CSCI 567 Fall 2022, Quiz 1

Instructor: Vatsal Sharan

Time limit: 2 hour and 20 minutes

Problem	1	2	3	4	5	6	7	Total
Max	10	10	10	10	15	12	33	100

Please read the following instructions carefully:

- Please go ahead and fill in your name and Student ID in the space above.
- The exam has 17 pages (including this cover page), numbered 1-17. Once you are permitted to open your exam (and not before), you should make sure you are not missing any pages.
- Answers should be concise and written down legibly. You must answer each question on the space provided below the question. You likely won't need all the provided space to answer most of the questions.
- There are 5 pages provided for scratch work at the end. These pages will not be graded.
- This is a closed-book/notes exam. Consulting any resource is NOT permitted. Do not use your phones during the exam.
- Any kind of cheating will lead to score 0 for the entire exam and be reported to SJACS.

1 Weighted linear regression (10 points)

Consider a modification of the standard linear regression setup where each datapoint is associated with an importance weight. Formally, given a dataset of n datapoints $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n) \in \mathbb{R}^d \times \mathbb{R}$, each datapoint is associated with an importance weight $r_i > 0$. The weighted residual sum of squares objective is defined as,

$$\text{WRSS}(\boldsymbol{w}) = \sum_{i=1}^{n} r_i \left(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_i - y_i\right)^2.$$

(a) Let $\boldsymbol{X}$ be the $n \times d$ matrix whose i-th row is $\boldsymbol{x}_i^{\mathrm{T}}$, $\boldsymbol{y}$ be the n-dimensional column vector whose i-th entry is y_i and $\boldsymbol{R}$ be the diagonal matrix where $\boldsymbol{R}_{ii} = r_i$ for all i and 0 for all other entries. Show that the WRSS objective can be written as follows in matrix form, (3 points)

$$\text{WRSS}(\boldsymbol{w}) = (\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})^{\mathrm{T}} \boldsymbol{R} (\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}). \tag{1}$$

Since the i-th row of $\boldsymbol{X}$ is $\boldsymbol{x}_i^{\mathrm{T}}$, $(\boldsymbol{X}\boldsymbol{w})_i = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i$.
As $\boldsymbol{R}$ is a diagonal matrix with $\boldsymbol{R}_{ii} = r_i$, $(\boldsymbol{R}(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}))_i = r_i(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})_i$.
Using these, we get:

$$\begin{aligned}
\text{WRSS}(\boldsymbol{w}) &= \sum_{i=1}^{n} r_i \left(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_i - y_i\right)^2 \\
&= \sum_{i=1}^{n} \left(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_i - y_i\right) r_i \left(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_i - y_i\right) \\
&= \sum_{i=1}^{n} (\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})_i (\boldsymbol{R}(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}))_i \\
&= (\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})^{\mathrm{T}} \boldsymbol{R} (\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}).
\end{aligned}$$

(b) Solve for the closed-form solution $\boldsymbol{w}^*$ which minimizes Eq 1 (assuming invertibility of any matrices as needed). (7 points)

Hint: You might find it helpful to rewrite WRSS($\boldsymbol{w}$) in the form WRSS($\boldsymbol{w}$) = $\boldsymbol{w}^T\boldsymbol{A}\boldsymbol{w} - 2\boldsymbol{b}^T\boldsymbol{w} + \boldsymbol{y}^T\boldsymbol{R}\boldsymbol{y}$, for some matrix $\boldsymbol{A}$ and some vector $\boldsymbol{b}$ (which you would need to find)

$$\begin{aligned}\text{WRSS}(\boldsymbol{w}) &= (\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})^{\mathrm{T}}\boldsymbol{R}(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}) \\ &= \boldsymbol{w}^{\mathrm{T}}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{R}\boldsymbol{X}\boldsymbol{w} - 2\boldsymbol{y}^{\mathrm{T}}\boldsymbol{R}\boldsymbol{X}\boldsymbol{w} + \boldsymbol{y}^{\mathrm{T}}\boldsymbol{R}\boldsymbol{y}\end{aligned}$$

As $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{R}\boldsymbol{X}$ is symmetric,

$$\nabla\text{WRSS}(\boldsymbol{w}) = 2\boldsymbol{X}^{\mathrm{T}}\boldsymbol{R}\boldsymbol{X}\boldsymbol{w} - 2\boldsymbol{X}^{\mathrm{T}}\boldsymbol{R}\boldsymbol{y}$$

As $r_i > 0$, WRSS($\boldsymbol{w}$) is a convex function of $\boldsymbol{w}$. To get its minimizer $\boldsymbol{w}^*$, we can solve for ∇WRSS($\boldsymbol{w}$) = 0, which gives:

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

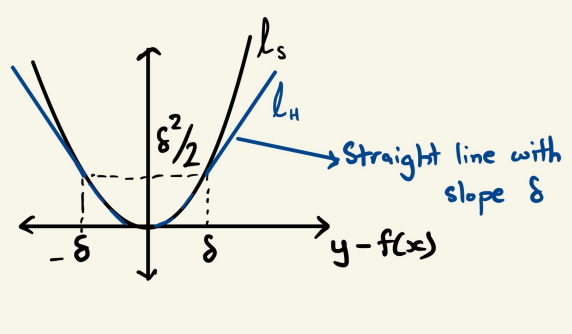
2 Huber loss (10 points)

In this question, we will consider an alternative to the squared loss for regression problems. Consider the following loss function, known as the *Huber loss*,

$$\ell_H(f(\boldsymbol{x}), y) = \begin{cases} \frac{1}{2}(y - f(\boldsymbol{x}))^2 & |y - f(\boldsymbol{x})| \leq \delta, \\ \delta\left(|y - f(\boldsymbol{x})| - (\delta/2)\right) & \text{otherwise.} \end{cases}$$

Let $\ell_S(f(\boldsymbol{x}), y) = \frac{1}{2}(y - f(\boldsymbol{x}))^2$ be the squared loss.

