CSCI 567 Fall 2018 Midterm Exam
DO NOT OPEN EXAM UNTIL INSTRUCTED TO DO SO
PLEASE TURN OFF ALL CELL PHONES

Problem	1	2	3	4	5	6	Total
Max	16	10	16	42	24	12	120
Points							

Please read the following instructions carefully:

- The exam has a total of **22 pages** (double-sided, including this cover and two blank pages in the end) and **6 questions**. Each question have sub-questions. Once you are permitted to open your exam (and not before), you should check and make sure that you are not missing any pages.
- This is a **closed-book/notes** exam. Consulting any resources is NOT permitted.
- Any kind of cheating will lead to **score 0** for the entire exam and be reported to SJACS.
- Duration of the exam is 2 hours. Questions are not ordered by their difficulty. Budget your time on each question carefully.
- Answers should be **concise** and written down **legibly**. All questions can be done within 5-10 lines.
- You must answer each question on the page provided. You can use the last two blank pages as scratch paper. Raise your hand to ask a proctor for more if needed.
- Select **one and only one answer** for all multiple choice questions.
- You **may not** leave your seat **for any reason** unless you submit your exam at that point.

1 Machine Learning Concepts (16 points)

1.1 Multiple Choice (12 points)

(a) Which of the following is a parametric model?

(A) Nearest neighbor classifier
(B) Support vector machines with a Gaussian kernel
(C) Linear regression with a polynomial kernel
(D) fully connected feedforward neural nets

Ans: D. (2 points)

(b) Which of the following phenomenon is called overfitting?

(A) low training error, low test error
(B) low training error, high test error
(C) high training error, low test error
(D) high training error, high test error

Ans: B. (2 points)

(c) Suppose the conditional probability of seeing label k given $\boldsymbol{x}$ is $\mathbb{P}(y = k \mid \boldsymbol{x})$. Which of the following is the Bayes optimal classifier?

(A) A randomized classifier that predicts label k with probability $\mathbb{P}(y = k \mid \boldsymbol{x})$ given $\boldsymbol{x}$.
(B) A deterministic classifier that predicts $\operatorname{argmax}_k \mathbb{P}(y = k \mid \boldsymbol{x})$ given $\boldsymbol{x}$.
(C) A deterministic classifier that predicts $\operatorname{argmin}_k \mathbb{P}(y = k \mid \boldsymbol{x})$ given $\boldsymbol{x}$.
(D) Both (A) and (B).

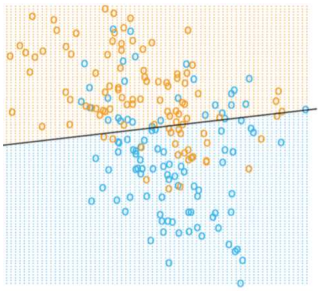
Ans: B. See Lec 1. (2 points)

(d) Which of the following cannot be used as regularization to control model complexity?

(A) $R(\boldsymbol{w}) = \sum_{d=1}^{D} |w_d|$
(B) $R(\boldsymbol{w}) = \sum_{d=1}^{D} |w_d|^3$
(C) $R(\boldsymbol{w}) = \sum_{d=1}^{D} w_d^3$
(D) $R(\boldsymbol{w}) = \sum_{d=1}^{D} |w_d|^4$

C. R is unbounded both above and below. As a result, minimizing or maximizing it will push the objective function to -∞ or ∞. (2 points)

(e) Which of the these classifiers could have generated this decision boundary?



(A) SVM
(B) 1-nearest-neighbor with L2 distance
(C) Logistic regression
(D) A & C

Ans: D. Both are linear classifiers. (2 points)

(f) For a fixed multiclass problem, which of the following multiclass-to-binary reductions has the smallest testing time complexity?

(A) One-versus-all
(B) One-versus-one
(C) Tree reduction
(D) Both (A) and (C)

Ans: C. (2 points)

1.2 Multiclass reduction (4 points)

Show that one-versus-all can be seen as a special case of error-correcting-output-code (ECOC). Specifically, write down the code matrix $\boldsymbol{M}$ for ECOC for a problem with C labels so that executing ECOC is the same as doing one-versus-all. (Note: the entry of $\boldsymbol{M}$ should be either -1 or $+1$.)

$\boldsymbol{M}$ is a $\mathsf{C} \times \mathsf{C}$ matrix with $+1$ on the diagonal and -1 on every other entries, or a $\mathsf{C} \times \mathsf{C}$ matrix with -1 on the diagonal and $+1$ on every other entries. Either one will get full credit.

Only getting the dimension right will get 1 point.

2 Nearest Neighbor Classification (10 points)

2.1 First Problem (4 points)

We mentioned that the Euclidean/L2 distance is often used as the *default* distance for nearest neighbor classification. It is defined as

$$E(\boldsymbol{x}, \boldsymbol{x}') = \|\boldsymbol{x} - \boldsymbol{x}'\|_2^2 = \sum_{d=1}^{D} (x_d - x'_d)^2 \tag{1}$$

