12. Cholesky factorization

- positive definite matrices
- examples
- Cholesky factorization
- complex positive definite matrices
- kernel methods

Definitions

- a symmetric matrix $A \in \mathbf{R}^{n \times n}$ is *positive semidefinite* if

$$x^T A x \geq 0 \quad \text{for all } x$$

- a symmetric matrix $A \in \mathbf{R}^{n \times n}$ is *positive definite* if

$$x^T A x > 0 \quad \text{for all } x \neq 0$$

  this is a subset of the positive semidefinite matrices

note: if A is symmetric and $n \times n$, then $x^T A x$ is the function

$$x^T A x = \sum_{i=1}^{n} \sum_{j=1}^{n} A_{ij} x_i x_j = \sum_{i=1}^{n} A_{ii} x_i^2 + 2 \sum_{i>j} A_{ij} x_i x_j$$

this is called a *quadratic form*

Example

$$A = \begin{bmatrix} 9 & 6 \\ 6 & a \end{bmatrix}$$

$$x^T A x = 9x_1^2 + 12x_1x_2 + ax_2^2 = (3x_1 + 2x_2)^2 + (a-4)x_2^2$$

- A is positive definite for $a > 4$

$$x^T A x > 0 \quad \text{for all nonzero } x$$

- A is positive semidefinite but not positive definite for $a = 4$

$$x^T A x \geq 0 \quad \text{for all } x, \qquad x^T A x = 0 \quad \text{for } x = (2, -3)$$

- A is not positive semidefinite for $a < 4$

$$x^T A x < 0 \quad \text{for } x = (2, -3)$$

Simple properties

- every positive definite matrix A is nonsingular

$$Ax = 0 \quad \Longrightarrow \quad x^T A x = 0 \quad \Longrightarrow \quad x = 0$$

(last step follows from positive definiteness)

- every positive definite matrix A has positive diagonal elements

$$A_{ii} = e_i^T A e_i > 0$$

- every positive semidefinite matrix A has nonnegative diagonal elements

$$A_{ii} = e_i^T A e_i \geq 0$$

Schur complement

partition $n \times n$ symmetric matrix A as

$$A = \begin{bmatrix} A_{11} & A_{2:n,1}^T \\ A_{2:n,1} & A_{2:n,2:n} \end{bmatrix}$$

- the *Schur complement* of A_{11} is defined as the $(n-1) \times (n-1)$ matrix

$$S = A_{2:n,2:n} - \frac{1}{A_{11}} A_{2:n,1} A_{2:n,1}^T$$

- if A is positive definite, then S is positive definite

  to see this, take any $x \neq 0$ and define $y = -(A_{2:n,1}^T x)/A_{11}$; then

$$x^T S x = \begin{bmatrix} y \\ x \end{bmatrix}^T \begin{bmatrix} A_{11} & A_{2:n,1}^T \\ A_{2:n,1} & A_{2:n,2:n} \end{bmatrix} \begin{bmatrix} y \\ x \end{bmatrix} > 0$$

  because A is positive definite

Singular positive semidefinite matrices

- we mentioned that positive definite matrices are nonsingular (page 12.4)
- if A is positive semidefinite, but not positive definite, then it is singular

to see this, suppose A is positive semidefinite but not positive definite

- there exists a nonzero x with $x^T Ax = 0$
- since A is positive semidefinite the following function is nonnegative:

$$\begin{aligned} f(t) &= (x - tAx)^T A(x - tAx) \\ &= x^T Ax - 2tx^T A^2 x + t^2 x^T A^3 x \\ &= -2t\|Ax\|^2 + t^2 x^T A^3 x \end{aligned}$$

- $f(t) \geq 0$ for all t is only possible if $\|Ax\| = 0$; therefore $Ax = 0$
- hence there exists a nonzero x with $Ax = 0$, so A is singular

Exercises

- show that if $A \in \mathbf{R}^{n\times n}$ is positive semidefinite, then

$$B^T AB$$

is positive semidefinite for any $B \in \mathbf{R}^{n\times m}$

- show that if $A \in \mathbf{R}^{n\times n}$ is positive definite, then

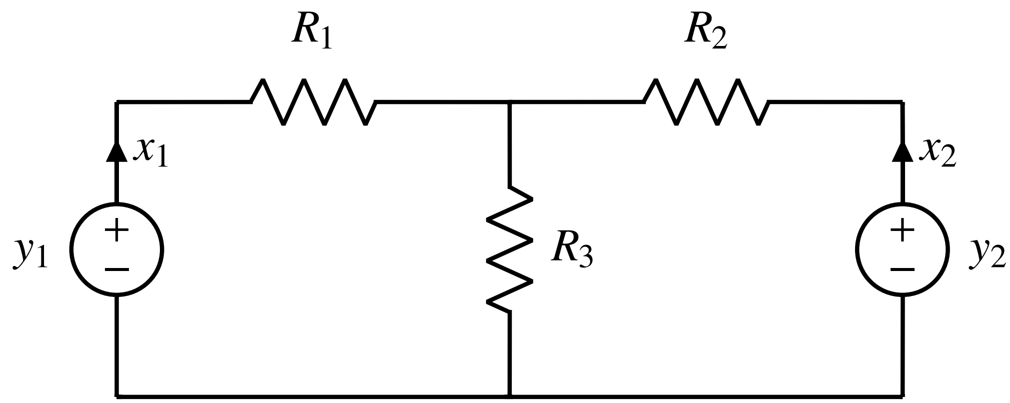
$$B^T AB$$

is positive definite for any $B \in \mathbf{R}^{n\times m}$ with linearly independent columns

Outline

- positive definite matrices
- **examples**
- Cholesky factorization
- complex positive definite matrices
- kernel methods

