# Finance Data Science
# Lecture 11: Portfolio Optimization, I

Laurent El Ghaoui

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

7/14/2017

# Outline

# Outline

# Portfolio Basics

- $r \in \mathbf{R}^n$: random (log-) return vector.
- $x \in \mathbf{R}^n$: describes portfolio (dollar) position.
- $\hat{r} \in \mathbf{R}^n$: expected (or, nominal) return.
- $C = C^T \succeq 0$: $n \times n$ covariance matrix.

- $\hat{r}^T x$: portfolio return.
- $x^T C x$: portfolio variance (a measure of risk).

*Basic portfolio optimization problem:* trade-off expected return against risk, subject to various constraints on position.

# An easy problem

Basic portfolio optimization problem:

$$\min_x \; x^T C x \;:\; \hat{r}^T x \geq t, \;\; x \geq 0, \;\; \sum_{i=1}^{n} x_i = 1.$$

- No shorting allowed.
- Set a minimum target $t$ for the expected return.
- Includes a very basic budget constraint.

Above is a "quadratic program" that can be solved via convex optimization algorithms, even in a large-scale setting (thousands of assets and constraints, $C$ dense).

# Going further?

- Can we add constraints? *e.g.*, no more than 50% of total budget allocated to any subset of 10 assets?
- Can we handle transaction costs? Market impact?
- Can we extend this to a multi-period setting?
- The model penalizes downside and upside risk equally; are there alternate, better measures of risk?
- The model hinges on estimating the returns, which is difficult. Can we handle noise in the estimates?
- Can we do away entirely with estimating returns?
- Can we efficiently handle structure of $C$ (such as, factor models, or sparse inverse)?

Optimization models can "cover" almost any problem but . . . Not all optimization problems are solvable (*e.g.*, fixed transaction costs).

However, *convex* problems are efficiently solvable. Let us learn what convex optimization is . . .

# Outline

# Convex functions
## Definition

A function $f : \mathbf{R}^n \to \mathbf{R}$ is *convex* if

$$\forall\, x_1, x_2, \quad \forall \lambda \in [0,1] \;:\; f(\lambda x_1 + (1-\lambda x_2)) \leq \lambda f(x_1) + (1-\lambda) f(x_2).$$

We say $f$ is *concave* if $-f$ is convex.

*Examples:*

- $f(x) = c^T x + d$ (affine function; both convex and concave).
- norms: $f(x) = \|x\|_2$, $f(x) = \|x\|_1$. (In fact, any norm is convex.)
- $f(x) = \max(0, 1-x)$ for scalar $x$.
- $f(x) = \log(1+e^x)$ for scalar $x$.

# Proving convexity

Proving convexity can be difficult.

In practice, we use some *combination* rules, *e.g.*:

- A *quadratic* function is convex if and only if . . . (see next)
- The sum or maximum of (any number of) convex functions is convex.
- The composition of a convex function with a linear or affine one is convex.
- The composition of a convex function with an increasing convex function is convex.

# Proving convexity

## Positive semidefinite matrices

*Flashback from linear algebra:*

- A square matrix $A$ has $n$ (possibly non-distinct) eigenvalues, which are (in general complex) numbers that solve $\det(\lambda I - A) = 0$.
- Symmetric matrices have real eigenvalues only. Those can be obtained by the eigenvalue decomposition (EVD) of the matrix.
- A symmetric matrix $Q$ is said to be *positive semi-definite* (PSD) if

$$\forall x \ : \ x^T Q x \geq 0.$$

  We write $Q \succeq 0$.
- $Q$ is PSD if and only if every one of its eigenvalues is non-negative.
- Any PSD matrix $Q$ can be written $Q = R^T R$ for some matrix $R$.

