Chapter 1

Related Works

This chapter aims to present an overview of the work done in the fields of stream mining particularly focusing on ensemble learning over streams.

1.1 Stream Mining

Compared to the classical data mining approaches stream mining is relatively a newer topic to be addressed in literature. Even though for batched approaches both classification and clustering problems have been vastly studied, their stream adaption remains a challenge due the restrictions imposed by the stream data. Possibility of temporal locality makes the classification problem harder in a streaming environment. Algorithms needs to address the evolution of underlying data stream.

Domingos and Hulten introduced a strict one-pass adaptation of decision tree [Breiman et al., 1984, Quinlan, 1993] approach in streams. Classic approaches like ID3 and C4.5 learners assumes that all training examples can be stored in the main memory altogether. This is a significant limitation to the number of examples these algorithms can handle. Similarly, disk based decision tree learners (SLIQ [Mehta et al., 1996], SPRINT [Shafer et al., 1996], etc.) become very expensive when datasets are very large and the expected trees has many levels. Domingos and Hulten proposed Very Fast Decision Trees (VFDT) [Domingos and Hulten, 2000] that uses Hoeffding bound [Hoeffding, 1963] to build an anytime decision tree for constant memory and time. The primary assumption in this approach is that, to find the best attribute for a node in a decision tree, it may be sufficient to consider only a fraction of the training set that pass through that node. Hoeffding bound provides an statistical measure to determine how much data is needed to ensure a certain degree of certainty, i.e. error margin would be bounded by a given value [Catlett, 1991].

Like most statistical and machine leaning algorithms VFDT assumes that training data is randomly drawn from a stationary distribution. This assumption is not valid for large databases and data streams. Over time underlying method or environment could change that generates data. The shift is sometimes also referred as *concept drift* in literature and can be abrupt as well as very slow. Data related to weather forecast, economic condition prediction, mis-calibrated sensors, etc. are examples of concept drifting environment. A

concept-adaptive variant of VFDT, CVFDT [Hulten et al., 2001], can handle such scenarios. CVFDT updates its decision rules, essentially the tree structure, by detecting the concept drift in the data. It maintains alternate subtrees whenever an old subtree becomes questionable, and replaces the old one with the alternative when it become more accurate. CVFDT uses a sliding window and updates sufficient statistics by increasing the count of newly arrived examples and decreasing the count of old examples in the window. Essentially CVFDT achieves same accuracy that would be achieved if VFDT would have been run again with the new data. CVFDT does this in $O(1)$ with additional space requirement as compared to the VFDT's $O(w)$ where w is the window size.

Another decision tree based approach based on so-called Peano Count Tree (P-tree) has been developed by Ding et al. for spatial data streams [Ding et al., 2002]. The Peano Count Tree is a spatial data structure that facilitates a lossless compressed representation of spatial data. This structure is used for fast calculation of information gain for branching in decision trees.

Aggarwal et al. employs a slightly different idea in handling time-evolving data in their on demand classification approach of data streams [Aggarwal et al., 2004]. They used a modified micro-clusters concept introduced in [Aggarwal et al., 2003]. Micro-clusters are created from the training data stream only. Each micro-cluster corresponds to a set of points from the training data belonging to the same class. To maintain statistics over different time horizons and avoid storage of unnecessary data points a geometric time frame is used. In the classification task, the k nearest neighbor based approach is taken, where micro-clusters are treated as node weighted by their instance counts.

In [Ganti et al., 2002] two algorithms named GEMM and FOCUS have been introduced for streams under block evolution. GEMM is used for model maintenance and FOCUS is for change detection between two data stream models. These algorithms has been tested using decision trees and frequent item set models. FOCUS uses bootstrapping methods to compute the distribution of deviation values when data characteristics remain the same. This distribution is then used to check whether the observed deviation value indicates a significant deviation. In another approach to handle concept drift, Last [Last, 2002] proposed an online classification system OLIN that would dynamically adjust the size of the training window and the number of new examples between model re-constructions to the current rate of concept drift. OLIN uses constant resources to produce models, and achieves nearly the same accuracy as the ones that would be produced by periodically re-constructing the model from all accumulated instances.

