CSE 512: Homework — Due Wednesday, Sept. 23

1. Consider the problem

$$\underset{x}{\text{minimize}} \quad \overbrace{\underbrace{\frac{1}{2}\|Ax-b\|_2^2}_{=:f(x)}+\frac{\rho}{2}\|x\|_2^2}^{F(x)} \tag{1}$$

where $A \in \mathbb{R}^{m\times n}$ and $n > m$.

(a) Is the function $f(x) = \frac{1}{2}\|Ax-b\|_2^2$ L-smooth? Is it μ- strongly convex? If not, why? If so, what are the value of L and μ?

(b) Is the function $F(x) = f(x) + \frac{\rho}{2}\|x\|_2^2$ L-smooth or μ-strongly convex? If not, why? If so, what are the values of L and μ?

(c) **Exponential convergence.** Consider $b = 0$ and $\rho > 0$. Show that in this case, then gradient descent with step size $\alpha < 1/L$ on (1) converges with *exponential complexity*, e.g. for some constant c, the error $f(x^{(t)}) - f(x^*) = O(c^t)$. Note that this result does not depend on μ at all.

(d) **Nullspace component.** Now consider $A = \begin{bmatrix}1 & 1\\ 1 & 1\end{bmatrix}$, $b = \begin{bmatrix}1\\ -1\end{bmatrix}$, and $\rho = 0$.

i. Recall that for any vector x, from the linear decomposition theorem, we can uniquely write

$$x = u + v, \qquad u \in \mathbf{null}(A), \quad v \in \mathbf{range}(A^T).$$

We will denote the operation of pulling out the nullspace component of x as $u = \mathbf{proj}_{\mathbf{null}(A)}(x)$. Suppose that we run gradient descent for (1), starting at $x^{(0)} = \begin{bmatrix}0\\ 0\end{bmatrix}$. After t iterations, what is $\mathbf{proj}_{\mathbf{null}(A)}(x^{(t)})$? Why?

ii. Now suppose that $x^{(0)} = \begin{bmatrix}1\\ -1\end{bmatrix}$. After t iterations, what is $\mathbf{proj}_{\mathbf{null}(A)}(x^{(t)})$? Why?

iii. Now suppose that $\rho = 1$. For both cases of initial values, after t iterations, what is $\mathbf{proj}_{\mathbf{null}(A)}(x^{(t)})$? Why?

2. **Convex functions**

(a) Recall the objective function in logistic regression

$$f(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\log(\sigma(y_i x_i^T\theta)), \qquad \sigma(s) = \frac{1}{1+\exp(-s)}.$$

Show that $f(\theta)$ is convex.

(b) In matrix factorization, we attempt to characterize a matrix $R \in \mathbb{R}^{m\times n}$ in terms of low-rank factors $R \approx UV^T$, where $U \in \mathbb{R}^{m\times r}$ and $V \in \mathbb{R}^{n\times r}$. Take $r = 1$. (Note that $\mathbf{rank}(R) > 1$ in general.) Show that

$$f(u,v) = \frac{1}{2}\|R - uv^T\|_F^2$$

is nonconvex.

3. **Convex sets**

(a) **Norm balls.** Consider $\|x\|$ any norm. Show that the norm properties dictate that the *norm balls*, defined as

$$\mathcal{B}_r = \{x : \|x\| \le r\}$$

are always convex sets. Show that for any $r > 0$, the complement set

$$\bar{\mathcal{B}}_r = \{x : \|x\| > r\}$$

is nonconvex.

(b) **Level sets.** For a function $f : \mathbb{R}^d \to \mathbb{R}$, we define the *level sets* as

$$\mathcal{S}_r = \{x : f(x) \leq r\}.$$

Show that if f is convex, then all of its level sets $\mathcal{S}_r$ are convex. Is the opposite true? (If a function has only convex level sets, is it necessarily convex?)

4. Consider the hard margin SVM

$$\underset{\theta \in \mathbb{R}^n}{\text{minimize}} \quad \frac{1}{2}\|\theta\|_2^2 \qquad \text{subject to} \quad y_i x_i^T \theta \geq 1, \quad i = 1, ..., m \tag{2}$$

We would like to solve (2) using projected gradient descent. However, projecting on this constraint set (a halfspace)

$$\mathcal{H} = \{\theta : y_i x_i^T \theta \geq 1, \quad \forall i\}$$

is extremely nontrivial. (One possible way to do it is to use Dykstra's projection on the intersection of convex sets, where you iteratively project on the halfspace determined by each i, and use proper scaling. However, this takes a long time to converge, so we won't go there.)