Exercise: resistor circuit



$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} R_1 + R_3 & R_3 \\ R_3 & R_2 + R_3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

show that the matrix

$$A = \begin{bmatrix} R_1 + R_3 & R_3 \\ R_3 & R_2 + R_3 \end{bmatrix}$$

is positive definite if R_1, R_2, R_3 are positive

Solution

Solution from physics

- $x^T A x = y^T x$ is the power delivered by sources, dissipated by resistors
- power dissipated by the resistors is positive unless both currents are zero

Algebraic solution

$$\begin{aligned} x^T A x &= (R_1 + R_3)x_1^2 + 2R_3 x_1 x_2 + (R_2 + R_3)x_2^2 \\ &= R_1 x_1^2 + R_2 x_2^2 + R_3(x_1 + x_2)^2 \\ &\geq 0 \end{aligned}$$

and $x^T A x = 0$ only if $x_1 = x_2 = 0$

Gram matrix

recall the definition of *Gram matrix* of a matrix B (page 4.20):

$$A = B^T B$$

- every Gram matrix is positive semidefinite

$$x^T A x = x^T B^T B x = \|Bx\|^2 \geq 0 \quad \forall x$$

- a Gram matrix is positive definite if

$$x^T A x = x^T B^T B x = \|Bx\|^2 > 0 \quad \forall x \neq 0$$

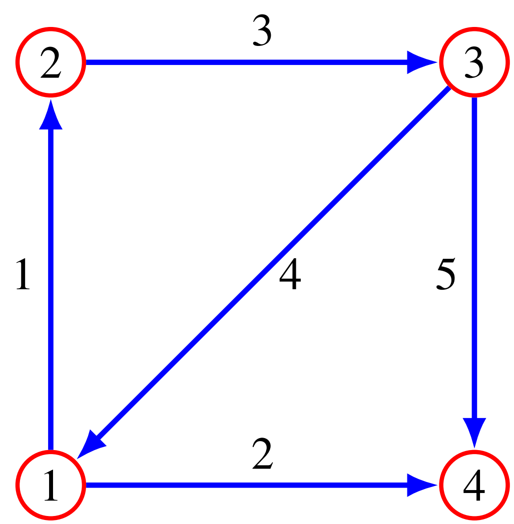
  in other words, B has linearly independent columns

Graph Laplacian

recall definition of node-arc incidence matrix of a directed graph (page 3.29)

$$B_{ij} = \begin{cases} 1 & \text{if edge } j \text{ ends at vertex } i \\ -1 & \text{if edge } j \text{ starts at vertex } i \\ 0 & \text{otherwise} \end{cases}$$

assume there are no self-loops and at most one edge between any two vertices



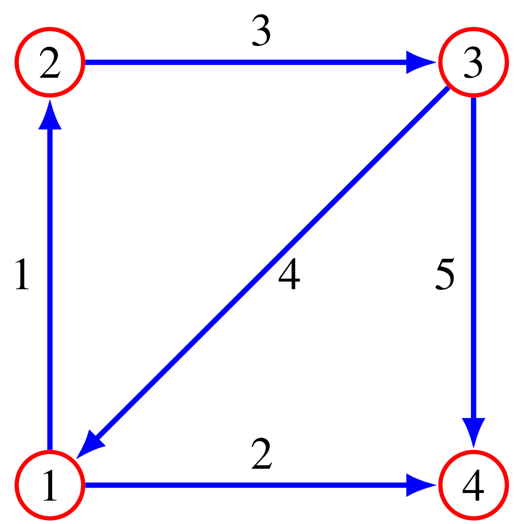
$$B = \begin{bmatrix} -1 & -1 & 0 & 1 & 0 \\ 1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 & -1 \\ 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

Graph Laplacian

the positive semidefinite matrix $A = BB^T$ is called the *Laplacian* of the graph

$$A_{ij} = \begin{cases} \text{degree of vertex } i & \text{if } i = j \\ -1 & \text{if } i \neq j \text{ and vertices } i \text{ and } j \text{ are adjacent} \\ 0 & \text{otherwise} \end{cases}$$

the degree of a vertex is the number of edges incident to it



$$A = BB^T = \begin{bmatrix} 3 & -1 & -1 & -1 \\ -1 & 2 & -1 & 0 \\ -1 & -1 & 3 & -1 \\ -1 & 0 & -1 & 2 \end{bmatrix}$$

Laplacian quadratic form

recall the interpretation of matrix–vector multiplication with B^T (page 3.31)

- if y is vector of node potentials, then $B^T y$ contains potential differences:

$$(B^T y)_j = y_k - y_l \quad \text{if edge } j \text{ goes from vertex } l \text{ to } k$$

- $y^T A y = y^T B B^T y$ is the sum of squared potential differences

$$y^T A y = \|B^T y\|^2 = \sum_{\text{edges } i \to j} (y_j - y_i)^2$$

  this is also known as the *Dirichlet energy* function

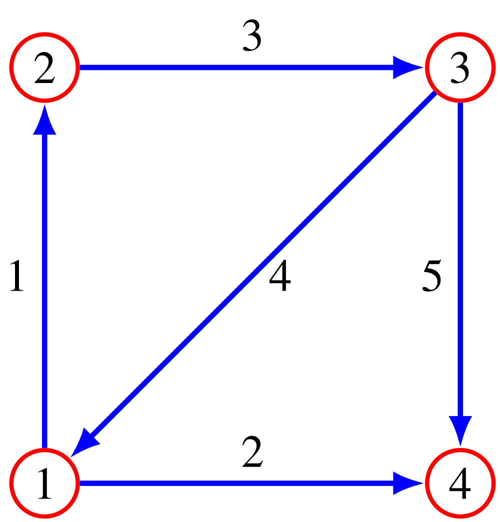
Example: for the graph on the previous page

$$y^T A y = (y_2 - y_1)^2 + (y_4 - y_1)^2 + (y_3 - y_2)^2 + (y_1 - y_3)^2 + (y_4 - y_3)^2$$

