real *symmetric* matrix are real.
- Eigenvectors associated with distinct eigenvalues of a symmetry matrix are orthogonal.
- Eigen-decompostion of a symmetric matrix: $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^{\mathsf{T}}$, where
  - $\boldsymbol{\Lambda} = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$
  - columns of $\boldsymbol{U}$ are the eigenvectors, which are (or can be chosen to be) mutually orthonormal. That is $\boldsymbol{U}$ is an orthogonal matrix.
- A real symmetric matrix is positive semidefinite (positive definite) if and only if all eigenvalues are nonnegative (positive).
- *Spectral radius* $\rho(\boldsymbol{A}) = \max_i |\lambda_i|$.
- $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ a square matrix (not required to be symmetric), then $\operatorname{tr}(\boldsymbol{A}) = \sum_i \lambda_i$ and $|\boldsymbol{A}| = \prod_i \lambda_i$.

13 Lecture 13, Oct 14

Announcements

- HW4 due today: hardcopy + email code (`LastFirstHW4.R` or `LastFirstHW4.Rmd`)
- HW5 posted, due next Tue Oct 21

Last time

- A list of "easy" linear system
- Linear algebra review: eigen-decomposition

Today

- Linear algebra review: SVD
- Applications of eigen-decomposition and SVD
- Algorithms for eigen-decomposition and SVD

Linear algebra review: SVD

- Singular value decomposition (SVD): For a rectangular matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$, let $p = \min\{m, n\}$, then we have the SVD

  $$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathsf{T}},$$

  where

  - $\boldsymbol{U} = (\boldsymbol{u}_1, \ldots, \boldsymbol{u}_m) \in \mathbb{R}^{m \times m}$ is orthogonal
  - $\boldsymbol{V} = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n) \in \mathbb{R}^{n \times n}$ is orthgonal
  - $\boldsymbol{\Sigma} = \text{diag}(\sigma_1, \ldots, \sigma_p) \in \mathbb{R}^{m \times n}$, $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_p \geq 0$.

  σ_i are called the *singular values*, $\boldsymbol{u}_i$ are the *left singular vectors*, and $\boldsymbol{v}_i$ are the *right singular vectors.*

- Thin SVD. Assume $m \geq n$. $\boldsymbol{A}$ can be factored as

$$\boldsymbol{A} = \boldsymbol{U}_n \boldsymbol{\Sigma}_n \boldsymbol{V}^\intercal = \sum_{i=1}^{n} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T,$$

  where

  - $\boldsymbol{U}_n \in \mathbb{R}^{m \times n}$, $\boldsymbol{U}_n^\intercal \boldsymbol{U}_n = \boldsymbol{I}_n$
  - $\boldsymbol{V} \in \mathbb{R}^{n \times n}$, $\boldsymbol{V}^\intercal \boldsymbol{V} = \boldsymbol{I}_n$
  - $\boldsymbol{\Sigma}_n = \text{diag}(\sigma_1, \ldots, \sigma_n) \in \mathbb{R}^{n \times n}$, $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$

- Denote $\sigma(\boldsymbol{A}) = (\sigma_1, \ldots, \sigma_p)^T$. Then

  - $r = \text{rank}(\boldsymbol{A}) = \#$ nonzero singular values $= \|\sigma(\boldsymbol{A})\|_0$
  - $\boldsymbol{A} = \boldsymbol{U}_r \boldsymbol{\Sigma}_r \boldsymbol{V}_r^T = \sum_{i=1}^{r} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T$
  - $\|\boldsymbol{A}\|_{\text{F}} = (\sum_{i=1}^{p} \sigma_i^2)^{1/2} = \|\sigma(\boldsymbol{A})\|_2$
  - $\|\boldsymbol{A}\|_2 = \sigma_1 = \|\sigma(\boldsymbol{A})\|_\infty$

- Assume $\text{rank}(\boldsymbol{A}) = r$ and partition $\boldsymbol{U} = (\boldsymbol{U}_r, \tilde{\boldsymbol{U}}_r) \in \mathbb{R}^{m \times m}$ and $\boldsymbol{V} = (\boldsymbol{V}_r, \tilde{\boldsymbol{V}}_r) \in \mathbb{R}^{n \times n}$, then

  - $\mathcal{C}(\boldsymbol{A}) = \mathcal{C}(\boldsymbol{U}_r)$, $\mathcal{N}(\boldsymbol{A}^T) = \mathcal{C}(\tilde{\boldsymbol{U}}_r)$
  - $\mathcal{N}(\boldsymbol{A}) = \mathcal{C}(\tilde{\boldsymbol{V}}_r)$, $\mathcal{C}(\boldsymbol{A}^T) = \mathcal{C}(\boldsymbol{V}_r)$
  - $\boldsymbol{U}_r \boldsymbol{U}_r^T$ is the orthogonal projection onto $\mathcal{C}(\boldsymbol{A})$
  - $\tilde{\boldsymbol{U}}_r \tilde{\boldsymbol{U}}_r^T$ is the orthogonal projection onto $\mathcal{N}(\boldsymbol{A}^T)$
  - $\boldsymbol{V}_r \boldsymbol{V}_r^T$ is the orthogonal projection onto $\mathcal{C}(\boldsymbol{A}^T)$
  - $\tilde{\boldsymbol{V}}_r \tilde{\boldsymbol{V}}_r^T$ is the orthogonal projection onto $\mathcal{N}(\boldsymbol{A})$

- Relation to eigen-decomposition. Using thin SVD,

$$\begin{aligned} \boldsymbol{A}^\intercal \boldsymbol{A} &= \boldsymbol{V} \boldsymbol{\Sigma} \boldsymbol{U}^\intercal \boldsymbol{U} \boldsymbol{\Sigma} \boldsymbol{V}^\intercal = \boldsymbol{V} \boldsymbol{\Sigma}^2 \boldsymbol{V}^\intercal \\ \boldsymbol{A} \boldsymbol{A}^\intercal &= \boldsymbol{U} \boldsymbol{\Sigma} \boldsymbol{V}^\intercal \boldsymbol{V} \boldsymbol{\Sigma} \boldsymbol{U}^\intercal = \boldsymbol{U} \boldsymbol{\Sigma}^2 \boldsymbol{U}^\intercal. \end{aligned}$$

- Another relation to eigen-decomposition. Using thin SVD,

$$\begin{pmatrix} \boldsymbol{0}_{n \times n} & \boldsymbol{A}^\intercal \\ \boldsymbol{A} & \boldsymbol{0}_{m \times m} \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} \boldsymbol{V} & \boldsymbol{V} \\ \boldsymbol{U} & -\boldsymbol{U} \end{pmatrix} \begin{pmatrix} \boldsymbol{\Sigma} & \boldsymbol{0}_{n \times n} \\ \boldsymbol{0}_{n \times n} & -\boldsymbol{\Sigma} \end{pmatrix} \frac{1}{\sqrt{2}} \begin{pmatrix} \boldsymbol{V}^\intercal & \boldsymbol{U}^\intercal \\ \boldsymbol{V}^\intercal & -\boldsymbol{U}^\intercal \end{pmatrix}.$$

Hence any symmetric eigen-solver can produce the SVD of a matrix $\boldsymbol{A}$ without forming $\boldsymbol{A}\boldsymbol{A}^\intercal$ or $\boldsymbol{A}^\intercal\boldsymbol{A}$.

- Yet another relation to eigen-decomposition: If the eigendecomposition of a real symmetric matrix is $\boldsymbol{A} = \boldsymbol{W}\boldsymbol{\Lambda}\boldsymbol{W}^T = \boldsymbol{W}\text{diag}(\lambda_1, \ldots, \lambda_n)\boldsymbol{W}^T$, then

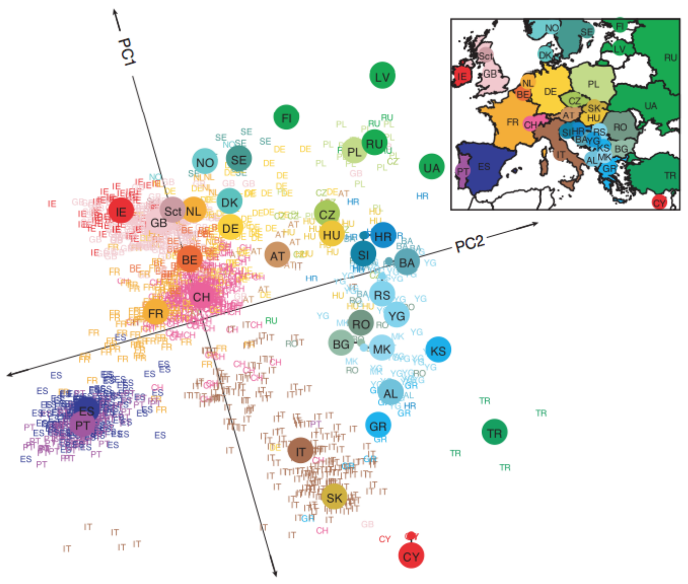
$$\boldsymbol{A} = \boldsymbol{W}\boldsymbol{\Lambda}\boldsymbol{W}^T = \boldsymbol{W}\begin{pmatrix} |\lambda_1| & & \\ & \ddots & \\ & & |\lambda_n| \end{pmatrix}\begin{pmatrix} \text{sgn}(\lambda_1) & & \\ & \ddots & \\ & & \text{sgn}(\lambda_n) \end{pmatrix}\boldsymbol{W}^T$$

is the SVD of $\boldsymbol{A}$.

Applications of eigen-decomposition and SVD

- Principal components analysis (PCA). $\boldsymbol{X} \in \mathbb{R}^{n\times p}$ is a centered data matrix. Perform SVD $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\intercal$ or equivalently $\boldsymbol{X}^\intercal\boldsymbol{X} = \boldsymbol{V}\boldsymbol{\Sigma}^2\boldsymbol{V}^\intercal$. The linear combinations $\tilde{\boldsymbol{x}}_i = \boldsymbol{X}\boldsymbol{v}_i$ are the principal components and have variance σ_i^2. Usages:

  1. Dimension reduction: reduce dimensionality p to $q \ll p$. Use top PCs $\tilde{\boldsymbol{x}}_1, \ldots, \tilde{\boldsymbol{x}}_q$ in downstream analysis.
  2. Use PCs to adjust for confounding – a serious issue in association studies in large data sets.



- Low rank approximation, e..g, image/data compression.
  Eckart-Young theorem:
  $$\min_{\text{rank}(\boldsymbol{Y})=r} \|\boldsymbol{X} - \boldsymbol{Y}\|_{\text{F}}^2$$
  is achieved by $\boldsymbol{Y} = \sum_{i=1}^{r} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^{\mathsf{T}}$ with optimal value $\sum_{i=1}^{r-1} \sigma_i^2$, where $(\sigma_i, \boldsymbol{u}_i, \boldsymbol{v}_i)$ are singular values and vectors of $\boldsymbol{X}$.

  Gene Golub's 2691×598 picture requires $2691 \times 598 \times 6 = 9,655,308$ bytes (RGB 16 bit per channel). Rank 120 approximation requires $120 \times (2691 + 598) \times 6 = 2,368,080$ bytes. Rank 50 approximation requires $50 \times (2691 + 598) \times 6 = 986,700$ bytes. Rank 12 approximation requires $12 \times (2691 + 598) \times 8 = 236,808$ bytes.

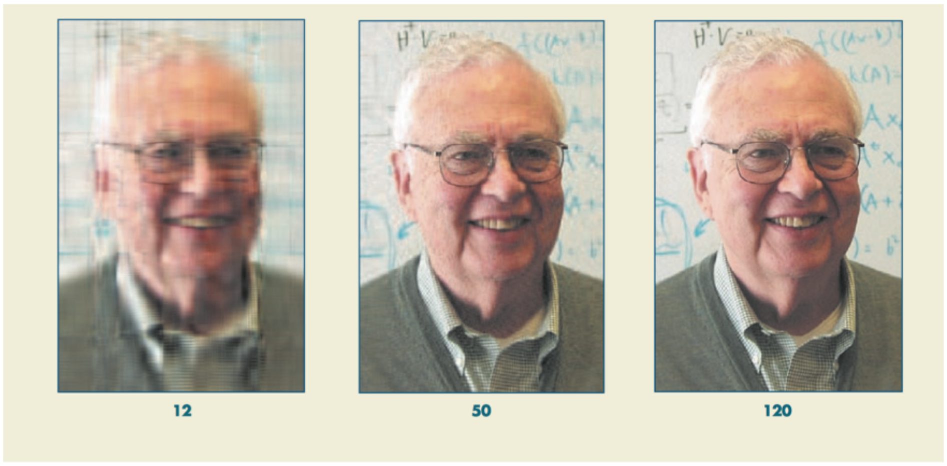


Figure 2. Rank 12, 50, and 120 approximations to a rank 598 color photo of Gene Golub.

- Least squares, ridge regression, least squares over a sphere, ...

- Moore-Penrose (MP) inverse: Using thin SVD,
  $$\boldsymbol{A}^{+} = \boldsymbol{V}\boldsymbol{\Sigma}^{+}\boldsymbol{U}^{\mathsf{T}},$$
  where $\boldsymbol{\Sigma}^{+} = \text{diag}(\sigma_1^{-1}, \ldots, \sigma_r^{-1}, 0, \ldots, 0)$, $r = \text{rank}(\boldsymbol{A})$. This is how the `ginv()` function is implemented in `MASS` package.

- Read KL and JM and do HW5 for some more applications.

One eigen-pair - power method

Assume $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ symmetric.

- *Power method* iterates according to

$$\begin{aligned} \boldsymbol{x}^{(t)} &\leftarrow \frac{1}{\|\boldsymbol{A}\boldsymbol{x}^{(t-1)}\|_2} \boldsymbol{A}\boldsymbol{x}^{(t-1)} \\ \lambda_1^{(t)} &\leftarrow \boldsymbol{x}^{(t)\,\mathsf{T}} \boldsymbol{A}\boldsymbol{x}^{(t)} \end{aligned}$$

  from an initial guess $\boldsymbol{x}^{(0)}$ of unit norm.

- Suppose we arrange $|\lambda_1| > |\lambda_2| \geq \cdots \geq |\lambda_n|$ (the first inequality strict) with corresponding eigenvectors $\boldsymbol{u}_i$, and expand $\boldsymbol{x}^{(0)} = c_1\boldsymbol{u}_1 + \cdots + c_n\boldsymbol{u}_n$, then

$$\begin{aligned} \boldsymbol{x}^{(t)} &= \frac{\left(\sum_i \lambda_i^t \boldsymbol{u}_i \boldsymbol{u}_i^{\mathsf{T}}\right)\left(\sum_i c_i \boldsymbol{u}_i\right)}{\|\left(\sum_i \lambda_i^t \boldsymbol{u}_i \boldsymbol{u}_i^{\mathsf{T}}\right)\left(\sum_i c_i \boldsymbol{u}_i\right)\|_2} \\ &= \frac{\sum_i c_i \lambda_i^t \boldsymbol{u}_i}{\|\sum_i c_i \lambda_i^t \boldsymbol{u}_i\|_2} \\ &= \frac{c_1\boldsymbol{u}_1 + c_2(\lambda_2/\lambda_1)^t \boldsymbol{u}_2 + \cdots + c_n(\lambda_n/\lambda_1)^t \boldsymbol{u}_n}{\|c_1\boldsymbol{u}_1 + c_2(\lambda_2/\lambda_1)^t \boldsymbol{u}_2 + \cdots + c_n(\lambda_n/\lambda_1)^t \boldsymbol{u}_n\|_2} \left(\frac{\lambda_1}{|\lambda_1|}\right)^t. \end{aligned}$$

  Thus $\boldsymbol{x}^{(t)} - \frac{c_1\boldsymbol{u}_1}{\|c_1\boldsymbol{u}_1\|_2}\left(\frac{\lambda_1}{|\lambda_1|}\right)^t \to 0$ as $t \to \infty$. The convergence rate is $|\lambda_2|/|\lambda_1|$.

- $\boldsymbol{x}^{(t)\,\mathsf{T}} \boldsymbol{A}\boldsymbol{x}^{(t)}$ converges to λ_1.

- *Inverse power method* for finding the eigenvalue of smallest absolute value: Substitute $\boldsymbol{A}$ by $\boldsymbol{A}^{-1}$ in the power method. (E.g., pre-compute LU or Cholesky of $\boldsymbol{A}$).

- *Shifted inverse power*: Substitute $(\boldsymbol{A} - \mu\boldsymbol{I})^{-1}$ in the power method. It converges to an eigenvalue close to the given μ.

- Initial guess of the desired eigenvalue can be obtain by Gerschgorin's circle theorem and so on.

- Power method also applies to asymmetric $\boldsymbol{A}$, e.g., PageRank problem costs $O(n)$ per iteration.

Top r eigen-pairs - orthogonal iteration

Generalization of power method to higher dimensional invariant subspace.

- *Orthogonal iteration*: Initialize $\boldsymbol{Q}^{(0)} \in \mathbb{R}^{n \times r}$ with orthonormal columns. For $t = 1, 2, \ldots$,

$$\begin{aligned} \boldsymbol{Z}^{(t)} &\leftarrow \boldsymbol{A}\boldsymbol{Q}^{(t-1)} \quad (n^2 r \text{ flops}) \\ \boldsymbol{Q}^{(t)}\boldsymbol{R}^{(t)} &\leftarrow \boldsymbol{Z}^{(t)} \quad (\text{QR factorization, } nr^2 - r^3/3 \text{ flops}) \end{aligned}$$

- $\boldsymbol{Z}^{(t)}$ converges to the eigenspace of the largest r eigenvalues if they are real and separated from remaining spectrum. The convergence rate is $|\lambda_{r+1}|/|\lambda_r|$.

(Impractical) full eigen-decomposition - QR iteration

Assume $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ symmetric.

- take $r = n$ in the orthogonal iteration. Then $\boldsymbol{Q}^{(t)}$ converges to the eigenspace $\boldsymbol{U}$ of $\boldsymbol{A}$. This implies that

$$\boldsymbol{T}^{(t)} := \boldsymbol{Q}^{(t)\,T}\boldsymbol{A}\boldsymbol{Q}^{(t)}$$

  converges to a diagonal form $\boldsymbol{\Lambda} = \text{diag}(\lambda_1, \ldots, \lambda_n)$.

- Note how to compute $\boldsymbol{T}^{(t)}$ from $\boldsymbol{T}^{(t-1)}$

$$\begin{aligned} \boldsymbol{T}^{(t-1)} &= \boldsymbol{Q}^{(t-1)\,T}\boldsymbol{A}\boldsymbol{Q}^{(t-1)} = \boldsymbol{Q}^{(t-1)\,T}(\boldsymbol{A}\boldsymbol{Q}^{(t-1)}) = (\boldsymbol{Q}^{(t-1)\,T}\boldsymbol{Q}^{(t)})\boldsymbol{R}^{(t)} \\ \boldsymbol{T}^{(t)} &= \boldsymbol{Q}^{(t)\,T}\boldsymbol{A}\boldsymbol{Q}^{(t)} = \boldsymbol{Q}^{(t)\,T}\boldsymbol{A}\boldsymbol{Q}^{(t-1)}\boldsymbol{Q}^{(t-1)\,\intercal}\boldsymbol{Q}^{(t)} = \boldsymbol{R}^{(t)}(\boldsymbol{Q}^{(t-1)\,\intercal}\boldsymbol{Q}^{(t)}). \end{aligned}$$

- *QR iteration*: Initialize $\boldsymbol{U}^{(0)} \in \mathbb{R}^{n \times n}$ orthogonal and set $\boldsymbol{T}^{(0)} = \boldsymbol{U}^{(0)\,T}\boldsymbol{A}\boldsymbol{U}^{(0)}$. For $t = 1, 2, \ldots$

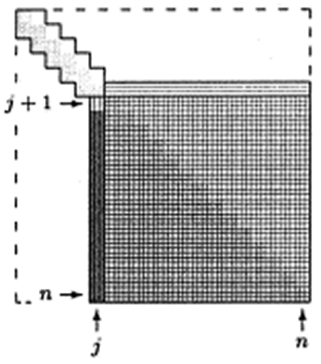
$$\begin{aligned} \boldsymbol{U}^{(t)}\boldsymbol{R}^{(t)} &\leftarrow \boldsymbol{T}^{(t-1)} \quad (\text{QR factorization}) \\ \boldsymbol{T}^{(t)} &\leftarrow \boldsymbol{R}^{(t)}\boldsymbol{U}^{(t)} \end{aligned}$$

- QR iteration is expensive: $O(n^3)$ per iteration and linear convergence rate.

QR algorithm for symmetric eigen-decomposition

Assume $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ symmetric.

- As done in LAPACK: `eigen()` in `R`, `eig()` in `Matlab`
- Idea: Tri-diagonalization (by Householder) + QR iteration on the tri-diagonal system with implicit shift
  - Step 1: Householder tri-diagonalization: $4n^3/3$ for eigenvalues only, $8n^3/3$ for both eigenvalues and eigenvectors. (Why can't we apply Householder to make it diagonal directly?)



  - Step 2: QR iteration on the tridiagonal matrix. Implicit shift accelerates convergence rate. On average 1.3-1.6 QR iteration per eigenvalue, $\sim 20n$ flops per QR iteration. So total operation count is about $30n^2$. Eigenvectors need an extra of about $6n^3$ flops.

	Eigenvalue	Eigenvector
Householder reduction	$4n^3/3$	$4n^3/3$
QR with implicit shift	$\sim 30n^2$	$\sim 6n^3$

- Don't request eigenvectors unless necessary: set `only.values = TRUE` when calling `eigen()` in `R`.
- Remark: there are at least two alternative ways (other than QR with implicit shift) to solve the symmetric tridiagonal eigenproblem

  - Bisection: good when only selected portions of eigensystem are needed
  - Divide and conquer: good for parallel implementation

- The *unsymmetric QR algorithm* obtains the real Schur decomposition of an asymmetric matrix $\boldsymbol{A}$.

14 Lecture 14, Oct 16

Announcements

- HW4 returned. Feedback:
  - Key to the success of iterative methods is fast matrix-vector multiplication. Here the structure in $\boldsymbol{P}^T$ or $\boldsymbol{I} - \boldsymbol{P}^T$ is "sparse + low rank". Say in power method, if we first form the matrix $\boldsymbol{P}^T$ and then iterate according to $\boldsymbol{x}^{(t)} \leftarrow \boldsymbol{P}^T \boldsymbol{x}^{(t-1)}$, it costs n^2 flops per iteration. If we keep the "sparse + low rank" structure and iterate according to

    $$\boldsymbol{x}^{(t)} \leftarrow \boldsymbol{A}^T(\text{diag}(\boldsymbol{r}^+)\boldsymbol{x}^{(t-1)}) + \mathbf{1}_n(\boldsymbol{z}^T\boldsymbol{x}^{(t-1)}),$$

    it costs $3n + \text{nnz}(\boldsymbol{A}) \ll n^2$ flops, where $\text{nnz}(\boldsymbol{A})$ is the number of non-zero elements in $\boldsymbol{A}$.
  - Sketch of solution: `http://hua-zhou.github.io/teaching/st758-2014fall/hw04sol.html`
  - Some run times in Matlab: using vs ignoring "sparse + low rank" structure.

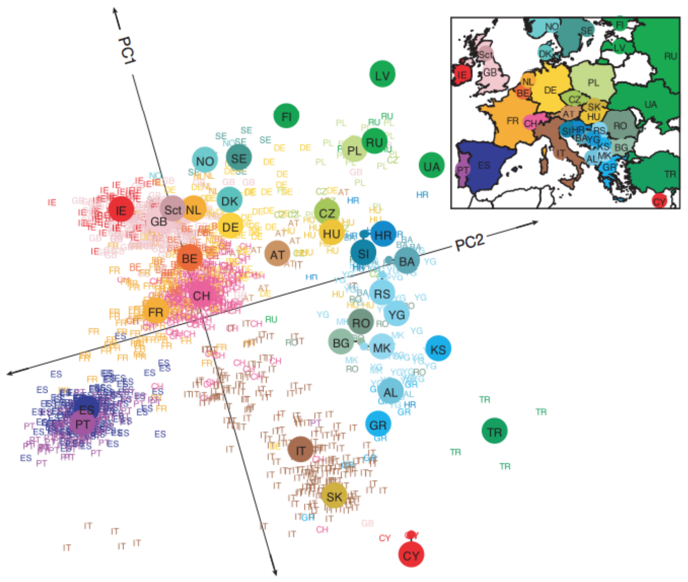
Matrix	Size	Sparsity	`svt` (fh input)	`svds`
bfwb398	398	0.9816	0.0176(0.0011)	0.0408(0.0009)
rdb800l	800	0.9928	0.0240(0.0005)	0.2115(0.0014)
tols1090	1090	0.9970	0.0780(0.0009)	0.9396(0.0079)
mhd4800b	4800	0.9988	0.0471(0.0001)	5.6700(0.0166)
cryg10000	10000	0.9995	0.1909(0.0022)	44.2213(0.4373)

Table 3: Top 6 singular values and vectors of "sparse + low rank" matrices by `svt` and `svds`. Structured matrices are formed by adding a random rank-10 matrix to the original sparse test matrix. Reported are the average run time (in seconds) and standard error (in parentheses) based on 10 simulation replicates.

- HW5 due next Tue Oct 21.

Last time

- Linear algebra review: SVD
- Applications of eigen-decomposition and SVD:
  PCA, data/image compression, least squares and its variants, MP inverse, stationary distribution of Markov chains (Google PageRank), ...
- Answer to Caleb's question: Following picture is from the article "Genes mirror geography within Europe" by Novembre et al. (2008) published in *Nature* `http://www.nature.com/nature/journal/v456/n7218/full/nature07331.html`.



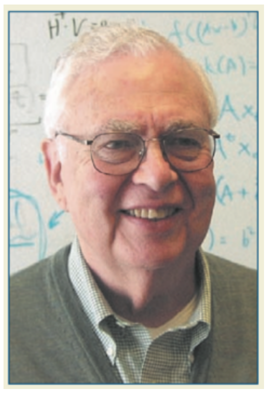
  Use of PCA to adjust for confounding in modern genetic studies is proposed in the paper "Principal components analysis corrects for stratification in genome-wide association studies" by Price et al. (2006) published in *Nature Genetics* `http://www.nature.com/ng/journal/v38/n8/full/ng1847.html`. It has been cited 3346 times as of Oct 14, 2014.

- Algorithms for eigen-decomposition:
  power method (top eigen-pair), orthogonal iteration (top r eigen-pairs), QR iteration for full eigen-decomposition ($O(n^4)$ flops), Householder tri-diagonalization + QR iteration with implicit shift ($O(n^3)$ flops).

Today

- SVD algorithm

Algorithm for singular value decomposition (SVD)



Gene Golub's license plate, photographed by Professor P. M. Kroonenberg of Leiden University.

Assume $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ and we seek the SVD $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^{\top}$.

- "Golub-Kahan-Reinsch" algorithm:
  - Stage 1: Transform $\boldsymbol{A}$ to an upper bidiagonal form $\boldsymbol{B}$ (by Householder).

$$U_B^T A V_B = \begin{bmatrix} B \\ 0 \end{bmatrix} \qquad B = \begin{bmatrix} d_1 & f_1 & & \cdots & 0 \\ 0 & d_2 & \ddots & & \vdots \\ & \ddots & \ddots & \ddots & \\ \vdots & & \ddots & \ddots & f_{n-1} \\ 0 & \cdots & & 0 & d_n \end{bmatrix} \in \mathbb{R}^{n \times n}$$

$$\begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \end{bmatrix} \xrightarrow{U_1} \begin{bmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \end{bmatrix} \xrightarrow{V_1}$$

$$\begin{bmatrix} \times & \times & 0 & 0 \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \end{bmatrix} \xrightarrow{U_2} \begin{bmatrix} \times & \times & 0 & 0 \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & \times & \times \end{bmatrix} \xrightarrow{V_2}$$

$$\begin{bmatrix} \times & \times & 0 & 0 \\ 0 & \times & \times & 0 \\ 0 & 0 & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & \times & \times \end{bmatrix} \xrightarrow{U_3} \begin{bmatrix} \times & \times & 0 & 0 \\ 0 & \times & \times & 0 \\ 0 & 0 & \times & \times \\ 0 & 0 & 0 & \times \\ 0 & 0 & 0 & \times \end{bmatrix} \xrightarrow{U_4} \begin{bmatrix} \times & \times & 0 & 0 \\ 0 & \times & \times & 0 \\ 0 & 0 & \times & \times \\ 0 & 0 & 0 & \times \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

  – Stage 2: Apply implicit-shift QR step to the tridiagonal matrix $\boldsymbol{B}^{\mathsf{T}}\boldsymbol{B}$ *implicitly.*

- See Golub and Van Loan (1996, Section 8.6) for more details.
- $4m^2n + 8mn^2 + 9n^3$ flops for a tall ($m \geq n$) matrix.
- `svd()` in `R` and `Matlab`: wrapper of the `_GESVD` subroutine in LAPACK.

15 Lecture 15, Oct 21

Announcements

- HW5 due today: submit hardcopy + email code
- Homework due date change to Thu?
- Spaces around "=" in function call?

Last time

- HW4 (PageRank algorithm). Ilse Ipsen's slides
  `http://www4.ncsu.edu/~ipsen/ps/slides_imacs.pdf`
- SVD (Golub-Kahan-Riensch) algorithm

Today

- Iterative methods (Lanszos and Arnoldi methods) for huge, structured $\boldsymbol{A}$
- Jacobi method for eigen-decomposition (parallel computing)
- Generalized eigen-problem: $\boldsymbol{Ax} = \lambda \boldsymbol{Bx}$ (PLS, SIR, CCA, ...)
- Variants of least squares problems (self-study)
- Concluding remarks for numerical linear algebra

Lanczos/Arnoldi iterative method for top eigen-pairs

- Motivation
  - Consider the Google PageRank problem. We want to find the top left eigenvector of the transition matrix

$$\boldsymbol{P} = p\boldsymbol{R}^{+}\boldsymbol{A} + \boldsymbol{z}\mathbf{1}_n^{\mathsf{T}},$$

    where $\boldsymbol{R} = \text{diag}(r_1, \ldots, r_n)$ and $z_j = (1-p)/n$ if $r_i > 0$ and $1/n$ if $r_i = 0$. Suppose there are $n \approx 1$ billion web pages.

The (unsymmetric) QR algorithm will take order

$$\frac{(1 \times 10^9)^3}{10^{12}} \approx 3.33 \times 10^{14} \text{ seconds } \approx 1 \times 10^7 \text{ years}$$

on a tera-flop supercomputer!

  - Consider adjusting for confounding by PCA in modern GWAS (genome-wide association studies). We want to find the top singular values/vectors of a genotype matrix $\boldsymbol{X} \in \mathbb{R}^{n \times p}$, where $n \sim 10^3$ and $p \sim 10^6$.

- Krylov subspace methods are the state-of-art iterative method for obtaining the top eigen-values/vectors or singular values/vectors of large sparse or structured matrices.

- Lanczos method: top eigen-pairs of a large *symmetric* matrix.

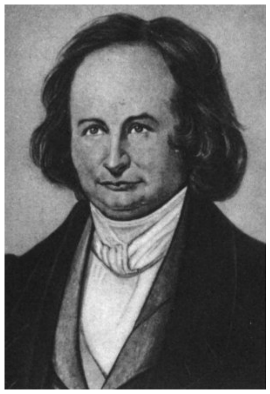
- Arnoldi method: top eigen-pairs of a large *asymmetric* matrix.

- Both methods are also adapted to obtain top singular values/vectors of large sparse or structured matrices.

- We will give a brief overview of these methods together with the conjugate gradient method for solving large linear system.

- `eigs()` and `svds()` in `Matlab` are wrappers of the `ARPACK` package. No native functions in R (?).

Jacobi method for symmetric eigen-decomposition (KL 8.2)



Assume $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ is symmetric and we seek the eigen-decomposition $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T$.

- Idea: Systematically reduce off-diagonal entries

$$\operatorname{off}(\boldsymbol{A}) = \sum_i \sum_{j\neq i} a_{ij}^2$$

  by Jacobi rotations.

- Jacobi/Givens rotations:

$$\boldsymbol{J}(p,q,\theta) = \begin{pmatrix} 1 & & 0 & & 0 & & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & & \cos(\theta) & & \sin(\theta) & & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & & -\sin(\theta) & & \cos(\theta) & & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & & 0 & & 0 & & 1 \end{pmatrix},$$

  $\boldsymbol{J}(p,q,\theta)$ is orthogonal.

- Consider $\boldsymbol{B} = \boldsymbol{J}^{\mathsf{T}}\boldsymbol{A}\boldsymbol{J}$. $\boldsymbol{B}$ preserves the symmetry and eigenvalues of $\boldsymbol{A}$. Taking

$$\begin{cases} \tan(2\theta) = 2a_{pq}/(a_{qq} - a_{pp}) & \text{if } a_{pp} \neq a_{qq} \\ \theta = \pi/4 & \text{if } a_{pp} = a_{qq} \end{cases}$$

  forces $b_{pq} = 0$.

