Financial Optimization
ISE 347/447

Lecture 5

Dr. Ted Ralphs

Reading for This Lecture

- C&T Chapter 2

Duality Theory: Motivation

- Consider the following minimization problem

$$z_P = \min x^2 + y^2$$
$$s.t.\ x + y = 1$$

- How could we solve this problem?
- Idea: Instead of strictly enforcing feasibility, penalize violation and make it part of the objective.
- Example: Consider the function

$$L(p) = \min_{y,y \in \mathbb{R}} x^2 + y^2 + p(1 - x - y)$$

- Then $L(p)$ is a lower bound on the optimal value of the original optimization problem.

The Lagrangian Dual

- Suppose we want to obtain the best possible lower bound on our original problem.
- This is itself an optimization probem and is known as the *Lagrangian dual*.
- In our example, the Lagrangian dual is

$$z_D = \max_{p \in \mathbb{R}} L(p) = \min_{y, y \in \mathbb{R}} x^2 + y^2 + p(1 - x - y)$$

- Since our original objective function was convex, we can calculate the value of $L(p)$ expicitly.
- We do this by taking the partial derivatives and setting them equal to zero yielding $x = y = p/2 \Rightarrow L(p) = p - p^2/2$.
- Since $L(p)$ is *concave*, we can maximize it by taking the derivative, yielding $p^* = 1$.
- Note that in this case, $x^* = y^* = p^*/2$, so the primal and dual problem have the same optimal value, but this is not always the case.

Lagrangian Duality for Linear Optimization

- We now show how to derive a Lagrangian dual for a linear optimization problem.
- Surprisingly, we will show that the Lagrangian dual is also an LP and that its value is the same as the original primal problem.
- As in the previous example associate a Lagrange multiplier, or *price*, with each constraint.
- Then we allow the constraint to be violated *for a price*.
- Consider an LP in standard form.
- Using Lagrange multipliers, we can formulate an alternative LP:

$$\begin{aligned} &\min\ c^\top x + p^\top(b - Ax) \\ &s.t. \quad x \geq 0 \end{aligned}$$

- How does the optimal solution of this compare to the original optimum?

Lagrange Multipliers

- As before, this new problem gives a lower bound.

$$g(p) = \min_{x \geq 0} \left[c^\top x + p^\top (b - Ax) \right] \leq c^\top x^* + p^\top (b - Ax^*) = c^\top x^*$$

- Since each value of p gives a lower bound, we consider maximizing $g(p)$.
- Think of this as finding the *best* lower bound.
- This is known as the *dual problem*.

Simplifying

- In linear programming, we can obtain an explicit form for the dual.

$$
\begin{aligned}
g(p) &= \min_{x \geq 0} \left[c^\top x + p^\top (b - Ax) \right] \\
&= p^\top b + min_{x \geq 0} (c^\top - p^\top A) x
\end{aligned}
$$

- Note that

$$
\min_{x \geq 0} (c^\top - p^\top A) x = \begin{cases} 0, & \text{if } c^\top - p^\top A \geq \mathbf{0}^\top, \\ -\infty, & \text{otherwise,} \end{cases}
$$

- Hence, we can show that the dual is equivalent to

$$
\begin{aligned}
&\max \; p^\top b \\
&s.t. \quad p^\top A \leq c^\top
\end{aligned}
$$

Inequality Form

- Suppose our feasible region is $\mathcal{P} = \{x \in \mathbb{R}^n \mid Ax \geq b, x \geq 0\}$.
- We can add slack variables and convert to standard form with constraints

$$[A| - I] \begin{bmatrix} x \\ s \end{bmatrix} = b$$

- This leads to dual constraints

$$p^\top [A| - I] \leq [c^\top | \mathbf{0}^\top]$$

- Hence, we get the dual

$$\begin{aligned} \max\ & p^\top b \\ s.t.\ & p^\top A \leq c^\top \\ & p \geq 0 \end{aligned}$$

From the Primal to the Dual

We can dualize general LPs as follows

PRIMAL	minimize	maximize	DUAL
constraints	$\geq b_i$ $\leq b_i$ $= b_i$	≥ 0 ≤ 0 free	variables
variables	≥ 0 ≤ 0 free	$\leq c_j$ $\geq c_j$ $= c_j$	constraints

Relationship of the Primal and the Dual

The following are the possible relationships between the primal and the dual:

	Finite Optimum	**Unbounded**	**Infeasible**
Finite Optimum	Possible	Impossible	Impossible
Unbounded	Impossible	Impossible	Possible
Infeasible	Impossible	Possible	Possible

Strong Duality

Proposition 1. *(Strong Duality) If a linear programming problem has an optimal solution, so does its dual, and the respective optimal costs are equal.*

- How might we go about proving this result?
- Let's consider again the simplex algorithm and the conditions under which it terminates with an optimal solution.

