file/after/tracklang.stypackage/after/bidipackage/after/luabidi

Classification

Machine learning just refers to a program P that learns with some experience E according to some way of measuring its performance P.

Supervised learning has labels along with the training data. This means that durign training we can compare our guess with the correct answer.

Unsupervised data's correct answer is unknown, and here we want to discover the undrelying relations or clusters within the data.

Definition

Given a set of attributes (among which there is a class label), our model is a function of the attributes and outputs a class label. Before we start processing the data, some data cleaning in order to normalize the values or fill in missing values might be required.

There are several ways of evaluating the performance of our model: - predictive accuracy - speed(inference, etc…) - robustness (how does it handle missing values?) - scalability - interpretability (how easy is it to understand conceptually what it's doing?) - goodness of rules (size, compactness of the representation)

some techniques include: - Decision Trees - Neural networks - K-nearest neighbors - Support Vector Machines - Random forest

Decision Trees

DTs are a hierarchical way of assigning class labels based on checking an attribute at a certain value.

The general way we build the decision trees is by having D_T at some node. If the items in the dataset are all one class, we assign the data's label to the node. If there is data belonging to more than one set, we use some metric to split the dataset at the node. Some trees could be split into two features, or we could choose to have a multi-split approach.

If the values in a feature are continuous, how do we get a value to split on? We could choose the average of two consecutive sorted values in the feature column. We then have multiple different splits of data, so which is the best one? Based on the proposed value, we need some metric of checking the **purity** of the resulting datasets. There are a couple of metrics:

The **gini index** measures the impurity of a node

$$\text{GINI}(t) = 1 - \sum_j [p(j|t)]^2 \tag{1}$$

where $p(j|t)$ is the relative measure of items in class j in set t, i.e. $|c_{j\in t}|/|t|$. When all the items belong to one class, the gini index is 0, so we are choosing the dataset that is the least “mixed”. Thus at every node, we have to calculate the gini index of both the resulting children nodes and subtract it from the impurity of the parent.

Another measure we could use is the **entropy**, which measures the homogeneity of a node.

$$\text{Entropy}(t) = -\sum_j p(j|t)\log p(j|t) \tag{2}$$

Yet another measure is the **information gain**, which measures the *reduction in entropy* of the resulting split.

$$\text{GAIN}_{split} = \text{Entropy}(p) - (\sum_{i=1}^{k} \frac{n_i}{n}\text{Entropy}(i)) \tag{3}$$

However, this apporach tends to prefer smaller and more numerous yet purer partitions.

The **error** is just the probability that the label is different from our prediction, $E(t) = 1 - (i|t)$.

Decision trees are a popular method for classification because they are easy to understand, deal well with symbolic feature, relatively quick to train, and deal well with data that is not numeric.

Classification Issues

When our model is too simple or too complicated, the accuracy in our predictions suffers. Why? If our model is too simple, whe nwe take a look at the training data, our error will still be relatively large when we finish training. This means that our model cannot entirely capture all the information in the training data. The error in the training data will be large, and the testing data’s error will be even larger.

When our model is too complicated, it might focus too much on generating useful parameters for the testing set, but not on the testing set. It might be getting derailed by noise in the testing data. Its accuracy on the training data might be very low, but it might be very high on the testing data due to it not being general enough to handle random noise. It might split the space into too small areas.