- Since orthogonal transform preserves Frobenius norm, we have

$$b_{pp}^2 + b_{qq}^2 = a_{pp}^2 + a_{qq}^2 + 2a_{pq}^2.$$

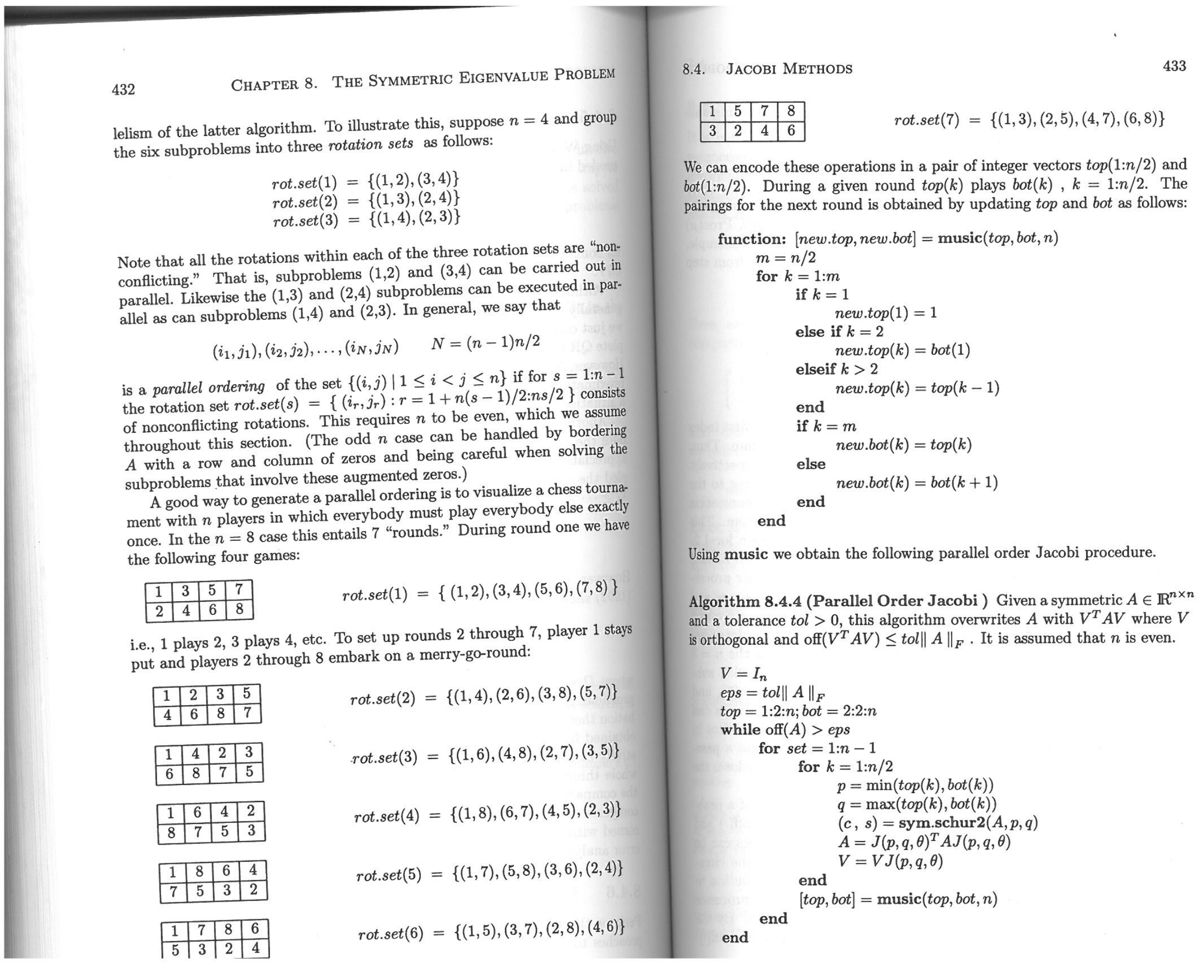
  (Just check the 2-by-2 block)

- Since $\|\boldsymbol{A}\|_{\mathrm{F}} = \|\boldsymbol{B}\|_{\mathrm{F}}$, this implies that the off-diagonal part

$$\operatorname{off}(\boldsymbol{B}) = \operatorname{off}(\boldsymbol{A}) - 2a_{pq}^2$$

  is decreased whenever $a_{pq} \neq 0$.

- One Jacobi rotation costs $O(n)$ flops.
- *Classical Jacobi*: search for the largest $|a_{ij}|$ at each iteration.
- $\text{off}(\boldsymbol{A}) \le n(n-1)a_{ij}^2$ and $\text{off}(\boldsymbol{B}) = \text{off}(\boldsymbol{A}) - 2a_{ij}^2$ together implies
$$\text{off}(\boldsymbol{B}) \le \left(1 - \frac{2}{n(n-1)}\right)\text{off}(\boldsymbol{A}).$$
- In practice, cyclic-by-row implementation, to avoid the costly $O(n^2)$ search in the classical Jacobi.
- Jacobi method attracts a lot recent attention because of its rich inherent parallelism.
- *Parallel Jacobi*: "merry-go-round" to generate parallel ordering.



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 $\{(i,j)\,|\,1 \le i < j \le n\}$ 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

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

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: $[new.top, new.bot] = \textbf{music}(top, bot, n)$
    $m = n/2$
    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^TAV where V is orthogonal and $\text{off}(V^TAV) \le tol\|\,A\,\|_F$. It is assumed that n is even.

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

Generalized eigen-problem

- Generalized eigen-problem: $\boldsymbol{A}\boldsymbol{x} = \lambda \boldsymbol{B}\boldsymbol{x}$, where $\boldsymbol{A}$ psd and $\boldsymbol{B}$ pd.
- Applications: partial least squares (PLS), sliced inverse regression (SIR), canonical correlation analysis (CCA).
- Method 1: $\boldsymbol{B}^{-1}\boldsymbol{A}\boldsymbol{x} = \lambda \boldsymbol{x}$. Non-symmetric eigen-problem ☹.
- Method 2: Cholesky $\boldsymbol{B} = \boldsymbol{L}\boldsymbol{L}^{\mathsf{T}}$. Then $\boldsymbol{L}^{-1}\boldsymbol{A}\boldsymbol{L}^{-T}\boldsymbol{y} = \lambda \boldsymbol{y}$ where $\boldsymbol{y} = \boldsymbol{L}^{\mathsf{T}}\boldsymbol{x}$.
- Method 3 (most numerically stable, $\boldsymbol{B}$ can be rank deficient): QZ algorithm.
- `eig()` and `qz()` in `Matlab` implement QZ. No native function in `R`?

Generalized singular value decomposition

- $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ and $\boldsymbol{B} \in \mathbb{R}^{p\times n}$. Then there exists orthogonal $\boldsymbol{U} \in \mathbb{R}^{m\times m}$ and $\boldsymbol{V} \in \mathbb{R}^{p\times p}$ and an invertible $\boldsymbol{X} \in \mathbb{R}^{n\times n}$ such that

$$\begin{aligned} \boldsymbol{U}^T\boldsymbol{A}\boldsymbol{X} &= \boldsymbol{C} = \operatorname{diag}(c_1,\ldots,c_n), \quad c_i \ge 0 \\ \boldsymbol{V}^T\boldsymbol{B}\boldsymbol{X} &= \boldsymbol{S} = \operatorname{diag}(s_1,\ldots,s_q), \quad s_i \ge 0, \end{aligned}$$

  where $q = \min\{p, n\}$.
- Applications: quadratically inequality-constrained least squares problem (LSQI).
- `gsvd()` in `Matlab` implements generalized SVD. No native function in `R`?

In the zoo of least squares (self-study)

Weighted least squares

- In weighted least squares, we minimize $\sum_{i=1}^n w_i(y_i - \boldsymbol{x}_i^T\boldsymbol{\beta})^2$, where $w_i > 0$ are observation weights.
- Let $\boldsymbol{W} = \operatorname{diag}(w_1,\ldots,w_n)$. Then the criterion is $\|\boldsymbol{W}^{1/2}\boldsymbol{y} - \boldsymbol{W}^{1/2}\boldsymbol{X}\boldsymbol{\beta}\|_2^2$, which can be solved by standard methods for least squares with $\tilde{\boldsymbol{y}} = \boldsymbol{W}^{1/2}\boldsymbol{y}$ and $\tilde{\boldsymbol{X}} = \boldsymbol{W}^{1/2}\boldsymbol{X}$.

General least squares

- In Aitken model: $E(\boldsymbol{y}) = \boldsymbol{X}\boldsymbol{\beta}$, $\text{Cov}(\boldsymbol{y}) = \sigma^2 \boldsymbol{V}$, where $\boldsymbol{V}$ is a positive semidefinite matrix. We minimize the generalized least squares criterion

$$(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})^T \boldsymbol{M}^- (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}),$$

where $\boldsymbol{M} \in \mathbb{R}^{n \times n}$ is some positive semidefinite matrix, e.g., $\boldsymbol{M} = \boldsymbol{V}$ for nonsingular $\boldsymbol{V}$ or $\boldsymbol{M} = \boldsymbol{V} + \boldsymbol{X}\boldsymbol{X}^T$ for singular $\boldsymbol{V}$.

- Let $\boldsymbol{M} = \boldsymbol{B}\boldsymbol{B}^T$ for some $\boldsymbol{B} \in \mathbb{R}^{n \times n}$ (e.g., the Cholesky factor). One approach is to minimize

$$\|\boldsymbol{B}^{-1}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})\|_2^2.$$

Unfortunately, when $\boldsymbol{B}$ is poorly conditioned (or even not invertible), the procedure produces a poor solution.

- Paige's method. The generalized least squares problem is equivalent to

$$\begin{aligned} \text{minimize} \quad & \boldsymbol{v}^T \boldsymbol{v} \\ \text{subject to} \quad & \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{B}\boldsymbol{v} = \boldsymbol{y}. \end{aligned}$$

To solve this problem, first compute the QR of $\boldsymbol{X}$

$$\boldsymbol{X} = (\boldsymbol{Q}_1, \boldsymbol{Q}_2) \begin{pmatrix} \boldsymbol{R}_1 \\ \boldsymbol{0} \end{pmatrix}.$$

Compute another QR for the (flat) matrix $\boldsymbol{Q}_2^T \boldsymbol{B}$ such that

$$\boldsymbol{Q}_2^T \boldsymbol{B} = (\boldsymbol{0}, \boldsymbol{S}) \begin{pmatrix} \boldsymbol{Z}_1^T \\ \boldsymbol{Z}_2^T \end{pmatrix},$$

where $\boldsymbol{S}$ is upper triangular and $(\boldsymbol{Z}_1, \boldsymbol{Z}_2) \in \mathbb{R}^{n \times n}$ is orthogonal. Then the constraint becomes

$$\begin{pmatrix} \boldsymbol{R}_1 \\ \boldsymbol{0} \end{pmatrix} \boldsymbol{\beta} + \begin{pmatrix} \boldsymbol{Q}_1^T \boldsymbol{B} \boldsymbol{Z}_1 & \boldsymbol{Q}_1^T \boldsymbol{B} \boldsymbol{Z}_2 \\ \boldsymbol{0} & \boldsymbol{S} \end{pmatrix} \begin{pmatrix} \boldsymbol{Z}_1^T \boldsymbol{v} \\ \boldsymbol{Z}_2^T \boldsymbol{v} \end{pmatrix} = \begin{pmatrix} \boldsymbol{Q}_1^T \boldsymbol{y} \\ \boldsymbol{Q}_2^T \boldsymbol{y} \end{pmatrix}.$$

From the bottom half we can solve for $\boldsymbol{v}$ from the equation (how?)

$$\boldsymbol{S}\boldsymbol{Z}_2^T \boldsymbol{v} = \boldsymbol{Q}_2^T \boldsymbol{y}.$$

Then we solve for $\boldsymbol{\beta}$ from the equation

$$\boldsymbol{R}_1 \boldsymbol{\beta} = \boldsymbol{Q}_1^T \boldsymbol{y} - (\boldsymbol{Q}_1^T \boldsymbol{B} \boldsymbol{Z}_1 \boldsymbol{Z}_1^T + \boldsymbol{Q}_1^T \boldsymbol{B} \boldsymbol{Z}_2 \boldsymbol{Z}_2^T) \boldsymbol{v} = \boldsymbol{Q}_1^T \boldsymbol{y} - \boldsymbol{Q}_1^T \boldsymbol{B} \boldsymbol{Z}_2 (\boldsymbol{Z}_2^T \boldsymbol{v}).$$

- Paige's method also works for singular $\boldsymbol{X}$ and $\boldsymbol{B}$ (using QR with column pivoting).
- MATLAB's `lscov()` function implements Paige's method for singular covariance $\boldsymbol{V}$. No R implementation (?)

Ridge regression

- In ridge regression, we minimize
$$\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_2^2,$$
where λ is a tuning parameter.
- Ridge regression by augmented linear regression. Ridge regression problem is equivalent to
$$\left\| \begin{pmatrix} \boldsymbol{y} \\ \boldsymbol{0}_p \end{pmatrix} - \begin{pmatrix} \boldsymbol{X} \\ \sqrt{\lambda}\boldsymbol{I}_p \end{pmatrix} \boldsymbol{\beta} \right\|_2^2 .$$
Therefore any methods for linear regression can be applied.
- Ridge regression by method of normal equation. The normal equation for the ridge problem is
$$(\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I}_p)\boldsymbol{\beta} = \boldsymbol{X}^T\boldsymbol{y}.$$
Therefore Cholesky or sweep can be used.
- Ridge regression by SVD. If we obtain the (thin) SVD of $\boldsymbol{X}$
$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}_{p\times p}\boldsymbol{V}^T.$$
Then the normal equation reads
$$(\boldsymbol{\Sigma}^2 + \lambda\boldsymbol{I}_p)\boldsymbol{V}^T\boldsymbol{\beta} = \boldsymbol{\Sigma}\boldsymbol{U}^T\boldsymbol{y}$$
and we get
$$\widehat{\boldsymbol{\beta}}(\lambda) = \sum_{i=1}^{p} \frac{\sigma_i \boldsymbol{u}_i^T\boldsymbol{y}}{\sigma_i^2 + \lambda}\boldsymbol{v}_i = \sum_{i=1}^{r} \frac{\sigma_i \boldsymbol{u}_i^T\boldsymbol{y}}{\sigma_i^2 + \lambda}\boldsymbol{v}_i, \quad r = \operatorname{rank}(\boldsymbol{X}).$$

It is clear that

$$\lim_{\lambda\to 0} \widehat{\boldsymbol{\beta}}(\lambda) = \widehat{\boldsymbol{\beta}}_{\text{OLS}}$$

and $\|\widehat{\boldsymbol{\beta}}(\lambda)\|_2$ is monotone decreasing as λ increases.

- Only one SVD is needed for all λ (!), in contrast to the method of augmented linear regression, Cholesky, or sweep.

Least squares over a sphere

- Ridge regression "shrinks" the solution via penalty. Alternatively we can simply fit a least squares problem subject to the constraint that the solution lives in a sphere

$$\begin{aligned} \text{minimize} \quad & \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 \\ \text{subject to} \quad & \|\boldsymbol{\beta}\|_2 \le \alpha. \end{aligned}$$

- Suppose we obtain the (thin) SVD $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}_{p\times p}\boldsymbol{V}^T$. If the ordinary least squares solution

$$\widehat{\boldsymbol{\beta}}_{\text{OLS}} = \sum_{i=1}^{r} \frac{\boldsymbol{u}_i^T \boldsymbol{y}}{\sigma_i} \boldsymbol{v}_i$$

has ℓ_2 norm less than α, then we are done. If not, we use the method of Lagrangian multipliers

$$\psi(\boldsymbol{\beta}, \lambda) = \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 + \frac{\lambda}{2}(\|\boldsymbol{\beta}\|_2^2 - \alpha^2).$$

Setting the gradient to 0, we have the shifted normal equation

$$(\boldsymbol{X}^T\boldsymbol{X} + \lambda \boldsymbol{I}_p)\boldsymbol{\beta} = \boldsymbol{X}^T\boldsymbol{y},$$

which has solution

$$\widehat{\boldsymbol{\beta}}(\lambda) = \sum_{i=1}^{r} \frac{\sigma_i \boldsymbol{u}_i^T \boldsymbol{y}}{\sigma_i^2 + \lambda} \boldsymbol{v}_i.$$

We need to choose the λ such that $\|\widehat{\boldsymbol{\beta}}(\lambda)\|_2 = \alpha$. That is we need to find the (unique) zero of the function

$$f(\lambda) = \|\widehat{\boldsymbol{\beta}}(\lambda)\|_2^2 - \alpha^2 = \sum_{i=1}^{r} \left(\frac{\sigma_i \boldsymbol{u}_i^T \boldsymbol{y}}{\sigma_i^2 + \lambda} \right)^2 - \alpha^2.$$

This is easily achieved by Newton's or other methods.

Least squares with equality constraints

- In many applications, there are *a priori* constraints on the regression parameters. Let's consider how to solve linear regression with equality constraints (LSE)

$$\begin{aligned} \text{minimize} \quad & \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 \\ \text{subject to} \quad & \boldsymbol{B}\boldsymbol{\beta} = \boldsymbol{d}. \end{aligned}$$

- LSE by QR. First compute QR of $\boldsymbol{B}^T \in \mathbb{R}^{p \times m}$

$$\boldsymbol{B}^T = \boldsymbol{Q} \begin{pmatrix} \boldsymbol{R} \\ \boldsymbol{0} \end{pmatrix}$$

  and set

$$\boldsymbol{X}\boldsymbol{Q} = (\boldsymbol{X}_1, \boldsymbol{X}_2) \quad \text{and} \quad \boldsymbol{Q}^T\boldsymbol{\beta} = \begin{pmatrix} \boldsymbol{\beta}_1 \\ \boldsymbol{\beta}_2 \end{pmatrix}.$$

  Then the original minimization problem becomes

$$\begin{aligned} \text{minimize} \quad & \|\boldsymbol{y} - \boldsymbol{X}_1\boldsymbol{\beta}_1 - \boldsymbol{X}_2\boldsymbol{\beta}_2\|_2^2 \\ \text{subject to} \quad & \boldsymbol{R}^T\boldsymbol{\beta}_1 = \boldsymbol{d}. \end{aligned}$$

  Now $\boldsymbol{\beta}_1$ is determined from the constraint $\boldsymbol{R}^T\boldsymbol{\beta}_1 = \boldsymbol{d}$ and $\boldsymbol{\beta}_2$ is solved from the unconstrained least squares problem

$$\text{minimize} \quad \|(\boldsymbol{y} - \boldsymbol{X}_1\boldsymbol{\beta}_1) - \boldsymbol{X}_2\boldsymbol{\beta}_2\|_2^2.$$

  Finally we recover the solution from

$$\boldsymbol{\beta} = \boldsymbol{Q} \begin{pmatrix} \boldsymbol{\beta}_1 \\ \boldsymbol{\beta}_2 \end{pmatrix}.$$

- LSE by augmented system. Define the Lagrangian function

$$\phi(\boldsymbol{\beta}, \boldsymbol{\lambda}) = \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 - \boldsymbol{\lambda}^T(\boldsymbol{B}\boldsymbol{\beta} - \boldsymbol{d}).$$

  Setting gradient to zero yields

$$\begin{aligned} \boldsymbol{X}^T\boldsymbol{X}\boldsymbol{\beta} - \boldsymbol{B}^T\boldsymbol{\lambda} &= \boldsymbol{X}^T\boldsymbol{y} \\ \boldsymbol{B}\boldsymbol{\beta} &= \boldsymbol{d}, \end{aligned}$$

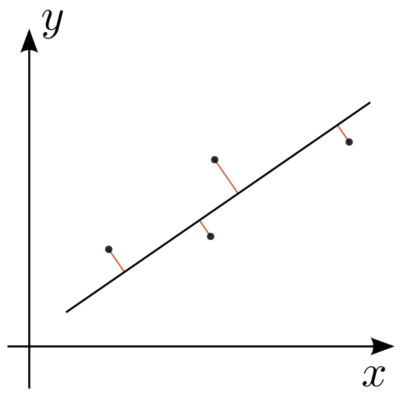
suggesting the augmented system

$$\begin{pmatrix} \boldsymbol{X}^T\boldsymbol{X} & \boldsymbol{B}^T \\ \boldsymbol{B} & \boldsymbol{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{\beta} \\ -\boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} \boldsymbol{X}^T\boldsymbol{y} \\ \boldsymbol{d} \end{pmatrix}.$$

This linear system is non-singular when $\boldsymbol{X}$ and $\boldsymbol{B}$ have full rank and can be solved by Cholesky, sweep, and so on.

- LSE by generalized SVD.

Total least squares (TLS)



TLS considers the case both predictors and observations are subject to errors. It is solved by SVD. Read KL 9.3.6 if interested.

Tikhonov regularization

Tikhonov regularization is an extension of the ridge regression

$$\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{B}\boldsymbol{\beta}\|_2^2,$$

where $\boldsymbol{B} \in \mathbb{R}^{m\times p}$ is a fixed regularization matrix and λ is a tuning parameter. It is solved by the generalized singular value decomposition (GSVD).

Least squares with quadratic inequality constraint (LSQI)

Least squares with quadratic inequality constraint (LSQI) minimizes the least squares criterion over a hyper-ellipsoid:

$$\begin{aligned} \text{minimize} \quad & \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 \\ \text{subject to} \quad & \|\boldsymbol{B}\boldsymbol{\beta}\|_2 \le \alpha, \end{aligned}$$

where $\boldsymbol{B} \in \mathbb{R}^{m \times p}$ is a fixed regularization matrix. It is solved by the generalized singular value decomposition (GSVD). See Golub and Van Loan (1996, Section 2.1.1).

Concluding remarks on numerical linear algebra

- Numerical linear algebra forms the building blocks of most computation we do. Most lines of our code are numerical linear algebra.
- Be flop and memory aware.

  > The form of a mathematical expression and the way the expression should be evaluated in actual practice may be quite different.

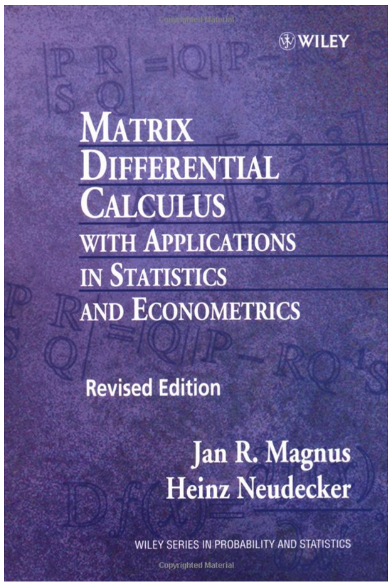
- Be alert to problem structure and make educated choice of software/algorithm.

  > The structure should be exploited whenever solving a problem.

- Do not write your own matrix computation routines unless for good reason. Utilize `BLAS` and `LAPACK` as much as possible!
- In contrast, for optimization, often we need to devise problem specific optimization routines, or even "mix and match" them.

Some useful reference books on (numerical) linear algebra

- Linear algebra books: Magnus and Neudecker (1999), Horn and Johnson (1985), Harville (1997), Gentle (2007)


WILEY

MATRIX DIFFERENTIAL CALCULUS WITH APPLICATIONS IN STATISTICS AND ECONOMETRICS

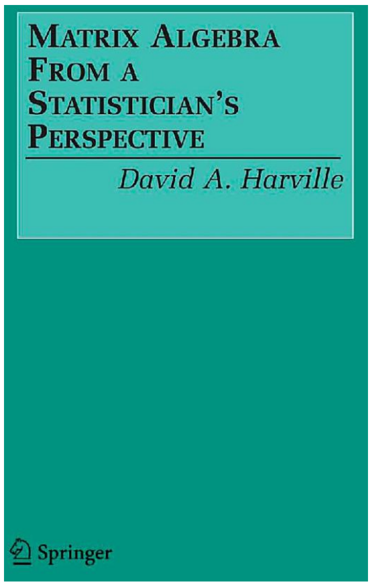
Revised Edition

Jan R. Magnus
Heinz Neudecker

WILEY SERIES IN PROBABILITY AND STATISTICS

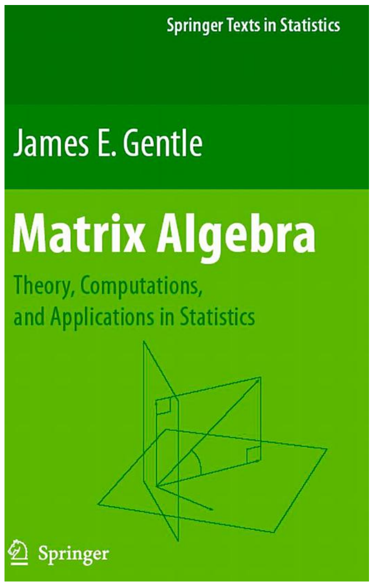
MATRIX ANALYSIS

ROGER A. HORN
AND
CHARLES R. JOHNSON


MATRIX ALGEBRA FROM A STATISTICIAN'S PERSPECTIVE

David A. Harville

Springer


Springer Texts in Statistics

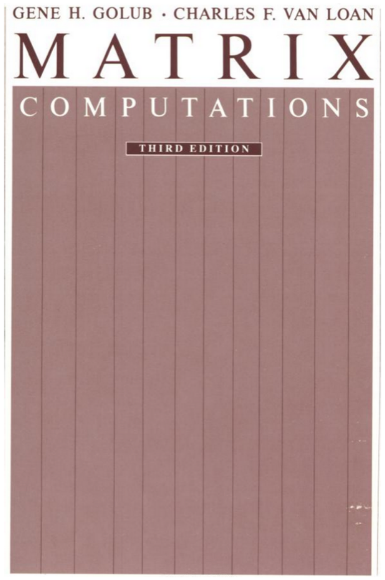
James E. Gentle

Matrix Algebra

Theory, Computations, and Applications in Statistics

Springer

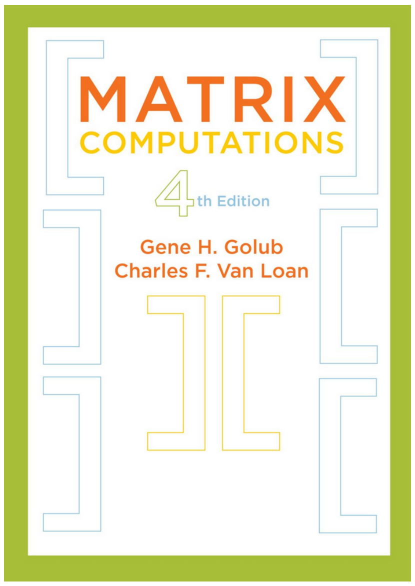
- Golub and Van Loan (1996): "Bible" in numerical linear algebra.


GENE H. GOLUB · CHARLES F. VAN LOAN

MATRIX

COMPUTATIONS

THIRD EDITION


MATRIX

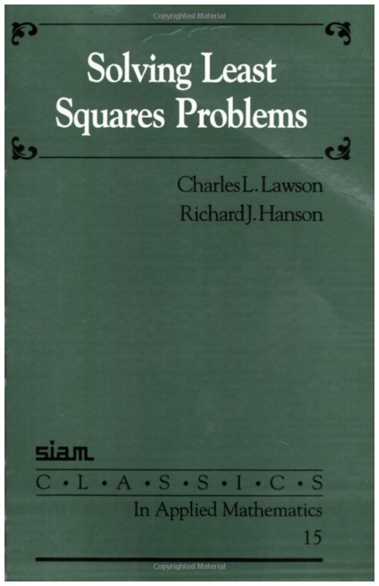
COMPUTATIONS

4th Edition

Gene H. Golub

Charles F. Van Loan

- Lawson and Hanson (1987) and Björck (1996): classical monographs on solving least squares problems.


Solving Least Squares Problems

Charles L. Lawson

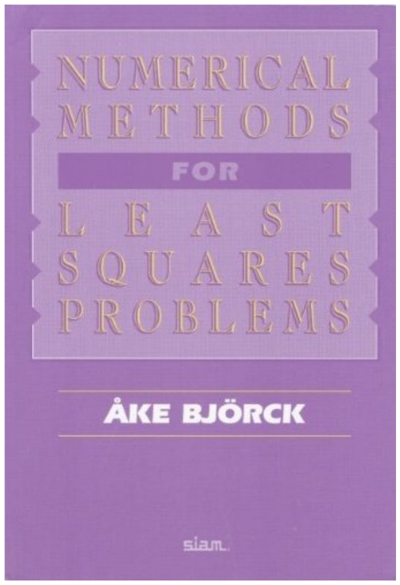
Richard J. Hanson

siam

C · L · A · S · S · I · C · S

In Applied Mathematics

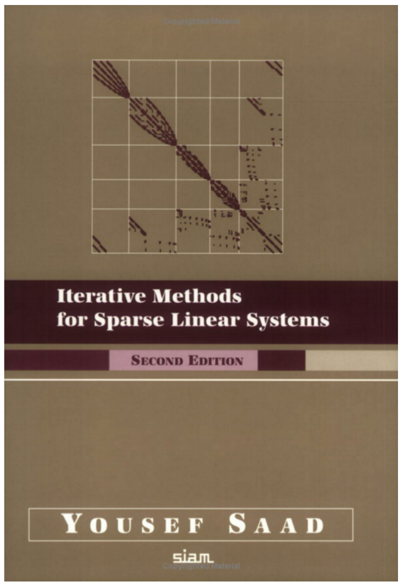
15


NUMERICAL METHODS FOR LEAST SQUARES PROBLEMS

ÅKE BJÖRCK

siam

- Saad (2003): standard reference for iterative methods



Iterative Methods for Sparse Linear Systems

SECOND EDITION

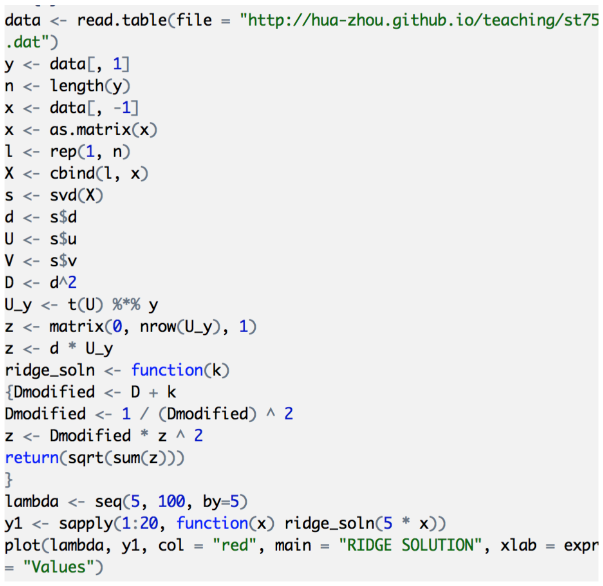
YOUSEF SAAD

siam

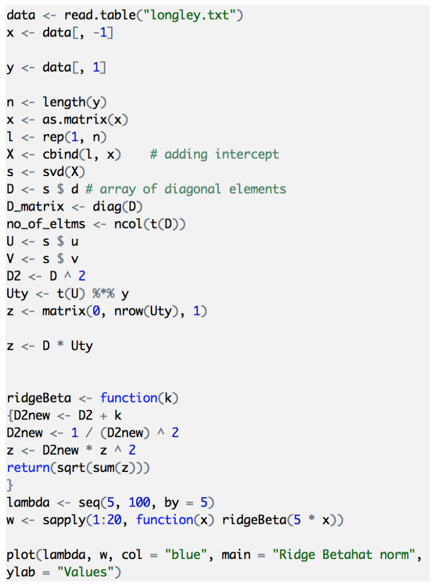
16 Lecture 16, Oct 23

Announcements

- HW5 returned. Feedback:
  - What's the likelihood of two independently written code looking like this?



```
data <- read.table(file = "http://hua-zhou.github.io/teaching/st75
.dat")
y <- data[, 1]
n <- length(y)
x <- data[, -1]
x <- as.matrix(x)
l <- rep(1, n)
X <- cbind(l, x)
s <- svd(X)
d <- s$d
U <- s$u
V <- s$v
D <- d^2
U_y <- t(U) %*% y
z <- matrix(0, nrow(U_y), 1)
z <- d * U_y
ridge_soln <- function(k)
{Dmodified <- D + k
Dmodified <- 1 / (Dmodified) ^ 2
z <- Dmodified * z ^ 2
return(sqrt(sum(z)))
}
lambda <- seq(5, 100, by=5)
y1 <- sapply(1:20, function(x) ridge_soln(5 * x))
plot(lambda, y1, col = "red", main = "RIDGE SOLUTION", xlab = expr
= "Values")
```



```
data <- read.table("longley.txt")
x <- data[, -1]

y <- data[, 1]

n <- length(y)
x <- as.matrix(x)
l <- rep(1, n)
X <- cbind(l, x)    # adding intercept
s <- svd(X)
D <- s $ d # array of diagonal elements
D_matrix <- diag(D)
no_of_eltms <- ncol(t(D))
U <- s $ u
V <- s $ v
D2 <- D ^ 2
Uty <- t(U) %*% y
z <- matrix(0, nrow(Uty), 1)

z <- D * Uty


ridgeBeta <- function(k)
{D2new <- D2 + k
D2new <- 1 / (D2new) ^ 2
z <- D2new * z ^ 2
return(sqrt(sum(z)))
}
lambda <- seq(5, 100, by = 5)
w <- sapply(1:20, function(x) ridgeBeta(5 * x))

plot(lambda, w, col = "blue", main = "Ridge Betahat norm",
ylab = "Values")
```

  Copying code from any source without acknowledgement is plagiarism. First time, hw score is 0; second time, course grade is F.

  - Q2 (ridge regression): avoid loops. Vectorize code (matrix/vector operation) > `apply`, `sapply` > `for` loop.
  - Organize your code into functions.
  - Sketch of solution: `http://hua-zhou.github.io/teaching/st758-2014fall/hw05sol.html`

- HW6 posted and due Nov 6. Start early.
- ST790: Advanced Statistical Computing.

Last time

- Iterative methods (Lanszos and Arnoldi methods) for huge, structured $\boldsymbol{A}$

- Jacobi method for eigen-decomposition (parallel computing)
- Generalized eigen-problem: $\boldsymbol{A}\boldsymbol{x} = \lambda \boldsymbol{B}\boldsymbol{x}$ (PLS, SIR, CCA, ...) and generalized singular value decomposition
- Variants of least squares problem
- Concluding remarks for numerical linear algebra

Today

- MLE
- General optimization theory

MLE (as a motivation for optimization)

A great idea due to Fisher in 20s, and made rigorous by Cramer and others in 40s.

