Homework 13

Self grades are due at 11 PM on May 3, 2024.

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.

Solution: Newton's method converges in a single step irrespective of the starting point. Let $\vec{x}_0$ be any starting point. We have:

$$\nabla^2 f(\vec{x}_0) = 2A \text{ and } \nabla f(\vec{x}_0) = 2(A\vec{x} - \vec{b}). \tag{3}$$

Therefore, we have:

$$\vec{x}_1 = \vec{x}_0 - A^{-1}(A\vec{x}_0 - \vec{b}) = A^{-1}\vec{b}. \tag{4}$$

Note that since this is an unconstrained convex quadratic optimization problem with A being full rank, we can find the optimum point by setting the derivative of the function to 0. Therefore, we have:

$$\nabla f(\vec{x}^*) = 2(A\vec{x}^* - \vec{b}) = 0 \implies \vec{x}^* = A^{-1}\vec{b}. \tag{5}$$

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

$$\min_{\vec{x}\in\mathbb{R}^2} f(\vec{x}) = \sigma x_1^2 + x_2^2. \tag{6}$$

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:

$$(x_{t+1})_1 = \begin{cases} \text{argmin}_{x_1} f(x_1, (x_t)_2) & \text{for odd t} \\ (x_t)_1 & \text{otherwise} \end{cases} \text{ and } (x_{t+1})_2 = \begin{cases} \text{argmin}_{x_2} f((x_t)_1, x_2) & \text{for even t} \\ (x_t)_2. & \text{otherwise} \end{cases} \tag{7}$$

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

Solution: On this problem, coordinate descent converges in 2 steps starting from any initialization point. Note that the optimal solution for each of the updates is 0, by setting the gradient to 0. Therefore, coordinate descent converges in two steps, one to update x_1 and the other to update x_2.

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

$$\min_{\vec{x}\in\mathbb{R}^2} f(\vec{x}) = \sigma(x_1 + x_2)^2 + (x_1 - x_2)^2. \tag{8}$$

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.*

Solution: We first find the update rule for x_1. Note that we only update x_1 when t is odd. Now, by taking the gradient and setting it to 0, we get:

$$\sigma((x_{t+1})_1 + (x_t)_2) + ((x_{t+1})_1 - (x_t)_2) = 0 \implies (x_{(t+1)})_1 = \frac{(1-\sigma)}{(1+\sigma)}(x_t)_2. \tag{9}$$

Note that the function, f, is symmetric in the variables, x_1 and x_2. Therefore, the update rule for x_2 (when t is even) is given by:

$$(x_{(t+1)})_2 = \frac{(1-\sigma)}{(1+\sigma)}(x_t)_1. \tag{10}$$

Therefore, we get for all $t \geq 2$:

$$(x_t)_1 = \left(\frac{1-\sigma}{1+\sigma}\right)^{2\left\lfloor\frac{t+1}{2}\right\rfloor - 1} \text{ and } (x_t)_2 = \left(\frac{1-\sigma}{1+\sigma}\right)^{2\left\lfloor\frac{t}{2}\right\rfloor}. \tag{11}$$

When σ grows large, the $\frac{1-\sigma}{1+\sigma}$ goes to -1 and this results in slow convergence as the algorithm converges quickly when $\left|\frac{1-\sigma}{1+\sigma}\right|$ is small.

(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

$$|1 - \eta\,(\lambda_i\{A\})| \tag{12}$$

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:

$$\eta^\star \in \operatorname*{argmin}_{\eta\in\mathbb{R}} \max_{i\in\{1,\ldots,n\}} |1 - \eta\,(\lambda_i\{A\})|\,. \tag{13}$$

