CSCI 567 Fall 2018 Second 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	30	6	17	8	23	16	100
Points							

Please read the following instructions carefully:

- The exam has a total of **19 pages** (double-sided, including this cover and two blank pages in the end) and **6 problems**. Each problem have several 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.
- Duration of the exam is **2 hours and 20 minutes**. Questions are not ordered by their difficulty. Budget your time on each question carefully.
- Select **one and only one answer** for all multiple choice questions.
- Answers should be **concise** and written down **legibly**. All questions can be done within 5-12 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.
- 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.
- You **may not** leave your seat **for any reason** unless you submit your exam at that point.

1 Multiple Choice (30 points)

(a) Suppose we apply the kernel trick with a kernel function k to the nearest neighbor algorithm (with L2 distance in the new feature space). What is the nearest neighbor of a new data point $\boldsymbol{x}$ from a training set $S = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$?

(A) $\operatorname{argmin}_{\boldsymbol{x}_n \in S}\ k(\boldsymbol{x}_n, \boldsymbol{x}_n) + k(\boldsymbol{x}, \boldsymbol{x}) + 2k(\boldsymbol{x}_n, \boldsymbol{x})$

(B) $\operatorname{argmin}_{\boldsymbol{x}_n \in S}\ k(\boldsymbol{x}_n, \boldsymbol{x})$

(C) $\operatorname{argmin}_{\boldsymbol{x}_n \in S}\ (k(\boldsymbol{x}_n, \boldsymbol{x}) - k(\boldsymbol{x}, \boldsymbol{x}_n))^2$

(D) $\operatorname{argmin}_{\boldsymbol{x}_n \in S}\ k(\boldsymbol{x}_n, \boldsymbol{x}_n) - 2k(\boldsymbol{x}_n, \boldsymbol{x})$

Ans: D. (2 points)

(b) Let $\boldsymbol{X} \in \mathbb{R}^{N\times D}$ be a data matrix with each row corresponding to the feature of an example and $\boldsymbol{y} \in \mathbb{R}^N$ be a vector of all the outcomes, as used in the class. The least square solution is $(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$. Which of the following is the least square solution if we scale each data point by one half (i.e. the new dataset is $\frac{1}{2}\boldsymbol{X}$)?

(A) $2(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$ (B) $4(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$

(C) $\frac{1}{2}(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$ (D) $\frac{1}{4}(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$

Ans: A. (2 points)

(c) Which of the following surrogate losses is not an upper bound of the 0-1 loss?

(A) perceptron loss $\max\{0, -z\}$

(B) hinge loss $\max\{0, 1 - z\}$

(C) logistic loss $\log(1 + \exp(-z))$

(D) exponential loss $\exp(-z)$

Ans: A. (2 points)

(d) 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. Which of the following is the final output of the algorithm?

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

(A) (2, -1) (B) (-1, -1) (C) (-2, 1) (D) (1, 1)

Ans: B. (2 points)

(e) Suppose we obtain a hyperplane $\boldsymbol{w}$ via logistic regression and are going to make a randomized prediction on the label y of a new point $\boldsymbol{x}$ based on the sigmoid model. What is the probability of predicting $y = -1$?

(A) $e^{-\boldsymbol{w}^T\boldsymbol{x}}$ (B) $\mathbb{I}[\boldsymbol{w}^T\boldsymbol{x} \le 0]$

(C) $\frac{1}{1+e^{-\boldsymbol{w}^T\boldsymbol{x}}}$ (D) $\frac{1}{1+e^{\boldsymbol{w}^T\boldsymbol{x}}}$

Ans: D. (2 points)

(f) The multiclass perceptron algorithm is shown in Alg 1, and it is essentially minimizing the multiclass perceptron loss via SGD with learning rate 1. Based on this information, which of the following is the multiclass perceptron loss?

Algorithm 1: Multiclass Perceptron

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

(A) $\sum_{n=1}^{N} \max_{k\neq y_n} \boldsymbol{w}_k^{\mathrm{T}}\boldsymbol{x}_n - \boldsymbol{w}_{y_n}^{\mathrm{T}}\boldsymbol{x}_n$

(B) $\sum_{n=1}^{N} \max\left\{0, \max_{k\neq y_n} \boldsymbol{w}_k^{\mathrm{T}}\boldsymbol{x}_n\right\}$

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

(D) $\sum_{n=1}^{N} \max\left\{0, \max_{k\in[C]} \boldsymbol{w}_k^{\mathrm{T}}\boldsymbol{x}_n\right\}$

Ans: C. (2 points)

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

(A) One-versus-all

(B) One-versus-one

(C) Tree reduction

(D) Both (A) and (B)

Ans: B. (2 points)

(h) Which of the following is wrong about kernel function?

(A) $k(\boldsymbol{x}, \boldsymbol{x}') = \mathbb{I}[\boldsymbol{x} = \boldsymbol{x}']$ is a kernel function.

(B) If k_1 and k_2 are kernel functions, then k_1k_2 is a kernel function too.