In some applications such as information retrieval, the cosine distance is widely used too. It is defined as

$$C(\boldsymbol{x}, \boldsymbol{x}') = 1 - \frac{\boldsymbol{x}^T \boldsymbol{x}'}{\|\boldsymbol{x}\|_2 \|\boldsymbol{x}'\|_2} = 1 - \frac{\sum_{d=1}^{D} (x_d \cdot x'_d)}{\|\boldsymbol{x}\|_2 \|\boldsymbol{x}'\|_2}, \tag{2}$$

where the L2 norm of $\boldsymbol{x}$ is defined as

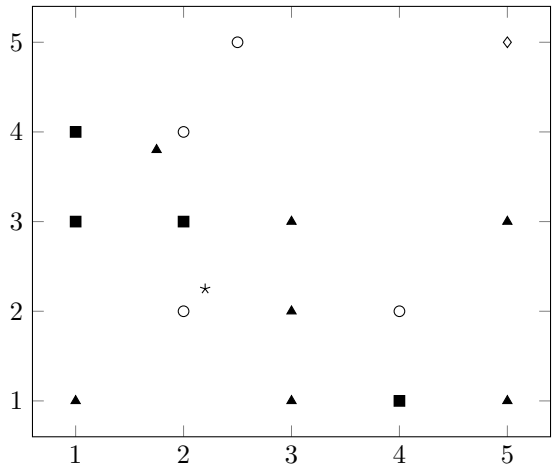
$$\|\boldsymbol{x}\|_2 = \sqrt{\sum_{d=1}^{D} x_d^2}. \tag{3}$$

Show that, if data is normalized with unit L2 norm, that is, $\|\boldsymbol{x}\| = 1$ for all $\boldsymbol{x}$ in the training and test sets, changing the distance function from the Euclidean distance to the cosine distance will NOT affect the nearest neighbor classification results.

When data is normalized, we have $E(\boldsymbol{x}, \boldsymbol{x}') = \sum_{d=1}^{D} (x_d - x'_d)^2 = \sum_{d=1}^{D} (x_d^2 + {x'_d}^2 - 2x_d x'_d) = 2(1 - \sum_{d=1}^{D} (x_d \cdot x'_d)) = 2C(\boldsymbol{x}, \boldsymbol{x}')$. Scaling any distance measurement by a factor of two clearly will not affect the results of NNC. (4 points)

2.2 Second Problem (6 points)

For the data given below, squares, triangles, and open circles are three different classes of data in the training set and the diamond (♢) and star (*) are test points. We denote the total number of training points as N and consider K-nearest-neighbor (KNN) classifier with L2 distance.



(a) When $K = 1$, how many *circles*(o) will be misclassified (as a validation point) when one performs leave-one-out validation (that is, N-fold cross validation)? (A single number as the answer is enough.)

3 (2 points)

(b) What is the minimum value of K for which the *star*(*) will be classified as a *triangle*? (A single number as the answer is enough.)

4 (2 points)

(c) What is the *diamond* classified as for $K = N$? Explain why.

Triangle (1 point)
When $K = N$ the prediction is always the majority label of the training set. Triangle is the majority in this example. (1 point)

3 Linear Regression (16 points)

3.1 First Problem (4 points)

For a training set $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N) \in \mathbb{R}^D \times \mathbb{R}$, let $\boldsymbol{w}_*$ be the least square solution with no regularization (assume $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}$ is invertible where $\boldsymbol{X} \in \mathbb{R}^{N \times D}$ is the data matrix with each row corresponding to the feature of an example, as used in the class). Find the least square solution (with no regularization again) of the following new training set: $(\boldsymbol{x}_1, y_1 - \boldsymbol{w}_*^{\mathrm{T}}\boldsymbol{x}_1), \ldots, (\boldsymbol{x}_N, y_N - \boldsymbol{w}_*^{\mathrm{T}}\boldsymbol{x}_N)$.

Using the formula derived in the class, we know the new least square solution $\boldsymbol{w}'_*$ is

$$\boldsymbol{w}'_* = (\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X})^{-1}\boldsymbol{X}^{\mathrm{T}}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}_*) = (\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X})^{-1}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{y} - (\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X})^{-1}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}((\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X})^{-1}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{y}) = \boldsymbol{0}.$$

Rubrics:

- Write down the correct formula for $\boldsymbol{w}_*$. (1 point)
- Use the correct formula to find $\boldsymbol{w}'_*$ or re-derive in a correct way (e.g. correct gradient). (2 points)
- Get the correct answer finally $\boldsymbol{w}'_* = \boldsymbol{0}$. (1 point)

3.2 Second Problem (12 points)

Assume 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_n is generated by a probabilistic model $\boldsymbol{w}_*^{\mathrm{T}} \boldsymbol{x}_n + \epsilon_n$ with ϵ_n being an independent Gaussian noise with zero-mean and variance σ^2 for some $\sigma > 0$. In other words, the probability of seeing any outcome $y \in \mathbb{R}$ given $\boldsymbol{x}_n$ is

$$\mathbb{P}(y \mid \boldsymbol{x}_n; \boldsymbol{w}_*, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(\frac{-(y - \boldsymbol{w}_*^{\mathrm{T}} \boldsymbol{x}_n)^2}{2\sigma^2}\right).$$

(a) Assume σ is fixed and given, find the maximum likelihood estimation for $\boldsymbol{w}_*$. In other words, first write down the probability of seeing the outcomes $y_1, \ldots, y_N$ given $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ as a function of the value of $\boldsymbol{w}_*$; then find the value of $\boldsymbol{w}_*$ that maximizes this probability. You can assume $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}$ is invertible where $\boldsymbol{X}$ is the data matrix as used in Problem 3.1.