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

$$\eta^\star = \frac{2}{\lambda_{\max}\{A\} + \lambda_{\min}\{A\}} \tag{14}$$

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

Solution: We have

$$|1-\eta^\star(\lambda_{\min}\{A\})| = |1-\eta^\star(\lambda_{\max}\{A\})| \tag{15}$$
$$1-\eta^\star(\lambda_{\min}\{A\}) = -(1-\eta^\star(\lambda_{\max}\{A\})) \tag{16}$$
$$1-\eta^\star(\lambda_{\min}\{A\}) = \eta^\star(\lambda_{\max}\{A\})-1 \tag{17}$$
$$2 = \eta^\star(\lambda_{\max}\{A\}+\lambda_{\min}\{A\}) \tag{18}$$
$$\eta^\star = \frac{2}{\lambda_{\max}\{A\}+\lambda_{\min}\{A\}}. \tag{19}$$

Here the second equality is the most challenging to derive. It follows from the first inequality by the following reasoning:

- If $1-\eta^\star(\lambda_{\min}\{A\})$ and $1-\eta^\star(\lambda_{\max}\{A\})$ have the same sign, then by the first equality, they must be equal. This means that $\lambda_{\max}\{A\} = \lambda_1\{A\} = \lambda_2\{A\} = \cdots = \lambda_n\{A\} = \lambda_{\min}\{A\}$ and the optimal learning rate $\eta^\star$ sets each rate $1-\eta^\star(\lambda_i\{A\})$ to 0 simultaneously, ensuring convergence in one step. If both sides are 0 then the second equality holds (because $0 = -0$).
- Otherwise, $1-\eta^\star(\lambda_{\min}\{A\})$ and $1-\eta^\star(\lambda_{\max}\{A\})$ have opposite signs. Since $\lambda_{\max}\{A\} > \lambda_{\min}\{A\}$ (since if they were equal we would be in the first case), we have $1-\eta^\star(\lambda_{\min}\{A\}) > 1-\eta^\star(\lambda_{\max}\{A\})$. Thus $1-\eta^\star(\lambda_{\min}\{A\})$ must be positive and $1-\eta^\star(\lambda_{\max}\{A\})$ must be negative. The absolute value of a negative number is its negative, so the second equality follows directly from the first equality.

(e) Finally, for the objective function (8), 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:*

$$f(\vec{x}) = \vec{x}^\top A\vec{x} \text{ where } A = 2\left(\sigma\begin{bmatrix}\frac{1}{\sqrt{2}}\\\frac{1}{\sqrt{2}}\end{bmatrix}\begin{bmatrix}\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}}\end{bmatrix} + \begin{bmatrix}\frac{1}{\sqrt{2}}\\-\frac{1}{\sqrt{2}}\end{bmatrix}\begin{bmatrix}\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}}\end{bmatrix}\right). \tag{20}$$

Solution: We first note that f is given by:

$$f(\vec{x}) = \vec{x}^\top A\vec{x} \text{ where } A = 2\left(\sigma\begin{bmatrix}\frac{1}{\sqrt{2}}\\\frac{1}{\sqrt{2}}\end{bmatrix}\begin{bmatrix}\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}}\end{bmatrix} + \begin{bmatrix}\frac{1}{\sqrt{2}}\\-\frac{1}{\sqrt{2}}\end{bmatrix}\begin{bmatrix}\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}}\end{bmatrix}\right). \tag{21}$$

Therefore, we have that $\lambda_{\max}$ of A is 2σ and $\lambda_{\min}$ is 2. Using the result from the previous part (and dividing by 2 since the optimal learning rate was computed for $\frac{1}{2}\vec{x}^\top A\vec{x} + \vec{x}^\top\vec{b} + c$ and not $\vec{x}^\top A\vec{x} - 2\vec{b}^\top\vec{x} + c$), the step size for gradient descent is set to $1/(2\sigma+2)$. Now, we have that:

$$\nabla f((1,-1)) = \begin{bmatrix}4\\-4\end{bmatrix}. \tag{22}$$

Therefore, we have that:

$$\vec{x}_1 = \vec{x}_0 - \eta\nabla f((1,-1)) = \left(1-\frac{2}{\sigma+1}\right)\begin{bmatrix}1\\-1\end{bmatrix}. \tag{23}$$

By iterating the above procedure we see that:

$$\vec{x}_t = \left(1-\frac{2}{\sigma+1}\right)^t\begin{bmatrix}1\\-1\end{bmatrix}. \tag{24}$$

Therefore, when σ is large, the convergence rate of gradient descent is really slow. However, Newton's method would find the optimum in one step.