Weighted graph Laplacian

- we associate a nonnegative weight w_k with edge k
- the weighted graph Laplacian is the matrix $A = B\,\mathbf{diag}(w)B^T$
  ($\mathbf{diag}(w)$ is the diagonal matrix with vector w on its diagonal)

$$A_{ij} = \begin{cases} \sum\limits_{k\in\mathcal{N}_i} w_k & \text{if } i = j \quad \text{(where } \mathcal{N}_i \text{ are the edges incident to vertex } i\text{)} \\ -w_k & \text{if } i \neq j \text{ and edge } k \text{ is between vertices } i \text{ and } j \\ 0 & \text{otherwise} \end{cases}$$



$$A = \begin{bmatrix} w_1+w_2+w_4 & -w_1 & -w_4 & -w_2 \\ -w_1 & w_1+w_3 & -w_3 & 0 \\ -w_4 & -w_3 & w_3+w_4+w_5 & -w_5 \\ -w_2 & 0 & -w_5 & w_2+w_5 \end{bmatrix}$$

this is the conductance matrix of a resistive circuit (w_k is conductance in branch k)

Outline

- positive definite matrices
- examples
- **Cholesky factorization**
- complex positive definite matrices
- kernel methods

Cholesky factorization

every positive definite matrix $A \in \mathbf{R}^{n \times n}$ can be factored as

$$A = R^T R$$

where R is upper triangular with positive diagonal elements

- complexity of computing R is $(1/3)n^3$ flops
- R is called the *Cholesky factor* of A
- can be interpreted as “square root” of a positive definite matrix
- gives a practical method for testing positive definiteness

Cholesky factorization algorithm

$$\begin{bmatrix} A_{11} & A_{1,2:n} \\ A_{2:n,1} & A_{2:n,2:n} \end{bmatrix} = \begin{bmatrix} R_{11} & 0 \\ R_{1,2:n}^T & R_{2:n,2:n}^T \end{bmatrix} \begin{bmatrix} R_{11} & R_{1,2:n} \\ 0 & R_{2:n,2:n} \end{bmatrix}$$

$$= \begin{bmatrix} R_{11}^2 & R_{11}R_{1,2:n} \\ R_{11}R_{1,2:n}^T & R_{1,2:n}^T R_{1,2:n} + R_{2:n,2:n}^T R_{2:n,2:n} \end{bmatrix}$$

1. compute first row of R:

$$R_{11} = \sqrt{A_{11}}, \qquad R_{1,2:n} = \frac{1}{R_{11}} A_{1,2:n}$$

2. compute $2,2$ block $R_{2:n,2:n}$ from

$$A_{2:n,2:n} - R_{1,2:n}^T R_{1,2:n} = R_{2:n,2:n}^T R_{2:n,2:n}$$

   this is a Cholesky factorization of order $n-1$

Discussion

the algorithm works for positive definite A of size $n \times n$

- step 1: if A is positive definite then $A_{11} > 0$
- step 2: if A is positive definite, then

$$A_{2:n,2:n} - R_{1,2:n}^T R_{1,2:n} = A_{2:n,2:n} - \frac{1}{A_{11}} A_{2:n,1} A_{2:n,1}^T$$

  is positive definite (see page 12.5)

- hence the algorithm works for $n = m$ if it works for $n = m - 1$
- it obviously works for $n = 1$; therefore it works for all n

Example

$$\begin{bmatrix} 25 & 15 & -5 \\ 15 & 18 & 0 \\ -5 & 0 & 11 \end{bmatrix} = \begin{bmatrix} R_{11} & 0 & 0 \\ R_{12} & R_{22} & 0 \\ R_{13} & R_{23} & R_{33} \end{bmatrix} \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ 0 & R_{22} & R_{23} \\ 0 & 0 & R_{33} \end{bmatrix}$$

$$= \begin{bmatrix} 5 & 0 & 0 \\ 3 & 3 & 0 \\ -1 & 1 & 3 \end{bmatrix} \begin{bmatrix} 5 & 3 & -1 \\ 0 & 3 & 1 \\ 0 & 0 & 3 \end{bmatrix}$$

Example

$$\begin{bmatrix} 25 & 15 & -5 \\ 15 & 18 & 0 \\ -5 & 0 & 11 \end{bmatrix} = \begin{bmatrix} R_{11} & 0 & 0 \\ R_{12} & R_{22} & 0 \\ R_{13} & R_{23} & R_{33} \end{bmatrix} \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ 0 & R_{22} & R_{23} \\ 0 & 0 & R_{33} \end{bmatrix}$$

- first row of R

$$\begin{bmatrix} 25 & 15 & -5 \\ 15 & 18 & 0 \\ -5 & 0 & 11 \end{bmatrix} = \begin{bmatrix} 5 & 0 & 0 \\ 3 & R_{22} & 0 \\ -1 & R_{23} & R_{33} \end{bmatrix} \begin{bmatrix} 5 & 3 & -1 \\ 0 & R_{22} & R_{23} \\ 0 & 0 & R_{33} \end{bmatrix}$$

- second row of R

$$\begin{bmatrix} 18 & 0 \\ 0 & 11 \end{bmatrix} - \begin{bmatrix} 3 \\ -1 \end{bmatrix} \begin{bmatrix} 3 & -1 \end{bmatrix} = \begin{bmatrix} R_{22} & 0 \\ R_{23} & R_{33} \end{bmatrix} \begin{bmatrix} R_{22} & R_{23} \\ 0 & R_{33} \end{bmatrix}$$

$$\begin{bmatrix} 9 & 3 \\ 3 & 10 \end{bmatrix} = \begin{bmatrix} 3 & 0 \\ 1 & R_{33} \end{bmatrix} \begin{bmatrix} 3 & 1 \\ 0 & R_{33} \end{bmatrix}$$