(a) Plot $\ell_H(f(\boldsymbol{x}), y)$ and $\ell_S(f(\boldsymbol{x}), y)$ (on the same plot) as a function of $y - f(\boldsymbol{x})$. (3 points)



(b) Consider a training datapoint $(\boldsymbol{x}, y)$ for which $y - f(\boldsymbol{x})$ is much larger than δ for some model f. Compare the value of $\ell_H(f(\boldsymbol{x}), y)$ and $\ell_S(f(\boldsymbol{x}), y)$ for such a training datapoint. Based on this, what can you say about how sensitive the Huber loss is to the presence of outliers in the data, compared to the squared loss? (2 points)

Let $y - f(\boldsymbol{x}) = \epsilon$ which we assume is much larger than δ. Then

$$\ell_S(f(\boldsymbol{x}), y) = \frac{1}{2}\epsilon^2$$
$$\ell_H(f(\boldsymbol{x}), y) = \delta(\epsilon - \delta/2) \approx \delta\epsilon$$

For a large error ϵ, we see that ℓ_H grows linearly with ϵ while ℓ_S is quadratic in ϵ. Thus, ℓ_H is much lower than ℓ_S for large ϵ. Thus, Huber loss is less sensitive to outliers (which have unusually large errors) than the squared loss.

(c) Given a dataset $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n) \in \mathbb{R}^d \times \mathbb{R}$, we will use the Huber loss to learn a linear model $\boldsymbol{w} \in \mathbb{R}^d$. To do this, we can minimize the empirical risk given by $F(\boldsymbol{w})$,

$$F(\boldsymbol{w}) = \frac{1}{n}\sum_{i=1}^{n} \ell_H(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i, y_i), \tag{2}$$

For a fixed i, write down the gradient $\nabla\ell_H(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i, y_i)$ of $\ell_H(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i, y_i)$ with respect to $\boldsymbol{w}$ (show your derivation), then fill in the missing details in the while-loop of the algorithm below which applies SGD to minimize F. (5 points)

Algorithm 1: SGD for minimizing Eq. (2)

Input: A training set $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n) \in \mathbb{R}^d \times \mathbb{R}$, learning rate $\eta > 0$, number of iterations T

Initialization: $\boldsymbol{w} = \boldsymbol{0}$;

while T *iterations are not complete* **do**

  randomly pick an example $(\boldsymbol{x}_i, y_i)$

  compute $\epsilon = y_i - \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_i$

  if $|\epsilon| \leq \delta$ **then**

    update $\boldsymbol{w} \leftarrow \boldsymbol{w} + \eta\, \epsilon\, \boldsymbol{x}_i$

  else

    update $\boldsymbol{w} \leftarrow \boldsymbol{w} + \eta\, \delta\, \textsc{sign}(\epsilon)\, \boldsymbol{x}_i$

  end

end

We consider two cases. When $|y_i - \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_i| \leq \delta$,

$$\begin{aligned}
\nabla \ell_H(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_i, y_i) &= \nabla \left(\frac{1}{2} (y_i - \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_i)^2 \right) \\
&= \frac{1}{2} \times 2 (y_i - \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_i) \times \nabla \Big((y_i - \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_i) \Big) \\
&= -(y_i - \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_i) \boldsymbol{x}_i
\end{aligned}$$

When $|y_i - \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_i| > \delta$,

$$\begin{aligned}
\nabla \ell_H(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_i, y_i) &= \nabla \Big(\delta \left(|y_i - \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_i| - (\delta/2) \right) \Big) \\
&= \delta \nabla \Big(|y_i - \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_i| \Big) \\
&= \delta\, \textsc{sign}(y_i - \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_i) \Big(- \boldsymbol{x}_i \Big) \\
&= -\delta\, \textsc{sign}(y_i - \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_i) \boldsymbol{x}_i
\end{aligned}$$

$$\nabla \ell_H(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_i, y) = \begin{cases} -(y_i - \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_i) \boldsymbol{x}_i & |y_i - \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_i| \leq \delta, \\ -\delta\, \textsc{sign}(y_i - \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_i) \boldsymbol{x}_i & \text{otherwise.} \end{cases}$$

3 Perceptron algorithm (10 points)

We will investigate the perceptron algorithm in this question (the algorithm is reproduced in Algorithm 2). The perceptron algorithm gets access to a dataset of n instances $(\boldsymbol{x}_i, y_i)$, where $\boldsymbol{x}_i \in \mathbb{R}^d$ and $y_i \in \{-1, 1\}$. It outputs a linear classifier $y = \text{SIGN}(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x})$. Assume $\boldsymbol{x}_i \neq \mathbf{0}\ \forall\ i \in \{1, \ldots, n\}$.