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 $Z \doteq \left[\vec{z}_1, \ldots, \vec{z}_n\right]^\top \in \mathbb{R}^{n \times d}$.

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

$$\begin{aligned} p_1^\star = \min_{\vec{w}, \vec{s}} \quad & \frac{1}{2} \|\vec{w}\|_2^2 + C \sum_{i=1}^n s_i \\ \text{s.t.} \quad & s_i = \max\{0, 1 - \vec{z}_i^\top \vec{w}\}, \qquad \forall i \in \{1, \ldots, n\}. \end{aligned} \tag{25}$$

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

$$\begin{aligned} p_2^\star = \min_{\vec{w}, \vec{s}} \quad & \frac{1}{2} \|\vec{w}\|_2^2 + \frac{C}{2} \sum_{i=1}^n s_i^2 \\ \text{s.t.} \quad & s_i = \max\{0, 1 - \vec{z}_i^\top \vec{w}\}, \qquad \forall i \in \{1, \ldots, n\}. \end{aligned} \tag{26}$$

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

$$\begin{aligned} p^\star = \min_{\vec{w}, \vec{s}} \quad & \frac{1}{2} \|\vec{w}\|_2^2 + \frac{C}{2} \|\vec{s}\|_2^2 \\ \text{s.t.} \quad & \vec{s} \geq \vec{0} \\ & \vec{s} \geq \vec{1} - Z\vec{w}, \end{aligned} \tag{27}$$

where the inequality constraints are componentwise (as usual).

(a) Choose the smallest class that problem (27) belongs to (LP/QP/SOCP/etc).

Solution: It is a QP – it has a quadratic objective and affine constraints.

(b) Prove that strong duality holds for (27).

Solution: The objective function is a convex quadratic and the constraints are affine (hence convex) in $\vec{w}$ and $\vec{s}$, so the problem is convex. Furthermore, there is a strictly feasible point – we can construct one by picking any $\vec{w}$ and then picking $\vec{s}$ whose components are large enough to fulfill the inequalities. This is always possible since there is no upper bound on the components of $\vec{s}$. Thus Slater's condition holds, so strong duality holds.

(c) Are the KKT conditions for problem (27) necessary, sufficient or both necessary and sufficient for global optimality?

Solution: The objective function is a convex quadratic and the constraints are affine (hence convex) in $\vec{w}$ and $\vec{s}$, so the problem is convex.

Since the problem is convex, all functions involved are continuously differentiable, and strong duality holds, the KKT conditions are both necessary and sufficient for optimality; that is, they are equivalent to optimality conditions.

(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 $\vec{\alpha}$?

Solution: $\vec{\alpha} \in \mathbb{R}^n$ since $\vec{s} \in \mathbb{R}^n$.

(e) Show that the Lagrangian $L(\vec{w}, \vec{s}, \vec{\alpha}, \vec{\beta})$ of problem (27), 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

$$L(\vec{w}, \vec{s}, \vec{\alpha}, \vec{\beta}) = \frac{1}{2}\|\vec{w}\|_2^2 + \frac{C}{2}\|\vec{s}\|_2^2 - \vec{s}^\top(\vec{\alpha} + \vec{\beta}) - \vec{w}^\top Z^\top \vec{\beta} + \vec{1}^\top \vec{\beta}. \tag{28}$$

