A Short Introduction to Boosting

In terms of machine learning, there is often the case where the rules used are quite weak. Boosting attempts to strengthen and combine those rules into one that is highly predictive. There have been many attempts at developing a useful boosting method, but the authors focus on their own AdaBoost which was the most efficient and adaptively changed depending on the strength of the rules. AdaBoost repeatedly used the weak learning rules on a training set, maintaining a distribution of weights that tended to narrow the focus of the algorithm onto those cases which were difficult to learn. The final rule was then created using an amalgamation of individual rules weighted appropriately.

AdaBoost was able to ensure that as long as the weak rules were better than random guessing, then the training error was reduced quickly. Even if it was unknown how much better the rules were over random guessing (a problem limiting other methods), AdaBoost still performed well. Another problem that AdaBoost avoided was overfitting data, which apparently causes rules to become too specific for the data if the boosting rounds are done too many times. Instead, AdaBoost reduces this generalization error and the margin, (a measure of confidence), by focusing on those cases which were not classified correctly. This differed slightly from support vector machines, mainly in the way that the margins were calculated and minimized and in the amount of computation required.

An important issue with the original AdaBoost algorithm was its inability to deal with non-binary classification. With extra effort though, it was extended to handle multi-classifications, by rephrasing binary questions in the form of does x belong to label y or some other label? With this addition, AdaBoost had significant benefits over previous methods, such as not requiring tuning and working with any kind of data. As long as there were rules that did better than random, the algorithm could strengthen those rules accordingly. AdaBoost did have some problematic issues, such as focusing too much attention on outliers (a natural extension of focusing on hard cases), which tended to skew results. Moreover, if the presence of noise was significant, AdaBoost failed to accurately classify data. Both problems were mitigated by further algorithms which ignored or de-emphasized data that was too hard to handle.
If events within turbulence can actually be predicted (with data from numerical simulations for measuring accuracy), then researchers can begin to answer fundamental questions on the evolution of structures within a flow, a goal which has great practical significance. Data mining, in terms of clustering, pattern discovery and predictive modeling can all add to this sum of knowledge. The authors focused on predicting velocity of a flow close to the wall, which has the potential to explain bursting behavior.

The specific goal of predicting movement is to obtain one value as a function of other values that are available. To do so, a training set is needed that should operate on a well defined data format that incorporates information from neighboring cells. The cells are part of the grid that is overlayed on the data. Attributes are associated with the cells that indicate the presence or absence of a swirl. Alternatively, data can have attributes that represent the number of swirls within a range of a grid point.

In the authors simulations, they had 10 time steps, each with a large number of transactions (representing grid points and their attributes). The first part of the data was used as training and the latter part used to test prediction ability. By using two methods for machine learning, (C4.5 rules and support vector machines), explanatory models were created. Different sized datasets were used, with various neighbor sizes. The training tried to classify whether velocity would go up, down or stay stationary. They found that they could detect with reasonable accuracy when a point would stay still. More interesting, was that a numerical set of rules for accurate classifiers shows exactly when an event would occur. For example, they showed that if there are many swirl counts in neighboring cells, then there will be downward movement.

A support vector machine was also used on the same data sets which seemed to offer better precision (number of correctly classified in a class compared to all classified to that class) and recall (number of correctly classified in a class compared to those actually in the class). Unfortunately, the authors made the case that SVM could not be used for further understanding, as it was not simple to extract rules from the results (as it was while using C4.5 rules). They did identify a few cases where bursting could be reliably predicted and offered some reasons, but in general their technique had a limited ability to explain the fundamental principles of turbulence.