Financial Optimization
ISE 347/447

Lecture 21

Dr. Ted Ralphs

Reading for This Lecture

- C&T Chapter 16

Formalizing: Random Linear Optimization

- Consider the following linear program $LP(\omega)$ that is parameterized by the random vector ω:

minimize

$$c^\top x$$

subject to

$$\begin{aligned} Ax &= b \\ T(\omega)x &= h(\omega) \\ x &\in X \end{aligned}$$

- For now, we will assume $X = \{x \in \mathbb{R}^n : l \le x \le u\}$
- How do we make sense of this?

Example From Lecture 19 Revisited

minimize

$$x_1 + x_2$$

subject to

$$\begin{aligned} \omega_1 x_1 + x_2 &\geq 7 \\ \omega_2 x_1 + x_2 &\geq 4 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \end{aligned}$$

Random Mathematical Programs

- Again, we are dealing with decision problems where the decision x must be made before the realization of ω is known.
- We do, however, know the distribution of ω on Ω.
- In recourse models, the random constraints are essentially modeled as "soft" constraints. Possible violation is accepted, but the cost of violations will influence the choice of x.
- A *second stage* linear program is introduced that will describe how the violated random constraints are dealt with.

The Recourse LP

- In the simplest case, we may just penalize deviation in the constraints by penalty coefficient vectors q_+ and q_-.
- Then we have the following simple recourse LP that accounts for the deviations.
- The vector x is now fixed to $\hat{x}$ (our first stage decision), so it does not appear in the objective.
- The constraints involving only x are removed.

minimize

$$q_+^\top s(\omega) + q_-^\top t(\omega)$$

subject to

$$\begin{aligned} T(\omega)\hat{x} + s(\omega) - t(\omega) &= h(\omega) \\ s(\omega), t(\omega) &\geq 0 \end{aligned}$$

The Stochastic Programming Version

- The stochastic programming version of the overall problem is...

minimize

$$c^\top x + \mathbb{E}_\omega \left[q_+^\top s(\omega) + q_-^\top t(\omega) \right]$$

subject to

$$\begin{array}{rcll} Ax & = & b & \\ T(\omega)x + s(\omega) - t(\omega) & = & h(\omega) & \forall \omega \in \Omega \\ s(\omega), t(\omega) & \geq & 0 & \forall \omega \in \Omega \\ x & \in & X & \end{array}$$

Recourse

- In general, we can *react* in an intelligent (or optimal) way.
- We have some *recourse!*
- A recourse structure is provided by three items
  - q : a vector of recourse costs.
  - W : a $m \times p$ matrix, called the *recourse matrix* that describes explicit constraints
  - A set $Y \subseteq \mathbb{R}^p$ that describes implicit, deterministic constraints on the feasible set of recourse actions (e.g., nonnegativity).

Two-stage Stochastic Programs with Recourse

minimize

$$c^\top x + \mathbb{E}_\omega \left[q^\top y(\omega) \right]$$

subject to

$$\begin{aligned} Ax &= b \\ T(\omega)x + Wy(\omega) &= h(\omega) \qquad \forall \omega \in \Omega \\ x &\in X \\ y(\omega) &\in Y \end{aligned}$$

$$Q(x, \omega) = \min_{y \in Y} \{ q^\top y : Wy = h(\omega) - T(\omega)x \}$$

The Discrete Case

- For now, we consider the discrete case, where $\Omega = \{\omega_1, \omega_2, \ldots \omega_S\} \subseteq \mathbb{R}^r$.
- $\mathsf{P}(\omega = \omega_s) = p_s, \forall s = 1, 2, \ldots, S$.
- $T_s \equiv T(\omega_s), h_s = h(\omega_s)$.
- Right now, and in nearly all problems we will see, we have only one W.
- In other words, our recourse does not change with the scenario.
- This is called *fixed recourse*.