(C) Kernel function must be symmetric, that is, $k(\boldsymbol{x}, \boldsymbol{x}') = k(\boldsymbol{x}', \boldsymbol{x})$.

(D) If k is a kernel function, then $\ln(k)$ is a kernel function too.

Ans: D. (2 points)

(i) Which of the following is wrong about neural nets?

(A) A fully connected feedforward neural net without nonlinear activation functions is the same as a linear model.

(B) Data augmentation helps prevent overfitting.

(C) A neural net with one hidden layer and a fixed number of neurons can represent any continuous function.

(D) A max-pooling layer has no parameters to be learned.

Ans: C. (2 points)

(j) 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 not on the learned hyperplane.

(C) Removing examples that are not support vectors will not affect the final hyperplane.

(D) Only misclassified training points could be support vectors, but not all of them are.

Ans: C. (2 points)

(k) Recall that the Gini impurity of a distribution p over a set of K items is defined as $\sum_{k=1}^{K} p(k)(1-p(k))$. Which of the following distributions has the largest Gini impurity?

(A) (0.25, 0.25, 0.25, 0.25)

(B) (0.2, 0.3, 0.5, 0)

(C) (1, 0, 0, 0)

(D) (0.5, 0.5, 0, 0)

Ans: A. Like entropy, Gini impurity is also maximized when the distribution is uniform. (2 points)

(l) Which of the following statement is true?

(A) AdaBoost outputs a linear classifier if the base classifiers are linear.

(B) In PCA, the first principal component is the direction where the projection of the dataset has the largest variance.

(C) Kernel density estimation is a parametric method.

(D) K-means algorithm always converges to the global optimum, but it can take very long for this to happen.

Ans: B. (2 points)

(m) Which of the following is wrong about PCA?

(A) The first step of kernel PCA is to center the original dataset.

(B) The first principal component is the eigenvector of the covariance matrix with the largest eigenvalue.

(C) PCA outputs a compressed dataset that is a linear transformation of the original dataset.

(D) Kernel PCA requires computing eigenvalues and eigenvectors of the Gram matrix.

Ans: A. (2 points)

(n) Which of the following has the most adaptive exploration for the multi-armed bandit problem?

(A) A completely random strategy

(B) Greedy strategy

(C) Explore-then-exploit

(D) Upper Confidence Bound (UCB) algorithm

Ans: D. (2 points)

(o) Which of the following is true about reinforcement learning?

(A) Value iteration is not guaranteed to converge to the actual value function.

(B) Model-free approaches requires less space than model-based approaches.

(C) Q-learning is a model-based approach.

(D) Mode-based approaches do not need to deal with the exploitation-exploration trade-off.

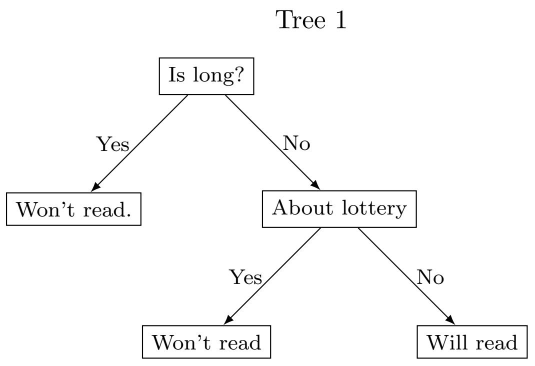
Ans: B. (2 points)

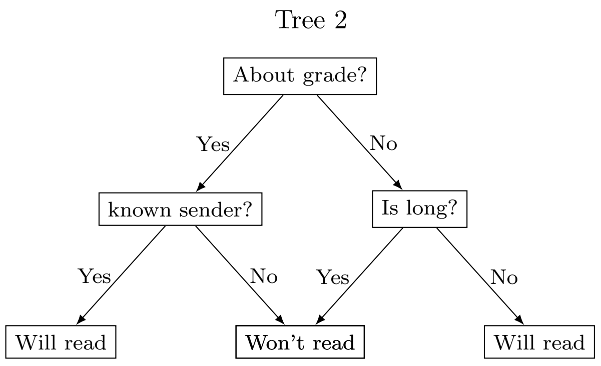
2 Decision Tree (6 points)

Suppose we would like to build a decision tree classifier to predict "whether the professor will read the email", using the following training dataset where the first 5 columns represent 5 binary features (whether the professor knows the sender, whether the email is too long, and whether it is about certain topics), and the last column is the label.

Known sender?	Is Long?	About research?	About grade?	About lottery?	Read?
no	no	yes	yes	no	no
yes	yes	no	yes	no	no
no	yes	yes	yes	yes	no
yes	yes	yes	yes	no	no
no	yes	no	no	no	no
yes	no	yes	yes	yes	yes
no	no	yes	no	no	yes
yes	no	no	no	no	yes
yes	no	yes	yes	no	yes
yes	yes	yes	yes	yes	no

Below are two examples of decision tree for this task:





(a) How many training points are correctly classified by Tree 1 and Tree 2 respectively?

