

CSCI 567 Fall 2021 Quiz One

Problem	1	2	3	4	5	Total
Max	30	12	20	24	14	100
Points						

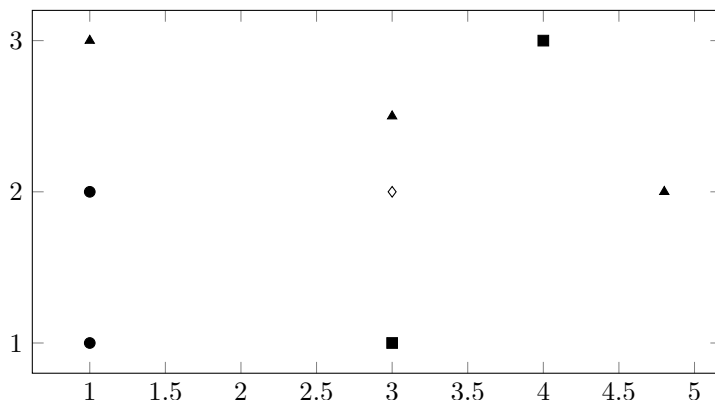
Please read the following instructions carefully:

- Use the same link for the lectures to join Zoom for the quiz (available on both the course website and the DEN website). You will be assigned to a breakout room. Each room has one proctor, and you have to **turn your camera on** during the entire quiz. If you finish early, you can submit your quiz and leave the meeting, but you cannot rejoin.
- This is an open-book/note quiz. Searching the Internet is technically not forbidden, but you are strongly encouraged to focus on solving the problems yourself. **Collaboration of any kind or consultation from others, however, is strictly prohibited.** Violation will lead to **0 score** for the entire quiz and be reported to SJACS.
- For all questions, write down your answers as you would normally do for the written assignments, and then submit a pdf/jpg/png for each of the 5 problems. Make sure that your writing is recognizable. Try to keep your solutions concise.
- Duration of the exam is 2.5 hours, which **already includes the time you need for scanning and uploading.** No extra time will be given if you fail to upload your solutions within the given 2.5 hours. Questions are **not ordered by their difficulty**, so budget your time for each question carefully.
- If you have any **clarification** questions during the quiz, make a private post on Piazza and it will be answered shortly.

1 Multiple-choice Questions (30 points)

IMPORTANT: Select *ALL* answers that you think are correct. You get 0.5 point for selecting each correct answer and similarly 0.5 point for not selecting each incorrect answer.

- (1) Which of the following on machine learning is correct?
- (A) Cross-validation is often used to tune the hyper-parameters of a machine learning algorithm.
 - (B) The goal of a machine learning algorithm is to achieve zero error on a training set.
 - (C) Regularization is a common way to prevent overfitting.
 - (D) Classification and regression are two common tasks in machine learning.
- (2) Consider the following two-dimensional dataset with $N = 7$ training points of three classes (triangle, square, and circle), and additionally one test point denoted by the diamond. Which of the following configuration of the K -nearest neighbor algorithm will predict triangle for the test point?



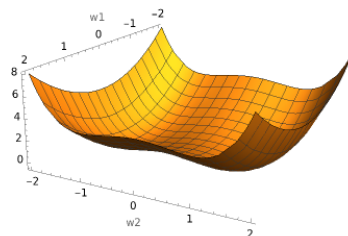
- (A) $K = 1$, L2 distance
 - (B) $K = 3$, L1 distance
 - (C) $K = 3$, L2 distance.
 - (D) $K = 7$, any distance.
- (3) Which of the following on linear regression is correct?
- (A) The least square solution has a closed-form formula, even if L2 regularization is applied.
 - (B) The covariance matrix $\mathbf{X}^T \mathbf{X}$ is not invertible if and only if the number of data points N is smaller than the dimension D .
 - (C) When the covariance matrix $\mathbf{X}^T \mathbf{X}$ is not invertible, the Residual Sum of Squares (RSS) objective has no minimizers.
 - (D) Linear regression is a parametric method, even when it is kernelized.
- (4) Which of the following on binary classification is correct?
- (A) Similarly to the Perceptron algorithm, logistic regression can also be kernelized.
 - (B) The Perceptron algorithm (with $\mathbf{0}$ initialization) is an instance of SGD where the learning rate $\eta > 0$ does not matter.
 - (C) One can apply either SGD or the Newton method to minimize the hinge loss.
 - (D) Minimizing 0-1 loss is NP-hard for every dataset, even if it is linearly separable.

