ECE 4803, Fall 2020

Homework #8

Due Thursday, November 19, at 11:59pm

1. Prepare a one paragraph summary of what we talked about in class since the last assignment. I do not want just a bulleted list of topics, I want you to use complete sentences and establish context (Why is what we have learned relevant? How does it connect with other classes?). The more insight you give, the better.

2. In the problems before you can/will make frequent use of the CVXPY package. You will probably need to install this package as it does not come standard with most python distributions. This should be easy – simply open up a python prompt and type `pip install cvxpy`. If you are on Windows you may also need to install the Visual Studio build tools if you don't already have them. If you are having any trouble, see https://www.cvxpy.org/install/ for details. Install CVXPY and poke around on the CVXPY website to get a feeling for what it can do.

3. In this problem we will explore two alternative approaches to solving a simple variant of the least squares problem where we add the constraint that the solution is non-negative, i.e., we wish to solve
$$\underset{\boldsymbol{x}\in\mathbb{R}^N}{\text{minimize}} \ \frac{1}{2}\|\boldsymbol{y}-\boldsymbol{A}\boldsymbol{x}\|_2^2 \qquad \text{subject to} \qquad \boldsymbol{x}\geq\boldsymbol{0}.$$
This is natural in many practical applications where the entries of $\boldsymbol{x}$ have physical interpretations (e.g., light intensity, power, concentration of some physical material, etc.) that don't really make sense as negative quantities. Below we will assume throughout that $\boldsymbol{A}$ is an $M\times N$ matrix with $M>N$ and that $\boldsymbol{A}$ is full rank.

   (a) Derive the Lagrangian function for this optimization problem (pay careful attention to the sign of each term).

   (b) Show that the dual optimization problem is itself another nonnegative least squares problem. Do this by deriving the dual function $d(\boldsymbol{\lambda})$ by first finding the $\boldsymbol{x}$ that minimizes the Lagrangian $\mathcal{L}(\boldsymbol{x},\boldsymbol{\lambda})$, and then plugging this into $\mathcal{L}(\boldsymbol{x},\boldsymbol{\lambda})$. Simplify the dual optimization problem as much as possible.

   (c) Recall from the notes that one of the KKT conditions (KKT4) is that if $\boldsymbol{x}^\star,\boldsymbol{\lambda}^\star$ are primal/dual optimal, then $\nabla_{\boldsymbol{x}}\mathcal{L}(\boldsymbol{x}^\star,\boldsymbol{\lambda}^\star)=\boldsymbol{0}$. Show that this implies that for the solution
$$\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{x}^\star-\boldsymbol{A}^{\mathrm{T}}\boldsymbol{y}-\boldsymbol{\lambda}^\star=0.$$
(Note that it is easy to find a $\boldsymbol{x}$ and $\boldsymbol{\lambda}$ that satisfy the condition above – for any $\boldsymbol{x}$ we can form a $\boldsymbol{\lambda}$ that will make this identity true. The trick is that usually the resulting $\boldsymbol{\lambda}$ will not actually satisfy $\boldsymbol{\lambda}\geq 0$.)

   (d) Try solving a nonnegative least squares problem using CVXPY. The file `hw08.py` contains some code to set up a nonnegative least squares problem and then solve it using CVXPY. Make sure you understand what the code is doing, and verify that the resulting solution satisfies the optimality condition from part (c).

(e) Implement the projected gradient descent approach described on page 15 of the notes. This should be a minor variation on something you have already implemented before, but there are two important caveats. First, when we solved least squares problems before using gradient descent, we calculated the optimal step size α_k at each iteration. This is much tougher to do in this case because the actual optimal step size would involve figuring out what α is optimal after accounting for the projection step, and this is not easy to do in closed form. You should either use a line search to choose α, or just take a fixed step size. For guidance, the theory guarantees convergence if $\alpha \leq 1/\|\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}\|_2$.

The second challenge here involves defining a stopping criterion. You cannot expect that the norm of the gradient will be zero at the solution. Instead you could either define a stopping criterion involving $\|\boldsymbol{x}^{(k)} - \boldsymbol{x}^{(k-1)}\|_2$, or alternatively you could do something inspired by part (c) above.