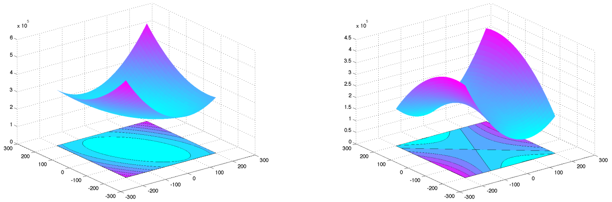
# Convex functions

## Convex quadratic functions

A quadratic function $q : \mathbf{R}^n \to \mathbf{R}$ can be represented as

$$q(x) = x^T Qx + b^T x + c,$$

for appropriate symmetric matrix $Q$, vector $b \in \mathbf{R}^n$, and scalar $c$.




A convex quadratic function (left) and a non-convex one (right).

Fact: $q$ convex $\iff$ $Q$ is PSD ($Q \succeq 0$).

# Convex functions

## Convex quadratic functions: example

Previous fact means that it is easy (via EVD) to recognize convexity of a quadratic function.

*Example:*

$$q(x) = x_1^2 - x_1 x_2 + 2x_2^2 - 3x_1 - 1.5x_2 = x^T Q x + c^T x,$$

with $c = (-3, -1.5)$, and

$$Q := \begin{pmatrix} 1 & -1/2 \\ -1/2 & 2 \end{pmatrix}.$$

We check that $Q$ is PSD via EVD, or noting that $Q$ is the sum of two PSD matrices:

$$Q = \frac{1}{2} \begin{pmatrix} 1 \\ -1 \end{pmatrix} \begin{pmatrix} 1 \\ -1 \end{pmatrix}^T + \begin{pmatrix} 1/2 & 0 \\ 0 & 3/2 \end{pmatrix}.$$

Indeed, for every $z = (z_1, z_2)$:

$$z^T Q z = \frac{1}{2}(z_1 - z_2)^2 + \frac{1}{2} z_1^2 + \frac{3}{2} z_2^2 \geq 0.$$

# Sum and maximum rules

- The sum of any number of convex (resp. concave) functions is convex (resp. concave).
- The maximum of any number of convex functions is convex.
- The minimum of any number of concave functions is concave.

*Examples:*

- The sum of the $k$ largest components of a $n$-vector is convex (here, $k \leq n$), since

$$s_k(x) = \max_u \; u^T x \;:\; u_i \in \{0, 1\}, \;\; i = 1, \ldots, n, \;\; \sum_{i=1}^{n} u_i = k.$$

  (*e.g.*, with $n = 3$, $k = 2$, $s_2(x) = \max(x_1 + x_2, x_2 + x_3, x_3 + x_1)$; each piece in the max is linear, hence convex.)

- The *smallest* eigenvalue of a symmetric matrix $X$ is *concave* in $X$, since

$$\lambda_{\min}(X) = \min_{z \,:\, z^T z = 1} \; z^T X z$$

  Likewise the *largest* eigenvalue function is *convex* .

# Affine composition rule

If $f : \mathbf{R}^n \to \mathbf{R}$ is convex, and $A \in \mathbf{R}^{m \times n}$, $b \in \mathbf{R}^m$ are given, then $g : \mathbf{R}^m \to \mathbf{R}$ with values

$$g(x) = f(Ax + b)$$

is convex.

*Example:* ("hinge loss function" from SVM learning)

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

where

$$Z := X\, \mathbf{diag}(y), \quad f(z) = \sum_{i=1}^{m} \max(0, z_i).$$

(Here, we invoke the sum and affine composition rules.)

# Composition rule

If $f : \mathbf{R}^n \to \mathbf{R}$ is convex and $g : \mathbf{R} \to \mathbf{R}$ is convex and increasing then $h = g \circ f$ (with values $h(x) = g(f(x))$) is convex.

*Examples:*

- $h(x) = f(x)^2$ is convex if $f(x) \geq 0$ for every $x$.
- $h(x) = \log(1 + e^{f(x)})$ if convex if $f$ is.

# Convex problems
## Definition

The problem in standard form

$$p^* := \min_x \; f_0(x) \text{ subject to } \begin{array}{l} f_i(x) \leq 0, \quad i = 1, \dots, m, \\ Ax = b, \end{array}$$

is convex if the functions $f_0, \dots, f_m$ are all convex. Here $A \in \mathbf{R}^{p \times n}$, $b \in \mathbf{R}^p$ are given, and $x \in \mathbf{R}^n$ is the decision variable.