- Notations:
  - Density: $f(\boldsymbol{x}|\boldsymbol{\theta})$, where $\boldsymbol{\theta} \in \boldsymbol{\Theta} \subset \mathbb{R}^p$
  - Log-likelihood function: $L(\boldsymbol{\theta}) = \ln f(\boldsymbol{x}|\theta)$
  - (Column) Gradient/score vector: $\nabla L(\boldsymbol{\theta}) \in \mathbb{R}^{p \times 1}$
  - Differential: $dL(\boldsymbol{\theta}) = [\nabla L(\boldsymbol{\theta})]^\intercal \in \mathbb{R}^{1 \times p}$
  - Hessian: $d^2 L(\boldsymbol{\theta}) = \nabla^2 L(\boldsymbol{\theta})$
  - Observed information matrix: $-d^2 L(\boldsymbol{\theta})$
  - Expected (Fisher) information matrix: $\boldsymbol{I}(\boldsymbol{\theta}) = \mathbf{E}_{\boldsymbol{\theta}}[-d^2 L(\boldsymbol{\theta})]$
  - Given iid observations $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ from $f(\cdot|\boldsymbol{\theta})$,

$$L_n(\boldsymbol{\theta}) = \sum_{i=1}^{n} \ln f(\boldsymbol{x}_i|\boldsymbol{\theta})$$

  - Maximum likelihood estimator (MLE):

$$\hat{\boldsymbol{\theta}}_{\text{MLE}} = \text{argmax}_{\boldsymbol{\theta}}\, L_n(\boldsymbol{\theta})$$

- Consistency of MLE

– Under the true parameter value $\boldsymbol{\theta}_0$,

$$\begin{aligned} & M_n(\boldsymbol{\theta}) = \frac{1}{n}[L_n(\boldsymbol{\theta}) - L_n(\boldsymbol{\theta}_0)] \\ \to \quad & M(\boldsymbol{\theta}) = \mathbf{E}_{\boldsymbol{\theta}_0}[\ln f(\boldsymbol{X}|\boldsymbol{\theta}) - \ln f(\boldsymbol{X}|\boldsymbol{\theta}_0)] \end{aligned}$$

for all $\boldsymbol{\theta}$ almost surely.

– Note that $M(\boldsymbol{\theta})$ is the negative Kullback-Leibler divergence between distribution at $\boldsymbol{\theta}$ and distribution at $\boldsymbol{\theta}_0$.

Assuming *identifiability*, by the *information inequality*, $M(\boldsymbol{\theta})$ achieves maximum uniquely at $\boldsymbol{\theta}_0$. We hope the MLE

$$\hat{\boldsymbol{\theta}}_n = \text{argmax}_{\boldsymbol{\theta}}\, \boldsymbol{M}_n(\boldsymbol{\theta})$$

converges to

$$\boldsymbol{\theta}_0 = \text{argmax}_{\boldsymbol{\theta}}\, \boldsymbol{M}(\boldsymbol{\theta}).$$

– Need *uniform convergence* of $M_n(\boldsymbol{\theta})$ to $M(\boldsymbol{\theta})$, i.e.,

$$\sup_{\Theta} |M_n(\boldsymbol{\theta}) - M(\boldsymbol{\theta})|$$

converges to 0 in probability. A set of sufficient conditions for uniform convergence:

* compactness of the parameter space $\boldsymbol{\Theta}$
* continuity of $M(\boldsymbol{\theta})$ in $\boldsymbol{\theta}$ for any $\boldsymbol{x}$
* $M(\boldsymbol{\theta})$ dominated by an integrable function

– Example of non-uniform convergence: $f_n(x) = 1_{\{n,n+1\}}$ (or a triangle on $[n, n+1]$ if we want f_n to be continuous). $f_n \to f \equiv 0$ pointwise but not uniformly.

• Asymptotic normality of MLE

– Assume $\hat{\boldsymbol{\theta}}_n$ is consistent for $\boldsymbol{\theta}_0$.

– Taylor expansion on $\mathbf{0}_p = \frac{1}{n}\nabla L_n(\hat{\boldsymbol{\theta}}_n)$ gives

$$\begin{aligned} \mathbf{0}_p &= \frac{1}{n}\nabla L_n(\boldsymbol{\theta}_0) + \left[\frac{1}{n} d^2 L_n(\boldsymbol{\theta}_0)\right] (\hat{\boldsymbol{\theta}}_n - \boldsymbol{\theta}_0) \\ & \quad + \frac{1}{2}[\boldsymbol{I}_p \otimes (\hat{\boldsymbol{\theta}}_n - \boldsymbol{\theta}_0)^{\intercal}] \left[\frac{1}{n} D d^2 L_n(\tilde{\boldsymbol{\theta}}_n)\right] (\hat{\boldsymbol{\theta}}_n - \boldsymbol{\theta}_0), \end{aligned}$$

where $\tilde{\boldsymbol{\theta}}_n$ is somewhere between $\boldsymbol{\theta}_0$ and $\hat{\boldsymbol{\theta}}_n$. If $\frac{1}{n}Dd^2L_n(\tilde{\boldsymbol{\theta}}_n) = O_p(1)$ (bounded in probability), then the third term is $o_p(1)(\hat{\boldsymbol{\theta}}_n - \boldsymbol{\theta}_0)$ and

$$\sqrt{n}(\hat{\boldsymbol{\theta}}_n - \boldsymbol{\theta}_0) = \left[-\frac{1}{n}d^2L_n(\boldsymbol{\theta}_0) + o_p(1)\right]^{-1} \frac{\sqrt{n}}{n}\nabla L_n(\boldsymbol{\theta}_0).$$

Now

* $-\frac{1}{n}d^2L_n(\boldsymbol{\theta}_0) + o_p(1) \to \mathbf{E}_{\boldsymbol{\theta}_0}[-d^2L(\boldsymbol{\theta}_0)] = \boldsymbol{I}(\boldsymbol{\theta}_0)$ almost surely by the law of large number.
* $n^{-1/2}\nabla L_n(\boldsymbol{\theta}_0)$ converges to a multivariate normal with mean $\mathbf{0}_p$ and variance

$$\mathbf{E}_{\boldsymbol{\theta}_0}[\nabla L(\boldsymbol{\theta}_0)dL(\boldsymbol{\theta}_0)],$$

  which equals $\boldsymbol{I}(\boldsymbol{\theta}_0)$ under exchangeability of integral and differentiation.

Then by the Slutsky theorem,

$$\begin{aligned} & \sqrt{n}(\hat{\boldsymbol{\theta}}_n - \boldsymbol{\theta}_0) \\ \to \ & N_p\left(\mathbf{0}_p, \boldsymbol{I}^{-1}(\boldsymbol{\theta}_0) \cdot \mathbf{E}_{\boldsymbol{\theta}_0}[\nabla \ln f(\boldsymbol{\theta}_0) d \ln f(\boldsymbol{\theta}_0)] \cdot \boldsymbol{I}^{-1}(\boldsymbol{\theta}_0)\right) \\ = \ & N_p\left(\mathbf{0}_p, \boldsymbol{I}^{-1}(\boldsymbol{\theta}_0)\right) \end{aligned}$$

in distribution.

– In practice, we can estimate the variance by
  * Fisher information matrix $\boldsymbol{I}^{-1}(\hat{\boldsymbol{\theta}})$,
  * observed information matrix $[-(1/n)d^2L_n(\hat{\boldsymbol{\theta}})]^{-1}$, or
  * the sandwich estimator

• Asymptotic efficiency of MLE.

"Cramer-Rao theorem" says the variance of any unbiased estimator is "at least" $(nI(\boldsymbol{\theta}_0))^{-1}$ (the difference is psd). So MLE has the smallest asymptotic variance within the class of unbiased estimators.

17 Lecture 17, Oct 28

Announcements

- No TA office hours this Friday.
- FAQs on HW6: `http://hua-zhou.github.io/teaching/st758-2014fall/st758fall2014/2014/10/26/hw6-hints.html`
- Roadmap (to winter break!): HW7, HW8, simulation project, final or not?

Last time

- Review of HW5
- MLE

Today

- Newton and scoring algorithm
- Hierarchy of optimization problems

Newton's method and Fisher's scoring (KL Chapter 14)

Joseph Raphson

Consider maximizing log-likelihood $L(\boldsymbol{\theta})$, $\boldsymbol{\theta} \in \boldsymbol{\Theta} \subset \mathbb{R}^p$.

- Newton's method was originally developed for finding roots of nonlinear equations $f(\boldsymbol{\theta}) = \mathbf{0}$ (KL 5.4).
- Newton's method (aka Newton-Raphson method) is considered the gold standard for its fast (quadratic) convergence

$$\frac{\|\boldsymbol{\theta}^{(t+1)} - \boldsymbol{\theta}^*\|}{\|\boldsymbol{\theta}^{(t)} - \boldsymbol{\theta}^*\|^2} \to \text{constant}.$$

- Idea: iterative quadratic approximation.
- Taylor expansion around the current iterate $\boldsymbol{\theta}^{(t)}$

$$L(\boldsymbol{\theta}) \approx L(\boldsymbol{\theta}^{(t)}) + dL(\boldsymbol{\theta}^{(t)})(\boldsymbol{\theta} - \boldsymbol{\theta}^{(t)}) + \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}^{(t)})^\intercal d^2L(\boldsymbol{\theta}^{(t)})(\boldsymbol{\theta} - \boldsymbol{\theta}^{(t)})$$

  and then maximize the quadratic approximation.
- To maximize the quadratic function, we equate its gradient to zero

$$\nabla L(\boldsymbol{\theta}^{(t)}) + [d^2L(\boldsymbol{\theta}^{(t)})](\boldsymbol{\theta} - \boldsymbol{\theta}^{(t)}) = \mathbf{0}_p,$$

  which suggests the next iterate

$$\begin{aligned} \boldsymbol{\theta}^{(t+1)} &= \boldsymbol{\theta}^{(t)} - [d^2L(\boldsymbol{\theta}^{(t)})]^{-1}\nabla L(\boldsymbol{\theta}^{(t)}) \\ &= \boldsymbol{\theta}^{(t)} + [-d^2L(\boldsymbol{\theta}^{(t)})]^{-1}\nabla L(\boldsymbol{\theta}^{(t)}). \end{aligned}$$

- Some issues with the Newton's iteration
  - Need to derive, evaluate, and "invert" the observed information matrix. Remedies:
    1. exploit structure in Hessian whenever possible,
    2. numerical differentiation (works for small problems), or
    3. quasi-Newton method (to be discussed later)
  - Stability: Newton's iterate is not guaranteed to be an ascent algorithm. It's equally happy to head uphill or downhill. Remedies:
    1. approximate $-d^2L(\boldsymbol{\theta}^{(t)})$ by a positive definite $\boldsymbol{A}$ (if it's not), *and*
    2. line search (backtracking).

In summary, Newton's method iterates according to

$$\boxed{\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} + s[\boldsymbol{A}^{(t)}]^{-1}\nabla L(\boldsymbol{\theta}^{(t)}) = \boldsymbol{\theta}^{(t)} + s\Delta\boldsymbol{\theta}^{(t)}}$$

where $\boldsymbol{A}^{(t)}$ is a pd approximation of $-d^2L(\boldsymbol{\theta}^{(t)})$ and s is a step length. Why? By first-order Taylor expansion,

$$\begin{aligned} & L(\boldsymbol{\theta}^{(t)} + s\Delta\boldsymbol{\theta}^{(t)}) - L(\boldsymbol{\theta}^{(t)}) \\ = \; & dL(\boldsymbol{\theta}^{(t)})s\Delta\boldsymbol{\theta}^{(t)} + o(s) \\ = \; & s dL(\boldsymbol{\theta}^{(t)})[\boldsymbol{A}^{(t)}]^{-1}\nabla L(\boldsymbol{\theta}^{(t)}) + o(s). \end{aligned}$$

For s sufficiently small, right hand side is strictly positive.

- Backtracking strategy: step-halving ($s = 1, 1/2, \ldots$), golden section search, cubic interpolation, Amijo rule, ...
- How to approximating $-d^2L(\boldsymbol{\theta})$? More of an art than science. Often requires problem specific analysis.
- Taking $\boldsymbol{A} = \boldsymbol{I}$ leads to the method of *steepest ascent*, aka *gradient ascent*.
- *Fisher's scoring method*: replace $-d^2L(\boldsymbol{\theta})$ by the expected Fisher information matrix

  $$\boldsymbol{I}(\boldsymbol{\theta}) = \mathbf{E}[-d^2L(\boldsymbol{\theta})] = \mathbf{E}[\nabla L(\boldsymbol{\theta})dL(\boldsymbol{\theta})] \succeq \mathbf{0}_{p\times p},$$

  which is psd under exchangeability of expectation and differentiation.

  Therefore the Fisher's scoring algorithm iterates according to

  $$\boxed{\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} + s[\boldsymbol{I}(\boldsymbol{\theta}^{(t)})]^{-1}\nabla L(\boldsymbol{\theta}^{(t)})}$$

Hierarchy of optimization problems

Difficulty of optimization problems *in general*

Harder	Easier
discrete (combinatorial) optimization	continuous optimization
non-smooth	smooth
non-convex	convex
constrained	un-constrained
inequality constraint	equality constrained

18 Lecture 18, Oct 30

Announcements

- No TA office hours this Friday. Makeup office hours next Mon, Nov 3 @ 3P-5P.

Last time

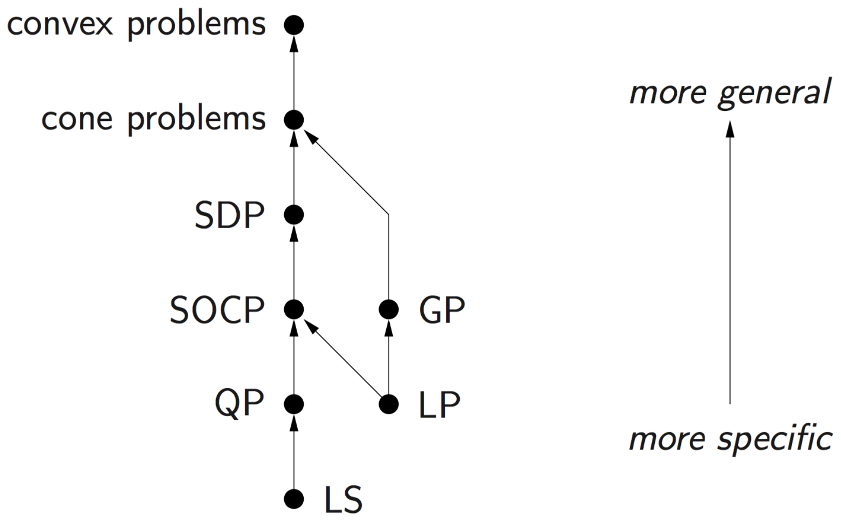
- Newton and Fisher scoring method
- Hierarchy of optimization problems

Today

- Convex optimization
- Some fundamentals of optimization theory

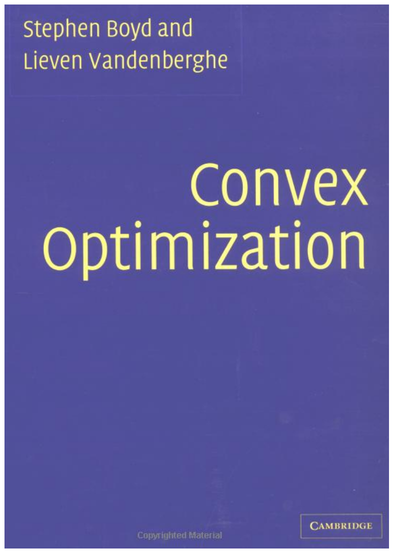
Convex optimization

- *Extremely important* skill to recognize or transform to convex problems



  - Examples: ℓ_∞ regression, ℓ_1 regression, quantile regression, and many more.

  - *Convex Optimization* by Boyd and Vandenberghe and accompanying slides
    `http://www.stanford.edu/~boyd/cvxbook/`



  - Lecture videos:
    `http://www.stanford.edu/class/ee364a/videos.html`
    `http://www.stanford.edu/class/ee364b/videos.html`

- Convex programming (LS, LP, QP, GP, SOCP, SDP) is almost becoming a technology (`Cplex`, `Gurobi`, `Mosek`, `cvx`, `Matlab`, ...)

- Non-convex optimization still occurs in many natural statistical applications. Statisticians have specialized tools to deal with them (Fisher scoring method, EM algorithm, simulated annealing, ...)

Unconstrained optimization (KL 11.2)

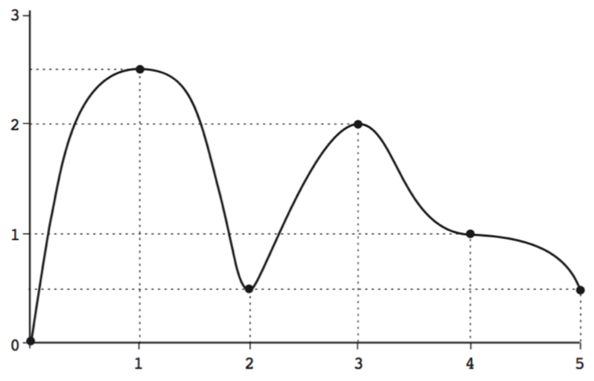


Figure 1 Unconstrained optimization in one variable

- Possible confusion:
  - We (statisticians) talk about *maximization*: max $L_n(\boldsymbol{\theta})$.
  - People talk about *minimization* in the optimization world: $\min_{\boldsymbol{x}} f(\boldsymbol{x})$.
- Fundamental questions: When does a function have minimum? How do we tell whether a point is minimum?
- When does a function have a minimum?

  (Weierstrass) A continuous function $f(x)$ defined on a compact (closed and bounded) set is bounded below and attains its minimum.
- None of the Weierstrass conditions can be taken out.
  - $f(x) = x$, $x \in (-\infty, \infty)$. Non-compact support.
  - $f(x) = \tan(x)$, $x \in (-\pi/2, \pi/2)$. Non-compact support.
  - $f(x) = x$, $x \in (-1, 1)$ and $f(-1) = f(1) = 0$. The minimum not attained by the *discontinuous* function f.
- None of the Weierstrass conditions are necessary. $f(x) = x$, $x \in [0, 2)$, $f(x) = 1$, $x \in (2, \infty)$.

- Coercive function: $\{\boldsymbol{x} \in U : f(\boldsymbol{x}) \leq f(\boldsymbol{y})\}$ is compact for all $\boldsymbol{y} \in U$.

  Weierstrass theorem also holds for a coercive function defined on a possibly open U.

- Necessary conditions for a local minimum.

  Assume f has a local minimum at interior point $\boldsymbol{y} \in U$.

  - (Fermat) If f is differentiable, then $\nabla f(\boldsymbol{x})$ vanishes at $\boldsymbol{y}$.
  - If f is twice differentiable, then $d^2 f(\boldsymbol{y})$ is psd.

- Points with $\nabla f(\boldsymbol{x}) = \mathbf{0}$ are called *stationary points* or *critical points.* Most optimization algorithms try to find the stationary points of the function and then check sufficient condition.

- Counter-examples to necessary conditions. (1) $f(x) = x^3$ has zero gradient at 0, which is not local minimum. (2) $f(x) = |x|$ has local minimum at 0, where the gradient does not exist.

- (A first-order sufficient condition; first derivative test) Suppose f is differentiable in a ball $B(\boldsymbol{y})$ around an interior point $\boldsymbol{y}$, and $\langle \nabla f(\boldsymbol{x}), \boldsymbol{x} - \boldsymbol{y} \rangle \geq 0$ for all $\boldsymbol{x} \in B(\boldsymbol{y})$, then $\boldsymbol{y}$ is a local minimum.

- (A second-order sufficient condition; second derivative test) If $\nabla f(\boldsymbol{y}) = \mathbf{0}$ and $d^2(\boldsymbol{y})$ is pd, then y is a strict local minimum.

- Remark: In case $d^2 f(\boldsymbol{y})$ is neither positive definite nor negative definite but non-singular, $\boldsymbol{y}$ is a *saddle point*, i.e., a stationary point that is neither a local minimum nor a local maximum. In case $d^2 f(\boldsymbol{y})$ is singular, we cannot tell.

- Example: $f_1(x, y) = x^4 + y^4$, $f_2(x, y) = -x^4 - y^4$, $f_3(x, y) = x^3 + y^3$. Origin is a stationary (critical) point and the Hessian $d^2 f_i(0, 0) = \mathbf{0}_{2\times 2}$ is singular. Origin is a minimum, maximum, and a saddle point respectively.

Convexity and global optima

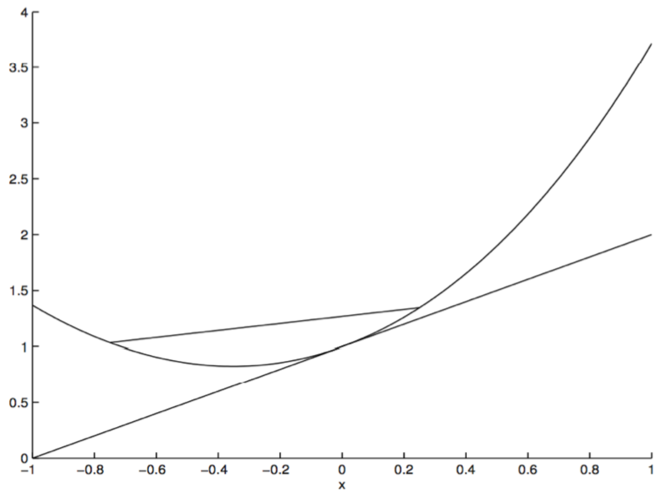


FIGURE 11.2. Plot of the Convex Function $e^x + x^2$

- $f : U \mapsto \mathbb{R}$ is *convex* if
  - U is a convex set ($\lambda \boldsymbol{x} + (1-\lambda)\boldsymbol{y} \in U$ for all $\boldsymbol{x}, \boldsymbol{y} \in U$ and $\lambda \in (0,1)$), and
  - $f(\lambda \boldsymbol{x} + (1-\lambda)\boldsymbol{y}) \leq \lambda f(\boldsymbol{x}) + (1-\lambda) f(\boldsymbol{y})$, for all $\boldsymbol{x}, \boldsymbol{y} \in U$ and $\lambda \in (0,1)$.

  f is *strictly convex* if the inequality if strict for all $\boldsymbol{x} \neq \boldsymbol{y} \in U$ and λ.

- (*Supporting hyperplane inequality*) A differentiable function f is convex if and only if $f(\boldsymbol{x}) \geq f(\boldsymbol{y}) + \langle \nabla f(\boldsymbol{y}), \boldsymbol{x} - \boldsymbol{y} \rangle$ for all $\boldsymbol{x}, \boldsymbol{y} \in U$.

- (Second-order condition for convexity) A twice differentiable function f is convex if and only if $d^2 f(\boldsymbol{x})$ is psd for all $\boldsymbol{x} \in U$.

  It is strictly convex if $d^2 f(\boldsymbol{x})$ is pd for all $\boldsymbol{x} \in U$.

- (Convexity and global optima) Suppose f is a convex function on a convex set U.
  1. Any stationary point $\boldsymbol{y}$ is a global minimum. (By supporting hyperplane inequality, $f(\boldsymbol{x}) \geq f(\boldsymbol{y}) + \langle \nabla f(\boldsymbol{y}), \boldsymbol{x} - \boldsymbol{y} \rangle = f(\boldsymbol{y})$ for all $\boldsymbol{x} \in U$.)
  2. Any local minimum is a global minimum.

3. The set of (global) minima $\{\boldsymbol{x} \in U : f(\boldsymbol{x}) = f(\boldsymbol{y})\}$ is convex.
4. If f is strictly convex, then the global minimum, if exists, is unique.

- Example: Least squares estimate. $f(\boldsymbol{\beta}) = \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2$ has Hessian $d^2 f = \boldsymbol{X}^\intercal \boldsymbol{X}$ which is psd. So f is convex and any stationary point (solution to the normal equation) is a global minimum. When $\boldsymbol{X}$ is rank deficient, the set of solutions is convex.

- (Jensen's inequality) W a random variable taking values in U and h is convex on U. Then

$$\mathbf{E}[h(W)] \geq h[\mathbf{E}(W)],$$

provided both expectations exist. For a strictly convex h, equality holds if and only if $W = \mathbf{E}(W)$ almost surely.

Proof: supporting hyperplane inequality taking $\boldsymbol{x} = \boldsymbol{W}$ and $\boldsymbol{y} = E(\boldsymbol{W})$.

- (Information inequality) Let f and g be two densities with respect to a common measure μ. $h, g > 0$ almost everywhere relative to μ. Then

$$\mathbf{E}_f(\ln f) \geq \mathbf{E}_f(\ln g),$$

with equality if and only if $f = g$ almost everywhere on μ.

Proof: Apply Jensen's inequality to the convex function $-\ln(t)$ and random variable $W = g(x)/f(x)$.
Applications: M-estimation, EM algorithm.

19 Lecture 19, Nov 4

Announcements

- Answer to Caleb's question: Why do people care about ℓ_∞ regression?

$$\min_{\boldsymbol{\beta}} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_\infty = \min_{\boldsymbol{\beta}} \max_i |y_i - \boldsymbol{x}_i^T \boldsymbol{\beta}|.$$

  Chebychev approximation, or *minimax approximation*

$$f(x) \sim \sum_{i=0}^{N} c_i T_i(x).$$

  For fixed degree N, find the coefficients c_i that minimize the worst possible approximation error.

Last time

- Convex optimization
- Fundamentals of optimization theory: optimality conditions for unconstrained optimization, convexity and global optima

Today

- Fundamentals of optimization theory (cont'd): optimality conditions for constrained optimization
- Application of Newton and Fisher scoring algorithm: GLM

Optimization with equality constraints (KL 11.3)

Consider the equality constrained minimization problem

$$\begin{array}{rl} \text{minimize} & f(\boldsymbol{x}) \\ \text{subject to} & g_i(\boldsymbol{x}) = 0, i = 1, \ldots, m \\ & \boldsymbol{x} \in U \subset \mathbb{R}^n. \end{array}$$

We write

$$g(\boldsymbol{x}) = \begin{pmatrix} g_1(\boldsymbol{x}) \\ \vdots \\ g_m(\boldsymbol{x}) \end{pmatrix} \in \mathbb{R}^m \text{ and } Dg(\boldsymbol{x}) = \begin{pmatrix} dg_1(\boldsymbol{x}) \\ \cdots \\ dg_m(\boldsymbol{x}) \end{pmatrix} \in \mathbb{R}^{m \times n}.$$

- Method of Lagrange multiplier. *Lagrangian* function

$$L(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + \boldsymbol{\lambda}^{\mathsf{T}} g(\boldsymbol{x}) = f(\boldsymbol{x}) + \sum_{i=1}^{m} \lambda_i g_i(\boldsymbol{x}).$$

  Strategy for finding the equality constrained minimum: find the stationary point $(\boldsymbol{x}^*, \boldsymbol{\lambda}^*)$ of the Lagrangian,

$$\begin{aligned} \nabla_{\boldsymbol{x}} L(\boldsymbol{x}, \boldsymbol{\lambda}) &= \nabla f(\boldsymbol{x}) + \sum_{i=1}^{m} \lambda_i \nabla g_i(\boldsymbol{x}) = \mathbf{0}_n \\ g(\boldsymbol{x}) &= \mathbf{0}_m. \end{aligned}$$

- Intuition: Null space of the matrix Dg is the tangent space. Movement along the tangent space does not change constraint function values. We need the ∇f to be orthogonal to the tangent space. In other words, ∇f is in the column space of $[Dg]^T$.

- Intuition of the Lagrange multiplier method: Hill climb along a trail which is a contour line of the constraint function. We feel effortless exactly when the direction of our movement is perpendicular to the steepest ascent direction of the hill. In other words, steepest ascent direction of the constraint function aligns with that of the hill.

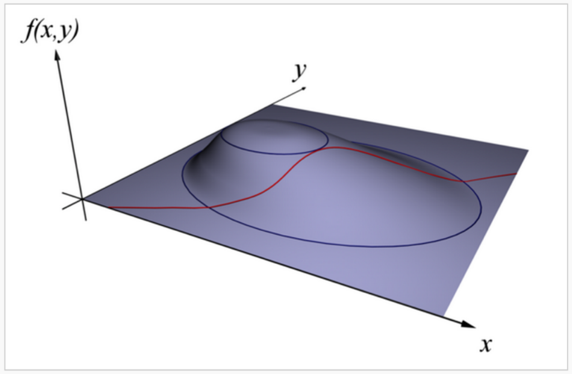


Figure 1: Find *x* and *y* to maximize $f(x,y)$ subject to a constraint (shown in red) $g(x,y)=c$.

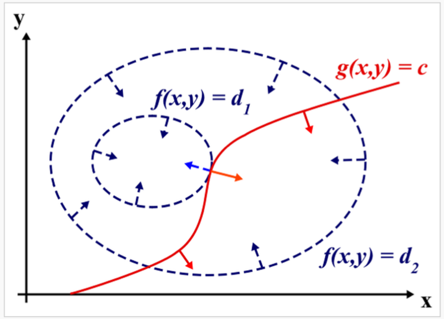


Figure 2: Contour map of Figure 1. The red line shows the constraint $g(x,y)=c$. The blue lines are contours of $f(x,y)$. The point where the red line tangentially touches a blue contour is our solution.Since $d_1 > d_2$, the solution is a maximization of $f(x,y)$

- (Necessary condition for a constrained local minimum) Assume conditions (i) $g(\boldsymbol{y}) = \mathbf{0}_m$, (2) f and g are differentiable in some n-ball $B(\boldsymbol{y})$, (iii) $Dg(\boldsymbol{y}) \in \mathbb{R}^{m\times n}$ is continuous at $\boldsymbol{y}$, (iv) $Dg(\boldsymbol{y})$ has full row rank, (v) $f(\boldsymbol{x}) \ge f(\boldsymbol{y})$ for any $\boldsymbol{x} \in B(\boldsymbol{y})$ satisfying $g(\boldsymbol{x}) = \mathbf{0}_m$ ($\boldsymbol{y}$ a local minimum subject to constraints). Then there exists $\boldsymbol{\lambda} \in \mathbb{R}^m$ satisfying $\nabla f(\boldsymbol{y}) + \sum_{i=1}^m \lambda_i \nabla g_i(\boldsymbol{y}) = \mathbf{0}_n$, i.e., $(\boldsymbol{y}, \boldsymbol{\lambda})$ is a stationarity point of the Lagrangian $L(\boldsymbol{x}, \boldsymbol{\lambda})$. In other words, there exists $\boldsymbol{\lambda} \in \mathbb{R}^m$, such that $\nabla L(\boldsymbol{y}, \boldsymbol{\lambda}) = \mathbf{0}_{m+n}$.

- (Sufficient condition for a constrained local minimum) (i) f twice differentiable at $\boldsymbol{y}$, (ii) g twice differentiable at $\boldsymbol{y}$, (iii) the Jacobian matrix $Dg(\boldsymbol{y}) \in \mathbb{R}^{m\times n}$ has full row rank m, (iv) it is a stationarity point of the Lagrangian at a given $\boldsymbol{\lambda} \in \mathbb{R}^m$, (v) $\boldsymbol{u}^\intercal d^2 f(\boldsymbol{y}) \boldsymbol{u} > 0$ for all $\boldsymbol{u} \neq \mathbf{0}_n$ satisfying $[Dg(\boldsymbol{y})]\boldsymbol{u} = \mathbf{0}_m$ (tangent vectors). Then $\boldsymbol{y}$ is a strict local minimum of f under constraint $g(\boldsymbol{y}) = \mathbf{0}_m$.

- Check condition (v). Condition (v) is equivalent to the "bordered determinantal criterion"

$$(-1)^m \det \begin{pmatrix} \mathbf{0}_{m\times m} & \boldsymbol{B}_r \\ \boldsymbol{B}_r^\intercal & \boldsymbol{A}_{rr} \end{pmatrix} > 0$$

  for $r = m+1, \ldots, n$, where

  - $\boldsymbol{A}_{rr}$ is the top left r-by-r block of $d^2 f(\boldsymbol{y}) + \sum_{i=1}^m \lambda_i d^2 g_i(\boldsymbol{y})$
  - $\boldsymbol{B}_r \in \mathbb{R}^{m\times r}$ is the first r columns of the $Dg(\boldsymbol{y})$.

- (Sufficient condition for a global constrained minimum) Lagrangian first order condition + convexity of the Lagrangian on U.

- (Interpretation of the Lagrange multipliers $\boldsymbol{\lambda}$). Consider $\min f(\boldsymbol{x})$ subject to some resource constraint $g(\boldsymbol{x}) = \boldsymbol{b}$. Consider the solution $\boldsymbol{x}^*(\boldsymbol{b})$ as a function of $\boldsymbol{b}$. Then it can be shown that
$$\frac{\partial f(\boldsymbol{x}^*(\boldsymbol{b}))}{\partial b_j} = \lambda_j.$$
That's why the score test is classically called the Lagrange multiplier test.

- Example: Linearly constrained least squares solution. $\min \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2$ subject to linear constrained $\boldsymbol{V}\boldsymbol{\beta} = \boldsymbol{d}$. Form the Lagrangian
$$L(\boldsymbol{\beta}, \boldsymbol{\lambda}) = \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 + \boldsymbol{\lambda}^\mathsf{T}(\boldsymbol{V}\boldsymbol{\beta} - \boldsymbol{d}).$$
Stationary condition says
$$\begin{aligned} \boldsymbol{X}^\mathsf{T}\boldsymbol{X}\boldsymbol{\beta} - \boldsymbol{X}^\mathsf{T}\boldsymbol{y} + \boldsymbol{V}^\mathsf{T}\boldsymbol{\lambda} &= \boldsymbol{0}_p \\ \boldsymbol{V}\boldsymbol{\beta} &= \boldsymbol{d} \end{aligned}$$
or equivalently
$$\begin{pmatrix} \boldsymbol{X}^\mathsf{T}\boldsymbol{X} & \boldsymbol{V}^\mathsf{T} \\ \boldsymbol{V} & \boldsymbol{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{\beta} \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} \boldsymbol{X}^\mathsf{T}\boldsymbol{y} \\ \boldsymbol{d} \end{pmatrix},$$
which can be solved by say sweeping on
$$\begin{pmatrix} \boldsymbol{X}^\mathsf{T}\boldsymbol{X} & \boldsymbol{V}^\mathsf{T} & \boldsymbol{X}^\mathsf{T}\boldsymbol{y} \\ \boldsymbol{V} & \boldsymbol{0} & \boldsymbol{d} \\ \boldsymbol{y}\boldsymbol{X}^\mathsf{T} & \boldsymbol{d}^\mathsf{T} & \boldsymbol{y}^\mathsf{T}\boldsymbol{y} \end{pmatrix},$$
or Cholesky or QR.