Later, Aggarwal proposed an concept drift technique based on velocity density estimation [Aggarwal, 2003]. Velocity density estimation is a technique to understand, visualize, and determine trends in the evolving data. The work presented a scheme to use velocity density estimation to create temporal velocity profiles and spatial velocity profiles at periodic instants in time. These profiles are then used to predict dissolution, coagulation, and shift in data. Proposed method could detect changes in trends in a single scan with linear order of number of data points. Additionally a batch processing techniques to identify combinations of dimensions which results the greatest amount of global evolution are

also introduced. In [Kifer et al., 2004] authors tried to formally define and quantify the change so that existing algorithms can precisely specify when and how the underlying distribution has changed. They employed a two fixed-length window model, where a current one is updated every time a new example arrives and a reference one is only updated when a change has been detected. To compare the distributions of the windows $L1$ distance has been used [confirm!read again]. Another method to compare two distribution has been presented in [Dasu et al., 2004] where authors used Kullback-Leibler (KL) distance to compare two distributions. KL distance is known to be related to the optimal error in determining the similarity of two distributions. In this non-parametric method no assumptions on the underlying distributions is required.

1.2 Ensemble Learning

Traditional machine learning algorithms generally feature a single model or classifier such as Naïve Bayes or Multilayer Perceptron (MLP). The free parameters of these learners (e.g. weights of feed-forward neural network) are set by realizing the complete training set. These classifier provides a measurement of the generalization performance i.e. how well the classifier generalizes the training set. However, given a finite set of training example, it is rather reasonable to assume that the data might contain several different generalization. For example, a different setting of neural network classifier (weights, node layers, node counts, etc.) changes the final network to some extent. For stream environment, this assumption becomes primitive. Thus, choosing a single classifier is not always optimal. Using the best classifier among several classifiers where each are trained with same training set would be an alternative, however, information is still being lost by discarding sub-optimal options. A better alternative would be to build a classifier ensemble. Ensemble classifiers combine the prediction of multiple base level model built on traditional algorithm. A simple process for combining prediction could be to choose the decision based on majority voting [Parhami, 1996]. As demonstrated in several works [Breiman, 1993, Schapire, 1990, Wolpert, 1992] ensemble methods (e.g. ensembles of neural networks) [Hansen and Salamo, 1990, Tumer and Ghosh, 1999] yield better performance.

Without proper selection and control over the training process of the base learners, ensemble classifiers could result in poorer performance. Simply choosing a base classifier and training it for several settings would surely produce highly correlated classifiers which would have adverse effect on the ensemble process. One solution of this issue is to train each classifier with its own training set generated by sampling the original one. However, with random sampling each classifier would receive a reduced number of training patterns, resulting a reduction in the accuracy of the individual base classifier. This reduction in the base classifier accuracy is generally not recovered by the gain of combining the classifier unless measures are taken to make the base classifiers diverse. Classifiers with complementary information would give the lowest correlation [Breiman, 1993, Tumer and Ghosh, 1999]. Many methods have been proposed to promote diversity among the base classifier: bagging [Breiman, 1994],

boosting [Drucker et al., 1994, Freund and Schapire, 1997, Tumer and Oza, 1999], cross-validation partitioning [Krogh and Vedelsby, 1995, Tumer and Ghosh, 1999], etc. These methods mainly process the entire training set repeatedly and require at least one pass for each base model. This is not suitable for streaming scenarios. Stream adaptation of bagging and boosting methods has been introduced by Oza et al. [Oza and Russell, 2001, Oza, 2001].