We do this by first introducing *slack variables* s_i, and rewriting (2) equivalently as

$$\begin{array}{ll} \underset{\theta \in \mathbb{R}^n, s \in \mathbb{R}^m}{\text{minimize}} & \frac{1}{2}\|\theta\|_2^2 \\ \text{subject to} & y_i x_i^T \theta = 1 + s_i, \quad i = 1, ..., m \\ & s \geq 0 \end{array} \tag{3}$$

This problem is still a bit hard to solve, because the simultaneous projection on the affine constraint $y_i x_i^T \theta = 1 + s_i$ and the halfspace constraint $s \geq 0$ is as hard as projecting on $\mathcal{H}$. So, we will use a trick by transforming a constraint to a penalty. Specifically, we want to find a function $\phi(s)$ which is large when s violates the nonnegativity constraint, and 0 otherwise. We pick

$$\phi(s) = \frac{1}{2}\sum_{i=1}^{m} (\max\{-s_i, 0\})^2$$

and we consider the revised problem

$$\begin{array}{ll} \underset{\theta \in \mathbb{R}^n, s \in \mathbb{R}^m}{\text{minimize}} & \frac{1}{2}\|\theta\|_2^2 + \rho\phi(s) \\ \text{subject to} & y_i x_i^T \theta = 1 + s_i, \quad i = 1, ..., m. \end{array} \tag{4}$$

(a) For a fixed value of ρ, what is the gradient of the objective function of (4)? Is it L-smooth, and if so, what is the value of L?

(b) The least squares solution given by most solvers is also the *least norm solution*. That is, if the linear system $A\theta = b$ is solvable (there exists at least one solution) then the command `solve(A,b)` gives the unique solution to the optimization problem

$$\underset{\theta}{\text{minimize}} \quad \|\theta\|_2 \qquad \text{subject to} \quad A\theta = b.$$

Use this information to describe the steps to computing the projection onto the feasible set in (4). That is, given any $\hat{\theta}$, $\hat{s}$, show how to return the solution to

$$\underset{\theta, s}{\text{minimize}} \quad \|\theta - \hat{\theta}\|_2^2 + \|s - \hat{s}\|_2^2 \qquad \text{subject to} \quad y_i x_i^T \theta = 1 + s_i, \ i = 1, ..., m. \tag{5}$$

(c) **Coding**

- Download `arrhythmia.zip` and take a look at the three files inside. The original data is loaded in `arrhythmia.data` which is a csv file, with 279 features followed by a class label. A description of each feature and label is given in `arrhythmia.names`. Take a look at it, and understand what it is saying.

- The file `arrhythmia.mat` contains the data from `arrhythmia.data` loaded and organized, so that you have $x_i \in \mathbb{R}^{279}$ containing the $n = 279$ features and $y_i \in \{1, 2, \cdots, 15\}$ indicating the patient's diognosis. The matrix X and vector y are packed so that

$$X = \begin{bmatrix} x_1^T \\ \vdots \\ x_m^T \end{bmatrix} \in \mathbb{R}^{m \times n} \text{ and } y = \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix} \in \mathbb{R}^m.$$

  We want to convert this multiclass classification problem to a binary classification problem, so adjust y so that $y = 1$ means the patient is normal, and $y = -1$ means the patient is anything but normal. Comment on the class balancing; e.g., does it look like each class has enough "data representatives"?

- **Train/test split** To eliminate randomness, I have given you two index vectors: `idx_train` and `idx_test`, which randomly form a 60/40 train/test split. Later, when we discuss cross validation, we will see that you actually need to make these splits multiple times, to convince yourself that you are not overfitting. For now, we will just use these.
  Form a train and test data and label set using these indices. In python, your code should look like

```
Xtrain = X[idx_train,:]
ytrain = y[idx_train]
Xtest = X[idx_test,:]
ytest = y[idx_test]
```

  and in MATLAB,

```
Xtrain = X(idx_train,:);
ytrain = y(idx_train);
Xtest = X(idx_test,:);
ytest = y(idx_test);
```

- **Normalizing** Now we will use a more "standardized" normalization scheme, which is what is recommended if you don't really know anything special about your data. If we were medical experts, we might try something fancier, but for this exercise let's just do the simplest thing: de-meaning, followed by de-variancing.
  As before, first calculate the mean data vector over the training data

$$x_{\text{mean}} = \frac{1}{m_{\text{train}}} \sum_{i=1}^{m_{\text{train}}} x_i$$

  and remove this offset from the train and test data.
  Next, calculate the mean standard deviation over the training data:

$$x_{\text{std}}[k] = \frac{1}{m_{\text{train}}} \sqrt{\sum_{i=1}^{m_{\text{train}}} x_i[k]^2}$$

  This can be done with the built in `std` function in MATLAB or Python. Remove this multiplicatively from the train and test data, e.g.

$$x_i[k] \leftarrow x_i[k] / x_{\text{std}}[k]$$

  Now we're ready to go!

i. Solve the hard margin SVM reformulation in (4), sweeping $\rho = 10^{-4}, 10^{-2}, 1., 10^2$. Use a zero initialization as before, and $1/L$ as your stepsize with L as calculated in part (a). Run for 1000 iterations.
   Plot the margin value and the penalty $\phi(s)$ for the train set, and the train and test misclassificatition errors. Do this for $t = 1, ..., 1000$. Report also these final values.

ii. Is the data separable? Is this a reasonable approach for constrained optimization, given large enough ρ?

iii. Comment a bit on the *test* misclassification rate for varying values of ρ.