(A) 8 and 7 (B) 9 and 8 (C) 8 and 8 (D) 9 and 7

Ans: A (2 points)

(b) Define the **Information Gain** of a node n with children Children(n) as

$$\text{Gain}(n) = \text{Entropy}(S_n) - \sum_{m \in \text{Children}(n)} \frac{|S_m|}{|S_n|} \text{Entropy}(S_m)$$

where S_n and S_m are the subsets of training examples that belong to the node n and one of its child node m respectively. Compute the information gain for the node <u>"known sender?"</u> of Tree 2. Express your answer in terms of "log", that is, you do not need to calculate the value of logarithm (and therefore the base of the logarithm also does not matter). Also recall $0 \log 0 = 0$.

The entropy of the node "known sender?" is

$$\text{Entropy}(S) = -\frac{2}{7}\log\frac{2}{7} - \frac{5}{7}\log\frac{5}{7} \qquad \text{(1 point)}$$

The entropy for the left child and right child are respectively

$$\text{Entropy}(S_1) = -\frac{2}{5}\log\frac{2}{5} - \frac{3}{5}\log\frac{3}{5} \qquad \text{(1 point)}$$

and

$$\text{Entropy}(S_2) = -\frac{0}{2}\log\frac{0}{2} - \frac{2}{2}\log\frac{2}{2} = 0 \qquad \text{(1 point)}$$

The information gain is therefore

$$\text{Entropy}(S) - \frac{5}{7}\text{Entropy}(S_1) \qquad \text{(1 point)}$$

Note: incorrect information gain due to mistakes from previous calculations of entropy can still get the 1 point if the formula is applied correctly.

3 Boosting (17 points)

A version of the AdaBoost algorithm is shown below where the base algorithm is simply searching for a classifier with the smallest weighted error from a fixed classifier set $\mathcal{H}$.

Algorithm 2: Adaboost

```
1 Given: A training set $\{(\boldsymbol{x}_n, y_n \in \{+1,-1\})\}_{n=1}^N$, and a set of classifier $\mathcal{H}$, where each $h \in \mathcal{H}$ takes a
    feature vector as input and outputs $+1$ or $-1$.
2 Goal: Learn $H(\boldsymbol{x}) = \text{sign}\left(\sum_{t=1}^T \beta_t h_t(\boldsymbol{x})\right)$, where $h_t \in \mathcal{H}$, $\beta_t \in \mathbb{R}$, and $\text{sign}(a) = \begin{cases} +1, & \text{if } a \geq 0, \\ -1, & \text{otherwise.} \end{cases}$
3 Initialization: $D_1(n) = \frac{1}{N}$, $\forall n \in [N]$.
4 for $t = 1, 2, \cdots, T$ do
5     Find $h_t = \text{argmin}_{h \in \mathcal{H}} \sum_{n: y_n \neq h(\boldsymbol{x}_n)} D_t(n)$.
6     Compute
          $\epsilon_t = \sum_{n: y_n \neq h_t(\boldsymbol{x}_n)} D_t(n)$    and    $\beta_t = \frac{1}{2} \log \frac{1-\epsilon_t}{\epsilon_t}$.
7     Compute
          $D_{t+1}(n) = \frac{D_t(n) \exp(-\beta_t y_n h_t(\boldsymbol{x}_n))}{Z_t}$
      for each $n \in [N]$, where
          $Z_t = \sum_{m=1}^N D_t(m) \exp(-\beta_t y_m h_t(\boldsymbol{x}_m))$
      is the normalization factor.
```

3.1 Executing AdaBoost

Imagine running AdaBoost with a 1-dimensional training set of 8 examples as shown in Fig. 1, where circles mean $y = +1$ and crosses mean $y = -1$. The base classifier set $\mathcal{H}$ consists of all decision stumps, where each of them is parameterized by a pair $(s, b) \in \{+1, -1\} \times \mathbb{R}$ such that

$$h_{(s,b)}(x) = \begin{cases} s, & \text{if } x > b, \\ -s, & \text{otherwise.} \end{cases}$$

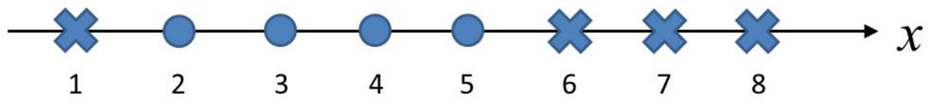


Figure 1: The 1-dimensional training set with 8 examples. Circles mean $y = +1$ and crosses mean $y = -1$. The number under each example is its x coordinate.

(a) Which of the following is the possible parameter for h_1?

(A) (s, b) = (+1, 3.5)
(B) (s, b) = (-1, 3.5)
(C) (s, b) = (+1, 5.5)
(D) (s, b) = (-1, 5.5)

Ans: D. (2 points)

(b) Suppose we run AdaBoost for two rounds and observe that β_1 and β_2 are both positive but not equal. Is it possible that the final classifier H after these two rounds (see Line 2) has zero training error? Why or why not?

