CSCI 567, Spring 2021
Sirisha Rambhatla

Homework #1

Released: Jan 22, 2021
Due: 11:59 pm, Feb 5, 2021
Total scores: 40 points

Instructions

Notes on notation:

- Unless stated otherwise, scalars are denoted by small letter in normal font, vectors are denoted by small letters in bold font and matrices are denoted by capital letters in bold font.
- $\|.\|$ means L2-norm unless specified otherwise i.e. $\|.\| = \|.\|_2$

Problem 1 Nearest Neighbor Classification (10 points)

In the class, we use the **Euclidean distance** as the distance metric for the nearest neighbor classification. Given data $\boldsymbol{x} \in \mathbb{R}^D$, the square of Euclidean distance between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ is defined as following:

$$E(\boldsymbol{x}_i, \boldsymbol{x}_j) = \left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\|_2^2 = \sum_{d=1}^{D} (x_{id} - x_{jd})^2 \tag{1}$$

In some applications such as information retrieval and neural language processing, the **cosine distance function**[1] is widely applied. It is defined as:

$$C(\boldsymbol{x}_i, \boldsymbol{x}_j) = 1 - \frac{\boldsymbol{x}_i^T \boldsymbol{x}_j}{\|\boldsymbol{x}_i\|_2 \|\boldsymbol{x}_j\|_2} = 1 - \frac{\sum_{d=1}^{D} (x_{id} \cdot x_{jd})}{\|\boldsymbol{x}_i\|_2 \|\boldsymbol{x}_j\|_2}, \tag{2}$$

where the norm of $\boldsymbol{x}$ is defined as

$$\|\boldsymbol{x}\|_2 = \sqrt{\sum_{d=1}^{D} x_d^2}. \tag{3}$$

Now you are asked to prove that for any $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ normalized to the unit norm, *i.e.* $\|\boldsymbol{x}_i\|_2 = \|\boldsymbol{x}_j\|_2 = 1$, changing the distance measure from the square of Euclidean distance to the cosine distance function will NOT affect the nearest neighbor classification results. Specifically, for any $\boldsymbol{x}_i, \boldsymbol{x}_j$ and $\boldsymbol{x}_o$, show that, if $C(\boldsymbol{x}_i, \boldsymbol{x}_j) \leq C(\boldsymbol{x}_i, \boldsymbol{x}_o)$, then $E(\boldsymbol{x}_i, \boldsymbol{x}_j) \leq E(\boldsymbol{x}_i, \boldsymbol{x}_o)$, where $\|\boldsymbol{x}_i\|_2 = \|\boldsymbol{x}_j\|_2 = \|\boldsymbol{x}_o\|_2 = 1$.

Problem 2 Linear Regression (15 points)

Review In the lectures, we have described the least mean square solution for linear regression as

$$\boldsymbol{w}^* = (\tilde{\boldsymbol{X}}^{\mathrm{T}} \tilde{\boldsymbol{X}})^{-1} \tilde{\boldsymbol{X}}^{\mathrm{T}} \boldsymbol{y} \tag{4}$$

where $\tilde{\boldsymbol{X}}$ is the design matrix (N rows, $D+1$ columns) and $\boldsymbol{y}$ is the N-dimensional column vector of the true values in the training data $\mathcal{D} = \{(\boldsymbol{x}_n, y_n)\}_{n=1}^N$.

[1] note that C is not a real distance metric since it does not satisfy the triangle inequality

2.1 We mentioned a practical challenge for linear regression: when $\tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{X}}$ is not invertible. Please use a concise mathematical statement (*in one sentence*) to summarize the relationship between the training data $\tilde{\boldsymbol{X}}$ and the dimensionality of $\boldsymbol{w}$ when this bad scenario happens. Then use this statement to explain why this scenario must happen when $N < D + 1$. **(6 points)**

2.2 In this problem we use the notation $w_0 + \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}$ for the linear model, that is, we do not append the constant feature 1 to x.

Under certain assumption, the bias w_0 has a solution being the mean of the samples

$$w_0^* = \frac{1}{N}\mathbf{1}_N^{\mathrm{T}}\boldsymbol{y} = \frac{1}{N}\sum_n y_n, \tag{5}$$

where $\mathbf{1}_N = [1, 1, \ldots, 1]^{\mathrm{T}}$ is an N-dimensional column vector whose entries are all ones.