Challenge!

This challenge will look at two interesting extensions of the SVM problem.

1. **Projected gradient descent** Consider the convex constrained optimization problem

$$\underset{\theta}{\text{minimize}} \quad f(\theta) \qquad \text{subject to} \quad \theta \in \mathcal{C} \tag{6}$$

where $f : \mathbb{R}^d \to \mathbb{R}$ is a convex function and $\mathcal{C}$ is a convex set. The *normal cone* to $\mathcal{C}$ at a given point θ is defined as the set of points orthogonal to the tangent cone of $\mathcal{C}$ at θ; explicitly, it is

$$\mathcal{N}_{\mathcal{C}}(\theta) := \{g : g^T(\theta' - \theta) \leq 0, \quad \forall \theta' \in \mathcal{C}\}.$$

Recall that in *unconstrained* optimization, an optimality condition for convex optimization is that θ^* minimizes $f(\theta)$ if and only if $0 = \nabla f(\theta^*)$. In *constrained* optimization, the optimality condition is a bit more complicated:

$$\theta^* \text{ optimizes (6)} \iff -\nabla f(\theta^*) \in \mathcal{N}_{\mathcal{C}}(\theta^*).$$

   (a) Show that if θ^* is in the interior of $\mathcal{C}$, then this reduces to the condition of $0 = \nabla f(\theta^*)$. (We say that θ is in the interior of $\mathcal{C}$ if for any vector v we can find $\epsilon > 0$ small enough such that $\theta + v \in \mathcal{C}$.)

   (b) Recall the *projected gradient descent* method for solving (6) at each iteration computes

$$\theta^{(t+1)} = \mathbf{proj}_{\mathcal{C}}(\theta^{(t)} - \alpha \nabla f(\theta^{(t)}))$$

   for some step size $\alpha > 0$. Show that when this method is stationary, then optimality is reached. That is, any $\alpha > 0$,

$$\theta^* = \mathbf{proj}_{\mathcal{C}}(\theta^* - \alpha \nabla f(\theta^*)) \quad \iff \quad -\nabla f(\theta^*) \in \mathcal{N}_{\mathcal{C}}(\theta^*).$$

2. **Kaczmarz method.** In question 4 we found a way to find a separation in our dataset by solving some projection problems at each iteration. But, what if the number of training samples were very large? Then computing the projection at each step is extremely burdensome.

   Von Neumann in the 1949 figured out that a way to project on the intersection of hyperplanes is to just alternatingly project on each hyperplane. That is, if I want to find some θ where $A\theta = b$, then I could break down A and b as

$$A = \begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_K \end{bmatrix}, \qquad b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ v_K \end{bmatrix}$$

   and we simply need to find θ where

$$A_i\theta = b_i \qquad \text{for } i = 1, ..., K.$$

   Then using von Neumann's approach, we would simply pick any $\theta^{(0)}$, and alternatingly do

$$\theta^{(t+1)} = \mathbf{proj}_{\mathcal{C}_i}(\theta^{(t)})$$

   where $\mathcal{C}_i = \{\theta : A_i\theta = b_i\}$. Then, for i large enough, $\theta^{(i)}$ will converge to the true intersection of hyperplanes.

   This principle, when applied to this data separation problem, is also sometimes known as Kaczmarz' method. Specifically, we are going to break down our data matrix so that we only operate on one sample at a time. Unlike von Neumann's approach, however, we are only going to make a few passes through the data. This doesn't ensure full feasibility, but the hypothesis is that we get something reasonable nonetheless.

   (a) Describe the projection method on feasibility of a *single* data sample: e.g. given $\hat{\theta}$, how would we solve

$$\underset{\theta}{\text{minimize}} \quad \|\theta - \hat{\theta}\|_2^2 \qquad \text{subject to} \quad y_i x_i^T \theta = 1?$$

   Use the linear decomposition theorem to give an *explicit* solution, e.g. using matrix-vector multiplications only, and no matrix inverses. (One scalar inverse should be used.)

   Note that we are ignoring the slack variable in this approach, and forcing an equality rather than inequality.

(b) Initialize at $\theta^{(0)} = 0$. Randomly permute the training data samples. Then, taking 5 passes through the training data (permuting the data at each pass) where for each data sample, you project the current iterate on the feasible set for each data sample. After the 5 passes, pass, plot the margin size, infeasibility ($\sum_i \max\{1 - y_i x_i^T \theta, 0\}$), and misclassification rate over the test and train set. Do this for 5 separate random permutations.

(c) Comment on what you see. Is this a viable method? How is its generalization behavior? Compare it with the behavior from the SVM approach.