Finance Data Science
Lecture 14: Review

Laurent El Ghaoui

MFE 230P, Summer 2017
MFE Program
Haas School of Business
UC Berkeley

7/21/2017

Outline

Outline

Vectors, scalar products & norms

A vector $x \in \mathbf{R}^n$ is an array of n numbers represented as a column:

$$x = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}.$$

The *transpose* (denoted x^T) is the corresponding row.

Scalar product: if x, y are two n-vectors,

$$x^T y := \sum_{i=1}^{n} x_i y_i.$$

Norms are ways to measure the "size" of a vector:

- The Euclidean norm (norm for short) $\|x\|_2 = \sqrt{x_1^2 + \ldots + x_n^2}$ corresponds to the usual distance in 3D.
- The l_1-norm $\|x\|_1 = |x_1| + \ldots + |x_n|$ is the "Manhattan" distance.
- The l_∞-norm $\|x\|_\infty = \max_{1 \le i \le n} |x_i|$ is useful to impose bounds on the elements of x.

Matrices

A $n \times m$ matrix A is a rectangular array of elements A_{ij}, $1 \leq i \leq n$, $1 \leq j \leq m$.

Matrices can be understood as *operators* via the matrix-vector product: $x \to Ax$ where for a vector $x \in \mathbf{R}^m$,

$$(Ax)_i = \sum_{j=1}^{m} A_{ij} x_j.$$

Special matrices:

- Diagonal matrices: $A_{ij} = 0$ for every $i \neq j$.
- Symmetric matrices: $A_{ij} = A_{ji}$ for every i, j.
- Orthogonal matrices: $A^T A = I$, that is, all the columns are unit-norm and orthogonal to each other.

Eigenvalue Decomposition of Symmetric Matrices

Theorem (EVD of symmetric matrices)

We can decompose any $p \times p$ symmetric matrix S as

$$S = \sum_{i=1}^{p} \lambda_i u_i u_i^T = U \Lambda U^T, \quad \Lambda = \mathbf{diag}(\lambda_1, \ldots, \lambda_p) \in \mathbf{R}^{p \times p}$$

where $\lambda_1, \ldots, \lambda_p$ are the eigenvalues, and $U = [u_1, \ldots, u_p]$ is a square, orthogonal matrix ($U^T U = I_p$).
The columns of U contains the eigenvectors of S, that is:

$$S u_i = \lambda_i u_i, \quad i = 1, \ldots, p.$$

PSD matrices

A symmetric matrix S is said to be positive semi-definite (PSD) if $x^T Sx \geq 0$ for every x; this is equivalent to: all the eigenvalues are non-negative, $\lambda_i \geq 0$, $i = 1, \ldots, p$.

Covariance matrices are prominent examples of PSD matrices. Indeed, if x is a portfolio vector, then $x^T Sx$ is the variance of the portfolio (hence, it is non-negative).

Singular Value Decomposition (SVD) of General Matrices

Theorem (SVD of general matrices)

We can decompose any non-zero $p \times m$ matrix A as

$$A = \sum_{i=1}^{r} \sigma_i u_i v_i^T = U \Sigma V^T, \;\; \Sigma = \mathbf{diag}(\sigma_1, \ldots, \sigma_r, 0, \ldots, 0) \in \mathbf{R}^{p \times m}$$

where $\sigma_1 \geq \ldots \geq \sigma_r > 0$ are the singular values, and

$$U = [u_1, \ldots, u_m], \;\; V = [v_1, \ldots, v_p]$$

are square, orthogonal matrices ($U^T U = I_p$, $V^T V = I_m$). The number $r \leq \min(p, m)$ (the number of non-zero singular values) is called the rank of A. The first r columns of U, V contains the left- and right singular vectors of A, respectively, that is:

$$A v_i = \sigma_i u_i, \;\; A^T u_i = \sigma_i v_i, \;\; i = 1, \ldots, r.$$

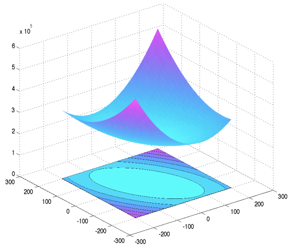
Convex function

Definition

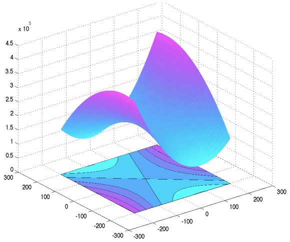
A function $f : \mathbf{R}^n \to \mathbf{R}$ is convex if it satisfies the condition

$$\forall\, x, y \in \mathbf{R}^n, \lambda \in [0,1] \;:\; f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda) f(y)$$

Geometrically, the graph of the function is “bowl-shaped”. A function f is concave if $-f$ is convex.



Convex function.



Non-convex function.

Composition rules

- The point-wise maximum of convex functions is convex.
- The sum of convex functions is convex.
- The composition of a convex function with an affine function is convex.

Examples of convex functions:

- $f(x) = \max(x_1 + 2x_2^2, 3 + 2x_2 - \min(-x_1^2 + 3x_2, 0))$.
- $g(x) = f(Ax)$, where A is a matrix and f is convex.

Outline

Optimization problem

A standard form

An optimization problem is a problem of the form

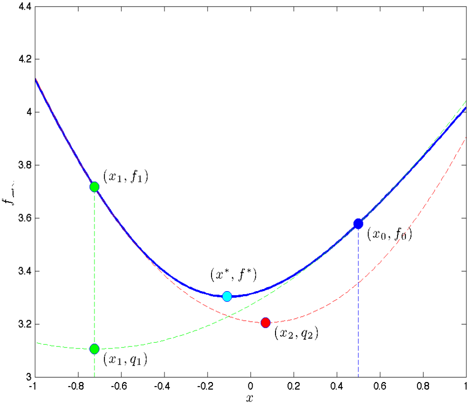
$$p^* := \min_x \; f_0(x) \text{ subject to } f_i(x) \leq 0, \quad i = 1, \ldots, m,$$

where

- $x \in \mathbf{R}^n$ is the *decision variable* ;
- $f_0 : \mathbf{R}^n \to \mathbf{R}$ is the *objective* (or, *cost*) function;
- $f_i : \mathbf{R}^n \to \mathbf{R}$, $i = 1, \ldots, m$ represent the *constraints* ;
- p^* is the *optimal value* .

Convexity and local minima

When trying to minimize convex functions, specialized algorithms will always converge to a global minimum, irrespective of the starting point, provided some (weak) assumptions on the function hold.



The Newton algorithm.

Convex optimization

Definition

The problem in standard form

$$p^* := \min_x \; f_0(x) \text{ subject to } f_i(x) \leq 0, \quad i = 1, \ldots, m,$$

is convex if the functions $f_0, \ldots, f_m$ are all convex.

Examples:

- Linear programming ($f_0, \ldots, f_m$ affine).
- Quadratic programming (f_0 convex quadratic, $f_1, \ldots, f_m$ affine).
- Second-order cone programming (f_0 linear, f_i's of the form $\|A_i x + b_i\|_2 + c_i^T x + d_i$, for appropriate data A_i, b_i, c_i, d_i).

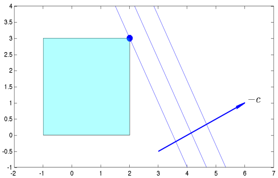
Problem classes

Linear programming

Linear programming (LP) is a convex problem in standard form, involving linear or affine functions only:

$$\min_x \; c^T x \;:\; Cx \leq d, \;\; Ax = b.$$

(Here, $u \leq v$ means that the two n-vectors u, v are such that $u_i \leq v_i$, $i = 1, \ldots, n$.)



The problem

$$\begin{array}{ll} \min_x & 3x_1 + 1.5x_2 \\ \text{s.t.} & -1 \leq x_1 \leq 2, \\ & 0 \leq x_2 \leq 3 \end{array}$$

is an LP, since the objective and constraint functions are all affine.