- third column of R: $10 - 1 = R_{33}^2$, *i.e.*, $R_{33} = 3$

Solving equations with positive definite A

solve $Ax = b$ with A a positive definite $n \times n$ matrix

Algorithm

- factor A as $A = R^T R$
- solve $R^T R x = b$
  - solve $R^T y = b$ by forward substitution
  - solve $Rx = y$ by back substitution

Complexity: $(1/3)n^3 + 2n^2 \approx (1/3)n^3$ flops

- factorization: $(1/3)n^3$
- forward and backward substitution: $2n^2$

Cholesky factorization of Gram matrix

- suppose B is an $m \times n$ matrix with linearly independent columns
- the Gram matrix $A = B^T B$ is positive definite (page 4.20)

two methods for computing the Cholesky factor of A, given B

1. compute $A = B^T B$, then Cholesky factorization of A

$$A = R^T R$$

2. compute QR factorization $B = QR$; since

$$A = B^T B = R^T Q^T Q R = R^T R$$

   the matrix R is the Cholesky factor of A

Example

$$B = \begin{bmatrix} 3 & -6 \\ 4 & -8 \\ 0 & 1 \end{bmatrix}, \qquad A = B^T B = \begin{bmatrix} 25 & -50 \\ -50 & 101 \end{bmatrix}$$

1. Cholesky factorization:

$$A = \begin{bmatrix} 5 & 0 \\ -10 & 1 \end{bmatrix} \begin{bmatrix} 5 & -10 \\ 0 & 1 \end{bmatrix}$$

2. QR factorization

$$B = \begin{bmatrix} 3 & -6 \\ 4 & -8 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 3/5 & 0 \\ 4/5 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 5 & -10 \\ 0 & 1 \end{bmatrix}$$

Comparison of the two methods

Numerical stability: QR factorization method is more stable

- see the example on page 8.16
- QR method computes R without "squaring" B (*i.e.*, forming B^TB)
- this is important when the columns of B are "almost" linearly dependent

Complexity

- method 1: cost of symmetric product B^TB plus Cholesky factorization

$$mn^2 + (1/3)n^3 \text{ flops}$$

- method 2: $2mn^2$ flops for QR factorization
- method 1 is faster but only by a factor of at most two (if $m \gg n$)

Sparse positive definite matrices

Cholesky factorization of dense matrices

- $(1/3)n^3$ flops
- on a standard computer: a few seconds or less, for n up to several 1000

Cholesky factorization of sparse matrices

- if A is very sparse, R is often (but not always) sparse
- if R is sparse, the cost of the factorization is much less than $(1/3)n^3$
- exact cost depends on n, number of nonzero elements, sparsity pattern
- very large sets of equations can be solved by exploiting sparsity

Sparse Cholesky factorization

if A is sparse and positive definite, it is usually factored as

$$A = PR^T RP^T$$

P a permutation matrix; R upper triangular with positive diagonal elements

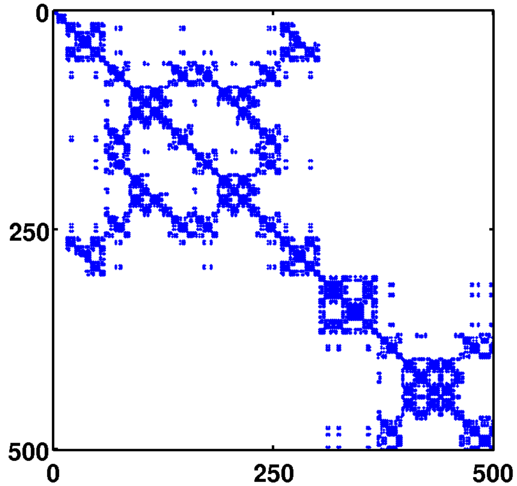
Interpretation: we permute the rows and columns of A and factor

$$P^T AP = R^T R$$

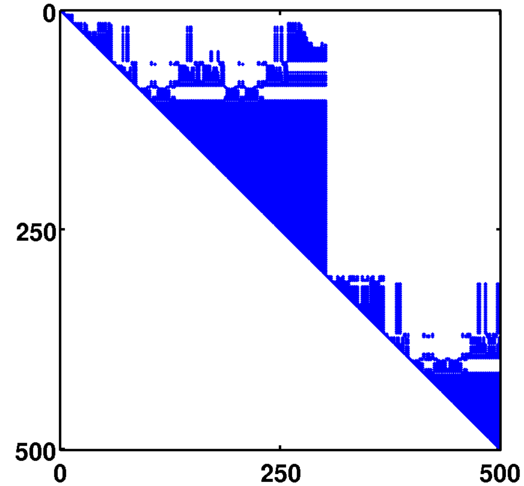
- choice of permutation greatly affects the sparsity R
- there exist several heuristic methods for choosing a good permutation