The probability of seeing the outcomes $y_1, \ldots, y_N$ given $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ for a linear model $\boldsymbol{w}$ is

$$\mathcal{P}(\boldsymbol{w}) = \prod_{i=1}^{N} \mathbb{P}(y_n \mid \boldsymbol{x}_n; \boldsymbol{w}, \sigma) = \prod_{i=1}^{N} \frac{1}{\sigma\sqrt{2\pi}} \exp\left(\frac{-(y_n - \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_n)^2}{2\sigma^2}\right).$$

Taking the negative log, this becomes

$$F(\boldsymbol{w}) = N \ln \sqrt{2\pi} + N \ln \sigma + \frac{1}{2\sigma^2} \sum_{n=1}^{N} (y_n - \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_n)^2 = N \ln \sqrt{2\pi} + N \ln \sigma + \frac{1}{2\sigma^2} \|\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}\|_2^2.$$

Maximizing $\mathcal{P}$ is the same as minimizing F, which is clearly the same as just minimizing $\sum_{n=1}^{N} (y_n - \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_n)^2$, the same objective as for least square regression. Therefore the MLE for $\boldsymbol{w}_*$ is exactly the same as the least square solution:

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

Rubrics:

- Write down the correct likelihood function. (2 points)
- Any derivation that reveals this is the same as least square regression. (2 points)
- Arrive at the correct final answer. (2 points)

(b) Now consider σ as a parameter of the probabilistic model too, that is, the model is specified by both $\boldsymbol{w}_*$ and σ. Find the maximum likelihood estimation for $\boldsymbol{w}_*$ and σ.

From previous calculation, the MLE for $\boldsymbol{w}_*$ and σ is the minimizer of the function

$$F(\boldsymbol{w}, \sigma) = N \ln \sqrt{2\pi} + N \ln \sigma + \frac{1}{2\sigma^2} \|\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}\|_2^2 .$$

We first fix σ and minimize over $\boldsymbol{w}$, which leads to the same MLE for $\boldsymbol{w}*$:

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

Next we minimize $F(\boldsymbol{w}_*, \sigma)$ as function of σ by setting the derivative w.r.t. σ to be 0:

$$\frac{\partial F(\boldsymbol{w}_*, \sigma)}{\partial \sigma} = \frac{N}{\sigma} - \frac{1}{\sigma^3} \|\boldsymbol{X}\boldsymbol{w}_* - \boldsymbol{y}\|_2^2 = 0.$$

Solving for σ gives the MLE estimate:

$$\sigma = \frac{1}{\sqrt{N}} \|\boldsymbol{X}\boldsymbol{w}_* - \boldsymbol{y}\|_2 = \frac{1}{\sqrt{N}} \left\|\boldsymbol{X}(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X})^{-1}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{y} - \boldsymbol{y}\right\|_2 .$$

Rubrics:

- Write down the correct likelihood as a function of both $\boldsymbol{w}$ and σ. (1 point)
- Arrive at the correct solution for $\boldsymbol{w}_*$. (2 points)
- Correct derivative with respect to σ. (2 points)
- Arrive at the correct solution for σ. (1 point)

4 Linear Classifiers (42 points)

4.1 Multiple Choice (6 points)

(a) Which of the following is true?

(A) When the data is linearly separable, logistic loss (without regularization) does not admit a minimizer.
(B) When the data is linearly separable, perceptron converges to a max-margin classifier.
(C) Normalizing the output $\boldsymbol{w}$ of the perceptron algorithm so that $\|\boldsymbol{w}\|_2 = 1$ changes its test error.
(D) Normalizing the output $\boldsymbol{w}$ of the perceptron algorithm so that $\|\boldsymbol{w}\|_1 = 1$ changes its test error.

Ans: A. Take the binary case as an example, logistic loss is $\sum_{n=1} \ln(1+\exp(-y_n \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_n))$. When data is separable, one can find $\boldsymbol{w}$ such that $y_n \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_n \geq 0$ for all n. Scaling this $\boldsymbol{w}$ up will always lead to smaller loss and thus the function does not admit a minimizer. (2 points)

(b) The perceptron algorithm makes an update $\boldsymbol{w}' \leftarrow \boldsymbol{w} + \eta y_n \boldsymbol{x}_n$ with $\eta = 1$ when $\boldsymbol{w}$ misclassifies $\boldsymbol{x}_n$. Using which of the following different values for η will make sure $\boldsymbol{w}'$ classifies $\boldsymbol{x}_n$ correctly?

(A) $\eta > \frac{y(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_n)}{\|\boldsymbol{x}_n\|_2^2}$ (B) $\eta < \frac{-y(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_n)}{\|\boldsymbol{x}_n\|_2^2+1}$ (C) $\eta < \frac{-y(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_n)}{\|\boldsymbol{x}_n\|_2^2}$ (D) $\eta > \frac{-y(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_n)}{\|\boldsymbol{x}_n\|_2^2}$

Ans: D. Solve $y_n \boldsymbol{w}'^{\mathrm{T}} \boldsymbol{x}_n = y_n(\boldsymbol{w} + \eta y_n \boldsymbol{x}_n)^{\mathrm{T}} \boldsymbol{x}_n > 0$ for η. (2 points)

(c) Which of the following about SVM is true?

(A) Support vectors are training points that are misclassified.
(B) Support vectors are training points that are on the learned hyperplane.
(C) It is possible that a support vector is on the learned hyperplane.
(D) Only misclassified training points could be support vectors, but not all of them are.

Ans: C. (2 points)

4.2 Perceptron (4 points)

The following table shows a binary classification training set and the number of times each point is misclassified during a run of the perceptron algorithm. Write down the final output of the algorithm (that is, the weight vector that represents the hyperplane) assuming we append constant 1 as the first feature to all examples and start with the all-zero weight vector.

