Homework 13

This homework is due at 11 PM on April 26, 2024.

Submission Format: Your homework submission should consist of a single PDF file that contains all of your answers (any handwritten answers should be scanned), as well as a printout of your completed Jupyter notebook(s).

1. **Newton's Method, Coordinate Descent and Gradient Descent**

   In this question, we will compare three different optimization methods: Newton's method, coordinate descent and gradient descent. We will consider the simple set-up of unconstrained convex quadratic optimization; i.e we will consider the following problem:

   $$\min_{\vec{x}\in\mathbb{R}^d} \vec{x}^\top A\vec{x} - 2\vec{b}^\top\vec{x} + c \tag{1}$$

   where $A \succ 0$ and $\vec{b} \in \mathbb{R}^d$.

   (a) How many steps does Newton's method take to converge to the optimal solution? Recall that the update rule for Newton's method is given by the equation:

   $$\vec{x}_{t+1} = \vec{x}_t - (\nabla^2 f(\vec{x}_t))^{-1}\nabla f(\vec{x}_t). \tag{2}$$

   when optimizing a function f.

   (b) Now, consider the simple two variable quadratic optimization problem for $\sigma > 0$:

   $$\min_{\vec{x}\in\mathbb{R}^2} f(\vec{x}) = \sigma x_1^2 + x_2^2. \tag{3}$$

   How many steps does coordinate descent take to converge on this problem? Assume that we start by updating the variable x_1 in the first step, x_2 in step two and so on; therefore, we will update x_1 and x_2 in odd and even iterations respectively:

   $$(x_{t+1})_1 = \begin{cases} \operatorname{argmin}_{x_1} f(x_1, (x_t)_2) & \text{for odd t} \\ (x_t)_1 & \text{otherwise} \end{cases} \text{ and } (x_{t+1})_2 = \begin{cases} \operatorname{argmin}_{x_2} f((x_t)_1, x_2) & \text{for even t} \\ (x_t)_2. & \text{otherwise} \end{cases} \tag{4}$$

   Here, $(x_t)_2$ represents x_2 at time t and so on.

   (c) We will now analyze the performance of coordinate descent on another quadratic optimization problem:

   $$\min_{\vec{x}\in\mathbb{R}^2} f(\vec{x}) = \sigma(x_1 + x_2)^2 + (x_1 - x_2)^2. \tag{5}$$

   where we have, as before, $\sigma > 0$. Note that $(0, 0)$ is the optimal solution to this problem. Now, starting from the point $\vec{x}_0 = (1, 1)$, write how each coordinate of $(\vec{x}_{t+1})_i$ relates to $(\vec{x}_t)_i$ for $i = 1, 2$. Use this to show how the algorithm converges from the initial point $(1, 1)$ to $(0, 0)$. What happens when σ grows large? *HINT: First find the update rule for $(\vec{x}_t)_1$, i.e. keep $(\vec{x}_t)_2$ fixed and figure out how $(\vec{x}_t)_1$ changes when t is odd. Then do the same for $(\vec{x}_t)_2$ when $(\vec{x}_t)_1$ is fixed for even t.*

   (d) Now, let $f(\vec{x}) = \frac{1}{2}\vec{x}^\top A\vec{x} + \vec{x}^\top\vec{b} + c$ where A is PD. When we run gradient descent on $f(\vec{x})$, the convergence along each of the unit eigenvectors $\vec{v}_i$ of A is

   $$|1 - \eta(\lambda_i\{A\})| \tag{6}$$

This can be derived similar to HW 8 Question 3e, which you may reference. Formally, in the current setting, we have

$$(\vec{x}_k - \vec{x}^\star) = (I - \eta A)^k (\vec{x}_0 - \vec{x}^\star)$$

One way we can derive an "optimal" learning rate $\eta^\star$ is to minimize the largest rate of convergence:

$$\eta^\star \in \operatorname*{argmin}_{\eta \in \mathbb{R}} \max_{i \in \{1,\ldots,n\}} \left| 1 - \eta\left(\lambda_i\{A\}\right) \right|. \tag{7}$$

One important property of $\eta^\star$ is that it makes the rates of convergence $|1 - \eta(\lambda_i\{A\})|$ associated with the largest and smallest singular values of A equal, i.e.,

$$|1 - \eta(\lambda_{\max}\{A\})| = |1 - \eta(\lambda_{\min}\{A\})|$$

Use this property to show that

$$\eta^\star = \frac{2}{\lambda_{\max}\{A\} + \lambda_{\min}\{A\}} \tag{8}$$

where $\lambda_{\min}\{A\} = \lambda_n\{A\}$ is the n^{th} largest singular value of A and similar for the maximum.