No, it's not possible. (1 point)

The example $x = 1$ is misclassified by h_1, and there are two cases for h_2:

- h_2 also misclassifies this example, and thus H misclassifies it too;
- h_2 correctly classifies this example, then it must misclassifies at least one of the other 7 points (call it x'); suppose H correctly classifies $x = 1$, which implies $\beta_1 < \beta_2$, but this implies that H will misclassifies x' since the majority weight is at h_2.

(As long as the argument is correct, you will get 3 points.)

3.2 Training Error of AdaBoost

In this problem we make the following so-called "weak learning assumption": for every iteration of AdaBoost,

$$\epsilon_t \leq \tfrac{1}{2} - \gamma$$

for some constant $\gamma \in (0, \frac{1}{2})$. In other words, the base algorithm is always able to return a base classifier h_t with the weighted error rate γ smaller than that of random guessing. Under this assumption, you are going to prove that the training error of AdaBoost decreases exponentially fast, following the two steps below.

(a) Prove that for each t, the normalization factor Z_t satisfies:

$$Z_t \leq \sqrt{1 - 4\gamma^2}.$$

$$\begin{aligned}
Z_t &= \sum_{n=1}^{N} D_t(n) \exp(-\beta_t y_n h_t(\boldsymbol{x}_n)) \\
&= \sum_{n: y_n h_t(\boldsymbol{x}_n)=1} D_t(n) \exp(-\beta_t) + \sum_{n: y_n h_t(\boldsymbol{x}_n)=-1} D_t(n) \exp(\beta_t) \\
&= (1-\epsilon_t) \exp(-\beta_t) + \epsilon_t \exp(\beta_t) && \text{(1 point)} \\
&= 2\sqrt{(1-\epsilon_t)\epsilon_t} && \text{(1 point)} \\
&\leq 2\sqrt{(\tfrac{1}{2}+\gamma)(\tfrac{1}{2}-\gamma)} && \text{(2 points)} \\
&= \sqrt{1-4\gamma^2}
\end{aligned}$$

where the inequality holds due to the weak learning assumption and the fact that function $(1-\epsilon)\epsilon$ is increasing in $[0, 1/2]$ (and you have to mention this reasoning in order to get the 2 points for this step; arriving at this step with incorrect reasons or no explanation at all gets 1 point).

(b) It can be shown that the training error of AdaBoost is bounded as (you are not asked to prove this)

$$\frac{1}{N}\sum_{n=1}^{N}\mathbb{I}[H(\boldsymbol{x}_n) \neq y_n] \leq \prod_{t=1}^{T} Z_t.$$

Use this fact and the inequality $1 + z \leq e^z$ for all $z \in \mathbb{R}$ to prove the following

$$\frac{1}{N}\sum_{n=1}^{N}\mathbb{I}[H(\boldsymbol{x}_n) \neq y_n] \leq \exp(-2T\gamma^2), \tag{1}$$

which implies that the training error of AdaBoost decreases in an exponential manner.

Using the inequality $1 + z \leq e^z$ with $z = -4\gamma^2$ we have

$$Z_t \leq \sqrt{\exp(-4\gamma^2)} = \exp(-2\gamma^2), \qquad \text{(2 points)}$$

and therefore with the provided fact we have

$$\frac{1}{N}\sum_{n=1}^{N}\mathbb{I}[H(\boldsymbol{x}_n) \neq y_n] \leq \prod_{t=1}^{T}\exp(-2\gamma^2) = \exp(-2T\gamma^2) \qquad \text{(1 point)}$$

(c) Based on Eq. (1), derive the minimum number of rounds T_0 so that AdaBoost has zero training error as long as $T \geq T_0$.

The key is to observe that as long as the training error is strictly smaller than $1/N$, it actually means that the error rate has to be zero (because there are only N examples). Therefore based on Eq. (1) we want

$$\exp(-2T_0\gamma^2) < \frac{1}{N}$$

which means

$$T_0 = \left\lfloor \frac{\ln N}{2\gamma^2} \right\rfloor + 1.$$

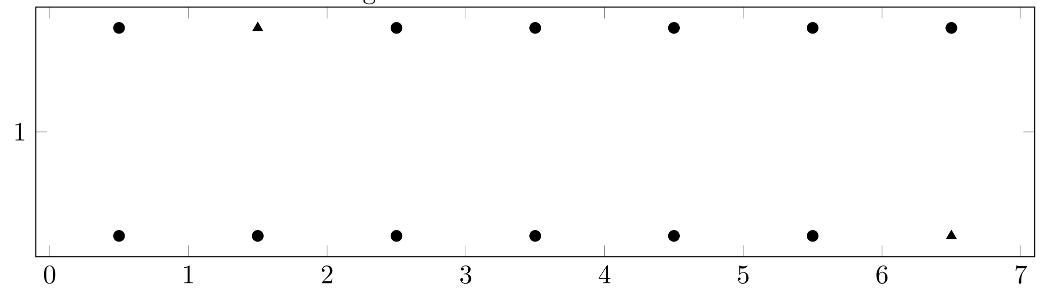
Rubrics: 1) giving the right argument gets 2 points. 2) giving the correct final answer gets another 2 points; giving something like $\lceil \frac{\ln N}{2\gamma^2} \rceil$, $\frac{\ln N}{2\gamma^2}$, or $\frac{\ln N}{2\gamma^2} \pm 1$ get the full 2 points too (even though they are not completely correct).