$\boldsymbol{x}$	y	Times misclassified
(-3, 2)	+1	5
(-1, 1)	-1	5
(5, 2)	+1	3
(2, 2)	-1	4
(1, -2)	+1	3

By the algorithm, the weight is a linear combination of data points weighted by the number of times they are misclassified and their label. That is (2 points)

$$\boldsymbol{w} = 5 \times (1, -3, 2) - 5 \times (1, -1, 1) + 3 \times (1, 5, 2) - 4 \times (1, 2, 2) + 3 \times (1, 1, -2) = (2, 0, -3). \quad (2 \text{ points})$$

4.3 Kernelized Perceptron (6 points)

Algorithm 1: Perceptron with nonlinear mapping $\boldsymbol{\phi}$

Input: A training set $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)$
Initialize: $\boldsymbol{w} = \mathbf{0}$
while *not converged* **do**
    randomly pick an example $(\boldsymbol{x}_n, y_n)$, make prediction $\hat{y} = \text{sign}(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x}_n))$
    if $\hat{y} \neq y_n$ **then**
        $\boldsymbol{w} \leftarrow \boldsymbol{w} + y_n \boldsymbol{\phi}(\boldsymbol{x}_n)$

Algorithm 2: Perceptron with kernel function k

Input: A training set $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)$
Initialize: $\alpha_n = 0$ for $n = 1, \ldots, N$ (2 points)
while *not converged* **do**
    randomly pick an example $(\boldsymbol{x}_n, y_n)$, make prediction $\hat{y} =$ $\text{sign}\left(\sum_{m=1}^{N} \alpha_m k(\boldsymbol{x}_m, \boldsymbol{x}_n)\right)$ (2 points)
    if $\hat{y} \neq y_n$ **then**
        $\alpha_n \leftarrow \alpha_n + y_n$ (2 points)

Algorithm 1 is the perceptron algorithm with a nonlinear mapping $\boldsymbol{\phi}$. Now given the corresponding kernel function k, please kernelize the algorithm and fill out the missing details in Algorithm 2.

4.4 Multiclass Perceptron (6 points)

Recall that a linear model for a multiclass classification problem with C classes is parameterized by C weight vectors $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_C \in \mathbb{R}^D$. In the class we derive the multiclass logistic regression by minimizing the multiclass logistic loss. In this problem you need to derive the multiclass perceptron algorithm in a similar way. Specifically, the multiclass perceptron loss on a training set $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N) \in \mathbb{R}^D \times [C]$ is defined as

$$F(\boldsymbol{w}_1, \ldots, \boldsymbol{w}_C) = \sum_{n=1}^{N} \max \left\{ 0, \max_{k \neq y_n} \boldsymbol{w}_k^{\mathrm{T}} \boldsymbol{x}_n - \boldsymbol{w}_{y_n}^{\mathrm{T}} \boldsymbol{x}_n \right\}.$$

Similarly to the binary case, multiclass perceptron is simply applying SGD with learning rate 1 to minimize the multiclass perceptron loss. Based on the above information, write down the multiclass perceptron algorithm below. (For simplicity, you do not need to worry about the non-differential points of F. In other words, the term $\max_{k \neq y_n} \boldsymbol{w}_k^{\mathrm{T}} \boldsymbol{x}_n - \boldsymbol{w}_{y_n}^{\mathrm{T}} \boldsymbol{x}_n$ is never 0.)

Solutions:

Algorithm 3: Multiclass Perceptron

Input: A training set $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)$
Initialize: $\boldsymbol{w}_1 = \cdots = \boldsymbol{w}_C = 0$ (or randomly)
while *not converged* **do**
    randomly pick an example $(\boldsymbol{x}_n, y_n)$, make prediction $\hat{y} = \operatorname{argmax}_{k \in [C]} \boldsymbol{w}_k^{\mathrm{T}} \boldsymbol{x}_n$ (2 points)
    if $\hat{y} \neq y_n$ **then**
        $\boldsymbol{w}_{\hat{y}} \leftarrow \boldsymbol{w}_{\hat{y}} - \boldsymbol{x}_n$ (2 points)
        $\boldsymbol{w}_{y_n} \leftarrow \boldsymbol{w}_{y_n} + \boldsymbol{x}_n$ (2 points)

4.5 Perceptron for Non-separable Data (20 points)

Warning: you might find this problem the hardest and want to finish other problems first.

In Homework 1 you are asked to prove a bound on the number of mistakes that perceptron makes for a separable data set. In this problem you need to follow the steps below to prove a bound for the general case with possibly non-separable data. In particular we consider the following variant of perceptron where instead of randomly picking a data point in each iteration, we simply make one pass of the data set. We also assume that the data is normalized so that $\|\boldsymbol{x}_n\|_2 = 1$ for all $n \in [N]$.

Algorithm 4: One-pass Perceptron

Input: A training set $S = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\}$
Initialize: $\boldsymbol{w}_1 = \boldsymbol{0}$
for $n = 1, \ldots, N$ **do**
    make prediction $\hat{y} = \text{sign}(\boldsymbol{w}_n^{\mathrm{T}} \boldsymbol{x}_n)$
    if $\hat{y} \neq y_n$ **then**
        $\boldsymbol{w}_{n+1} = \boldsymbol{w}_n + y_n \boldsymbol{x}_n$

(a) Let M be the number of mistakes Algorithm 4 makes. Prove $\|\boldsymbol{w}_{N+1}\|_2 \leq \sqrt{M}$.

This is exactly the same as the question in Homework 1. By the algorithm we have for any n,

$$\begin{aligned} \|\boldsymbol{w}_{n+1}\|_2^2 &= \left\|\boldsymbol{w}_n + \mathbb{I}[y_n \boldsymbol{w}_n^{\mathrm{T}} \boldsymbol{x}_n \leq 0] y_n \boldsymbol{x}_n\right\|_2^2 \\ &= \|\boldsymbol{w}_n\|_2^2 + \mathbb{I}[y_n \boldsymbol{w}_n^{\mathrm{T}} \boldsymbol{x}_n \leq 0](2 y_n \boldsymbol{w}_n^{\mathrm{T}} \boldsymbol{x}_n + y_n^2 \|\boldsymbol{x}_n\|_2^2) \\ &\leq \|\boldsymbol{w}_n\|_2^2 + \mathbb{I}[y_n \boldsymbol{w}_n^{\mathrm{T}} \boldsymbol{x}_n \leq 0] \end{aligned}$$ (2 points)

