
EW7 Assignment: Linear programming

Note: A template will not be provided for this assignment. You still need to publish and submit your m-file, so neatly organize it using code cells and comments as was done in the previous templates. The autograder does not rely on the format of this file.

Turn in the following files (upload to Canvas; see Introduction module for step-by-step instructions):

- LP_Firstname_Lastname.m
- LP_Firstname_Lastname.pdf
- LP_Score.mat
- All function files that you are asked to create in the assignment

-
1. (Not graded) The Matlab command `fill` (a specialized wrapper around `patch`) is used to display polygonal regions in a plane. Examine and run the following code to acquaint yourself with the behavior of `fill`, which you will use later in this assignment.

```
begin code
1  XYpts = [-1 0; -1 2; 0 3; 2 3; 2 0];
2  Xshift = 0.5;
3  Yshift = 0.25;
4  XYpts_shifted=[XYpts(:,1)+Xshift XYpts(:,2)+Yshift];
5  figure
6  hold on
7  fill(XYpts(:,1), XYpts(:,2),'-');
8  fill(XYpts_shifted(:,1), XYpts_shifted(:,2),'-');
9  axis([-2 3 -1 4])
10 hold off
11
12 figure
13 hold on
14 hf1 = fill(XYpts(:,1), XYpts(:,2),'-');
15 get(hf1,'Type')
16 hf2 = fill(XYpts_shifted(:,1), XYpts_shifted(:,2),'-');
17 set(hf1,'Facecolor','cyan','Facealpha',.1,'Edgealpha',1,'Edgecolor','b');
18 set(hf2,'Facecolor','cyan','Facealpha',.1,'Edgealpha',1,'Edgecolor','b');
19 axis([-2 3 -1 4])
20 hold off
end code
```

-
2. A company makes and sells two different widgets, A and B. The demand for widgets is unlimited, but the company is constrained by the machine capacity and government enforced emissions restrictions. Production of A requires 3 machine hours/widget, and production of B requires 4 machine hours/widget. There are 20,000 machine hours available in the current production period. The factory produces 3 lbs. of CO₂ for each A produced, and 1 lb. of CO₂ for each B produced. The government has imposed a limit of the total CO₂ produced by the factory to be less than or equal to 12000 lbs. The production costs are \$3/widget for A and \$2/widget for B. The sale price of A is \$6/widget, and the sale price of B is \$5.40/widget.

- (a) Formulate a linear program (LP) to maximize profit, subject to the machine capacity and emissions restrictions. The LP should be in the standard form

$$\begin{aligned} \min_x \quad & c^T x \\ \text{Subject to} \quad & Ax \leq b \end{aligned}$$

where x is the vector of decision variables in \mathbb{R}^n , c is the cost vector in \mathbb{R}^n , A is a matrix in $\mathbb{R}^{m \times n}$ and b is a vector in \mathbb{R}^m .

Note: You do not need to turn anything in for the following items i - vi, but will need to determine them to complete part b, c, and d of this problem.

Consider the following as you proceed:

- i. What are the decision variables?
 - ii. What is the inequality constraint imposed by the machine capacity?
 - iii. What is the inequality constraint imposed by the emission restrictions?
 - iv. Are there other constraints? If so, what are they?
 - v. What quantity should be maximized?
 - vi. Write the LP in standard form, including a description of the meaning of the decision variable x . (**NOTE:** how do you transform a maximization into a minimization?)
- (b) Solve the problem graphically (**hand graded**)
- i. Using the `plot` command in MATLAB, plot the inequality constraints of the Linear Program problem. Use the `fill` command to define the feasibility set. Use appropriate axis limits when plotting the constraints.
 - ii. On the same graph, plot the α -levelset of the objective function to be minimized for $\alpha = -18000, -20000, -22000$. **Definition:** For any function f , the α -levelset is the set of all x in the domain of f such that $f(x) = \alpha$. For linear functions of n -variables, level sets are $(n - 1)$ -dimensional planes. So, for functions of 2 variables, the α -levelsets are straight lines in the 2-dimensional plane.
 - iii. What is the optimal solution of the Linear Program? **HINT:** The optimal solution is the coordinate of some vertex of the feasibility set.
 - iv. Plot the α -levelset of the objective function to be minimized corresponding to the optimal solution on the same graph of the inequality constraints.
- (c) Solve the problem using a MATLAB LP solver.