Dual Solutions from the Simplex Algorithm

- The optimality conditions implicit in the statement of the simplex algorithm are that
$$\bar{c}_j = c_j - c_B B^{-1} A_j \geq 0$$
- If we set $p = c_B B^{-1}$, then the above condition is exactly the feasibility condition for the j^{th} dual constraint.
- The condition that all the reduced costs be nonnegative is equivalent to dual feasibility.
- Hence, the simplex algorithm can be interpreted as maintaining primal feasibility while trying to achieve dual feasibility.
- There is an alternative algorithm that maintains dual feasibility while trying to achieve primal feasibility.

Complementary Slackness

Proposition 2. *If x and p are feasible primal and dual solutions to a general linear program with constraint matrix $A \in \mathbb{R}^{m \times n}$ and right-hand side vector $b \in \mathbb{R}^m$, then x and p are optimal if and only if*

$$\begin{aligned} p^\top (Ax - b) &= 0, \\ (c^\top - p^\top A)x &= 0. \end{aligned}$$

Optimality Without Simplex

Let's consider an LP in standard form. We have now shown that the optimality conditions for (nondegenerate) x are

1. $Ax = b$ (primal feasibility)
2. $x \geq 0$ (primal feasibility)
3. $x_i = 0$ if $p^\top a_i \neq c_i$ (complementary slackness)
4. $p^\top A \leq c$ (dual feasibility)

- In standard form, the complementary slackness condition is simply $x^\top \bar{c} = 0$.
- This condition is always satisfied during the simplex algorithm, since the reduced costs of the basic variables are zero.

More on Complementary Slackness

- Recall the complementary slackness conditions,

$$\begin{aligned} p^\top(Ax - b) &= 0, \\ (c^\top - p^\top A)x &= 0. \end{aligned}$$

- If the primal is in standard form, then any feasible primal solution satisfies the first condition.
- If the dual is in standard form, then any feasible dual solution satisfies the second condition.
- Typically, we only need to worry about satisfying the second condition, which is enforced by the simplex method.

Dual Variables and Marginal Costs

- Consider an LP in standard form with a nondegenerate, optimal basic feasible solution x^* and optimal basis B.
- Suppose we wish to perturb the right hand side slightly by replacing b with $b + d$.
- As long as d is "small enough," we have $B^{-1}(b + d) > 0$ and B is still an optimal basis.
- The optimal cost of the perturbed problem is

$$c_B^\top B^{-1}(b + d) = p^\top (b + d)$$

- This means that the optimal cost changes by $p^\top d$.
- Hence, we can interpret the optimal dual prices as the marginal cost of changing the right hand side of the i^{th} equation.

Economic Interpretation

- The dual prices, or *shadow prices* allow us to put a value on "resources" (broadly construed).
- Alternatively, they allow us to consider the sensitivity of the optimal solution value to changes in the input.
- Consider the bond portfolio problem from Lecture 3.
- By examining the dual variable for the each constraint, we can determine the value of an extra unit of the corresponding "resource".
- We can then determine the maximum amount we would be willing to pay to have a unit of that resource.
- Note that the reduced costs can be thought of as the shadow prices associated with the nonnegativity constraints.

Shadow Prices in AMPL

Again, recall the simple bond portfolio model from Lecture 3.

```
ampl: model bonds.mod;
ampl: solve;
...
ampl: display rating_limit, cash_limit;
rating_limit = 1
cash_limit = 2
```

- This tells us that the optimal dual value of the `rating_limit` constraint is 1.
- What does this tell us about the “cost” of improving the average rating?
- What is the return on an extra $\$1K$ of cash available to invest?

Another Interpretation of Dual Prices

- Let's consider again the dual prices for the constraints in the simple bond portfolio model.
- By combining the two constraints with nonzero prices, we can get a third inequality that must be satisfied by any feasible solution:

$$\begin{aligned} 2\,[x_1 + x_2 \leq 100] & \quad + \\ 1\,[2x_1 + x_2 \leq 150] & \quad = \\ 4x_1 + 3x_2 & \quad \leq 350 \end{aligned}$$

- What does this tell us about the optimal solution value?

