CO395 - Introduction to Machine Learning (70050)

Week 2 (Introduction to ML)

Week 3 (Instance-based Learning + Decision Trees)

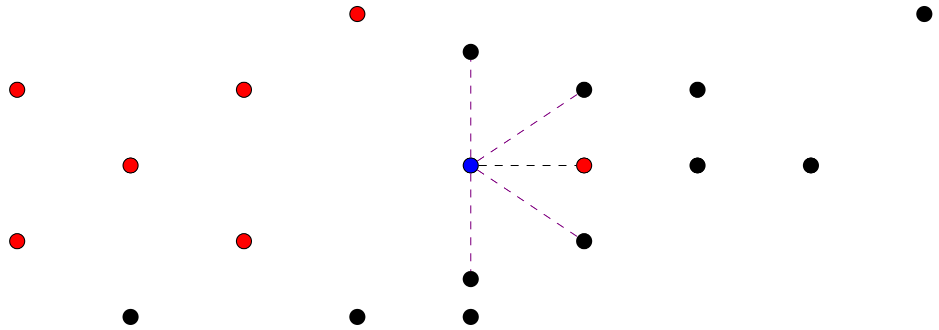
The **k Nearest Neighbours (k-NN)** classifier is classified as a **lazy learner**. A lazy learner stores all the training examples in the data set, and postpone any processing until a request is made (such as a prediction). On the other hand, **decision trees** are classified as a **eager learner**. An eager learner will attempt to construct a general target decision function, which is prepared prior to a query being made.

Classification with Instance-based Learning

The concept behind instance-based learning is that we will use samples in a training data set in order to make inference on a query.

The **Nearest Neighbour** classifier is a specific example, where it classifies a test instance to the label of the nearest training instance, where nearest is subject to some distance metric. This is a **non-parametric model**, which means it naturally emerges from the training set. Note in the example below, an issue with this is that it can be sensitive to noise, as it would classify the blue point to be red, as it is the closest instance in the training set, even though it's more likely to be black - it is very sensitive to noise, and can **overfit** to the training data.

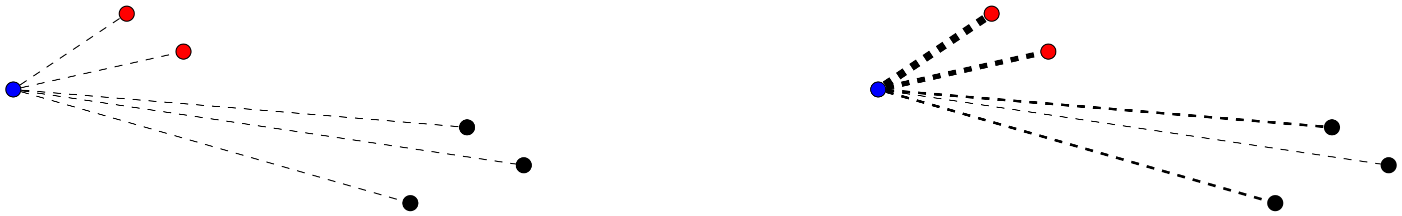
On the other hand, if we consider the **k Nearest Neighbours**, highlighted by the lines in violet, we get the class to be black, as we have 4 against 1. Usually, we need k to be odd, to ensure a winner for the decision task.



Increasing k will give the classifier have a smoother decision boundary (higher bias), and less sensitive to training data (lower variance). Choosing k is dependant on the dataset, normally with a validation dataset.

The distance metric can be defined in many different ways, including the ℓ_1, ℓ_2 and ℓ_∞-norms as seen in **CO233**. Other metrics exist such as the **Mahalanobis distance** for non-isotropic spaces, typically used for Gaussian distributions, or the **Hamming distance** for binary strings.

Another variation is the **Distance Weighted k-NN**. For example, we may not want to trust neighbours which are further away, such as in the example below.



The idea is that we add weights to each neighbour (depending on distance), typically a higher weight for closer neighbours. We then assign the class based on which class has the largest sum. This metric, $w^{(i)}$, is any measure favouring the votes of nearby neighbours, such as;

- inverse of distance

$$w^{(i)} = \frac{1}{d(x^{(i)}, x^{(q)})}$$

- Gaussian distribution

$$w^{(i)} = \frac{1}{\sqrt{2\pi}} e^{-\frac{d\left(x^{(i)}, x^{(q)}\right)^2}{2}}$$

The value of k is less important in the weighted case, as distant examples won't greatly affect classification. If $k = N$, where N is the size of the training set, it is a global method, otherwise it is a local method (only considering the samples close by). This method is also more robust to noisy training data, however it can be slow for large datasets.