The file `lpsolver.m` (along with `lpLV.m`) is available at the Linear Programming module. The function defined in this file is a wrapper around an LP solver written by Prof. Lieven Vandenberghe of UCLA. We can use this function to solve for x , the optimal solution of a linear program in standard form, with the command:

```

1      _____ begin code _____
      [optx,lambda,status,gamma] = lp solver(c,A,b);
      _____ end code _____

```

Use `lp solver` to find the solution of the Linear Program you formulated in part (a). Verify that the solution computed by `lp solver` matches what you found in part (b).

- (d) Suppose that the company could reduce the emissions produced for B to 0.5 lbs., but this would increase the production time for B to 5 hours. Should they do it? (The answer is “yes” if doing so would increase the achievable profit, and “no” otherwise.)
3. Suppose you want to invest a certain amount of money in the stock market. You would like to maximize the expected return, but also limit the risk of the investment by having a diverse portfolio. The available stocks and there expected return are listed in the table below.

Stock	Industry	Country	Expected Return
BMW	Automotive	Germany	10.5%
Nissan	Automotive	Japan	10.1%
Apple	Electronics	USA	11.8%
Sony	Electronics	Japan	11.4%
State Farm	Insurance	USA	9.5%
Allianz	Insurance	Germany	9.3%

Suppose that in order to diversify the portfolio you also specify a minimum fraction (a single scalar value) of the money that will be invested in each industry, and a maximum fraction (also a scalar) of the money that should be invested in companies that are in a single country.

Using the given table above, write a function with the following declaration line

```

1      _____ begin code _____
      function [portfolio, expReturn] = maxReturn(money, industryDiv, countryDiv)
      _____ end code _____

```

where the input arguments are

- `money`: the amount of money to be invested,
- `industryDiv`: the minimum fraction of the money that should be invested in each industry,
- `countryDiv`: the maximum fraction of the money that should be invested in each country.

and the output arguments are

- `portfolio`: a 6-by-1 vector that contains the amount of money to be invested in each of the stocks. The order of this vector should correspond to the order that the stocks are listed in the table above.
- `expReturn`: a scalar value that is the expected return on the investment.

As an example, suppose you want to invest \$5,000. You decide that at least 25% of the \$5,000 should be invested in each industry, while no more than 40% of the \$5,000 should be invested in industries in a single country. Then the `maxReturn` function should be called as shown below and the function should return the outputs shown below.

```

begin code
1  [portfolio, expReturn] = maxReturn(5000, 0.25, 0.4)
2
3  portfolio =
4
5      1.0e+03 *
6
7      1.2500
8      0.0000
9      1.5000
10     1.0000
11     0.5000
12     0.7500
13
14
15  expReturn =
16
17     539.5000
end code

```

In this function you should formulate and solve a linear program(LP) to maximize the expected return, subject to the constraints described above. The LP should be in the standard form

$$\begin{aligned}
 & \min_x && c^T x \\
 & \text{Subject to} && Ax \leq b
 \end{aligned}$$

where x is the vector of decision variables in \mathbb{R}^n , c is the cost vector in \mathbb{R}^n , A is a matrix in $\mathbb{R}^{m \times n}$ and b is a vector in \mathbb{R}^m . In your function you should call the `lpsolver` function, to find the solution of the LP that you formulated. (**Note:** A more interesting/challenging problem would be that the table is given in a `struct` array, with fields `Stock`, `Industry`, `Country` and `ExpectedReturn`, and this would also be passed to the function `maxReturn`. For this introduction though, you will simply write the function to address the specific table listed above.)

Note: You do not need to turn anything in for the following items (a) - (e), but they should be considered when you work on this problem.

Consider the following as you proceed:

- (a) What are the decision variables?
 - (b) What is the inequality constraint imposed by the amount of money that will be invested?
 - (c) You can not invest negative amounts of money. What inequality constraints must be included to enforce this?
 - (d) What inequality constraints are necessary to enforce that the portfolio is diverse?
 - (e) What quantity should be maximized? How do you transform a maximization into a minimization problem?
4. In the previous few modules you studied the problem of minimizing $\|Ax - b\|_2$ by choice of x . So far you've done this in Matlab using either the backslash operator or the command `pinv`. Now

that you've been exposed to linear programming, you have the tools to solve two variations on this problem, namely minimizing

1. $\|Ax - b\|_1$
2. $\|Ax - b\|_\infty$

Recall that the 1-norm of a vector v with components (v_1, \dots, v_N) is defined to be

$$\|v\|_1 = \sum_{i=1}^N |v_i|,$$

and the ∞ -norm of the same vector is defined to be

$$\|v\|_\infty = \max_i |v_i|,$$

and minimization of either of these norms can be represented as a linear program.