Algorithm 2: Perceptron

Input: A training set $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n) \in \mathbb{R}^d \times \mathbb{R}$, number of iterations T
Initialize $\boldsymbol{w}_{(0)} \leftarrow \mathbf{0}$;
for t *in* $\{0, \ldots, T-1\}$ **do**
  Pick a data point $(\boldsymbol{x}_i, y_i)$ randomly
  Make a prediction $\hat{y} = \text{SIGN}({\boldsymbol{w}_{(t)}}^T \boldsymbol{x}_i)$ using $\boldsymbol{w}_{(t)}$
  if $\hat{y} \neq y_i$ **then**
    $\boldsymbol{w}_{(t+1)} \leftarrow \boldsymbol{w}_{(t)} + y_i \boldsymbol{x}_i$
  else
    $\boldsymbol{w}_{(t+1)} \leftarrow \boldsymbol{w}_{(t)}$

As the algorithm proceeds, suppose the same weight vector is seen twice, despite at least one update in between. In particular, suppose there is some j and k where $j < k$ such that $\boldsymbol{w}_{(j)} = \boldsymbol{w}_{(k)}$, and there is at least one ℓ where $j < \ell < k$ such that $\boldsymbol{w}_{(j)} \neq \boldsymbol{w}_{(\ell)}$. We will show that if this happens, then the given dataset is not linearly separable. To prove this, follow the following two steps.

(a) Write down an expression which relates $\boldsymbol{w}_{(j)}$ and $\boldsymbol{w}_{(k)}$. Your expression will involve the datapoints observed in the intermediate iterations. (5 points).

Suppose the data points that trigger updates from $\boldsymbol{w}_{(j)}$ to $\boldsymbol{w}_{(k)}$ are $\boldsymbol{x}_{(1)}, \ldots, \boldsymbol{x}_{(m)}$. Without loss of generality, assume that the labels for $\boldsymbol{x}_{(1)}, \ldots, \boldsymbol{x}_{(q)}$ are all 1, and the labels for $\boldsymbol{x}_{(q+1)}, \ldots, \boldsymbol{x}_{(m)}$ are -1. Then, we have

$$\boldsymbol{w}_{(k)} = \boldsymbol{w}_{(j)} + (\boldsymbol{x}_{(1)} + \ldots + \boldsymbol{x}_{(q)}) - (\boldsymbol{x}_{(q+1)} + \ldots + \boldsymbol{x}_{(m)}),$$

which gives $\boldsymbol{x}_{(1)} + \ldots + \boldsymbol{x}_{(q)} = \boldsymbol{x}_{(q+1)} + \ldots + \boldsymbol{x}_{(m)}$ by $\boldsymbol{w}_{(k)} = \boldsymbol{w}_{(j)}$.

(b) Now suppose there is some linear classifier $\boldsymbol{w}^*$ which classifiers all datapoints perfectly, i.e. $y_i = \text{SIGN}({\boldsymbol{w}^*}^{\text{T}}\boldsymbol{x}_i)$ for all $i \in \{1, \ldots, n\}$. Use your expression from the previous part and the fact that $\boldsymbol{w}_{(j)} = \boldsymbol{w}_{(k)}$ to arrive at some contradiction, hence proving that the dataset cannot be linearly separable if $\boldsymbol{w}_{(j)} = \boldsymbol{w}_{(k)}$. (5 points).

By the definition of $\boldsymbol{w}^*$, we have $\boldsymbol{x}_{(i)}^\top \boldsymbol{w}^* \geq 0$ for $i = 1, \ldots, q$ and $\boldsymbol{x}_{(i)}^\top \boldsymbol{w}^* < 0$ for $i = q+1, \ldots, m$. Therefore, $(\boldsymbol{x}_{(1)} + \ldots + \boldsymbol{x}_{(q)})^\top \boldsymbol{w}^* \geq 0$ and $(\boldsymbol{x}_{(q+1)} + \ldots + \boldsymbol{x}_{(m)})^\top \boldsymbol{w}^* < 0$. This leads to a contradiction by (a).

4 SVM (10 points)

Various kernel functions have been ingeniously devised for many prediction tasks, lending more power to kernel based methods such as SVM. We consider a kernel function defined over *strings* in this problem. A string is just a sequence of charactes, and in this question we will assume that it only consists of upper-case letters. Given any two strings s_1 and s_2, define the kernel

$$k(s_1, s_2) = |\ \{\text{all upper-case letters which appear at least once in both } s_1 \text{ and } s_2\}\ | .$$

Informally, $k(s_1, s_2)$ just counts the number of letters which occur in both the strings ($|\cdot|$ here denotes the cardinality of a set). We will use this kernel for a toy task.

