

EE364a Homework 8

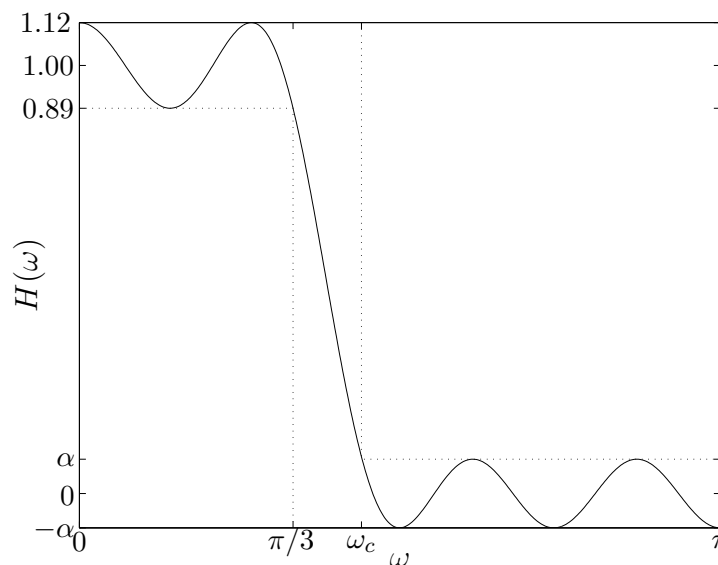
1. *FIR filter design.* Consider the (symmetric, linear phase) FIR filter described by

$$H(\omega) = a_0 + \sum_{k=1}^N a_k \cos k\omega.$$

The design variables are the real coefficients $a = (a_0, \dots, a_N) \in \mathbf{R}^{N+1}$. In this problem we will explore the design of a low-pass filter, with specifications:

- For $0 \leq \omega \leq \pi/3$, $0.89 \leq H(\omega) \leq 1.12$, *i.e.*, the filter has about ± 1 dB ripple in the ‘passband’ $[0, \pi/3]$.
- For $\omega_c \leq \omega \leq \pi$, $|H(\omega)| \leq \alpha$. In other words, the filter achieves an attenuation given by α in the ‘stopband’ $[\omega_c, \pi]$. ω_c is called the ‘cutoff frequency’.

These specifications are depicted graphically in the figure below.



- (a) Suppose we fix ω_c and N , and wish to maximize the stop-band attenuation, *i.e.*, minimize α such that the specifications above can be met. Explain how to pose this as a convex optimization problem.
- (b) Suppose we fix N and α , and want to minimize ω_c , *i.e.*, we set the stopband attenuation and filter length, and wish to minimize the ‘transition’ band (between $\pi/3$ and ω_c). Explain how to pose this problem as a quasiconvex optimization problem.

- (c) Now suppose we fix ω_c and α , and wish to find the smallest N that can meet the specifications, *i.e.*, we seek the shortest length FIR filter that can meet the specifications. Can this problem be posed as a convex or quasiconvex problem? If so, explain how. If you think it cannot be, briefly and informally explain why.
- (d) Plot the optimal tradeoff curve of attenuation (α) versus cutoff frequency (ω_c) for $N = 7$. Is the set of achievable specifications convex? Briefly explain any interesting features, *e.g.*, flat portions, of the optimal tradeoff curve.

For this subproblem, you may sample the constraints in frequency, which means the following. Choose $K \gg N$ (perhaps $K \approx 10N$), and set $\omega_k = k\pi/K$, $k = 0, \dots, K$. Then replace the specifications with

- For k with $0 \leq \omega_k \leq \pi/3$, $0.89 \leq H(\omega_k) \leq 1.12$.
- For k with $\omega_c \leq \omega_k \leq \pi$, $|H(\omega_k)| \leq \alpha$.

With this approximation, the problem in part (a) becomes an LP, which allows you to solve part (d) numerically.

2. *Maximum likelihood prediction of team ability.* A set of n teams compete in a tournament. We model each team's ability by a number $a_j \in [0, 1]$, $j = 1, \dots, n$. When teams j and k play each other, the probability that team j wins is equal to $\mathbf{prob}(a_j - a_k + v > 0)$, where $v \sim \mathcal{N}(0, \sigma^2)$.

You are given the outcome of m past games. These are organized as

$$(j^{(i)}, k^{(i)}, y^{(i)}), \quad i = 1, \dots, m,$$

meaning that game i was played between teams $j^{(i)}$ and $k^{(i)}$; $y^{(i)} = 1$ means that team $j^{(i)}$ won, while $y^{(i)} = -1$ means that team $k^{(i)}$ won. (We assume there are no ties.)

- (a) Formulate the problem of finding the maximum likelihood estimate of team abilities, $\hat{a} \in \mathbf{R}^n$, given the outcomes, as a convex optimization problem. You will find the *game incidence matrix* $A \in \mathbf{R}^{m \times n}$, defined as

$$A_{il} = \begin{cases} y^{(i)} & l = j^{(i)} \\ -y^{(i)} & l = k^{(i)} \\ 0 & \text{otherwise,} \end{cases}$$

useful.