- Note that only affine equality constraints are allowed.
- Can't replace $\leq$ signs by $\geq$ signs, without destroying convexity.
- The set of vectors $x$ that are *feasible*, that is, satisfy the constraints, is called the feasible set.
- The feasible is a convex set, in the set that the line segment joining any two feasible points is feasible.

# Definition

## Maximization problems

The problem in standard form

$$\max_x \; f_0(x) \text{ subject to } \quad \begin{array}{l} f_i(x) \leq 0, \quad i = 1, \ldots, m, \\ Ax = b, \end{array}$$

is convex if

- The function $f_0$ is concave.
- The functions $f_1, \ldots, f_m$ are all convex.

*Note:* the feasible set must again be convex.

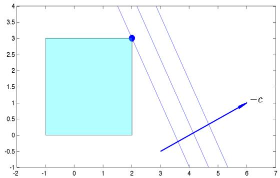
# Problem classes

## 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

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

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

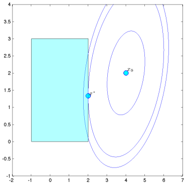
# Problem classes

## Quadratic programming

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

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

Here, $Q$ must be PSD.



The problem

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

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 case.

*Application:* Chance-constrained linear programming

$$\min_x \; c^T x \; : \; \mathbf{Prob}\{a_i^T x \le b\} \ge 0.99, \quad i = 1, \ldots, m,$$

where each $a_i$ is a Gaussian random variable with mean $\hat{a}_i$ and covariance matrix $\Sigma_i$, $i = 1, \ldots, m$.

# 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:

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

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

# CVX

CVX [3] is a matlab toolbox for convex optimization.

- Handles LP, QP, SOCP, and much more.
- Well-suited to moderate-size problems (1000's of constraints, variables).
- Great to quickly prototype a solution approach.

Recall QP formulation of mean-variance trade-off problem:

$$\min_x \ x^T C x \ : \ \sum_i x_i = 1, \quad x \geq 0, \quad r^T x \geq t.$$

*CVX code:* (assume $p$, $C$, $r$, $n$ exist in matlab's workspace)

```
cvx_begin
variable x(n,1);
minimize( x'*C*x )
subject to
    sum(x) == 1;
    x >= 0;
    r'*x >= t;
cvx_end
```

Note the *double equality sign* == that is used to encode affine equality constraints.

# Outline

# Mean-variance trade-off

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

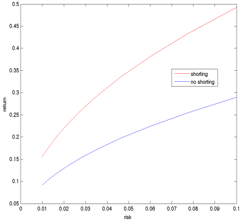
with $\lambda > 0$ a ”risk aversion” parameter.

- Curve $(x(\lambda)^T C x(\lambda), r^T x(\lambda))$ shows “efficient frontier”.
- Same curve obtained if we sweep over $t$ in model

$$\min_x \; x^T C x \; : \; x \in \mathcal{X}, \;\; \hat{r}^T x \geq t.$$

In the above, $\mathcal{X}$ models constraints on our portfolio, *e.g.*:

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




Efficient frontier for a portfolio problem with and without no-shorting constraints.

By varying $t$, we plot the efficient frontier $(r^T x, \sqrt{x^T C x})$.

# Linear transaction costs

## Cost per asset

Denote by $x_i^0$ the current position in asset $i$, and by $x_i$ a new position we wish to take. Let $\delta := x_i - x_i^0$ denote the difference in dollar amount invested.