Economic Interpretation of Optimality

Example: A simple product mix problem.

```
ampl: var X1;
ampl: var X2;
ampl: maximize profit: 3*X1 + 3*X2;
ampl: subject to hours: 3*X1 + 4*X2 <= 120000;
ampl: subject to cash: 3*X1 + 2*X2 <= 90000;
ampl: subject to X1_limit: X1 >= 0;
ampl: subject to X2_limit: X2 >= 0;
ampl: solve;
...
ampl: display X1;
X1 = 20000
ampl: display X2;
X2 = 15000
```

Shadow Prices in Product Mix Model

```
ampl: model simple.mod
ampl: solve;
...
ampl: display hours, cash;
hours = 0.5
cash = 0.5
```

- This tells us that increasing the hours by 2000 will increase profit by $(2000)(0.5) = \$1000$.
- Hence, we should be willing to pay up to $\$.50/\text{hour}$ for additional labor hours (as long as the solution remains feasible).
- We can also see that the availability of cash and man hours are contributing equally to the cost of each product.

Economic Interpretation of Optimality

- In the preceding example, we can use the shadow prices to determine how much each product "costs" in terms of its constituent "resources."
- The reduced cost of a product is the difference between its selling price and the (implicit) cost of the constituent resources.
- If we discover a product whose "cost" is less than its selling price, we try to manufacture more of that product to increase profit.
- With the new product mix, the demand for various resources is changed and their prices are adjusted.
- We continue until there is no product with cost less than its selling price.
- This is the same as having the reduced costs nonpositive (recall this was a maximization problem).
- Complementary slackness says that we should only manufacture products for which cost and selling price are equal.
- This can be viewed as a sort of multi-round auction.

Shadow Prices in Short Term Financing Model

```
display balance;

balance [*] :=
0  -1.03729
1  -1.0302
2  -1.02
3  -1.01695
4  -1.01
5  -1
;
```

How do we interpret these shadow prices?

More Sensitivity Analysis

- Using the simplex algorithm to solve a standard form problem, we know that if B is an optimal basis, then two conditions are satisfied:
  - $B^{-1}b \geq 0$
  - $c^T - c_B^T B^{-1} A \geq 0$
- When the problem is changed, we can check to see how these conditions are affected.
- When using the simplex method, we always have B^{-1} available, so we can easily recompute the appropriate quantities.
- If the change causes the optimality conditions to be violated, we can usually re-solve from the current basis using either primal or dual simplex.

Local Sensitivity Analysis

- For changes in the right-hand side,
  - Recompute the values of the basic variables, $B^{-1}b$.
  - Re-solve using dual simplex if necessary.
- For a changes in the cost vector,
  - Recompute the reduced costs.
  - Re-solve using primal simplex.
- For changes in a nonbasic column A_j
  - Recompute the reduced cost, $c_j - c_B B^{-1} A_j$.
  - Recompute the column in the tableau, $B^{-1} A_j$.
- For all of these changes, we can compute ranges within which the current basis remains optimal.

AMPL: Displaying Auxiliary Values with Suffixes

- In AMPL, it's possible to display much of the auxiliary information needed for sensitivity using suffixes.
- For example, to display the reduced cost of a variable, type the variable name with the suffix *.rc*.
- Recall again the short term financing example (`short_term_financing.mod`).

```
ampl: display credit.rc;
credit.rc [*] :=
 0  -0.003212
 1   0
 2  -0.0071195
 3  -0.00315
 4   0
 5   0
;
```

- How do we interpret this?

AMPL: Other Auxiliary Information

- You can display the *status* of each variable

```
ampl: display buy.status;
buy.status [*] :=
A   low
B   bas
C   bas
;
```

- You can also display such things as the slack in the constraints

```
ampl: display maturity_limit.slack;
maturity_limit.slack = 10
```

- Or the status of a slack variable

```
ampl: display maturity_limit.status;
maturity_limit.status = bas
```

- A list of all the possible suffixes is on the AMPL Web site.

AMPL: Sensitivity Ranges

- AMPL does not have built-in sensitivity analysis commands.
- AMPL/CPLEX does provide such capability.
- To get sensitivity information, type the following

```
ampl: option cplex_options 'sensitivity';
```

- Solve the model from product mix model:

```
ampl: solve;
...
suffix up OUT;
suffix down OUT;
suffix current OUT;
```

AMPL: Accessing Sensitivity Information

Access sensitivity information using the suffixes *.up* and *.down*. This is from the model `bonds.mod`.

```
ampl: display cash_limit.up, rating_limit.up, maturity_limit.up;
cash_limit.up = 102
rating_limit.up = 200
maturity_limit.up = 1e+20

ampl: display cash_limit.down, rating_limit.down, maturity_limit.down;
cash_limit.down = 75
rating_limit.down = 140
maturity_limit.down = 350

ampl: display buy.up, buy.down;
: buy.up buy.down    :=
A    6       3
B    4       2
;
```

