Find information about Machine learning, training algorithms and learning types.

Building machines that do not need to be programed explicitly is often called cognitive computing, which refers to the sentence from Arthur Samuel. A better definition comes from another machine learning researcher Tom Mitchell where he states

Well posed Learning Problem: A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E.

Let's consider a computer game. If the game has a character that must travel and collect points, the travel and collect are two tasks that it must complete. For every move he makes, that move is judged based on some kind of a value the move gave him which can be named as the performance of the move. These performances are then stored in memory, or an experience, for the character so when he encounters a familiar situation he can look into memory to take the best move based on the best performance for the task from his experience.

There are many different methods and strategies used for variety of problems. The most common algorithms are split into two different strategies, *supervised* and *unsupervised*. In the supervised category, the algorithm is fed with a training data with values that contains a correct expect answer of that value. If a supervised algorithm would be used on the game described earlier, it would be fed by a dataset of moves which could hold answers for expected values of that move. By using unsupervised, the game would simply look for patterns and best values in many runs to determine what a good value is and what isn't. This is of course a really narrow example, but the basics are that the supervised is being told what to look for (hence, it is being supervised) but the unsupervised finds them itself.

Some classic algorithms related to supervised are both *regression* and *classification*. In regression problems we find the approximate answer by learning it from a continuous value. Classification is what an object with a set of features can be classified by the values of those features. If you would like to classify a picture of a dog and a cat, it could have features like height, weight, color, fur and more. That classification of the picture could then be determined from those values.

Unsupervised algorithms are for example *segmentation* and *random forest*. Segmentation is when a structure is to be learned from a clusters of similar behavior. Often used in market analysis to group customers into groups based in similar buying behavior. Random forests are decision trees based on values and expected values found to either clusters of attributes or by more important values that the algorithm can define by itself.

The dataset set used in these algorithms comes as a set of examples that is based on some features and responding answer to it. The main objective of the algorithms is then to find the correct function for that set of data so that the expected guess of a value of unforeseen data is at minimum.

Question 1

What is the difference between supervised and unsupervised learning? Explain with a sample data or give an example of results in both cases.

Supervised learning is guided training. We can let the computer know which labels we are looking for so when looking for the numbers 1, 2… 9, 0 in the MINST dataset the computer knows that the value is 1 and can classify it as 1. It will give us a correct answer based on the input given, such as determining if the picture is a cat or a dog.

In unsupervised learning, the computer does not know what data it is seeing and what result it should give back, thus the methods used are different. Instead of returning precise a label or classification, it would rather give information about how correlated columns are, what attributes affect other attributes or grouping structures into clusters.

Question 2

Why do we often split our dataset into training data, validation data and then cross validate our samples. Explain these 3 terms, why they are used and how they come in handy.

By splitting up our dataset we can train the code on a specific data it has seen before, hence the training set. The other part of the validation set is data that the algorithm hasn't seen before to make sure that we can take appropriate decision when faced with a new data that might be generated tomorrow or the day after (the split is usually about 70/30). Cross validation is then the test when we compare the success rate of analyzing new data based on the trained data.

Question 3

When we want to detect important features in data, one of the machine learning algorithms currently used is PCA. Why is this method so popular and how does it work?

PCA is a dimension redundancy. It uses linear algebra and eigenvectors to find which attributes of a dataset span the biggest space of the correlative data. So instead of learning from a dataset or finding clusters from 20 attributes, we can use the top 3 or 4 features instead (usually attributes that cover 80% of data) to find both better and more precise results. Using too much data that doesn't tell us much will more likely skew the results rather than improve it. It will also slow down performance.

Question 4

What is meant by over fitting and under fitting data in regressions or gradient descent?

When trying to approximate the desired value we want to reduce the amount of error generated in each calculation by a set of features. In every calculation we have an alpha value (usually called the learning rate) that that is multiplied with the value and the error, which brings us closer to the correct value. When we over fit the data, the learning rate is way too high so we drift further from the correct outcome and increase the error in every step. The same goes for under fitting, in that case the alpha value is too low.

Question 5

Discuss why functional programing is more popular in data science and machine learning rather than object oriented programing. Name a few of them.

While object oriented programming is more focused on dealing with complexity of software and user interactions, functional programming rather focuses on data pipelines for input and output without the need of user interaction. It often results in fewer lines of code and produces less memory stamp since variables are immutable. C# and JAVA are OOP languages, not popular for data science. Matlab, F# are pure functional programming languages better for machine learning. Python is a hybrid.