(e) Finally, for the objective function (5), write an equation relating $\vec{x}_t$ to $\vec{x}_0$ if $\vec{x}_0 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$. Assume for this part that $\sigma > 1$ and reason about how quickly gradient descent converges when σ grows large. *HINT: What is the optimal step size for gradient descent, using the previous part? HINT: Also note that f is given by:*

$$f(\vec{x}) = \vec{x}^\top A \vec{x} \textit{ where } A = 2\left(\sigma \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} + \begin{bmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix}\right). \tag{9}$$

2. **Gradient Descent vs Newton Method**

   Run the Jupyter notebook `gradient_vs_newton.ipynb` which demonstrates differences between gradient descent and Newton's method.

3. Modified SVM

Let $C > 0$. Suppose we have labeled data $(\vec{x}_i, y_i) \in \mathbb{R}^d \times \{-1, 1\}$ for $i = 1, \ldots, n$. For each i, define $\vec{z}_i \doteq y_i \vec{x}_i$. Finally, define $Z \doteq \begin{bmatrix} \vec{z}_1, \ldots, \vec{z}_n \end{bmatrix}^\top \in \mathbb{R}^{n \times d}$.

Recall that the soft-margin support vector machine problem can be expressed using slack variables as

$$\begin{aligned} p_1^\star = \min_{\vec{w}, \vec{s}} \quad & \frac{1}{2} \|\vec{w}\|_2^2 + C \sum_{i=1}^n s_i \\ \text{s.t.} \quad & s_i = \max\{0, 1 - \vec{z}_i^\top \vec{w}\}, \qquad \forall i \in \{1, \ldots, n\}. \end{aligned} \tag{10}$$

In this problem we consider a modified SVM program with a squared penalty:

$$\begin{aligned} p_2^\star = \min_{\vec{w}, \vec{s}} \quad & \frac{1}{2} \|\vec{w}\|_2^2 + \frac{C}{2} \sum_{i=1}^n s_i^2 \\ \text{s.t.} \quad & s_i = \max\{0, 1 - \vec{z}_i^\top \vec{w}\}, \qquad \forall i \in \{1, \ldots, n\}. \end{aligned} \tag{11}$$

We will use another representation of this program, namely one with affine constraints:

$$\begin{aligned} p^\star = \min_{\vec{w}, \vec{s}} \quad & \frac{1}{2} \|\vec{w}\|_2^2 + \frac{C}{2} \|\vec{s}\|_2^2 \\ \text{s.t.} \quad & \vec{s} \geq \vec{0} \\ & \vec{s} \geq \vec{1} - Z\vec{w}, \end{aligned} \tag{12}$$

where the inequality constraints are componentwise (as usual).

(a) Choose the smallest class that problem (12) belongs to (LP/QP/SOCP/etc).

(b) Prove that strong duality holds for (12).

(c) Are the KKT conditions for problem (12) necessary, sufficient or both necessary and sufficient for global optimality?

(d) Let $\vec{\alpha}$ be the dual variable corresponding to the constraint $\vec{s} \geq \vec{0}$. What is the dimension (i.e., number of entries) of $\vec{\alpha}$?

(e) Show that the Lagrangian $L(\vec{w}, \vec{s}, \vec{\alpha}, \vec{\beta})$ of problem (12), where $\vec{\alpha}$ is the dual variable corresponding to the constraint $\vec{s} \geq \vec{0}$, and $\vec{\beta}$ is the dual variable corresponding to the constraint $\vec{s} \geq \vec{1} - Z\vec{w}$, is equal to

$$L(\vec{w}, \vec{s}, \vec{\alpha}, \vec{\beta}) = \frac{1}{2} \|\vec{w}\|_2^2 + \frac{C}{2} \|\vec{s}\|_2^2 - \vec{s}^\top (\vec{\alpha} + \vec{\beta}) - \vec{w}^\top Z^\top \vec{\beta} + \vec{1}^\top \vec{\beta}. \tag{13}$$

(f) Write the KKT conditions for problem (12). Show that if $(\vec{w}^\star, \vec{s}^\star, \vec{\alpha}^\star, \vec{\beta}^\star)$ obey the KKT conditions for problem (12), then

$$\vec{w}^\star = Z^\top \vec{\beta}^\star \qquad \text{and} \qquad \vec{s}^\star = \frac{\vec{\alpha}^\star + \vec{\beta}^\star}{C}. \tag{14}$$

HINT: For the first order/stationarity condition on the Lagrangian you will need to consider partial derivatives with respect to both $\vec{w}$ and $\vec{s}$.

