

A Method for Automatically Generating Network Structure in Bayesian Classifier

Lionel Barrow

Adviser: Cliona Golden

Mathematics Program, Bard College, May 2011

Abstract

The Naive Bayes algorithm for classification has been a staple in machine learning for decades. Simple and efficient, the algorithm makes unrealistic independence assumptions about the data; yet it performs very well, often nearly matching the performance of far more complex modern algorithms. Only recently have researchers understood the theoretical reasons for this unreasonably good performance. In 2004, Professor Harry Zhang of the University of New Brunswick articulated the notion of a dependence-derivative factor, which more precisely defines how much Naive Bayes is harmed by certain violations of its independence assumption. In this project, I present a way to use Zhang's dependence derivatives to create classifiers similar to Naive Bayes, but with a network structure more resilient of these violations.

Introduction

Classification is the process of examining something and deciding what it is. A typical classification problem might involve being given a description of an object – say a piece of fruit – and being asked to determine what type of fruit is being described. While humans are able to classify objects without much thought, developing algorithms to enable machine classification is an ongoing problem.

Algorithms that perform classification known as **classifiers** can quickly and quietly reduce the number of inconsequential decisions in our lives. Probably the most well-known example of a classifier is an email spam filter, which simply determines if an incoming email matches the known profile of an unwanted advertisement. Other common classification problems include medical diagnosis and handwriting recognition.

Bayesian Classifiers

A **Bayesian Classifier** f_b is a function from the space of possible examples to the space of possible classes. For an **example**, $E = (x_1, x_2, \dots, x_n)$,

$$f_b(E) = \operatorname{argmax}_c p(c|E),$$

where $p(c|E)$ indicates the probability of a class c given the example E . Evaluating $f_b(E)$ requires us to calculate complicated probabilities of the form $p(x_1, x_2, x_3, \dots, x_n, c)$. This cannot be done without making assumptions about the way the columns of data (i.e. the attributes) relate to each other.

Our Bayesian Classifier

For the purposes of our project, we assumed that attributes can be related to each other using a simple linear model. We assume that if an attribute value x_a is dependent on another attribute x_b , then the dependent value of x_a is the sum of x_b and some other, independent part of x_a .

$$x_a^{\text{dependent}} = x_a^{\text{independent}} + \operatorname{cov} \left(\frac{x_a^{\text{dependent}}}{x_a^{\text{dependent}}}, \frac{x_b - \mu_b}{\sigma_b} \right) \frac{x_b - \mu_b}{\sigma_b},$$

where, if X and Y are random variables, $\operatorname{cov}(X, Y)$ is the covariance of X and Y . The classifier using this model of data codependence is called the **full Bayesian classifier** or **full Bayes network**. While it is unrealistic to believe that data actually interact this way in the real world, we can easily generate artificial sets of data that fit this assumption. Then, later on, we can compare the performance of this classifier with others, knowing that this classifier is sure to do well.