4 K-means (8 points)

Consider the following dataset. All points are unlabeled and part of the same set. The triangles are used to distinguish two points later. *Please do not draw on this diagram until you have read the problems below.*

Suppose we run the K-means algorithm on this dataset with $K = 2$ and the two points indicated by triangles as the initial centroids. When the algorithm converges, there will be two clearly separated clusters. Directly on Figure 2, draw a straight line that separates these two clusters, as well as the centroids of these two clusters.

Figure 2: An unlabeled dataset



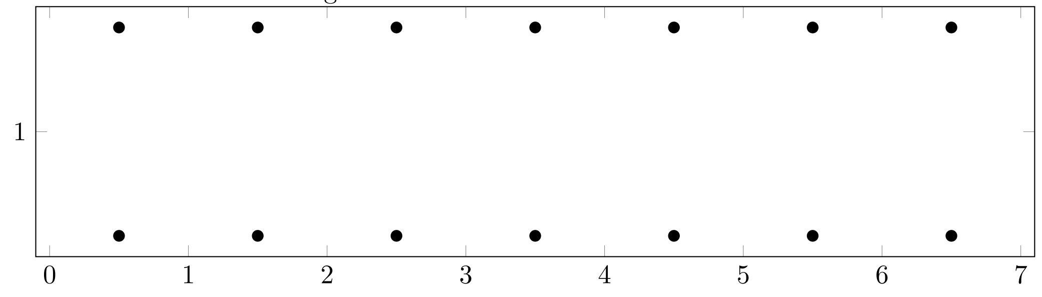
There are infinitely many choices for the separating line. Any vertical line between 3.5 and 4.5 would work for example. (2 points)

The two centroids are (2, 1) and (5.5, 1). (2 points)

Next, find two other different sets of initialize centroids that will converge to the exact same result if we apply K-means. Please follow the instructions below

- the initialize centroids have to be points of the dataset;
- directly on Figure 3, use two triangles to indicate the first set, and two squares to indicate the second;
- these two sets of points can overlap with each other, but of course cannot be the same;
- similarly these sets can overlap with the initialization of Figure 2 but cannot be the same;
- do not pick those that lead to ambiguous results due to different ways of breaking ties.

Figure 3: The same unlabeled dataset



There are many choices, for example (with y_b and y_t being the y-coordinate of the bottom and top points), $(2.5, y_b)$ and $(5.5, y_t)$, $(2.5, y_t)$ and $(5.5, y_b)$, $(2.5, y_t)$ and $(5.5, y_t)$, $(1.5, y_b)$ and $(6.5, y_b)$, and so on. Note that answers like $(1.5, y_b)$ and $(5.5, y_b)$ are not acceptable due to the ambiguousness from tie-breaking.

Two points for each set.

5 Generative Models (23 points)

5.1 Naive Bayes

Suppose we have the following training data. Each data point has three features (Weather, Emotion, Homework), where Weather $\in \{Sunny, Cloudy\}$, Emotion $\in \{Happy, Normal, Unhappy\}$, Homework $\in \{Much, Little\}$. The label PlayBasketball indicates whether it is suitable to play basketball. You are asked to build a naive Bayes classifier. Recall the naive Bayes assumption is

$$P(\text{Weather, Emotion, Homework} \mid \text{PlayBasketball}) =$$
$$P(\text{Weather} \mid \text{PlayBasketball}) \times P(\text{Emotion} \mid \text{PlayBasketball}) \times P(\text{Homework} \mid \text{PlayBasketball}).$$

Weather	Emotion	Homework	PlayBasketball
Sunny	Happy	Little	Yes
Sunny	Normal	Little	Yes
Cloudy	Happy	Much	Yes
Cloudy	Unhappy	Little	Yes
Sunny	Unhappy	Little	No
Cloudy	Normal	Much	No

(a) Write down the MLE for the following parameters (you only need to provide the final number):

- $P(\text{PlayBasketball} = No) =$
- $P(\text{Weather} = Cloudy \mid \text{PlayBasketball} = Yes) =$
- $P(\text{Emotion} = Unhappy \mid \text{PlayBasketball} = No) =$
- $P(\text{Homework} = Little \mid \text{PlayBasketball} = Yes) =$

The answers are $\frac{1}{3}, \frac{1}{2}, \frac{1}{2}$ and $\frac{3}{4}$ respectively (one point for each).

