A math equations with black text

Description automatically generated with medium confidence

KNN:

A k-Nearest Neighbor model with a large value of k is analogous to... a short Decision Tree with a low branching factor.

For kNN: Increasing k reduces sensitivity to noise, decreases prediction time.

Increasing k leads to higher bias, lower variance.

Using Euclidean distance as the distance measure, the decision boundary of k-NN for k = 1 is a piece-wise straight line because 1-NN decision boundary always follows a boundary equidistant from the two closest points in that region.

Despite k=1 having lowest training error, we would use validation dataset (validation error) to choose k, since k is a hyperparameter.

k-NN works great with a small amount of data but struggles when the amount of data becomes large (high runtime).

Lower k is more prone to overfitting.

kNN can be used for regression by average the regression value of the nearest neighbors.

Larger k tends to give a smoother decision boundary.

Increasing k reduces impact of noise/outliers.

Can use cross-val to select value of k. Select k-val that minimizes error on Dval.

Cannot be overfit by making k too large (k becomes maj vote eventually).

Model Selection:

A classifier trained until convergence on some training data and tested on a separate test set. The test error is very high. The training error is close to 0. This is Overfitting. Increasing training data size and decreasing model complexity should help.

A diagram of a curve

Description automatically generated

Curve 2 is the training error. The gap between curves represents overfitting.

Increasing: max dec. tree depth; max DT # of nodes, increases overfitting chances.

Increasing: MI Split threshold, k in kNN, DT Train set size decreases overfitting chances.

Increasing train data size for 1NN when data represents true dist. Decreases overfitting chances.

Cross validation: increasing % of data used for validation gives better estimate of performance on held-out data.

Decision trees:

Decision tree boundaries do not have to be linear.

For small values of max depth, decision trees are more likely to underfit the data than perceptron trees. For smaller values of max depth, decision trees are basically majority-vote classifiers at the leaves.

Decision trees: Non-linear decision boundaries

Perceptron: Ability to gracefully handle unseen attribute values in training data/better generalization at leaf nodes

Given an input feature vector where x ∈ Rn, y = 1 or y = −1. You have no info about the distributions of x and of y (not guaranteed to be linear). k-Nearest Neighbors and Decision Tree with unlimited depth, work, since these two methods do not assume linear separation.

One advantage of decision trees is that they are not easy to overfit -FALSE.

Perceptron:

If multiplying all x vectors in dataset by constant c:

Mistake bound is unchanged (radius and margin scaled by same amt). Classifier is unchanged (after adjusting by constant c), algorithm still converges.

If a dataset is linearly separable, convergence time does not depend on sample size n- max # mistakes determined only by radius and margin.

Perceptron biases: True boundary is linear, prefer to correct recent mistakes.

NOT a perc. Bias: prefer simplest hypothesis that explains data.

Even if training data is linearly separable, perceptron will not always find optimal boundary (training data may not represent true distribution).

Batch perceptron alg:

To evaluate point during training: if θTx + b >= 0, prediction = +1/yes/positive

If we misclassify a negative label, w = w - x. b = b - 1.

Misclassify positive label: w = w + x. b = b + 1.

Guess right, no change.

If all points are classified correctly, no need to update perceptron.

Only mistakes affect perceptron weights.

A table with numbers and symbols

Description automatically generated

[18, 30, 63, 61]

b = (10+5+8)-(2+3) = 18

theta1 = (2\*10+5\*5+1\*8)-(7\*2+3\*3) theta2, 3, etc.

To calculate margin: projection of nearest point p to boundary θ; p = [a,b,c]

A math equations with numbers and symbols

Description automatically generated with medium confidence

R is calculated as furthest point from origin.

Mistake bound = (R/y)2 (radius/margin). Use b in all calculations if folded in.

Scaling pairwise distance between points will not change mistake bound.

Increasing margin w/out changing anything else will reduce mistake bound.

Regression:

θ = (XT X)−1XTy

X is an n x m+1 matrix (with bias folded in)

Y is nx1

θ is 6x1.

Where n = num examples, m = num features.

MSE Obj. function:

Red text on a white background

Description automatically generated

Example:

A number of numbers on a white background

Description automatically generated



A white board with red writing on it

Description automatically generated

All ^^ x2 (so final ans. = 209.2)

A white board with red writing on it

Description automatically generated

Repeated w.r.t. b (multiply all by 2 again)^

Gradient formula (for MSE):