Naive Bayes

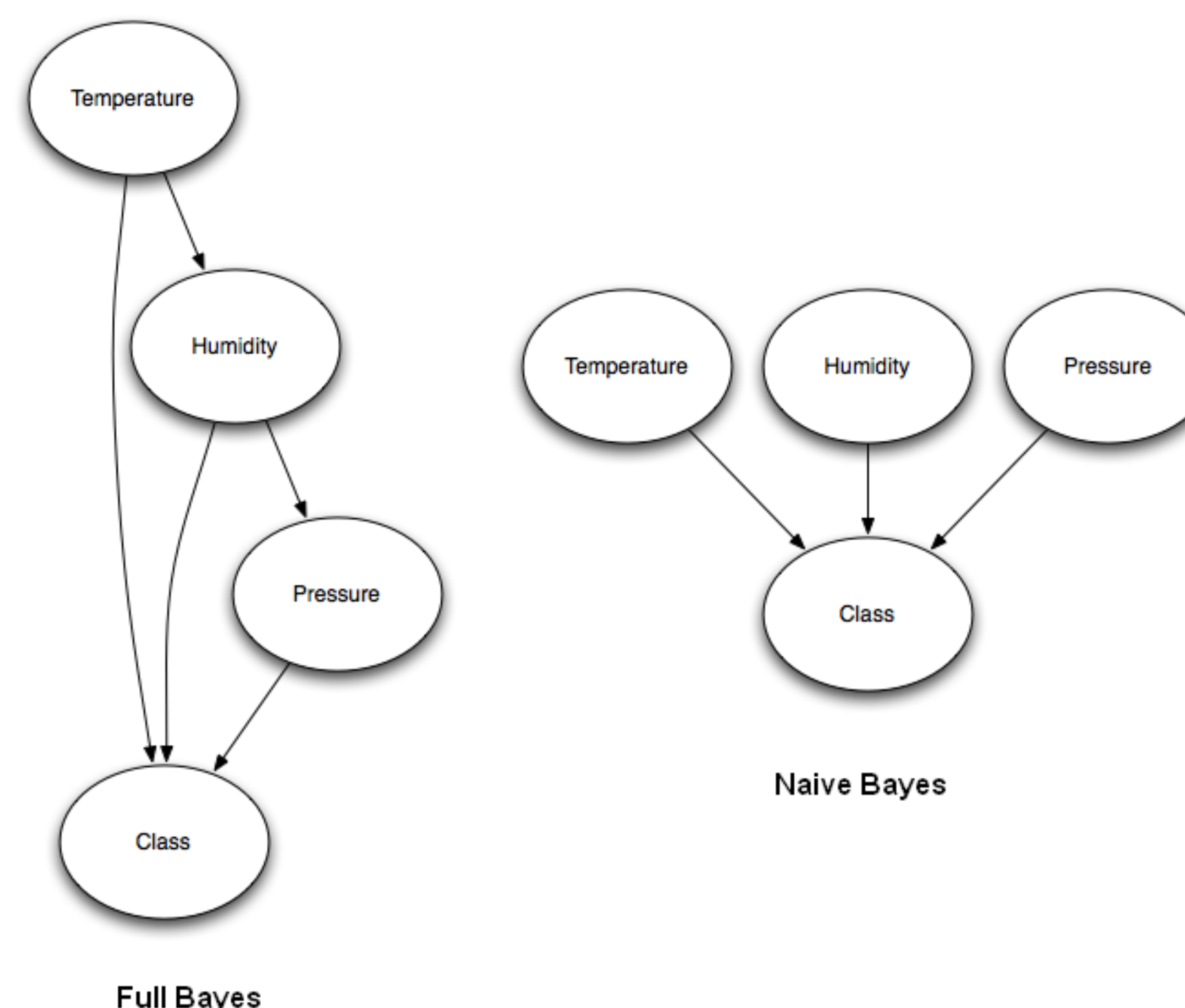
Another classifier is **Naive Bayes**, which assumes that attributes are independent of each other. Naive Bayes can therefore calculate the likelihood of a class as

$$p(E|c) = p(x_1, x_2, \dots, x_n|c) = \prod_{i=1}^n p(x_i|c).$$

While this calculation is extremely simple and fast, the assumption that enables it is unrealistic. Nevertheless, Naive Bayes performs very well, for reasons that remained unclear until relatively recently.

Full and Naive Bayes as Networks

We can illustrate the difference between full and Naive Bayes by examining the graph of how they represent attributes internally. Suppose we were trying to predict whether or not it would rain in Red Hook tomorrow. We might have variables like temperature, pressure and humidity as attributes and the likelihood of rain as a class. The graphs below depict how full and Naive Bayes would interpret the relationship between these vertices: a line with an arrow means that causality is moving from one vertex to another.



Dependence Derivatives

In 2004, Harry Zhang of the University of New Brunswick showed that the unreasonably good performance of Naive Bayes can be explained by two possible situations [1].

- Attributes in the data actually are independent of each other. In this case, Naive Bayes' assumption is not unreasonable at all, and we expect it to perform well.
- Dependence between attributes that pushes the classifier to believe an example is in one class is **balanced** by additional dependence between other attributes that pushes the classifier to believe the example is in another class. In other words, the dependence between attributes is canceled out by being evenly distributed among the classes.