1.2.1 Ensemble Learning in Streams

Learning algorithms in data streams require maintainance of a hypothesis based on the training instances seen thus far with the need for storage and reprocessing. Facilitating this requirement, Oza and Russell developed an online version [Oza and Russell, 2001, Oza, 2001] of traditional bagging and boosting. Bagging works by randomly sampling with replacement from the training set to form a given number of intermediate training sets which are used to train same number of classifiers. During testing a majority voting scheme is employed on the decisions of all classifiers to deduce the final decision. Boosting uses an iterative procedure to adaptively change distribution of training data by focusing more precisely on misclassified instances. Initially all instances have equal weights, and at the end of a boosting round weight of each instance is updated by increasing or decreasing if the instance was classified wrongly or correctly, respectively. For online variant of these algorithms, not knowing the size of the training data poses a problem in determining the size of training sets to build the base models. In [Oza and Russell, 2001] authors address this situation by training k models with each instances where k is a suitable Poisson random variable. Later on, [Pelossof et al., 2008] proposed the Online Coordinate Boosting algorithm where the number of weight updates of [Oza and Russell, 2001] is reduced using few simple alteration.

Online bagging and boosting method do not particularly give attention to the concept drifting nature in the data. Accuracy Weighted Ensemble (AWE) [Wang et al., 2003] is one of the earliest work on concept-drifting stream data. AWE assumes that the stream is delivered in chunks of defined size. With each incoming chunk, AWE updates its k classifiers. Each classifier is associated a weight which is inversely proportional to the expected error of the respective classifier. To estimate this error, it is assumed that the distribution of test set is closest to the most recent chunks. Concept drift is adapted by effectively manipulating the number and the magnitude of the weights that are changing. An extension of AWE has been proposed in [Brzezinski and Stefanowski, 2011], namely Accuracy Updated Ensemble (AUE). AUE takes the weighting motivation from AWE, but improves the limitation of AWE. In AWE each classifier learns from the incoming chunks in a "batched" fashion. AUE employs an online scheme instead. AUE also adapts the weighting function to reduce the adverse effect in AWE of sudden drift in data. AWE weighting function is prone to suffer by rapid change in the stream and most or even all classifiers assuming they are "risky". This limitation has been addressed in AUE. Result shows that AUE performs marginally better than AWE, however, also requires slightly longer time and larger space.

As mentioned in the previous section, Hoeffding Tree (HT) e.g. VFDT [Domingos and Hulten, 2000], can be used to build classifiers for concept drifting streams. The Hoeffding Tree has the property that it adapts itself for the newer examples. The number of examples that a HT is build upon is determined by two numbers: (i) the be size of the tree, and (ii) the number of examples used to create a node. Thus, smaller trees adapt faster to the changes in the data, while larger trees tries to retain the rules that reflect longer time-frame, simply because they are built on more data. In other words tree bounded by size n would be reset twice as often as tree bounded by size 2n. Adaptive Size Hoeffding Tree (ASHT) [Bifet et al., 2009] uses this intuition to build an ensemble of classifiers of different sized Hoeffding trees. ASHT attempts to increase the diversity in the bagging approach. The maximum allowed size for n-th tree is twice the size of $(n-1)$-th, where the 1st tree has a size of 2. Additionally inverse of the squared error has been used as the weights for the trees. Diversity between the traditional bagging and bagging using tree are compared using kappa statistic. If two claffier agree on every example then k = 0, and if they agree on the predictions purely by chance then k = 0. To test this approach authors have used Interleaved Test-then-Train method. At the leaf level of HT, Naive Bayes predictions are used. The experiments were performed on several different genereted dataset such as SEA Concepts Generator, STAGGER Concepts Generator, Rotating Hyperplane, Random RBF Generator, etc. Performance are compared with traditional Naive Bayes, HT, and boosting methods. Evaluation concluded that bagging provides best accuracy, however, with the higher cost in terms of running time and memory. Authors made an observation that even bagging using 5 trees of different size might be sufficient for gain higher accuracy, as error level for bagging with 10 trees does not drop much but takes twice time.

Same authors also proposed an adaptive window size bagging method- ADWIN [Bifet et al., 2009]. ADWIN automatically detects and adapts to the current rate of change. To do so ADWIN adapts its window size to maximize the statistically consistently length that conforms following hypothesis "there has been no change in the average value inside the window". Window is not maintained explicitly rather using a variant of exponential histogram technique that takes $O(logw)$ memory and $O(logW)$ processing time where w is the length of the window. Experimental evaluation showed that ADWIN has better accuracy than ASHT, however, requires more time and memory.