Example

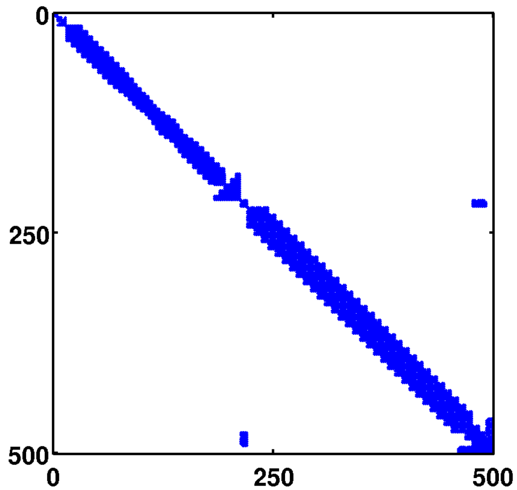
sparsity pattern of A



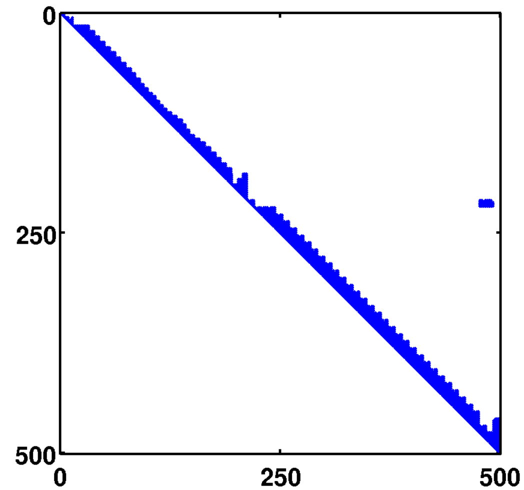
Cholesky factor of A



pattern of $P^T A P$



Cholesky factor of $P^T A P$



Solving sparse positive definite equations

solve $Ax = b$ with A a sparse positive definite matrix

Algorithm

1. compute sparse Cholesky factorization $A = PR^TRP^T$
2. permute right-hand side: $c := P^Tb$
3. solve $R^Ty = c$ by forward substitution
4. solve $Rz = y$ by back substitution
5. permute solution: $x := Pz$

Outline

- positive definite matrices
- examples
- Cholesky factorization
- **complex positive definite matrices**
- kernel methods

Quadratic form

suppose A is $n \times n$ and Hermitian ($A_{ij} = \bar{A}_{ji}$)

$$\begin{aligned} x^H A x &= \sum_{i=1}^{n} \sum_{j=1}^{n} A_{ij} \bar{x}_i x_j \\ &= \sum_{i=1}^{n} A_{ii} |x_i|^2 + \sum_{i>j} (A_{ij} \bar{x}_i x_j + \bar{A}_{ij} x_i \bar{x}_j) \\ &= \sum_{i=1}^{n} A_{ii} |x_i|^2 + 2 \operatorname{Re} \sum_{i>j} A_{ij} \bar{x}_i x_j \end{aligned}$$

note that $x^H A x$ is real for all $x \in \mathbf{C}^n$

Complex positive definite matrices

- a Hermitian $n \times n$ matrix A is positive semidefinite if

$$x^H A x \geq 0 \quad \text{for all } x \in \mathbf{C}^n$$

- a Hermitian $n \times n$ matrix A is positive definite if

$$x^H A x > 0 \quad \text{for all nonzero } x \in \mathbf{C}^n$$

Cholesky factorization

every positive definite matrix $A \in \mathbf{C}^{n \times n}$ can be factored as

$$A = R^H R$$

where R is upper triangular with positive real diagonal elements

Outline

- positive definite matrices
- examples
- Cholesky factorization
- complex positive definite matrices
- **kernel methods**

Regularized least squares model fitting

- we revisit the data fitting problem with linear-in-parameters model (page 9.9)

$$\begin{aligned}\hat{f}(x) &= \theta_1 f_1(x) + \theta_2 f_2(x) + \cdots + \theta_p f_p(x) \\ &= \theta^T F(x)\end{aligned}$$

- $F(x) = (f_1(x), \ldots, f_p(x))$ is a p-vector of basis functions $f_1(x), \ldots, f_p(x)$

Regularized least squares model fitting (page 10.7)

$$\text{minimize} \quad \sum_{k=1}^{N} \left(\theta^T F(x^{(k)}) - y^{(k)}\right)^2 + \lambda \sum_{j=1}^{p} \theta_j^2$$

- $(x^{(1)}, y^{(1)}), \ldots, (x^{(N)}, y^{(N)})$ are N examples
- to simplify notation, we add regularization for all coefficients $\theta_1, \ldots, \theta_p$
- next discussion can be modified to handle $f_1(x) = 1$, regularization $\sum_{j=2}^{p} \theta_j^2$

Regularized least squares problem in matrix notation

$$\text{minimize} \quad \|A\theta - b\|^2 + \lambda\|\theta\|^2$$

- A has size $N \times p$ (number of examples $\times$ number of basis functions)

$$A = \begin{bmatrix} F(x^{(1)})^T \\ F(x^{(2)})^T \\ \vdots \\ F(x^{(N)})^T \end{bmatrix} = \begin{bmatrix} f_1(x^{(1)}) & f_2(x^{(1)}) & \cdots & f_p(x^{(1)}) \\ f_1(x^{(2)}) & f_2(x^{(2)}) & \cdots & f_p(x^{(2)}) \\ \vdots & \vdots & & \vdots \\ f_1(x^{(N)}) & f_2(x^{(N)}) & \cdots & f_p(x^{(N)}) \end{bmatrix}$$

- b is the N-vector $b = (y^{(1)}, \ldots, y^{(N)})$
- we discuss methods for problems with $N \ll p$ (A is very wide)
- the equivalent "stacked" least squares problem (p.10.3) has size $(p + N) \times p$
- QR factorization method may be too expensive when $N \ll p$

Solution of regularized LS problem

from the normal equations:

$$\hat{\theta} = (A^T A + \lambda I)^{-1} A^T b = A^T (AA^T + \lambda I)^{-1} b$$

- second expression follows from the "push-through" identity

$$(A^T A + \lambda I)^{-1} A^T = A^T (AA^T + \lambda I)^{-1}$$

  this is easily proved, by writing it as $A^T (AA^T + \lambda I) = (A^T A + \lambda I) A^T$