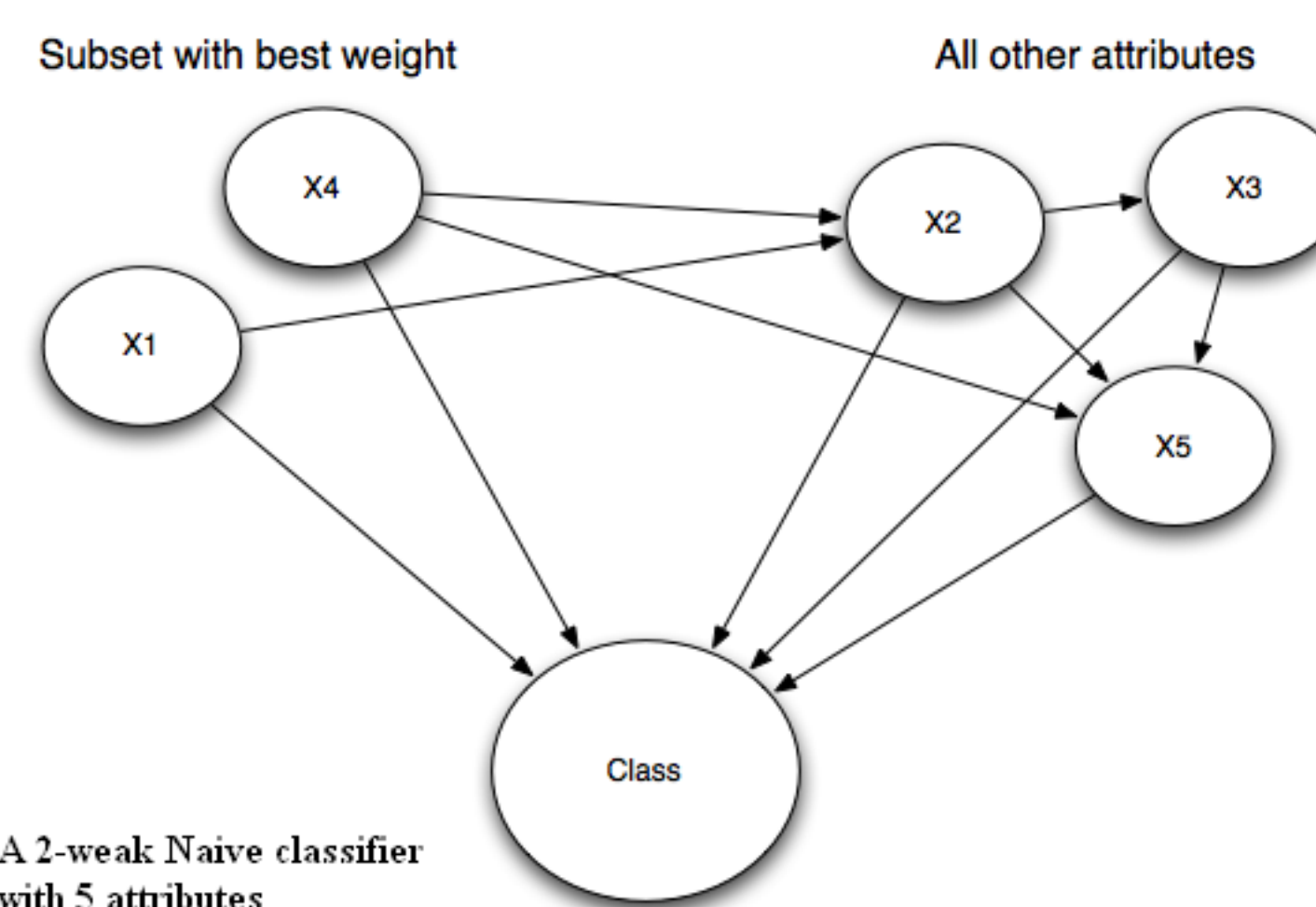
Zhang defined a measure of the balance between classes called the **dependence derivative ratio** and showed that, in a given example, as the product of dependence derivative ratios approached 1, Naive Bayes performance increased.

My project was the use this insight to create hybrid classifier that optimized performance regardless of whether or not these conditions were fulfilled.

Weak k-Naive Bayes

Our hybrid algorithm, **weak k-Naive Bayes**, sets k attributes aside to process in a manner analogous to Naive Bayes, leaving the remaining $n - k$ attributes in a full Bayes network. In Naive Bayes proper, these k attributes are independent of all other attributes; however, we found that this was too strong a condition. Instead, we require that the k attributes be independent of each other – they can still affect attributes not in the naive set.

The topology of a weak 2-Naive Bayes algorithm in a 5-attribute classification problem is shown below.



