Eliciting Event Prediction via Expert Predictions

Alexander L. Churchill
achur@stanford.edu

Simon H. Ye
sye@stanford.edu

Dec. 12, 2011

1 Introduction

The task of predicting real-world events via Machine Learning is a relatively recent innovation in the field. Within the past decade, there has been fairly significant work regarding the theoretical backing of predictions, even among potentially irrational agents. David Pennock lays the groundwork for deterministic algorithms to reduce risk and provide proper incentives for eliciting truthful expert predictions, even in markets which are not zero-sum [1]. More recently, Lambert provides a framework for eliciting true probability distributions from rational experts [2].

Despite this theoretical background, there has been little work in actually eliciting probability distributions from expert predictions. Eliciting such distributions must overcome two challenges. First, any machine learning agent must have a proper reward function to maximize the success rate of predictions. Second, a classifier must be resistant to a high noise-to-expert ratio: that is, even when true experts only make up a small percent of the prediction population, the classifier must still perform well. In this preliminary study, we address the first challenge by limiting the task to binary events; for whom there are two possible predictions, and maximizing over the percentage of correct predictions.

We curry a dataset, mining predictions on NFL football games, from weeks 1-14 of the 2011 football season. We implement a number of machine learning algorithms, and analyze the classification accuracy of each under various dataset conditions. We discuss our dataset and methodology before revealing our preliminary results. We conclude with some brief discussion and error analysis.

2 Dataset Description

The core dataset consisted of approximately 3,000 data points curated from expert predictions on NFL football games from the 2011 football season. The experts logged predictions publically on sports sites *espn.com* and *sports.cbs.com*. In addition to thirteen expert predictions, two game prediction services logged their (presumably aggregate) predictions as well. Each game paired two consistent data points (the home and away teams) as well as a data point for each expert predictor and a data point logging the winner.

We further curried this into four additional datasets by adding “experts” in quantities of 10, 50, 100, and 1000. Each random “expert” randomly predicted the outcomes of the games. These datasets were designed to test each algorithm with a variety of noise-to-expert ratios.

Each dataset was broken into two pieces: a test and training set, broken by time (i.e. the training set is composed entirely of games which occurred prior to every game in the test set).

3 Algorithm Descriptions

3.1 Baselines

3.1.1 Majority Rule

The most straightforward baseline we used simply ignores the training set and makes whichever prediction is most popular.

3.1.2 Top Expert

This baseline determines the expert who was most accurate during the training set, then makes the same predictions in the test set.

3.2 Naïve Bayes

The simplest algorithm we implemented was the Naive Bayesian classifier.

The Bayesian classifier is as follows: given a training set:

$$\big((X^{(n)}, Y^{(n)}), ..., (X^{(n)}, Y^{(n)})\big)$$

where each $X^{(i)}$ is a vector of features and each $Y^{(i)}$ is a result of the corresponding game $r \in \{-1, 1\}$, we let $W = \{w_1, ..., w_p\}$ be the set of all NFL teams in the training set.

For each $(X^{(i)}, Y^{(i)})$, define $\eta_j^{(i)} = \sum_{k=1}^{n} \sum_j 1_{\{X^{(i)}|Y^{(i)}=1\}}$ and define $\hat{\eta}_j^{(i)} = \sum_{k=1}^{n} \sum_j 1_{\{X^{(i)}|Y^{(i)}=-1\}}$. The multinomial Bayesian classifier applied to a single $(X^{(i)}, Y^{(i)})$ yeilds:

$$P(Y_k^{(i)} = 1 | w_j \in X_k^{(i)}) = \frac{\eta_j^{(i)}}{\eta_j^{(i)} + \hat{\eta}_j^{(i)}} \text{ and } P(Y_k^{(i)} = -1 | w_j \in X_k^{(i)}) = \frac{\hat{\eta}_j^{(i)}}{\eta_j^{(i)} + \hat{\eta}_j^{(i)}}$$

so we derive that

$$P(Y_\ell^{(i)} = 1 | X_\ell^{(i)}) = \frac{\prod_{w_j \in X_k^{(i)}} P(w_j | Y_k^{(i)} = 1) P(Y^{(i)} = 1)}{\prod_{w_j \in X_k^{(i)}} P(w_j | Y_k^{(i)} = 1) P(Y^{(i)} = 1) + \prod_{w_j \in X_k^{(i)}} P(w_j | Y_k^{(i)} = -1) P(Y^{(i)} = -1)}.$$

