03 – knn and svm rbf

training errors does not matter we look at the mean\_cross\_validation\_error more (given two same cross\_validation\_errors, we pick the model that is simpler / less depth in the case

of a decision tree)

k-nn:

~ k-nn selects no feature at all, it's gonna use up all features to calculate the euclidean tree (The decision tree on the other hand may be working at some small sets of a feature) (E.g. a tree stump (with depth = 1) has one split and 1 good feature)

~ training error is 0 when k = 1 (always find the closest one which is itself!)

~ We can also use knn to do regression problem (take the average of k neighbours)

Pros of k-NNs for supervised learning

~ Easy to understand, interpret.

~ Simple hyperparameter k (n\_neighbors) controlling the fundamental tradeoff.

~ Can learn very complex functions given enough data.

~ Lazy learning: Takes no time to fit

Cons of k-NNs for supervised learning

~ Can be potentially be VERY slow during prediction time, especially when the training set is very large.

~ Often not that great test accuracy compared to the modern approaches.

~ It does not work well on datasets with many features or where most feature values are 0 most of the time (sparse datasets).

**If you need to store at least O(n) worth of stuff to make predictions, then it’s non-parametric.**

parametric: stump tree

non-parametric: knn

Curse of dimensionality (More features might result in bad results for k-nn!):

~ Affects all learners but especially bad for nearest-neighbour.

~ k-NN usually works well when the number of dimensions d is small but things fall apart quickly as d goes up.

~ If there are many irrelevant attributes, k-NN is hopelessly confused because all of them contribute to finding similarity between examples.

~ With enough irrelevant attributes the accidental similarity swamps out meaningful similarity and k-NN is no better than random guessing.

SVM RBF (Support Vector Machine with RBF kernel)

Superficially, SVM RBFs are more like weighted k-NNs.

~ The decision boundary is defined by a set of positive and negative examples and their weights together with their similarity measure.

~ A test example is labeled positive if on average it looks more like positive examples than the negative examples.

The primary difference between k-NNs and SVM RBFs is that

~ Unlike k-NNs, SVM RBFs only remember the key examples (support vectors). So it’s more efficient than k-NN.

~ SVMs use a different similarity metric which is called a “kernel” in SVM land. A popular kernel is Radial Basis Functions (RBFs) They usually perform better than k-NNs!

Support vectors¶

~ Each training example either is or isn’t a “support vector”.

~ This gets decided during fit.

~ Main insight: the decision boundary only depends on the support vectors.

Key hyperparameters of rbf SVM are

~ gamma (larger gamma → more complex) gamma = [0.001, 0.01, 0.1, 1.0, 10.0]

~ C (larger C → more complex) C = [0.1, 1.0, 100.0, 1000.0, 100000.0]