We proved that it is true when $D = 0$ (i.e. ignore the features such that the design matrix is a column of 1's), by the following procedure:

$$w_0^* = \arg\min_{w_0} \|\boldsymbol{y} - w_0\mathbf{1}_N\|^2 \qquad \text{Residual sum of squares} \tag{6}$$

$$\mathbf{1}_N^{\mathrm{T}}(\boldsymbol{y} - w_0^*\mathbf{1}_N) = 0 \qquad \text{Taking derivatives w.r.t } w_0 \tag{7}$$

$$w_0^* = \frac{1}{N}\mathbf{1}_N^{\mathrm{T}}\boldsymbol{y} \tag{8}$$

In this Problem, we would like you to generalize the proof above to arbitrary D and arrive at a more general condition where Eqn. 5 holds.

Please follow the three-step recipe: 1) write out the residual sum of squares objective function; 2) take derivatives w.r.t. the variable you are interested in and set the gradient to 0; 3) solve w_0^* and conclude that Eqn. 5 holds if

$$\frac{1}{N}\sum_n x_{nd} = 0, \quad \forall d = 1, 2, \ldots, D, \tag{9}$$

that is, each feature has zero mean. (In fact, centering the input data to be zero mean is a common pre-processing technique used in practice.)

(9 points)

Problem 3 Convergence of Perceptron Algorithm (15 points)

In this problem you need to show that when the two classes are linearly separable, the perceptron algorithm will converge. Specifically, for a binary classification dataset of N data points, where every $\mathbf{x}_i$ has a corresponding label $y_i \in \{-1, 1\}$ and is normalized: $\|\mathbf{x}_i\| = \sqrt{\mathbf{x}_i^T\mathbf{x}_i} = 1$, $\forall i \in \{1, 2, ..., N\}$, the perceptron algorithm proceeds as below:

In other words, weights are updated right after the perceptron makes a mistake (weights remain unchanged if the perceptron makes no mistakes). Let the (classification) margin for a hyperplane $\mathbf{w}$ be $\gamma(\mathbf{w}) = \min_{i\in[N]} \frac{|\mathbf{w}^T\mathbf{x}_i|}{\|\mathbf{w}\|}$ (convince yourself that $\gamma(\mathbf{w})$ is the smallest distance of any data point from the hyperplane). Let $\mathbf{w}_{opt}$ be the optimal hyperplane, i.e. it linearly separates the classes with maximum margin. Note that since data is linearly separable there will always exist some $\mathbf{w}_{opt}$. Let $\gamma = \gamma(\mathbf{w}_{opt})$.

Following the steps below, you will show that the perceptron algorithm makes a finite number of mistakes that is at most γ^{-2}, and therefore the algorithm must converge.

Algorithm 1 Perceptron

while not converged **do**
  Pick a data point $\mathbf{x}_i$ randomly
  Make a prediction $y = sign(\mathbf{w}^T \mathbf{x}_i)$ using current $\mathbf{w}$
  if $y \neq y_i$ **then**
    $\mathbf{w} \leftarrow \mathbf{w} + y_i \mathbf{x}_i$

3.1 Show that if the algorithm makes a mistake, the update rule moves it towards the direction of the optimal weights $\mathbf{w}_{opt}$. Specifically, denoting explicitly the updating iteration index by k, the current weight vector by $\mathbf{w}_k$, and the updated weight vector by $\mathbf{w}_{k+1}$, show that, if $y_i \mathbf{w}_k^T \mathbf{x}_i < 0$, we have

$$\mathbf{w}_{k+1}^T \mathbf{w}_{opt} \geq \mathbf{w}_k^T \mathbf{w}_{opt} + \gamma \|\mathbf{w}_{opt}\|. \tag{10}$$

Hint: Consider $(\mathbf{w}_{k+1} - \mathbf{w}_k)^T \mathbf{w}_{opt}$ *and consider the property of* $\mathbf{w}_{opt}$. **(4 points)**

3.2 Show that the length of updated weights does not increase by a large amount. Mathematically show that, if $y_i \mathbf{w}_k^T \mathbf{x}_i < 0$

$$\|\mathbf{w}_{k+1}\|^2 \leq \|\mathbf{w}_k\|^2 + 1. \tag{11}$$

Hint: Consider $\|\mathbf{w}_{k+1}\|^2$ *and substitute* $\mathbf{w}_{k+1}$. **(4 points)**

3.3 Assume that the initial weight vector $\mathbf{w}_0 = \mathbf{0}$ (an all-zero vector). Using results from Problem 3.1 and 3.2, show that for any iteration $k+1$, with M being the total number of mistakes the algorithm has made for the first k iterations, we have

$$\gamma M \leq \|\mathbf{w}_{k+1}\| \leq \sqrt{M} \tag{12}$$

Hint: use Cauchy-Schwartz inequality $\mathbf{a}^T \mathbf{b} \leq \|\mathbf{a}\| \|\mathbf{b}\|$ *and telescopic sum.* **(6 points)**

3.4 Using result of Problem 3.3, conclude $M \leq \gamma^{-2}$. **(1 points)**