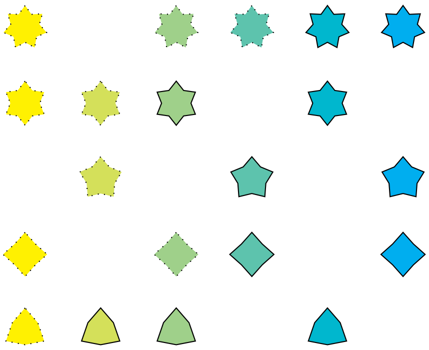
As this method relies on distance metrics, it may not work well if using all features in high dimensional spaces. If these features are irrelevant, instances in the same class may be far from each other. One solution to this is to weight features differently.

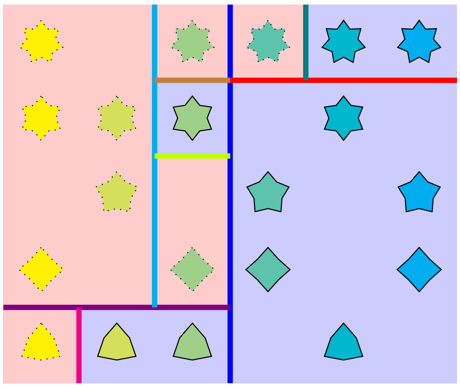
k-NN can also be used for regression, either by computing the mean value across k nearest neighbours (which leads to a very rough curve), or by using locally weighted regression, which computes the weighted mean value across k nearest neighbours, leading to a smoother curve.

Classification with Decision Trees

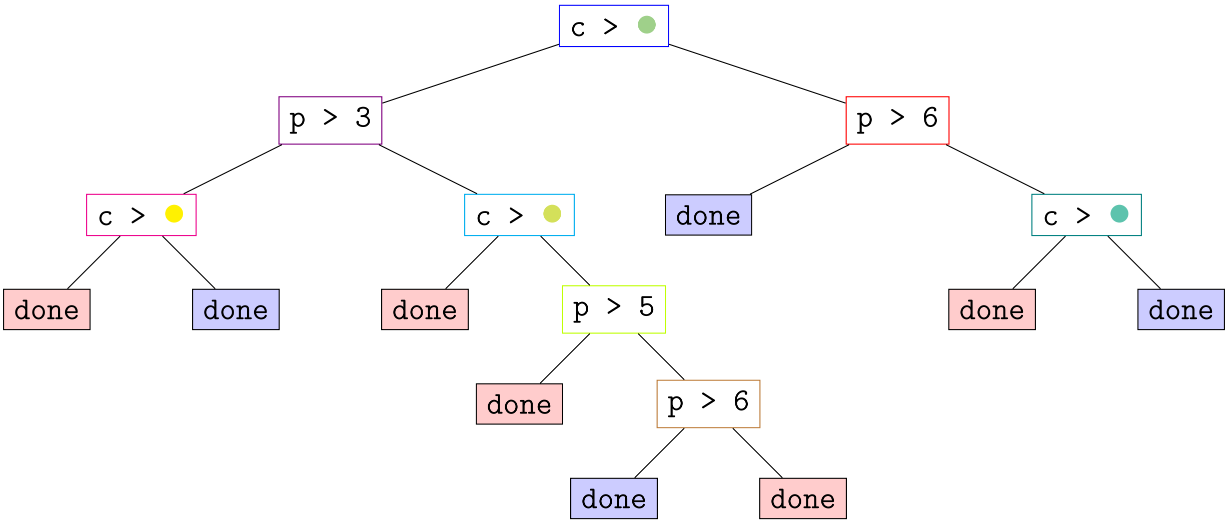
Decisions trees are the principal of focusing on a subset or single feature of each sample and then make a decision whether it's true or false (for each feature), and repeat this process to finer decisions until we manage to classify the sample that we want to check.

In decision trees, we learn a succession of linear decision boundaries that we can use to eventually correctly classify samples.





In the example above, we repeatedly choose divisions that result in the fewest number of errors, until we are able to classify everything. This results in the following decision tree, when we are using the attributes of colour and number of points. For brevity, the left branch is the `false` branch, `p` means points, and `c` means colour.



Decision trees are a method of approximating discrete classifciation functions, by representing them as a tree (a set of if-then rules). The general algorithm (ID3) for constructing a decision tree is as follows;

1. search for the optimal splitting rule on training data
2. split data according to rule
3. repeat 1 and 2 on each subset until each subset is pure (only containing a single class)

How to select the 'optimal' split rule

Worked example for construction decision tree

Summary and other considerations with decision tree

Week 4 (Machine Learning Evaluation)

Week 5 (Artificial Neural Networks I)

Week 6 (Artificial Neural Networks II)

Week 7 (Unsupervised Learning)

Week 8 (Genetic Algorithms)