- (5) Consider a training set $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)$ and a probabilistic model $\mathbb{P}(y_n | \mathbf{x}_n; \mathbf{w})$ which specifies for each n the probability of seeing outcome y_n given feature \mathbf{x}_n and parameter \mathbf{w} . Which of the following is the Maximum Likelihood Estimation (MLE) for \mathbf{w} ?

- (A) $\operatorname{argmax} \sum_{n=1}^N \mathbb{P}(y_n | \mathbf{x}_n; \mathbf{w})$ (B) $\operatorname{argmax} \prod_{n=1}^N \mathbb{P}(y_n | \mathbf{x}_n; \mathbf{w})$
 (C) $\operatorname{argmax} \sum_{n=1}^N \ln \mathbb{P}(y_n | \mathbf{x}_n; \mathbf{w})$ (D) $\operatorname{argmax} \prod_{n=1}^N \ln \mathbb{P}(y_n | \mathbf{x}_n; \mathbf{w})$

- (6) Consider a two-dimensional function $F(\mathbf{w}) = w_1^2 - w_2^2 + \frac{1}{2}w_2^4$ (a plot is provided below). Which of the following statement is correct?

- (A) F has three stationary points: $(0, 0)$, $(0, -1)$, and $(0, 1)$.
 (B) F has three local minimizers: $(0, 0)$, $(0, -1)$, and $(0, 1)$.
 (C) $(0, -1)$ is a local minimizer of F , so Gradient Descent always converges to this point.
 (D) $(0, 0)$ is a saddle point of F .



- (7) Machine learning objective is usually of the form $F(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^N F_n(\mathbf{w})$ for some loss function $F_n : \mathbb{R}^D \rightarrow \mathbb{R}$ corresponding to the n -th training point. We know that $\nabla F_n(\mathbf{w})$ for a uniformly at random chosen n is a stochastic gradient of this objective such that $\mathbb{E}[\nabla F_n(\mathbf{w})] = \nabla F(\mathbf{w})$. Which of the following is also a stochastic gradient?

- (A) $\sum_{n \in S} \nabla F_n(\mathbf{w})$ for a subset $S \subset \{1, \dots, N\}$ (of some fixed size) chosen uniformly at random.
 (B) $\frac{1}{|S|} \sum_{n \in S} \nabla F_n(\mathbf{w})$ for a subset $S \subset \{1, \dots, N\}$ (of some fixed size) chosen uniformly at random.
 (C) $\nabla F_n(\mathbf{w}) - \nabla F_n(\mathbf{w}_0) + \nabla F(\mathbf{w}_0)$ for an n chosen uniformly at random and some fixed point \mathbf{w}_0 .
 (D) $\frac{\partial F(\mathbf{w})}{\partial w_i} D\mathbf{e}_i$ for a coordinate i chosen uniformly at random ($\mathbf{e}_i \in \mathbb{R}^D$ is the standard basis vector with 1 in the i -th coordinate and 0 in all other coordinates).

- (8) Which of the following on reduction from multiclass classification to binary classification is correct?

- (A) One-versus-one is usually more robust than one-versus-all, but it is slower since it creates more binary training points.
 (B) Multiclass logistic regression was invented since its binary version cannot be combined with one-versus-one or other reductions.
 (C) A random code matrix is often a good choice for the Error-correcting Output Codes reduction.
 (D) Tree-based reduction is especially useful when the number of possible classes is huge.

- (9) Which of the following about neural nets is correct?