(f) Another way to solve this problem that avoids having to worry about a step size is a primal-dual approach where we take alternating steps over $\boldsymbol{x}$ and $\boldsymbol{\lambda}$ with the goal of satisfying the condition in (c). Specifically, suppose that we are given an estimate $\boldsymbol{\lambda}^{(k)}$. We can then update $\boldsymbol{x}$ by solving the optimality condition from part (c) to obtain

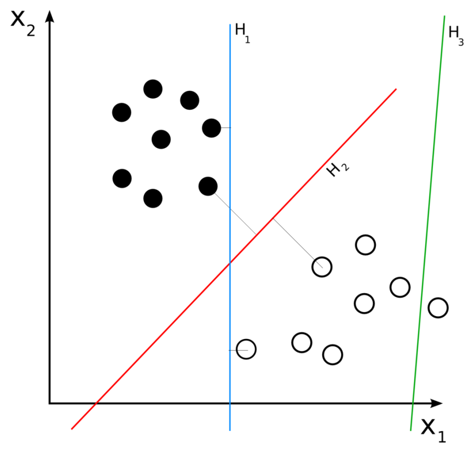
$$\boldsymbol{x}^{(k+1)} = \Big((\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A})^{-1}(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{y} + \boldsymbol{\lambda}^{(k)})\Big)_+ .$$

Note that we include the projection onto the set of nonnegative $\boldsymbol{x}$ since, in general, $(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A})^{-1}(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{y}+\boldsymbol{\lambda}^{(k)})$ will lead to negative entries (unless we are already at convergence). Next we can update $\boldsymbol{\lambda}$ by again solving the optimality condition from part (c), which in this case yields

$$\boldsymbol{\lambda}^{(k+1)} = \Big(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{x}^{(k+1)} - \boldsymbol{A}^{\mathrm{T}}\boldsymbol{y}\Big)_+ ,$$

where again we project the naïve solution to the optimality condition onto the set of nonnegative $\boldsymbol{\lambda}$. Implement this algorithm and compare the results to the projected gradient descent approach from the previous problem.

4. In this problem we explore another approach to binary classification problem: *support vector machines*. To fix our notation, suppose we are given points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_M \in \mathbb{R}^N$ with labels $y_1, \ldots, y_M$, where $y_m \in \{-1, +1\}$. We would like to find a hyperplane (i.e., an affine function) which *separates* the points:

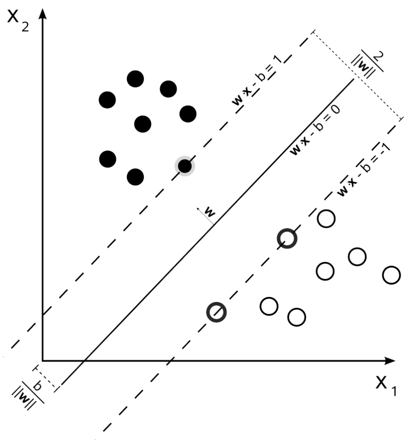


$\mathcal{H}_1$ and $\mathcal{H}_2$ above separate the points in $\mathbb{R}^2$, but $\mathcal{H}_3$ does not. To choose among the hyperplanes which separate the points,[1] we will take the one with maximum margin (maximize the distance to the closest point in either class).

To restate this, we want to find a $\boldsymbol{w} \in \mathbb{R}^N$ and $b \in \mathbb{R}$ such that

$$\begin{aligned} \langle \boldsymbol{x}_m, \boldsymbol{w} \rangle - b &\geq 1, \quad \text{when } y_m = 1, \\ \langle \boldsymbol{x}_m, \boldsymbol{w} \rangle - b &\leq -1, \quad \text{when } y_m = -1. \end{aligned}$$

In the formulation above, the distance between the two (parallel) hyperplanes is $2/\|\boldsymbol{w}\|_2$:



Thus maximizing this distance is the same as minimizing $\|\boldsymbol{w}\|_2$, and we can formulate our problem as

$$\underset{\boldsymbol{w} \in \mathbb{R}^N,\ b \in \mathbb{R}}{\text{minimize}} \ \frac{1}{2}\|\boldsymbol{w}\|_2^2 \quad \text{subject to} \quad y_m(b - \langle \boldsymbol{x}_m, \boldsymbol{w} \rangle) + 1 \ \leq 0, \quad m = 1, \ldots, M.$$

This is a convex problem (more specifically, a linearly constrained quadratic program). Notice that we are minimizing this with respect to both $\boldsymbol{w}$ and b. Even though b does not appear in the objective function, it does appear in the constraints.