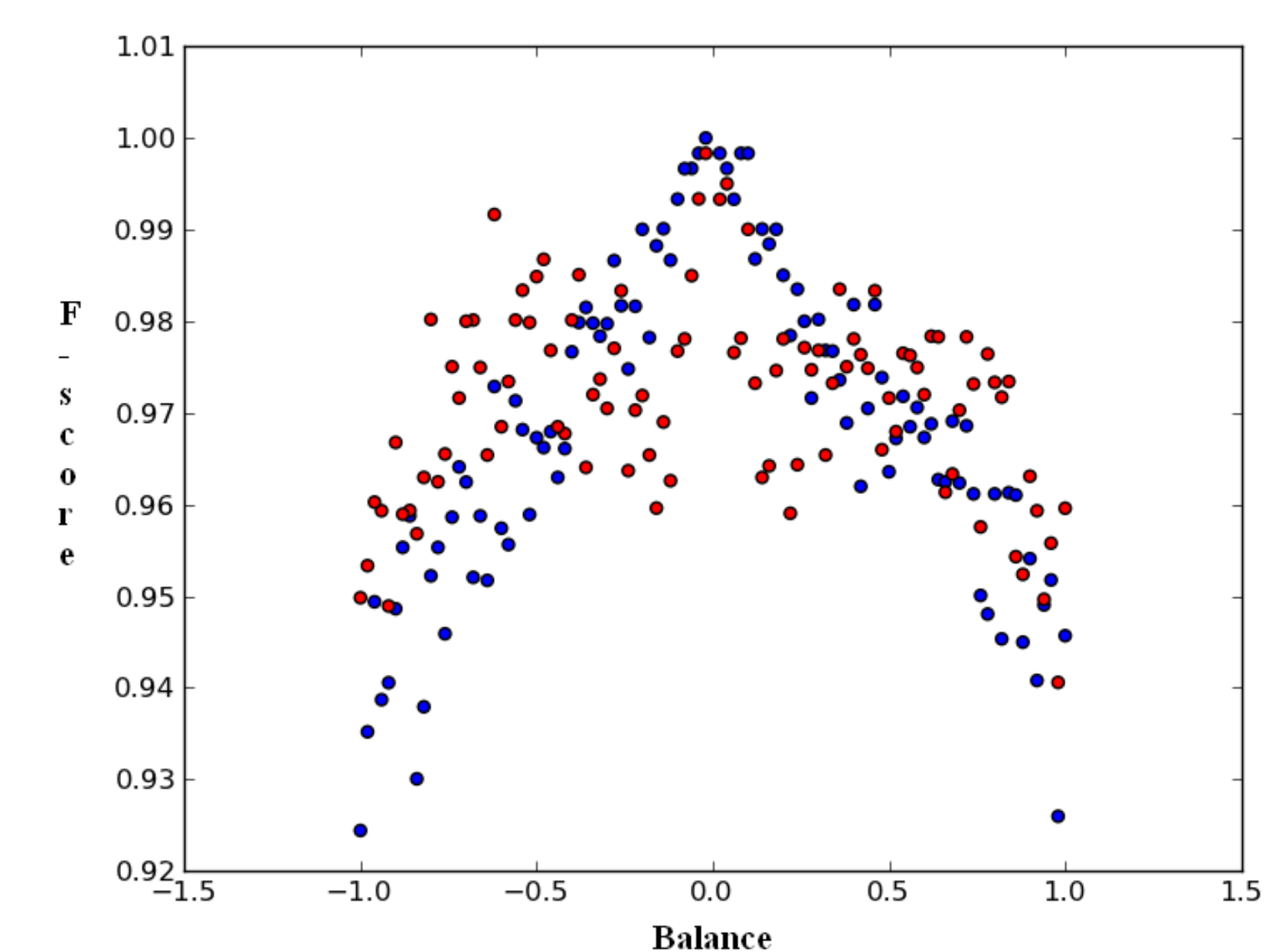
Notice that while X_1 and X_4 send signals to the other attributes, they never send signals to each other.

Methods

We created a series of Python programs implementing these classifiers. We then generated artificial datasets with a desired property, and observed how classifier performance varied as this property varied.