Optimization with both equality and inequality constraints (KL 11.4)

Consider the constrained minimization problem

$$\begin{aligned} \text{minimize} \quad & f(\boldsymbol{x}) \\ \text{subject to} \quad & g_i(\boldsymbol{x}) = 0, i = 1, \ldots, p \\ & h_j(\boldsymbol{x}) \le 0, i = 1, \ldots, q \\ & \boldsymbol{x} \in U \subset \mathbb{R}^n. \end{aligned}$$

- Lagrangian function:

$$L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = f(\boldsymbol{x}) + \sum_{i=1}^{p} \lambda_i g_i(\boldsymbol{x}) + \sum_{j=1}^{q} \mu_j h_j(\boldsymbol{x}).$$

- Karush-Kuhn-Tucker (KKT) necessary condition: If (1) $\boldsymbol{y}$ is a local constrained minimum and (2) satisfies certain constraint qualifications (Kuhn-Tucker, Mangasarian-Fromovitz), then

  1. (Lagrangian stationarity condition) there exist $\boldsymbol{\lambda} \in \mathbb{R}^p$, $\boldsymbol{\mu} \in \mathbb{R}^q$ such that

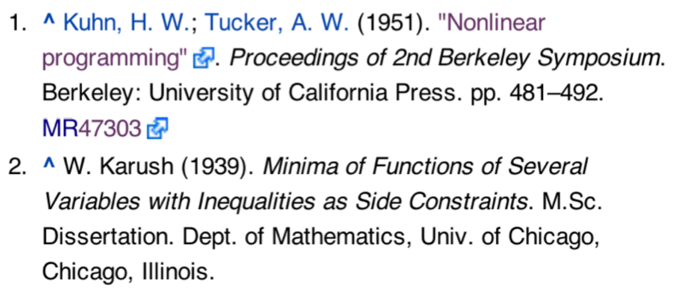
  $$\nabla f(\boldsymbol{y}) + \sum_{i=1}^{p} \lambda_i \nabla g_i(\boldsymbol{y}) + \sum_{j=1}^{q} \mu_j \nabla h_j(\boldsymbol{y}) = \mathbf{0},$$

  2. (Complementary slackness) $\mu_j = 0$ if $h_j(\boldsymbol{y}) < 0$ and $\mu_j > 0$ otherwise.

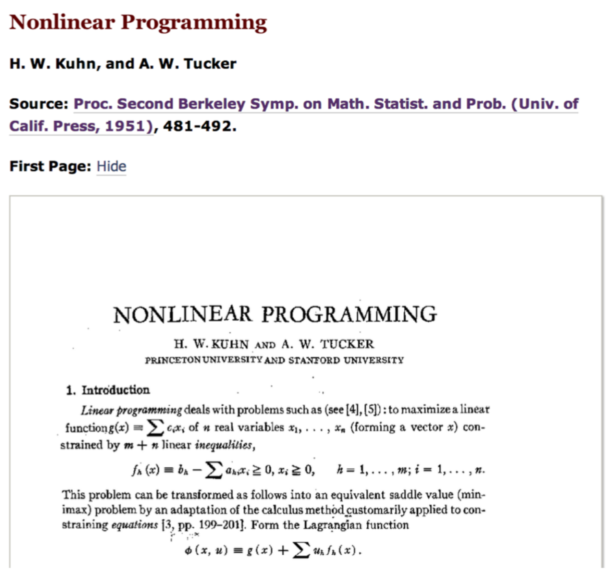
- Sufficient condition: KKT + second order condition.

- Global minimum: KKT conditions + convexity.

- Read KL Section 11.4 for more details. KKT is “one of the great triumphs of 20th century applied mathematics”.



1. ^ Kuhn, H. W.; Tucker, A. W. (1951). "Nonlinear programming". *Proceedings of 2nd Berkeley Symposium*. Berkeley: University of California Press. pp. 481–492. MR47303
2. ^ W. Karush (1939). *Minima of Functions of Several Variables with Inequalities as Side Constraints*. M.Sc. Dissertation. Dept. of Mathematics, Univ. of Chicago, Chicago, Illinois.



Nonlinear Programming

H. W. Kuhn, and A. W. Tucker

Source: Proc. Second Berkeley Symp. on Math. Statist. and Prob. (Univ. of Calif. Press, 1951), **481-492.**

First Page: Hide

NONLINEAR PROGRAMMING

H. W. KUHN AND A. W. TUCKER

PRINCETON UNIVERSITY AND STANFORD UNIVERSITY

1. Introduction

Linear programming deals with problems such as (see [4], [5]): to maximize a linear function $g(x) \equiv \sum c_i x_i$ of n real variables $x_1, \ldots, x_n$ (forming a vector x) constrained by $m + n$ linear *inequalities*,

$$f_h(x) \equiv b_h - \sum a_{hi} x_i \geqq 0, \; x_i \geqq 0, \qquad h = 1, \ldots, m; \; i = 1, \ldots, n.$$

This problem can be transformed as follows into an equivalent saddle value (minimax) problem by an adaptation of the calculus method customarily applied to constraining *equations* [3, pp. 199–201]. Form the Lagrangian function

$$\phi(x, u) \equiv g(x) + \sum u_h f_h(x).$$

Generalized linear model (GLM) (KL 14.7, JM 9.7)

Let's consider a concrete example: logistic regression.

- You want to make a fame by participating the Capital One competition. The goal is to predict whether a credit card transaction is fraud ($y_i = 1$) or not ($y_i = 0$). Predictors ($\boldsymbol{x}_i$) include: time of transaction, last location, merchant, ...
- $y_i \in \{0, 1\}$, $\boldsymbol{x}_i \in \mathbb{R}^p$. Model $y_i \sim$Bernoulli(p_i).

- Logistic regression. Density

$$\begin{aligned} f(y_i|p_i) &= p_i^{y_i}(1-p_i)^{1-y_i} \\ &= e^{y_i \ln p_i + (1-y_i)\ln(1-p_i)} \\ &= e^{y_i \ln \frac{p_i}{1-p_i} + \ln(1-p_i)}, \end{aligned}$$

  where

$$\begin{aligned} E(y_i) = p_i &= \frac{e^{\boldsymbol{x}_i^\top \boldsymbol{\beta}}}{1+e^{\boldsymbol{x}_i^\top \boldsymbol{\beta}}} \quad \text{(mean function, inverse link function)} \\ \boldsymbol{x}_i^T \boldsymbol{\beta} &= \ln\left(\frac{p_i}{1-p_i}\right) \quad \text{(logit link function).} \end{aligned}$$

- Given data $(y_i, \boldsymbol{x}_i)$, $i = 1, \ldots, n$,

$$\begin{aligned} L_n(\boldsymbol{\beta}) &= \sum_{i=1}^n [y_i \ln p_i + (1-y_i)\ln(1-p_i)] \\ &= \sum_{i=1}^n \left[y_i \boldsymbol{x}_i^\top \boldsymbol{\beta} - \ln(1+e^{\boldsymbol{x}_i^\top \boldsymbol{\beta}})\right] \\ \nabla L_n(\boldsymbol{\beta}) &= \sum_{i=1}^n \left(y_i \boldsymbol{x}_i - \frac{e^{\boldsymbol{x}_i^\top \boldsymbol{\beta}}}{1+e^{\boldsymbol{x}_i^\top \boldsymbol{\beta}}}\boldsymbol{x}_i\right) \\ &= \sum_{i=1}^n (y_i - p_i)\boldsymbol{x}_i = \boldsymbol{X}^\top(\boldsymbol{y}-\boldsymbol{p}) \\ -d^2 L_n(\boldsymbol{\beta}) &= \sum_{i=1}^n p_i(1-p_i)\boldsymbol{x}_i \boldsymbol{x}_i^\top = \boldsymbol{X}^\top \boldsymbol{W} \boldsymbol{X}, \\ \text{where} & \quad \boldsymbol{W} = \text{diag}(w_1, \ldots, w_n), w_i = p_i(1-p_i) \\ \boldsymbol{I}_n(\boldsymbol{\beta}) &= \mathbf{E}[-d^2 L_n(\boldsymbol{\beta})] = -d^2 L_n(\boldsymbol{\beta}). \end{aligned}$$

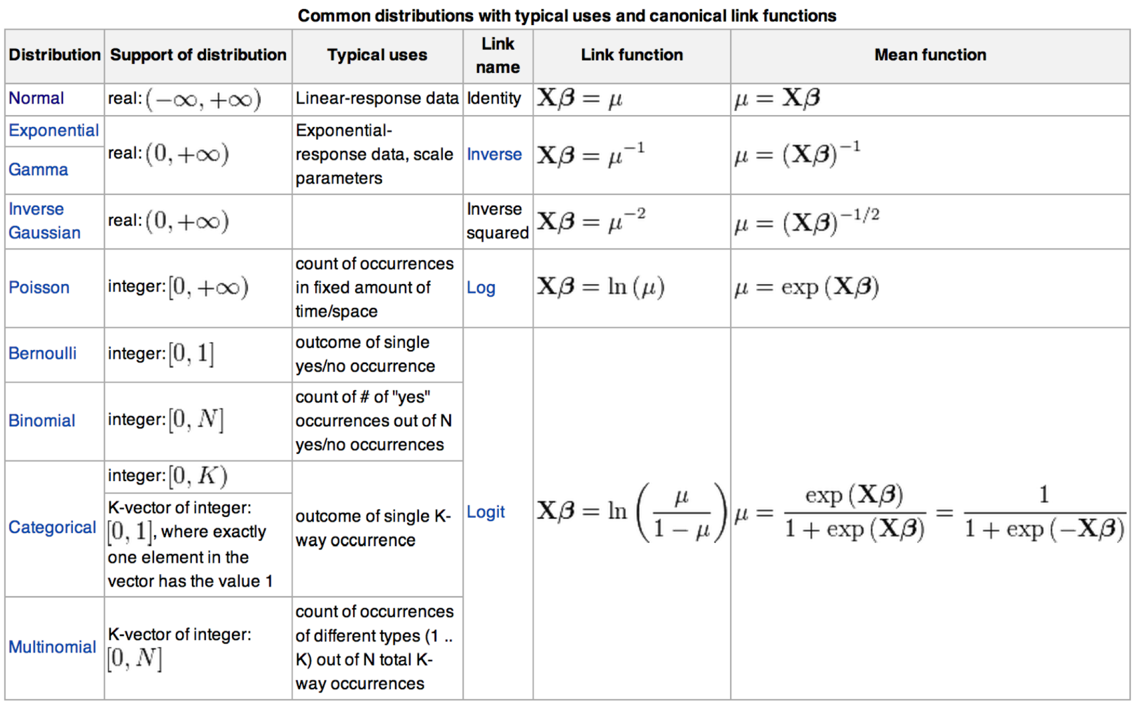
- Newton's method = Fisher's scoring iteration:

$$\begin{aligned} \boldsymbol{\beta}^{(t+1)} &= \boldsymbol{\beta}^{(t)} + s[-d^2 L(\boldsymbol{\beta}^{(t)})]^{-1} \nabla L(\boldsymbol{\beta}^{(t)}) \\ &= \boldsymbol{\beta}^{(t)} + s(\boldsymbol{X}^\top \boldsymbol{W}^{(t)} \boldsymbol{X})^{-1} \boldsymbol{X}^\top (\boldsymbol{y} - \boldsymbol{p}^{(t)}) \\ &= (\boldsymbol{X}^\top \boldsymbol{W}^{(t)} \boldsymbol{X})^{-1} \boldsymbol{X}^\top \boldsymbol{W}^{(t)} \left[\boldsymbol{X}\boldsymbol{\beta}^{(t)} + s(\boldsymbol{W}^{(t)})^{-1}(\boldsymbol{y} - \boldsymbol{p}^{(t)})\right] \\ &= (\boldsymbol{X}^\top \boldsymbol{W}^{(t)} \boldsymbol{X})^{-1} \boldsymbol{X}^\top \boldsymbol{W}^{(t)} \boldsymbol{z}^{(t)}, \end{aligned}$$

  where

$$\boldsymbol{z}^{(t)} = \boldsymbol{X}\boldsymbol{\beta}^{(t)} + s(\boldsymbol{W}^{(t)})^{-1}(\boldsymbol{y} - \boldsymbol{p}^{(t)})$$

are the working responses. A Newton's iteration is equivalent to solving a weighed least squares problem $\sum_{i=1}^n w_i(z_i - \boldsymbol{x}_i^\mathsf{T}\boldsymbol{\beta})^2$. Thus the name IRWLS (iteratively re-weighted least squares).



Common distributions with typical uses and canonical link functions

Distribution	Support of distribution	Typical uses	Link name	Link function	Mean function
Normal	real: $(-\infty, +\infty)$	Linear-response data	Identity	$\mathbf{X}\boldsymbol{\beta} = \mu$	$\mu = \mathbf{X}\boldsymbol{\beta}$
Exponential	real: $(0, +\infty)$	Exponential-response data, scale parameters	Inverse	$\mathbf{X}\boldsymbol{\beta} = \mu^{-1}$	$\mu = (\mathbf{X}\boldsymbol{\beta})^{-1}$
Gamma					
Inverse Gaussian	real: $(0, +\infty)$		Inverse squared	$\mathbf{X}\boldsymbol{\beta} = \mu^{-2}$	$\mu = (\mathbf{X}\boldsymbol{\beta})^{-1/2}$
Poisson	integer: $[0, +\infty)$	count of occurrences in fixed amount of time/space	Log	$\mathbf{X}\boldsymbol{\beta} = \ln(\mu)$	$\mu = \exp(\mathbf{X}\boldsymbol{\beta})$
Bernoulli	integer: $[0, 1]$	outcome of single yes/no occurrence	Logit	$\mathbf{X}\boldsymbol{\beta} = \ln\left(\frac{\mu}{1-\mu}\right)$	$\mu = \frac{\exp(\mathbf{X}\boldsymbol{\beta})}{1+\exp(\mathbf{X}\boldsymbol{\beta})} = \frac{1}{1+\exp(-\mathbf{X}\boldsymbol{\beta})}$
Binomial	integer: $[0, N]$	count of # of "yes" occurrences out of N yes/no occurrences			
Categorical	integer: $[0, K)$	outcome of single K-way occurrence			
	K-vector of integer: $[0, 1]$, where exactly one element in the vector has the value 1				
Multinomial	K-vector of integer: $[0, N]$	count of occurrences of different types (1 .. K) out of N total K-way occurrences			

Let's consider the more general class of generalized linear models (GLM).

- Y belongs to an exponential family with density

$$p(y|\theta, \phi) = \exp\left\{\frac{y\theta - b(\theta)}{a(\phi)} + c(y, \phi)\right\}.$$

θ: natural parameter. $\phi > 0$: dispersion parameter. GLM relates the mean $\mu = \mathbf{E}(Y|\boldsymbol{x})$ via a strictly increasing link function

$$g(\mu) = \eta = \boldsymbol{x}^\mathsf{T}\boldsymbol{\beta}, \quad \mu = g^{-1}(\eta)$$

- Score, Hessian, information

$$\begin{aligned}
\nabla L_n(\boldsymbol{\beta}) &= \sum_{i=1}^n \frac{(y_i - \mu_i)\mu_i'(\eta_i)}{\sigma_i^2}\boldsymbol{x}_i \\
-d^2 L_n(\boldsymbol{\beta}) &= \sum_{i=1}^n \frac{[\mu_i'(\eta_i)]^2}{\sigma_i^2}\boldsymbol{x}_i\boldsymbol{x}_i^{\mathsf{T}} - \sum_{i=1}^n \frac{(y_i - \mu_i)\theta''(\eta_i)}{\sigma_i^2}\boldsymbol{x}_i\boldsymbol{x}_i^{\mathsf{T}} \\
\boldsymbol{I}_n(\boldsymbol{\beta}) &= \mathbf{E}[-d^2 L_n(\boldsymbol{\beta})] = \sum_{i=1}^n \frac{[\mu_i'(\eta_i)]^2}{\sigma_i^2}\boldsymbol{x}_i\boldsymbol{x}_i^{\mathsf{T}} = \boldsymbol{X}^T\boldsymbol{W}\boldsymbol{X}.
\end{aligned}$$

- Fisher scoring method

$$\boldsymbol{\beta}^{(t+1)} = \boldsymbol{\beta}^{(t)} + s[\boldsymbol{I}(\boldsymbol{\beta}^{(t)})]^{-1}\nabla L_n(\boldsymbol{\beta}^{(t)})$$

  IRWLS with weights $w_i = [\mu_i(\eta_i)]^2/\sigma_i^2$ and some working responses z_i.

- For *canonical link*, $\theta = \eta$, the second term of Hessian vanishes and Hessian coincides with Fisher information matrix. Convex problem ☺

$$\text{Fisher's scoring} = \text{Newton's method}.$$

- Non-canonical link, non-convex problem ☹

$$\text{Fisher's scoring algorithm} \neq \text{Newton's method}.$$

  Example: Probit regression (binary response with probit link). $y_i \sim \text{Bernoulli}(p_i)$ and

$$p_i = \Phi(\boldsymbol{x}_i^T\boldsymbol{\beta}), \quad \eta_i = \boldsymbol{x}_i^T\boldsymbol{\beta} = \Phi^{-1}(p_i),$$

  where $\Phi(\cdot)$ is the cdf of a standard normal.

- `glmfit()` in R and MATLAB implements the Fisher scoring method, aka IRWLS, for GLMs.

20 Lecture 20, Nov 6

Announcements

- HW6 due today. Submit hardcopy and email code to me (`LastFirstHW6.R` or `LastFirstHW6.Rmd`)
- HW7 due date changed to Tue Nov 18

Last time

- Optimality conditions for constrained optimization (KKT)
- Application of Newton and Fisher scoring algorithm: GLM

Today

- Application of Newton and Fisher scoring algorithm: nonlinear regression
- EM algorithm

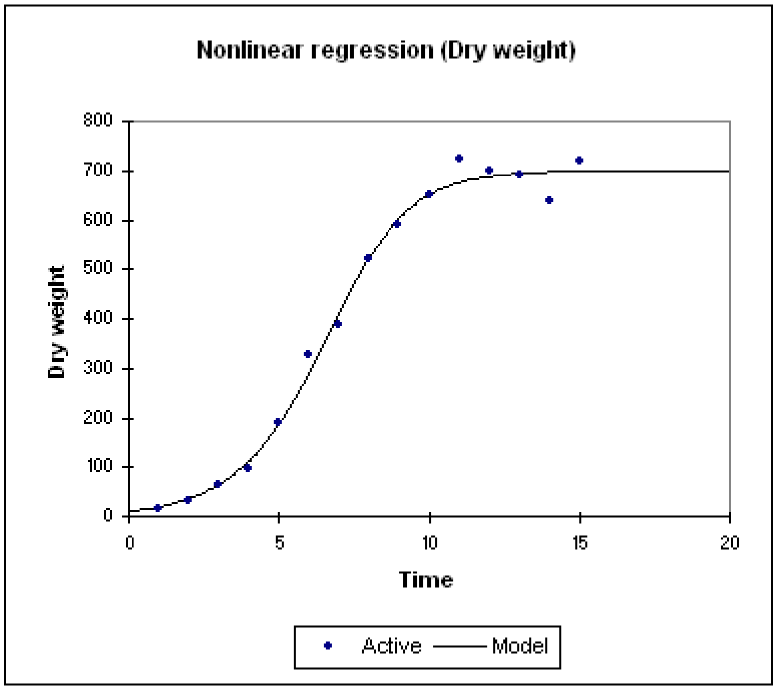
Nonlinear regression – Gauss-Newton method (KL 14.4-14.6, JM 9.8)

- Now we finally get to the problem Gauss faced in 1800!
  Relocate Ceres by fitting 41 observations to a 6-parameter (nonlinear) orbit.
- Nonlinear least squares (curve fitting):

  $$\text{minimize } f(\boldsymbol{\beta}) = \frac{1}{2}\sum_{i=1}^{n}[y_i - \mu_i(\boldsymbol{x}_i, \boldsymbol{\beta})]^2$$

  For example, y_i = dry weight of onion and x_i = growth time, and we want to fit a 3-parameter growth curve

  $$\mu(x, \beta_1, \beta_2, \beta_3) = \frac{\beta_3}{1 + e^{-\beta_1 - \beta_2 x}}.$$



- "Score" and "information matrices"

$$
\begin{aligned}
\nabla f(\boldsymbol{\beta}) &= -\sum_{i=1}^{n}[y_i - \mu_i(\boldsymbol{\beta})]\nabla\mu_i(\boldsymbol{\beta}) \\
d^2 f(\boldsymbol{\beta}) &= \sum_{i=1}^{n}\nabla\mu_i(\boldsymbol{\beta})d\mu_i(\boldsymbol{\beta}) - \sum_{i=1}^{n}[y_i - \mu_i(\boldsymbol{\beta})]d^2\mu_i(\boldsymbol{\beta}) \\
\boldsymbol{I}(\boldsymbol{\beta}) &= \sum_{i=1}^{n}\nabla\mu_i(\boldsymbol{\beta})d\mu_i(\boldsymbol{\beta}) = \boldsymbol{J}(\boldsymbol{\beta})^{\mathsf{T}}\boldsymbol{J}(\boldsymbol{\beta}),
\end{aligned}
$$

  where $\boldsymbol{J}(\boldsymbol{\beta})^{\mathsf{T}} = [\nabla\mu_1(\boldsymbol{\beta}), \ldots, \nabla\mu_n(\boldsymbol{\beta})] \in \mathbb{R}^{p\times n}$.

- *Gauss-Newton* (= "Fisher's scoring algorithm") uses $\boldsymbol{I}(\boldsymbol{\beta})$, which is always psd.

$$
\boxed{\boldsymbol{\beta}^{(t+1)} = \boldsymbol{\beta}^{(t)} + s\boldsymbol{I}(\boldsymbol{\beta}^{(t)})^{-1}\nabla L(\boldsymbol{\beta}^{(t)})}
$$

- *Levenberg-Marquardt* method, aka *damped least squares algorithm* (DLS), adds a ridge term to the approximate Hessian

$$
\boxed{\boldsymbol{\beta}^{(t+1)} = \boldsymbol{\beta}^{(t)} + s[\boldsymbol{I}(\boldsymbol{\beta}^{(t)}) + \tau\boldsymbol{I}_p]^{-1}\nabla L(\boldsymbol{\beta}^{(t)})}
$$

  bridging between Gauss-Newton and steepest descent.

- Other approximation to Hessians: nonlinear GLMs.
  See KL 14.4 for examples.

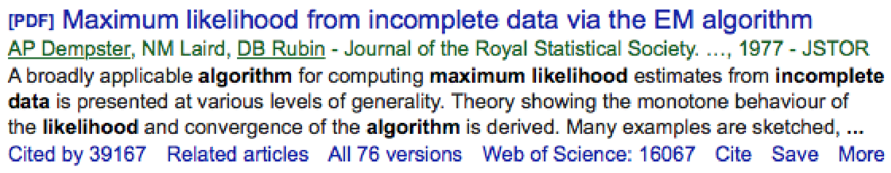
Which statistical papers are most cited?

Paper	Citations	Per Year
Kaplan-Meier (Kaplan and Meier, 1958)	44502	795
EM (Dempster et al., 1977a)	39167	**1059**
Cox model (Cox, 1972)	38036	906
Metropolis (Metropolis et al., 1953)	28093	461
FDR (Benjamini and Hochberg, 1995)	24093	**1268**
Unit root test (Dickey and Fuller, 1979)	15354	439
Lasso (Tibshirani, 1996)	11438	635
bootstrap (Efron, 1979)	10953	313
FFT (Cooley and Tukey, 1965)	10134	207
Gibbs sampler (Gelfand and Smith, 1990)	5901	246

- Citation counts from Google Scholar on 11/03/2014.
- EM is one of the most influential statistical ideas, finding applications in various branches of science.

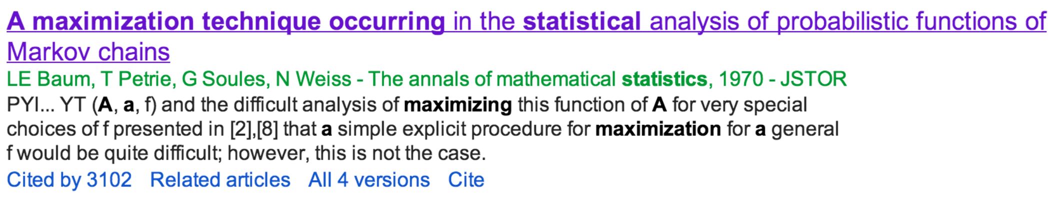
EM algorithm

- History: Dempster et al. (1977b).



[PDF] Maximum likelihood from incomplete data via the EM algorithm
AP Dempster, NM Laird, DB Rubin - Journal of the Royal Statistical Society. ..., 1977 - JSTOR
A broadly applicable **algorithm** for computing **maximum likelihood** estimates from **incomplete data** is presented at various levels of generality. Theory showing the monotone behaviour of the **likelihood** and convergence of the **algorithm** is derived. Many examples are sketched, ...
Cited by 39167 Related articles All 76 versions Web of Science: 16067 Cite Save More

Same idea appears in parameter estimation in HMM (Baum-Welch algorithm) (Baum et al., 1970).



A maximization technique occurring in the **statistical** analysis of probabilistic functions of Markov chains
LE Baum, T Petrie, G Soules, N Weiss - The annals of mathematical **statistics**, 1970 - JSTOR
PYI... YT (**A**, **a**, f) and the difficult analysis of **maximizing** this function of **A** for very special choices of f presented in [2],[8] that **a** simple explicit procedure for **maximization** for **a** general f would be quite difficult; however, this is not the case.
Cited by 3102 Related articles All 4 versions Cite

- Notations

- $\boldsymbol{Y}$: observed data
- $\boldsymbol{Z}$: missing data
- $\boldsymbol{X} = (\boldsymbol{Y}, \boldsymbol{Z})$: complete data

- Goal: maximize the log-likelihood of the observed data $\ln g(\boldsymbol{y}|\boldsymbol{\theta})$ (optimization!)
- Idea: choose $\boldsymbol{Z}$ such that MLE for the complete data is trivial.
- Let $f(\boldsymbol{x}|\boldsymbol{\theta}) = f(\boldsymbol{y}, \boldsymbol{z}|\boldsymbol{\theta})$ be the density of complete data
- Iterative two step procedure
  - E step: calculate the conditional expectation

$$Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)}) = \mathbf{E}_{\boldsymbol{Z}|\boldsymbol{Y}=\boldsymbol{y},\boldsymbol{\theta}^{(t)}}[\ln f(\boldsymbol{Y}, \boldsymbol{Z}|\boldsymbol{\theta}) \mid \boldsymbol{Y} = \boldsymbol{y}, \boldsymbol{\theta}^{(t)}]$$

  - M step: maximize $Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)})$ to generate the next iterate

$$\boldsymbol{\theta}^{(t+1)} = \text{argmax}_{\boldsymbol{\theta}}\, Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)})$$

- (Ascent property of EM algorithm) By the information inequality,

$$\begin{aligned}
& Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^{(t)}) - \ln g(\boldsymbol{y}|\boldsymbol{\theta}) \\
= \ & \mathbf{E}[\ln f(\boldsymbol{Y}, \boldsymbol{Z}|\boldsymbol{\theta})|\boldsymbol{Y} = \boldsymbol{y}, \boldsymbol{\theta}^{(t)}] - \ln g(\boldsymbol{y}|\boldsymbol{\theta}) \\
= \ & \mathbf{E}\left\{\ln\left[\frac{f(\boldsymbol{Y}, \boldsymbol{Z} \mid \boldsymbol{\theta})}{g(\boldsymbol{Y} \mid \boldsymbol{\theta})}\right] \mid \boldsymbol{Y} = \boldsymbol{y}, \boldsymbol{\theta}^{(t)}\right\} \\
\leq \ & \mathbf{E}\left\{\ln\left[\frac{f(\boldsymbol{Y}, \boldsymbol{Z} \mid \boldsymbol{\theta}^{(t)})}{g(\boldsymbol{Y} \mid \boldsymbol{\theta}^{(t)})}\right] \mid \boldsymbol{Y} = \boldsymbol{y}, \boldsymbol{\theta}^{(t)}\right\} \\
= \ & Q(\boldsymbol{\theta}^{(t)} \mid \boldsymbol{\theta}^{(t)}) - \ln g(\boldsymbol{y}|\boldsymbol{\theta}^{(t)}).
\end{aligned}$$

  Rearranging shows that (fundamental inequality of EM)

$$\ln g(\boldsymbol{y} \mid \boldsymbol{\theta}) \geq Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^{(t)}) - Q(\boldsymbol{\theta}^{(t)} \mid \boldsymbol{\theta}^{(t)}) + \ln g(\boldsymbol{y} \mid \boldsymbol{\theta}^{(t)})$$

  for all $\boldsymbol{\theta}$ and in particular

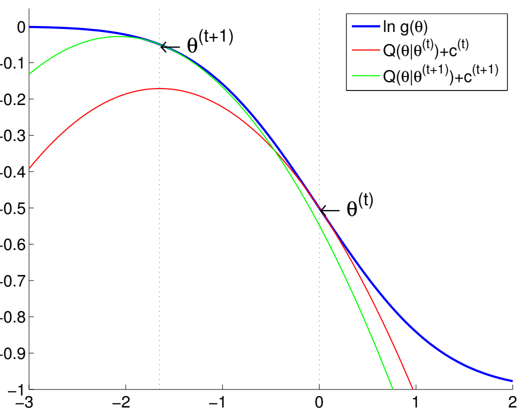
$$\begin{aligned}
\ln g(\boldsymbol{y} \mid \boldsymbol{\theta}^{(t+1)}) & \geq Q(\boldsymbol{\theta}^{(t+1)} \mid \boldsymbol{\theta}^{(t)}) - Q(\boldsymbol{\theta}^{(t)} \mid \boldsymbol{\theta}^{(t)}) + \ln g(\boldsymbol{y} \mid \boldsymbol{\theta}^{(t)}) \\
& \geq \ln g(\boldsymbol{y} \mid \boldsymbol{\theta}^{(t)}).
\end{aligned}$$

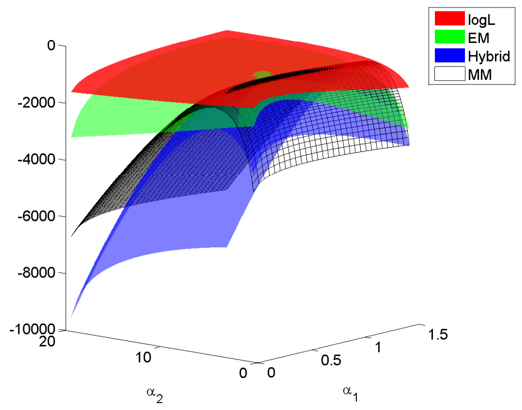
Obviously we only need

$$Q(\boldsymbol{\theta}^{(t+1)} \mid \boldsymbol{\theta}^{(t)}) - Q(\boldsymbol{\theta}^{(t)} \mid \boldsymbol{\theta}^{(t)}) \geq 0$$

for this ascent property to hold (*generalized EM*).

- Intuition? Keep these pictures in mind

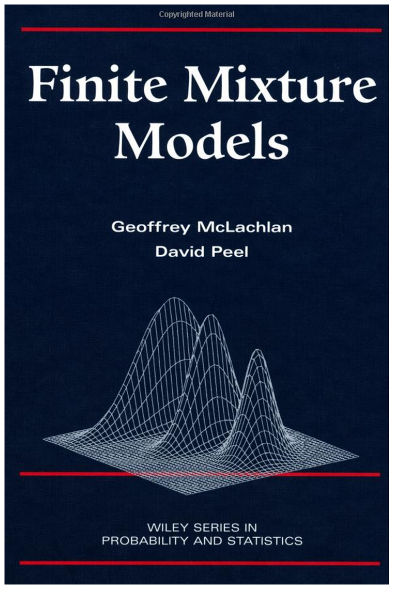


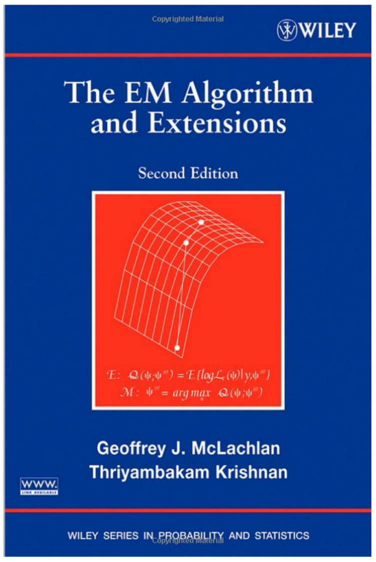


- Under mild regularity conditions, $\boldsymbol{\theta}^{(t)}$ converges to a stationary point of $\ln g(\boldsymbol{y}|\boldsymbol{\theta})$.

- Numerous applications of EM:
  finite mixture model, HMM (Baum-Welch algorithm), factor analysis, variance components model aka linear mixed model (LMM), hyper-parameter estimation in empirical Bayes procedure $\max_{\boldsymbol{\alpha}} \int f(\boldsymbol{y}|\boldsymbol{\theta})\pi(\boldsymbol{\theta}|\boldsymbol{\alpha})\, d\boldsymbol{\theta}$ (e.g., HW6/7), missing data, group/censorized/truncated model, ...

A canonical example: finite mixture models





- Gaussian finite mixture models: mixture density

$$h(\boldsymbol{y}) = \sum_{j=1}^{k} \pi_j h_j(\boldsymbol{y} \mid \boldsymbol{\mu}_j, \boldsymbol{\Omega}_j), \quad \boldsymbol{y} \in \mathbb{R}^d,$$

  where

$$h_j(\boldsymbol{y} \mid \boldsymbol{\mu}_j, \boldsymbol{\Omega}_j) = \left(\frac{1}{2\pi}\right)^{d/2} |\det(\boldsymbol{\Omega}_j)|^{-1/2} e^{-\frac{1}{2}(\boldsymbol{y}-\boldsymbol{\mu}_j)^{\mathsf{T}} \boldsymbol{\Omega}_j^{-1} (\boldsymbol{y}-\boldsymbol{\mu}_j)}$$

  are multivariate normals $N_d(\boldsymbol{\mu}_j, \boldsymbol{\Omega}_j)$.