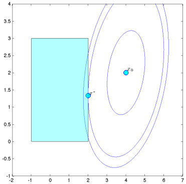
Problem classes

Quadratic programming

Quadratic programming (QP) involves the minimization of a quadratic convex function under linear or affine constraints.

$$\min_x \; c^T x + x^T Q x \;:\; Cx \le d, \;\; Ax = b.$$

Here, Q must be PSD.



The problem

$$\begin{array}{ll} \min\limits_x & x_1^2 - x_1 x_2 + 2x_2^2 \\ & -3x_1 - 1.5x_2 \\ \text{s.t.} & -1 \le x_1 \le 2, \\ & 0 \le x_2 \le 3 \end{array}$$

is a QP, since objective is quadratic convex, and the the constraint functions are all affine.

Example of a quadratic program

Mean-variance trade-off problem

- *Data:* expected returns $r \in \mathbf{R}^n$, covariance matrix $C \in \mathbf{R}^{n \times n}$.
- *Problem:* minimize the risk of a portfolio $x \in \mathbf{R}^n$, subject to constraints:
  - No shorting.
  - Sum of all positions equal to 1 (maximum amount to invest).
  - Expected return above a target t.

QP formulation:

$$\min_x \; x^T C x \;:\; \begin{array}{l} \sum_i x_i = 1, \\ x \geq 0, \;\; r^T x \geq t. \end{array}$$

This is indeed a QP, since C is PSD (as is any covariance matrix).

Problem classes

Second-order cone programming

Second-order cone programming (SOCP) generalizes LP and QP via the inclusion of Euclidean norms in the constraint functions.

$$\min_x \; c_0^T x \;:\; \|A_i x + b_i\|_2 \le c_i^T x + d_i, \quad i = 1, \ldots, m.$$

Includes LP and QP as special cases. (The latter, thanks to the rotated second-order cone constraints $uv \ge \|z\|_2^2$.)

Application: Risk parity; robust and chance programming.

Problem classes

Semi-definite programming programming

Semi-definite programming (SDP) involves the minimization of a linear function over the constraint that a symmetric matrix affine in the decision variables be positive-semidefinite:

$$\min_x \; c_0^T x \;:\; F_0 + \sum_{i=1}^n x_i F_i \succeq 0.$$

(Here, $F_0, \ldots, F_n$ are given symmetric matrices.)

Application: worst-case risk of a portfolio with partially known covariance matrix:

$$\max_C \; w^T C w \;:\; C \succeq 0, \;\; \underline{C} \le C \le \overline{C},$$

where $\overline{C}$, $\underline{C}$ contains lower and upper bounds of confidence on the elements of the partially known covariance matrix.

Outline

Variance maximization

If C is the *empirical covariance matrix* of data points $x_i \in \mathbf{R}^p$:

$$C = \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{x})(x_i - \hat{x})^T, \quad \hat{x} := \frac{1}{n}\sum_{i=1}^{n} x_i,$$

then the maximum variance direction can be found by solving

$$\max_x \; x^T C x \; : \; \|x\|_2 = 1.$$

- Can be solved via eigenvalue decomposition of C, or singular value decomposition of centered data matrix $[x_1 - \hat{x}, \ldots, x_n - \hat{x}]$.
- Optimal x is any eigenvector corresponding to the largest eigenvalue.

Principal component analysis

Principal component analysis (PCA) is a process where we solve variance maximization problems repeatedly:

- Project data points on the subspace orthogonal to the max-variance direction we found.
- Find a direction of maximal variance for projected data.
- Iterate until variance explained is close enough to total variance.

The whole process can be obtained in *one* singular value decomposition of the (centered) data matrix.

Low-rank approximation of a matrix

For a given $p \times m$ matrix A, and integer $k \leq m, p$, the *k-rank approximation* problem is

$$A^{(k)} := \arg\min_X \ \|X - A\|_F \ : \ \textbf{Rank}(X) \leq k,$$

where $\|\cdot\|_F$ is the Frobenius norm (Euclidean norm of the vector formed with all the entries of the matrix).