- (A) A neural net with a fixed architecture can represent any continuous function.
 (B) Dropout is useful for preventing overfitting when training a neural net.
 (C) A neural net with only ReLU activation is a convex model, since ReLU is a convex function.
 (D) A convolution layer is a special case of a fully connected layer.

- (10) Suppose a convolution layer takes a 4×6 image with 3 channels as input and outputs a $3 \times 4 \times 8$ volume. Which of the following is a possible configuration of this layer?
- (A) One 2×3 filter with depth 8, stride 1, and no zero-padding.
 - (B) Eight 2×3 filters with depth 3, stride 1, and no zero-padding.
 - (C) Eight 2×2 filters with depth 3, stride 2, and 1 pixel of zero-padding.
 - (D) Eight 2×2 filters with depth 3, stride 2, and 2 pixels of zero-padding.
- (11) 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 4 filters of size 3×3 (stride 1 and 1 pixel of zero-padding), then another convolution layer with 3 filters of size 2×2 (stride 2 and no zero-padding), and finally a max-pooling layer with a 2×2 filter (stride 2 and no zero-padding). (Note: the depth of all filters are not explicitly spelled out, and we assume no bias/intercept terms used.)
- (A) 48 (B) 144 (C) 156 (D) 168
- (12) What is the final output dimension of the last question?
- (A) $2 \times 2 \times 3$ (B) $4 \times 4 \times 3$ (C) $2 \times 2 \times 1$ (D) $4 \times 4 \times 1$
- (13) Which of the following on kernel is correct?
- (A) A machine learning algorithm can be kernelized if it only uses the feature vectors through their inner products.
 - (B) A Gram/kernel matrix must be positive semidefinite.
 - (C) The product of two kernel functions is still a kernel function.
 - (D) $k(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\|_1$ is not a kernel function.
- (14) Which of the following on SVM is correct?
- (A) SVM tries to find a hyperplane with maximum margin, and thus it can only be applied to linearly separable data.
 - (B) The primal formulation of SVM minimizes L2 regularized hinge loss.
 - (C) A support vector must be a correctly classified point according to complementary slackness.
 - (D) It is possible that a support vector does not satisfy the hard-margin constraint.
- (15) Consider a training set of N data points $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N) \in \mathbb{R}^D \times \{-1, +1\}$ and a nonlinear mapping $\phi : \mathbb{R}^D \rightarrow \mathbb{R}^\infty$ with a corresponding kernel function $k(\cdot, \cdot)$. Given the solution $\alpha_1^*, \dots, \alpha_N^*$ of the SVM dual formulation for this training set (with a hyper-parameter C discussed in the lecture), which of the following is a correct way to make a prediction on a test point \mathbf{x} ? (S below is the set of support vectors $\{n \in [N] : \alpha_n^* > 0\}$ and S' is a subset $\{n \in S : \alpha_n^* < C\}$.)

- (A) First compute $\mathbf{w}^* = \sum_{m \in S} \alpha_m^* y_m \phi(\mathbf{x}_m)$, then compute $b^* = y_n - \mathbf{w}^{*\top} \phi(\mathbf{x}_n)$ with any $n \in S'$, and finally predict with $\text{SGN}(\mathbf{w}^{*\top} \mathbf{x} + b^*)$.
- (B) First compute $b^* = y_n - \sum_{m \in S'} \alpha_m^* y_m k(\mathbf{x}_m, \mathbf{x}_n)$ with any $n \in S'$, then predict with

$$\text{SGN} \left(\sum_{m \in S} \alpha_m^* y_m k(\mathbf{x}_m, \mathbf{x}) + b^* \right). \quad (1)$$

- (C) First compute $b^* = y_n - \sum_{m \in S} \alpha_m^* y_m k(\mathbf{x}_m, \mathbf{x}_n)$ with any $n \in S'$, then predict with Eq. (1).
- (D) First compute $b^* = \frac{1}{|S'|} \sum_{n \in S'} (y_n - \sum_{m \in S} \alpha_m^* y_m k(\mathbf{x}_m, \mathbf{x}_n))$, then predict with Eq. (1).