- Given data $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n$, want to estimate parameters

$$\boldsymbol{\theta} = (\pi_1, \ldots, \pi_k, \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k, \boldsymbol{\Omega}_1, \ldots, \boldsymbol{\Omega}_k).$$

  (Incomplete) data log-likelihood is

$$\ln g(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n | \boldsymbol{\theta}) = \sum_{i=1}^{n} \ln h(\boldsymbol{y}_i) = \sum_{i=1}^{n} \ln \sum_{j=1}^{k} \pi_j h_j(\boldsymbol{y}_i \mid \boldsymbol{\mu}_j, \boldsymbol{\Omega}_j).$$

- Let $z_{ij} = I\{\boldsymbol{y}_i \text{ comes from group } j\}$. Complete data likelihood is

$$f(\boldsymbol{y}, \boldsymbol{z}|\boldsymbol{\theta}) = \prod_{i=1}^n \prod_{j=1}^k [\pi_j h_j(\boldsymbol{y}_i|\boldsymbol{\mu}_j, \boldsymbol{\Omega}_j)]^{z_{ij}}$$

and thus complete log-likelihood is

$$\ln f(\boldsymbol{y}, \boldsymbol{z}|\boldsymbol{\theta}) = \sum_{i=1}^n \sum_{j=1}^k z_{ij}[\ln \pi_j + \ln h_j(\boldsymbol{y}_i|\boldsymbol{\mu}_j, \boldsymbol{\Omega}_j)].$$

- E step: need to evaluate conditional expectation

$$\begin{aligned} & Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)}) \\ = \;& \mathbf{E}\left\{\sum_{i=1}^n \sum_{j=1}^k z_{ij}[\ln \pi_j + \ln h_j(\boldsymbol{y}_i|\boldsymbol{\mu}_j, \boldsymbol{\Omega}_j) \mid \boldsymbol{Y} = \boldsymbol{y}, \boldsymbol{\pi}^{(t)}, \boldsymbol{\mu}_1^{(t)}, \ldots, \boldsymbol{\mu}_k^{(t)}, \boldsymbol{\Omega}_1^{(t)}, \ldots, \boldsymbol{\Omega}_k^{(t)}]\right\}. \end{aligned}$$

By Bayes rule, we have

$$\begin{aligned} w_{ij}^{(t)} &:= \mathbf{E}[z_{ij} \mid \boldsymbol{y}, \boldsymbol{\pi}^{(t)}, \boldsymbol{\mu}_1^{(t)}, \ldots, \boldsymbol{\mu}_k^{(t)}, \boldsymbol{\Omega}_1^{(t)}, \ldots, \boldsymbol{\Omega}_k^{(t)}] \\ &= \frac{\pi_j^{(t)} h_j(\boldsymbol{y}_i|\boldsymbol{\mu}_j^{(t)}, \boldsymbol{\Omega}_j^{(t)})}{\sum_{j'=1}^k \pi_{j'}^{(t)} h_{j'}(\boldsymbol{y}_i|\boldsymbol{\mu}_{j'}^{(t)}, \boldsymbol{\Omega}_{j'}^{(t)})}. \end{aligned}$$

So the Q function becomes

$$\begin{aligned} & Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)}) \\ = \;& \sum_{i=1}^n \sum_{j=1}^k w_{ij}^{(t)} \ln \pi_j + \sum_{i=1}^n \sum_{j=1}^k w_{ij}^{(t)} \left[-\frac{1}{2} \ln \det \boldsymbol{\Omega}_j - \frac{1}{2}(\boldsymbol{y}_i - \boldsymbol{\mu}_j)^\intercal \boldsymbol{\Omega}_j^{-1}(\boldsymbol{y}_i - \boldsymbol{\mu}_j)\right]. \end{aligned}$$

- M step: maximizer of the Q function gives the next iterate

$$\begin{aligned} \pi_j^{(t+1)} &= \frac{\sum_i w_{ij}^{(t)}}{n} \\ \boldsymbol{\mu}_j^{(t+1)} &= \frac{\sum_{i=1}^n w_{ij}^{(t)} \boldsymbol{y}_i}{\sum_{i=1}^n w_{ij}^{(t)}} \\ \boldsymbol{\Omega}_j^{(t+1)} &= \frac{\sum_{i=1}^n w_{ij}^{(t)} (\boldsymbol{y}_i - \boldsymbol{\mu}_j^{(t+1)})(\boldsymbol{y}_i - \boldsymbol{\mu}_j^{(t+1)})^\intercal}{\sum_i w_{ij}^{(t)}}. \end{aligned}$$

See KL Example 11.3.1 for multinomial MLE. See KL Example 11.2.3 for multivariate normal MLE.

- Compare these extremely simple updates to Newton type algorithms!
- Also note the ease of parallel computing with this EM algorithm. See, e.g., Suchard, M. A.; Wang, Q.; Chan, C.; Frelinger, J.; Cron, A. & West, M. Understanding GPU programming for statistical computation: studies in massively parallel massive mixtures. *Journal of Computational and Graphical Statistics*, 2010, 19, 419-438.
- In general, EM/MM algorithms are particularly attractive for parallel computing. See, e.g.,
  H Zhou, K Lange, & M Suchard. (2010) Graphical processing units and high-dimensional optimization, *Statistical Science*, 25:311-324.

21 Lecture 21, Nov 11

Announcements

- HW6 returned. Sketch of solution: `http://hua-zhou.github.io/teaching/st758-2014fall/hw06sol.html`
- HW7 due next Tue Nov 18
- HW8 due Nov 25

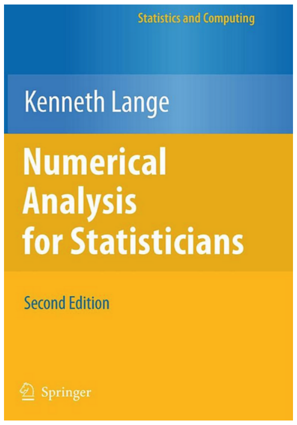
Last time

- Nonlinear regression (Gauss-Newton algorithm)
- EM algorithm
- Example: finite mixture model

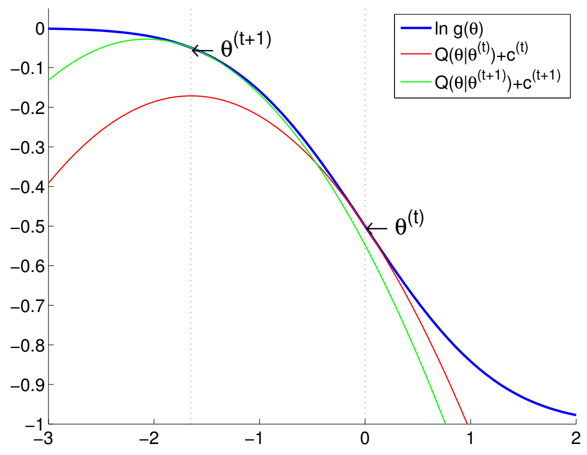
Today

- MM algorithm

MM algorithm (KL Ch12)



- Recall our picture for understanding the ascent property of EM



- EM as a minorization-maximization (MM) algorithm
  - The Q function constitutes a *minorizing* function of the objective function up to an additive constant

$$\begin{aligned} L(\boldsymbol{\theta}) &\geq Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)}) + c^{(t)} \quad \text{for all } \boldsymbol{\theta} \\ L(\boldsymbol{\theta}^{(t)}) &= Q(\boldsymbol{\theta}^{(t)}|\boldsymbol{\theta}^{(t)}) + c^{(t)} \end{aligned}$$

  - *Maximizing* the Q function generates an ascent iterate $\boldsymbol{\theta}^{(t+1)}$

- Questions:
  - Is EM principle only limited to maximizing likelihood model?
  - Is there any other way to produce such surrogate function?
  - Can we flip the picture and apply same principle to *minimization* problem?

These thoughts lead to a powerful tool – MM principle (Lange et al., 2000). Lange, K., Hunter, D. R., and Yang, I. (2000). Optimization transfer using surrogate objective functions. *J. Comput. Graph. Statist.*, 9(1):159. With discussion, and a rejoinder by Hunter and Lange.

- For maximization of $f(\boldsymbol{\theta})$ – minorization-maximization (MM) algorithm
  - **M**inorization step: Construct a surrogate function $g(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)})$ such that

$$\begin{aligned} f(\boldsymbol{\theta}) &\geq g(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)}) \quad \text{(dominance condition)} \\ f(\boldsymbol{\theta}^{(t)}) &= g(\boldsymbol{\theta}^{(t)}|\boldsymbol{\theta}^{(t)}) \quad \text{(tangent condition).} \end{aligned}$$

- **M**aximization step:

$$\boldsymbol{\theta}^{(t+1)} = \text{argmax}\, g(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)}).$$

- *Ascent* property of minorization-maximization algorithm

$$f(\boldsymbol{\theta}^{(t)}) = g(\boldsymbol{\theta}^{(t)}|\boldsymbol{\theta}^{(t)}) \leq g(\boldsymbol{\theta}^{(t+1)}|\boldsymbol{\theta}^{(t)}) \leq f(\boldsymbol{\theta}^{(t+1)}).$$

- EM is a special case of minorization-maximization (MM) algorithm.

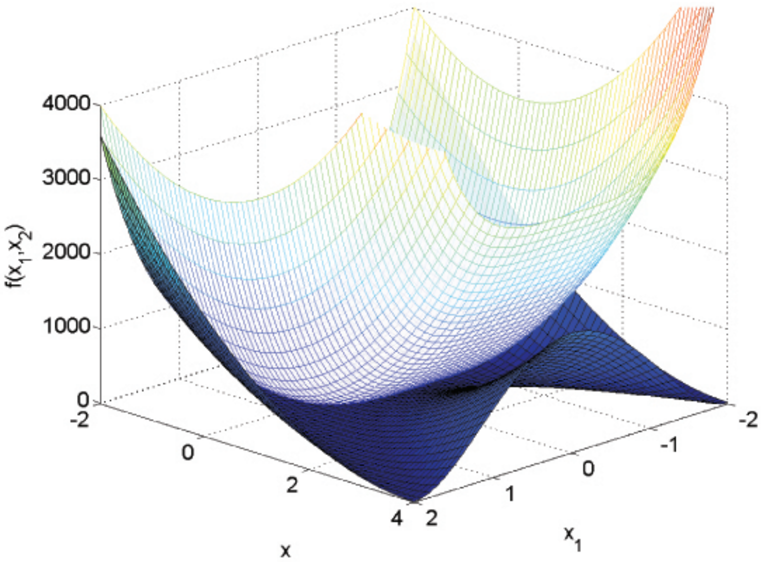
- For minimization of $f(\boldsymbol{\theta})$ – majorization-minimization (MM) algorithm

  - **M**ajorization step: Construct a surrogate function $g(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)})$ such that

$$\begin{aligned} f(\boldsymbol{\theta}) &\leq g(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)}) \quad \text{(dominance condition)} \\ f(\boldsymbol{\theta}^{(t)}) &= g(\boldsymbol{\theta}^{(t)}|\boldsymbol{\theta}^{(t)}) \quad \text{(tangent condition).} \end{aligned}$$

  - **M**inimization step:

$$\boldsymbol{\theta}^{(t+1)} = \text{argmin}\, g(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)}).$$

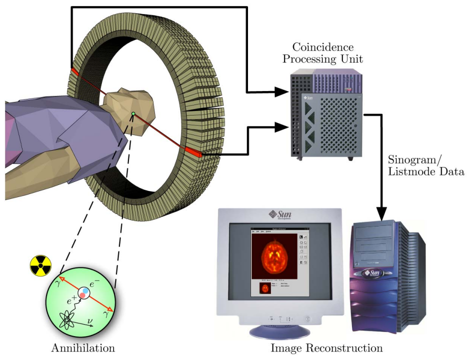


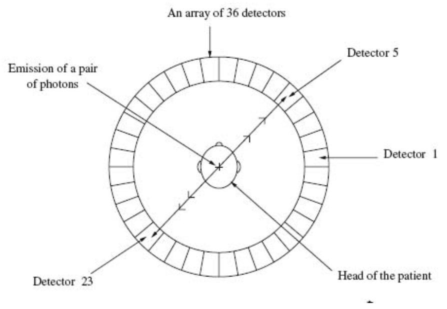
- Similarly we have the *descent* property of majorization-minimization algorithm.

- Same convergence theory as EM.

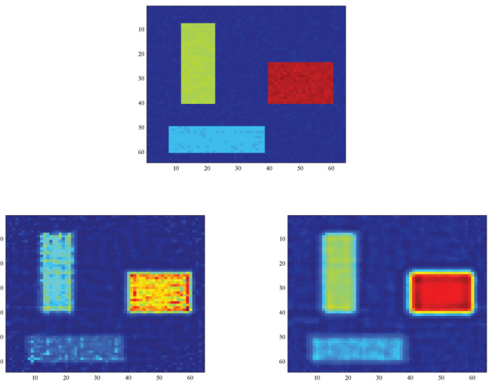
- Rational of the MM principle for developing optimization algorithms
  - Free the derivation from missing data structure.
  - Avoid matrix inversion.
  - Linearize an optimization problem.
  - Deal gracefully with certain equality and inequality constraints.
  - Turn a non-differentiable problem into a smooth problem.
  - Separate the parameters of a problem (perfect for massive, *fine-scale* parallelization).
- Generic methods for majorization and minorization – *inequalities*
  - Jensen's (information) inequality – EM algorithms
  - The Cauchy-Schwartz inequality - multidimensional scaling
  - Supporting hyperplane property of a convex function
  - Arithmetic-geometric mean inequality
  - Quadratic upper bound principle - Böhning and Lindsay
  - ...
- Numerous examples in KL chapter 12.
- History: the name *MM algorithm* originates from the discussion (by Xiaoli Meng) and rejoinder of the Lange et al. (2000) paper.

Example: 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 $\lambda_j \geq 0$.
- Which algorithm?
  - Fisher scoring (IRWLS) = Newton.
    Need to solve a large linear system at each iteration ☹
  - EM algorithm: Lange and Carson (1984), Vardi et al. (1985)
$$\lambda_j^{(t+1)} = \lambda_j^{(t)} \sum_i \frac{y_i c_{ij}}{\sum_k c_{ik}\lambda_k^{(t)}}.$$
    Scales well with data size. Converges to the global maximum under mild conditions.
  - Exercise: derive the EM algorithm. (Hint: missing data z_{ij} = # of photons emitted from pixel i and received by detector j.)
- 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.

- EM algorithm does not (or is hard to) apply to the regularization term. Let's derive an MM algorithm.

- Minorization step:

  - By concavity of the $\ln s$ function

$$
\begin{aligned}
\ln \left(\sum_j c_{ij}\lambda_j \right) &= \ln \left(\sum_j \frac{c_{ij}\lambda_j^{(t)}}{\sum_j c_{ij'}\lambda_{j'}^{(t)}} \cdot \frac{\sum_{j'} c_{ij'}\lambda_{j'}^{(t)}}{c_{ij}\lambda_j^{(t)}} \cdot c_{ij}\lambda_j \right) \\
&\geq \sum_j \frac{c_{ij}\lambda_j^{(t)}}{\sum_j c_{ij'}\lambda_{j'}^{(t)}} \ln \left(\frac{\sum_{j'} c_{ij'}\lambda_{j'}^{(t)}}{c_{ij}\lambda_j^{(t)}} c_{ij}\lambda_j \right) \\
&= \sum_j \frac{c_{ij}\lambda_j^{(t)}}{\sum_j c_{ij'}\lambda_{j'}^{(t)}} \ln \lambda_j + c^{(t)}.
\end{aligned}
$$

    Remark: this minorization depends on the positivity of both c_{ij} and λ_j.

  - By concavity of the $-s^2$ function

$$
\begin{aligned}
-(\lambda_j - \lambda_k)^2 &= -\left(\frac{2\lambda_j - \lambda_j^{(t)} - \lambda_k^{(t)}}{2} + \frac{-2\lambda_k + \lambda_j^{(t)} + \lambda_k^{(t)}}{2} \right)^2 \\
&\geq -\frac{1}{2}(2\lambda_j - \lambda_j^{(t)} - \lambda_k^{(t)})^2 - \frac{1}{2}(2\lambda_k - \lambda_j^{(t)} - \lambda_k^{(t)})^2.
\end{aligned}
$$

  - Combining minorizing terms gives an overall surrogate function

$$
\begin{aligned}
g(\boldsymbol{\lambda}|\boldsymbol{\lambda}^{(t)}) &= \sum_i y_i \sum_j \frac{c_{ij}\lambda_j^{(t)}}{\sum_{j'} c_{ij'}\lambda_{j'}^{(t)}} \ln \lambda_j - \sum_i \sum_j c_{ij}\lambda_j \\
& -\frac{\mu}{4} \sum_{\{j,k\}\in\mathcal{N}} [(2\lambda_j - \lambda_j^{(t)} - \lambda_k^{(t)})^2 + (2\lambda_k - \lambda_j^{(t)} - \lambda_k^{(t)})^2].
\end{aligned}
$$

- Maximization step:
  - $g(\boldsymbol{\lambda}|\boldsymbol{\lambda}^{(t)})$ is trivial to maximize because all λ_j are separated!
  - Solving for the root of

$$\begin{aligned} & \frac{\partial}{\partial \lambda_j} g(\boldsymbol{\lambda}|\boldsymbol{\lambda}^{(t)}) \\ = & \left(\sum_i y_i \frac{c_{ij}\lambda_j^{(t)}}{\sum_{j'} c_{ij}\lambda_{j'}^{(t)}} \right) \lambda_j^{-1} - \sum_i c_{ij} - \mu \sum_{k \in \mathcal{N}_j} (2\lambda_j - \lambda_j^{(t)} - \lambda_k^{(t)}) \\ = & \ 0 \end{aligned}$$

    gives $\lambda_j^{(t+1)}$.

- 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.

- 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: i7 @ 3.20GHZ (1 thread), implemented using BLAS in the GNU Scientific Library (GSL)
  GPU: NVIDIA GeForce GTX 580, implemented using cuBLAS

	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

22 Lecture 22, Nov 13

Announcements

- HW7 due next Tue Nov 18
- HW8 due Nov 25
- HW9 (simulation project) will be posted this week. Due Dec 9, 2014 @ 11A.

Last time

- MM algorithm
- Example: PET imaging

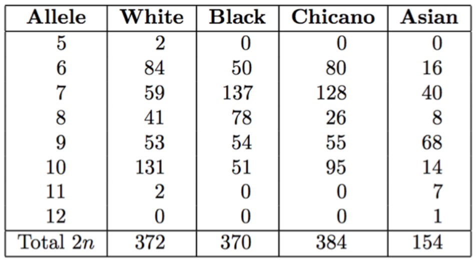
Today

- HW6 review
- Principles of Monte carlo simulation studies
- EM/MM example: Netflix problem

Feedback on HW6

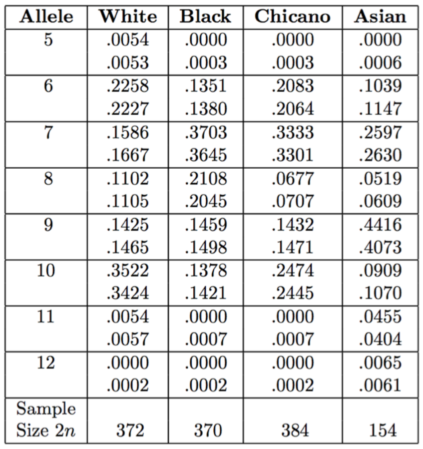
- Who cares?

TABLE 3.4. Allele Counts in Four Subpopulations



Allele	White	Black	Chicano	Asian
5	2	0	0	0
6	84	50	80	16
7	59	137	128	40
8	41	78	26	8
9	53	54	55	68
10	131	51	95	14
11	2	0	0	7
12	0	0	0	1
Total $2n$	372	370	384	154

TABLE 3.5. Classical and Bayesian Allele Frequency Estimates



Allele	White	Black	Chicano	Asian
5	.0054	.0000	.0000	.0000
	.0053	.0003	.0003	.0006
6	.2258	.1351	.2083	.1039
	.2227	.1380	.2064	.1147
7	.1586	.3703	.3333	.2597
	.1667	.3645	.3301	.2630
8	.1102	.2108	.0677	.0519
	.1105	.2045	.0707	.0609
9	.1425	.1459	.1432	.4416
	.1465	.1498	.1471	.4073
10	.3522	.1378	.2474	.0909
	.3424	.1421	.2445	.1070
11	.0054	.0000	.0000	.0455
	.0057	.0007	.0007	.0404
12	.0000	.0000	.0000	.0065
	.0002	.0002	.0002	.0061
Sample Size $2n$	372	370	384	154

- Bayesian approach to estimate a multinomial parameter $\boldsymbol{p}$
  - Data likelihood: $\boldsymbol{x}|\boldsymbol{p} \sim \text{multinomial}(\boldsymbol{p})$
  - Prior: $\boldsymbol{p} \sim \text{Dirichlet}(\boldsymbol{\alpha})$
  - Posterior: $\boldsymbol{p}|\boldsymbol{x} \sim \text{Dirichlet}(\boldsymbol{\alpha} + \boldsymbol{x})$
  - Do estimation and inference of $\boldsymbol{p}$ based on the posterior distribution

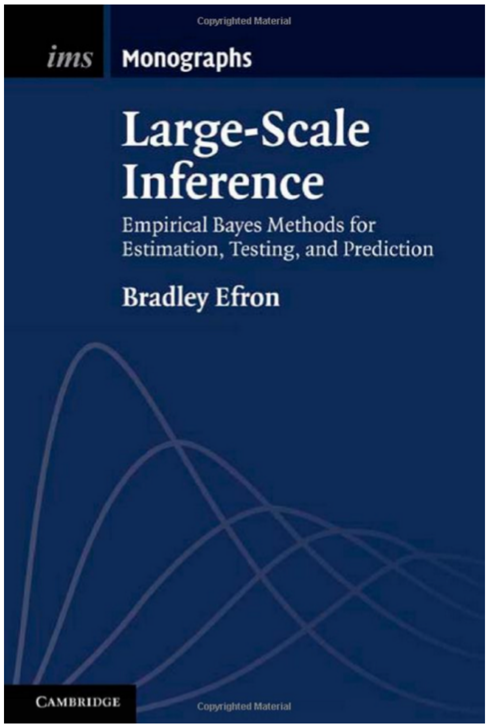
  But what value of $\boldsymbol{\alpha}$ to use in the prior?

- Empirical Bayes idea:
  - Estimate $\boldsymbol{\alpha}$ from data, say by maximizing the marginal likelihood of

$$\boldsymbol{x}_i \sim \text{DirMult}(\boldsymbol{\alpha}).$$

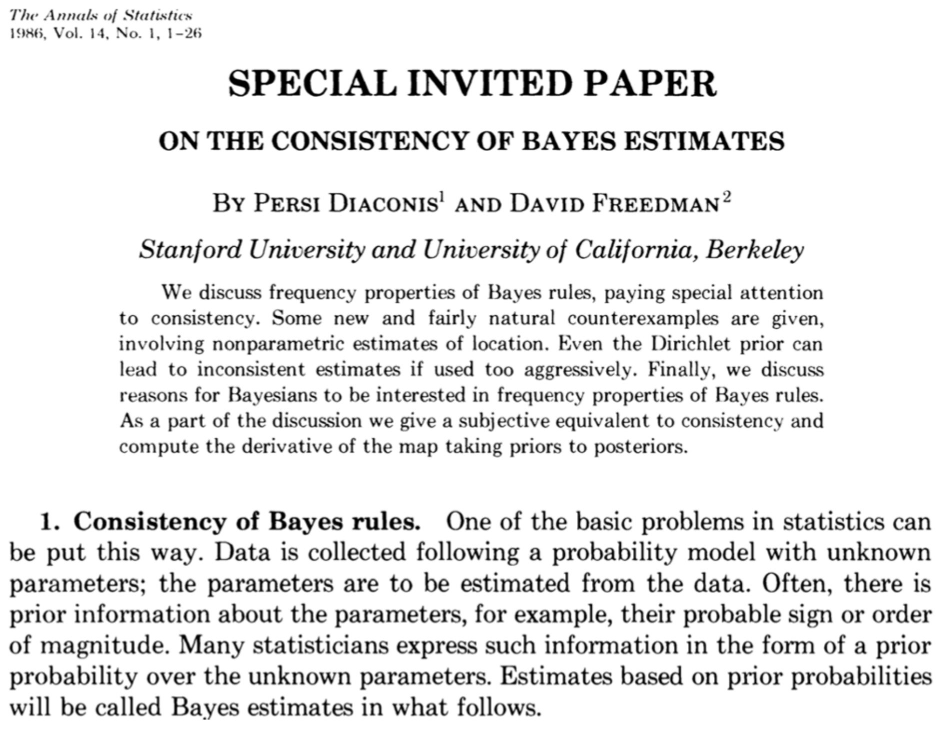
  - Then estimate $\boldsymbol{p}$ by posterior mean $\hat{\boldsymbol{p}}_{\text{EB}} = (\boldsymbol{x} + \hat{\boldsymbol{\alpha}})/(|\boldsymbol{x}| + |\hat{\boldsymbol{\alpha}}|)$.

  For a celebrated binomial estimation problem (batting averages of major league baseball players), see Efron and Morris (1973, 1977).



- The Bayes estimate (under certain conditions) enjoys both good asymptotic and finite sample properties.

  "borrow information across populations", "shrinkage", "learning from the experience of the others"



The Annals of Statistics
1986, Vol. 14, No. 1, 1–26

SPECIAL INVITED PAPER

ON THE CONSISTENCY OF BAYES ESTIMATES

BY PERSI DIACONIS[1] AND DAVID FREEDMAN[2]

Stanford University and University of California, Berkeley

We discuss frequency properties of Bayes rules, paying special attention to consistency. Some new and fairly natural counterexamples are given, involving nonparametric estimates of location. Even the Dirichlet prior can lead to inconsistent estimates if used too aggressively. Finally, we discuss reasons for Bayesians to be interested in frequency properties of Bayes rules. As a part of the discussion we give a subjective equivalent to consistency and compute the derivative of the map taking priors to posteriors.

1. Consistency of Bayes rules. One of the basic problems in statistics can be put this way. Data is collected following a probability model with unknown parameters; the parameters are to be estimated from the data. Often, there is prior information about the parameters, for example, their probable sign or order of magnitude. Many statisticians express such information in the form of a prior probability over the unknown parameters. Estimates based on prior probabilities will be called Bayes estimates in what follows.

- Machine learning applications: Handwritten digit recognition, text mining, email spam detection, ...

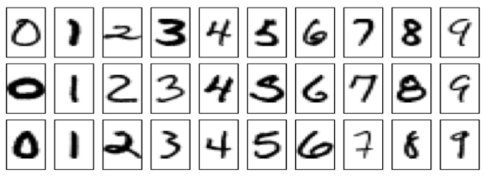


FIGURE 13.9. *Examples of grayscale images of handwritten digits.*

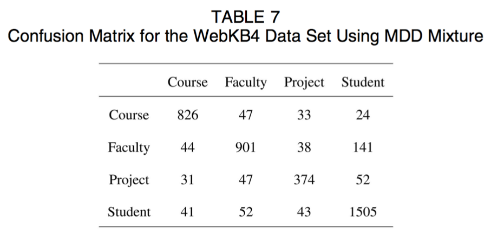


TABLE 7
Confusion Matrix for the WebKB4 Data Set Using MDD Mixture

	Course	Faculty	Project	Student
Course	826	47	33	24
Faculty	44	901	38	141
Project	31	47	374	52
Student	41	52	43	1505

- In HW6/HW7, we estimate $\boldsymbol{\alpha}$ by MLE.

- HW6 (Newton's method)

- Check solution. It's not optimal but you may learn things. `http://hua-zhou.github.io/teaching/st758-2014fall/hw06sol.html`
- Implementation of density function: deal with $\ln\Gamma(0) - \ln\Gamma(0) = \infty - \infty = NaN$ issue. Is the first formulation more efficient for computation (log instead of log gamma evaluation)?
- Implementation of MLE function. Allowing observation weights is always a good idea.
- Implementation of gradient/score function. Is the first formulation more efficient (free of digamma evaluations)?
- Special structure in the observed information matrix

  $$-d^2 L(\boldsymbol{\alpha}) = \boldsymbol{D} - c\mathbf{1}\mathbf{1}^T,$$

  where $\boldsymbol{D} = \text{diag}(d_1, \ldots, d_d)$ and

  $$\begin{aligned} d_j &= \sum_{i=1}^{n} \sum_{k=0}^{x_{ij}-1} \frac{1}{(\alpha_j + k)^2}, \quad j = 1, \ldots, d, \\ c &= \sum_{i=1}^{n} \sum_{k=0}^{|\boldsymbol{x}_i|} = \frac{1}{(|\boldsymbol{\alpha}| + k)^2}. \end{aligned}$$

  Is the first formulation more efficient (free of trigamma evaluations)?
- By Sherman-Morrison

  $$[-d^2 L(\boldsymbol{\alpha})]^{-1} = \boldsymbol{D}^{-1} + \frac{1}{c^{-1} - \sum_i d_i^{-1}} (\boldsymbol{D}^{-1}\mathbf{1})(\mathbf{1}^T \boldsymbol{D}^{-1}),$$

  which is pd if and only if $c^{-1} > \sum_i d_i^{-1}$.
- Ad hoc approximation of Hessian: use $\tilde{c} = 0.95(\sum_i d_i^{-1})^{-1}$ if it's not satisfied.
- Newton's direction is computed fast: $O(d)$ flops. No need to store any matrix! Why do I still see `solve(D)` or matrix-vector multiplication?
- How to calculate Fisher information matrix?
- Deal with boundary constraint. Choose initial step size s such that all components of

  $$\boldsymbol{\alpha}^{(t)} + s\Delta\boldsymbol{\alpha}$$

  are positive.

- What's that damn MoM starting point? If $\boldsymbol{P} = (P_1, \ldots, P_d)$ is from Dirichlet with parameter $\boldsymbol{\alpha}$, then

  $$\mathbf{E}(P_j) = \frac{\alpha_j}{|\boldsymbol{\alpha}|}, \quad \mathbf{E}(P_j^2) = \frac{\alpha_j(\alpha_j+1)}{|\boldsymbol{\alpha}|(|\boldsymbol{\alpha}|+1)}.$$

  Therefore

  $$\sum_j \frac{\mathbf{E}(P_j^2)}{\mathbf{E}(P_j)} = \frac{|\boldsymbol{\alpha}|+d}{|\boldsymbol{\alpha}|+1}.$$

  We estimate the left hand side by

  $$\hat{\rho} := \sum_j \frac{\sum_i (x_{ij}/|\boldsymbol{x}_i|)^2}{\sum_i (x_{ij}/|\boldsymbol{x}_i|)}$$

  and then solve for $|\boldsymbol{\alpha}^{(0)}| = (d-\hat{\rho})/(\hat{\rho}-1)$. Then we initialize

  $$\alpha_j^{(0)} = |\boldsymbol{\alpha}^{(0)}| \frac{\sum_i x_{ij}}{\sum_i |\boldsymbol{x}_i|} = \left(\frac{d-\hat{\rho}}{\hat{\rho}-1}\right)\left(\frac{\sum_i x_{ij}}{\sum_i |\boldsymbol{x}_i|}\right).$$

  Is there any proof $(d-\hat{\rho})/(\hat{\rho}-1)$ is always positive? Anyway we just need a good starting point. Other heuristics should also work.

- HW7: EM/MM algorithm for maximizing

  $$L(\boldsymbol{\alpha}) = \sum_{i=1}^n \ln \binom{|\boldsymbol{x}_i|}{\boldsymbol{x}_i} + \sum_{i=1}^n \sum_{j=1}^d \sum_{k=0}^{x_{ij}-1} \ln(\alpha_j + k) - \sum_{i=1}^n \sum_{k=0}^{|\boldsymbol{x}_i|-1} \ln(|\boldsymbol{\alpha}| + k)$$

  Hint: Apply Jensen's inequality to $\ln(\alpha_j + k)$ and supporting hyperplane inequality to $\ln(|\boldsymbol{\alpha}| + k)$.

- Examining Newton iterates, we are delighted to see its fast (quadratic) convergence. In HW7, you might see another scenario (linear convergence) for MM iterates. On the other hand, for machine learning problem, do we really need a super-accurate solution?

- Main messages:

  - basics of Newton's method (pd approximation, line search)
  - examine and exploit problem structure whenever possible

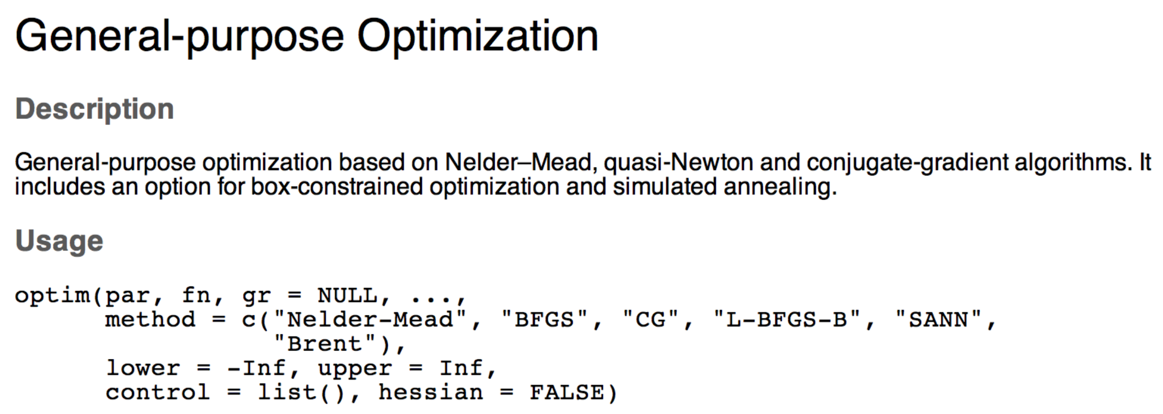
  - EM and MM turn out to be different
  - Comparing Newton versus MM: number of iterations, human efforts, efficiency, ...

> (Optimization is a) fascinating blend of theory and computation, heuristics and rigor.
>
> Roger Fletcher

> There is simply no such thing as a universal 'gold standard' when it comes to algorithms.
>
> Unknown Reviewer

- Implementation details: `lgamma`, `digamma`, `trigamma`, `psigamma` in R are vectorized function, vectorize code (numerical linear algebra is preferred over `apply()`), ...

Some black-box optimization routines in R



General-purpose Optimization

Description

General-purpose optimization based on Nelder–Mead, quasi-Newton and conjugate-gradient algorithms. It includes an option for box-constrained optimization and simulated annealing.

Usage

```
optim(par, fn, gr = NULL, ...,
      method = c("Nelder-Mead", "BFGS", "CG", "L-BFGS-B", "SANN",
                 "Brent"),
      lower = -Inf, upper = Inf,
      control = list(), hessian = FALSE)
```

Optimization using PORT routines

Description

Unconstrained and box-constrained optimization using PORT routines.