The solution is

$$A^{(k)} = \sum_{i=1}^{k} \sigma_i u_i v_i^T,$$

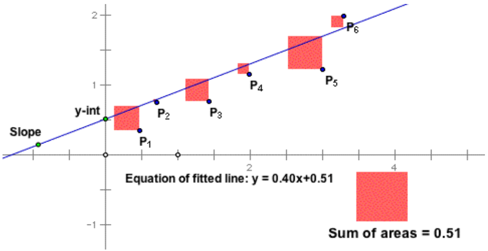
where

$$A = U\Sigma V^T = \sum_{i=1}^{r} \sigma_i u_i v_i^T$$

is an SVD of the matrix A.

PCA (on the covariance matrix) is equivalent to finding a low-rank approximation to the (centered) data matrix.

Linear least-squares

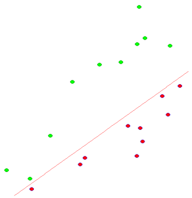


$$\min_{w} \|X^T w - y\|_2$$

where

- $X = [x_1, \ldots, x_m]$ is a $n \times m$ matrix of data points ($x_i \in \mathbf{R}^n$);
- y is a response vector;
- $\| \cdot \|_2$ is the l_2 (*i.e.*, Euclidean) norm.
- Many variants (with *e.g.*, constraints) exist (*e.g.*, penalized version).
- Perhaps the most popular / useful optimization problem.

Linear classification



$$\min_{w,b} \sum_{i=1}^{m} \max(0, 1 - y_i(w^T x_i + b))$$

where

- $X = [x_1, \ldots, x_m]$ is a $n \times m$ matrix of data points ($x_i \in \mathbf{R}^n$);
- $y \in \{-1, 1\}$ is a *binary* response vector;
- Many variants (with *e.g.*, constraints) exist (more on this later).
- Very useful for classifying data (*e.g.*, text documents).

Outline

Basic mean-variance trade-off

Given expected returns $\hat{r}$ and covariance matrix C:

$$\max_x \; \hat{r}^T x - \lambda x^T C x \; : \; x \in \mathcal{X}$$

with $\lambda > 0$ a "risk aversion" parameter, and $\mathcal{X}$ models constraints on our portfolio, *e.g.*:

$$\mathcal{X} = \left\{ x \; : \; x \geq 0, \;\; \mathbf{1}^T x = 1 \right\}.$$

If $\mathcal{X}$ is a polytope (it is defined by affine inequalities and equalities), then the above is a QP.

Variants

- Transaction costs.
- Market impact.
- Diversification.
- Chance constraints, value-at-risk.
- Robustness.

If C has a "diagonal plus low-rank" structure then this can be exploited to speed up the algorithm.

Simple ALM

The simple asset liability management (cash-flow matching) problem of lecture 13:

$$
\begin{array}{ll}
\max\limits_{x,y,z} & z_6 \\
\text{s.t.} & x_1 + y_1 - z_1 = 150, \\
& x_2 + y_2 - 1.01x_1 + 1.003z_1 - z_2 = 100, \\
& x_3 + y_3 - 1.01x_2 + 1.003z_2 - z_3 = -200, \\
& x_4 - 1.02y_1 - 1.01x_3 + 1.003z_3 - z_4 = 200, \\
& x_5 - 1.02y_2 - 1.01x_4 + 1.003z_4 - z_5 = -50, \\
& -1.02y_3 - 1.01x_5 + 1.003z_5 - z_6 = -300, \\
& 0 \leq x \leq 100, \quad y \geq 0, \quad z \geq 0.
\end{array}
$$

The right-hand side contains the liabilities that we must meet.

Affine recourse

In practice the liability vector (RHS of cash-flow balance constraints) is unknown in advance.

- Robust solution: make a decision under worst-case assumption.
- Stochastic programming solution: use scenarios, expected values.
- Affine recourse solution: go robust, but allow for recourse in decisions as information becomes available.

Outline

Going further

- Kernel methods in machine learning.
- Stochastic programming and robust programming.
- Very large-scale problems.
- Integer programming.