- from the second expression for $\hat{\theta}$ and the definition of A,

$$\hat{f}(x) = \hat{\theta}^T F(x) = w^T A F(x) = \sum_{i=1}^{N} w_i F(x^{(i)})^T F(x)$$

  where $w = (AA^T + \lambda I)^{-1} b$

Algorithm

1. compute the $N \times N$ matrix $Q = AA^T$, which has elements

$$Q_{ij} = F(x^{(i)})^T F(x^{(j)}), \quad i, j = 1, \ldots, N$$

2. use a Cholesky factorization to solve the equation

$$(Q + \lambda I)w = b$$

Remarks

- $\hat{\theta} = A^T w$ is not needed; w is sufficient to evaluate the function $\hat{f}(x)$:

$$\hat{f}(x) = \sum_{i=1}^{N} w_i F(x^{(i)})^T F(x)$$

- complexity: $(1/3)N^3$ flops for factorization plus cost of computing Q

Example: multivariate polynomials

$\hat{f}(x)$ is a polynomial of degree d (or less) in n variables $x = (x_1, \ldots, x_n)$

- $\hat{f}(x)$ is a linear combination of all possible monomials

$$x_1^{k_1} x_2^{k_2} \cdots x_n^{k_n}$$

  where $k_1, \ldots, k_n$ are nonnegative integers with $k_1 + k_2 + \cdots + k_n \leq d$

- number of different monomials is

$$\left(\begin{array}{c} n+d \\ n \end{array} \right) = \frac{(n+d)!}{n!\; d!}$$

Example: for $n = 2$, $d = 3$ there are ten monomials

$$1, \quad x_1, \quad x_2, \quad x_1^2, \quad x_1 x_2, \quad x_2^2, \quad x_1^3, \quad x_1^2 x_2, \quad x_1 x_2^2, \quad x_2^3$$

Multinomial formula

$$(x_0 + x_1 + \cdots + x_n)^d = \sum_{k_0+\cdots+k_n=d} \frac{(d+1)!}{k_0!\, k_1! \,\cdots\, k_n!}\, x_0^{k_0} x_1^{k_1} \cdots x_n^{k_n}$$

sum is over all nonnegative integers $k_0, k_1, \ldots, k_n$ with sum d

- setting $x_0 = 1$ gives

$$(1 + x_1 + x_2 + \cdots + x_n)^d = \sum_{k_1+\cdots+k_n \le d} c_{k_1 k_2 \cdots k_n} x_1^{k_1} x_2^{k_2} \cdots x_n^{k_n}$$

- the sum includes all monomials of degree d or less with variables $x_1, \ldots, x_n$
- coefficient $c_{k_1 k_2 \cdots k_n}$ is defined as

$$c_{k_1 k_2 \cdots k_n} = \frac{(d+1)!}{k_0!\, k_1!\, k_2! \,\cdots\, k_n!} \quad \text{with} \quad k_0 = d - k_1 - \cdots - k_n$$

Vector of monomials

write polynomial of degree d or less, with variables $x \in \mathbf{R}^n$, as

$$\hat{f}(x) = \theta^T F(x)$$

- $F(x)$ is vector of basis functions

$$\sqrt{c_{k_1 \cdots k_n}}\, x_1^{k_1} x_2^{k_2} \cdots x_n^{k_n} \qquad \text{for all } k_1 + k_2 + \cdots + k_n \leq d$$

- length of $F(x)$ is $p = (n+d)!/(n!\,d!)$
- multinomial formula gives simple formula for inner products $F(u)^T F(v)$:

$$\begin{aligned} F(u)^T F(v) &= \sum_{k_1+\cdots+k_n \leq d} c_{k_1 k_2 \cdots k_n} (u_1^{k_1} \cdots u_n^{k_n})(v_1^{k_1} \cdots v_n^{k_n}) \\ &= (1 + u_1 v_1 + \cdots + u_n v_n)^d \end{aligned}$$

- only $2n+1$ flops needed for inner product of length $p = (n+d)!/(n!\,d!)$

Example

vector of monomials of degree $d = 3$ or less in $n = 2$ variables

$$
\begin{aligned}
F(u)^T F(v) &= \begin{bmatrix} 1 \\ \sqrt{3}u_1 \\ \sqrt{3}u_2 \\ \sqrt{3}u_1^2 \\ \sqrt{6}u_1u_2 \\ \sqrt{3}u_2^2 \\ u_1^3 \\ \sqrt{3}u_1^2u_2 \\ \sqrt{3}u_1u_2^2 \\ u_2^3 \end{bmatrix}^T \begin{bmatrix} 1 \\ \sqrt{3}v_1 \\ \sqrt{3}v_2 \\ \sqrt{3}v_1^2 \\ \sqrt{6}v_1v_2 \\ \sqrt{3}v_2^2 \\ v_1^3 \\ \sqrt{3}v_1^2v_2 \\ \sqrt{3}v_1v_2^2 \\ v_2^3 \end{bmatrix} \\
&= (1 + u_1v_1 + u_2v_2)^3
\end{aligned}
$$

Least squares fitting of multivariate polynomials

fit polynomial of n variables, degree $\leq d$, to points $(x^{(1)}, y^{(1)}), \ldots, (x^{(N)}, y^{(N)})$

Algorithm (see page 12.33)

1. compute the $N \times N$ matrix Q with elements

$$Q_{ij} = K(x^{(i)}, x^{(j)}) \quad \text{where } K(u, v) = (1 + u^T v)^d$$

2. use a Cholesky factorization to solve the equation $(Q + \lambda I)w = b$

- the fitted polynomial is

$$\hat{f}(x) = \sum_{i=1}^{N} w_i K(x^{(i)}, x) = \sum_{i=1}^{N} w_i (1 + (x^{(i)})^T x)^d$$

- complexity: nN^2 flops for computing Q, plus $(1/3)N^3$ for the factorization, *i.e.*,

$$nN^2 + (1/3)N^3 \text{ flops}$$

Kernel methods

Kernel function: a generalized inner product $K(u,v)$

- $K(u,v)$ is inner product of vectors of basis functions $F(u)$ and $F(v)$
- $F(u)$ may be infinite-dimensional
- kernel methods work with $K(u,v)$ directly, do not require $F(u)$

Examples

- the polynomial kernel function $K(u,v) = (1 + u^T v)^d$
- the *Gaussian radial basis function* kernel

$$K(u,v) = \exp\left(-\frac{\|u-v\|^2}{2\sigma^2}\right)$$

- kernels exist for computing with graphs, texts, strings of symbols, ...