(b) Given a new data instance $\boldsymbol{x} = (\text{Weather} = \textit{Cloudy}, \text{Emotion} = \textit{Unhappy}, \text{Homework} = \textit{Little})$, compute $P(\text{PlayBasketball} = No \mid \boldsymbol{x})$ (show your work).

$$\begin{aligned}
&P(\text{PlayBasketball} = Yes \mid \boldsymbol{x}) \\
&\propto P(\text{PlayBasketball} = Yes, \boldsymbol{x}) \\
&= P(\text{PlayBasketball} = Yes) \times P(\text{Weather} = \textit{Cloudy} \mid \text{PlayBasketball} = Yes) \times \\
&\quad P(\text{Emotion} = \textit{Unhappy} \mid \text{PlayBasketball} = Yes) \times P(\text{Homework} = \textit{Little} \mid \text{PlayBasketball} = Yes) \\
&\hfill \text{(1 point)} \\
&= \tfrac{2}{3} \times \tfrac{1}{2} \times \tfrac{1}{4} \times \tfrac{3}{4} = \tfrac{3}{48} \qquad \text{(1 point)}
\end{aligned}$$

Similarly

$$\begin{aligned}
&P(\text{PlayBasketball} = No \mid \boldsymbol{x}) \\
&\propto P(\text{PlayBasketball} = No) \times P(\text{Weather} = \textit{Cloudy} \mid \text{PlayBasketball} = No) \times \\
&\quad P(\text{Emotion} = \textit{Unhappy} \mid \text{PlayBasketball} = No) \times P(\text{Homework} = \textit{Little} \mid \text{PlayBasketball} = No) \\
&\hfill \text{(1 point)} \\
&= \tfrac{1}{3} \times \tfrac{1}{2} \times \tfrac{1}{2} \times \tfrac{1}{2} = \tfrac{1}{24} \qquad \text{(1 point)}
\end{aligned}$$

Therefore

$$P(\text{PlayBasketball} = No \mid \boldsymbol{x}) = \frac{\frac{1}{24}}{\frac{1}{24} + \frac{3}{48}} = \frac{2}{5} \qquad \text{(1 point)}$$

5.2 Gaussian Mixture Models and EM

Let $X \in \mathbb{R}$ be a one-dimensional random variable distributed according to a mixture of two Gaussian distributions $\mathcal{N}(\mu_1, \sigma^2)$ and $\mathcal{N}(\mu_2, \sigma^2)$ (note that *the two distributions have the same variance*). In particular,

$$P(X = x) = \sum_{k=1}^{2} P(X = x, Z = k) = \sum_{k=1}^{2} P(Z = k) P(X = x | Z = k)$$
$$= \omega_1 \mathcal{N}(x|\mu_1, \sigma^2) + \omega_2 \mathcal{N}(x|\mu_2, \sigma^2),$$

where ω_1 and ω_2 are weights of mixture components such that $\omega_1 \geq 0,\ \omega_2 \geq 0,$ and $\omega_1 + \omega_2 = 1$. Recall that the Gaussian density is $\mathcal{N}(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-(x-\mu)^2}{2\sigma^2}\right)$, where μ is the mean and σ^2 is the variance of the Gaussian distribution.

Now suppose $x_1, x_2, \cdots, x_N$ are i.i.d. samples of this model. Derive the EM algorithm for this model by following the two steps below.

(a) E-Step: derive the expected (complete) log-likelihood explicitly, assuming

$$\boldsymbol{\theta}^{\text{OLD}} = \{\omega_1^{\text{OLD}}, \omega_2^{\text{OLD}}, \mu_1^{\text{OLD}}, \mu_2^{\text{OLD}}, \sigma^{\text{OLD}}\}$$

are the current estimates of the parameters. Recall the definition of the expected (complete) log-likelihood is the following:

$$Q(\boldsymbol{\theta}\,; \boldsymbol{\theta}^{\text{OLD}}) = \sum_{n=1}^{N} \sum_{k=1}^{2} \gamma_{nk} \ln P(X_n = x_n, Z_n = k\,; \boldsymbol{\theta}), \tag{2}$$

where

$$\gamma_{nk} = P(Z_n = k \mid X_n = x_n\,; \boldsymbol{\theta}^{\text{OLD}}).$$

and $\boldsymbol{\theta}$ is the set of parameters $\omega_1, \omega_2, \mu_1, \mu_2, \sigma$. To simplify your answer you can use the density function $\mathcal{N}$ without plugging in its formula.

By Bayes' rule

$$\gamma_{nk} = \frac{\omega_k^{\text{OLD}} \mathcal{N}(x_n \mid \mu_k^{\text{OLD}}, \sigma^{\text{OLD}})}{\omega_1^{\text{OLD}} \mathcal{N}(x_n \mid \mu_1^{\text{OLD}}, \sigma^{\text{OLD}}) + \omega_2^{\text{OLD}} \mathcal{N}(x_n \mid \mu_2^{\text{OLD}}, \sigma^{\text{OLD}})} \qquad \text{(2 points)}$$

So the expected complete log-likelihood $Q(\boldsymbol{\theta}\,; \boldsymbol{\theta}^{\text{OLD}})$ is

$$\sum_{n=1}^{N} \sum_{k=1}^{2} \frac{\omega_k^{\text{OLD}} \mathcal{N}(x_n \mid \mu_k^{\text{OLD}}, \sigma^{\text{OLD}})}{\omega_1^{\text{OLD}} \mathcal{N}(x_n \mid \mu_1^{\text{OLD}}, \sigma^{\text{OLD}}) + \omega_2^{\text{OLD}} \mathcal{N}(x_n \mid \mu_2^{\text{OLD}}, \sigma^{\text{OLD}})} \ln\left(\omega_k \mathcal{N}(x_n \mid \mu_k, \sigma^2)\right) \qquad \text{(2 points)}$$

(Deduct 1 point for each of the above two calculations if notation such as σ_1, σ_2 or σ_k is still used, which implies that the difference from the practice final is not noticed.)