(g) Compute the dual function of problem (12) as

$$g(\vec{\alpha}, \vec{\beta}) \doteq L(\vec{w}^\star(\vec{\alpha}, \vec{\beta}), \vec{s}^\star(\vec{\alpha}, \vec{\beta}), \vec{\alpha}, \vec{\beta}) \tag{15}$$

where from the previous part we have that

$$\vec{w}^\star(\vec{\alpha}, \vec{\beta}) = Z^\top \vec{\beta} \qquad \text{and} \qquad \vec{s}^\star(\vec{\alpha}, \vec{\beta}) = \frac{\vec{\alpha} + \vec{\beta}}{C}. \tag{16}$$

Your final expression for $g(\vec{\alpha}, \vec{\beta})$ should not contain any maximizations, minimizations or terms including $\vec{w}$, $\vec{s}$, $\vec{w}^\star$, or $\vec{s}^\star$. It should only contain $\vec{\alpha}$, $\vec{\beta}$, C, Z, and numerical constants.

(h) Let $\vec{\alpha}^\star$ and $\vec{\beta}^\star$ be optimal dual variables that solve the problem

$$d^\star \doteq \max_{\vec{\alpha}, \vec{\beta} \geq \vec{0}} g(\vec{\alpha}, \vec{\beta}). \tag{17}$$

It turns out that $\vec{\alpha}^\star$ can also be obtained by solving the quadratic program:

$$\begin{aligned} \min_{\vec{\alpha}} \quad & \left\| \vec{\alpha} + \vec{\beta}^\star \right\|_2^2 \\ \text{s.t.} \quad & \vec{\alpha} \geq \vec{0}. \end{aligned} \tag{18}$$

Solve this quadratic program (18) directly and find $\vec{\alpha}^\star$.

HINT: The duality or KKT approaches are not recommended. Consider $\vec{\alpha} = \begin{bmatrix} \alpha_1 & \cdots & \alpha_n \end{bmatrix}^\top$*, and use the components of* $\vec{\alpha}$ *to decompose the problem into* n *separate scalar problems. Solve each one by checking critical points; that is, points where the gradient is* 0*, the boundary of the feasible set, and* $\pm\infty$*.*

(i) Let $\beta^\star$ be a solution to the dual problem. Characterize the pairs $(\vec{x}_i, y_i)$ which are "support vectors", i.e., contribute to the optimal weight vector $\vec{w}^\star$, in terms of $\beta^\star$.

4. Ridge Regression Classifier Vs. SVM

In this problem, we explore Ridge Regression as a classifier, and compare it to SVM. Recall Ridge Regression solves the problem

$$\min_{\vec{w}} \|X\vec{w} - \vec{y}\|_2^2 + \lambda \|\vec{w}\|_2^2 \,, \tag{19}$$

where $X \in \mathbb{R}^{m\times n}$, and $\vec{y} \in \mathbb{R}^n$

(a) Ridge Regression as is solves a regression problem. Given data $X \in \mathbb{R}^{m\times n}$ and labels $\vec{y} \in \{-1, 1\}^m$, explain how we might be able to train a Ridge Regression model and use it to classify a test point.

(b) Complete the accompanying Jupyter notebook to compare Ridge Regression and SVM.

5. Wasserstein distance between distributions

The Wasserstein distance is a measure of distance between probability distributions. The Wasserstein distance can roughly be thought of as the cost of turning one distribution to another distribution by moving probability mass around from one location to another. It is also sometimes called the earth-mover distance, because it may be visualized as the cost of moving a pile of dirt from one configuration to another.

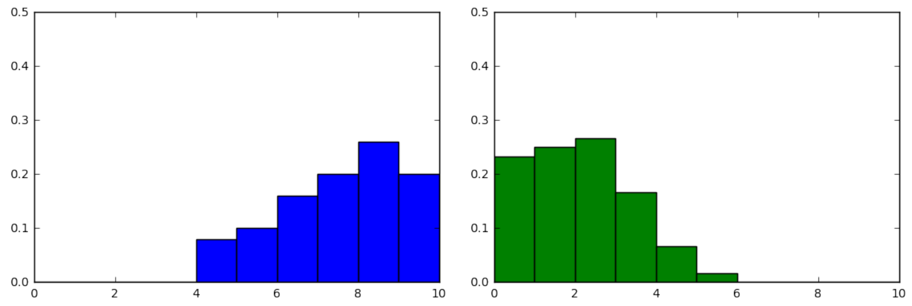

Figure 1: Visualization of μ histogram on left and ν histogram on right.