2 GD/SGD for Linear Regression (12 points)

- (a) Least square solution is defined as $\mathbf{w}^* = \operatorname{argmin}_{\mathbf{w} \in \mathbb{R}^D} F(\mathbf{w})$ where

$$F(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^N (\mathbf{w}^T \mathbf{x}_n - y_n)^2$$

for a training set $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N) \in \mathbb{R}^D \times \mathbb{R}$. Although we know that \mathbf{w}^* has a closed-form, computing it requires inverting the $D \times D$ covariance matrix, which could be computationally expensive for large D . Instead, we can apply GD to approximately minimize $F(\mathbf{w})$. Fill in the missing details in the following implementation of this idea (no reasoning required). (3 points)

Algorithm 1: GD for minimizing RSS

- 1 **Input:** A training set $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)$, learning rate η
 - 2 **Initialization:** $\mathbf{w} = \mathbf{0} \in \mathbb{R}^D$
 - 3 **Repeat:**
|
-

- (b) To speed up the algorithm further when N is large, we can instead apply SGD. Fill in the missing details in the following implementation of this idea (no reasoning required). (4 points)

Algorithm 2: SGD for minimizing RSS

- 1 **Input:** A training set $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)$, learning rate η
 - 2 **Initialization:** $\mathbf{w} = \mathbf{0} \in \mathbb{R}^D$
 - 3 **Repeat:**
|
-

- (c) To avoid over-fitting, we consider minimizing the regularized version of F :

$$F_\lambda(\mathbf{w}) = \left(\sum_{n=1}^N (\mathbf{w}^T \mathbf{x}_n - y_n)^2 \right) + \lambda \|\mathbf{w}\|_2^2.$$

Fill in the missing details in the following algorithm that approximately minimizes F_λ using SGD (no reasoning required). (5 points)

Algorithm 3: SGD for minimizing regularized RSS

- 1 **Input:** A training set $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)$, learning rate η , regularization coefficient λ
 - 2 **Initialization:** $\mathbf{w} = \mathbf{0} \in \mathbb{R}^D$
 - 3 **Repeat:**
|
-

3 Linear Classifiers

(20 points)

In Lecture 3 we have seen the hinge loss $\ell(z) = \max\{0, 1 - z\}$, which is non-differentiable at $z = 1$. To avoid this issue, we can consider the square of hinge loss $\ell(z)^2$, which is differentiable everywhere. More specifically, given a binary dataset $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N) \in \mathbb{R}^D \times \{-1, 1\}$, we define the following new loss function for a linear model $\mathbf{w} \in \mathbb{R}^D$:

$$F(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^N F_n(\mathbf{w}), \quad \text{where } F_n(\mathbf{w}) = (\max\{0, 1 - y_n \mathbf{w}^T \mathbf{x}_n\})^2. \quad (2)$$

- (a) For a fixed n , write down the gradient $\nabla F_n(\mathbf{w})$ (show your derivation), then fill in the missing details in the repeat-loop of the algorithm below which applies SGD to minimize F . (5 points)

Algorithm 4: SGD for minimizing Eq. (2)

- 1 **Input:** A training set $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N) \in \mathbb{R}^D \times \{-1, 1\}$, learning rate $\eta > 0$
- 2 **Initialization:** $\mathbf{w} = \mathbf{0}$
- 3 **Repeat:**

|

- (b) Now kernelize Algorithm 4 using a kernel function $k(\cdot, \cdot)$ (with a corresponding feature mapping ϕ). Specifically, fill in the missing details in the repeat-loop of the algorithm below which maintains and updates weights $\alpha_1, \dots, \alpha_N$ such that $\mathbf{w} = \sum_{n=1}^N \alpha_n \phi(\mathbf{x}_n)$ is always the same as what one would get by running Algorithm 4 with \mathbf{x}_n replaced by $\phi(\mathbf{x}_n)$ for all n . (No reasoning is required.) (5 points)