We have provided partial code for the function

```

1  _____ begin code _____
   x = regressionNorms(A, b, nFlag)
   _____ end code _____

```

with inputs

1. A: the evaluated “basis” matrix in the regression problem
2. b: the “measurements” in the regression problem
3. nFlag: a **number** that is either 1, 2, or Inf, specifying which norm p to use when minimizing $\|Ax - b\|_p$

and output

1. x: minimizer of $\|Ax - b\|_p$

In particular, we have provided partial-code to set up and solve the case where $p = 1$ by transforming it into a linear program in standard form. You will complete the function using the backslash operator to solve the case where $p = 2$ (simple), and using the tools you learned in this module to solve the case where $p = \infty$ by transforming it into a linear program in standard form. For this latter case ($p = \infty$), your code will include a call to `lpsolver`.

5. *Linear Classification*: In this problem, a population P (often people, but not necessary) refers to a collection of distinct objects, called its *members*. Associated with each member m_k is a vector v_k of values of known, delineated traits of that member, called the member's *features*. The dimension of each feature vector is $n_F \times 1$. Hence, using this feature vector, each member of the population can be represented as a point in n_F dimensional space. Note though, it is possible for two different members to have identical features.

Suppose the the population P is divided into two distinct groups, G_1 and G_2 . The goal of *linear classification*, is to find a linear function, $L(v) := c^T v + b$, such that

$$c^T v_k + b > 0 \text{ for all } v_k \in G_1$$

and

$$c^T v_k + b < 0 \text{ for all } v_k \in G_2$$

In other words, the linear inequalities $c^T v + b > 0$ and $c^T v + b < 0$ split the n_F -dimensional space into two nonintersecting halfspaces, and one group of the population lies entirely in one halfspace, while the other group lies in the other halfspace. The linear function $L(v) := c^T v + b$ is said to *classify* the population into the two groups. **Such a function is often used as a predictor for another person, attempting to predict which group it belongs to, based on the value of $L(v)$, where v is the known vector of this person's traits.**

A concrete example will make the ideas clear. Suppose $n_F = 2$, and the traits (features) are the 2-dimensional vector

$$v = \begin{bmatrix} \text{age} \\ \text{number of action movies seen this year} \end{bmatrix}$$

The population consists of n_P people who have just viewed a pre-release showing of a new movie, called "MovieX". The two groups consists of the people who liked (based on a short exit survey) "MovieX" (G_1), and those people who did not like "MovieX" (G_2). Using this data, the producers of "MovieX" want to build a linear classifier (based on *age* and *number of action movies seen this year*) which will predict if another person (outside this population set) will like the movie, so they can target their promotions accordingly (eg., if they discover that older viewers who don't see many action movies prefer "MovieX", then they might do a pamphlet-handout to middle-aged people buying non-action movie tickets a week before "MovieX" is released).

The generalization is as follows: Given two sets of points

$$G_1 = (v_1, v_2, \dots, v_N), \quad G_2 = (v_{N+1}, v_{N+2}, \dots, v_P)$$

where each $v_k \in \mathbf{R}^{n_F}$. The task is to find a linear constraint that "separates" them. Specifically, find $c \in \mathbf{R}^{n_F}$ and $b \in \mathbf{R}$ such that

$$c^T v_k + b > 0 \text{ for all } k \leq N$$

and

$$c^T v_k + b < 0 \text{ for all } k > N.$$

Note that if this is possible, then simply by scaling c and b , it follows that

$$c^T v_k + b \geq 1 \text{ for all } k \leq N$$

and

$$c^T v_k + b \leq -1 \text{ for all } k > N.$$

A serious problem is that there may be some "outliers" in the data so that the two sets G_1 and G_2 simply can't be separated by a hyperplane (ie., there is some discrete "overlap" in the points). For those points, you need to add/subtract a nonnegative slack variable, $t_k \geq 0$ to make it work, so perhaps we adjust the requirement to

