TITLE

Todd Davies

September 30, 2014

Introduction

Machine learning is concerned with creating mathematical "data structures" that allow a computer to exhibit behaviour that would normally require a human. Typical applications might be spam filtering, speech recognition, medical diagnosis, or weather prediction. The data structures we use (known as "models") come in various forms, e.g. trees, graphs, algebraic equations, probability distributions. The emphasis is on constructing these models automatically from data—for example making a weather predictor from a datafile of historical weather patterns. This course will introduce you to the concepts behind various Machine Learning techniques, including how they work, and use existing software packages to illustrate how they are used on data. The course has a fairly mathematical content although it is intended to be self-contained.

Aims

To introduce methods for extracting rules or learning from data, and provide the necessary mathematical background to enable students to understand how the methods work and how to get the best performance from them. This course covers basics of both supervised and unsupervised learning paradigms and is pitched towards any student with a mathematical or scientific background who is interested in adaptive techniques for learning from data as well as data analysis and modelling.

Additional reading

Introduction to machine learning (2nd edition)	Alpaydin, Ethem	2010

Contents

1 Machine Learning

Machine Learning is the creation of self-configuring data structures that allow a computer to do things that would be classed as 'intelligent' if a human did it.

Machine learning has been around in it's infancy since the 40's, where reasoning and logic were first studied by Claude Shannon and Kurt Godel. Steady progress was made with lots of funding through until the 70's, where people then realised AI was very hard, causing funding to dry up, and the term 'AI Winter' to be coined.

In the 80's people started to look at biologically inspired algorithms such as neural networks and genetic algorithms, and there is more investment. This leads to the field of AI diverging into many other fields such as Computer Vision, NLP, Machine Learning etc. In the 00's, ML begins to overlap with statistics, and the first *useful* applications emerge.

2 Nearest Neighbour Classifier

The Nearest Neighbour (NN) classifier is a simple implementation of a machine learning algorithm. We can give it a set of training data with each point labelled by a class, and then give it another datapoint, and that datapoint will be classified according to the data.

The premise is that you plot all the points of the training data on a graph, and when you want to classify another

datapoint, you simply plot it on the existing graph, and classify it by the classes of it's nearest neighbour(s).

One nice aspect of the NN classifier is that you can use it with as many features as you like with little extra effort. A standard implementation of the NN classifier might plot graphs in two dimensions, and thus will only be able to classify data with two features. However, if you want to include more (or less) features, you need to adjust the number of dimensions on your graph to match the number of features you're including. Then when you find the nearest neighbours on the graph, you find them in n-dimensional space rather than 2-dimensional space, where n is the number of dimensions on the graph.

2.1 Finding the distance between two n-dimensional points

In order to calculate which points in the training dataset are the nearest neighbours to the new data point, we can calculate the *Euclidean Distance* between the two points. In n dimensions, this how:

$$d(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_{n-1} - q_{n-1})^2 + (p_n -$$

This is actually a very simple computation. If you're given two arrays (containing the coordinates of your points) of equal lengths, then you can do something similar to Listing 1.

```
val c1: List[Int] = List(4,6,2,..)
val c2: List[Int] = List(8,4,9,..)
val delta: Set[Double] = for {(a,b) <- c1 zip c2} yield math
    .pow(a-b, 2)
val distance: Double = math.sqrt(delta.sum)
```

Listing 1: Scala Nearest Neighbour