ADWIN has later been used in leverage bagging [Bifet et al., 2010b]. Leverage bagging improves randomization by increasing resampling and using output detection codes. Resampling with replacement is done in Online Bagging using Poisson(1). Instead leverage bagging increases the weights of resampling using a larger value λ to compute the value of the Poisson distribution. The Poisson distribution is used to model the number of events occurring within a given time interval. In other improvement randomization is added at the output of the ensemble using output codes. Method works by assigning a binary string of length n to each class and building an ensemble of n binary classifiers. Each of the classifiers learns one bit for each position in the string. A new instance is classified to the class whose binary code is closest.

ADWIN has also been used in building ensemble of Restricted Hoeffding Trees [Bifet et al., 2010a]. A mechanism for setting the learning rate of perceptrons using ADWIN's change detection method is used to restrict the tree. Additionally, a mechanism for reseting the member Hoeffding trees is also been introduced when a particular member is no longer performing well. The method outperforms traditional bagging in terms of accuracy, but requires additional memory and time.

Chapter 2

Background

This chapter discusses the primitives of data and stream classification. First, a brief overview of traditional data classification methods are presented. Followed by a section on data stream classification where challenges and approaches for stream classification are introduces. Finally, overview of current state-of-the-art ensemble learning methods are discussed. These discussion are lays the foundation of the approach introduced in this thesis.

2.1 Learning Algorithms

The goal of data classification process is to predict a certain outcome based on some give data. In order to do so, data mining algorithms first processes a training set containing a set of attributes and corresponding outcome: a class or value. Algorithms develop hypotheses that best describe the relationship among the attributes and the outcome for the total instance set. Formally, learning are defined as follow: Given a data set of m instances $D \equiv \{\{\vec{x}_1, y_1\}, \{\vec{x}_2, y_2\}, \{\vec{x}_3, y_3\}, \ldots, \{\vec{x}_m, y_m\}\}$ for $i = \{1, 2, 3, \ldots, m\}$ and $\vec{x} = \{x_1, x_2, x_3, \ldots, x_n\}$, learning algorithms try to approximate $H \equiv y = f(\vec{x})$. Additional to being typical linear, polynomial, etc. functions, H can also be a set of IF-THEN rules. These rules or functions are then used to predict unseen instances where outcome class is not known. These algorithms are known as learning algorithms, and in last few decades, have widely been researched in the field of machine learning. This section briefly discusses few of the basic algorithms upon which the foundation of ensemble methods and this thesis is laid.

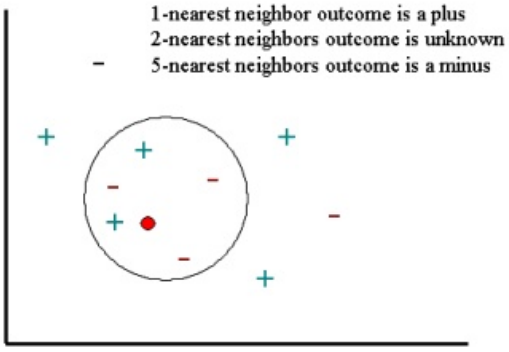


Figure 2.1: Concept of k-NN.

2.1.1 k-Nearest Neighbors

2.1.2 Naïve Bayes

2.1.3 Decision Tree

2.1.4 Neural Network

2.2 Data Stream Classification

Traditional data mining algorithms work in a memory bounded environment and requires multiple scan of the training data. In stream environment, one of the major assumptions is that new data samples are introduced in the system with such a high rate that repetitive analysis becomes infeasible. Thus, for stream classification, algorithms should be able to look into a instance only once and decide upon that. A bounded memory buffer can be used to facilitate some level of repetition. However, which instances are to remember and which are to forget would then become a decision choice. An alternate choice is to maintain sufficient statistics to have a representation of the data. The process of deletion or summarization of instances, however, means that some information are being lost over the time.

