Concept Drift Adaptation

Tarushii Goel

April 2019

1 Introduction

Many models using machine learning are trained with the assumption that the underlying concept the model is predicting is static. In the real world, this is hardly ever the case. Take the example of fraud email detection. A model may be very accurately trained to detect fraudulent emails that are being sent right now, but in the next few years, the common language, subject, etc. of phishing attacks would have changed, making your model useless. Concept drift is the variation in the underlying function the model is predicting. Models can be trained to detect and update themselves utilizing the data that is sent through them, essentially learning the concept drift.

2 Principles Causing Change

The best way to handle concept drift in a model is by learning from the data stream. At time step t the learning algorithm is presented with a set of labeled instances $[X_0, ..., X_t]$. The algorithm then makes a prediction on an unlabeled instance X_{t+1}. After the prediction is made, a true label for instance would be available and the time step increments. Concept drift is when f_t, the function generating an instance at time step t, changes. We can use Bayes' theorem to find the probability of X_{t+1} being an instance of class c_i.

$$p(c_i|X_{t+1}) = \frac{p(c_i)p(X_{t+1}|c_i)}{p(X_{t+1})}$$

Given this, there are three ways drift occurs:

1. $p(c_i)$ may change (i.e., class priors).
2. $p(X_{t+1}|c)$ may change (i.e., the distributions of the classes).
3. $p(c|X_{t+1})$ may change (i.e., the posterior distributions of class membership).

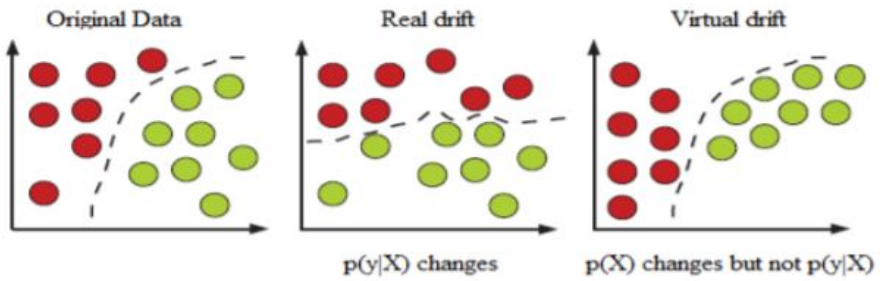

Figure 1: Real drift vs. Virtual drift

3 Speed of Drift

As shown in figure 2 there are four different types of concept drift models.

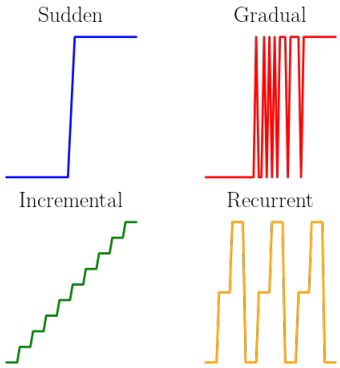

Figure 2: The 4 types of drift

3.1 Sudden Drift

If the concept drift can happen suddenly at specific moments in time t where suddenly instead of function f function g becomes applicable, its defined as sudden drift. Sudden drift is often the easiest to detect.

3.2 Gradual Drift

When there is a relatively smooth transition of sampling from f to sampling from g over a longer period of time, its called a gradual drift. The existence of a very gradual shift may often go unnoticed by the classifier, making this drift harder to detect.

3.3 Incremental Drift

Another form of drift could result in small changes from initial function f over a period of time resulting in final change to g. It poses similar difficulties as gradual drift.

3.4 Recurrent Drift

If the same drift could repeat and come back to same functions from time to time, its called a recurrent drift. Recurrent drift does not need to be periodic. However the learning algorithm could still take the advantage of the prior data and archive previously learned model and use them appropriately to quickly respond to recurrent drift.

3.5 Adversarial Drift

Adversarial drift occurs when the source of the data is intelligent, such as an adversary trying to get their SPAM through the SPAM filter, and beat the model. That's a subject of active research.