$$c^T v_k + b \geq 1 - t_k \text{ for all } k \leq N$$

and

$$c^T v_k + b \leq -(1 - t_k) \text{ for all } k > N.$$

But, hopefully **most points don't need this adjustment**, so you force the use of such t_k to a minimum by minimizing $\sum_{k=1}^P t_k$. This results in a linear program of the form

$$\min_{c,b,t_1,\dots,t_P} t_1 + t_2 + \dots + t_P$$

subject to

$$t_1 \geq 0, t_2 \geq 0, \dots, t_P \geq 0$$

and

$$c^T v_k + b \geq 1 - t_k \text{ for all } k \leq N$$

and

$$c^T v_k + b \leq -(1 - t_k) \text{ for all } k > N.$$

Write a function call `buildLinClass`, with function declaration line

```

_____ begin code _____
1  function [c,b,t] = buildLinClass(G1, G2)
_____ end code _____

```

The input arguments G_1 and G_2 are real-values arrays of dimension $n_F \times p_1$ and $n_F \times p_2$, respectively. Each column is a feature vector of a particular person in that group.

The output arguments are a $n_F \times 1$ vector c and a scalar value b , such that the linear function $L(v) := c^T v + b$ approximately classifies the two groups, with $L(v) > 0$ for members of G_1 and $L(v) < 0$ for members in G_2 . Of course, because of outliers, the classification is not necessarily exact, but the linear program minimization has minimized the total amount of slack variables used to create the separation. The 3rd output argument is the vector of required "corrections" t , and is of dimension $(p_1 + p_2) \times 1$.

You can try your function out on synthetic data (which has no outliers) below. Here there are only 2 features, so we can visually see the classification (could also do in 3-dimensions). Note that in practice, the number of features is usually much larger than 2, and hence cannot be visualized.

```

_____ begin code _____
1  nF = 2;
2  nP = 100;
3  cTrue = randn(nF,1);
4  bTrue = randn(1,1);
5  Pop = randn(nF, nP);
6  LPop = cTrue'*Pop + bTrue;
7  G1 = Pop(:,LPop>0); % create the populations based on their L-value
8  G2 = Pop(:,LPop<0); % create the populations based on their L-value
9  [cEst, bEst, tAdjust] = buildLinClass(G1,G2);
10 max(abs(tAdjust)) % should be 0 (or very close)
11 f1Min = min(Pop(1,:));
12 f1Max = max(Pop(1,:));
13 f2Min = min(Pop(2,:));
14 f2Max = max(Pop(2,:));
15 plot([f1Min-1 f1Max+1],-(cEst(1)*[f1Min-1 f1Max+1]+bEst)/cEst(2),'b',...

```

```

16     G1(1,:),G1(2,:), 'b*', G2(1,:),G2(2,:), 'ro')
17     ylim([f2Min-1 f2Max+1])
                                     end code

```

We can add some random noise to 10% of the data, so that outliers appear. Now the classification will likely not be perfect, but will still make good sense. Run this several times to gain intuition.

```

                                     begin code
1   nF = 2;
2   nP = 100;
3   nOut = floor(0.1*nP);
4   cTrue = randn(nF,1);
5   bTrue = randn(1,1);
6   Pop = randn(nF,nP);
7   Noise = zeros(1,nP);
8   Noise(1:nOut) = randn(1,nOut);
9   LPop = (cTrue'*Pop + bTrue) + Noise; % corrupt some L-values with noise
10  G1 = Pop(:,LPop>0);
11  G2 = Pop(:,LPop<0);
12  if isempty(G1) || isempty(G2)
13      disp('One population is empty, so there is nothing to separate');
14  end
15  [cEst, bEst, tAdjust] = buildLinClass(G1,G2);
16  max(abs(tAdjust)) % likely nonzero, and >1, dealing with non-separability
17  f1Min = min(Pop(1,:));
18  f1Max = max(Pop(1,:));
19  f2Min = min(Pop(2,:));
20  f2Max = max(Pop(2,:));
21  plot([f1Min-1 f1Max+1], -(cEst(1)*[f1Min-1 f1Max+1]+bEst)/cEst(2), 'b', ...
22       G1(1,:),G1(2,:), 'b*', G2(1,:),G2(2,:), 'ro')
23  ylim([f2Min-1 f2Max+1])
                                     end code

```