In this section, these challenges are first discussed in details. Then it presents the basis of some of the concepts arisen to handle these challenges. Finally, before moving onto the ensemble leaning, it discusses current state-of-the-art algorithms for stream mining.

2.2.1 Challenges

Challenges posed by the streaming environment can be categorized into two groups: (i) relating to runtime and memory requirements and (ii) relating to underlying concept identification. Speed of incoming data, unbounded memory requirement, single-pass learning fall into the first category. On the other hand, lack of labeled data, concept drifting, evolution and recurrence are examples of latter category.

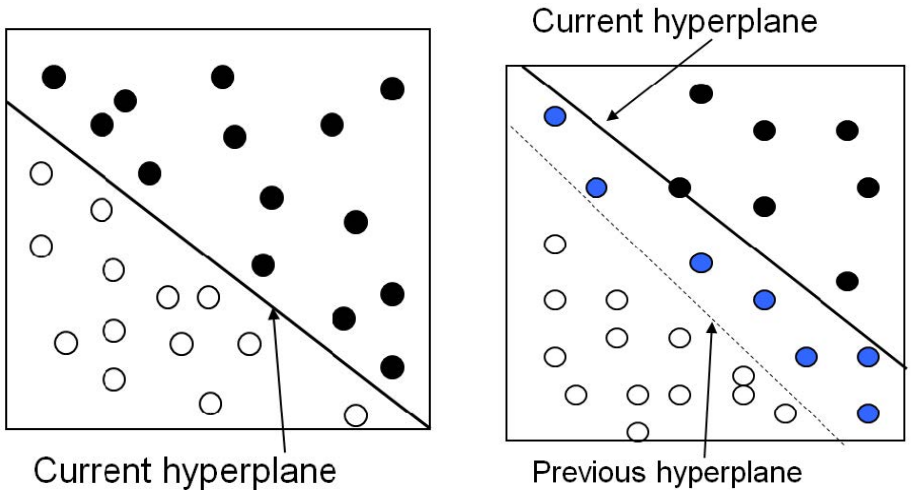
Speed of data arrival As mentioned before, it is an inherent characteristic of data streams that it arrives with a high speed. The algorithm should be able to adapt to the high speed nature of streaming information. The rate of building the classifier model should be higher than the data rate. This gives a very limited amount of available time for classification as compared to the traditional batch classification models.

Memory requirements: To apply traditional batched approaches in streaming data, an unbounded memory would be needed. This challenge has been addressed using load shedding, sampling, aggregation, etc. Rather than storing all the instances, algorithms store a subset of the data set or some statistical values or a combination of both which represents the data seen thus far. New instances can be classified only by looking into these stored information.

Single-pass learning: The premise of this requirement is two-fold. First, as mentioned above, data would not be available in the memory after a short period of time due to the volume of data. Secondly, even if the data remain available, running a batched-like approach for millions of data points would highly increase the running time of the algorithm. To attain faster processing time with limited storage, algorithm should access the data stream only once, or a small number of times. Mining models must posses the capability to learn the underlying nature of data in a single pass over the data.

Lack of labeled data: Unlike most data sets or settings of batched approaches, steam mining data sets are often poorly labeled. A large number of experimentations are done with generated data set where the data generation process can easily be controlled to have proper labeling. However, in data set collected from real world are often lack this. For example, to setup a supervised learning experimentation using a data set collected from social media, e.g. Twitter, data needs to be first categorized by human intervention. Manual labeling of such data is often costly, both in terms of resources and time. In practice, only a small fraction of data is labeled by human experts or automated scripts. A stream classification algorithm is thus required to be able to predict after observing a small number of instances, i.e. to be ready to predict anytime.