4 Addressing the concept drift

To address the concept drift the algorithms need to address two sub-problems. First is the identify the existence of the drift, also known as change detection or anomaly detection. Once the drift has been identified, the next task is to find out the best way to address the drift and define the new model to predict the data. These approaches fall into three broad categories.

- adaptive base learners
- learners which modify the training set
- ensemble techniques

4.1 Adaptive base learners

This is conceptually probably the simplest way of addressing drift. Such learners are able to dynamically adapt to new training data that contradicts a learned concept. Depending on the base learner employed, this adaptation can take on many forms, but usually relies on restricting (or expanding) the data that the classifier uses to predict new instances in some region of the feature space.

4.1.1 Decision Trees

One base learner that has been heavily studied in the context of concept drift is the decision tree; of which the most common variant is C4.5. The original extension to decision tree learning algorithm, called Very Fast Decision Tree

(VFDT) proposed by Domingos and Hulten dealt with building decision trees from streaming data. Hulten, Spencer, and Domingos adapted VFDT to create the Concept-Adapting Very Fast Decision Tree (CVFDT). In CVFDT, a sliding window of instances is retained in short term memory. After a fixed number of new instances arrive, the relevant statistics at each of the nodes are updated. If a better splitting attribute is found, the (sub)tree determines that a concept drift may have occurred and a new (sub)tree is learned. Bifet and Gavald'a proposed two new methods: the Hoeffding Window Tree (HWT), and the Hoeffding Adaptive Tree (HAT). HWT is similar to CVFDT with two major differences. First instead of waiting for a fixed number of samples, an (sub)tree modification is made as soon as there is a sufficient evidence of a drift. Second, the (sub)tree update is made as soon as there is sufficient evidence of an improvement. HWT also removes the restriction of fixed size sliding window, using an adaptive window size based on the data characteristics. Option trees retain both the new and old (sub)trees for decision making.

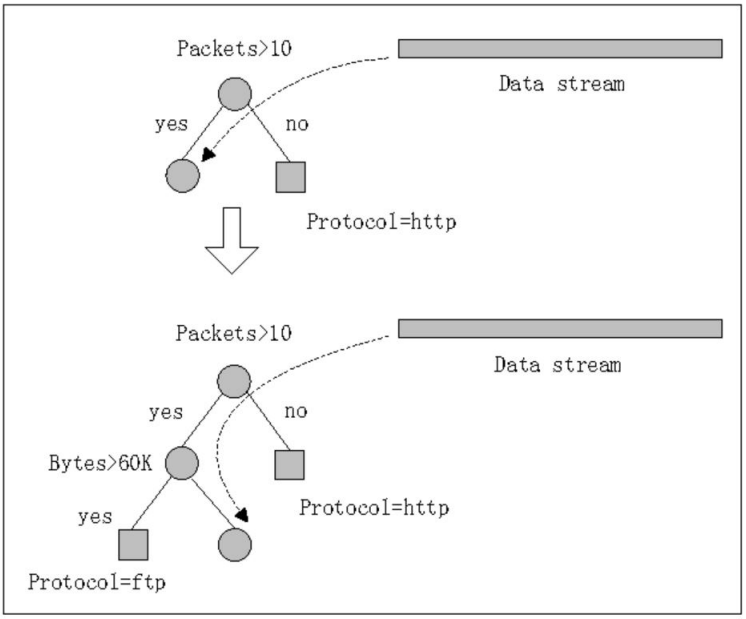

Figure 3: The adaptive Hoeffding Tree

4.1.2 k Nearest Neighbors

Alippi and Roveri demonstrate how to modify the kNN algorithm for use in the streaming case. First, they demonstrate how to appropriately choose k in a data stream which does not exhibit concept drift. Then they describe how to update the kNN classifier when no concept drift is detected (add new instances to the

knowledge base), and when concept drift is detected (remove obsolete examples from the knowledge base).