Usage

```
nlminb(start, objective, gradient = NULL, hessian = NULL, ...,
       scale = 1, control = list(), lower = -Inf, upper = Inf)
```

23 Lecture 23, Nov 18

Announcements

- HW7 (EM/MM for Dirichlet-multinomial) due today: submit hardcopy + email code (`LastFirstHW7.R` or `LastFirstHW7.Rmd`)
- HW8 (ranking MLB teams) due next Tue Nov 25
- HW9 (Monte Carlo simulation project, 100 pts) posted. Due Dec 9 @ 11A. Send both your report (pdf) and code (`R` or `Rmd`) by email.
  - Read posted course materials
  - Start early and send me your questions or draft by Nov 30
  - Discussion on Dec 2 (last lecture)

Last time

- HW6 review (Newton's method in action)

Today

- Last EM/MM example: Netflix problem

Example: Netflix and matrix completion

- Snapshot of the kind of data collected by Netflix. Only 100,480,507 ratings (1.2% entries of the 480K-by-18K matrix) are observed

ID	movie 1	movie 2	movie 3	movie 4	movie 5	movie 6	...	movie 17,770
user 1	5	3	4	3	3	*NA*	...	1
user 2	4	*NA*	*NA*	*NA*	*NA*	*NA*	...	*NA*
user 3	*NA*	*NA*	*NA*	*NA*	*NA*	*NA*	...	*NA*
user 4	4	*NA*	*NA*	*NA*	*NA*	2	...	4
user 5	*NA*	*NA*	*NA*	5	*NA*	*NA*	...	*NA*
user 6	3	*NA*	*NA*	5	1	*NA*	...	3
user 7	*NA*	*NA*	*NA*	*NA*	*NA*	*NA*	...	*NA*
user 8	5	*NA*	5	*NA*	*NA*	*NA*	...	*NA*
user 9	*NA*	*NA*	*NA*	*NA*	3	*NA*	...	*NA*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
user 480,189	*NA*	5	*NA*	*NA*	*NA*	*NA*	...	*NA*

- Netflix challenge: impute the unobserved ratings for personalized recommendation. `http://en.wikipedia.org/wiki/Netflix_Prize`

NETFLIX
Netflix Prize Progress
RMSE
1.0500
1.0000
0.9500
0.9000
0.8500
Nov 02
Nov 30
Dec 28
Jan 25
Feb 22
Mar 22
Apr 19
Date
NETFLIX
2009
DATE 09.21.09
PAY TO THE ORDER OF BellKor's Pragmatic Chaos
$1,000,000 00
AMOUNT ONE MILLION 00/100
FOR The Netflix Prize
Reed Hastings
Netflix Prize
Home
Rules
Leaderboard
Register
Update
Submit
Download

Leaderboard 10.05% Display top 20 leaders.

Rank	Team Name	Best Score	% Improvement	Last Submit Time
1	BellKor's Pragmatic Chaos	0.8558	10.05	2009-06-26 18:42:37
Grand Prize - RMSE <= 0.8563				
2	PragmaticTheory	0.8582	9.80	2009-06-25 22:15:51
3	BellKor in BigChaos	0.8590	9.71	2009-05-13 08:14:09
4	Grand Prize Team	0.8593	9.68	2009-06-12 08:20:24
5	Dace	0.8604	9.56	2009-04-22 05:57:03
6	BigChaos	0.8613	9.47	2009-06-23 23:06:52

- *Matrix completion problem.* Observe a very sparse matrix $\boldsymbol{Y} = (y_{ij})$. Want to impute all the missing entries. It is possible only when the matrix is structured, e.g., of *low rank*.

- Let $\Omega = \{(i,j) : \text{observed entries}\}$ index the observed entries and $P_\Omega(\boldsymbol{M})$ denote the projection of matrix $\boldsymbol{M}$ to Ω. The problem

$$\min_{\text{rank}(\boldsymbol{X}) \le r} \frac{1}{2} \|P_\Omega(\boldsymbol{Y}) - P_\Omega(\boldsymbol{X})\|_{\text{F}}^2 = \frac{1}{2} \sum_{(i,j)\in\Omega} (y_{ij} - x_{ij})^2$$

unfortunately is non-convex and difficult.

- *Convex relaxation* (Mazumder et al., 2010)

$$\min_{\boldsymbol{X}} f(\boldsymbol{X}) = \frac{1}{2} \|P_\Omega(\boldsymbol{Y}) - P_\Omega(\boldsymbol{X})\|_{\text{F}}^2 + \lambda \|\boldsymbol{X}\|_*,$$

where $\|\boldsymbol{X}\|_* = \|\sigma(\boldsymbol{X})\|_1 = \sum_i \sigma_i(\boldsymbol{X})$ is the nuclear norm.

- Majorization step:

$$
\begin{aligned}
f(\boldsymbol{X}) &= \frac{1}{2}\sum_{(i,j)\in\Omega}(y_{ij}-x_{ij})^2 + \frac{1}{2}\sum_{(i,j)\notin\Omega} 0 + \lambda\|\boldsymbol{X}\|_* \\
&\le \frac{1}{2}\sum_{(i,j)\in\Omega}(y_{ij}-x_{ij})^2 + \frac{1}{2}\sum_{(i,j)\notin\Omega}(x_{ij}^{(t)}-x_{ij})^2 + \lambda\|\boldsymbol{X}\|_* \\
&= \frac{1}{2}\|\boldsymbol{X}-\boldsymbol{Z}^{(t)}\|_{\mathrm{F}}^2 + \lambda\|\boldsymbol{X}\|_* \\
&= g(\boldsymbol{X}|\boldsymbol{X}^{(t)}),
\end{aligned}
$$

  where $\boldsymbol{Z}^{(t)} = P_\Omega(\boldsymbol{Y}) + P_{\Omega^\perp}(\boldsymbol{X}^{(t)})$. "fill in missing entries by current imputation"

- Minimization step:

  Rewrite the surrogate function

$$
g(\boldsymbol{X}|\boldsymbol{X}^{(t)}) = \frac{1}{2}\|\boldsymbol{X}\|_{\mathrm{F}}^2 - \mathrm{tr}(\boldsymbol{X}^T\boldsymbol{Z}^{(t)}) + \frac{1}{2}\|\boldsymbol{Z}^{(t)}\|_{\mathrm{F}}^2 + \lambda\|\boldsymbol{X}\|_*.
$$

  Let σ_i be the singular values of $\boldsymbol{X}$ and ω_i be the singular values of $\boldsymbol{Z}^{(t)}$. Observe

$$
\begin{aligned}
\|\boldsymbol{X}\|_{\mathrm{F}}^2 &= \|\sigma(\boldsymbol{X})\|_2^2 = \sum_i \sigma_i^2 \\
\|\boldsymbol{Z}^{(t)}\|_{\mathrm{F}}^2 &= \|\sigma(\boldsymbol{Z}^{(t)})\|_2^2 = \sum_i \omega_i^2
\end{aligned}
$$

  and by the Fan-von Neuman's inequality

$$
\mathrm{tr}(\boldsymbol{X}^{\mathsf{T}}\boldsymbol{Z}^{(t)}) \le \sum_i \sigma_i\omega_i
$$

  with equality achieved if and only if the left and right singular vectors of the two matrices coincide. Thus we can choose $\boldsymbol{X}$ to have same singular vectors as $\boldsymbol{Z}^{(t)}$ and

$$
\begin{aligned}
g(\boldsymbol{X}|\boldsymbol{X}^{(t)}) &= \frac{1}{2}\sum_i \sigma_i^2 - \sum_i \sigma_i\omega_i + \frac{1}{2}\omega_i^2 + \lambda\sum_i \sigma_i \\
&= \frac{1}{2}\sum_i(\sigma_i-\omega_i)^2 + \lambda\sum_i \sigma_i,
\end{aligned}
$$

  with minimizer given by $\sigma_i^{(t+1)} = (\omega_i - \lambda)_+$. "Singular value thresholding"

- Algorithm for matrix completion:

```
Initialize $\boldsymbol{X}^{(0)} \in \mathbb{R}^{m \times n}$
**repeat**
    $\boldsymbol{Z}^{(t+1)} \leftarrow P_{\Omega}(\boldsymbol{Y}) + P_{\Omega^{\perp}}(\boldsymbol{X}^{(t)})$
    Compute SVD: $\boldsymbol{U}\text{diag}(\boldsymbol{w})\boldsymbol{V}^{\top} \leftarrow \boldsymbol{Z}^{(t+1)}$
    $\boldsymbol{X}^{(t+1)} \leftarrow \boldsymbol{U}\text{diag}[(\boldsymbol{w} - \lambda)_{+}]\boldsymbol{V}^{\top}$
**until** objective value converges
```

- "Golub-Kahan-Reinsch" algorithm takes $4m^2n + 8mn^2 + 9n^3$ flops for a $m \geq n$ matrix and is not going to work for 480K-by-18K Netflix matrix.

  Notice only top singular values are needed since low rank solutions are achieved by large λ. Lanczos/Arnoldi algorithm is the way to go. Matrix-vector multiplication $\boldsymbol{Z}^{(t)}\boldsymbol{v}$ is fast (why?)

24 Lecture 24, Nov 20

Announcements

- HW7 (50pts) returned. Sketch of solution: `http://hua-zhou.github.io/teaching/st758-2014fall/hw07sol.html` Feedback:
  - Vectorize code by `outer` + `apply` or `table` + `apply`
  - Pre-compute certain quantities s_{jk} and r_k
  - Convergence criterion: relative change in objective values, or first oder optimality condition
- HW8 (ranking MLB teams) due next Tue Nov 25.
- HW9 (simulation project) due Dec 9 @ 11A.

Last time

- EM/MM example: Netflix problem

Today

- Quasi-Newton method.
- Conjugate gradient method.

Quasi-Newton methods (KL 14.9)

- Quasi-Newton is probably the most successful "black-box" optimizers in use. E.g., implemented in the general purpose optimization routine `optim()` in R.
- Consider the general Newton scheme for minimizing $f(\boldsymbol{x})$, $\boldsymbol{x} \in \mathcal{X} \subset \mathbb{R}^p$:

$$\boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{(t)} - s[\boldsymbol{A}^{(t)}]^{-1} \nabla f(\boldsymbol{x}^{(t)}),$$

  where $\boldsymbol{A}^{(t)}$ a pd approximation of the Hessian $d^2 f(\boldsymbol{x}^{(t)})$.
  - Pros: fast (quadratic) convergence

- Cons: compute and store Hessian at each iteration (usually $O(np^2)$ cost in statistical problems), solving a linear system ($O(p^3)$ cost in general), human efforts (derive and implement gradient and Hessian, pd approximation, ...)

- Any pd $\boldsymbol{A}$ gives a descent algorithm – tradeoff between convergence rate and cost per iteration.

- Setting $\boldsymbol{A} = \boldsymbol{I}$ leads to the *steepest descent* algorithm. Picture: slow convergence (zig-zagging) of steepest descent (with exact line search) for minimizing a convex quadratic function (ellipse).

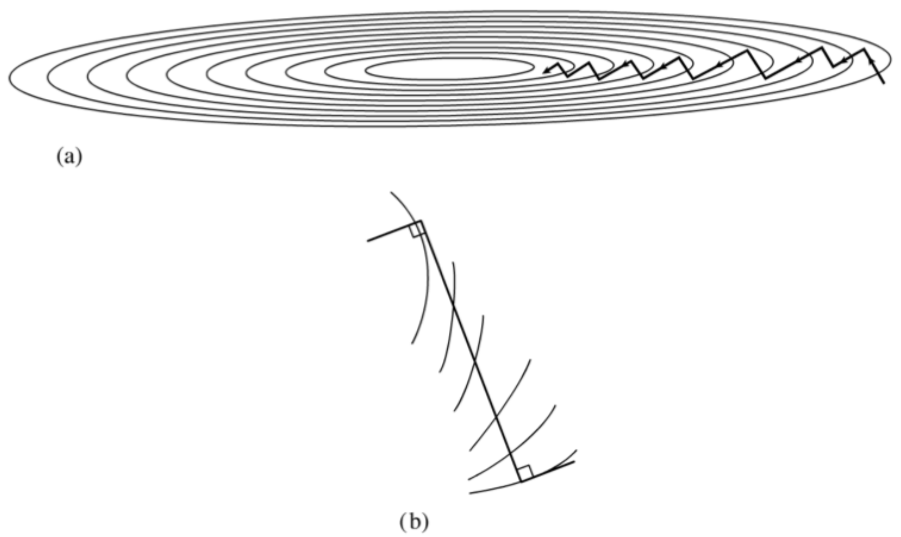


Figure 10.6.1. (a) Steepest descent method in a long, narrow "valley." While more efficient than the strategy of Figure 10.5.1, steepest descent is nonetheless an inefficient strategy, taking many steps to reach the valley floor. (b) Magnified view of one step: A step starts off in the local gradient 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.

How many steps does the Newton's method take for a convex quadratic f?

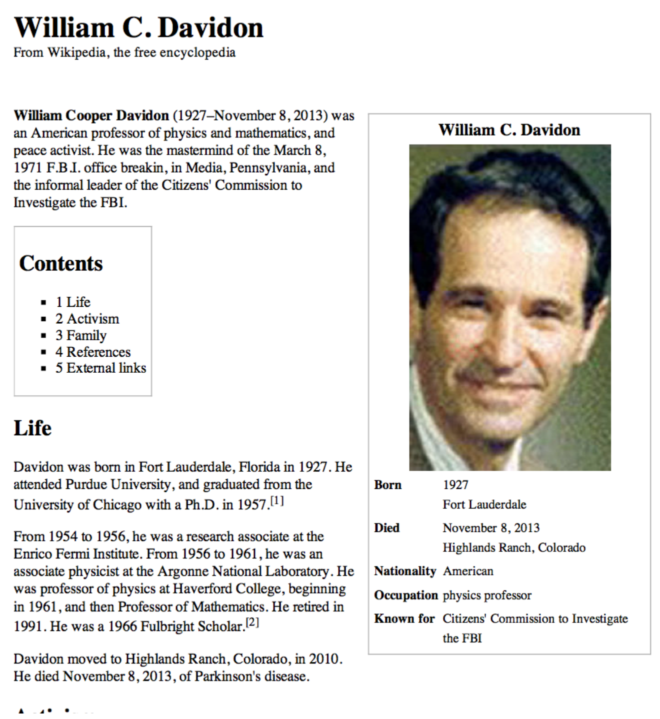
- Idea of Quasi-Newton: No analytical Hessian is required (still need gradient). Update approximate Hessian $\boldsymbol{A}$ according to the *secant condition*

$$\nabla f(\boldsymbol{x}^{(t-1)}) - \nabla f(\boldsymbol{x}^{(t)}) \approx d^2 f(\boldsymbol{x}^{(t)})(\boldsymbol{x}^{(t-1)} - \boldsymbol{x}^{(t)}).$$

Instead of computing $\boldsymbol{A}$ from scratch at each iteration, we update an approximation $\boldsymbol{A}$ to $d^2 f(\boldsymbol{x})$ which satisfies (1) p.d., (2) the secant condition, and (3) closest to the previous approximation.

- Super-linear convergence, compared to the quadratic convergence of Newton's method. But each iteration only takes $O(p^2)$!

- History: Davidon was a physicist at Argonne National Lab in 50s and proposed the very first idea of quasi-Newton method in 1959. Later studied, implemented, and popularized by Fletcher and Powell. Davidon's original paper was not accepted for publication ☹; 30 years later it appeared as the first article in the first issue of *SIAM Journal of Optimization* (Davidon, 1991).



William C. Davidon

From Wikipedia, the free encyclopedia

William Cooper Davidon (1927–November 8, 2013) was an American professor of physics and mathematics, and peace activist. He was the mastermind of the March 8, 1971 F.B.I. office breakin, in Media, Pennsylvania, and the informal leader of the Citizens' Commission to Investigate the FBI.

Contents

- 1 Life
- 2 Activism
- 3 Family
- 4 References
- 5 External links

Life

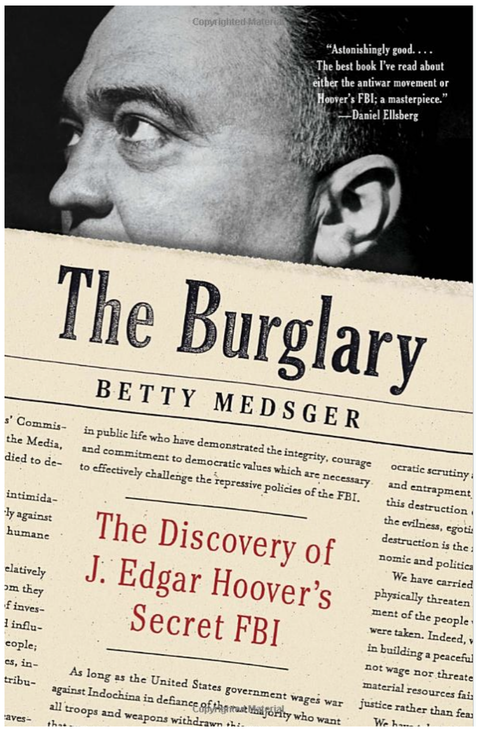
Davidon was born in Fort Lauderdale, Florida in 1927. He attended Purdue University, and graduated from the University of Chicago with a Ph.D. in 1957.[1]

From 1954 to 1956, he was a research associate at the Enrico Fermi Institute. From 1956 to 1961, he was an associate physicist at the Argonne National Laboratory. He was professor of physics at Haverford College, beginning in 1961, and then Professor of Mathematics. He retired in 1991. He was a 1966 Fulbright Scholar.[2]

Davidon moved to Highlands Ranch, Colorado, in 2010. He died November 8, 2013, of Parkinson's disease.

William C. Davidon

Born	1927 Fort Lauderdale
Died	November 8, 2013 Highlands Ranch, Colorado
Nationality	American
Occupation	physics professor
Known for	Citizens' Commission to Investigate the FBI


"Astonishingly good. . . . The best book I've read about either the antiwar movement or Hoover's FBI; a masterpiece."
—Daniel Ellsberg
The Burglary
BETTY MEDSGER
The Discovery of J. Edgar Hoover's Secret FBI

- Davidon-Fletcher-Powell (DFP) rank-2 update. Solve

$$
\begin{aligned}
\text{minimize} \quad & \|\boldsymbol{A} - \boldsymbol{A}^{(t)}\|_{\mathrm{F}} \\
\text{subject to} \quad & \boldsymbol{A} = \boldsymbol{A}^{\mathsf{T}} \\
& \boldsymbol{A}(\boldsymbol{x}^{(t)} - \boldsymbol{x}^{(t-1)}) = \nabla f(\boldsymbol{x}^{(t)}) - \nabla f(\boldsymbol{x}^{(t-1)})
\end{aligned}
$$

for the next approximation $\boldsymbol{A}^{(t+1)}$. The solution is a low rank (rank 1 or rank 2) update of $\boldsymbol{A}^{(t)}$. The inverse is too thanks to Sherman-Morrison-Woodbury! $O(p^2)$ operations. Need to store a p-by-p dense matrix. Positive definiteness is guaranteed by the same trick you used in HW6! See KL 14.9 for details.

- Broyden-Fletcher-Goldfarb-Shanno (BFGS) rank 2 update is considered by many the most effective among all quasi-Newton updates. BFGS imposes secant condition on the inverse of Hessian $\boldsymbol{H} = \boldsymbol{A}^{-1}$.

$$\begin{array}{ll} \text{minimize} & \|\boldsymbol{H} - \boldsymbol{H}^{(t)}\|_{\text{F}} \\ \text{subject to} & \boldsymbol{H} = \boldsymbol{H}^{\top} \\ & \boldsymbol{H}[\nabla f(\boldsymbol{x}^{(t)}) - \nabla f(\boldsymbol{x}^{(t-1)})] = \boldsymbol{x}^{(t)} - \boldsymbol{x}^{(t-1)}. \end{array}$$

  Again $\boldsymbol{H}^{(t+1)}$ is a rank two update of $\boldsymbol{H}^{(t)}$. $O(p^2)$ operations. Still need to store a dense p-by-p matrix.

- Limited-memory BFGS (L-BFGS). Only store the secant pairs. Particularly useful for large scale optimization.

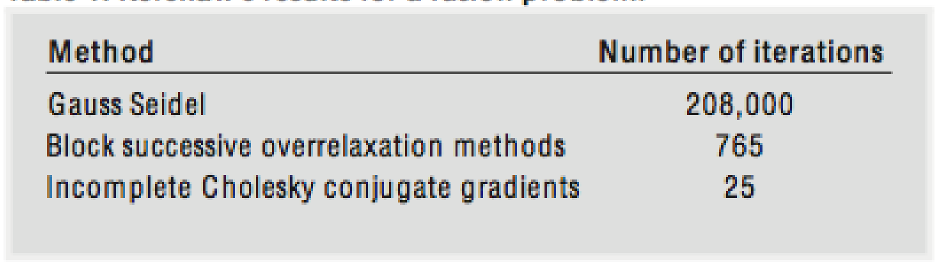
- L-BFGS-B: with box-constraints. Implemented in the general purpose optimization routine `optim()` in R.

- How to set the initial approximation $\boldsymbol{A}^{(0)}$? Identity or Hessian (if pd) or Fisher information matrix at starting point.

(Linear) Conjugate gradient method

- (Linear) Conjugate gradient is the top-notch iterative method for solving large, structured linear systems $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$. Earlier we talked about Jacobi and Gauss-Seidel as the more classical iterative solvers.

Table 1. Kershaw's results for a fusion problem.



Method	Number of iterations
Gauss Seidel	208,000
Block successive overrelaxation methods	765
Incomplete Cholesky conjugate gradients	25

- *Linear* conjugate gradient method: for solving large linear systems of equations.

  History: Hestenes and Stiefel in 50s.

- Solve linear equation $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, where $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ is pd, is equivalent to

$$\text{minimize } f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^\intercal \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}^\intercal \boldsymbol{x}.$$

  Note $\nabla f(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b} =: r(\boldsymbol{x})$.

- Consider a simple idea: coordinate descent, that is to update x_j alternatingly. Same as the Gauss-Seidel iteration.

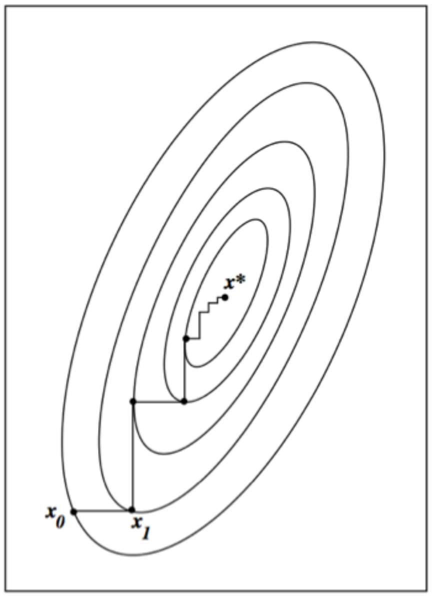


Figure 9.1
Coordinate search method makes slow progress on this function of two variables.

- A set of vectors $\{\boldsymbol{p}^{(0)}, \ldots, \boldsymbol{p}^{(l)}\}$ is said to be conjugate wrt $\boldsymbol{A}$ if

$$\boldsymbol{p}_i^\intercal \boldsymbol{A}\boldsymbol{p}_j = 0, \quad \text{for all } i \neq j.$$

- *Conjugate direction* method: Given a set of conjugate vectors $\{\boldsymbol{p}^{(0)}, \ldots, \boldsymbol{p}^{(l)}\}$, at iteration t, we search along the conjugate direction $\boldsymbol{p}^{(t)}$

$$\boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{(t)} + \alpha^{(t)}\boldsymbol{p}^{(t)},$$

  where

$$\alpha^{(t)} = -\frac{\boldsymbol{r}^{(t)\,\intercal}\boldsymbol{p}^{(t)}}{\boldsymbol{p}^{(t)\,\intercal}\boldsymbol{A}\boldsymbol{p}^{(t)}}.$$

- Theorem: $\boldsymbol{x}^{(t)}$ converges to the solution in at most n steps.

  Intuition: Look at graph.

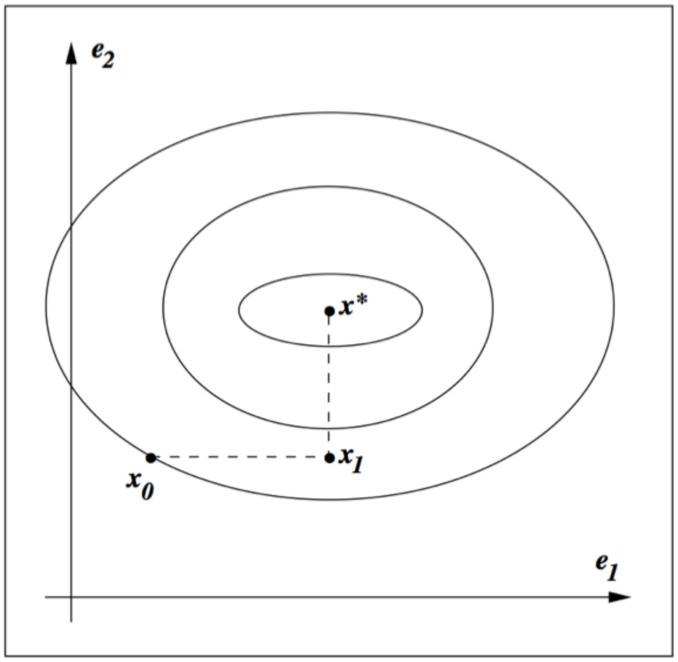


Figure 5.1 Successive minimizations along the coordinate directions find the minimizer of a quadratic with a diagonal Hessian in *n* iterations.

- *Conjugate gradient* method. Idea: generate $\boldsymbol{p}^{(t)}$ using only $\boldsymbol{p}^{(t-1)}$

$$\boldsymbol{p}^{(t)} = -\boldsymbol{r}^{(t)} + \beta^{(t)}\boldsymbol{p}^{(t-1)},$$

  where $\beta^{(t)}$ is determined by the conjugacy condition $\boldsymbol{p}^{(t-1)\,\mathsf{T}}\boldsymbol{A}\boldsymbol{p}^{(t)} = 0$

$$\beta^{(t)} = \frac{\boldsymbol{r}^{(t)\,\mathsf{T}}\boldsymbol{A}\boldsymbol{p}^{(t-1)}}{\boldsymbol{p}^{(t-1)\,\mathsf{T}}\boldsymbol{A}\boldsymbol{p}^{(t-1)}}.$$

- CG algorithm (preliminary version):

```
Given $\boldsymbol{x}^{(0)}$
Initialize: $\boldsymbol{r}^{(0)} \leftarrow \boldsymbol{A}\boldsymbol{x}^{(0)} - \boldsymbol{b}$, $\boldsymbol{p}^{(0)} \leftarrow -\boldsymbol{r}^{(0)}$, $t = 0$
**while** $\boldsymbol{r}^{(0)} \neq \boldsymbol{0}$ **do**
  $\alpha^{(t)} \leftarrow -\frac{\boldsymbol{r}^{(t)\,\mathsf{T}}\boldsymbol{p}^{(t)}}{\boldsymbol{p}^{(t)\,\mathsf{T}}\boldsymbol{A}\boldsymbol{p}^{(t)}}$
  $\boldsymbol{x}^{(t+1)} \leftarrow \boldsymbol{x}^{(t)} + \alpha^{(t)}\boldsymbol{p}^{(t)}$
  $\boldsymbol{r}^{(t+1)} \leftarrow \boldsymbol{A}\boldsymbol{x}^{(t+1)} - \boldsymbol{b}$
  $\beta^{(t+1)} \leftarrow \frac{\boldsymbol{r}^{(t+1)\,\mathsf{T}}\boldsymbol{A}\boldsymbol{p}^{(t)}}{\boldsymbol{p}^{(t)\,\mathsf{T}}\boldsymbol{A}\boldsymbol{p}^{(t)}}$
  $\boldsymbol{p}^{(t+1)} \leftarrow -\boldsymbol{r}^{(t+1)} + \beta^{(t+1)}\boldsymbol{p}^{(t)}$
```

```
    $t \leftarrow t+1$
end while
```

- Theorem: With CG algorithm

  1. $\boldsymbol{r}^{(t)}$ are mutually orthogonal.
  2. $\{\boldsymbol{r}^{(0)}, \ldots, \boldsymbol{r}^{(t)}\}$ is contained in the *Krylov* subspace of degree t for $\boldsymbol{r}^{(0)}$, denoted by

  $$\mathcal{K}(\boldsymbol{r}^{(0)}; t) = \text{span}\{\boldsymbol{r}^{(0)}, \boldsymbol{A}\boldsymbol{r}^{(0)}, \boldsymbol{A}^2\boldsymbol{r}^{(0)}, \ldots, \boldsymbol{A}^t\boldsymbol{r}^{(0)}\}.$$

  3. $\{\boldsymbol{p}^{(0)}, \ldots, \boldsymbol{p}^{(t)}\}$ is contained in $\mathcal{K}(\boldsymbol{r}^{(0)}; t)$.
  4. $\boldsymbol{p}^{(0)}, \ldots, \boldsymbol{p}^{(t)}$ are conjugate wrt $\boldsymbol{A}$.

  The iterates $\boldsymbol{x}^{(t)}$ converge to the solution in at most n steps.

- CG algorithm (economical version): saves one matrix-vector multiplication.

```
Given $\boldsymbol{x}^{(0)}$
Initialize: $\boldsymbol{r}^{(0)} \leftarrow \boldsymbol{A}\boldsymbol{x}^{(0)} - \boldsymbol{b}$, $\boldsymbol{p}^{(0)} \leftarrow -\boldsymbol{r}^{(0)}$, $t = 0$
while $\boldsymbol{r}^{(0)} \neq \boldsymbol{0}$ do
    $\alpha^{(t)} \leftarrow \frac{\boldsymbol{r}^{(t)\top}\boldsymbol{r}^{(t)}}{\boldsymbol{p}^{(t)\top}\boldsymbol{A}\boldsymbol{p}^{(t)}}$
    $\boldsymbol{x}^{(t+1)} \leftarrow \boldsymbol{x}^{(t)} + \alpha^{(t)}\boldsymbol{p}^{(t)}$
    $\boldsymbol{r}^{(t+1)} \leftarrow \boldsymbol{r}^{(t)} + \alpha^{(t)}\boldsymbol{A}\boldsymbol{p}^{(t)}$
    $\beta^{(t+1)} \leftarrow \frac{\boldsymbol{r}^{(t+1)}\boldsymbol{r}^{(t+1)}}{\boldsymbol{r}^{(t)}\boldsymbol{r}^{(t)}}$
    $\boldsymbol{p}^{(t+1)} \leftarrow -\boldsymbol{r}^{(t+1)} + \beta^{(t+1)}\boldsymbol{p}^{(t)}$
    $t \leftarrow t+1$
end while
```

- Computation cost per iteration is one matrix vector multiplication: $\boldsymbol{A}\boldsymbol{p}^{(t)}$.

  Consider PageRank problem, $\boldsymbol{A}$ has dimension $n \approx 10^{10}$ but is highly structured (sparse + low rank). Each matrix vector multiplication takes $O(n)$.

- Theorem: If $\boldsymbol{A}$ has r distinct eigenvalues, $\boldsymbol{x}^{(t)}$ converges to solution $\boldsymbol{x}^*$ in at most r steps.

25 Lecture 25, Nov 25

Announcements

- HW8 (ranking MLB teams) due today. Submit hardcopy and email code (`LastFirstHW8.R` or `LastFirstHW8.Rmd`).
- HW9 (simulation project) due Dec 9 @ 11A.
- No office hours this Thu and Fri (Thanksgiving).

Last time

- Quasi-Newton method.
  - DFP: keep rank-2 updating inverse of approximate Hessian
  - BFGS: keep rank-2 updating approximate Hessian inverse
- (Linear) conjugate gradient method for solving linear equation $\boldsymbol{Ax} = \boldsymbol{b}$.

Today

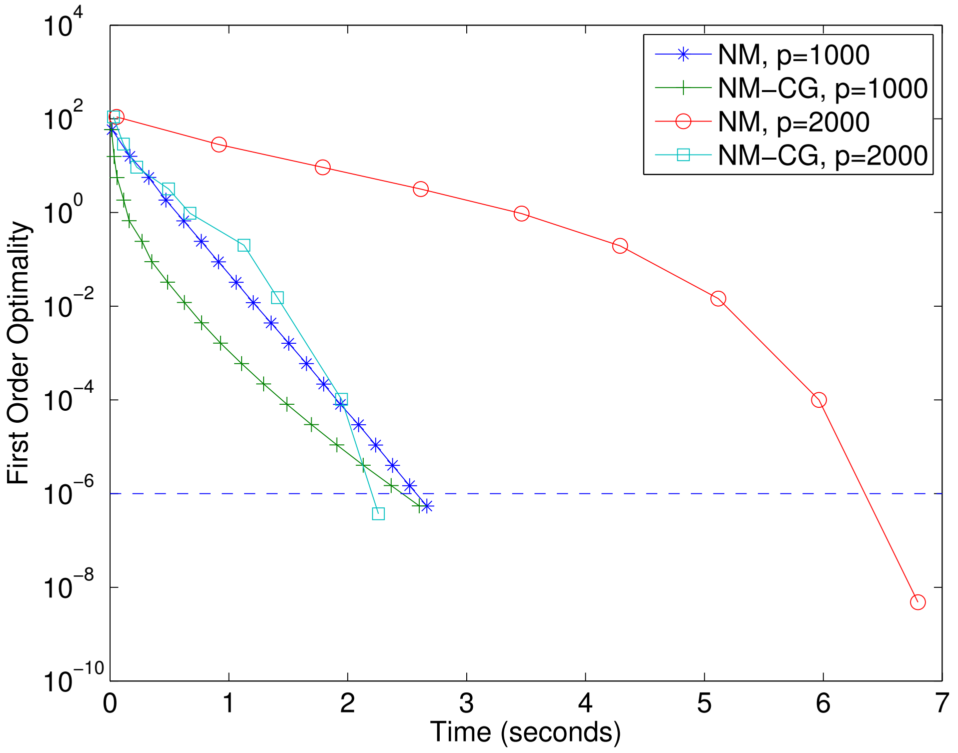
- Preconditioned conjugate gradient.
- (Nonlinear) conjugate gradient method.
- Concluding remarks on optimization.

