CS 435, Spring 2019, Prof. Calvin
Program #2.
Due: April 24, 11:30 am.

Consider the following classification problem: We have a set of p points $\mathcal{A} \subset \mathbb{R}^q$ and a set of r points $\mathcal{B} \subset \mathbb{R}^q$. The points can be represented by matrices $A \in \mathbb{R}^{p\times q}$ and $B \in \mathbb{R}^{r\times q}$, respectively. We want to find a hyperplane $H = \{x \in \mathbb{R}^q : w^T x = \gamma\}$ so that, to the extent possible, the points in $\mathcal{A}$ fall on one side of the hyperplane, and the points in $\mathcal{B}$ fall on the other side. This hyperplane is defined by the normal vector w and distance to zero $|\gamma|/\|w\|$, where for a vector $x = (x_1, x_2, \ldots, x_n)$,

$$\|x\| \equiv \|x\|_2 \equiv \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}.$$

We want the points of $\mathcal{A}$ to lie in the open half-space $\{x \in \mathbb{R}^q : w^T x > \gamma\}$ and the points of $\mathcal{B}$ to lie in the open half-space $\{x \in \mathbb{R}^q : w^T x < \gamma\}$; equivalently, $Aw > \gamma e_p$ and $Bw < \gamma e_r$. Normalizing, this becomes

$$Aw \geq \gamma e_p + e_p, \quad Bw \leq \gamma e_r - e_r. \tag{1}$$

If the convex hulls of $\mathcal{A}$ and $\mathcal{B}$ are disjoint, then we can find such a hyperplane. We can't count on such a favorable situation, so instead of requiring strict separation, we "penalize" the violations.

For a vector $x = (x_1, x_2, \ldots, x_n)$, denote by x^+ the vector with ith component x_i if $x_i > 0$, otherwise the ith component is 0. Using this notation, the violations of our desired inequalities (1) are

$$(-Aw + \gamma e_p + e_p)^+, \quad (Bw - \gamma e_r + e_r)^+.$$

A natural penalty is to apply some norm to these vectors. Let us use the "1-norm" defined by the sum of absolute values of the vector components:

$$\|x\|_1 = |x_1| + |x_2| + \cdots + |x_n|.$$

Then we consider the problem of choosing $w \in \mathbb{R}^q, \gamma \in \mathbb{R}$ to minimize

$$\frac{1}{p}\|(-Aw + \gamma e_p + e_p)^+\|_1 + \frac{1}{r}\|(Bw - \gamma e_r + e_r)^+\|_1.$$

This is equivalent to the following linear program: choose $w \in \mathbb{R}^q, \gamma \in \mathbb{R}, y \in \mathbb{R}^p, z \in \mathbb{R}^r$ to minimize

$$\frac{1}{p}\sum_{i=1}^{p} y_i + \frac{1}{r}\sum_{j=1}^{r} z_j$$

subject to:

$$\begin{aligned} -Aw + \gamma e_p + e_p &\leq y \\ Bw - \gamma e_r + e_r &\leq z \\ y \geq 0,\ z &\geq 0. \end{aligned}$$

(1) Formulate this optimization problem as a linear program.
(2) Describe the dual program.
(3) Test your classifier on the data on moodle. Each line in the files wdbcBenigh.txt and wdbcMalig.txt starts with an integer identifer, followed by 30 floating point numbers separated by commas. The data are extracted from the UCI machine learning repository at

`http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29`

> The 30 floating point numbers represent the results of diagnostic tests for breast cancer. The two files contain results for patients eventually found to have the disease and those found to be free of the disease. Use the first 100 lines of wdbcMalig.txt and the first 200 lines of wdbcBenigh.txt to build your classifier, and then test it on the remaining lines of the two files.

Type your dual program into comments at the beginning of your program file. Include an Instructions file that describes how to run your program, and any parameters it takes. (Move your files from Program 1 elsewhere prior to the due date.)

For the third part you need to solve a linear program to construct your classifier. Consider the LP example from the notes (Example 1, which we solved with the simplex method):

$$\begin{aligned} & \max x_1 + 2x_2 \\ s.t. \quad & \frac{2}{3}x_1 + x_2 \leq 4 \\ & x_1 + x_2 \leq 5 \\ & x_1 \leq 4 \\ & x_2 \leq 3 \\ & x_1, x_2 \geq 0. \end{aligned}$$

This is equivalent to:

$$\begin{aligned} & \max x_1 + 2x_2 \\ s.t. \quad & \frac{2}{3}x_1 + x_2 \leq 4 \\ & x_1 + x_2 \leq 5 \\ & x_1 \leq 4 \\ & x_2 \leq 3 \\ & -x_1 \leq 0 \\ & -x_2 \leq 0. \end{aligned}$$

The following Matlab program solves this linear program:

```
A = [2.0/3.0 1; 1.0 1.0; 1.0 0.0; 0.0 1.0; -1.0 0.0; 0.0 -1.0];
c = [-1 -2]';
b = [4.0 5.0 4.0 3.0 0.0 0.0]';
lps = linprog(c,A,b)
```

The following Python program also solves it:

```
A=[[2.0/3.0, 1.0], [1.0, 1.0], [1.0, 0.0], [0.0, 1.0], [-1.0, 0.0], [0.0, -1.0]]
c = [-1.0, -2.0]
b = [4.0, 5.0, 4.0, 3.0, 0.0, 0.0]
x0_bounds = (None, None)
x1_bounds = (None, None)
from scipy.optimize import linprog
res = linprog(c, A_ub=A, b_ub=b, bounds=(x0_bounds, x1_bounds), options={"disp": True})
print(res)
```