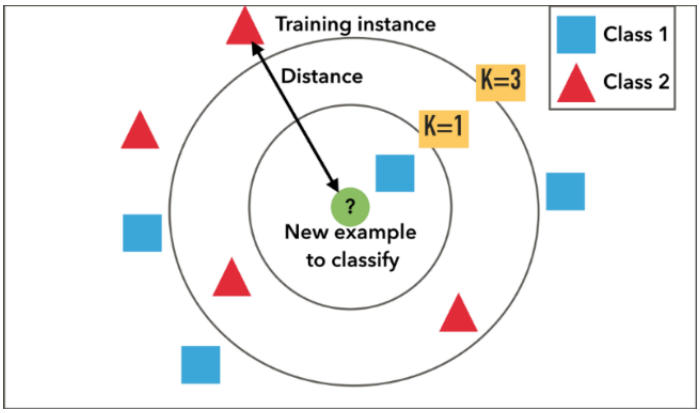

Figure 4: Illustration of k-selection with different levels of k

4.2 Modifying the training set

Another popular approach of addressing concept drift is by modifying the training set seen by the classification algorithm. The most common approaches employed are windowing or instance selection (i.e., where only a subset of previously seen instances are used), and instance weighting. One of the strengths of the modification approaches over the adaptive base learners approach is that the modification strategies are often classifier agnostic. Indeed, much of the research into modifying the training set deals not with building an entire classification algorithm, but merely with how to select or weight instances which are used to build a classifier.

4.2.1 Windowing

One standard example of windowing is the *block-based* technique. Data is divided into chunks and is processed one chunk at a time. Usually, each block consists of same number of training samples and learners are only trained on a block once it is complete. However, some algorithms change the size of the blocks based on how strong the drift is at that moment.

The *sliding window* technique is even more prevalent. Instead of processing data as a chunks, data is put in a window, which is one, changing data block of a fixed size. Over time, older training data is pushed out of the window (or forgotten) and the new instances enter. The window only ever has the most recent training examples. **AdWin** is a popular algorithm that uses adaptive windows to detect drift. Adaptive window techniques use statistical tests to determinf if the data in the window is homogeneous. That is, to ensure that

concept drift is not happening within the window. Other criterion involve looking at the coverage and predictive value of the models. Another technique uses two windows one at least twice the size of the other, and once the concept drift is detected using these two windows, the window size is increased to capture the drift.

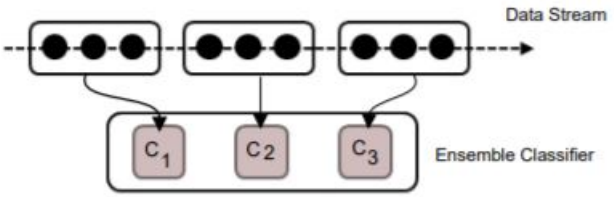

Figure 5: The block-based technique

4.2.2 Instance Weighting

Instance weighting builds off of instance selection, by assigning weights to each data instance. Examples with higher weights have more influence over the model. The highest weight is typically given to the most recent examples, since it is most likely more related to the current concept, and the lowest weight to the oldest piece of data. Weighting has more precise control over how data is used in the model than blocking and windowing techniques, since instance selection only considers two weights, "present" or "not present." One formula used to assign weights is listed below.

$$w_\lambda(x) = exp(-\lambda t_x)$$

4.3 Ensemble Learning

Ensemble learning constitutes of combining multiple (slightly different) classifiers trained potentially on different subsets of the data. A voting mechanism is used for decision-making with the ensamble classifiers. In ensemble learners, a common method is to evaluate learners on their performance, then substitute the weakest one with a new candidate learner. This way, the ensemble adapts to only keep learners most suitable to represent the current concept. A unique quality of instance selection is that ensembles can contain different learners that were trained with different blocks, which is ideal in cases of gradual drift. Ensamble techniques are particularly efficient in the case of recurrent drifts. However, in cases of abrupt drift, large ensembles would be slow to adapt.

References

[1] Learning from streaming data with concept drift and imbalance: an overview