Concept drift: Concept drift is a statistical property of data streams where the target variable drifts away from the model that is trying to predict it. In other words the underlying data distribution is changing over time. As a result accuracy of the classifier model decreases over time. For example, buying pattern of the customers in a store changes over time, mostly due to the seasonality. Electrical and mechanical devices wear off over time, producing shifted result which would cause drift in the observing data. Learning models should adapt to these changes quickly and accurately. Let us consider the example in Figure 2.2. As a new chunk arrives (Figure 2.2a), a new classifier is learned. The decision boundary is denoted by the straight line. The positive examples are represented by unfilled circles while the negative examples are represented by filled circles. With time the concept of some of the examples may change. As shown in Figure 2.2b, due to concept drift

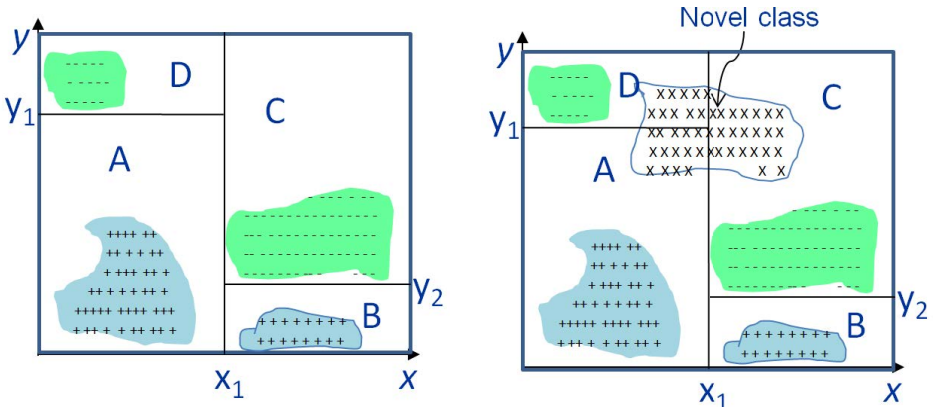


(a) (b)

Figure 2.2: Concept drift in data streams.

some negative examples may have become positive. So, the previous decision boundary has become outdated and a new model has to be learned.

One challenge posed here is to differentiate the noise in data and the actual shift of the concept. Often in streaming environment data contains the both.



(a) (b)

Figure 2.3: Concept evolution in data streams.

Concept evolution: Concept evolution is referred to the emergence of a new class or a set of classes in stream data as the time flows. Twitter stream is an ideal example where concept evolution is very easily identifiable. Twitter reacts, seemingly, very fast upon important news around the globe. Looking into the different hash tag usages in such social media currently trending topics can be identified. To present a clearer picture let's consider following example in Figure 2.3. At certain point of time four classes and their corresponding decision boundaries are shown in the Figure (2.3a). With the more incoming data a novel class emerges, and for that decision boundaries need updating. Emergence of new class can affect any number of decision boundary/ rule, from one to all.

Concept evolution is also prone to noise. Furthermore, clear distinction between drift and evolution might not always be possible, partially due the lack of unlabeled data.

Class recurrence: Class recurrence is a special case of concept drift and evolution. A this case, the model forgets a class due the drift, however, later the class reappears (evolution) from the stream. Seasonality could be one cause of this situation. A intrusion in network traffic may reappear after a long time. Forgetting the earlier intrusions are not desired in such case. A fast recognition of the previously seen classes are desired in mining streams.

Following sections discuss the potential solutions to these challenges. First, how to address the limited resources and then the change detection schemes.

2.2.2 Maintaining Sufficient Statistics

In statistical evaluation, a statistic is sufficient for a family of probability distributions if the sample from which it is calculated gives no additional information than does the statistic, as to which of those probability distributions is that of the population from which the sample was taken [Fisher, 1922]. Mathematically, given a set $\mathbf{X}$ of independent identically distributed data conditioned on an unknown parameter θ, a sufficient statistic is a function $T(\mathbf{X})$ whose value contains all the information needed to compute any estimate of the parameter (e.g. a maximum likelihood estimate). Due to the factorization theorem (see below), for a sufficient statistic $T(\mathbf{X})$, the joint distribution can be written as $p(\mathbf{X}) = h(\mathbf{X})g(\theta, T(\mathbf{X}))$. From this factorization, it can easily be seen that the maximum likelihood estimate of θ will interact with $\mathbf{X}$ only through $T(\mathbf{X})$. Typically, the sufficient statistic is a simple function of the data, e.g. the sum of all the data points.