(a) Show that the Lagrangian can be written as

$$\mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\lambda}) = \frac{1}{2}\|\boldsymbol{w}\|_2^2 + b\,\boldsymbol{\lambda}^{\mathrm{T}}\boldsymbol{y} - \boldsymbol{\lambda}^{\mathrm{T}}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{w} + \boldsymbol{\lambda}^{\mathrm{T}}\mathbf{1},$$

where $\boldsymbol{X}$ is the $N \times M$ matrix

$$\boldsymbol{X} = \begin{bmatrix} y_1\boldsymbol{x}_1 & y_2\boldsymbol{x}_2 & \cdots & y_M\boldsymbol{x}_M \end{bmatrix}$$

and $\mathbf{1}$ is a vector of all 1's.

(b) Use this to show that the dual function is given by

$$d(\boldsymbol{\lambda}) = \begin{cases} \frac{1}{2}\|\boldsymbol{X}\boldsymbol{\lambda}\|_2^2 - \boldsymbol{\lambda}^{\mathrm{T}}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{\lambda} + \boldsymbol{\lambda}^{\mathrm{T}}\mathbf{1}, & \langle \boldsymbol{\lambda}, \boldsymbol{y} \rangle = 0, \\ -\infty, & \text{otherwise.} \end{cases}$$

[1] Of course, it is possible that no separating hyperplane exists; in this case, there will be no feasible points in the program above. It is straightforward, though, to modify this discussion to allow "mislabeled" points.

(c) Argue from the above that the dual SVM program is then

$$\underset{\boldsymbol{\lambda}}{\text{maximize}} \quad -\frac{1}{2}\|\boldsymbol{X}\boldsymbol{\lambda}\|_2^2 + \sum_{m=1}^{M} \lambda_m \quad \text{subject to} \quad \langle \boldsymbol{\lambda}, \boldsymbol{y} \rangle = 0$$
$$\boldsymbol{\lambda} \geq \boldsymbol{0}.$$

Note that this optimization problem depends on the data only through $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}$, i.e., a matrix formed via inner products between the data points:

$$\|\boldsymbol{X}\boldsymbol{\lambda}\|_2^2 = \sum_{\ell=1}^{M}\sum_{m=1}^{M} y_\ell y_m \langle \boldsymbol{x}_\ell, \boldsymbol{x}_m \rangle.$$

(d) Recall from the notes that one of the KKT conditions is that if $\boldsymbol{w}^\star, b^\star$ is the solution to the primal problem and $\boldsymbol{\lambda}^\star$ is the solution to dual problem, then we must have

$$\lambda_m^\star g_m(\boldsymbol{w}^\star, b^\star) = 0$$

for $m = 1, \ldots, M$, where g_m represent the constraints in our primal problem. Show that this condition implies that given the solution $\boldsymbol{\lambda}^\star$ above, we can take $\boldsymbol{w}^\star = \boldsymbol{X}\boldsymbol{\lambda}^\star$.

(e) Using the same KKT condition from above, derive a formula for $b^\star$ in terms of $\boldsymbol{w}^\star$ and $\boldsymbol{\lambda}^\star$. [Hint: the answer is not necessarily unique. Consider any m for which $\lambda_m \neq 0$.]

(f) Show that if $\boldsymbol{w}^\star = \boldsymbol{X}\boldsymbol{\lambda}^\star$, then we can write our classifier as

$$h(\boldsymbol{x}) = \operatorname{sign}\left(\sum_{m=1}^{M} \lambda_m^\star y_m \langle \boldsymbol{x}, \boldsymbol{x}_m \rangle - b^\star\right).$$

Notice that the classifier only involved the data $\boldsymbol{x}_m$ through inner products with $\boldsymbol{x}$.

(g) The power of SVMs is that we can replace the inner products $\langle \boldsymbol{x}_\ell, \boldsymbol{x}_m \rangle$ with any "kernel function" $K(\boldsymbol{x}_\ell, \boldsymbol{x}_m) : \mathbb{R}^N \otimes \mathbb{R}^N \to \mathbb{R}$, provided that K satisfy certain properties (mainly that the $M \times M$ matrix with entries $K(\boldsymbol{x}_\ell, \boldsymbol{x}_m)$ is guaranteed to be positive semidefinite). As an example, suppose that you take

$$K(\boldsymbol{x}_\ell, \boldsymbol{x}_m) = (1 + \langle \boldsymbol{x}_\ell, \boldsymbol{x}_m \rangle)^2 = 1 + 2\langle \boldsymbol{x}_\ell, \boldsymbol{x}_m \rangle + \langle \boldsymbol{x}_\ell, \boldsymbol{x}_m \rangle^2.$$