We assume “linear” transaction costs (no fixed costs)

$$f(\delta) = \begin{cases} a_{\text{long}}\delta & \text{if } \delta \geq 0 \\ -a_{\text{short}}\delta & \text{if } \delta \leq 0 \end{cases},$$

where $a_{\text{long}}, a_{\text{short}} > 0$ are transaction costs parameters (usually $a_{\text{short}} > a_{\text{long}}$, as shorting is more expensive). We can write

$$f(\delta) = \max(a_{\text{long}}\delta, -a_{\text{short}}\delta).$$

# Linear transaction costs

## Total cost

For a *vector* position $x \in \mathbf{R}^n$, with $x^0 \in \mathbf{R}^n$ the initial position, the total transaction cost is

$$TC(x) = \sum_{i=1}^{n} \max(a_{\text{long}}(x_i - x_i^0), -a_{\text{short}}(x_i - x_i^0)).$$

Portfolio problem becomes:

$$\min_x \; x^T C x + c \cdot TC(x) \;:\; x \in \mathcal{X},$$

with $c > 0$ a parameter that allows to trade-off risk, return, and transaction costs.

Not yet a QP ... But CVX will accept this as written!

# Linear transaction costs

Our problem Reduces to $l_1$-norm of $x - x^0$ when $a_{\text{long}} = a_{\text{short}}$!

$$\min_x \; x^T C x + c\|x - x_0\|_1 \;:\; x \in \mathcal{X}.$$

- $c$ is transaction cost parameter (has to be estimated).
- $x_0$ is initial position.
- Can model transaction costs with different slopes for buy or sell.
- However the case of "fixed plus linear" transaction costs cannot be directly solved via convex optimization.

# CVX syntax

Assume that $\mathcal{X}$ corresponds to linear equalities and inequalities, *i.e.*:

$$\mathcal{X} = \left\{ x \in \mathbf{R}^n \;:\; Ax \leq b, \quad Cx = d \right\},$$

where $A$, $C$ are given matrices and $b$, $d$ vectors.

If $r, S, A, b, C, d, \lambda, c, x^0$ and , $a_{\text{long}}, a_{\text{short}}$ exist in the workspace:

```
cvx_begin
    variable x(n,1);
    maximize( r'*x - lambda*x'*S*x - ...
              c*sum(max(along*(x-x0),-ashort*(x-x0))));
    subject to
       Ax <= b;
 Cx == d;
cvx_end
```

# QP model with transaction costs

Unlike CVX, most commercial software for QP require the user to transform the problem in standard form.

Represent the transaction cost as the solution of an LP:

$$TC(x) = \min_u \sum_{i=1}^n u_i \ : \ u \geq a_{\text{long}} x, \quad u \geq -a_{\text{short}} x.$$

In the above, the inequalities are *component-wise* .

Portfolio problem with transaction costs is a QP:

$$\begin{array}{ll} \max_{x,u} & r^T x - \lambda x^T C x - c \cdot \sum_{i=1}^n u_i \\ \text{subject to} & Ax \leq b, \quad Cx = d, \\ & u \geq a_{\text{long}} x, \quad u \geq -a_{\text{short}} x. \end{array}$$

# Market impact

$$\min_x \; x^T C x + c_M \sum_{i=1}^n |x_i - x_{0,i}|^\alpha \;:\; x \in \mathcal{X}.$$

Market impact parameters $\alpha > 1$ and $c_M > 0$ must be estimated.

In CVX, use the function `pow_abs`. For $\alpha = 3/2$ (say), can model this as an SOCP.

# Diversification constraints

- "No more than 10% of total budget $B$ invested in assets of a given sector": assume sector corresponds to the first 10 assets, then we impose

$$x_1 + \ldots + x_{10} \leq 0.1B.$$

  This is a simple linear constraint.

- "No group of $k$ assets contains more than 80% of the total investment":

$$x_{[1]} + \ldots + x_{[k]} \leq 0.8 \sum_{i=1}^{n} x_i$$

  Here $x_{[i]}$ is the stock with $i$-th largest position.

  In CVX, use the function `sum_largest`.

# How does CVX handle this?

The condition

$$s_k(x) := x_{[1]} + \ldots + x_{[k]} \leq 0.8 \sum_{i=1}^{n} x_i$$

can be written as "$n$ choose $k$" linear inequalities, each of the form

$$x_{i_1} + \ldots + x_{i_k} \leq 0.8 \sum_{i=1}^{n} x_i$$

for a particular set of indices $\{i_1, \ldots, i_k\}$. Clearly, the number of inequalities grows very fast!