Bounds of Random Variable

An estimator is a function of the observable sample data that is used to estimate an unknown population parameter. We are particularly interested in interval estimators that compute an interval for the true value of the parameter, associated with a confidence $1-\delta$. Two types of intervals are:

- Absolute approximation: $\bar{X} - \epsilon \leq \mu \leq \bar{X} + \epsilon$, where ϵ is the absolute error.
- Relative approximation: $(1-\delta)\bar{X} \leq \mu \leq (1+\delta)\bar{X}$, where is the relative error.

2.2.3 Change Detection

2.2.4 Naïve Bayes Adaptation

2.2.5 Very Fast Decision Tree

2.3 Ensemble Learning

Ensemble learning is a commonly used tool for building prediction models from data streams, due to its intrinsic merits of handling large volumes stream data. Different from

traditional incremental and online learning approaches that merely rely on a single model [11, 38], ensemble learning employs a divide-and-conquer approach to first split the continuous data streams into small data chunks, and then build light-weight base classifiers from the small chunks. At the final stage, all base classifiers are combined together for prediction. By doing so, an ensemble model can enjoy a number of advantages, such as scaling up to large volumes of stream data, adapting quickly to new concepts, achieving lower variances than a single model, and easily to be parallelized. Ensemble classifiers on data streams provide a generic framework for handling massive volume data streams with concept drifting. The idea of ensemble classifiers is to partition continuous data streams into small data chunks, from which a number of base classifiers are built and combined together for prediction. Two main motivations for combining classifiers are as follows:

Statistical (or worst case) motivation: It is possible to avoid the worst classifier by averaging several classifiers. It was confirmed theoretically by Fumera and Roli in [39]. This simple combination was demonstrated to be efficient in many applications. There is no guarantee,however, the combination will perform better than the best classifier.

Representational (or best case) motivation: Under particular situations, fusion of multiple classifiers can improve the performance of the best individual classifier. It happens when the optimal classifier for a problem is outside of the considered classifier space". There are many experimental evidences that it is possible if the classifiers in an ensemble make different errors. This assumption has a theoretical support in some cases when linear combination is performed.

2.3.1 Bagging

2.3.2 Boosting

2.3.3 ASHT

2.3.4 ...

Bibliography

[Aggarwal et al., 2003] Aggarwal, C., Han, J., Wang, J., and Yu, P. (2003). Clustream: A framework for clustering evolving data streams. In *VLDB*.

[Aggarwal et al., 2004] Aggarwal, C., Han, J., Wang, J., and Yu, P. (2004). On-demand classification of data stream. In *ACM KDD*, pages 503–508.

[Aggarwal, 2003] Aggarwal, C. C. (2003). A framework for diagnosing changes in evolving data streams. In *ACM SIDMOD*, pages 575–586.

[Bifet et al., 2010a] Bifet, A., Frank, E., Holmes, G., and Pfahringer, B. (2010a). Accurate ensembles for data streams: Combining restricted hoeffding trees using stacking. 13:225–240.

[Bifet et al., 2010b] Bifet, A., Holmes, G., and Pfahringer, B. (2010b). Leveraging bagging for evolving data streams. In *ECML PKDD*, pages 135–150.

[Bifet et al., 2009] Bifet, A., Holmes, G., Pfahringer, B., Kirkby, R., and Gavaldà, R. (2009). New ensemble methods for evolving data streams. In *SIGKDD*, pages 139–148.

[Breiman, 1993] Breiman, L. (1993). Stacked regression.

[Breiman, 1994] Breiman, L. (1994). Bagging prediction.

[Breiman et al., 1984] Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (1984). Classification and regression trees.

[Brzezinski and Stefanowski, 2011] Brzezinski, D. and Stefanowski, J. (2011). Accuracy updated ensemble for data streams with concept drift. In *HAIS*, pages 155–163.