(a) Suppose we have a binary classification dataset with the following $n = 2$ datapoints:

Datapoint	Label
LA	$+1$
NYC	-1

Write down the 2×2 dimensional kernel/Gram matrix for this data. (3 points)

$k(\text{“}LA\text{”}, \text{“}LA\text{”}) = 2, k(\text{“}LA\text{”}, \text{“}NYC\text{”}) = k(\text{“}NYC\text{”}, \text{“}LA\text{”}) = 0, k(\text{“}NYC\text{”}, \text{“}NYC\text{”}) = 3.$

$$K = \begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix}$$

(b) Recall that the dual SVM formulation for separable data $(s_1, y_1), (s_2, y_2)$ is given by,

$$\begin{aligned} \max_{\{\alpha_i\}} \quad & \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} y_i y_j \alpha_i \alpha_j k(s_i, s_j) \\ \text{s.t.} \quad & \sum_i \alpha_i y_i = 0 \quad \text{and} \quad \alpha_i \geq 0, \quad \forall\ i \in \{1, 2\}. \end{aligned}$$

Write down the dual formulation for the dataset in part (1), and solve the dual formulation to find the optimal value for α_1 and α_2. (7 points)

Substituting $k(\cdot, \cdot)$ and y, we have

$$\begin{aligned} \max_{\{\alpha_1, \alpha_2\}} \quad & \alpha_1 + \alpha_2 - \frac{1}{2}(2\alpha_1^2 + 3\alpha_2^2) \\ \text{s.t.} \quad & \alpha_1 - \alpha_2 = 0 \quad \text{and} \quad \alpha_i \geq 0, \quad \forall\ i \in \{1, 2\}. \end{aligned}$$

Rewrite the objective equation with the constraint $\alpha_1 = \alpha_2$ we have,

$$\begin{aligned} & \max_{\{\alpha_1 \in \mathbb{R}_+\}} \quad 2\alpha_1 - \frac{5}{2}\alpha_1^2 \\ \iff & \max_{\{\alpha_1 \in \mathbb{R}_+\}} \quad \frac{2}{5} - (\sqrt{\frac{5}{2}}\alpha_1 - \sqrt{\frac{2}{5}})^2 \\ \iff & \min_{\{\alpha_1 \in \mathbb{R}_+\}} \quad (\sqrt{\frac{5}{2}}\alpha_1 - \sqrt{\frac{2}{5}})^2 \end{aligned}$$

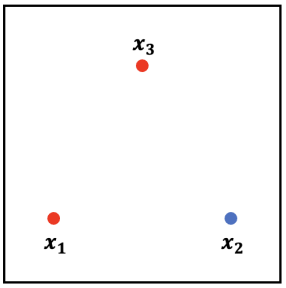
where the optimal value achieves at $\alpha_1 = 2/5$. As $\alpha_1 = \alpha_2$, $\alpha_2 = 2/5$. The solution satisfies all the constraints.

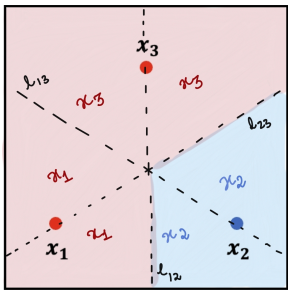
5 Short answer questions (15 points)

5.1 1-NN (3 points)

Consider the two-dimensional binary classification dataset in the figure below. There are three points with labels as follows: $\{(\boldsymbol{x}_1, +1), (\boldsymbol{x}_2, -1), (\boldsymbol{x}_3, +1)\}$ (in the figure, red corresponds to label $+1$ and blue corresponds to -1). If we train a 1-nearest neighbor model on this data using the usual Euclidean distance $d(\boldsymbol{x}, \boldsymbol{x}') = \|\boldsymbol{x} - \boldsymbol{x}'\|_2$ to compute distances, what will be the decision boundary we get? Mark the decision bounary in the figure, and also explain your answer briefly.

Hint: You may find it helpful to first consider the case where only $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are present, and then the case where only $\boldsymbol{x}_2$ and $\boldsymbol{x}_3$ are present.





The lines l_{21}, l_{23} and l_{13} are the bisectors of the line-segments joining x_1x_2, x_2x_3 and x_2x_3 respectively. There are six regions due to the three line segments dividing the space and we label each region to the nearest point x_i. Now we color each region based on the color of the assigned point/label.

5.2 Generalization (4 points)

Recall the generalization theorem that we proved in class:

Theorem 1. *Let $\mathcal{F}$ be a function class with size $|\mathcal{F}|$. Let $y = f^*(\boldsymbol{x})$ for all inputs $\boldsymbol{x}$, for some $f^* \in \mathcal{F}$. Suppose we get a training set $S = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$ of size n with each datapoint drawn i.i.d. from the data distribution D. Let*

$$f_S^{ERM} = \operatorname*{argmin}_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \ell(f(\boldsymbol{x}_i), y_i),$$

where $\ell(f(\boldsymbol{x}_i), y_i)$ denotes the 0-1 loss. For any constants $\epsilon, \delta \in (0, 1)$, if $n \geq \frac{\ln(|\mathcal{F}|/\delta)}{\epsilon}$, then with probability $(1 - \delta)$ over $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$, $R(f_S^{ERM}) < \epsilon$.

Consider the case where the input $\boldsymbol{x}$ is binary valued, i.e. $\boldsymbol{x} \in \{0,1\}^d$. Define the class of *conjunctions on two variables* as follows, $\mathcal{F} = \{f_{i,j}, 1 \leq i, j \leq d\}$ where $f_{i,j}(\boldsymbol{x})$ is defined as the AND function on the i-th and j-th coordinates of input $\boldsymbol{x}$:

$$f_{i,j}(\boldsymbol{x}) = \begin{cases} 1 & \text{if } \boldsymbol{x}(i) = 1 \text{ AND } \boldsymbol{x}(j) = 1 \\ -1 & \text{otherwise.} \end{cases}$$

Here $\boldsymbol{x}(i)$ refers to the i-th coordinate of input $\boldsymbol{x}$.