AMPL: Sensitivity for the Short Term Financing Model

```
ampl: short_term_financing.mod;
ampl: short_term_financing.dat;
ampl: solve;
ampl: display credit, credit.rc, credit.up, credit.down;
:     credit      credit.rc     credit.up  credit.down    :=
0      0         -0.00321386   0.00321386    -1e+20
1     50.9804     0            0.00318204         0
2      0         -0.00711864   0.00711864    -1e+20
3      0         -0.00315085   0.00315085    -1e+20
4      0          0            0             -1e+20
;
```

AMPL: Sensitivity for the Short Term Financing Model (cont.)

```
ampl: display bonds, bonds.rc, bonds.up, bonds.down;
:      bonds       bonds.rc     bonds.up     bonds.down      :=
0     150           0          0.00399754   -0.00321386
1      49.0196      0          0            -0.00318204
2     203.434       0          0.00706931    0
3       0           0          0             0
4       0           0          0             0
;
```

AMPL: Sensitivity for the Short Term Financing Model (cont.)

```
ampl: display invest, invest.rc, invest.up, invest.down;
:      invest      invest.rc        invest.up          invest.down
-1      0          0                 0                    0
0       0         -0.00399754        0.00399754    -1e+20
1       0         -0.00714           0.00714       -1e+20
2     351.944      0                 0.00393091          -0.0031603
3       0         -0.00391915        0.00391915    -1e+20
4       0         -0.007             0.007         -1e+20
5      92.4969     0             1e+20                      2.76446e-14
;
```

Sensitivity Analysis of the Dedication Model

Let's look at the sensitivity information in the dedication model

```
ampl: model dedication.mod;
ampl: data dedication.dat;
ampl: solve;
ampl: display cash_balance, cash_balance.up, cash_balance.down;
: cash_balance cash_balance.up cash_balance.down    :=
1    0.971429         1e+20             5475.71
2    0.915646        155010             4849.49
3    0.883046        222579             4319.22
4    0.835765        204347             3691.99
5    0.656395        105306             2584.27
6    0.619461        123507             1591.01
7    0.5327          117131              654.206
8    0.524289        154630                0
;
```

How can we interpret these?

Sensitivity Analysis of the Dedication Model

```
ampl: display buy, buy.rc, buy.up, buy.down;
:      buy          buy.rc         buy.up      buy.down     :=
A     62.1361   -1.42109e-14      105          96.4091
B      0          0.830612       1e+20         98.1694
C    125.243    -1.42109e-14      101.843      97.6889
D    151.505     1.42109e-14      101.374      93.2876
E    156.808    -1.42109e-14      102.917      80.7683
F    123.08       0               113.036     100.252
G      0          8.78684        1e+20         91.2132
H    124.157      0               104.989      92.3445
I    104.09       0               111.457     101.139
J     93.4579     0                94.9        37.9011
;
```

Sensitivity Analysis of the Dedication Model

```
ampl: display cash, cash.rc, cash.up, cash.down;
: cash    cash.rc   cash.up   cash.down     :=
0   0    0.0285714   1e+20     0.971429
1   0    0.0557823   1e+20    -0.0557823
2   0    0.0326005   1e+20    -0.0326005
3   0    0.0472812   1e+20    -0.0472812
4   0    0.17937     1e+20    -0.17937
5   0    0.0369341   1e+20    -0.0369341
6   0    0.0867604   1e+20    -0.0867604
7   0    0.0084114   1e+20    -0.0084114
8   0    0.524289    1e+20    -0.524289
;
```

Sensitivity Analysis in PuLP

- PuLP creates suffixes by default when supported by the solver.
- The supported suffixed are `.pi` and `.rc`.

Sensitivity Analysis of the Dedication Model with PuLP

```
for t in Periods[1:]:
    prob += (cash[t-1] - cash[t]
             + lpSum(BondData[b, 'Coupon'] * buy[b]
               for b in Bonds if BondData[b, 'Maturity'] >= t)
             + lpSum(BondData[b, 'Principal'] * buy[b]
               for b in Bonds if BondData[b, 'Maturity'] == t)
             == Liabilities[t]), "cash_balance_%s"%t

status = prob.solve()

for t in Periods[1:]:
    print 'Present value of $1 liability for period', t,
    print prob.constraints["cash_balance_%s"%t].pi
```