Pre-conditioned conjugate gradient (PCG)

- Summary of conjugate gradient method for solving $\boldsymbol{Ax} = \boldsymbol{b}$ or equivalently minimizing $\frac{1}{2}\boldsymbol{x}^T\boldsymbol{Ax} - \boldsymbol{b}^T\boldsymbol{x}$:
  - Each iteration needs one matrix vector multiplication: $\boldsymbol{Ap}^{(t+1)}$. For structured $\boldsymbol{A}$, often $O(n)$ cost per iteration.
  - Guaranteed to converge in n steps.

- One example of using CG. Consider the Newton method implemented in HW8 for ranking teams. Suppose the number of teams/players p is huge (e.g. online Chess). Most likely the Hessian is sparse. Then CG can be used to compute the Newton direction.

  Simulation setup in the following figure: $p = 1000$ or 2000, strengths of team are $\lambda_i = i + (p/10)$, each player competes with about $p/20$ opponents.



- Two important bounds for conjugate gradient algorithm:

  Let $\lambda_1 \le \cdots \le \lambda_n$ be the ordered eigenvalues of a pd $\boldsymbol{A}$.

$$
\begin{aligned}
\|\boldsymbol{x}^{(t+1)} - \boldsymbol{x}^*\|_{\boldsymbol{A}}^2 &\le \left(\frac{\lambda_{n-t} - \lambda_1}{\lambda_{n-t} + \lambda_1}\right)^2 \|\boldsymbol{x}^{(0)} - \boldsymbol{x}^*\|_{\boldsymbol{A}}^2 \\
\|\boldsymbol{x}^{(t+1)} - \boldsymbol{x}^*\|_{\boldsymbol{A}}^2 &\le 2\left(\frac{\sqrt{\kappa(\boldsymbol{A})} - 1}{\sqrt{\kappa(\boldsymbol{A})} + 1}\right)^t \|\boldsymbol{x}^{(0)} - \boldsymbol{x}^*\|_{\boldsymbol{A}}^2,
\end{aligned}
$$

  where $\kappa(\boldsymbol{A}) = \lambda_n/\lambda_1$ is the condition number of $\boldsymbol{A}$.

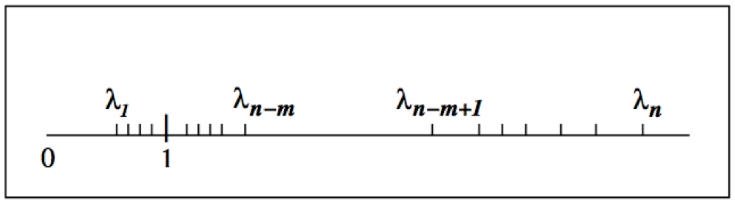


Figure 5.3 Two clusters of eigenvalues.

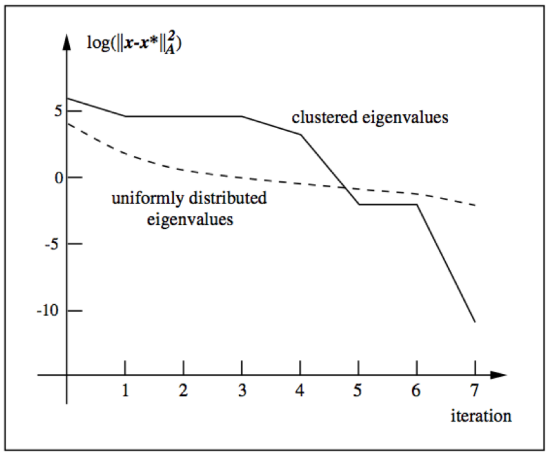


Figure 5.4 Performance of the conjugate gradient method on (a) a problem in which five of the eigenvalues are large and the remainder are clustered near 1, and (b) a matrix with uniformly distributed eigenvalues.

- Messages:
  - Roughly speaking, if the eigenvalues of $\boldsymbol{A}$ occur in r distinct clusters, the CG iterates will *approximately* solve the problem after $O(r)$ steps.
  - $\boldsymbol{A}$ with a small condition number $(\lambda_1 \approx \lambda_n)$ converges fast.
- *Pre-conditioning*: Change of variables $\hat{\boldsymbol{x}} = \boldsymbol{C}\boldsymbol{x}$ via a nonsingular $\boldsymbol{C}$ and solve

$$(\boldsymbol{C}^{-T}\boldsymbol{A}\boldsymbol{C}^{-1})\hat{\boldsymbol{x}} = \boldsymbol{C}^{-T}\boldsymbol{b}.$$

  Choose $\boldsymbol{C}$ such that
  - $\boldsymbol{C}^{-T}\boldsymbol{A}\boldsymbol{C}^{-1}$ has small condition number, or
  - $\boldsymbol{C}^{-T}\boldsymbol{A}\boldsymbol{C}^{-1}$ has clustered eigenvalues
  - Inexpensive solution of $\boldsymbol{C}^T\boldsymbol{C}\boldsymbol{y} = \boldsymbol{r}$

- Preconditioned CG does not make use of $\boldsymbol{C}$ explicitly, but rather the matrix $\boldsymbol{M} = \boldsymbol{C}^T\boldsymbol{C}$.

- Preconditioned CG (PCG) algorithm:

```
Given $\boldsymbol{x}^{(0)}$, pre-conditioner $\boldsymbol{M}$
$\boldsymbol{r}^{(0)} \leftarrow \boldsymbol{A}\boldsymbol{x}^{(0)} - \boldsymbol{b}$
solve $\boldsymbol{M}\boldsymbol{y}^{(0)} = \boldsymbol{r}^{(0)}$ for $\boldsymbol{y}^{(0)}$
$\boldsymbol{p}^{(0)} \leftarrow -\boldsymbol{r}^{(0)}$, $t = 0$
while $\boldsymbol{r}^{(0)} \neq \boldsymbol{0}$ do
  $\alpha^{(t)} \leftarrow \frac{\boldsymbol{r}^{(t)\top}\boldsymbol{y}^{(t)}}{\boldsymbol{p}^{(t)\top}\boldsymbol{A}\boldsymbol{p}^{(t)}}$
  $\boldsymbol{x}^{(t+1)} \leftarrow \boldsymbol{x}^{(t)} + \alpha^{(t)}\boldsymbol{p}^{(t)}$
  $\boldsymbol{r}^{(t+1)} \leftarrow \boldsymbol{r}^{(t)} + \alpha^{(t)}\boldsymbol{A}\boldsymbol{p}^{(t)}$
  Solve $\boldsymbol{M}\boldsymbol{y}^{(t+1)} \leftarrow \boldsymbol{r}^{(t+1)}$ for $\boldsymbol{y}^{(t+1)}$
  $\beta^{(t+1)} \leftarrow \frac{\boldsymbol{r}^{(t+1)\top}\boldsymbol{y}^{(t+1)}}{\boldsymbol{r}^{(t)\top}\boldsymbol{r}^{(t)}}$
  $\boldsymbol{p}^{(t+1)} \leftarrow -\boldsymbol{y}^{(t+1)} + \beta^{(t+1)}\boldsymbol{p}^{(t)}$
  $t \leftarrow t + 1$
end while
```

  Remark: Only extra cost in the pre-conditioned CG algorithm is the need to solve the linear system $\boldsymbol{M}\boldsymbol{y} = \boldsymbol{r}$.

- Pre-conditioning is more like an art than science. Some choices include

  - Incomplete Cholesky. $\boldsymbol{A} \approx \tilde{\boldsymbol{L}}\tilde{\boldsymbol{L}}^T$, where $\tilde{\boldsymbol{L}}$ is a sparse approximate Cholesky factor. Then $\tilde{\boldsymbol{L}}^{-1}\boldsymbol{A}\tilde{\boldsymbol{L}}^{-T} \approx \boldsymbol{I}$ (perfectly conditioned) and $\boldsymbol{M}\boldsymbol{y} = \tilde{\boldsymbol{L}}\tilde{\boldsymbol{L}}^T\boldsymbol{y} = \boldsymbol{r}$ is easy to solve.
  - Banded pre-conditioners.
  - Choose $\boldsymbol{M}$ as a coarsened version of $\boldsymbol{A}$.
  - *Subject knowledge.* Knowledge about the structure and origin of a problem is often the key to devising efficient pre-conditioner. For example, see recent work of Stein et al. (2012) for pre-conditioning large covariance matrices. `http://epubs.siam.org/doi/abs/10.1137/110834469`

More buzzwords and softwares

Here are a few variants of CG that we should at least know the names and what they are for (so we can Google later in need).

- MINRES (minimum residual method): symmetric indefinite $\boldsymbol{A}$.
- Bi-CG (bi-conjugate gradient): unsymmetric $\boldsymbol{A}$.
- Bi-CGSTAB (Bi-CG stabilized): improved version of Bi-CG.
- GMRES (generalized minimum residual method): current *de facto* method for unsymmetric $\boldsymbol{A}$. E.g., PageRank problem.
- Lanczos method: top eigen-pairs of a large symmetric matrix.
- Arnoldi method: top eigen-pairs of a large unsymmetric matrix.
- Lanczos bidiagonalization algorithm: top singular triples of large matrix.

  Remark: For Lanczos/Arnoldi methods, the critical computation is still matrix vector multiplication $\boldsymbol{A}\boldsymbol{v}$.

Softwares:

- Matlab:
  - Iterative methods for solving linear equations:
    `pcg`, `bicg`, `bicgstab`, `gmres`, ...
  - Iterative methods for top eigen-pairs and singular pairs:
    `eigs`, `svds`, ...
  - Pre-conditioner:
    `cholinc`, `luinc`, ...

  Get familiar with the reverse communication interface (RCI) for utilizing iterative solvers:

  ```
  x = gmres(A, b)
  x = gmres(@Afun, b)
  eigs(A)
  eigs(@Afun)
  ```

- Consider the PageRank problem. We want to find the top left eigenvector of the transition matrix

$$\boldsymbol{P} = p\boldsymbol{R}^{+}\boldsymbol{A} + \boldsymbol{z}\boldsymbol{1}_n^{\mathsf{T}},$$

where $\boldsymbol{R} = \text{diag}(r_1, \ldots, r_n)$ and $z_j = (1-p)/n$ if $r_i > 0$ and $1/n$ if $r_i = 0$. Size of $\boldsymbol{P}$ can be huge: $n \approx 40$ billion web pages. How to call the `gmres` or `eigs` function?

- R: Try Google and good luck ...

(Nonlinear) Conjugate gradient method

- *Linear* conjugate gradient method is for solving linear system $\boldsymbol{Ax} = \boldsymbol{b}$, or equivalently, minimizing $\frac{1}{2}\boldsymbol{x}^T\boldsymbol{Ax} - \boldsymbol{b}^T\boldsymbol{x}$.

- *Nonlinear* conjugate gradient is for nonlinear optimization

$$\text{minimize}\, f(\boldsymbol{x}).$$

- History: Fletcher and Reeves in 60s.

- Fletcher-Reeves CG for nonlinear minimization:

```
Given $\boldsymbol{x}^{(0)}$
Evaluate $\nabla f^{(0)} = \nabla f(\boldsymbol{x}^{(0)})$
Set $\boldsymbol{p}^{(0)} \leftarrow -\nabla f^{(0)}$, $t \leftarrow 0$
**while** $\nabla f^{(t)} \neq \boldsymbol{0}$ **do**
    Compute $\alpha^{(t)}$ and set $\boldsymbol{x}^{(t+1)} \leftarrow \boldsymbol{x}^{(t)} + \alpha^{(t)}\boldsymbol{p}^{(t)}$
    Evaluate $\nabla f^{(t+1)} = \nabla f(\boldsymbol{x}^{(t+1)})$
    $\beta^{(t+1)} \leftarrow \frac{df^{(t+1)}\nabla f^{(t+1)}}{df^{(t)}\nabla f^{(t)}}$
    $\boldsymbol{p}^{(t+1)} \leftarrow -\nabla f^{(t+1)} + \beta^{(t+1)}\boldsymbol{p}^{(t)}$
    $t \leftarrow t + 1$
**end while**
```

- Most cost is evaluation of objective function and its gradient. No matrix operations are needed. Appealing for large nonlinear optimization problems.

- Line search (choose $\alpha^{(t)}$) is necessary to get a descending algorithm.

Concluding remarks on optimization

- Nonlinear optimization algorithms

Algorithm	Convergence Rate	Per-iteration Cost	Example
Newton	quadratic	high, usually $O(np^2) + O(p^3)$	GLM with canonical link
Fisher Scoring	super-linear	high, usually $O(np^2) + O(p^3)$	GLM with non-canonical link
Gauss-Newton	super-linear	high, usually $O(np^2) + O(p^3)$	nonlinear GLM
Quasi-Newton	super-linear	moderate, usually $O(np) + O(p^2)$	
Conjugate gradient	super-linear	moderate, usually $O(np) + O(p^2)$	
Coordinate descent	linear	low	
Steepest descent	linear	low	
EM/MM	linear	low	

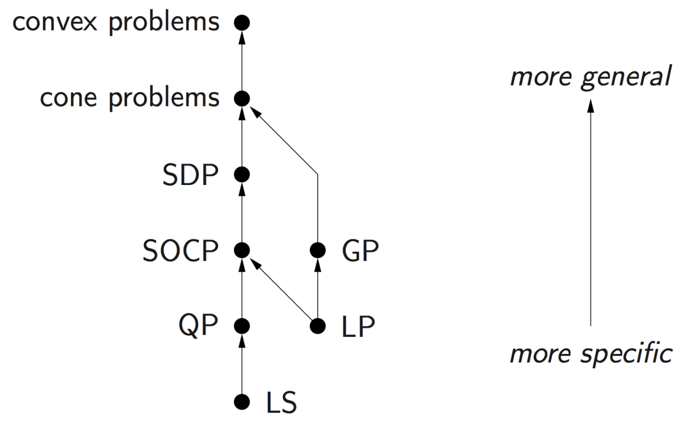
There is simply no such thing as a universal 'gold standard' when it comes to algorithms.

Unknown Reviewer

- *Problem specific analysis* is critical for developing a successful optimization algorithm.

  E.g., I don't think any black-box procedure can beat our safe-guarded Newton's method for the Dirichlet-Multinomial MLE problem (HW6/7) in terms of efficiency.

- Use "black-box" for
  - numerical linear algebra
  - least squares
  - convex programming (TODO in ST790-003 *Advanced Statistical Computing*). Current "technology" (`Cplex`, `Gurobi`, `Mosek`, `cvx`, `Matlab`, ..., `R` is not here ☹) can deal with problem with up to $10^3 \sim 10^4$ variables and constraints or even more with structure.



For example, the optimization problem in HW8 (ranking MLB teams)

$$\text{minimize} \quad \prod_{i,j} \left(\frac{\gamma_i}{\gamma_i + \gamma_j} \right)^{-y_{ij}} = \prod_{i,j} \gamma_i^{-y_{ij}} (\gamma_i + \gamma_j)^{y_{ij}}$$

is recognized as a standard geometric programming (GP) problem. Using the open source convex optimization software CVX (Grant and Boyd, 2012) amounts to only 6 lines of Matlab code

```
p = size(Y, 1);
[rowidx, colidx, yvec] = find(Y);
cvx_begin gp
    variable gamma(p)
    minimize prod(((gamma(rowidx) + gamma(colidx)) ./ gamma(rowidx)) .^ yvec)
cvx_end
```

- First order methods (EM/MM, CD, steepest descent) is easier for parallel computing.

Algorithm development goes hand in glove with hardware advancement.

Hua

- Reference books:

- *Numerical Optimization* (Nocedal and Wright, 2006)
- *Convex Optimization* (Boyd and Vandenberghe, 2004)
- *The EM algorithm and Extensions* (McLachlan and Krishnan, 2008)
- *Numerical Analysis for Statisticians* (Lange, 2010)

26 Lecture 26, Dec 2

Announcements

- HW8 (ranking MLB teams) returned. Feedback:
  - Bradley-Terry model. KL 12.6 or David Hunter's paper.
  - Non-concavity of the log-likelihood in $\boldsymbol{\lambda}$ parameterization. Enough to find a simple counter-example. However the negative log-likelihood is an example of geometric program, a branch of convex programming.
  - Vectorization of MM update: `outer` function.
  - Concavity of the log-likelihood function in $\boldsymbol{\gamma}$ parameterization. "log-sum-exp" terms are convex.
  - Implementation of Newton's method. Hessian is singular due to identifiability. Setting γ_1.
  - Problem structure: sparsity in large league.
  - Check David Hunter's code for taking advantage of sparsity.

  Sketch of solution: `http://hua-zhou.github.io/teaching/st758-2014fall/hw08sol.html`
- HW9 (simulation project) due Dec 9 @ 11A.
- FAQs on HW9 (simulation project)
- Regular office hours this week: Tue (Hua), Thu (Hua) and Fri (William). No office hours next week.
- **Course evaluation**!: `https://classeval.ncsu.edu/`

Last time

- Pre-conditioning for conjugate gradient (PCG) method.
- Nonlinear conjugate gradient for optimization.
- Concluding remarks on optimization.

Today

- Introduction to Markov chains and MCMC.
- Fast algorithms: sorting, FFT.
- Take home messages.

Introduction to MCMC

Some topics I'll briefly talk about.

- History of Markov chain
- History of Monte Carlo and birth of MCMC
- Convergence rate of Markov chain

Markov chains

- Markov chain is a stochastic process $X_0, X_1, X_2, \ldots$ with the *Markov property*

$$\mathbf{P}(X_{t+1}|X_t, X_{t-1}, \ldots, X_0) = \mathbf{P}(X_{t+1}|X_t).$$

  Given current state, the future is independent of the past.

- Stochastic analog of ordinary differential equations.

$$\frac{dx(t)}{dt} = F(x(t))$$
$$\mathcal{L}(X_{t+1}|X_t = x) = K(x, \cdot)$$

- Notations (for discrete time, finite Markov chains)
  - a finite state space $\mathcal{X}$
  - transition matrix $K(x, y)$, $x, y \in \mathcal{X}$
  - stationary distribution π on $\mathcal{X}$, defined as a probability vector that satisfies

$$\pi^T K = \pi^T.$$

    Existence of such π is guaranteed by the Perron-Frobenius theorem.

  - $K^l(x,y)$ denotes the l-step transition probabilities

- Markov chains in early years:
  - Fermat and Pascal (circa 1654): Gambler's ruin.
  - Bernoulli (1769) and Laplace (1812): Urn model.
  - I. J. Bienaymé (1845), and later Sir Francis Galton and Watson (*Educational Times*, 1873): Branching process.
  - Paul and Tatiana Ehrenfest (1906): Statistical physics.
  - Poincaré (1912): Card shuffling. *Calcul des Probabilités*

  - Markov's contribution
    - Markov, A. A. (1906)
      Extension of the law of large numbers to dependent quantities [in Russian], *Izv. Fiz.-Matem. Obsch. Kazan Univ. (2nd Ser.)*
    - St. Petersburg School vs Moscow School.
    - Example: 20,000 letters in Pushkin's *Eugene Onegin*.

$$\begin{array}{l} \\ \text{vowel} \\ \text{consonant} \end{array} \begin{array}{c} \begin{array}{cc} \text{vowel} & \text{consonant} \end{array} \\ \begin{pmatrix} 0.128 & 0.872 \\ 0.663 & 0.337 \end{pmatrix} \end{array} \quad \begin{array}{c} \pi \\ \begin{pmatrix} 0.432 \\ 0.568 \end{pmatrix} \end{array}$$

    - See Seneta's interesting account on the history (Seneta, 1996)
  - Some other leading pioneers: Kolmogorov, Fréchet, and Doeblin.

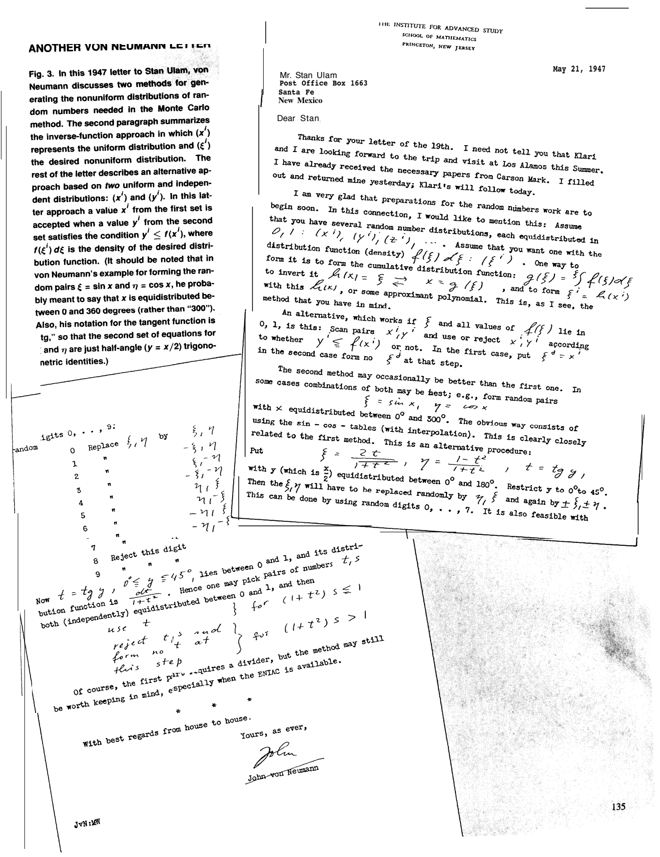
- Why are Markov chains important in statistics?

  - Modeling tool.
    E.g., Marc Coram's "jail message" example: Markov model for letter sequence; PageRank's (imaginary) random web surfer; ...
  - A methodology that revolutionized statistics: Markov Chain Monte Carlo (MCMC).
    * Metropolis, Rosenbluth, Rosenbluth, Teller and Teller (1953)
      Equation of state calculation by fast computing machines, *The Journal of Chemical Physics*, 21, 1087–1092.
    * Want to sample from a distribution $\pi(x) \propto f(x)$.
      Metropolis algorithm constructs a Markov chain that converges to π.
    * Marc Coram's "jail message" example.

Markov chain Monte Carlo (MCMC)

- *Monte Carlo method* is a generic name for "computational algorithms that rely on repeated random sampling to obtain numerical results". They are in contrast to the deterministic algorithms. They have wide applications in
  - integration
  - drawing sample from a distribution. E.g., HW9 (simulation study).
  - optimization (simulated annealing). E.g., Marc Coram's "jail message" example, traveling salesman, Soduku, ...
- Monte Carlo in early years

- Stanislaw Ulam conceived it in 1946 while playing solitaire in hospital bed. He wanted to know the probability of getting a perfect solitaire hand, and wondered whether computers can help answer this.
- John von Neumann was intrigued by the idea and developed a way to generate pseudorandom numbers (inversion, importance sampling, acceptance-rejection sampling) on electronic digital computers (ENIAC) to realize Ulam's idea, for the neutron fission and diffusion problem.



ANOTHER VON NEUMANN LETTER

Fig. 3. In this 1947 letter to Stan Ulam, von Neumann discusses two methods for generating the nonuniform distributions of random numbers needed in the Monte Carlo method. The second paragraph summarizes the inverse-function approach in which (x^i) represents the uniform distribution and (ξ^i) the desired nonuniform distribution. The rest of the letter describes an alternative approach based on *two* uniform and independent distributions: (x^i) and (y^i). In this latter approach a value x^i from the first set is accepted when a value y^i from the second set satisfies the condition $y^i \leq f(x^i)$, where $f(\xi^i)\,d\xi$ is the density of the desired distribution function. (It should be noted that in von Neumann's example for forming the random pairs $\xi = \sin x$ and $\eta = \cos x$, he probably meant to say that x is equidistributed between 0 and 360 degrees (rather than "300"). Also, his notation for the tangent function is tg," so that the second set of equations for and η are just half-angle ($y = x/2$) trigonometric identities.)

THE INSTITUTE FOR ADVANCED STUDY
SCHOOL OF MATHEMATICS
PRINCETON, NEW JERSEY

May 21, 1947

Mr. Stan Ulam
Post Office Box 1663
Santa Fe
New Mexico

Dear Stan

Thanks for your letter of the 19th. I need not tell you that Klari and I are looking forward to the trip and visit at Los Alamos this Summer. I have already received the necessary papers from Carson Mark. I filled out and returned mine yesterday; Klari's will follow today.

I am very glad that preparations for the random numbers work are to begin soon. In this connection, I would like to mention this: Assume that you have several random number distributions, each equidistributed in $0, 1$: $(x^i), (y^i), (z^i), \ldots$. Assume that you want one with the distribution function (density) $f(\xi)\,d\xi$: (ξ^i) . One way to form it is to form the cumulative distribution function: $g(\xi) = \int^{\xi} f(\xi)\,d\xi$ to invert it $h(x) = \xi \rightleftarrows x = g(\xi)$, and to form $\xi^i = h(x^i)$ with this $h(x)$, or some approximant polynomial. This is, as I see, the method that you have in mind.

An alternative, which works if ξ and all values of $f(\xi)$ lie in 0, 1, is this: Scan pairs x^i, y^i and use or reject x^i, y^i according to whether $y^i \leqq f(x^i)$ or not. In the first case, put $\xi^j = x^i$ in the second case form no ξ^j at that step.

The second method may occasionally be better than the first one. In some cases combinations of both may be best; e.g., form random pairs $\xi = \sin x, \ \eta = \cos x$ with x equidistributed between 0° and 300°. The obvious way consists of using the sin - cos - tables (with interpolation). This is clearly closely related to the first method. This is an alternative procedure:

Put $\xi = \frac{2t}{1+t^2}, \ \eta = \frac{1-t^2}{1+t^2}, \ t = tg\, y,$

with y (which is $\frac{x}{2}$) equidistributed between 0° and 180°. Restrict y to 0° to 45°. Then the ξ, η will have to be replaced randomly by η, ξ and again by $\pm\xi, \pm\eta$. This can be done by using random digits 0, . . . , 7. It is also feasible with random digits 0, . . . , 9:

0	Replace ξ, η by	ξ, η
		$-\xi, \eta$
1	"	$\xi, -\eta$
2	"	$-\xi, -\eta$
3	"	η, ξ
4	"	$\eta, -\xi$
5	"	$-\eta, \xi$
6	"	$-\eta, -\xi$
7	Reject this digit	
8	" " "	
9	" " "	

Now $t = tg\, y$, $0° \leqq y \leqq 45°$, lies between 0 and 1, and its distribution function is $\frac{dt}{1+t^2}$. Hence one may pick pairs of numbers t, s both (independently) equidistributed between 0 and 1, and then

$$\left.\text{use } t\right\} \text{ for } (1+t^2)\,s \leqq 1$$
$$\left.\begin{matrix}\text{reject } t, s \text{ and} \\ \text{form no } t \text{ at} \\ \text{this step}\end{matrix}\right\} \text{ for } (1+t^2)\,s > 1$$

Of course, the first part requires a divider, but the method may still be worth keeping in mind, especially when the ENIAC is available.

* * *

With best regards from house to house.

Yours, as ever,

John

John von Neumann

JvN:MW

135

– Ulam and von Neumann, working on Manhattan Project, used the code name *Monte Carlo*. It is a casino in Monaco where Ulam's uncle frequented.

– Nicholas Metropolis, fascinated by the Monte Carlo idea too, designed and built computing devices (MANIAC) to handle such calculations. His paper with Ulam in JASA (Metropolis and Ulam, 1949) formed the basis of modern sequential Monte Carlo methods.

– Birth of MCMC (Metropolis et al., 1953):

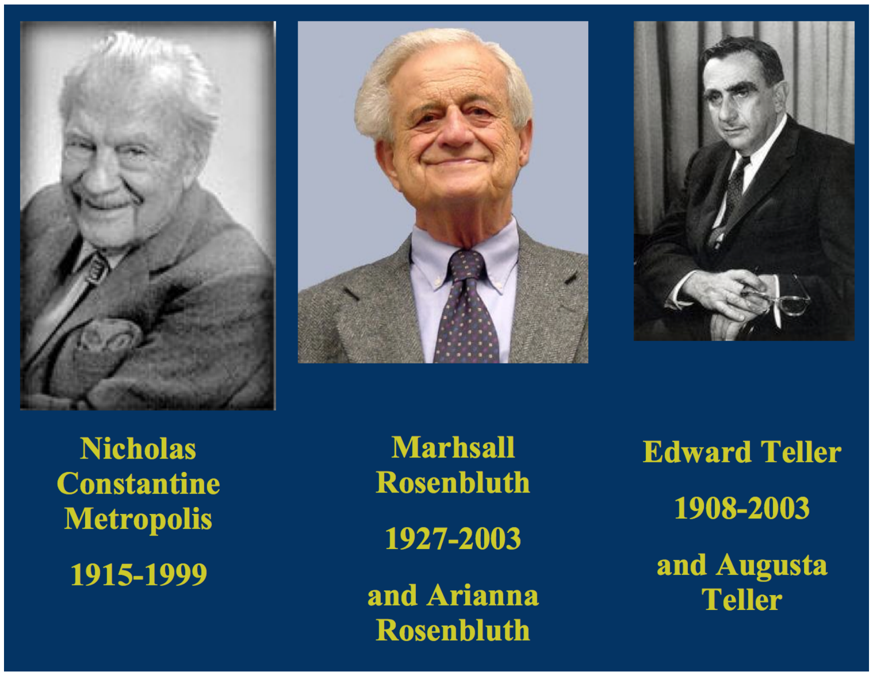
  ∗ Metropolis, Rosenbluth, Rosenbluth, Teller and Teller (1953) Equation of state calculation by fast computing machines, *The Journal of Chemical Physics*, 21, 1087–1092.

  ∗ Want to sample from a distribution $\pi(x) \propto f(x)$. Metropolis algorithm constructs a Markov chain that converges to π.

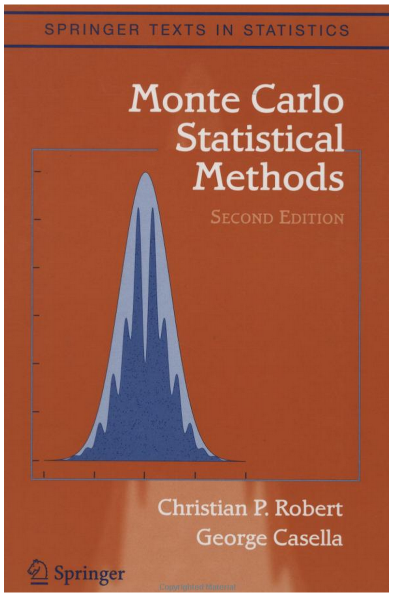
  ∗ *Metropolis chain*: From current state x, generate a new state x' (from a

proposal distribution $p(x, x')$ such that $p(x, x') = p(x', x)$) and accept x' with probability $\min\left\{\frac{f(x')}{f(x)}, 1\right\}$.
Fact: Metropolis chain has π as stationary distribution.

$*$ Marc Coram's "jail message" example.



- Given π, generic ways to construct a Markov chain K that has π as stationary distribution:
  - Metropolis algorithm: (Metropolis et al., 1953)
  - Hastings algorithm: (Hastings, 1970)
  - Gibbs sampler: Glauber dynamics (1963), Tanner and Wong (1987), Gelfand and Smith (1990)
- See KL Ch25-27 and JM Ch13 for a general introduction, or take a Bayesian course. A comprehensive textbook is (Robert and Casella, 2004).



Convergence rate of Markov chains

- Classical result: For finite, irreducible, and aperiodic Markov chains

$$\lim_{l\to\infty} K^l(x,y) = \pi(y).$$

  In practice, we often want to know *how many* steps to make the difference between $K^l(x,\cdot)$ and π small?

- Example: "How many shuffles do I need to do to mix a deck of 52 cards?"

Consider *riffle shuffle.* Gilbert-Shannon-Reeds model: binomial(52, 0.5) cut + cards drop according to probability $L/(L+R)$, where L and R are the number of cards in the left and right hand respectively.

- Example: "How long do I need to run my Gibbs sampler?"

  Consider the Beta-Binomial Gibbs sampler

  - Likelihood $f(x \mid p) \sim \text{Bin}(n,p)$ and prior $\pi(p) \sim \text{Beta}(\alpha,\beta)$
  - Want to sample from joint density $f(x,p) = f(x \mid p)\pi(p)$
  - *Gibbs sampler*: Repeat the following
    - Sample x from $\text{Bin}(n,p)$
    - Sample p from $\text{Beta}(x+\alpha, n-x+\beta)$
  - $(X_l, p_l)_{l\geq 1}$ form a Markov chain on $\{0,1,\ldots,n\} \times [0,1]$
  - Let $\tilde{K}(x,p;x',p')$ be the transition density
  - How many steps (obviously depending on n,α,β) does this Markov chain converge to the stationary distribution?

- In a typical Bayesian course, we learn many convergence diagnostics that are often heuristic. Is there anything rigorous we can say about convergence rate of Markov chains?

- Distances between distributions:

  - *Total variation distance*:

$$\begin{aligned}\|\mu-\pi\|_{\text{TV}} &= \frac{1}{2}\sum_{x\in\mathcal{X}} |\mu(x)-\pi(x)| \\ &= \max_{A\subset\mathcal{X}} |\mu(A)-\pi(A)| \\ &= \frac{1}{2}\sup_{\|f\|_\infty\leq 1} |\mu(f)-\pi(f)|.\end{aligned}$$

  - L^p *distance* wrt π:
    Let $f(x) = \frac{\mu(x)}{\pi(x)}$ and $g(x) = \frac{\nu(x)}{\pi(x)}$. For $1 \leq p < \infty$,

$$d_{\pi,p}(\mu,\nu) = \|f-g\|_{L^p(\pi)} = \left(\sum_{x\in\mathcal{X}} |f(x)-g(x)|^p \pi(x)\right)^{1/p}$$

- The usual limit theorems are useless in practice:

  "There exist constants $A, B > 0$, $\rho \in (0,1)$ such that $\|K^l(x,\cdot) - \pi\|_{TV} \leq A\rho^{Bl}$."

A, B, and ρ are some mysterious constants.

Can we get some more quantitative, useful bounds?

