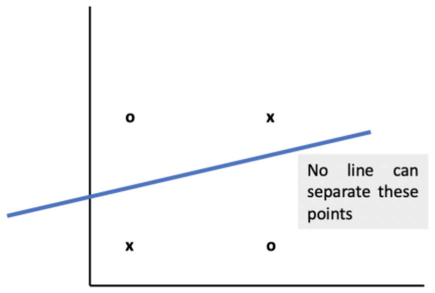
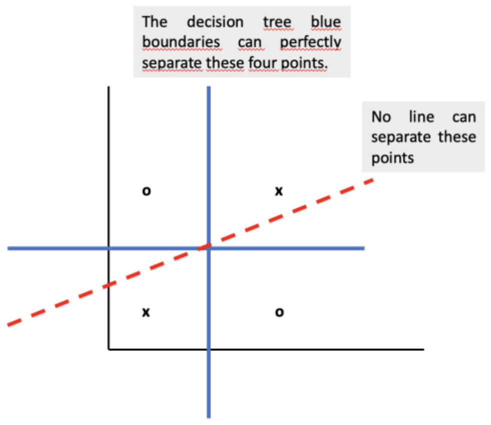
Q&A session – Week 39: Practice Questions with solutions

Q1: Draw a 2-dimensional real-valued dataset with two classes that can be classified with 100% accuracy by a 1-NN classifier, but not by a linear SVM. Explain briefly why this holds for your drawn dataset.

The 1-NN classifier will by definition obtain 0 classification error and perfectly classify any dataset. However, a linear SVM will fail for the following configuration.



Q2: Draw a 2-dimensional real-valued dataset with two classes that can be classified with 100% accuracy by a decision tree, but not by a Perceptron. Explain briefly why this holds for your drawn dataset.


The decision tree blue boundaries can perfectly separate these four points.
No line can separate these points
o
x
x
o

Q3: Consider a split criterion for decision trees, which favors splits resulting in groups with as evenly distributed classes as possible. What will be the effect on the resulting decision trees compared to using the information gain criterion? How would trees generated with the suggested criterion be expected to perform in terms of accuracy on independent test examples?

The split criterion is performing the opposite of that of information gain. It will tend to generate evenly distributed class labels within each tree node. The leaf nodes will end up having 50% training examples of the positive class and 50% of the negative class. Hence, we would expect the accuracy of the decision tree to be as good as random guessing.

Q4: Assume we are trying to learn a decision tree. Our input data consists of N samples, each with k attributes ($N \gg k$). We define the depth of a tree as the maximum number of nodes between the root and any of the leaf nodes (including the leaf, not the root).

1. If all attributes are binary, what is the maximum number of leaf (decision) nodes that we can have in a decision tree for this data? What is the maximum possible depth of a decision tree for this data?

   If the attributes are binary, in the worst case, we will perform a binary split for each and every attribute in each path of the tree. Moreover, in the worst case each attribute may appear in each path once. Hence, this leads to 2^k leaf nodes.

   The maximum depth of the tree is k (equal to the number of attributes per path in the tree) plus the final leaf node of the path.

2. If all attributes are continuous, what is the maximum number of leaf nodes that we can have in a decision tree for this data? What is the maximum possible depth for a decision tree for this data?

   In the worst case, we will end up splitting using the same attribute several times on the same path until we run out of examples. This means that the number of leaf nodes is N, i.e., as many as the examples. The tree depth is N-1, since each split will leave one example out per level, while the last level will have two examples.

Q5: Discuss the importance of the kernel trick for training an SVM.

The main idea behind the kernel trick when training an SVM is to define a function for computing the dot product of two examples in the φ space without actually having to project these examples.

Q6: Consider the following dataset of six examples (people) and three categorical attributes (the city where they live, their profession, and their nationality). Explain how would you use *cluster aggregation* for clustering the following dataset, and provide the final clustering.

U	*City*	*Profession*	*Nationality*
x_1	New York	Doctor	U.S.
x_2	New York	Teacher	Canada
x_3	Boston	Doctor	U.S.
x_4	Boston	Teacher	Canada
x_5	Los Angeles	Lawer	Mexican
x_6	Los Angeles	Actor	Mexican

Each feature (City, Profession, Nationality) can be considered as a result of a clustering algorithm. Then running clustering aggregation on the three features will produce the consensus clustering, which will also be the final clustering of the six examples.

Q7: Consider a transactional database, with A, B, and C being the possible items that can occur in the database. Suppose that we have mined all *frequent closed* itemsets in the datavase with a minimum support count threshold *min_sup = 2*.

These itemsets are:

{A, B, C}	with support count = 2
{A, B}	with support count = 4,
{B, C}	with support count = 3,
{B}	with support count = 5.

Using only this information, infer the *remaining frequent itemsets* in the database and their *support count* values. Provide a brief justification of your answer.

{A}:	4 – closest subset {A, B} with support 4.
{C}:	3 – closest subset {B, C} with support 3.
{A, C}:	2 – closest subset {A, B, C} with support 2.