(b) M-step: derive the updates of all the parameters $\omega_1, \omega_2, \mu_1, \mu_2$ and σ by maximizing $Q(\boldsymbol{\theta} ; \boldsymbol{\theta}^{\text{OLD}})$. You should use the notation γ_{nk} from the last question to simplify your answer.

Plugging in the Gaussian density we have

$$Q(\boldsymbol{\theta} ; \boldsymbol{\theta}^{\text{OLD}}) = \sum_{n=1}^{N} \sum_{k=1}^{2} \gamma_{nk} \left(\ln \omega_k + \ln \frac{1}{\sqrt{2\pi\sigma^2}} - \frac{(x_n - \mu_k)^2}{2\sigma^2} \right) \qquad \text{(1 point)}$$

The update for ω_k is simply

$$\omega_k = \frac{\sum_{n=1}^{N} \gamma_{nk}}{\sum_{n=1}^{N} \sum_{k=1}^{2} \gamma_{nk}} = \frac{\sum_{n=1}^{N} \gamma_{nk}}{N} \qquad \text{(1 point)}$$

(It is okay to omit the derivation here since we have seen this many times.)

Next setting the derivative w.r.t. μ_k gives

$$\sum_{n=1}^{N} \frac{\gamma_{nk}(x_n - \mu_k)}{\sigma^2} = 0 \qquad \text{(2 points)}$$

and therefore the update for μ_k is

$$\mu_k = \frac{\sum_{n=1}^{N} \gamma_{nk} x_n}{\sum_{n=1}^{N} \gamma_{nk}} \qquad \text{(2 points)}$$

Finally setting the derivative w.r.t. σ gives

$$\sum_{n=1}^{N} \sum_{k=1}^{2} \gamma_{nk} \left(-\frac{1}{\sigma} + \frac{(x_n - \mu_k)^2}{\sigma^3} \right) \qquad \text{(2 points)}$$

and solving for σ leads to the update

$$\sigma^2 = \frac{\sum_{n=1}^{N} \sum_{k=1}^{2} \gamma_{nk}(x_n - \mu_k)^2}{\sum_{n=1}^{N} \sum_{k=1}^{2} \gamma_{nk}} = \frac{\sum_{n=1}^{N} \sum_{k=1}^{2} \gamma_{nk}(x_n - \mu_k)^2}{N} \qquad \text{(2 points)}$$

(Please note the difference from the practice final where each cluster has a different variance parameter. No points will be given for the update of σ if this difference is not noticed.)

6 Hidden Markov Model (16 points)

Recall a hidden Markov model is parameterized by

- initial state distribution $P(Z_1 = s) = \pi_s$
- transition distribution $P(Z_{t+1} = s' \mid Z_t = s) = a_{s,s'}$
- emission distribution $P(X_t = o \mid Z_t = s) = b_{s,o}$

These parameters are assumed to be known for this problem.

6.1 Missing data

Suppose we observe a sequence of outcomes $x_1, \ldots, x_{t-1}, x_{t+1}, \ldots, x_T$ with the outcome at time t missing ($2 \le t \le T-1$). Derive the conditional probability of the *outcome at time t being some o*, that is,

$$P(X_t = o \mid X_{1:t-1} = x_{1:t-1}, X_{t+1:T} = x_{t+1:T}).$$

Express you answer in terms of the forward message at time $t-1$

$$\alpha_{s'}(t-1) = P(Z_{t-1} = s', X_{1:t-1} = x_{1:t-1}),$$

and the backward message at time t

$$\beta_{s'}(t) = P(X_{t+1:T} = x_{t+1:T} \mid Z_t = s').$$

By marginalization and conditional independence, we have

$$\begin{aligned}
&P(X_t = o \mid X_{1:t-1} = x_{1:t-1}, X_{t+1:T} = x_{t+1:T})\\
&= \sum_s P(X_t = o, Z_t = s \mid X_{1:t-1} = x_{1:t-1}, X_{t+1:T} = x_{t+1:T})\\
&= \sum_s P(X_t = o \mid Z_t = s) P(Z_t = s \mid X_{1:t-1} = x_{1:t-1}, X_{t+1:T} = x_{t+1:T})\\
&= \sum_s b_{s,o} P(Z_t = s \mid X_{1:t-1} = x_{1:t-1}, X_{t+1:T} = x_{t+1:T}). \qquad \text{(2 points)}
\end{aligned}$$