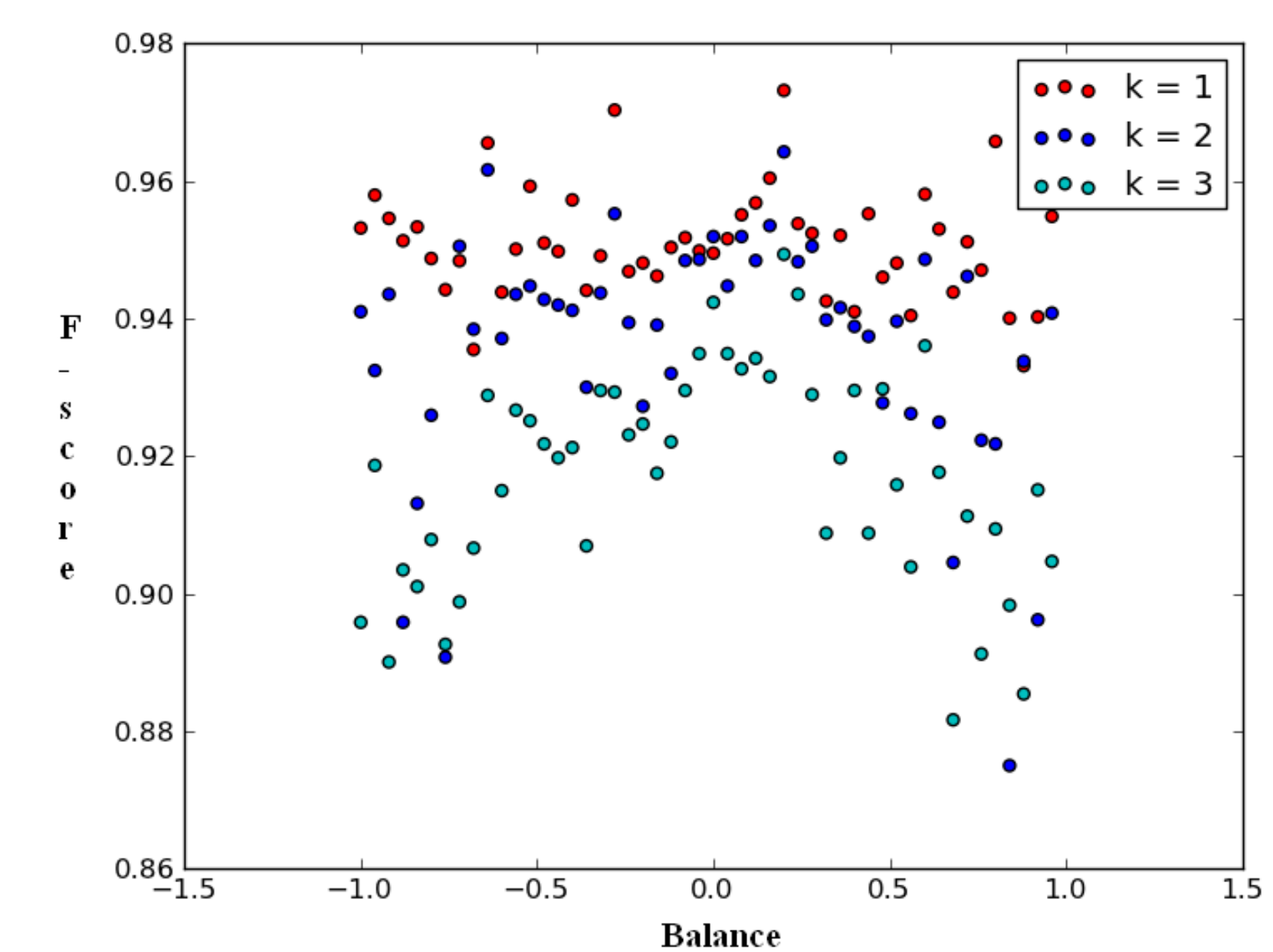
Results

We succeeded in demonstrating the performance variations predicted by Zhang's article on dependence derivatives. The following scatterplot depicts classifier performance for Naive Bayes (blue) and full Bayes (red) as "balance," a measure of covariance among attributes, is changed. The Y-axis, F-score, scales from 0 to 1 as an overall measure of classifier performance.



Notice the performance of Naive Bayes improves as the absolute value of balance approaches 0. This is exactly what we expect to see, as is the relative lack of change in full Bayes' performance.

The next question to ask is, how do k -Naive Bayes perform on similar balance problems? The scatterplot below examines this. We expect that for high values of k , the shape of the plot will resemble that of Naive Bayes (blue) above, and that as k decreases, the plot will increasingly resemble full Bayes (red).



That is exactly the pattern we expect to see. This plot shows that our classifiers are working as intended, and that theory behind them is sound.

Conclusions

We succeeded in building classifiers that can compensate for Naive Bayes' inability to deal with unbalanced covariance in data! These classifiers have potential applications in lots of fields where fast, reliable classification is needed.

Works Cited

- Zhang, Harry. *The Optimality of Naive Bayes*. Proceedings of the 17th International FLAIRS Conference, 2004.