Q1. Generalization

We consider the following different classifiers for classification of samples in a 2-dimensional feature space.

**PNoBias** Linear perceptron *without* a bias term (features \([x_1 \ x_2]^T\))

**PBias** Linear perceptron with a bias term (features \([1 \ x_1 \ x_2]^T\))

**PQuad** Kernel perceptron with the quadratic kernel function \(K(x, z) = (1 + x \cdot z)^2\)

**PCutoff** Kernel perceptron with the kernel function \(K(x, z) = \max\{0, 0.01 - ||x - z||_2\}\) (||a - b||_2 is the Euclidean distance between a and b)

**1NN** 1-nearest neighbor classifier

**3NN** 3-nearest neighbor classifier

(a) In each of the plots below you are given points from two classes, shown as filled rectangles and open circles. For each plot, fill in the bubble next to each classifier that will be able to perfectly classify all of the training data (or, if none, mark “None of these will classify the data perfectly”).

Note that when computing the nearest neighbors for a training data point, the training data point will be its own nearest neighbor.

- PNoBias
- PBias
- PQuad
- PCutoff
- 1NN
- 3NN
- None of these will be able to classify the training data perfectly.
(b) Suppose you are given the choice between using the normal perceptron algorithm, which directly works with \( \phi(x) \), and the dual (kernelized) perceptron algorithm, which does not explicitly compute \( \phi(x) \) but instead works with the kernel function \( K \). Keeping space and time complexities in consideration, when would you prefer using the kernelized perceptron algorithm over the normal perceptron algorithm.

*Note: Here \( N \) denotes the total number of training samples and \( d \) is the dimensionality of \( \phi(x) \).*

- \( d \gg N \)
- \( d \ll N \)
- Always
- Never

(c) (i) Suppose you train a classifier and test it on a held-out validation set. It gets 80% classification accuracy on the training set and 20% classification accuracy on the validation set.

From what problem is your model most likely suffering?

- Underfitting
- Overfitting

Fill in the bubble next to any measure of the following which could reasonably be expected to improve your classifier’s performance on the validation set.

- Add extra features
- Remove some features

Briefly justify:

- Collect more training data
- Throw out some training data

Assuming features are outcome counts (\( k \) is the Laplace smoothing parameter controlling the number of extra times you “pretend” to have seen an outcome in the training data):

- Increase \( k \)
- Decrease \( k \) (assuming \( k > 0 \) currently)

Assuming your classifier is a Bayes’ net:

- Add edges
- Remove edges

(ii) Suppose you train a classifier and test it on a held-out validation set. It gets 30% classification accuracy on the training set and 30% classification accuracy on the validation set.

From what problem is your model most likely suffering?

- Underfitting
- Overfitting

Fill in the bubble next to any measure of the following which could reasonably be expected to improve your classifier’s performance on the validation set.

- Add extra features
- Remove some features

Briefly justify:

- Collect more training data
- Throw out some training data
Q2. Clustering

In this question, we will do \( k \)-means clustering to cluster the points \( A, B \ldots F \) (indicated by \( \times \)'s in the figure on the right) into 2 clusters. The current cluster centers are \( P \) and \( Q \) (indicated by the ■ in the diagram on the right).

Recall that \( k \)-means requires a distance function. Given 2 points, \( A = (A_1, A_2) \) and \( B = (B_1, B_2) \), we use the following distance function \( d(A, B) \) that you saw from class,

\[
d(A, B) = (A_1 - B_1)^2 + (A_2 - B_2)^2
\]

(a) Update assignment step: Select all points that get assigned to the cluster with center at \( P \):

- A  B  C  D  E  F  No point gets assigned to cluster P

(b) Update cluster center step: What does cluster center \( P \) get updated to?

Changing the distance function: While \( k \)-means used Euclidean distance in class, we can extend it to other distance functions, where the assignment and update phases still iteratively minimize the total (non-Euclidian) distance. Here, consider the Manhattan distance:

\[
d'(A, B) = |A_1 - B_1| + |A_2 - B_2|
\]

We again start from the original locations for \( P \) and \( Q \) as shown in the figure, and do the update assignment step and the update cluster center step using Manhattan distance as the distance function:

(c) Update assignment step: Select all points that get assigned to the cluster with center at \( P \), under this new distance function \( d'(A, B) \).

- A  B  C  D  E  F  No point gets assigned to cluster P

(d) Update cluster center step: What does cluster center \( P \) get updated to, under this new distance function \( d'(A, B) \)?
Q3. Decision Trees

You are given points from 2 classes, shown as +’s and ·’s. For each of the following sets of points,

1. Draw the decision tree of depth at most 2 that can separate the given data completely, by filling in binary predicates (which only involve thresholding of a single variable) in the boxes for the decision trees below. If the data is already separated when you hit a box, simply write the class, and leave the sub-tree hanging from that box empty.

2. Draw the corresponding decision boundaries on the scatter plot, and write the class labels for each of the resulting bins somewhere inside the resulting bins.

If the data can not be separated completely by a depth 2 decision tree, simply cross out the tree template. We solve the first part as an example.