Mining Time Varying Data

Harrison McKenzie Chapter

Introduction

Motivation In the real world the concepts which govern sets of collected data are often not stable for long periods of time. Unfortunately, most supervised learning techniques base their classifiers’ development on a static training set, decaying their effectiveness as time progresses.

Concept Drift The process of an underlying pattern, or concept, in a set of data changing is called concept drift. It means that the “functions” which constructed the resulting data have slowly changed over time, and no longer generate the data they did at earlier points in time. To combat concept drift, the classification strategy must be refined to adapt to moving concepts in the data.

The Problem For a set of training data $T$, composed of many subsets $\{C_1 \ldots C_n\}$ varying in time and generated by different underlying concepts, develop a classifier which minimizes incorrect classifications.

Input Data To solve the problem, it is expected that the solution can be trained with already classified sample data (supervised learning), with the classified samples continuing for the whole period of use. Benefit is still derived by allowing the extrapolation to much larger sets of unclassified data which are present at the latest time period, and by promoting the understanding of historic trend variance.

Alternate Strategies

Naive Solution A simple way to handle many sets of data is perform windowing on the samples based on time, and run standard classification methods on each subset, and use the resulting classifier on each subset of data independently. This approach has two chief problems:

- Robustness For subsets which do not vary widely, large amounts of sample data is discounted, increasing the probability of over-fitting. On the other hand, when subsets do vary widely, a classifier built on a monolithic training set is likely to perform poorly.

- Efficiency The transient sample window means that the classifier calculated has a very short window of utility before it must be recalculated.

The Ensemble Strategy

Overview

1. Acquire the next temporal chunk from the data stream
2. Train a classifier $C'$ from the chunk
3. Compute the error rate of $C'$
4. Derive the weight $w'$ for $C'$
5. For all the other classifiers, compute weight $w_i$ for the current data set
6. Determine top classifiers
Algorithm Breakdown

Chunking To break the data set up, a set of the next samples is taken. The size of this chunk is some relatively small value relative to the size of the data set. This minimizes the time required to train classifiers for it, as well as ensuring that it will fit easily into system memory.

Classification For the current chunk, a classifier is trained using only data from that chunk. The data size is relatively small, which allows it to capture fast moving changes in the data.

Computation of error rates For the current classifier, as well as all previous classifiers, the mean squared error that classifier would have on the current data set is computed. That is:

\[ MSE_i = \frac{1}{|S_n|} \sum_{(x,c) \in S_n} (1 - f_i^c(x))^2 \]  

(1)

Where \( S \) is the current Sample (the chunk), composed of classified points \((x, c)\).

Computation of classifier weights To maximize the effect of good classifiers and minimize or eliminate the effect of poor classifiers, each classifier is weighted according to the error which it generates on the current chunk. The weight used is computed as the difference of the MSE for a random classifier, and the MSE computed in the previous step for each classifier.

Determine top classifiers While the weighting of the classifiers reduces the influence of old, misfitting classifiers on the current data, to increase efficiency, such classifiers can be removed before the classifiers are applied to real data to reduce the effort required to calculate the final result for each point under examination.

Classification of data To calculate the class of a data point using the ensemble classifier, the weighted probability of that point being in class \( c \) is summed for all classifiers. That sum is then normalized by dividing it by the sum of all the weights.

\[ f^E_c(x) = \frac{\sum_{i=0}^{n} w_i f_i^c(x)}{\sum_{i=0}^{n} w_i} \]  

(2)

References