Let $n \in \mathbb{N}$. We define two discrete probability distributions $\vec{\mu} = (\mu_1, \cdots, \mu_n)$ and $\vec{\nu} = (\nu_1, \cdots, \nu_n)$; that is, $\mu_i, \nu_i \geq 0$ and $\sum_i \mu_i = \sum_i \nu_i = 1$.

We define $C \in \mathbb{R}^{n \times n}$ to be a cost matrix where $c_{ij} \geq 0$ is the cost of transporting one unit of probability mass from location $i \in \{1, \cdots, n\}$ to location $j \in \{1, \cdots, n\}$. We define a matrix $M \in \mathbb{R}^{n \times n}$ where $m_{ij} \geq 0$ denotes the quantity of probability mass to be moved from location i to location j. In summary, if we move m_{ij} units of probability mass from location i to location j, we incur cost $c_{ij} m_{ij}$.

In addition, the M matrix satisfies the following conditions. Row i of M indicates where all the probability mass in location i in the $\vec{\mu}$ distribution ends up. Hence, the sum of all the entries in row i must equal μ_i. Similarly, column j indicates where all the probability mass in location j in the $\vec{\nu}$ distribution came from. Hence, the sum of all the entries in column j must equal ν_j. We can summarize these conditions in math:

$$M\vec{1} = \vec{\mu} \tag{20}$$

$$M^\top \vec{1} = \vec{\nu}, \tag{21}$$

where $\vec{1}$ is a vector of 1s.

(a) What is the total cost of transporting the mass $\vec{\mu}$ into $\vec{\nu}$ by following the transportation plan dictated by the matrix M?

(b) Given the cost matrix C, write the optimization problem of finding the transportation plan $M^\star$ with minimal total cost. What type of optimization problem is it? (LP, QP, $\cdots$?).

Now, we apply the idea of Wasserstein distance to document similarity as illustrated in Fig. 2. Here, our application is that we want to identify words in two different documents that are most similar. This is mostly just a fun application, but may be of interest if you are trying to compare documents that are identical but in different languages. Here we consider a contrived example.

Natural Language Processing techniques have standard tools for converting words into vectors and embedding them in vector spaces, so that we can use machine learning and optimization tools on them. One such embedding is called *word2vec*. Assume we are provided with a *word2vec* embedding for the words in two documents. The word travel cost c_{ij} between word i and word j is the Euclidean distance $\|x_i - x_j\|_2$ in the word embedding space. We can compute the similarity between two documents as the minimum cumulative cost required to move all non-stop words from one document to the other.

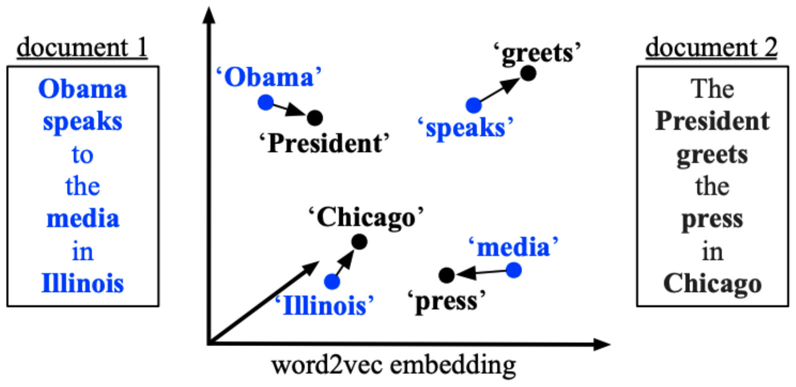

Figure 2: An illustration of the Wasserstein distance. All non-stop words (**bold**) of both documents are embedded into a *word embedding* space. The similarity between the two documents is the minimum cumulative distance that all words in document 1 need to travel to exactly match document 2.

(c) Using the `text_kantorovich.ipynb` Juypter notebook, implement the calculation of the Wasserstein distance in the notebook and use the provided code to visualize the resulting matrix M. Comment on the results.

6. Homework Process

With whom did you work on this homework? List the names and SIDs of your group members.

NOTE: If you didn't work with anyone, you can put "none" as your answer.