Example: handwritten digit classification

we apply the method of page 12.38 to least squares classification

- training set is 10000 images from MNIST data set ($\approx$ 1000 examples per digit)
- vector x is vector of pixel intensities (size $n = 28^2 = 784$)
- we use the polynomial kernel with degree $d = 3$:

$$K(u, v) = (1 + u^T v)^3$$

  hence $F(z)$ has length $p = (n + d)!/(n!\, d!) = 80931145$

- we calculate ten Boolean classifiers

$$\hat{f}_k(x) = \text{sign}(\tilde{f}_k(x)), \quad k = 1, \ldots 10$$

  $\hat{f}_k(x)$ distinguishes digit $k - 1$ (outcome $+1$) form other digits (outcome -1)

- the Boolean classifiers are combined in the multi-class classifier

$$\hat{f}(x) = \underset{k=1,\ldots,10}{\text{argmax}}\; \tilde{f}_k(x)$$

Least squares Boolean classifier

Algorithm: compute Boolean classifier for digit $k - 1$ versus the rest

1. compute $N \times N$ matrix Q with elements

$$Q_{ij} = (1 + (x^{(i)})^T x^{(j)})^d, \quad i, j = 1, \ldots, N$$

2. define N-vector $b = (y^{(1)}, \ldots, y^{(N)})$ with elements

$$y^{(i)} = \begin{cases} +1 & x^{(i)} \text{ is an example of digit } k - 1 \\ -1 & \text{otherwise} \end{cases}$$

3. solve the equation $(Q + \lambda I)w = b$

the solution w gives the Boolean classifier for digit $k - 1$ versus rest

$$\tilde{f}_k(x) = \sum_{i=1}^{N} w_i (1 + (x^{(i)})^T x)^d$$

Complexity

- the matrix Q is the same for each of the ten Boolean classifiers
- hence, only the right-hand side of the equation