Deterministic Equivalent

- We can then write the *deterministic equivalent* as:

minimize

$$c^\top x \quad + \quad p_1 q^\top y_1 \quad + \quad p_2 q^\top y_2 \quad + \quad \cdots \quad + \quad p_s q^\top y_s$$

subject to

$$\begin{array}{ccccccccc}
Ax & & & & & & & = & b \\
T_1 x & + & W y_1 & & & & & = & h_1 \\
T_2 x & & & + & W y_2 & & & = & h_2 \\
\vdots & & & + & & \ddots & & & \\
T_S x & & & & & & + \quad W y_s & = & h_s \\
x \in X & & y_1 \in Y & & y_2 \in Y & & y_s \in Y & &
\end{array}$$

About the DE

- $y_s \equiv y(\omega_s)$ is the recourse action to take if scenario ω_s occurs.
- Pro: It's a linear program.
- Con: It's a BIG linear program.
  - $n + p|S|$ variables
  - $m_1 + m|S|$ constraints.
- Pro: The matrix of the linear program has a very special (staircase) structure.
  - Has anyone heard of Bender's Decomposition?
  - We will discuss this in the Lecture 22.

What is BIG

We have r random variables (That is why $\Omega \subseteq \mathbb{R}^r$)

- Imagine the following (real) problem. A Telecom company wants to expand its network in a way in which to meet an unknown (random) demand.
- There are 86 unknown demands. Each demand is independent and may take on one of seven values.
- $S = |\Omega| = \Pi_{k=1}^{86}(5) = 5^{86} = 4.77 \times 10^{72} = \#$ of subatomic particles in the universe.
- How do we solve a problem that has more variables and more constraints than the number of subatomic particles in the universe?

But Its Even Worse!

- If Ω doesn't have finite support, our "deterministic equivalent" would have an infinite number of variables and constraints.
- How do we solve that?
- Generally, we can't!
- We solve an approximating problem obtained through sampling.
- More on this later.

An Example

Let's solve a deterministic equivalent version of our example problem...

minimize

$$x_1 + x_2$$

subject to

$$\begin{aligned} \omega_1 x_1 + x_2 &\geq 7 \\ \omega_2 x_1 + x_2 &\geq 4 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \end{aligned}$$

- $\omega_1 \sim \mathcal{U}[1, 4]$
- $\omega_2 \sim \mathcal{U}[1/3, 1]$

A Recourse Formulation

- As usual, we approximate with a finite set of scenarios S.

minimize

$$x_1 + x_2 + \sum_{s \in S} p_s \lambda (y_{1s} + y_{2s})$$

subject to

$$\begin{aligned}
\omega_{1s} x_1 + x_2 + y_{1s} &\geq 7 \qquad \forall s \in S \\
\omega_{2s} x_1 + x_2 + y_{2s} &\geq 4 \qquad \forall s \in S \\
x_1 &\geq 0 \\
x_2 &\geq 0 \\
y_{1s} &\geq 0 \\
y_{2s} &\geq 0
\end{aligned}$$

AMPL Model

```
param n := 50;
set S := 1 .. n;
param p{s in S} default 1/card(S);
param w1{S} := Uniform(1,4);
param w2{S} := Uniform(1/3,1);

param PENALTY := 5;

var x1 >= 0;
var x2 >= 0;

var y1{S} >= 0;
var y2{S} >= 0;
```

AMPL Model (cont.)

```
minimize ObjPlusRecourse:
 x1 + x2 + sum{s in S} p[s] * PENALTY * (y1[s] + y2[s]);

subject to c1{s in S}:
w1[s] * x1 + x2 + y1[s] >= 7;

subject to c2{s in S}:
w2[s] * x1 + x2 + y2[s] >= 4;
```

Example: Multiperiod Production Planning

- A factory makes several different products
- A known quantity of resources (e.g., machines and labor) are needed to produce each product.
- A random demand must be met at the end of each period.
- Costs are induced when inventory is too large or too small
- To satisfy demand, additional labor and machine hours can be used, but these additions are bounded.
- There is a “hire and fire” cost associated with changing the workforce level.

Decision Problem

- Right now, we will decide
  - The number of each product to be produced in *each* period
  - The extra capacity to be used in *each* period
  - The hirings and firings to be done in *each* period
- Then the random demand is realized
  - Conceptually, this occurs for all periods at once.
  - One can think of this as the realization of a new forecast for all future periods.
- After we observe demand, we can decide how to best store product in inventory or purchase from an outside source.
- In this framework, how many stages are in the stochastic programming instance?