The prior constraints $\hat{a}_i \in [0, 1]$ should be included in the problem formulation. Also, we note that if a constant is added to all team abilities, there is no change in the probabilities of game outcomes. This means that \hat{a} is determined only up to a constant, like a potential. But this doesn't affect the ML estimation problem, or any subsequent predictions made using the estimated parameters.

- (b) Find \hat{a} for the team data given in `team_data.m`, in the matrix `train`. (This matrix gives the outcomes for a tournament in which each team plays each other team once.) You may find the `cvx` function `log_normcdf` helpful for this problem. You can form A using the commands

```
A = sparse(1:m,train(:,1),train(:,3),m,n) + ...
      sparse(1:m,train(:,2),-train(:,3),m,n);
```

- (c) Use the maximum likelihood estimate \hat{a} found in part (b) to predict the outcomes of next year's tournament games, given in the matrix `test`, using $\hat{y}^{(i)} = \mathbf{sign}(\hat{a}_{j^{(i)}} - \hat{a}_{k^{(i)}})$. Compare these predictions with the actual outcomes, given in the third column of `test`. Given the fraction of correctly predicted outcomes.

The games played in `train` and `test` are the same, so another, simpler method for predicting the outcomes in `test` it to just assume the team that won last year's match will also win this year's match. Give the percentage of correctly predicted outcomes using this simple method.

3. *Planning production with uncertain demand.* You must order (nonnegative) amounts r_1, \dots, r_m of raw materials, which are needed to manufacture (nonnegative) quantities q_1, \dots, q_n of n different products. To manufacture one unit of product j requires at least A_{ij} units of raw material i , so we must have $r \succeq Aq$. (We will assume that A_{ij} are nonnegative.) The per-unit cost of the raw materials is given by $c \in \mathbf{R}_+^m$, so the total raw material cost is $c^T r$.

The (nonnegative) demand for product j is denoted d_j ; the number of units of product j sold is $s_j = \min\{q_j, d_j\}$. (When $q_j > d_j$, $q_j - d_j$ is the amount of product j produced, but not sold; when $d_j > q_j$, $d_j - q_j$ is the amount of unmet demand.) The revenue from selling the products is $p^T s$, where $p \in \mathbf{R}_+^n$ is the vector of product prices. The profit is $p^T s - c^T r$. (Both d and q are real vectors; their entries need not be integers.)

You are given A , c , and p . The product demand, however, is not known. Instead, a set of K possible demand vectors, $d^{(1)}, \dots, d^{(K)}$, with associated probabilities π_1, \dots, π_K , is given. (These satisfy $\mathbf{1}^T \pi = 1$, $\pi \succeq 0$.)

You will explore two different optimization problems that arise in choosing r and q (the variables).

I. Choose r and q ahead of time. You must choose r and q , knowing only the data listed above. (In other words, you must order the raw materials, and commit to producing the chosen quantities of products, before you know the product demand.) The objective is to maximize the expected profit.

II. Choose r ahead of time, and q after d is known. You must choose r , knowing only the data listed above. Some time after you have chosen r , the demand will become known to you. This means that you will find out which of the K demand vectors is the true demand. Once you know this, you must choose the quantities to be manufactured. (In other words, you must order the raw materials before the product demand is known; but you can choose the mix of products to manufacture after you have learned the true product demand.) The objective is to maximize the expected profit.

- (a) Explain how to formulate each of these problems as a convex optimization problem. Clearly state what the variables are in the problem, what the constraints are, and describe the roles of any auxiliary variables or constraints you introduce.
- (b) Carry out the methods from part (a) on the problem instance with numerical data given in `planning_data.m`. This file will define A , D , K , c , m , n , p and π . The K columns of D are the possible demand vectors. For both of the problems described above, give the optimal value of r , and the expected profit.

Standard form LP barrier method

In the following three exercises, you will implement a barrier method for solving the standard form LP

$$\begin{aligned} & \text{minimize} && c^T x \\ & \text{subject to} && Ax = b, \quad x \succeq 0, \end{aligned}$$

with variable $x \in \mathbf{R}^n$, where $A \in \mathbf{R}^{m \times n}$, with $m < n$. Throughout this exercise we will assume that A is full rank, and the sublevel sets $\{x \mid Ax = b, x \succeq 0, c^T x \leq \gamma\}$ are all bounded. (If this is not the case, the centering problem is unbounded below.)

1. *Centering step.* Implement Newton's method for solving the centering problem

$$\begin{aligned} & \text{minimize} && c^T x - \sum_{i=1}^n \log x_i \\ & \text{subject to} && Ax = b, \end{aligned}$$

with variable x , given a strictly feasible starting point x_0 .

Your code should accept A , b , c , and x_0 , and return x^* , the primal optimal point, ν^* , a dual optimal point, and the number of Newton steps executed.