Applying this inequality recursively we get $\|\boldsymbol{w}_{N+1}\|_2^2 \leq \sum_{n=1}^{N} \mathbb{I}[y_n \boldsymbol{w}_n^{\mathrm{T}} \boldsymbol{x}_n \leq 0] = M$ and thus $\|\boldsymbol{w}_{N+1}\|_2 \leq \sqrt{M}$. (1 point)

(b) Next show that for any $\boldsymbol{w}_*$ with $\|\boldsymbol{w}_*\|_2 \leq 1$, we have $\boldsymbol{w}_{N+1}^{\mathrm{T}}\boldsymbol{w}_* \geq M - M_*$ where $M_* = \sum_{n=1}^{N} \max\{0, 1 - y_n \boldsymbol{w}_*^{\mathrm{T}}\boldsymbol{x}_n\}$ is the total hinge loss of $\boldsymbol{w}_*$.

For any n, we have

$$\begin{aligned}
\boldsymbol{w}_{n+1}^{\mathrm{T}}\boldsymbol{w}_* &= \boldsymbol{w}_n^{\mathrm{T}}\boldsymbol{w}_* + \mathbb{I}[y_n \boldsymbol{w}_n^{\mathrm{T}}\boldsymbol{x}_n \leq 0] y_n \boldsymbol{x}_n^{\mathrm{T}}\boldsymbol{w}_* \\
&= \boldsymbol{w}_n^{\mathrm{T}}\boldsymbol{w}_* + \mathbb{I}[y_n \boldsymbol{w}_n^{\mathrm{T}}\boldsymbol{x}_n \leq 0] \left(1 - (1 - y_n \boldsymbol{x}_n^{\mathrm{T}}\boldsymbol{w}_*)\right) && \text{(1 point)} \\
&\geq \boldsymbol{w}_n^{\mathrm{T}}\boldsymbol{w}_* + \mathbb{I}[y_n \boldsymbol{w}_n^{\mathrm{T}}\boldsymbol{x}_n \leq 0] \left(1 - \max\{0, 1 - y_n \boldsymbol{x}_n^{\mathrm{T}}\boldsymbol{w}_*\}\right) && \text{(1 point)} \\
&\geq \boldsymbol{w}_n^{\mathrm{T}}\boldsymbol{w}_* + \mathbb{I}[y_n \boldsymbol{w}_n^{\mathrm{T}}\boldsymbol{x}_n \leq 0] - \max\{0, 1 - y_n \boldsymbol{x}_n^{\mathrm{T}}\boldsymbol{w}_*\} && \text{(1 point)}
\end{aligned}$$

Applying this inequality recursively we get

$$\boldsymbol{w}_{N+1}^{\mathrm{T}}\boldsymbol{w}_* \geq \sum_{n=1}^{N} \mathbb{I}[y_n \boldsymbol{w}_n^{\mathrm{T}}\boldsymbol{x}_n \leq 0] - \sum_{n=1}^{N} \max\{0, 1 - y_n \boldsymbol{x}_n^{\mathrm{T}}\boldsymbol{w}_*\} = M - M_*. \quad \text{(1 point)}$$

(c) Combine the above two results to prove a mistake bound $M \leq 2M_* + 1$. (Hint: as in Homework 1 you need to use the Cauchy-Schwarz inequality $\boldsymbol{a}^{\mathrm{T}}\boldsymbol{b} \leq \|\boldsymbol{a}\|_2 \|\boldsymbol{b}\|_2$)

Since

$$\boldsymbol{w}_{N+1}^{\mathrm{T}}\boldsymbol{w}_* \leq \|\boldsymbol{w}_{N+1}\|_2 \|\boldsymbol{w}_*\|_2 \leq \sqrt{M} \quad \text{(1 point)}$$

and

$$\sqrt{M} \leq \frac{1}{2}(M + 1), \quad \text{(1 point)}$$

we have $M - M_* \leq \frac{1}{2}(M + 1)$. Solving for M gives $M \leq 2M_* + 1$. (1 point)

(Directly solving $M - M_* \leq \sqrt{M}$ for M also works too.)

(d) Assume each data point in the training set S is an i.i.d. sample of a fixed distribution $\mathcal{P}$, which we denote as $S \sim \mathcal{P}$. Recall that the risk (in terms of 0-1 loss) of a classifier f is defined as $R(f) = \mathbb{E}[\mathbb{I}[f(\boldsymbol{x}) \neq y]]$ where the expectation is with respect to the random draw of $(\boldsymbol{x}, y) \sim \mathcal{P}$ and the potential internal randomness of the classifier f. We also define the risk in terms of the hinge loss of a linear classifier $\boldsymbol{w}$ as $R_{\text{hinge}}(\boldsymbol{w}) = \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{P}}[\max\{0, 1 - y\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}\}]$.