Lots of Definitions

Sets

- T : Number of periods. (Also set T).
- N : Set of products
- M : Set of resources

Variables

- x_{jt} : Amount of product $j \in N$ produced in period $t \in T$
- u_{it} : Additional amount of resource $i \in M$ to procure in period $t \in T$
- z_t^+, z_t^-: Planned increase/decrease of work force from period $t-1$ to t.
- y_{jt}^-, y_{jt}^+: Surplus/Shortage of product $j \in N$ at the end of period $t \in T$.

Parameters

- All of the above variables have associated costs $(\alpha, \beta, \gamma, \delta)$.
- ω_{jt} : (Random) demand for product $j \in N$ in period $t \in T$.
- U_{it} : Upper bound on u_{it}.
- a_{ij} : Amount of resource $i \in M$ needed to produce one unit of product $j \in N$
- b_{it} : Amount of resource $i \in M$ available at time $t \in T$.

The Stochastic Programming Model

minimize

$$\sum_{j\in N}\sum_{t\in T}\alpha_{jt}x_{jt}+\sum_{i\in M}\sum_{t\in T}\beta_{it}u_{it}+\sum_{t\in T\setminus 1}(\gamma^{+}_{t-1,t}z^{+}_{t-1,t}+\gamma^{-}_{t-1,t}z^{-}_{t-1,t})+$$
$$\sum_{s\in S}\sum_{j\in N}\sum_{t\in T}p_s(\delta^{+}_{jt}y^{+}_{jts}+\delta^{-}_{jt}y^{-}_{jts})$$

subject to

$$\begin{aligned}
\sum_{j\in N}a_{ij}x_{jt} &\leq b_{it}+u_{it} && \forall i\in M,\forall t\in T\\
u_{it} &\leq U_{it} && \forall i\in M,\forall t\in T\\
z^{+}_{t}-z^{-}_{t} &= \sum_{j\in N}a_{Lj}(x_{jt}-x_{j,t-1}) && \forall t\in T\setminus 1\\
x_{jt}+y^{-}_{j,t-1,s}+y^{+}_{jts}-y^{-}_{jts} &= \omega_{jts} && \forall j\in N,\forall t\in T,\forall s\in S.
\end{aligned}$$

Formalizing: Some Notation

$$\min_{x \in X : Ax = b} \left\{ c^\top x + \mathbb{E}_\omega \left[\min_{y \in Y} \{ q^\top y : Wy = h(\omega) - T(\omega)x \} \right] \right\}$$

Second stage value function, or *recourse (penalty) function* $v : \mathbb{R}^m \mapsto \mathbb{R}$:

- $v(z) \equiv \min_{y \in Y} \{ q^\top y : Wy = z \}$
- Given "policy" x and realization of randomness ω.
- If z measures the first-stage deviation $z = h(\omega) - T(\omega)x$, $v(z)$ is the minimum cost way to "correct" so that the constraints hold again.

Expected minimum recourse function $\mathcal{Q} : \mathbb{R}^n \mapsto \mathbb{R}$:

- $Q(x, \omega) = v(h(\omega) - T(\omega)x)$
- $\mathcal{Q}(x) \equiv \mathbb{E}_\omega [Q(x, \omega)]$
- For any policy $x \in \mathbb{R}^n$, it describes the expected cost of the recourse.

A Compact Formulation

- Using these definitions, we can write our recourse problem in terms only of the x variables:

$$\min_{x \in X} \{c^\top x + \mathcal{Q}(x) : Ax = b\}$$

- This is a (nonlinear) programming problem in $\mathbb{R}^n$.
- The ease of solving such a problem depends on the properties of $\mathcal{Q}(x)$.
- What does $\mathcal{Q}(x)$ look like?
  - Linear (?)
  - Convex (?)
  - Continuous (?)
  - Differentiable (?)

The Value Function

- For the time being, let $Y = \mathbb{R}^p_+$.

$$v(z) = \min_{y \in \mathbb{R}^p_+} \{q^\top y : Wy = z\}, z \in \mathbb{R}^m$$

- Thus, for a fixed z, we solve a linear program to evaluate $v(z)$.
- Assume for the moment that $-\infty < v(z) < \infty \forall z \in \mathbb{R}^m$
- Later, we will need some notation.
  - $\{y \in \mathbb{R}^p_+ : Wy = z\} = \emptyset \Rightarrow v(z) = \infty$
  - $\exists d \in \mathbb{R}^n_+$ such that $Wd = 0, q^\top d < 0 \Rightarrow v(z) = -\infty$.