Now the last term is exactly what we derived in the practice final:

$$\begin{aligned}
&P(Z_t = s \mid X_{1:t-1} = x_{1:t-1}, X_{t+1:T} = x_{t+1:T})\\
&\propto P(Z_t = s, X_{1:t-1} = x_{1:t-1}, X_{t+1:T} = x_{t+1:T})\\
&= P(X_{t+1:T} = x_{t+1:T} \mid Z_t = s, X_{1:t-1} = x_{1:t-1}) P(Z_t = s, X_{1:t-1} = x_{1:t-1})\\
&= P(X_{t+1:T} = x_{t+1:T} \mid Z_t = s) \sum_{s'} P(Z_t = s, Z_{t-1} = s', X_{1:t-1} = x_{1:t-1})\\
&= \beta_s(t) \sum_{s'} P(Z_t = s \mid Z_{t-1} = s') P(Z_{t-1} = s', X_{1:t-1} = x_{1:t-1})\\
&= \beta_s(t) \sum_{s'} a_{s',s} \alpha_{s'}(t-1) \qquad \text{(3 points)}
\end{aligned}$$

Combining everything gives

$$P(X_t = o \mid X_{1:t-1} = x_{1:t-1}, X_{t+1:T} = x_{t+1:T}) = \sum_s b_{s,o} \left(\beta_s(t) \sum_{s'} a_{s',s} \alpha_{s'}(t-1) \right). \qquad \text{(1 point)}$$

6.2 Viterbi algorithm

Recall that the Viterbi algorithm starts with $\delta_s(1) = \pi_s b_{s,x_1}$ for each s, and then computes in a forward manner

$$\delta_s(t) \triangleq \max_{z_{1:t-1}} P(Z_t = s, Z_{1:t-1} = z_{1:t-1}, X_{1:t} = x_{1:t}) = b_{s,x_t} \max_{s'} a_{s',s} \delta_{s'}(t-1)$$

and

$$\Delta_s(t) \triangleq \operatorname*{argmax}_{z_{t-1}} \max_{z_{1:t-2}} P(Z_t = s, Z_{1:t-1} = z_{1:t-1}, X_{1:t} = x_{1:t}) = \operatorname*{argmax}_{s'} a_{s',s} \delta_{s'}(t-1)$$

for each s and $t = 2, \ldots, T$. Using these quantities, describe

(a) how to find the most likely hidden state path $z_1^*, \ldots, z_T^*$ given the entire observations $x_1, \ldots, x_T$.

(b) for an arbitrary $T_0 < T$, how to find the most likely hidden state path $z_1^*, \ldots, z_{T_0}^*$ given *the entire observations $x_1, \ldots, x_T$*. Feel free to use forward and backward messages if needed.

(a) The last state of the most likely path is

$$z_T^* = \operatorname*{argmax}_s \delta_s(T) \qquad \text{(2 points)}$$

the rest of the states $z_{T-1}^*, \ldots, z_1^*$ can be found in the backward manner:

$$z_{t-1}^* = \Delta_{z_t^*}(t) \qquad \text{(2 points)}$$

(b) Note that this last state $z_{T_0}^*$ is different from the situation in the practice final where the inference is conditioned on only the first T_0 observations, and is also *different from what you will find from Question (a) for time T_0*, but the backtracking step is the same. Specifically,

$$\begin{aligned}
z_{T_0}^* &= \operatorname*{argmax}_s \max_{z_{1:T_0-1}} P(Z_{T_0} = s, Z_{1:T_0-1} = z_{1:T_0-1}, X_{1:T} = x_{1:T}) && \text{(1 point)} \\
&= \operatorname*{argmax}_s \max_{z_{1:T_0-1}} P(Z_{T_0} = s, Z_{1:T_0-1} = z_{1:T_0-1}, X_{1:T_0} = x_{1:T_0}) \times \\
&\qquad P(X_{T_0+1,T} = x_{T_0+1:T} \mid Z_{T_0} = s, Z_{1:T_0-1} = z_{1:T_0-1}, X_{1:T_0} = x_{1:T_0}) \\
&= \operatorname*{argmax}_s \left(\max_{z_{1:T_0-1}} P(Z_{T_0} = s, Z_{1:T_0-1} = z_{1:T_0-1}, X_{1:T_0} = x_{1:T_0}) \right) P(X_{T_0+1,T} = x_{T_0+1:T} \mid Z_{T_0} = s) \\
& && \text{(1 point)} \\
&= \operatorname*{argmax}_s \ \delta_s(T_0) \beta_s(T_0) && \text{(2 points)}
\end{aligned}$$

where $\beta_s(T_0)$ is the backward message (and the only different term compared to the practice final).

The rest of the states can be found in the same way

$$z_{t-1}^* = \Delta_{z_t^*}(t) \qquad \text{(2 points)}$$

for $t = T_0, \ldots, 2$.

This blank page can be used as scratch paper.

This blank page can be used as scratch paper.