Now consider the following *randomized* classifier f: for any $\boldsymbol{x}$, the prediction is $\text{sign}(\boldsymbol{w}_n^{\mathrm{T}}\boldsymbol{x})$ where $\boldsymbol{w}_n$ is drawn uniformly at random from $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_N$ defined in Algorithm 4. Prove

$$\mathbb{E}_{S\sim\mathcal{P}}[R(f)] \leq 2 \min_{\boldsymbol{w}:\|\boldsymbol{w}\|_2\leq 1} R_{\text{hinge}}(\boldsymbol{w}) + \frac{1}{N}.$$

The result from the last question can be written as

$$\frac{1}{N}\sum_{n=1}^{N} \mathbb{I}[\text{sign}(\boldsymbol{w}_n^{\mathrm{T}}\boldsymbol{x}_n) \neq y_n] \leq \frac{2}{N}\sum_{n=1}^{N} \max\{0, 1 - y_n\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_n\} + \frac{1}{N} \qquad \text{(1 point)}$$

for any $\boldsymbol{w}$ such that $\|\boldsymbol{w}\|_2 \leq 1$. Taking the expectation on both sides with respect to $S \sim \mathcal{P}$ we arrive at

$$\frac{1}{N}\sum_{n=1}^{N} \mathbb{E}_{(\boldsymbol{x}_1,y_1)\sim\mathcal{P},\ldots,(\boldsymbol{x}_{n-1},y_{n-1})\sim\mathcal{P}}\mathbb{E}_{(\boldsymbol{x}_n,y_n)\sim\mathcal{P}}[\mathbb{I}[\text{sign}(\boldsymbol{w}_n^{\mathrm{T}}\boldsymbol{x}_n) \neq y_n]] \leq 2R_{\text{hinge}}(\boldsymbol{w}) + \frac{1}{N}. \qquad \text{(2 points)}$$

Since $\boldsymbol{w}_n$ is independent of the flash new example $(\boldsymbol{x}_n, y_n)$, we have

$$\mathbb{E}_{(\boldsymbol{x}_n,y_n)\sim\mathcal{P}}[\mathbb{I}[\text{sign}(\boldsymbol{w}_n^{\mathrm{T}}\boldsymbol{x}_n) \neq y_n]] = R(\boldsymbol{w}_n), \qquad \text{(2 points)}$$

the risk of the linear classifier $\boldsymbol{w}_n$. Therefore we have

$$\mathbb{E}_{S\sim\mathcal{P}}\left[\frac{1}{N}\sum_{n=1}^{N} R(\boldsymbol{w}_n)\right] \leq 2R_{\text{hinge}}(\boldsymbol{w}) + \frac{1}{N}. \qquad \text{(1 point)}$$

By the definition of the randomized classifier f, the left hand side is exactly $\mathbb{E}_{S\sim\mathcal{P}}[R(f)]$, proving the claimed statement since $\boldsymbol{w}$ is any weight such that $\|\boldsymbol{w}\|_2 \leq 1$. (1 point)

(e) Suppose we now make T passes (instead of one pass) of the data to obtain $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_{TN}$ using the perceptron algorithm. Similar to f defined in the last question, we define a new randomized classifier f' which predicts by randomly picking one of the TN hyperplanes $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_{TN}$ and then following the prediction of that hyperplane. Does the following hold

$$\mathbb{E}_{S\sim\mathcal{P}}[R(f')] \leq 2 \min_{\boldsymbol{w}:\|\boldsymbol{w}\|_2\leq 1} R_{\text{hinge}}(\boldsymbol{w}) + \frac{1}{TN}?$$

Prove it if you think so, otherwise briefly explain why this is not true.

This is not true. (1 point)

Following the previous proof you will find that it breaks if one wants to claim for example

$$\mathbb{E}_{(\boldsymbol{x}_1,y_1)\sim\mathcal{P}}[\mathbb{I}[\text{sign}(\boldsymbol{w}_{N+1}^{\mathrm{T}}\boldsymbol{x}_1) \neq y_1]] = R(\boldsymbol{w}_{N+1}).$$

This is because $\boldsymbol{w}_{N+1}$ depends on $(\boldsymbol{x}_1, y_1)$.

Another way to see that this should not be true is: if it was true, then by making many passes of a training set of size one, the term $\frac{1}{TN}$ will still go to zero, leading to a very strong risk bound, which is clearly impossible.

Similar explanations along this line will all get 2 points.

5 Neural Networks (24 points)

5.1 Multiple Choice (10 points)

(a) Overfitting is a major problem for neural networks. Which of the following can help prevent overfitting?

(A) Retraining on the same data many times.
(B) Using a larger learning rate for Backpropagation .
(C) Dropping random neurons in each iteration of Backpropagation.
(D) Training until you get the smallest training error.

Ans: C. (2 points)

(b) Which of the following is true about neural nets?

(A) A fully connected feedforward neural net without nonlinear activation functions is the same as a linear model.
(B) Since we usually apply Backpropagation (i.e. SGD) to learn neural networks, it is a convex problem.
(C) A neural net with one hidden layer and a fixed number of neurons can represent any continuous function.
(D) A max-pooling layer with a 2×2 filter has 4 parameters to be learned.

Ans: A. (2 points)

(c) Suppose a convolution layer takes a 4×6 image with 3 channels as input and outputs a $3 \times 4 \times 6$ volume. Which of the following is a possible configuration of this layer?

(A) One 2×3 filter with depth 6, stride 1, no zero-padding.
(B) Six 2×3 filters with depth 3, stride 1, no zero-padding.
(B) Six 3×4 filters with depth 3, stride 2, no zero-padding.
(B) Six 3×4 filters with depth 3, stride 1, 1 pixel of zero-padding.

Ans: B. (2 points)

(d) How many parameters do we need to learn for the following network structure? An $8 \times 8 \times 3$ image input, followed by a convolution layer with 2 filters of size 2×2 (stride 1, no zero-padding), then another convolution layer with 4 filters of size 3×3 (stride 2, no zero-padding), and finally a max-pooling layer with a 2×2 filter (stride 1, no zero-padding). (Note: the depth of all filters are not explicitly spelled out, and we assume no bias/intercept terms are used.)