Suppose that $\boldsymbol{x}_\ell, \boldsymbol{x}_m \in \mathbb{R}^2$. Argue that $K(\boldsymbol{x}_\ell, \boldsymbol{x}_m)$ is just computing the inner product $\langle \Phi(\boldsymbol{x}_\ell), \Phi(\boldsymbol{x}_m) \rangle$, where

$$\Phi(\boldsymbol{x}) = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_1^2 \\ x_2^2 \\ \sqrt{2}x_1x_2 \end{bmatrix}.$$

(h) Show that we can represent a circular decision boundary in $\mathbb{R}^2$ by a linear classifier in the space defined by the mapping Φ.

(i) (Optional.) Use CVXPY to implement an SVM classifier by solving the dual formulation. Apply it to the data generated in the file `hw08.py` using the quadratic kernel defined above, and plot the result.

5. In this problem we will explore the idea of *group testing* as a strategy for testing a large population for a rare disease by pooling samples together. Suppose that we have a population of N people and we collect saliva samples from each of them. We are looking for genetic signatures of a particular virus in these samples. Let x_n denote the concentration of this material in the sample for the n^{th} person. We will assume that for healthy people $x_n = 0$, but for infected people, $x_n > 0$. Our testing procedure[2] will be to form a series of M mixtures of samples from different subsets of people, and then only run tests on these mixtures. The goal here is to set $M < N$, and the question is then whether we can identify the infected people from the results of these tests.

   We will consider the following approach: we will form mixtures by constructing *random* combinations of samples, and we will attempt to recover the original $\boldsymbol{x}$ using a simple convex optimization problem.

   To mathematically represent the sampling/testing process, assume that we will ultimately run M tests, each of which will tell us the concentration of viral material in the combined sample being tested. For each person, their sample will be divided into $K < M$ equal portions, which will be assigned at random to the M tests. We will do this independently for each of the N people. We can ultimately represent the concentration of viral material in each of the mixed samples that we will ultimately test as a vector $\boldsymbol{y} \in \mathbb{R}^M$. We can write $\boldsymbol{y}$ as

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x},$$

   where $\boldsymbol{A}$ is a matrix that represents the assignment of people to mixed samples/tests. Specifically, $\boldsymbol{A}$ is a $M \times N$ matrix where each column is constructed independently by picking K entries at random, setting them to 1, and setting the remaining entries to 0.

   Suppose there is no noise in our tests, so that we can estimate $\boldsymbol{y}$ perfectly. Our inference problem is now to estimate $\boldsymbol{x}$ given knowledge of $\boldsymbol{y}$ and $\boldsymbol{A}$. In general, since $M < N$, recovering $\boldsymbol{x}$ is impossible. However, when $\boldsymbol{x}$ is sparse, meaning that it has only a nonzeros (in this case meaning that most of the population is negative), then recovering $\boldsymbol{x}$ *is* possible, although this fact was only really appreciated within the last 15 years or so.

   We will try to estimate $\boldsymbol{x}$ by solving the following optimization problem:

$$\underset{\boldsymbol{x}}{\text{minimize}} \ \|\boldsymbol{x}\|_1 \qquad \text{subject to} \qquad \boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}, \quad \boldsymbol{x} \geq \boldsymbol{0}.$$

   Below we will explore when and how well this works.

   (a) Suppose that you are testing a population of size $N = 1000$, but you can only process $M = 100$ tests. Assume that only 1% of the population is positive (meaning that there will be 10 infected individuals). Each person's sample will be split and added to $K = 10$ different batches. The file `hw08.py` contains code that sets up this problem. Use CVXPY to solve the optimization problem above and verify that this approach correctly identifies the 10 infected individuals.

   (b) Experiment with K. In practice, you might not want to divide a person's sample too many ways. How low can you set K before things begin to fail?

[2] Just to be clear, this is not exactly what Georgia Tech is doing, but it is similar in spirit. The approach that Georgia Tech is somewhat simpler and builds on ideas that date back to World War II, when they were used to test soldiers for syphilis.

(c) Suppose that the prevalence of the disease begins to grow beyond 1%. How widespread can the disease become before the approach begins to fail (holding M and N fixed). As the disease becomes more widespread, you may need to adjust K. What value of K seems to work best when the disease is widespread?