Structure of v

- Under our assumptions...

$$v(z) = \min_{y \in \mathbb{R}^p_+} \{q^\top y : Wy = z\} = \max_{t \in \mathbb{R}^m} \{z^\top t : W^\top t \leq q\}$$

- Let $\Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_{|\Lambda|}\}$ be the set of extreme points of $\{t \in \mathbb{R}^m \mid W^\top t \leq q\}$.
  - Each of those extreme points λ_k is a potential optimal solution to the (dual) LP.
  - In fact, we know that if there is an optimal solution, there is one that occurs at an extreme point, so we can write...

$$v(z) = \max_{k=1,\ldots,|\Lambda|} \{z^\top \lambda_k\}, z \in \mathbb{R}^m.$$

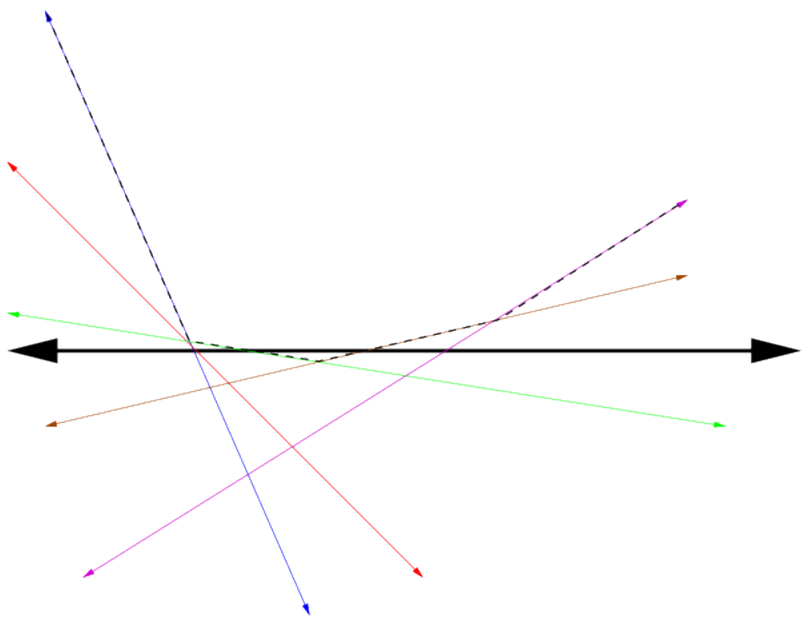
Proving Convexity

$$
\begin{aligned}
v(\alpha z_1 + (1-\alpha) z_2) &= \max_{k=1,\ldots,|\Lambda|} \{(\alpha z_1 + (1-\alpha) z_2)^\top \lambda_k\} \\
&= (\alpha z_1 + (1-\alpha) z_2)^\top \lambda_{k^*} \\
&= (\alpha z_1^\top \lambda_k^* + (1-\alpha) z_2^\top \lambda_k^* \\
&\leq \alpha \max_{k=1,\ldots,|\Lambda|} z_1^\top \lambda_k + (1-\alpha) \max_{k=1,\ldots,|\Lambda|} z_2^\top \lambda_k \\
&= \alpha v(z_1) + (1-\alpha) v(z_2)
\end{aligned}
$$

Convex!

- So $v(z)$ is convex of $z \in \mathbb{R}^m$.
- In fact...

Theorem 1. *If* $f_1(x), f_2(x), \ldots f_q(x)$ *is an arbitrary collection of convex functions, then* $M(x) = \max\{f_1(x), f_2(x), \ldots f_q(x)\}$ *is also a convex function.*



Structure of $Q(x, \omega)$

- What about $Q(x, \omega)$?
- Recall $Q(x, \omega) \equiv v(h(\omega) - T(\omega)x)$

$$\lambda Q(x_1, \omega) + (1 - \lambda) Q(x_2, \omega)$$

$$\begin{aligned}
&= \lambda v(h(\omega) - T(\omega)x_1) + (1 - \lambda) v(h(\omega) - T(\omega)x_2) \\
&\geq v(\lambda(h(\omega) - T(\omega)x_1) + (1 - \lambda)(h(\omega) - T(\omega)x_2)) \\
&= v(h(\omega) - T(\omega)(\lambda x_1 + (1 - \lambda)x_2)) \\
&= Q(\lambda x_1 + (1 - \lambda)x_2, \omega)
\end{aligned}$$

Continuing On

- So $Q(x, \omega)$ is convex in x for a fixed ω.
- In fact...