(a) Suppose we are given a dataset S of n datapoints drawn i.i.d. from some distribution where the labels are given by some conjunction f^* on two variables. If we use empirical risk minimization to learn a conjunction on two variables f_S^{ERM} to fit the data, how large does n have to be to ensure that the predictor we learn has expected classification error $R(f_S^{ERM})$ at most 5% with probability at least 90% over the randomness in the n datapoints?

$\delta = 0.1, \epsilon = 0.05, |\mathcal{F}| = d^2$ (working with $|\mathcal{F}| = \binom{d}{2}$ is also fine)

$\implies n \geq \frac{\ln(d^2/0.1)}{0.05} = 20[2\ln(d) - \ln(0.1)] = 40\ln(d) + 20\ln(10).$

(b) Now, consider the class of conjunctions defined on k variables where the predictor $f_{i_1,i_2,\ldots,i_k}(\boldsymbol{x})$ has k parameters $i_1, \ldots, i_k$ and is defined as follows:

$$f_{i_1,i_2,\ldots,i_k}(\boldsymbol{x}) = \begin{cases} 1 & \text{if } \boldsymbol{x}(i_1) = 1 \text{ AND } \boldsymbol{x}(i_2) = 1 \text{ AND } \ldots \boldsymbol{x}(i_k) = 1 \\ -1 & \text{otherwise.} \end{cases}$$

Roughly, how much training data is needed to learn a conjunction over k variables which generalizes well to test data?

Since the model has about k free parameters (the coordinates $i_1, i_2, \ldots, i_k$) about k datapoints should suffice.

More detailed calculation for the interested:

$\delta = 0.1, \epsilon = 0.05, |\mathcal{F}| = d^k$ (similar to before, working with $|\mathcal{F}| = \binom{d}{k}$ is also fine)

So we need $n \geq \frac{\ln(d^k/0.1)}{0.05} = 20[k\ln(d) - \ln(0.1)] = 20k\ln(d) + 20\ln(10).$

5.3 SVM vs. Logistic regression (4 points)

Let $\boldsymbol{w}_{svm}$ and $\boldsymbol{w}_{log}$ be linear classifiers obtained using the SVM algorithm and logistic regression (respectively) on some training dataset $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$. Consider a new datapoint $(\boldsymbol{x}_{n+1}, y_{n+1})$ which is correctly classifier by both the linear models ($\boldsymbol{w}_{svm}$ and $\boldsymbol{w}_{log}$) and is far away from both their decision boundaries. Suppose I rerun the SVM and logisic regression algorithm on the modified training dataset $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n), (\boldsymbol{x}_{n+1}, y_{n+1})\}$ to get new solutions $\boldsymbol{w}'_{svm}$ and $\boldsymbol{w}'_{log}$ respectively. How would $\boldsymbol{w}'_{svm}$ and $\boldsymbol{w}'_{log}$ change with respect to $\boldsymbol{w}_{svm}$ and $\boldsymbol{w}_{log}$? Explain.

$\boldsymbol{w}'_{svm} = \boldsymbol{w}_{svm}$ but $\boldsymbol{w}'_{log}$ will change slightly compared to $\boldsymbol{w}_{log}$. This is because the SVM objective is unaffected by correctly classified points which are far from the margin, but logistic regression will still put some small weight on such points.

More detailed calculation: We can look at the two loss functions. For SVM, we are minimizing $\max\{0, 1 - y(\boldsymbol{w}^T\boldsymbol{x})\}$. Since the newly introduced data point $(\boldsymbol{x}_{n+1}, y_{n+1})\}$ is correctly predicted by $\boldsymbol{w}_{svm}$ and is far away from the decision boundary, we know that $y_{n+1}(\boldsymbol{w}_{svm}^T\boldsymbol{x}_{n+1})$ should be much larger than 1. Hence, this newly introduced training data won't introduce any additional loss and the original solution will still be optimal. Therefore, $\boldsymbol{w}'_{svm}$ will be the same as $\boldsymbol{w}_{svm}$.

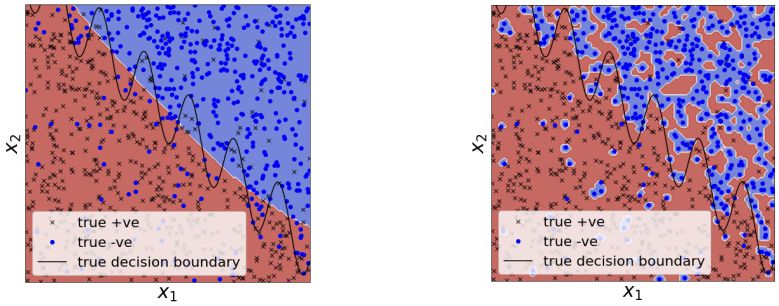
On the other hand, for logistic regression, we are minimizing $\log(1 + \exp(-y(\boldsymbol{w}^T\boldsymbol{x})))$. This is non-zero for the the newly introduced data point $(\boldsymbol{x}_{n+1}, y_{n+1})\}$. Therefore the algorithm will try and incorporate this into the overall objective and will attempt to minimize it.