Solution: We have

$$L(\vec{w}, \vec{s}, \vec{\alpha}, \vec{\beta}) = \frac{1}{2}\|\vec{w}\|_2^2 + \frac{C}{2}\|\vec{s}\|_2^2 + \vec{\alpha}^\top(-\vec{s}) + \vec{\beta}^\top(\vec{1} - Z\vec{w} - \vec{s}) \tag{29}$$

$$= \frac{1}{2}\|\vec{w}\|_2^2 + \frac{C}{2}\|\vec{s}\|_2^2 - \vec{s}^\top(\vec{\alpha} + \vec{\beta}) - \vec{w}^\top Z^\top \vec{\beta} + \vec{1}^\top \vec{\beta}. \tag{30}$$

(f) Write the KKT conditions for problem (27). Show that if $(\vec{w}^\star, \vec{s}^\star, \vec{\alpha}^\star, \vec{\beta}^\star)$ obey the KKT conditions for problem (27), then

$$\vec{w}^\star = Z^\top \vec{\beta}^\star \qquad \text{and} \qquad \vec{s}^\star = \frac{\vec{\alpha}^\star + \vec{\beta}^\star}{C}. \tag{31}$$

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

Solution: Let $(\vec{w}^\star, \vec{s}^\star, \vec{\alpha}^\star, \vec{\beta}^\star)$ satisfy the KKT conditions. We have:

- Primal feasibility: $\vec{s}^\star \geq 0$ and $\vec{s}^\star \geq \vec{1} - Z\vec{w}^\star$.
- Dual feasibility: $\vec{\alpha}^\star \geq \vec{0}$, $\vec{\beta}^\star \geq \vec{0}$.
- Complementary slackness: $\alpha_i^\star s_i^\star = 0$ and $\beta_i^\star(1 - \vec{z}_i^\top \vec{w}^\star - s_i^\star) = 0$ for each i.
- Stationarity: $\nabla_{\vec{w}} L(\vec{w}^\star, \vec{s}^\star, \vec{\alpha}^\star, \vec{\beta}^\star) = \vec{0}$ and $\nabla_{\vec{s}} L(\vec{w}^\star, \vec{s}^\star, \vec{\alpha}^\star, \vec{\beta}^\star) = \vec{0}$. These become

$$\vec{0} = \vec{w}^\star - Z^\top \vec{\beta}^\star \tag{32}$$

$$\vec{0} = C\vec{s}^\star - (\vec{\alpha}^\star + \vec{\beta}^\star) \tag{33}$$

which rearrange to the claimed equalities.

(g) Compute the dual function of problem (27) as

$$g(\vec{\alpha}, \vec{\beta}) \doteq L(\vec{w}^\star(\vec{\alpha}, \vec{\beta}), \vec{s}^\star(\vec{\alpha}, \vec{\beta}), \vec{\alpha}, \vec{\beta}) \tag{34}$$

where from the previous part we have that

$$\vec{w}^\star(\vec{\alpha}, \vec{\beta}) = Z^\top \vec{\beta} \qquad \text{and} \qquad \vec{s}^\star(\vec{\alpha}, \vec{\beta}) = \frac{\vec{\alpha} + \vec{\beta}}{C}. \tag{35}$$

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.

Solution: The dual function is

$$g(\vec{\alpha}, \vec{\beta}) = L(\vec{w}^\star(\vec{\alpha}, \vec{\beta}), \vec{s}^\star(\vec{\alpha}, \vec{\beta}), \vec{\alpha}, \vec{\beta}) \tag{36}$$

$$= \frac{1}{2}\left\|\vec{w}^\star(\vec{\alpha}, \vec{\beta})\right\|_2^2 + \frac{C}{2}\left\|\vec{s}^\star(\vec{\alpha}, \vec{\beta})\right\|_2^2 - \vec{s}^\star(\vec{\alpha}, \vec{\beta})^\top(\vec{\alpha} + \vec{\beta}) - \vec{w}^\star(\vec{\alpha}, \vec{\beta})^\top Z^\top \vec{\beta} + \vec{1}^\top \vec{\beta} \tag{37}$$

$$= \frac{1}{2}\left\|Z^\top \vec{\beta}\right\|_2^2 + \frac{C}{2}\left\|\frac{\vec{\alpha} + \vec{\beta}}{C}\right\|_2^2 - \left(\frac{\vec{\alpha} + \vec{\beta}}{C}\right)^\top (\vec{\alpha} + \vec{\beta}) - \vec{\beta}^\top ZZ^\top \vec{\beta} + \vec{1}^\top \vec{\beta} \tag{38}$$

$$= -\frac{1}{2}\vec{\beta}^\top ZZ^\top \vec{\beta} - \frac{1}{2C}\left\|\vec{\alpha} + \vec{\beta}\right\|_2^2 + \vec{1}^\top \vec{\beta}. \tag{39}$$

