[1]

In this lecture we will consider a machine learning strategy based on similarity measurements between features in our dataset. As before, we are dealing with a supervised learning scenario where we start with a labelled dataset having many instances of one or more descriptive features and a categorical or continuous target label. Our goal is to train a model with labelled instances which will allow us to make predictions of instances yet unseen.

[2]

The big idea in similarity-based modelling is that features which are close to each other are strong predictors of a categorical class or continuous variable. Let’s now consider how we might implement a similarity-based learning approach

[3]

Consider an instance with N features and an associated target label. We can treat the value of each feature as coordinate on one axis of an N-dimensional space. To illustrate this idea, consider a two-dimensional feature space having just two features per instance. Graphically, each instance can be represented as a single plot on a 2D scatter plot as shown. The first feature value represents the x-coordinate distance and the second feature represents the y-coordinate distance from the origin. The axes are scaled according to the ranges of the respective feature values. Although we can’t graph dimensions higher than three, conceptually, an instance of N features can be represented as coordinates in an N-dimensional feature space.

[4]

So why is it interesting to think of feature values in this way? Well, the idea is that we can define our similarity measure between feature instances as a distance metric. As distance metric is some measure of the spatial distance between two points in space, in our case an N-dimensional feature space. We can use different distance metrics for model training depending on the problem domain in question. That choice will be a trade-off between accuracy and computational complexity based on trial-and-error experimentation. When starting out with similarity-based modelling, it is common to choose a simple distance metric such as Euclidean distance or Manhattan distance, both of which are relatively easy to compute. Of these, Euclidean distance is the most popular and is based on Pythagorean ratios.

[5]

The Nearest Neighbor algorithm, which is the simplest algorithm we will study, is a machine learning model for predicting a categorical class or continuous class. Let’s start by considering it as a classifier. Unlike other algorithms that we study in this course, the Nearest Neighbor algorithm has no formal training step. The model state is already implied by and captured by the N-dimensional feature space represented by the training dataset instances. We can make a prediction using Nearest Neighbor by considering an unlabelled feature we have not previously seen. To do this we calculate a distance metric between our unlabelled feature values and each of the features in our training data set.

[6]

The class of the training set feature which is closest in distance to our unlabelled feature will be our class prediction by this algorithm. The assumption is that the distance metric is the discriminator of similarity and therefore the predictor of the categorical class. It should be clear from this algorithm outline why we cannot now test our model accuracy with any data that we already used in the training set as that would, by definition, include unfair bias in the test results. As will all model testing and validation, we must use non-training data to test the performance of the model. Using pre-labelled data as our test data allows us to measure the proportion of the test result that our model successfully predicts.

[7]

From a classification perspective, the Nearest Neighbor algorithm effectively divides the feature space into a set of distinct regions. This is one way to understand the operation of the algorithm at a conceptual level. At one level, this division is by instance having as many regions (or tessellations) as there are instances in the training set. At another level you can think of the regions as dividing the feature space into a number close to the cardinality of the target feature. This is because similar features (having the same target label) cluster together in the feature space forming decision boundaries for making predictions of unlabelled features.

[8]

As you might suspect, because the nearest neighbour algorithm is so simple and has no training step which transforms the input data in any way, it is particularly sensitive to the training set used to determine the model decision boundaries. Outliers and invalid feature values are a particular problem if retained in the training. It is recommended that best practice in data exploration and data preparation is performed before using a dataset with a Nearest Neighbor algorithm.

[9]

There is a simple extension to the Nearest Neighbour algorithm which can both improve its accuracy and improve its resilience against data imbalance and noise. This is known as K Nearest Neighbors in which the algorithm considers not just one, but k neighbors as the discriminator for class and value predictions. When making a prediction for an unlabelled feature, we find the k nearest labelled features using the similarity metric and use their labels to determine the predicted class using a majority vote. For this reason, it is typical to choose k to be an odd number such as three, five or seven neighbors. In extreme cases where each k prediction is the same, something which should not happen very often, then we can just choose the most frequently occurring class of the target feature.

[10]

The biggest problem with the Nearest Neighbor algorithm variants is the runtime performance. To make a prediction for an unlabelled feature, we have to compute a distance metric to all other features in the dataset. Several factors impact this performance – the number of training features, the number of features per instance and the complexity of the metric. A large dataset can be slow to process so this algorithm, as described, is not usually suitable when a fast prediction, say in real time, is required. One approach to improving the performance of Nearest Neighbors is to reduce the number of features which need to be considered in the prediction. A K-dimensional tree (called a K-D Tree) can be used to used to improve the performance from linear in N to logarithmic in N where N is the length of the dataset. The idea is that each sub-tree root node partitions the dimensions of the feature space in turn (say using a median value of the axis to split the sets into two parts). The search space for predictions is dramatically smaller leading to faster runtime performance. However, the trade-off is that some classifications will be missed, particularly at the decision boundaries of the K-D Tree splits as these are somewhat arbitrary with respect to the feature space distribution

[11]

The Nearest Neighbor algorithm as described can be easily adapted to deal to continuous value predictions. Instead of using the k nearest neighbors classes to determine a categorical variable, when can take an average value of the k nearest continuous target feature values. This represents a point in the feature space equidistant from the nearest neighbors representing an average value between them.

[12]

It is recommended to normalise the continuous feature data in advance of performing predictions to smooth our any differences between the value ranges on each axis. This is because distance metrics can be sensitive to very high or very low values on one axis with respect to another access. Normalisation eliminates these magnitude differences by forcing all of the feature ranges into a common, constrained range. This will improve the overall performance of the predictions on average. Categorical feature values should be contiguous in range, this is there should be no gaps in the class values

[13]

In summary, we have looked at a simple predictive model using similarity-based learning. By mapping feature values into an N-dimensional space, the K nearest neighbors algorithm uses a distance metric to determine how similar features are to each other. A previously unlabelled feature can be predicted by measuring its distance from each of the other features in the training space and selecting the nearest feature’s target value to make the new prediction. Although simple, the nearest neighbour variants are sensitive to training data inputs and do not perform well over large datasets unless we can reduce the search space to be considered.