(A) 96
(B) 44
(C) 100
(D) 48

Ans: A. $2 \times (2 \times 2 \times 3) + 4 \times (3 \times 3 \times 2) = 96$. (2 points)

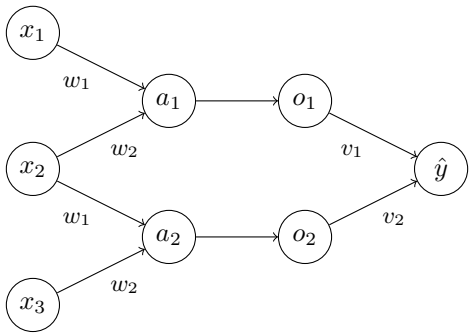
(e) What is the final output dimension of the last question?

(A) $2 \times 2 \times 3$
(B) $5 \times 5 \times 4$
(C) $2 \times 2 \times 1$
(D) $2 \times 2 \times 4$

Ans: D. (2 points)

5.2 Backpropagation for CNN (14 points)

Consider the following mini convolutional neural net, where (x_1, x_2, x_3) is the input, followed by a convolution layer with a filter (w_1, w_2), a ReLU layer, and a fully connected layer with weight (v_1, v_2).



More concretely, the computation is specified by

$$\begin{aligned} a_1 &= x_1 w_1 + x_2 w_2 \\ a_2 &= x_2 w_1 + x_3 w_2 \\ o_1 &= \max\{0, a_1\} \\ o_2 &= \max\{0, a_2\} \\ \hat{y} &= o_1 v_1 + o_2 v_2 \end{aligned}$$

For an example $(\boldsymbol{x}, y) \in \mathbb{R}^3 \times \{-1, +1\}$, we define the logistic loss of the CNN as

$$\ell = \ln(1 + \exp(-y\hat{y}))$$

which is a function of the parameters of the network: w_1, w_2, v_1, v_2.

(a) Write down $\frac{\partial \ell}{\partial v_1}$ and $\frac{\partial \ell}{\partial v_2}$. You can use the sigmoid function $\sigma(z) = \frac{1}{1+e^{-z}}$ to simplify your notation.

$$\frac{\partial \ell}{\partial v_1} = \frac{\partial \ell}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial v_1} \qquad \text{(1 point)}$$

$$= \frac{-ye^{-y\hat{y}}}{1 + e^{-y\hat{y}}} o_1 = -\sigma(-y\hat{y}) y o_1 = (\sigma(y\hat{y}) - 1) y o_1 \qquad \text{(1 point)}$$

$$\frac{\partial \ell}{\partial v_2} = \frac{\partial \ell}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial v_2} \qquad \text{(1 point)}$$

$$= \frac{-ye^{-y\hat{y}}}{1 + e^{-y\hat{y}}} o_2 = -\sigma(-y\hat{y}) y o_2 = (\sigma(y\hat{y}) - 1) y o_2 \qquad \text{(1 point)}$$

Either one of the last three expressions is acceptable.

(b) Write down $\frac{\partial \ell}{\partial w_1}$ and $\frac{\partial \ell}{\partial w_2}$. The derivative of the ReLU function is $H(a) = \mathbb{I}[a > 0]$, which you can use directly in your answer.

$$\frac{\partial \ell}{\partial w_1} = \frac{\partial \ell}{\partial a_1}\frac{\partial a_1}{\partial w_1} + \frac{\partial \ell}{\partial a_2}\frac{\partial a_2}{\partial w_1} \quad \text{(1 point)}$$

$$= \frac{\partial \ell}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial o_1}\frac{\partial o_1}{\partial a_1}\frac{\partial a_1}{\partial w_1} + \frac{\partial \ell}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial o_2}\frac{\partial o_2}{\partial a_2}\frac{\partial a_2}{\partial w_1} = (\sigma(y\hat{y}) - 1)y(v_1 H(a_1)x_1 + v_2 H(a_2)x_2) \quad \text{(1 point)}$$

Similarly

$$\frac{\partial \ell}{\partial w_2} = \frac{\partial \ell}{\partial a_1}\frac{\partial a_1}{\partial w_2} + \frac{\partial \ell}{\partial a_2}\frac{\partial a_2}{\partial w_2} \quad \text{(1 point)}$$

$$= \frac{\partial \ell}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial o_1}\frac{\partial o_1}{\partial a_1}\frac{\partial a_1}{\partial w_2} + \frac{\partial \ell}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial o_2}\frac{\partial o_2}{\partial a_2}\frac{\partial a_2}{\partial w_2} = (\sigma(y\hat{y}) - 1)y(v_1 H(a_1)x_2 + v_2 H(a_2)x_3). \quad \text{(1 point)}$$

Again, other equivalent expressions are acceptable.

(c) Using the derivations above, fill out the missing details for the Backpropagation algorithm below that is used to train this mini CNN.

Algorithm 5: Backpropagation for the above mini CNN

Input: A training set $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)$, learning rate η

Initialize: $w_1 = w_2 = v_1 = v_2 = 0$ (or randomly) (1 point)

while *not converged* **do**

  randomly pick an example $(\boldsymbol{x}_n, y_n)$

  Forward propagation: compute (2 points)

$$a_1 = x_1 w_1 + x_2 w_2, a_2 = x_2 w_1 + x_3 w_2$$

$$o_1 = \max\{0, a_1\}, o_2 = \max\{0, a_2\}, \hat{y} = o_1 v_1 + o_2 v_2$$

  Backward propagation: update (3 points)

$$w_1 \leftarrow w_1 - \eta(\sigma(y\hat{y}) - 1)y(v_1 H(a_1)x_1 + v_2 H(a_2)x_2)$$
$$w_2 \leftarrow w_2 - \eta(\sigma(y\hat{y}) - 1)y(v_1 H(a_1)x_2 + v_2 H(a_2)x_3)$$
$$v_1 \leftarrow v_1 - \eta(\sigma(y\hat{y}) - 1)y o_1$$
$$v_2 \leftarrow v_2 - (\sigma(y\hat{y}) - 1)y o_2$$

Deduct 2 points if updating w_1/w_2 with the updated value of v_1/v_2.
Do not deduct points for using the wrong gradients due to mistakes from previous two questions.