Theorem 2. *If* A *is a linear transformation from* $\mathbb{R}^n \mapsto \mathbb{R}^n$*, and* $f(x)$ *is a convex function on* $\mathbb{R}^m$*, the composite function* $(fA)(x) \equiv f(Ax)$ *is a convex function on* $\mathbb{R}^n$*.*

Almost Done...

- What about $\mathcal{Q}(x) \equiv \mathbb{E}_\omega Q(x, \omega)$?
- Let's now assume that ω comes from a probability space with finite support.
- This means that there are a finite number of discrete values $\{\omega_1, \omega_2, \ldots, \omega_m\}$ that ω can take.

$$\mathcal{Q}(x) = \sum_{i=1}^{m} P(\omega = \omega_i) Q(x, \omega_i)$$

Finishing up

Theorem 3. *If $f(x)$ is convex, and $\alpha \geq 0$, $g(x) \equiv \alpha f(x)$ is convex.*

Theorem 4. *If $f_k(x), k = 1, 2, \ldots K$ are convex functions, so is $g(x) \equiv \sum_{k=1}^{K} f_k(x)$.*

Put it all together and you get...$\mathcal{Q}(x)$ is a convex function of x!

A Simple Example

- Consider a two-stage version of the financial planning example from Lecture 20.
- In our current framework, we can say that the recourse LP for a fixed first-stage solution $\hat{x}$ (investment plan) is
  minimize
  $$qz + pw$$
  subject to
  $$w - z = G - \sum_{i \in \mathcal{N}} \mu_i x_i$$
- For this simple recourse LP, we can write the function $Q(x, \omega)$ in closed form.
  $$Q(x,\omega) = \begin{cases} q(G - \sum_{i\in\mathcal{N}} \mu_i(\omega) x_i) & \text{if } \sum_{i\in\mathcal{N}} \mu_i(\omega) x_i \geq G, \\ p(G - \sum_{i\in\mathcal{N}} \mu_i(\omega) x_i) & \text{otherwise.} \end{cases}$$
- In other words, it is a piecewise linear, convex function with two pieces.

Back to Mean-Variance Portfolio Optimization

- Finally, let's consider how to modify the portfolio optimization models from earlier in the course to fit into a stochastic programming framework.
- Question: Why do we need recourse?
- Answer: Rebalancing
- After some period of time, the portfolio we own may no longer be optimal (or even feasible) for the the same model solved *today*.
- We may have some additional constraints in our original formulation, e.g., to ensure diversification.
  - We may want to limit exposure to certain sectors.
  - We may want to impose a lower bound on the percentage of our portfolio each holding represents.
- We want to ensure these constraints continue to be satisfied.
- *Rebalancing* can be interpreted as a recourse action in a stochastic programming sense.
- We need to take into account transaction costs.

What is the Objective Function?

- In general, we try to maximize $\mathbb{E}v(x)$, where v is a utility function.
  - When v is linear, we are *risk neutral*.
  - When v is concave, we are *risk averse*.
- The Markowitz model can be seen as a simple stochastic programming model where $v = R^\top x + \lambda(R^\top x - \mu^\top x)^2$.
- We could develop multi-stage models using the same objective and including rebalancing (see Dantzig and Infanger '93).
- We will see a different utility function in Lecture 24.

Examining Assumptions

- What assumptions have we made?
- We assumed that $-\infty < v(z) < \infty \; \forall z = (h(\omega) - T(\omega)x)$, where x is any feasible first-stage solution and $\omega \in \Omega$.
  - This is a reasonable assumption in most applications and it is up to the modeler to ensure that it holds.
- We also assumed that Ω has finite support.
  - Although this is not technically the case for most applications, we usually have no choice but to work with discrete approximations.
  - In most cases, this is sufficient.