5.4 RBF kernel (4 points)

Recall the Gaussian kernel (or RBF kernel) from class:

$$k(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\gamma\|\boldsymbol{x} - \boldsymbol{x}'\|_2^2\right) \quad \text{for some } \gamma > 0. \tag{3}$$

The two figures below show the result of training SVM using the RBF kernel on some data, with different values of γ. The true positive and true negative points are marked by black x and blue circles respectively, and the true decision boundary is given by the black line. The decision boundary of the learned classifier is colored, with all points in the red region labeled as positive, and all points in the blue region labeled as negative. Which figure corresponds to a larger value of γ? Explain.



(a) (b)

Figure (b) has a larger value of γ. When γ is small, the contribution from points that are far away from any datpoint $\boldsymbol{x}$, i.e, large $\|\boldsymbol{x}_i - \boldsymbol{x}\|_2^2$, will become similar to points that are close to $\boldsymbol{x}$. Hence, the decision boundary will become more global. In contrast, if γ is large, the decision boundary will become more local, as figure (b) shows.

6 Adventures of a CSCI567 graduate (12 points)

You graduate from CSCI567, and go on to take a position as a machine learning expert in a tech company. The following questions chronicle some of your adventures as you navigate the world of machine learning practice.

6.1 Least squares (4 points)

Your first task is to fit a linear regression model for some customer data. The data has $d = 100$ real-valued attributes, and $n = 50$ datapoints are available.

(a) What could go wrong if you try to naively find the least-squares estimator on the data?

Since we have more dimensions than datapoints, the data matrix $\boldsymbol{X} \in \mathbb{R}^{n \times d}$ is not invertible and we cannot apply the least squares solution $(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X})^{-1}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{y}$. In this case, there are in fact infinitely many solutions $\boldsymbol{w}$ which fit the data perfectly, i.e. they satisfy $\boldsymbol{X}\boldsymbol{w} = \boldsymbol{y}$.

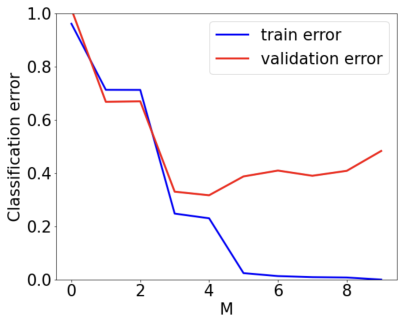
(b) How can you rectify the issue you identified in the previous question?

We can apply ℓ_2 regularization (or ℓ_1 is an option too if we want a sparse solution). The ℓ_2 regularization solution $(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X} + \lambda\boldsymbol{I})^{-1}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{y}$ is well-defined even if $\boldsymbol{X}$ is not invertible, and picks the solution with minimum norm among the infinitely many solutions $\boldsymbol{w}$ which satisfy $\boldsymbol{X}\boldsymbol{w} = \boldsymbol{y}$. (Other answers such as trying to get more data or reducing the dimensionality are also acceptable).

6.2 Training, test and validation (4 points)

Your next challenge is a binary classification task for another client. You have collected a dataset with 10,000 labelled datapoints. You divide the data into 7000 training datapoints, 1500 validation datapoints and 1500 test datapoints.

(a) Your model has a hyperparameter M which you need to tune. Based on the plot below, what would be a good choice for M?



$M = 3$ or $M = 4$ since the validation error is the smallest for these values.

(b) After tuning all hyperparameters you obtain a model which gets 90% accuracy on test data, and send the results to your team. One of your colleagues (who unfortunately didn't take CSCI567) comes back to you and says, *"Why did you only train on 7000 datapoints? See, here I trained this fancy model on all the 10,000 datapoints and have obtained 95% accuracy on the datapoints. Let's send this model to the client instead!"*. How would you explain to your colleague that their methodology is incorrect?

In ML, our end goal is always to do well on new test data. Since the colleague's model has seen all the datapoints, we can't evaluate if the model has really learned to predict well on new data or has simply memorized the training datapoints. A 90% accuracy on the test set of size 1500 says that the model can generalize well, but in the case of the colleague who trained on the whole dataset there is no way to test the model on unseen datapoints to test generalization of the learned model, since there are no more unseen datapoints left.

6.3 Generalization (4 points)

Based on your previous successes, you are now working with a team of computation biologists who are trying to build a model to predict a phenotype of an individual, such as their risk of diabetes, from their gene sequence. A dataset of the gene sequence of n individuals has been compiled for this purpose. For each individual in the dataset, you have their DNA sequenced at $d = 10^6$ locations. The biologists in your team feel that mutations at certain *pairs* of DNA locations should together be responsible for the phenotype, hence you want to experiment with a linear classifier with the genes at pairs of DNA locations as the features.[1]

(a) You want to ensure that when the linear classifier you train is applied to other individuals from the same populations, it should get an error rate within some small constant of the error rate it obtained on your training data. Roughly how large should the training data be to ensure this is the case? (only a rough, order of magnitude answer with a brief explanation is needed)

Hint: What is the dimensionality of the input to the linear classifier?

Since the classifier works on genes on pairs of DNA locations as input, the input is $(10^6)^2 = 10^{12}$ dimensional. Therefore the linear classifier has around 10^{12} free parameters, and we need around that many datapoints.