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

$$d^\star \doteq \max_{\vec{\alpha},\vec{\beta}\geq\vec{0}} g(\vec{\alpha},\vec{\beta}). \tag{40}$$

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

$$\begin{aligned} \min_{\vec{\alpha}} \quad & \left\|\vec{\alpha}+\vec{\beta}^\star\right\|_2^2 \\ \text{s.t.} \quad & \vec{\alpha}\geq\vec{0}. \end{aligned} \tag{41}$$

Solve this quadratic program (41) 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* $\pm\infty$*.*

Solution: We have that

$$\left\|\vec{\alpha}+\vec{\beta}^\star\right\|_2^2 = \sum_{i=1}^n (\alpha_i+\beta_i^\star)^2. \tag{42}$$

Also, the $\vec{\alpha}\geq\vec{0}$ constraint is n separate constraints of the form $\alpha_i\geq 0$. Thus, we can solve for each α_i separately as

$$\alpha_i^\star \in \operatorname*{argmin}_{\alpha_i\geq 0} (\alpha_i+\beta_i^\star)^2. \tag{43}$$

This problem is convex and so we can solve it by checking the critical points.

- The gradient (w.r.t. α_i) is 0 if and only if $\alpha_i=-\beta_i^\star$. If $\beta_i^\star>0$ then this solution is infeasible, and if $\beta_i^\star=0$ then $\alpha_i=0$.
- The constraint boundary is $\alpha_i=0$; this solution is feasible with objective value $(\beta_i^\star)^2$.
- The limit $\alpha_i\to+\infty$ makes the objective value arbitrarily large, much larger than $(\beta_i^\star)^2$. The limit $\alpha_i\to-\infty$ makes the solution infeasible.

Thus the optimal solution for each scalar problem is $\alpha_i^\star=0$. Thus $\vec{\alpha}^\star=\vec{0}$.

(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 $\beta^\star$.

Solution: We have that $\vec{w}^\star=\sum_{i=1}^n \beta_i^\star\vec{z}_i$. If $\beta_i^\star>0$ then $\vec{z}_i$ contributes to $\vec{w}^\star$, so $(\vec{x}_i,y_i)$ is a support vector. Otherwise $\beta_i^\star=0$, then $\vec{z}_i$ does not contribute to $\vec{w}^\star$, so $(\vec{x}_i,y_i)$ is not a support vector.

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

$$\min_{\vec{w}} \|X\vec{w} - \vec{y}\|_2^2 + \lambda \|\vec{w}\|_2^2, \tag{44}$$

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.

Solution: We have that the optimal $\vec{w}$ from solving ridge regression is given by

$$\vec{w}^* = (X^\top X + \lambda I)^{-1} X^\top \vec{y} \tag{45}$$

Hence given a new data point x_{test}, we look at $x_{\text{test}}^\top \vec{w}^*$ and if it is positive, we say $y_{\text{test}} = 1$ and otherwise we say $y_{\text{test}} = -1$. In other words, we can say $y_{\text{test}} = \text{sign}(x_{\text{test}}^\top \vec{w}^*)$.

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

Solution: See Jupyter notebook for coding solution.

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.