Algorithm 5: Kernelized version of Algorithm 4

- 1 **Input:** A training set $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N) \in \mathbb{R}^D \times \{-1, 1\}$, learning rate $\eta > 0$, kernel $k(\cdot, \cdot)$
- 2 **Initialization:** $\alpha_1 = \dots = \alpha_N = 0$
- 3 **Repeat:**

|

- (c) Continuing from Question (a) (and forgetting about kernel from Question (b)), we now want to generalize the method to multiclass classification with C classes. Instead of coming up with a multiclass version of Eq. (2), we will apply the one-versus-all approach to learn C weight vectors $\mathbf{w}_1, \dots, \mathbf{w}_C$, each obtained by properly running Algorithm 4 for T iterations. Based on this information, fill in the missing details in the repeat-loop of the algorithm below. (No reasoning is required.) (6 points)

Algorithm 6: One-versus-all applied to Algorithm 4

```

1 Input: A training set  $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N) \in \mathbb{R}^D \times [C]$ , learning rate  $\eta > 0$ , iteration number  $T$ 
2 Initialization:  $\mathbf{w}_1 = \dots = \mathbf{w}_C = \mathbf{0}$ 
3 for  $c = 1, \dots, C$  do
4     Repeat for  $T$  iterations: ▷ a loop to update  $\mathbf{w}_c$ 

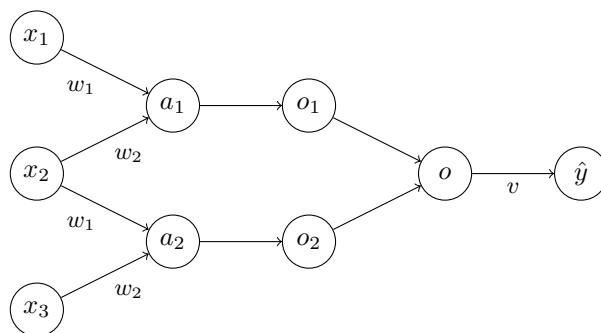
```

- (d) After running Algorithm 6, state how you would make a prediction for a test point \mathbf{x} based on what we discussed in the lecture for one-versus-all. (2 points)

- (e) Continuing from Question (d), can you propose another reasonable (and potentially better) way to make a prediction? Simply write down your proposal with no reasoning needed. (2 points)

4 Backpropagation for CNN (24 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, an average-pooling layer, and finally a fully connected layer with weight v .



More concretely, the computation is specified by

$$\begin{aligned}
 a_1 &= x_1 w_1 + x_2 w_2 \\
 a_2 &= x_2 w_1 + x_3 w_2 \\
 o_1 &= \max\{0, a_1\} \\
 o_2 &= \max\{0, a_2\} \\
 o &= (o_1 + o_2)/2 \\
 \hat{y} &= ov
 \end{aligned}$$

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

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

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

- (a) Write down $\frac{\partial \ell}{\partial v}$ (show the intermediate steps that use chain rule). You can use the sigmoid function $\sigma(z) = \frac{1}{1+e^{-z}}$ to simplify your notation. (4 points)

- (b) Write down $\frac{\partial \ell}{\partial w_1}$ and $\frac{\partial \ell}{\partial w_2}$ (show the intermediate steps that use chain rule). The derivative of the ReLU function is $H(a) = \mathbb{I}[a > 0]$, which you can use directly in your answer. (12 points)

- (c) Using the derivations above, fill in the missing details of the repeat-loop of the Backpropagation algorithm below that is used to train this mini CNN. (8 points)

Algorithm 7: Backpropagation for the above mini CNN

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

2 Initialize: set w_1, w_2, v randomly

3 Repeat:

4 randomly pick an example (\mathbf{x}_n, y_n)

5 Forward propagation:

6 Backward propagation:

5 Nonlinear Mappings and Kernel Methods (14 points)

Consider the following Gaussian/RBF kernel

$$k(\mathbf{x}, \mathbf{x}') = \exp\left(\frac{-\|\mathbf{x} - \mathbf{x}'\|_2^2}{2}\right). \quad (3)$$

It is known that there exists an infinite-dimensional nonlinear mapping ϕ_{RBF} such that

$$\phi_{\text{RBF}}(\mathbf{x})^T \phi_{\text{RBF}}(\mathbf{x}') = k(\mathbf{x}, \mathbf{x}') \quad (4)$$

for any \mathbf{x} and \mathbf{x}' . In this problem, you will investigate a way to approximate this nonlinear mapping ϕ_{RBF} .