(b) As sequencing these many individuals is too expensive, how can you still train your model while ensuring that it does not overfit? If you know that the ground truth depends on only $k \approx 10^3$ gene pairs, then roughly how many samples do you need to learn a good model? (only a rough, order of magnitude answer with a brief explanation is needed)

We can add ℓ_1 regularization to encourage sparsity. If the true solution is k-sparse, then roughly $O(k)$ datapoints suffice (since a k-sparse linear model has about $O(k)$ free parameters, ignoring logarithmic terms), so around 10^3 datapoints could suffice.

[1] If it helps your reasoning, you can think of the weights of the linear classifier as belonging to $\{-1, 0, 1\}$ for the sake of this problem, even though it actually doesn't matter.

7 Multiple choice questions (33 points)

IMPORTANT: *Select ALL options among {A,B,C,D} that you think are correct (no justification needed). You get 0.5 point for selecting each correct option and similarly 0.5 point for not selecting each incorrect option. You get 1 additional point for selecting all four options correctly.*

(1) Which of the following statements are true?

(A) In supervised learning, we assume that we are provided the desired output labels for each datapoint.
(B) A classifier that attains 100% accuracy on the training set and 70% accuracy on the test set is better than a classifier that attains 70% accuracy on the training set and 75% accuracy on the test set.
(C) A model which has high training error and high test error is said to be underfitting.
(D) It is not advisable to use the test set to tune hyperparameters of our machine learning model.

Answer: ACD.

(2) Let $\boldsymbol{X} \in \mathbb{R}^{n \times d}$ be a data matrix with each row corresponding to the features of an example and $\boldsymbol{y} \in \mathbb{R}^n$ be a vector of all the outcomes. Which of the following statements are true about linear regression (minimizing $F(\boldsymbol{w}) = \|\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}\|_2^2$)?
(A) If n is much larger than d, then the least squares solution $(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X})^{-1}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{y}$ should have small gap between training and test accuracy.
(B) When $d = 1$ and $\boldsymbol{X}$ is the all ones vector ($\boldsymbol{X} = [1, 1, \ldots, 1]^{\mathrm{T}}$), the optimal value w^* for w is the average of the outcomes $\boldsymbol{y}$, i.e. $w^* = (1/n)\sum_{i=1}^n y_i$.
(C) When $n < d$, the problem of minimizing $F(\boldsymbol{w})$ is non-convex.
(D) If the step size is too large, then gradient descent on $F(\boldsymbol{w})$ may never converge.

Answer: ABD. A is true since the generalization gap goes down if we have a lot of data (as also seen for this problem in HW1). C is not true since the problem is always convex. D is true, as also seen in HW1.

(3) Which of the following statements are true about k-nearest neighbors (k-NN)?
(A) k-NN always gives a linear decision boundary.
(B) 50-NN is more likely to overfit the data compared to 1-NN.
(C) Using different distance metrics (such as ℓ_2 distance, ℓ_1 distance, cosine similarity etc.) never affects the decision boundary of 1-NN.
(D) k-NN is similar to kernel methods in the sense that both of them may require us to store the entire training data to make predictions on the test set.

Answer: D. B is false since 50-NN considers 50 neighbors and is more robust to overfitting.

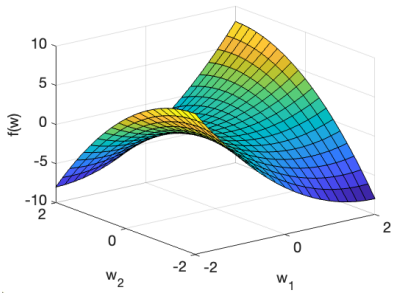
(4) Which of the following statements are true about optimization algorithms?

(A) Gradient descent (GD) will always converge to a local or a global minimizer of the function being optimized.
(B) The SVM objective can be solved by stochastic gradient descent (SGD).
(C) One iteration of Newton's method is much faster to execute than one iteration of GD.
(D) One iteration of SGD is much faster to execute than one iteration of GD.

Answer: BD. A is false since GD may also converge to a saddle point.

(5) Consider the two-dimensional function $f(\boldsymbol{w}) : \mathbb{R}^2 \to \mathbb{R}$ defined by $f(\boldsymbol{w}) = w_1^2 - w_2^2 + 2w_1w_2$ (here $\boldsymbol{w} = (w_1, w_2)$). A plot is provided below on the domain $[-2, 2] \times [-2, 2]$. Which of the following statements are true about the function?

(A) The function is non-convex.
(B) $(0, 0)$ is the only stationary point of the function.
(C) $(0, 0)$ is a local minima of the function.
(D) Gradient descent always converges to $(0, 0)$, regardless of initialization.



Answer: AB. (0,0) is a saddle point, but GD will not converge there for most initializations.

(6) Which of the following statements are true about Maximum likelihood estimation (MLE)?