[Catlett, 1991] Catlett, J. (1991). Megainduction: Machine learning on very large databases. In *PhD thesis, University of Sydney.*

[Dasu et al., 2004] Dasu, T., Krishnan, S., Venkatasubramanian, S., and Yi, K. (2004). An information-theoretic approach to detecting changes in multi-dimensional data streams. In *Duke University Technical Report CS-2005-06*, pages 180–191.

[Ding et al., 2002] Ding, Q., Ding, Q., and Perrizo, W. (2002). Decision tree classification of spatial data streams using peano count trees. In *ACM Symposium on Applied Computing*, pages 413–417.

[Domingos and Hulten, 2000] Domingos, P. and Hulten, G. (2000). Mining high-speed data streams. In *Proceedings of the ACM KDD*.

[Drucker et al., 1994] Drucker, H., Cortes, C., Jackel, L. D., LeCun, Y., and Vapnik, V. (1994). Boosting and other ensemble methods. 6(6):1289–1301.

[Fisher, 1922] Fisher, R. A. (1922). On the mathematical foundations of theoretical statistics. 222:309–368.

[Freund and Schapire, 1997] Freund, Y. and Schapire, R. (1997). A decision theoretic generalization of on-line learning and an application to boosting. 55(1):119–139.

[Ganti et al., 2002] Ganti, V., Gehrke, J., and Ramakrishnan, R. (2002). Mining data streams under block evolution. 3(2):1–10.

[Hansen and Salamo, 1990] Hansen, L. K. and Salamo, P. (1990). Neural network ensembles. 12(10):993–1000.

[Hoeffding, 1963] Hoeffding, W. (1963). Probability inequalities for sums of bounded random variables. 58:13–30.

[Hulten et al., 2001] Hulten, G., Spencer, L., and Domingos, P. (2001). Mining time changing data stream. In *ACM KDD*, pages 97–106.

[Kifer et al., 2004] Kifer, D., Ben-David, S., and Gehrke, J. (2004). Detecting changes in data streams. In *VLDB*, pages 180–191.

[Krogh and Vedelsby, 1995] Krogh, A. and Vedelsby, J. (1995). Neural network ensembles, cross validation and active learning. pages 231–238.

[Last, 2002] Last, M. (2002). Online classification of non-stationary data streams. 6(2):127–147.

[Mehta et al., 1996] Mehta, M., Agrawal, A., and Rissanen, J. (1996). Sliq: A fast scalable classifier for data mining. In *Extending Database Technology*, pages 18–32.

[Oza, 2001] Oza, N. C. (2001). Online ensemble learning. In *Ph.D. thesis, Department of EECS, UC Berkeley*.

[Oza and Russell, 2001] Oza, N. C. and Russell, S. (2001). Online bagging and boosting. In *Artificial Intelligence and Statistics*, pages 105–112.

[Parhami, 1996] Parhami, B. (1996). Voting algorithms. 43(4):617–629.

[Pelossof et al., 2008] Pelossof, R., Jones, M., Vovsha, I., and Rudin, C. (2008). Online coordinate boosting.

[Quinlan, 1993] Quinlan, J. R. (1993). C4.5: Programs for machine learning.

[Schapire, 1990] Schapire, R. (1990). Strength of weak learnability. 5(2):197–227.

[Shafer et al., 1996] Shafer, J. C., Agrawal, R., and Mehta, M. (1996). Sprint: A scalable parallel classifier for data mining. In *VLDB*, pages 544–555.

[Tumer and Ghosh, 1999] Tumer, K. and Ghosh, J. (1999). Linear and order statistics combiners for pattern classification. In *A. J. C. Sharkey, editor, Combining Artificial Neural Nets: Ensemble and Modular Multi-Net Systems*, pages 127–162.

[Tumer and Oza, 1999] Tumer, K. and Oza, N. C. (1999). Decimated input ensembles for improved generalization. In *IJCNN*, pages 105–112.

[Wang et al., 2003] Wang, H., Fan, W., Yu, P. S., and Han, J. (2003). Mining concept-drifting data streams using ensemble classifiers. In *SIGKDD*, pages 226–235.

[Wolpert, 1992] Wolpert, D. H. (1992). Stacked generalization. 5:244–259.