Use the block elimination method to compute the Newton step. (You can also compute the Newton step via the KKT system, and compare the result to the Newton step computed via block elimination. The two steps should be close, but if any x_i is very small, you might get a warning about the condition number of the KKT matrix.)

Plot $\lambda^2/2$ versus iteration k , for various problem data and initial points, to verify that your implementation gives asymptotic quadratic convergence. As stopping criterion, you can use $\lambda^2/2 \leq 10^{-6}$. Experiment with varying the algorithm parameters α and β , observing the effect on the total number of Newton steps required, for a fixed problem instance. Check that your computed x^* and ν^* (nearly) satisfy the KKT conditions.

To generate some random problem data (*i.e.*, A , b , c , x_0), we recommend the following approach. First, generate A randomly. (You might want to check that it has full rank.) Then generate a random positive vector x_0 , and take $b = Ax_0$. (This ensures that x_0 is strictly feasible.) The parameter c can be chosen randomly. To be sure the sublevel sets are bounded, you can add a row to A with all positive elements. If you want to be able to repeat a run with the same problem data, be sure to set the state for the uniform and normal random number generators.

Here are some hints that may be useful.

- We recommend computing λ^2 using the formula $\lambda^2 = -\Delta x_{\text{nt}}^T \nabla f(x)$. You don't really need λ for anything; you can work with λ^2 instead. (This is important for reasons described below.)
- There can be small numerical errors in the Newton step Δx_{nt} that you compute. When x is nearly optimal, the computed value of λ^2 , *i.e.*, $\lambda^2 = -\Delta x_{\text{nt}}^T \nabla f(x)$, can

actually be (slightly) negative. If you take the squareroot to get λ , you'll get a complex number, and you'll never recover. Moreover, your line search will never exit. However, this only happens when x is nearly optimal. So if you exit on the condition $\lambda^2/2 \leq 10^{-6}$, everything will be fine, even when the computed value of λ^2 is negative.

- For the line search, you must first multiply the step size t by β until $x + t\Delta x_{\text{nt}}$ is feasible (*i.e.*, strictly positive). If you don't, when you evaluate f you'll be taking the logarithm of negative numbers, and you'll never recover.

2. *LP solver with strictly feasible starting point.* Using the centering code from part (1), implement a barrier method to solve the standard form LP

$$\begin{aligned} & \text{minimize} && c^T x \\ & \text{subject to} && Ax = b, \quad x \succeq 0, \end{aligned}$$

with variable $x \in \mathbf{R}^n$, given a strictly feasible starting point x_0 . Your LP solver should take as argument A , b , c , and x_0 , and return x^* .

You can terminate your barrier method when the duality gap, as measured by n/t , is smaller than 10^{-3} . (If you make the tolerance much smaller, you might run into some numerical trouble.) Check your LP solver against the solution found by `cvx`, for several problem instances.

The comments in part (1) on how to generate random data hold here too.

Experiment with the parameter μ to see the effect on the number of Newton steps per centering step, and the total number of Newton steps required to solve the problem.

Plot the progress of the algorithm, for a problem instance with $n = 500$ and $m = 100$, showing duality gap (on a log scale) on the vertical axis, versus the cumulative total number of Newton steps (on a linear scale) on the horizontal axis.

Your algorithm should return a $2 \times k$ matrix `history`, (where k is the total number of centering steps), whose first row contains the number of Newton steps required for each centering step, and whose second row shows the duality gap at the end of each centering step. In order to get a plot that looks like the ones in the book (*e.g.*, figure 11.4, page 572), you should use the following code:

```
[xx, yy] = stairs(cumsum(history(1,:)),history(2,:));
semilogy(xx,yy);
```

3. *LP solver.* Using the code from part (2), implement a general standard form LP solver, that takes arguments A , b , c , determines (strict) feasibility, and returns an optimal point if the problem is (strictly) feasible.

You will need to implement a phase I method, that determines whether the problem is strictly feasible, and if so, finds a strictly feasible point, which can then be fed to

the code from part (2). In fact, you can use the code from part (2) to implement the phase I method.

To find a strictly feasible initial point x_0 , we solve the phase I problem

$$\begin{aligned} & \text{minimize} && t \\ & \text{subject to} && Ax = b \\ & && x \succeq (1-t)\mathbf{1}, \quad t \geq 0, \end{aligned}$$

with variables x and t . If we can find a feasible (x, t) , with $t < 1$, then x is strictly feasible for the original problem. The converse is also true, so the original LP is strictly feasible if and only if $t^* < 1$, where t^* is the optimal value of the phase I problem.

We can initialize x and t for the phase I problem with any x^0 satisfying $Ax^0 = b$, and $t^0 = 2 - \min_i x_i^0$. (Here we can assume that $\min_i x_i^0 \leq 0$; otherwise x^0 is already a strictly feasible point, and we are done.) You can use a change of variable $z = x + (t - 1)\mathbf{1}$ to transform the phase I problem into the form in part (2).

Check your LP solver against `cvx` on several numerical examples, including both feasible and infeasible instances.