(A) To do MLE of some parameter $\boldsymbol{w}$, we need to first write a probablistic model $\mathbb{P}[y|\boldsymbol{x}; \boldsymbol{w}]$ which specifies how the label y of a datapoint $\boldsymbol{x}$ is generated based on $\boldsymbol{w}$.
(B) If we have a training set $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n) \in \mathbb{R}^d \times \{-1, +1\}$, where each outcome y_i is generated by a probabilistic model $\mathbb{P}[y_i = 1|\boldsymbol{x}_i; \boldsymbol{w}] = \sigma(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i)$ where $\sigma(\cdot)$ is the sigmoid function, then MLE for $\boldsymbol{w}$ is equivalent to empirical risk minimization on the logistic loss.
(C) If we have a training set $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n) \in \mathbb{R}^d \times \mathbb{R}$, where each outcome y_i is generated by a probabilistic model $y_i = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + \epsilon_i$ with ϵ_i being an independent Gaussian noise with zero-mean, then MLE for $\boldsymbol{w}$ is equivalent to least squares estimation.
(D) Maximum a posteriori probability (MAP) estimation is an extension of MLE where we assume a prior over $\boldsymbol{w}$.

Answer: ABCD

(7) Consider a spam classification task where the inputs $\boldsymbol{x}$ are emails, and label $y = 1$ denotes *spam* and $y = -1$ denotes *not spam*. Suppose you have trained a model which can recover $\mathbb{P}[y = 1|\boldsymbol{x}]$ for all inputs $\boldsymbol{x}$. You now want to convert the probabilities to $\{-1, +1\}$ predictions. To do this you choose some threhold T, and predict label $+1$ if $\mathbb{P}[y = 1|\boldsymbol{x}] \geq T$, and predict -1 otherwise. Which of the following statements are true about the threshold T?

(A) T should always lie in the interval $[0, 1]$.
(B) If $T = 0.01$, then it is likely that there will be a lot of emails classified as *spam*.
(C) As T is increased, more and more emails which are actually *not spam* are classified as *spam*.
(D) If the goal is to learn a classifier such that very few emails which are actually *not spam* are classified as *spam*, then it is better to choose $T = 0.9$ as compared to choosing $T = 0.1$.

Answer: ABD. A is correct because $\mathbb{P}[y = 1|\boldsymbol{x}]$ always lies in $[0, 1]$. B is correct because when T is small, there might be a lot of x satisfying $\mathbb{P}[y = 1|\boldsymbol{x}] \geq T$. C is incorrect because as T increases, it is harder for an email to be classified as *spam*. D is correct because with a high threshold $T = 0.9$, we only classify an email as *spam* when we are very confident.

(8) Which of the following statements are true about generalization?

(A) A model which always makes the same prediction on any input datapoint will have small difference in accuracy between training and test data.
(B) The gap between training and test accuracies will generally *increase* as the size of the training dataset *increases*.
(C) To measure the expected risk of a machine learning model, we usually estimate the risk on new datapoints drawn i.i.d. from the data distribution.
(D) If the i.i.d. assumption is not valid, then our model may not perform well on new unseen examples when deployed in the real world.

Answer: ACD. A is correct because training and test data are i.i.d samples from the same distribution. B is incorrect because the gap should reduce: when the training set is small, the model overfits and has a large gap between train and test. As the size of dataset increases, the gap shrinks. C is correct by the definition of expected risk. D is correct because our model is designed for the i.i.d. assumption.

(9) Which of the following statements are true about bias and variance of ML models?

(A) Bias and variance are terms used in the context of underfitting and overfitting.
(B) If a model has large variance, then its performance will not improve even if we add a lot of training data.
(C) If a model has large bias, then its performance will not improve even if we add a lot of training data.
(D) Choosing a complicated, non-linear mapping on the input features can increase the bias of a logistic regression model trained on the data, but will reduce its variance.

Answer: AC. B is incorrect because high variance means the model complexity is large, and adding more training data increases the performance. C is correct because large bias means the model complexity is small, and adding more training data does not improve the performance. D is incorrect because a complicated mapping increases the variance and reduces the bias.

(10) Which of the following are **NOT** valid kernel functions?

(A) $k(x, x') = \cos(x - x') = \cos(x)\cos(x') + \sin(x)\sin(x')$ (defined on univariate inputs x, x').
(B) $k(x, x') = (x - x')^2$ (defined on univariate inputs x, x').
(C) $k(x, x') = \mathbf{1}(|x - x'| \geq 1)$ (defined on univariate inputs x, x', $\mathbf{1}(\cdot)$ is the indicator function which is 1 if the input is true, 0 otherwise).
(D) $k(x, x') = \ln(xx')$ (defined on univariate inputs $x, x' > 0$).

Answer: BCD. A is a kernel as the sum of two kernels. BC are not kernels, using $x = 0, x' = 1$ to construct non-PSD Gram matrices. D is not a kernel, using $x = 1, x' = 2$ to construct a non-PSD Gram matrix.

(11) Which of the following statements are true about SVMs?

(A) Supports vectors are always points which are misclassified by the classifier.
(B) The model can overfit if the penalty term C in soft-margin SVM is too large (recall that the primal objective in soft-margin SVM is to minimize $C\sum_i \xi_i + \frac{1}{2}\|\boldsymbol{w}\|_2^2$ subject to slack constraints).
(C) SVM with a linear kernel will always give a linear decision boundary.
(D) SVM is equivalent to minimizing hinge loss with ℓ_2 regularization.

Answer: BCD. For B, larger C means less slack is allowed and hence the model tries to fit the data better. Another way to reason is that $C = 1/\lambda$ in the ℓ_2 regularized hinge loss formulation, and λ too small leads to overfitting.

You can use this page for scratch work

You can use this page for scratch work

You can use this page for scratch work

You can use this page for scratch work

You can use this page for scratch work