- Cutoff phenomenon (Diaconis, 1996)

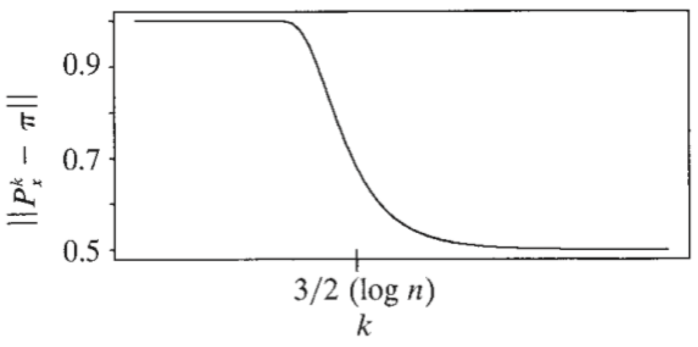


FIG. 1. **The cutoff phenomenon for repeated riffle shuffles of n = 52 cards.**

  - Riffle shuffle.

l	1	2	3	4	5	6	7	8	9	10
$\|K^l - \pi\|_{TV}$	1.000	1.000	1.000	1.000	0.924	0.624	0.312	0.161	0.083	0.041

  - Many more examples: Ehrenfest chain, random transposition, Gibbs sampler, ...

    "Cutoff phenomenon for XXX chain."

  - Not every Markov chain has a cutoff.
    A chain without cutoff: simple random walk on the integers mod n.

- Generic methods for studying convergence rates:

  - Algebraic methods (spectral analysis) - E.g, random walks on groups (shuffling cards), some Gibbs samplers, ...
  - Analytic methods - Geometric Inequalities.
  - Probabilistic methods - Coupling and strong stationary times.

- Algebraic method

- *Reversible* Markov chains.

  If π is a probability distribution on $\mathcal{X}$ and

  $$\pi(x)K(x,y) = \pi(y)K(y,x), \quad \text{for all } x, y \in \mathcal{X},$$

  then π is the unique stationary distribution of K. E.g., Metropolis chain.

- K operates on $L^2(\pi) = \{f : \mathcal{X} \mapsto \mathbb{R}, \mathbf{E}_\pi[f^2] < \infty\}$ by

  $$Kf(x) = \sum_{y \in \mathcal{X}} K(x,y)f(y).$$

- Reversibility of K is equivalent to *self-adjointness* of the operator K.
- By standard spectral theorem for self-adjoint operators, K has eigenvalues $1 = \beta_0 \geq \beta_1 \geq \cdots \geq \beta_{|\mathcal{X}|-1} \geq -1$ with (right) eigenfunctions $\{\phi_0, \ldots, \phi_{|\mathcal{X}|-1}\}$ that are orthonormal on $L^2(\pi)$

  $$< \phi_i, \phi_j >_{L^2(\pi)} = \sum_{x \in \mathcal{X}} \phi_i(x)\phi_j(x)\pi(x) = 1_{\{i=j\}}.$$

- If we know all the spectral information (lucky!), then

  $$d^2_{\pi,2}(K^l(x,\cdot),\pi) = \sum_{i=1}^{|\mathcal{X}|-1} \beta_i^{2l}\phi_i^2(x).$$

- Usually the upper bound is tight.

  $$\|K^l(x,\cdot) - \pi\|_{TV} \leq \frac{1}{2} d_{\pi,2}(K^l(x,\cdot),\pi)$$

- When are we lucky? In presence of *symmetry.*

  E.g., for random walks on groups, we only need eigenvalues, which can be derived from the irreducible representations of the group.

  Definite reference for this topic is the book Diaconis (1988)

• Algebraic method for analyzing the baby Gibbs sampler

  - The joint chain $\tilde{K}(x,p;x',p')$ is *irreversible.*
  - The x-marginal chain $K(x,x')$ is *reversible*, with $m \sim$Beta-Bin(n,α,β) as stationary distribution. And

    $$\|K_x^l - m\|_{TV} \leq \|\tilde{K}_{x,p}^l - f\|_{TV} \leq \|K_x^{l-1} - m\|_{TV}.$$

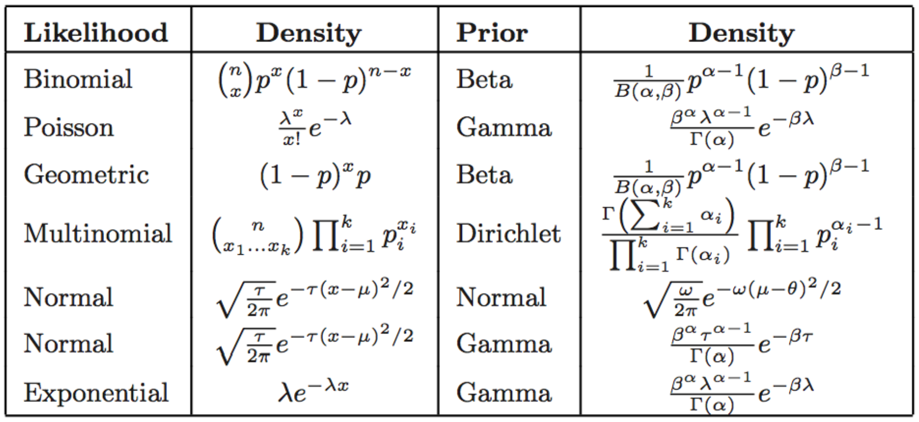
    Thus sufficient to study convergence rate of the x-marginal chain.

- Some analysis (Diaconis et al., 2008) shows that K has
  * eigenvalues: $\beta_0 = 1$, $\beta_j = n_{[j]}/(n+\alpha+\beta)_{(j)}$, $j = 1, \ldots, n$.
  * eigen-functions: ϕ_j are the *Hahn polynomials.*
- Doing the summation gives

$$0.5\beta_1^l \le \|\tilde{K}_{0,p}^l - f\|_{TV} \le 3\beta_1^{l-1/2}.$$

- Cutoff phenomenon: $\frac{n+\alpha+\beta}{2(\alpha+\beta)}$ steps are necessary and sufficient for convergence.
- Similar analysis can be carried out for all following conjugate pairs (see KL 27.9)

TABLE 26.1. Conjugate Pairs



Likelihood	**Density**	**Prior**	**Density**
Binomial	$\binom{n}{x}p^x(1-p)^{n-x}$	Beta	$\frac{1}{B(\alpha,\beta)}p^{\alpha-1}(1-p)^{\beta-1}$
Poisson	$\frac{\lambda^x}{x!}e^{-\lambda}$	Gamma	$\frac{\beta^\alpha\lambda^{\alpha-1}}{\Gamma(\alpha)}e^{-\beta\lambda}$
Geometric	$(1-p)^x p$	Beta	$\frac{1}{B(\alpha,\beta)}p^{\alpha-1}(1-p)^{\beta-1}$
Multinomial	$\binom{n}{x_1 \ldots x_k}\prod_{i=1}^k p_i^{x_i}$	Dirichlet	$\frac{\Gamma\left(\sum_{i=1}^k \alpha_i\right)}{\prod_{i=1}^k \Gamma(\alpha_i)}\prod_{i=1}^k p_i^{\alpha_i-1}$
Normal	$\sqrt{\frac{\tau}{2\pi}}e^{-\tau(x-\mu)^2/2}$	Normal	$\sqrt{\frac{\omega}{2\pi}}e^{-\omega(\mu-\theta)^2/2}$
Normal	$\sqrt{\frac{\tau}{2\pi}}e^{-\tau(x-\mu)^2/2}$	Gamma	$\frac{\beta^\alpha\tau^{\alpha-1}}{\Gamma(\alpha)}e^{-\beta\tau}$
Exponential	$\lambda e^{-\lambda x}$	Gamma	$\frac{\beta^\alpha\lambda^{\alpha-1}}{\Gamma(\alpha)}e^{-\beta\lambda}$

- Analytic method
  - Upper bound through spectral gap

$$4\|K^l(x,\cdot) - \pi\|_{TV} \le d_{\pi,2}^2(K^l(x,\cdot),\pi) \le \frac{1}{\pi(x)}\beta_*^{2l},$$

    where $\beta_* = \max\{|\beta_1|, |\beta_{|\mathcal{X}|-1}|\}$.
  - Use some geometric inequalities to bound β_*.
  - Where to look up the material? Lecture notes by Saloff-Coste (1997).
  - (K,π) reversible on $\mathcal{X}$ (finite). Dirichlet form

$$\epsilon(f,g) =< (I-K)f, g > .$$

Fact

$$\epsilon(f,f) = \frac{1}{2}\sum_{x,y}[f(x)-f(y)]^2\pi(x)K(x,y).$$

- Lemma

$$\begin{aligned} 1-\beta_1 &= \min_{f\neq 1}\frac{\epsilon(f,f)}{\operatorname{var}(f)} \\ 1-\beta_{|\mathcal{X}|-1} &= \max_{f\neq 0}\frac{\epsilon(f,f)}{\operatorname{var}(f)}. \end{aligned}$$

- Definition: Poincaré inequality

$$\operatorname{var}(f) \le A\epsilon(f,f), \quad \text{for all } f \in L^2(\pi).$$

- We can bound β_1 by finding A

$$\beta_1 \le 1-\frac{1}{A}.$$

- Theorem (Poincaré Ineq for Markov Chains (Diaconis and Stroock, 1991)):

$$\beta_1 \le 1-\frac{1}{A}, \quad \text{where } A = \max_e \frac{1}{Q(e)}\sum_{\gamma_{x,y}\ni e}|\gamma_{x,y}|\pi(x)\pi(y).$$

- Apply Poincaré inequality to the baby Gibbs sampler gives a horrible bound (something *exponential* ...)
- Other geometric inequalities: Cheeger, Nash, Log-Soblov inequalities (in a series of papers by Diaconis and Saloff-Coste).

• Probabilistic methods

  - By cleverness, we can get good bounds for convergence rate without using all those analytic methods.
  - Good starting point is the book (Diaconis, 1988) and the unpublished book by Aldous and Fill (available on Aldous' website).
  - *Coupling* - Wolfgang Doeblin.

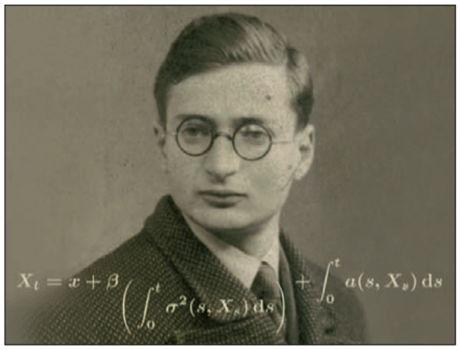
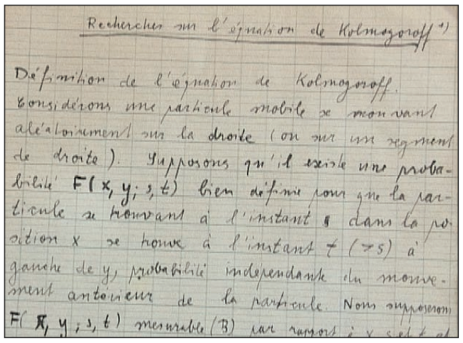

$X_t = x + \beta\left(\int_0^t \sigma^2(s, X_s)\,ds\right) + \int_0^t a(s, X_s)\,ds$

Fig. 20: Cover of the DVD: "Wolfgang Doeblin. A mathematician rediscovered"

Fig. 21: The archive of the "Académie des Sciences



Recherches sur l'équation de Kolmogoroff *)

Définition de l'équation de Kolmogoroff.
Considérons une particule mobile se mouvant
aléatoirement sur la droite (ou sur un segment
de droite). Supposons qu'il existe une proba-
bilité $F(x, y; s, t)$ bien définie pour que la par-
ticule se trouvant à l'instant s dans la po-
sition x se trouve à l'instant t (>s) à
gauche de y, probabilité indépendante du mouve-
ment antérieur de la particule. Nous supposerons
$F(x, y; s, t)$ mesurable (B) par rapport à x

Fig.22: First page of the "pli cacheté no. 11668"

- For the baby Gibbs sampler, coupling gives an upper bound of order $n \ln n$ (off by $\ln n$).
- Yields useful bounds for hierarchical random effects model (Hobert and Geyer, 1998).
- Let's work on a simpler example: Borel's Shuffle (random to top, random to bottom, ...).
- *Coupling*: Two processes evolve until they are equal. *Coupling time* T. Coupling inequality

$$\|K_x^l - \pi\|_{TV} \leq \mathbf{P}(T > l).$$

- For Borel's shuffle, bound on coupon collector problem gives

$$\|K_x^l - \pi\|_{TV} \leq n\left(1 - \frac{1}{n}\right)^l.$$

Thus $l = n \ln n$ steps suffice.

- Summary
  - Keep Markov chains in your toolbox - useful for modeling, simulation, and combinatorial optimization.
  - Convergence rate of Markov chains is an interesting applied probability problem that often gives more insights into the chains.
  - A lot remains to be done for analyzing many MCMC algorithms being used.

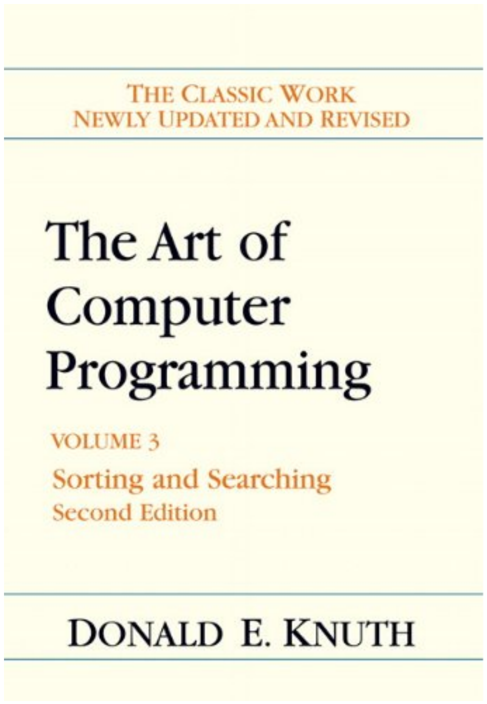
Sorting algorithms (JM 14.3, KL 1.10)

- Applications: order statistics (median, quantiles), QQ-plot, multiple testing (sorting p-values), Wilcoxon rank-sum test, ...
- *Bubble sort*: Locate maximum and put on top; find maximum in the $(n-1)$ list and put on the top second position; ...

  $O(n^2)$ average cost.
- Think about sorting *massive* data $n = 10^{12}$. On a teraflop computer. n^2 flops take 10^{12} seconds ≈ 31710 years, while $n \ln n$ flops take $10^{12} \log(10^{12})/10^{12} \approx 27$ seconds.
- Key idea: Divide and conquer.
- *Merge sort*: Recursively partition into two lists, sort them respectively, and then merge. $T(n) = 2T(n/2) + O(n)$. Solution is $T(n) = O(n \log_2 n)$.
- *Quick sort*: Randomly select a pivot element, split into 3 lists, and do some swaps so that the pivot is in the right position.

$$T(n) = \frac{1}{n} \sum_{j=1}^{n-1} [T(j-1) + T(n-j)] + n - 1 = \frac{2}{n} \sum_{j=1}^{n-1} T(j) + n - 1.$$

  Solution is $O(n \ln n)$.
- Sorting is a well-trodden area in computer science. Mature functions/libraries in standard softwares. The “bible” on this topic is (of course) Knuth (2005).



THE CLASSIC WORK
NEWLY UPDATED AND REVISED

The Art of Computer Programming

VOLUME 3

Sorting and Searching

Second Edition

DONALD E. KNUTH

Fast Fourier transform (FFT) (KL Chapter 20, JM 14.5)

- History: Cooley and Tukey (1965)

  John Tukey: "bit", box-plot, "learning from the experience of the others", multiple comparison, FFT, ...

  Tukey conceived the FFT algorithm during meetings with President JFK's Science Advisory Committee. They need fast ways to analyze seismic waves to detect nuclear weapon tests in Soviet Union. Richard Garwin of IBM immediately realize the potential of this fast algorithm and referred Tukey to Cooley to implement it.

  People also believe Gauss essentially used the same strategy when solving his least squares problem!

- Applications in statistics: convolution, time series, branching process, ...

- Consider two independent random variables on $\{0, 1, \ldots, N-1\}$:

$$X \sim \{p_0, \ldots, p_{N-1}\}, \quad Y \sim \{q_0, \ldots, q_{N-1}\}.$$

  What's the distribution of the sum $Z = X + Y$?

  - $z_k = \sum_{j=0}^{k} p_j q_{k-j}$, $k = 0, \ldots, 2N-2$. $O(N^2)$ computation.
  - Do DFT of both sequences, multiply together, and inverse DFT. $O(N \ln N)$ computation!

- Discrete Fourier transform (DFT) of a vector $\boldsymbol{x} \in \mathbb{R}^N$.

$$a_k = \sum_{j=0}^{N-1} w^{jk} x_j, \quad k = 0, \ldots, N-1.$$

where $w = e^{-2\pi\sqrt{-1}/N}$. Note w is an N-th root of 1. DFT is essentially matrix-vector multiplication

$$\boldsymbol{a}^\intercal = \boldsymbol{x}^\intercal \boldsymbol{W}, \quad \boldsymbol{W} = (w^{jk}),$$

which usually costs $O(N^2)$ flops.

- Suppose $N = N_1 N_2$. Index rewriting:
  - $j \leftarrow j_1 N_2 + j_2$ (fill out N_1-by-N_2 matrix in row major),
  - $k \leftarrow k_2 N_1 + k_1$ (fill out N_1-by-N_2 matrix in column major),
  - $j_1, k_1 \in \{0, \ldots, N_1 - 1\}$, $j_2, k_2 \in \{0, \ldots, N_2 - 1\}$.

Then

$$\begin{aligned}
a_k &= a_{k_2 N_1 + k_1} = \sum_{j=0}^{N-1} w^{jk} x_j \\
&= \sum_{j_1=0}^{N_1-1} \sum_{j_2=0}^{N_2-1} w^{(j_1 N_2 + j_2)(k_2 N_1 + k_1)} x_{j_1 N_2 + j_2} \\
&= \sum_{j_1=0}^{N_1-1} \sum_{j_2=0}^{N_2-1} w^{j_1 k_1 N_2 + j_2(k_2 N_1 + k_1)} x_{j_1 N_2 + j_2} \\
&= \sum_{j_2=0}^{N_2-1} w^{j_2(k_2 N_1 + k_1)} \sum_{j_1=0}^{N_1-1} (w^{N_2})^{j_1 k_1} x_{j_1 N_2 + j_2} \\
&= \sum_{j_2=0}^{N_2-1} (w^{N_1})^{j_2 k_2} w^{j_2 k_1} \sum_{j_1=0}^{N_1-1} (w^{N_2})^{j_1 k_1} x_{j_1 N_2 + j_2}.
\end{aligned}$$

- Essentially we need to do N_2 DFT of length-N_1 sequences and then do N_1 DFT of length-N_2 sequences. Total cost

$$T(N) = T(N_1N_2) = N_2T(N_1) + N_1T(N_2).$$

Suppose N is a power of 2. Then $T(N) = (N/2)T(2)+2T(N/2)$ and the solution is $T(N) = O(N \ln N)$!.

- Inverse DFT. $\boldsymbol{W}^{-1}$ has entries w^{-jk}/N. Then

$$\boldsymbol{x}^{\mathsf{T}} = \boldsymbol{a}^{\mathsf{T}}\boldsymbol{W}^{-1}.$$

- Variants for prime N: still $O(N \ln N)$. But always a good idea to pad with zero to get N as a power of 2.

- Generalizations to 2D, 3D FFT available.

- Mature libraries/functions for both CPU and GPU.

- Galton-Watson process. Survival of families. Lotka data (Lotka, 1931a,b). Using 1920 census data, the *progeny generating function* for a white male

$$\begin{aligned} P(s) &= .4982 + .2103s + .1270s^2 + .0730s^3 + .0418s^4 + .0241s^5 \\ &\quad +.0132s^6 + .0069s^7 + 0.0035s^8 + .0015s^9 + .0005s^{10}. \end{aligned}$$

PGF for the first generation $P_1(s) = P(s)$. PGF for the second generation $P_2(s) = \sum_k p_k P(s)^k = P(P(s))$. In general, PGF for the i-th generation is $P_i(s) = P(\cdots P(s))$ (i recursions).

Extinction probability of a family: $\lim_{i\to\infty} P_i(0) = P(\cdots P(0)) = 0.88$, or solving for $P(s) = s$.

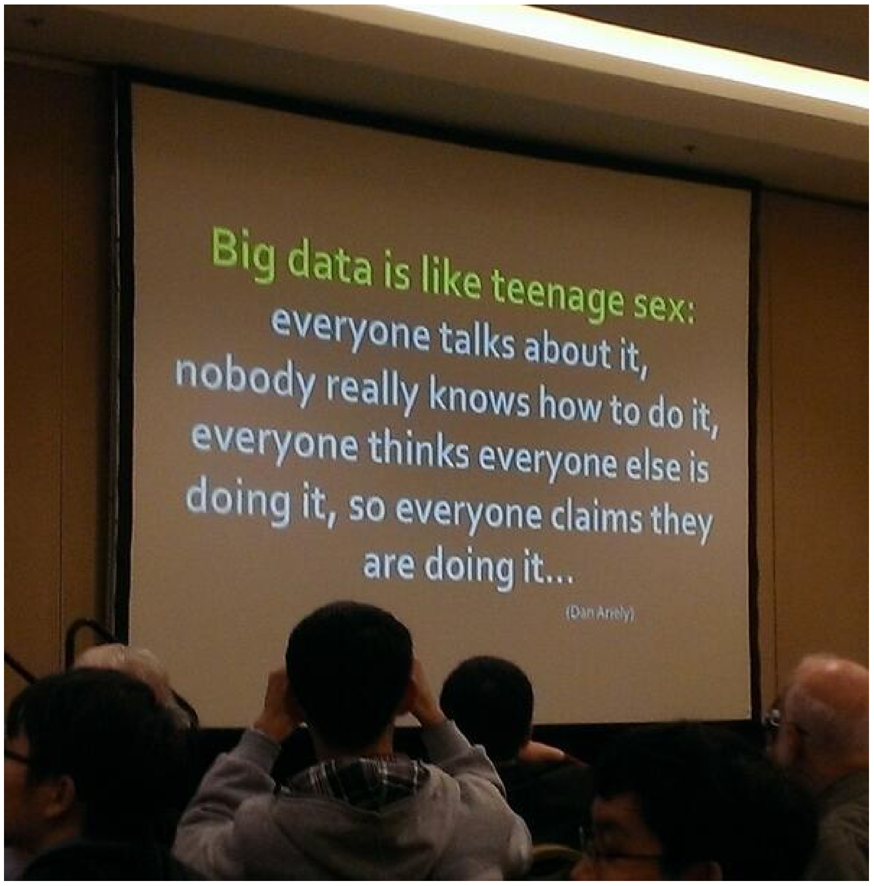
What if we want to know the distribution of the i-th generation? Extend the generating function P_i to unit circle $P_i(w^k) = \sum_j p_j w^{jk}$, $w = e^{-2\pi\sqrt{-1}/N}$, where $k = 0, \ldots, N-1$ for N large. So $P_i(w^k)$ is the DFT of distribution p_j of i-th generation. Then apply inverse DFT to retrieve p_j. $O(N \log N)$ cost!

- Continuous-time branching process. Solve differential equation for $P_t(w^j)$ at any time t. Then apply inverse DFT.

- See JM 14.7 for more applications of FFT in statistics.

Take-home messages from this course

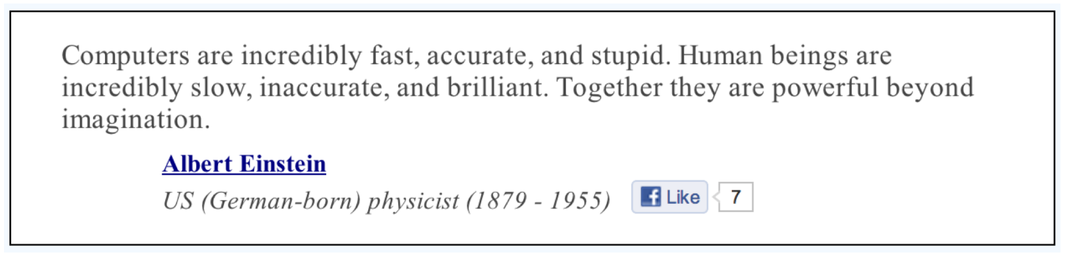
- Statistics, the science of *data analysis*, is the applied mathematics in the 21st century
  - Read the first and last few pages of Tukey (1962)'s *Future of data analysis*. `http://www.stat.ncsu.edu/people/zhou/courses/st810/notes/Tukey61FutureDataAnalysis.pdf`.
- *Big data* era: Challenges also mean opportunities for statisticians
  - methodology: big p
  - efficiency: big n and/or big p
  - memory: big n, distributed computing via MapReduce (Hadoop), online algorithms



Big data is like teenage sex:
everyone talks about it,
nobody really knows how to do it,
everyone thinks everyone else is
doing it, so everyone claims they
are doing it...

(Dan Ariely)

- Being good at computing (*both* programming and algorithms) is a must for today's working statisticians.



Computers are incredibly fast, accurate, and stupid. Human beings are incredibly slow, inaccurate, and brilliant. Together they are powerful beyond imagination.

Albert Einstein

US (German-born) physicist (1879 - 1955)

- HPC (high performance computing) $\neq$ abusing computers.

  Always optimize your algorithms *as much as possible* before resorting to cluster computing resources.

- Coding
  - Prototyping: R, Matlab, Julia
  - A "real" programming language: C/C++, Fortran, Python
  - Scripting language: Python, Linux/Unix script, Perl, JavaScript

- Numerical linear algebra – building blocks of most computing we do. Use standard *libraries* (BLAS, LAPACK, ...)! Sparse linear algebra and iterative solvers such as conjugate gradient methods are critical for exploiting structure in big data.

- Optimization
  - *Convex programming* (LS, LP, QP, GP, SOCP, SDP). To do in ST790-003. Convex programming is becoming a *technology*, just like least squares (LS).
  - Specialized optimization algorithms for modern statistical learning problems. To do in ST790-003.
  - Generic nonlinear optimization tools: Newton, quasi-Newton, (nonlinear) conjugate gradient, ...
  - Specialized tools in statistics: EM/MM, Fisher scoring, Gauss-Newton, simulated annealing, ...

  - Combinatorial optimization techniques: divide-and-conquer, dynamic programming, greedy algorithm, ...

- MCMC: take a Bayesian course!

References

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.

Benjamini, Y. and Hochberg, Y. (1995). Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society, Ser. B*, 57:289–300.

Björck, Å. (1996). *Numerical methods for least squares problems.* Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA.

Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization.* Cambridge University Press, Cambridge.

Cooley, J. W. and Tukey, J. W. (1965). An algorithm for the machine calculation of complex Fourier series. *Math. Comp.*, 19:297–301.

Cox, D. R. (1972). Regression models and life-tables. *J. Roy. Statist. Soc. Ser. B*, 34:187–220. With discussion by F. Downton, Richard Peto, D. J. Bartholomew, D. V. Lindley, P. W. Glassborow, D. E. Barton, Susannah Howard, B. Benjamin, John J. Gart, L. D. Meshalkin, A. R. Kagan, M. Zelen, R. E. Barlow, Jack Kalbfleisch, R. L. Prentice and Norman Breslow, and a reply by D. R. Cox.

Davidon, W. C. (1991). Variable metric method for minimization. *SIAM J. Optim.*, 1(1):1–17.

Dempster, A., Laird, N., and Rubin, D. (1977a). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Soceity Series B.*, 39(1-38).

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977b). Maximum likelihood from incomplete data via the EM algorithm. *J. Roy. Statist. Soc. Ser. B*, 39(1):1–38. With discussion.

Diaconis, P. (1988). *Group representations in probability and statistics.* Institute of Mathematical Statistics Lecture Notes—Monograph Series, 11. Institute of Mathematical Statistics, Hayward, CA.

Diaconis, P. (1996). The cutoff phenomenon in finite Markov chains. *Proc. Nat. Acad. Sci. U.S.A.*, 93(4):1659–1664.

Diaconis, P. (2009). The Markov chain Monte Carlo revolution. *Bull. Amer. Math. Soc. (N.S.)*, 46(2):179–205.

Diaconis, P., Khare, K., and Saloff-Coste, L. (2008). Gibbs sampling, exponential families and orthogonal polynomials. *Statist. Sci.*, 23(2):151–200. With comments and a rejoinder by the authors.

Diaconis, P. and Stroock, D. (1991). Geometric bounds for eigenvalues of Markov chains. *Ann. Appl. Probab.*, 1(1):36–61.

Dickey, D. A. and Fuller, W. A. (1979). Distribution of the estimators for autoregressive time series with a unit root. *J. Amer. Statist. Assoc.*, 74(366, part 1):427–431.

Efron, B. (1979). Bootstrap methods: another look at the jackknife. *Ann. Statist.*, 7(1):1–26.

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.

Gelfand, A. E. and Smith, A. F. M. (1990). Sampling-based approaches to calculating marginal densities. *J. Amer. Statist. Assoc.*, 85(410):398–409.

Gentle, J. E. (2007). *Matrix Algebra*. Springer Texts in Statistics. Springer, New York. Theory, computations, and applications in statistics.

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.

Golub, G. H. and Van Loan, C. F. (2013). *Matrix Computations*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, Baltimore, MD, fourth edition.

Grant, M. and Boyd, S. (2012). CVX: Matlab software for disciplined convex programming, version 2.0 beta. `http://cvxr.com/cvx`.

Harville, D. A. (1997). *Matrix Algebra From a Statistician's Perspectives*. Springer-Verlag, New York.

Hastings, W. K. (1970). Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109.

Hobert, J. P. and Geyer, C. J. (1998). Geometric ergodicity of Gibbs and block Gibbs samplers for a hierarchical random effects model. *J. Multivariate Anal.*, 67(2):414–430.

Horn, R. A. and Johnson, C. R. (1985). *Matrix Analysis*. Cambridge University Press, Cambridge.

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.

Kaplan, E. L. and Meier, P. (1958). Nonparametric estimation from incomplete observations. *J. Amer. Statist. Assoc.*, 53:457–481.

Knuth, D. E. (2005). *The Art of Computer Programming. Vol. 1. Fasc. 1*. Addison-Wesley, Upper Saddle River, NJ. MMIX, a RISC computer for the new millennium.

Lange, K. (2010). *Numerical Analysis for Statisticians*. Statistics and Computing. Springer, New York, second edition.

Lange, K. and Carson, R. (1984). EM reconstruction algorithms for emission and transmission tomography. *J. Comput. Assist. Tomogr.*, 8(2):306–316.

Lange, K., Hunter, D. R., and Yang, I. (2000). Optimization transfer using surrogate objective functions. *J. Comput. Graph. Statist.*, 9(1):1–59. With discussion, and a rejoinder by Hunter and Lange.

Lawson, C. L. and Hanson, R. J. (1987). *Solving Least Squares Problems.* Classics in Applied Mathematics. Society for Industrial Mathematics, new edition edition.

Lotka, A. (1931a). Population analysis - the extinction of families i. *J. Wash. Acad. Sci.*, 21:377–380.

Lotka, A. (1931b). Population analysis - the extinction of families ii. *J. Wash. Acad. Sci.*, 21:453–459.

Magnus, J. R. and Neudecker, H. (1999). *Matrix Differential Calculus with Applications in Statistics and Econometrics.* Wiley Series in Probability and Statistics. John Wiley & Sons Ltd., Chichester.

Mazumder, R., Hastie, T., and Tibshirani, R. (2010). Spectral regularization algorithms for learning large incomplete matrices. *Journal of Machine Learning Research*, 11:2287–2322.

McLachlan, G. J. and Krishnan, T. (2008). *The EM Algorithm and Extensions.* Wiley Series in Probability and Statistics. Wiley-Interscience [John Wiley & Sons], Hoboken, NJ, second edition.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953). Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092.

Metropolis, N. and Ulam, S. (1949). The Monte Carlo method. *J. Amer. Statist. Assoc.*, 44:335–341.

Nocedal, J. and Wright, S. J. (2006). *Numerical Optimization.* Springer Series in Operations Research and Financial Engineering. Springer, New York, second edition.

Novembre, J., Johnson, T., Bryc, K., Kutalik, Z., Boyko, A. R., Auton, A., Indap, A., King, K. S., Bergmann, S., Nelson, M. R., Stephens, M., and Bustamante, C. D. (2008). Genes mirror geography within europe. *Nature*, 456(7218):98–101.

Price, A. L., Patterson, N. J., Plenge, R. M., Weinblatt, M. E., Shadick, N. A., and Reich, D. (2006). Principal components analysis corrects for stratification in genome-wide association studies. *Nature Genetics*, 38(8):904–909.

Robert, C. P. and Casella, G. (2004). *Monte Carlo Statistical Methods.* Springer Texts in Statistics. Springer-Verlag, New York, second edition.

Saad, Y. (2003). *Iterative Methods for Sparse Linear Systems.* Society for Industrial and Applied Mathematics, Philadelphia, PA, second edition.

Saloff-Coste, L. (1997). Lectures on finite Markov chains. In *Lectures on probability theory and statistics (Saint-Flour, 1996)*, volume 1665 of *Lecture Notes in Math.*, pages 301–413. Springer, Berlin.

Seneta, E. (1996). Markov and the birth of chain dependence theory. *International Statistical Review*, 64:255–263.

Stein, M., Chen, J., and Anitescu, M. (2012). Difference filter preconditioning for large covariance matrices. *SIAM Journal on Matrix Analysis and Applications*, 33(1):52–72.

Tanner, M. A. and Wong, W. H. (1987). The calculation of posterior distributions by data augmentation. *J. Amer. Statist. Assoc.*, 82(398):528–550. With discussion and with a reply by the authors.

Teets, D. and Whitehead, K. (1999). The discovery of Ceres: how Gauss became famous. *Math. Mag.*, 72(2):83–93.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *J. Roy. Statist. Soc. Ser. B*, 58(1):267–288.

Tukey, J. W. (1962). The future of data analysis. *Ann. Math. Statist.*, 33:1–67.

Vardi, Y., Shepp, L. A., and Kaufman, L. (1985). A statistical model for positron emission tomography. *J. Amer. Statist. Assoc.*, 80(389):8–37. With discussion.