PROJECT 1 - WINNOW2 AND NAIVE BAYES

DAVID ATLAS

Abstract. This paper will introduce the Winnow2 algorithm and the Naive Bayes algorithm. Both will be treated in context of classification problems, with strategies for categorical and continuous valued inputs. We find that in several experiments on real-world datasets, both algorithms perform comparably, especially when Naive Bayes is limited to booelan inputs and outputs.

1. Problem Statement & Hypothesis

We seek to introduce two basic supervised learning methods, and apply them to classification problems. The first, Winnow2, is a very simplistic learning model for boolean inputs and outputs. The second, Naive Bayes, is a simple graphical model that can handle multivalued discrete and continuous inputs. We will compare both models using Boolean-values inputs and outputs to make the comparison more fair, while also applying Naive Bayes in its more flexible context with multivalued discrete and continuous inputs. We expect that Naive Bayes will provide slight performance inprovements over Winnow2, as its representation space is larger, and we expect that the performance gap will grow when Naive Bayes is applied with multivalued discrete and continuous inputs.

2. Description of Algorithms

Winnow2. The Winnow2 Algorithm is a linear-threshold boolean classifier, in which a set of boolean inputs is mapped to a single boolran output. The algorithm works as follows:

Given a boolean input vector, $X = (x_1, \cdots, x_k)$, create a vector of weights $W = (w_1, \cdots, w_k)$. We also initialize two parameters, θ and $\alpha > 1$. For each example, we calculate $Z(X, W) = \sum_{i=1}^{k} x_i w_i$. If $Z \geq \theta$, we predict a 1, and if $Z < \theta$, we predict a 0.

Our learning mechanism is composed of two operations - promotion and demotion. Under promotion, we update $w_i = \alpha w_i \forall i \mid x_i = 1$. Under demotion, we update $w_i = \frac{w_i}{\alpha} \forall i \mid x_i = 1$. If our prediction is a 1, while the correct response is a 0, we apply demotion. If our prediction is a 0 while the correct response is a 1, we apply promotion. This has the effect of raising the value of Z when our function predicts a value that is too small, and lowers the value of Z when our function predicts a value that is too large.

Naive Bayes. The Naive Bayes algorithm is a graphical model that leverages Bayes' Rule and some simplifying assumptions. To recap, under Bayes' Rule:

$$P(C \mid X) = \frac{P(X \mid C)P(C)}{P(X)}.$$

Suppose C is the random variable representing the class of an instance, while X is a set of random variables representing feature values of a given instance. In creating a classifer, we want to know that likelihood that a given instance belongs to class C, given its feature values X, and so if we can calculate the right hand side of Bayes' Rule, we have a classifier that reflects our goal.

To make the equation above more managable, a simplifying assumption can be made - suppose all of the features are independent conditional on the class. Then we can say that for a set of k feature values,

$$P(C \mid X_i, \cdots, X_k) = \frac{P(X_i, \cdots, X_k \mid C)P(C)}{P(X_i, \cdots, X_k)} = \frac{\prod_{i=1}^{k} P(X_i \mid C)P(C)}{P(X_i, \cdots, X_k)}$$

Note that the denominator is simply a normalizing constant with respect to the class that we choose for our instance, and so it does not effect the

We can define our classification function as choosing the value of C that is most likely, given the feature values, and by extension:

$$\arg\max_C P(C \mid X_i, \cdots, X_k) = \arg\max_C \prod_{i=1}^{k} P(X_i \mid C)P(C).$$

With that background, $P(X_i \mid C)$ and $P(C)$ must be estimated. If X_i is a discrete random variable, $P(X_i \mid C)$ can be described as multinomial, where p is the proportion of each value of X_i for training instances of class C. If X_i is continuous, $P(X_i \mid C)$ can be described as Gaussian, with a mean of the sample mean of X_i for training instances of class C and a standard deviation of the sample standard deviation of X_i for training instances of class C.

$P(C)$ is multinomial, with p equal to the proportion of total training examples of class C. $P(C)$ would be binomial if it is boolean under this model.

Note that all of these distributional family assumptions can be changed if appropriate, but are generally sensible defaults where no other distributional information is known.

Smoothing can be applied in situations where there are no training examples of class C that take on a value $X_i = Q$ (where Q is a constant). such that there is a non-zero likelihood for $P(X_i \mid C)$, which can lead to better generalization. For a multinomial distribution, $p_i = \frac{\sum_{i \in C} x_i + \alpha}{\sum_{i=1}^{n} x_i + \alpha m}$, where $\alpha = 1$ and m is the number of values in X_i. Obviously, this can be done in many other ways, but this is a sensible default. For a continuous distribution, you can create the

After calculating the conditional distributions of the training examples, the prediction is simply

$$\arg\max_C \hat{P}(X_i \mid C)\hat{P}(C),$$

where $\hat{P}$ indicates the estimated distribution from the training process.