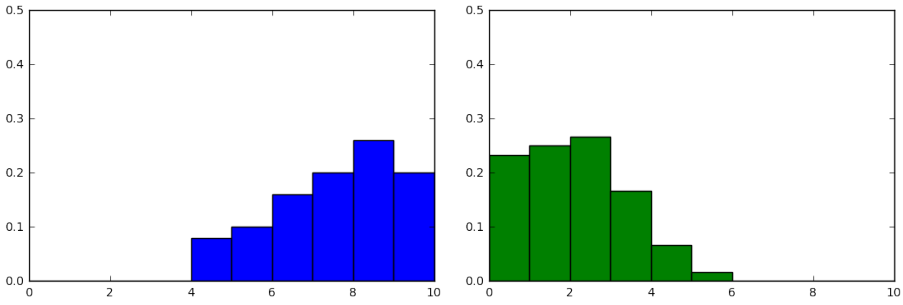


Figure 1: Visualization of μ histogram on left and ν histogram on right.

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:

$$M\vec{1} = \vec{\mu} \tag{46}$$

$$M^\top \vec{1} = \vec{\nu}, \tag{47}$$

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?

Solution: The total cost by following the transportation plan is $\sum_{i,j} c_{ij} m_{ij} = \text{Tr}(C^T M) = \langle C, M \rangle_F$ where $\langle C, M \rangle_F$ is the Frobenius inner product.

(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, $\cdots$?).

Solution: To find $M^\star$ that incurs minimal total cost given the fixed cost matrix C, we can formulate a minimization problem over M. The objective of the problem is the total cost we derived in part (a), $\langle C, M \rangle_F$. This minimization problem is also subject to the following constraints from the definition of M:

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

$$M^\top \vec{1} = \nu$$
$$M \geq 0$$

So, finially we piece everything together and write the minimization problem as:

$$\begin{aligned} M^\star = \underset{M}{\text{argmin}} \quad & \langle C, M \rangle_F \\ \text{s.t.} \quad & M\vec{1} = \mu \\ & M^\top \vec{1} = \nu \\ & M \geq 0 \end{aligned}$$

Since both the objective and the constraints are affine in M, this optimization problem is an LP.

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.

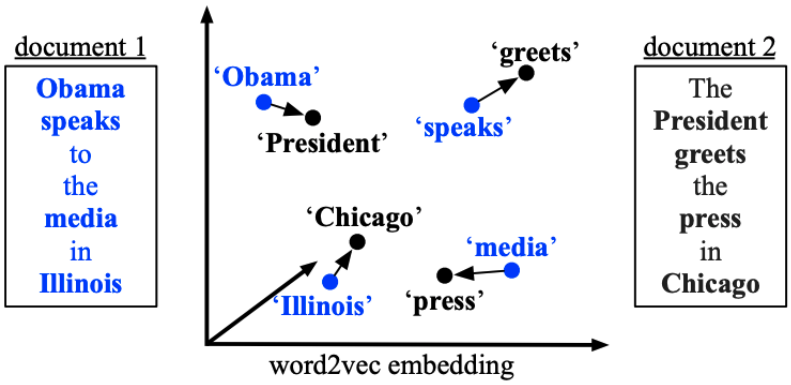
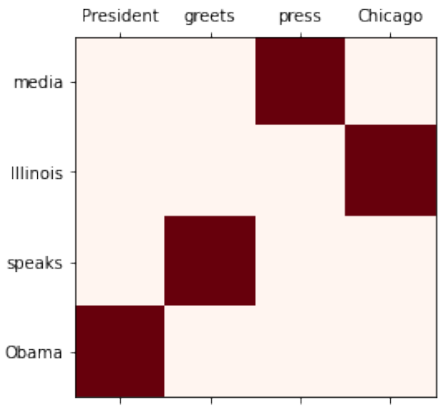


Figure 2: An illustration of the Wasserstein distance. All non-stop words (**bold**) of both documents are embedded into a *word embedding* space. The similarity between the two documents is the minimum cumulative distance that all words in document 1 need to travel to exactly match document 2.

(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.

Solution: Formulating and solving the problem with cvxpy produces the following plot. We see that the matrix M agrees with our intuition and "moves" words to semantically similar words.



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.