6 Kernel Methods (12 points)

6.1 Multiple Choice (6 points)

(a) Which of the following is true about kernel function?

(A) If k_1 and k_2 are kernel, then $c_1k_1 + c_2k_2$ is a kernel too for any $c_1, c_2 \in \mathbb{R}$.
(B) Kernel function must be symmetric, that is, $k(\boldsymbol{x}, \boldsymbol{x}') = k(\boldsymbol{x}', \boldsymbol{x})$.
(C) If k is a kernel, then $-k$ is a kernel too.
(D) If k is a kernel, then $\ln k$ is a kernel too.

Ans: B. (2 points)

(b) Which of the following is not a kernel function?

(A) $k(x, x') = (\boldsymbol{x}^{\mathrm{T}}\boldsymbol{x}')^2$
(B) $k(x, x') = (\boldsymbol{x}^{\mathrm{T}}\boldsymbol{x}' + 1)^2$
(C) $k(x, x') = -\left\|\boldsymbol{x} - \boldsymbol{x}'\right\|_2^2$
(D) $k(x, x') = \exp(-\left\|\boldsymbol{x} - \boldsymbol{x}'\right\|_2^2)$

Ans: C. (2 points)

(c) Vovk's real polynomial kernel $k : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$ is defined as:

$$k(\boldsymbol{x}, \boldsymbol{x}') = \frac{1 - (\boldsymbol{x}^{\mathrm{T}}\boldsymbol{x}')^p}{1 - (\boldsymbol{x}^{\mathrm{T}}\boldsymbol{x}')}$$

where p is a non-negative integer. Which of the following is the corresponding non-linear mapping for this kernel when $D = 2$ and $p = 3$?

(A) $\boldsymbol{\phi}(\boldsymbol{x}) = [x_1^2x_2^2, 2x_1x_2, x_1, x_2, 1]^T$
(B) $\boldsymbol{\phi}(\boldsymbol{x}) = [x_1^2x_2^2, \sqrt{2}x_1x_2, x_1, x_2, 1]^T$
(C) $\boldsymbol{\phi}(\boldsymbol{x}) = [x_1^2, x_2^2, \sqrt{2}x_1x_2, x_1, x_2, 1]^T$
(D) $\boldsymbol{\phi}(\boldsymbol{x}) = [x_1^2, x_2^2, 2x_1x_2, x_1, x_2, 1]^T$

Ans: C. (2 points)

$$\begin{aligned}\frac{1 - (\boldsymbol{x}^{\mathrm{T}}\boldsymbol{x}')^3}{1 - (\boldsymbol{x}^{\mathrm{T}}\boldsymbol{x}')} &= \frac{(1 - (\boldsymbol{x}^{\mathrm{T}}\boldsymbol{x}'))(1 + \boldsymbol{x}^{\mathrm{T}}\boldsymbol{x}' + (\boldsymbol{x}^{\mathrm{T}}\boldsymbol{x}')^2)}{1 - (\boldsymbol{x}^{\mathrm{T}}\boldsymbol{x}')} \\ &= x_1x_1' + x_2x_2' + (x_1x_1' + x_2x_2')^2 + 1 \\ &= x_1^2{x_1'}^2 + x_2^2{x_2'}^2 + 2x_1x_1'x_2x_2' + x_1x_1' + x_2x_2' + 1\end{aligned}$$

Thus, $\boldsymbol{\phi}(\boldsymbol{x}) = [x_1^2, x_2^2, \sqrt{2}x_1x_2, x_1, x_2, 1]^T$.

6.2 Kernel Composition (6 points)

Prove that if $k_1, k_2 : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$ are both kernel functions, then $k(\boldsymbol{x}, \boldsymbol{x}') = k_1(\boldsymbol{x}, \boldsymbol{x}')k_2(\boldsymbol{x}, \boldsymbol{x}')$ is a kernel function too. Specifically, suppose $\boldsymbol{\phi}_1$ and $\boldsymbol{\phi}_2$ are the corresponding mappings for k_1 and k_2 respectively. Construct the mapping $\boldsymbol{\phi}$ that certifies k being a kernel function.

Observe that

$$k(\boldsymbol{x}, \boldsymbol{x}') = k_1(\boldsymbol{x}, \boldsymbol{x}')k_2(\boldsymbol{x}, \boldsymbol{x}') = \left(\sum_{i=1}^{D} \phi_1(\boldsymbol{x})_i \phi_1(\boldsymbol{x}')_i\right)\left(\sum_{j=1}^{D} \phi_2(\boldsymbol{x})_j \phi_2(\boldsymbol{x}')_j\right) \quad \text{(2 points)}$$

$$= \sum_{i=1}^{D}\sum_{j=1}^{D} \phi_1(\boldsymbol{x})_i \phi_1(\boldsymbol{x}')_i \phi_2(\boldsymbol{x})_j \phi_2(\boldsymbol{x}')_j = \sum_{i=1}^{D}\sum_{j=1}^{D} \left(\phi_1(\boldsymbol{x})_i \phi_2(\boldsymbol{x})_j\right)\left(\phi_1(\boldsymbol{x}')_i \phi_2(\boldsymbol{x}')_j\right) \quad \text{(2 points)}$$

Therefore, $\boldsymbol{\phi}(\boldsymbol{x}) = \boldsymbol{\phi}_1(\boldsymbol{x})\boldsymbol{\phi}_2(\boldsymbol{x})^{\mathrm{T}}$. Writing it as a D^2-dimensional vector is acceptable too. (2 points)

This blank page can be used as scratch paper.

This blank page can be used as scratch paper.