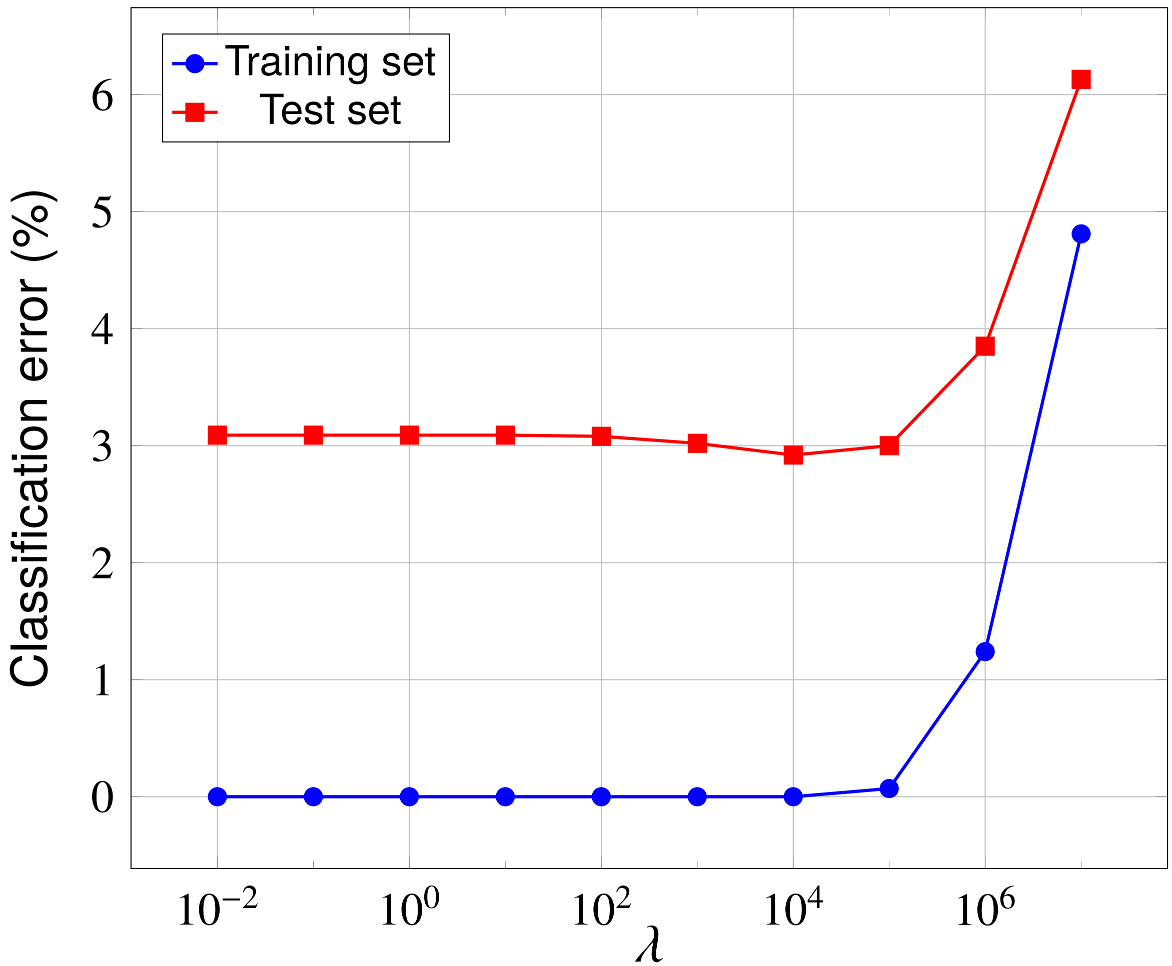
$$(Q + \lambda I)w = y^{\mathrm{d}}$$

  is different for each Boolean classifier

Complexity

- constructing Q requires $N^2/2$ inner products of length n: nN^2 flops
- Cholesky factorization of $Q + \lambda I$: $(1/3)N^3$ flops
- solve the equation $(Q + \lambda I)w = y^{\mathrm{d}}$ for the 10 right-hand sides: $20N^2$ flops
- total is $(1/3)N^3 + nN^2$

Classification error



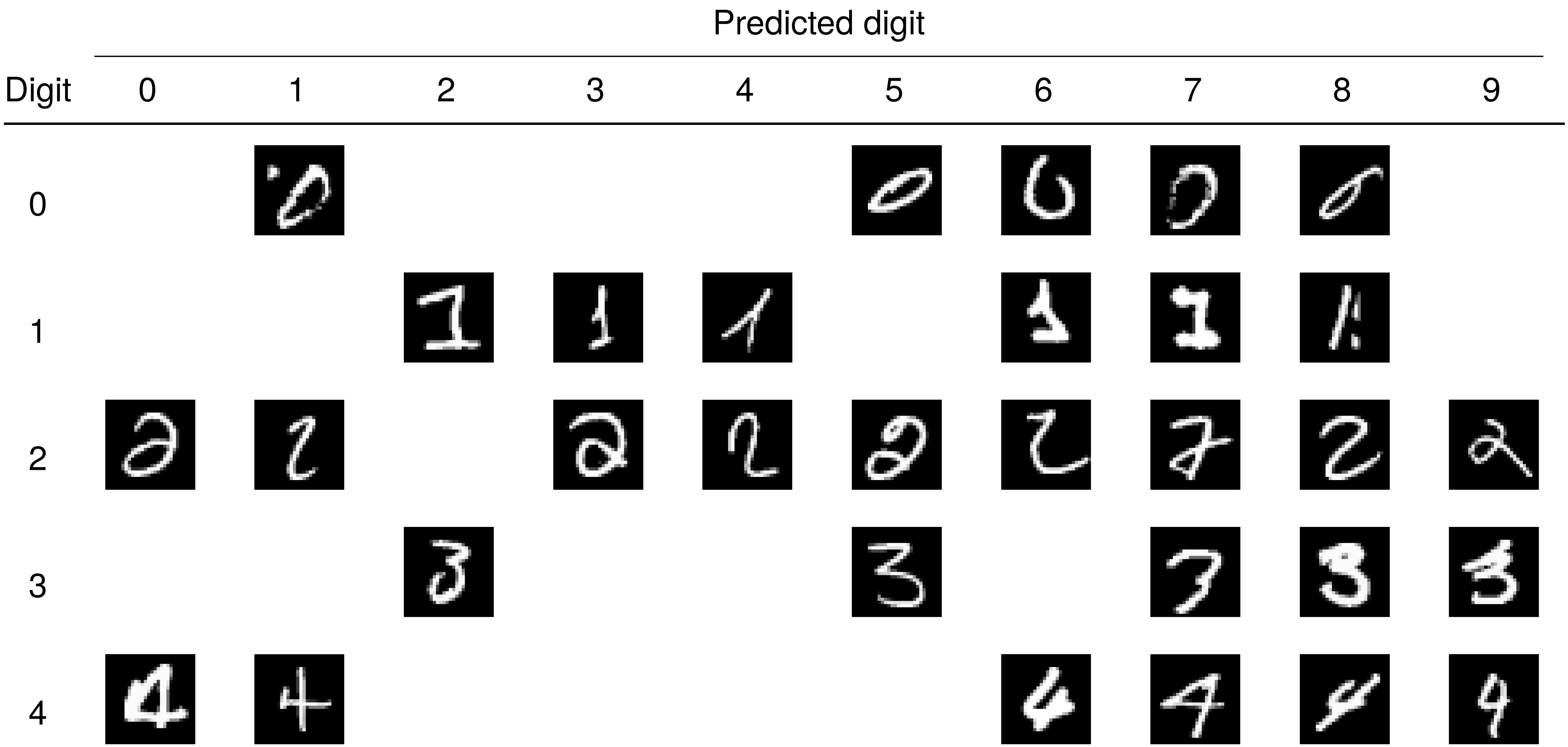
percentage of misclassified digits versus λ

Confusion matrix

	Predicted digit										
Digit	0	1	2	3	4	5	6	7	8	9	Total
0	965	1	0	0	0	1	8	2	3	0	980
1	0	1127	2	1	1	0	2	1	1	0	1135
2	6	2	988	4	1	1	5	16	8	1	1032
3	0	0	7	973	0	12	0	8	6	4	1010
4	1	3	0	0	957	0	3	1	3	14	982
5	3	0	0	5	0	874	5	2	2	1	892
6	9	4	0	0	5	2	937	0	1	0	958
7	0	13	13	1	5	0	0	987	2	7	1028
8	3	1	3	11	4	4	3	5	934	6	974
9	3	4	2	7	13	3	1	6	4	966	1009
All	990	1155	1015	1002	986	897	964	1028	964	999	10000

- multiclass classifier ($\lambda = 10^4$) on 10000 test examples
- 292 digits are misclassified (2.9% error)

Examples of misclassified digits



Examples of misclassified digits