- (a) Consider a nonlinear mapping $\phi_{\mathbf{v},b} : \mathbb{R}^D \rightarrow \mathbb{R}$ constructed as follows: randomly draw a vector $\mathbf{v} \in \mathbb{R}^D$ from the standard Gaussian and a scalar b from the uniform distribution over $[0, \pi]$, then define $\phi_{\mathbf{v},b}(\mathbf{x}) = \sqrt{2} \cos(\mathbf{v}^T \mathbf{x} + b)$ for any input feature vector $\mathbf{x} \in \mathbb{R}^D$.

For any two feature vectors \mathbf{x} and \mathbf{x}' , prove the following

$$\mathbb{E}[\phi_{\mathbf{v},b}(\mathbf{x}) \phi_{\mathbf{v},b}(\mathbf{x}')] = k(\mathbf{x}, \mathbf{x}') \quad (5)$$

where the expectation is over the randomness of \mathbf{v} and b , and $k(\cdot, \cdot)$ is defined in Eq. (3). You can directly use the following two identities in your proof:

- trigonometric identity: $2 \cos(\alpha) \cos(\beta) = \cos(\alpha - \beta) + \cos(\alpha + \beta)$;
- integral identity: $\mathbb{E}[\cos(\mathbf{v}^T \mathbf{z})] = \exp\left(\frac{-\|\mathbf{z}\|_2^2}{2}\right)$ where the expectation is with respect to \mathbf{v} randomly drawn from the standard Gaussian. (With this, you do not even need to know what the standard Gaussian is to solve this problem.)

(5 points)

- (b) Comparing Eq. (4) and Eq. (5), we see that $\phi_{\mathbf{v},b}$ can be used as an approximation for ϕ_{RBF} . However, using only one sample (\mathbf{v}, b) leads to large variance for this approximation. Based on this information, for any given dimension $M > 1$, can you come up with a random nonlinear mapping $\phi : \mathbb{R}^D \rightarrow \mathbb{R}^M$, such that it is a better approximation of ϕ_{RBF} satisfying $\mathbb{E} [\phi(\mathbf{x})^T \phi(\mathbf{x}')] = k(\mathbf{x}, \mathbf{x}')$? Write down your proposal, prove $\mathbb{E} [\phi(\mathbf{x})^T \phi(\mathbf{x}')] = k(\mathbf{x}, \mathbf{x}')$, and finally explain why it is a better approximation (in one concise sentence). (5 points)

- (c) As discussed in Lecture 5, in RBF-kernelized linear regression with training set $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)$, we maintain a weight vector $\boldsymbol{\alpha} = (\mathbf{K} + \lambda \mathbf{I})^{-1} \mathbf{y} \in \mathbb{R}^N$, where $\mathbf{K} \in \mathbb{R}^{N \times N}$ is the Gram matrix (such that $K_{n,m} = k(\mathbf{x}_n, \mathbf{x}_m)$), $\lambda > 0$ is the regularization coefficient, and $\mathbf{y} = (y_1, \dots, y_N)^T$ is the response vector. For a test point \mathbf{x} , we make a prediction via $\sum_{n=1}^N \alpha_n k(\mathbf{x}_n, \mathbf{x})$. While powerful, this method can be computationally expensive when N is huge.

Based on the nonlinear mapping you proposed in the last question for M much smaller than N , describe how you can approximate the kernelized linear regression described above with a much better time and space complexity. You only need to describe what quantities your method maintains, and how it makes a prediction for a test point. (4 points)