For, say, $n = 500$, $k = 20$, we have $\sim 10^{40}$ inequalities ...

# How does CVX handle this?

## Under CVX's hood

CVX relies on the following: for any $x$,

$$s_k(x) = \min_{\alpha}\; k\alpha + \sum_{i=1}^{n} \max(0, x_i - \alpha).$$

*Proof:* For every $\alpha$, and with $x_{[i]}$ the $i$-th largest element of $x$:

$$\begin{aligned} s_k(x) = k\alpha + \sum_{i=1}^{k}(x_{[i]} - \alpha) \quad &\leq \quad k\alpha + \sum_{i=1}^{n} \max(0, x_{[i]} - \alpha) \\ &= \quad k\alpha + \sum_{i=1}^{n} \max(0, x_i - \alpha), \end{aligned}$$

so that $s_k(x) \leq \min_\alpha (\ldots)$, and that bound is attained by $\alpha = x_{[k+1]}$.

# How does CVX handle this?

## Under CVX's hood

Hence, the condition

$$s_k(x) \leq 0.8 \sum_{i=1}^{n} x_i$$

iff there exist $u \in \mathbf{R}^n$ such that

$$k\alpha + \sum_{i=1}^{n} u_i \leq 0.8 \sum_{i=1}^{n} x_i, \quad u \geq 0, \quad u_i \geq x_i - \alpha, \quad i = 1, \ldots, n,$$

This involves only $2n+1$ linear inequalities involving $n+1$ new variables $(u, \alpha)$.

By adding $\sim n$ variables and constraints, we are able to represent a very complicated set in $x$-space ... Life is definitely easier in high dimensions!

# Computational benefits of factor models

## Direct approach

Assume $C = D + LL^T$, with $D \succ 0$, diagonal, and $F \in \mathbf{R}^{p \times k}$, with $k << p$.

We have seen in lecture 4 that the simple mean-variance model

$$\max_x \; r^T x - x^T C x$$

which leads to $x = C^{-1} r$, can be solved much faster by exploiting the "diagonal-plus-low-rank" structure of $C$.

How about a more complicated portfolio problem?

# Fast CVX implementation

Consider the model with transaction costs

$$\max_x \; r^T x - x^T C x - c\|x - x^0\|_1 \;:\; x \geq 0, \;\; \sum_i x_i = 1.$$

Rewrite this as

$$\max_x \; r^T x - x^T D x - z^T z - c\|x - x^0\|_1 \;:\; x \geq 0, \;\; \sum_i x_i = 1, \;\; z = F^T x.$$

In this form, CVX is much faster!

- In general, it is worth adding variables and equality constraints, if that makes the "$Q$" matrix of the standard-form QP sparser.
- Original QP scales as $\sim n^3$; factor form linear in $n$ (but, cubic in $k$).

# CVX code

```
cvx_begin
    variables x(n,1) z(k,1);
    maximize( r'*x - lambda*x'*D*x - lambda*z'*z ...
              - c*norm(x-x0,1))
    subject to
        x >= 0;
        sum(x) == B;
        z == F'*x;
cvx_end
```

- For $n = 100$, $k = 10$: $1 \sim 1.2$ faster.
- For $n = 500$, $k = 20$: $40 \sim 50$ times faster.
- For $n = 1000$, $k = 20$: $300 \sim 600$ times faster.

Another order of magnitude is gained by implementing this on commercial software (*e.g.*, Mosek [2], CPLEX [1].)

# Outline

# References


CPLEX user's manual.

E. D. Andersen and K. D. Andersen.
The MOSEK optimization package.

S. Boyd and M. Grant.
The CVX optimization package, 2010.

G. Cornuejols and R. Tütüncü.
*Optimization Methods in Finance*.
Cambridge University Press, 2007.