In short, for each unclassified game, we observe the probability of an outcome $w \in W$ given the home/away team features and the prediction panel.

3.3 SVM

Support Vector Machines make the observation that for linearly separating data, it is possible to consider only the support vectors of a given dataset to make predictions. In this case, each element in the dataset was treated as an n-dimensional prediction point, where n is the number of experts. We used *libSVM*, a popular SVM library widely available for data classification using SVMs, to perform the analysis.

3.4 Logistic Regression

In the logistic regression model, each feature was mapped to a logistic function which takes values between 0 and 1. Each element in the dataset was treated as an n-dimensional prediction point, where n is the number of experts. We classified a logistic curve with the following update rule:

$$\theta_{j+1} = \theta_j + \alpha \underline{x}_j^T (y_j - f(x_j, \theta_j))$$

where $x_j \in \{0,1\}^n$ is the vector of predictions of game j, y_j is the outcome of game j, and f is a function which computes the logistic function parametrized by θ_j and returns 1 if the outcome is above 0.5, 0 otherwise. We then use $f(\cdot, \theta)$ to classify points in the test set, where θ is the last-updated classifier parameterization.

For testing, we used the open-source Java SimpleLogistic classifier.

3.5 Bayesian Network

In the Bayesian Network classifier, we attempted to determine any structure to the datapoints (e.g. if one expert's predictions were influenced by another expert's), then to explit this structure to improve classification accuracy.

Initially, we learned the Bayesian Network using the standard hill-climbing learning algorithm with random graph changes, optimizing over the BIC score of the network, defined as

$$score_{BIC}(\mathcal{G} : \mathcal{D}) = \ell(\mathcal{G} : \mathcal{D}) - \frac{\log M}{2}\mathrm{Dim}[\mathcal{G}]$$

where ℓ is the likelihood score of the graph given the data and M is the number of datapoints.

We then used this network with the standard Bayesian inference algorithm to predict the outcome on the datapoints.

4 Experimental Proceedure

Testing was performed on each test set over each algorithm. Each algorithm was given a training set and tested over a test set, split in week 7.

5 Results

Percentage accuracy:

Algorithm	Only Experts	10 Noise	50 Noise	100 Noise	1000 Noise
Baselines					
Majority Rule	59.24%	60.98%	60.16%	56.91%	47.97%
Top Predictor	60.16%	60.16%	60.16%	60.16%	52.03%
Algorithms					
Naïve Bayes	65.32%	65.32%	67.74%	64.51%	61.29%
SVMs	63.49%	61.29%	62.09%	62.90%	60.48%
Logistic Regression	64.51%	64.51%	64.51%	64.51%	64.51%
Bayesian Network	60.48%	62.09%	64.51%	62.90%	55.64%

6 Conclusion

In analysis of the results, the outcomes of the baselines are unsurprising. The majority rule baseline performs fairly well with low noise, but as the ratio of noise-to-experts increases from 3.3 to 6.6, it begins to decrease and by the noise-to-experts ratio of 66.6, it is completely ineffectual, actually performing slightly worse than random. The top predictor actually is one of the original experts in all but the highest noise level, and predicts with an accuracy of around 0.6. We can estimate the probability a random predictor will perform better than 60% on the training set with the Normal approximation to $X \sim Bin(100, 0.5)$ to be approximately $1 - \Phi(2) \approx 2.27\%$, so for high noise, there is a high probability that a random predictor will set the best-so-far baseline.

In analysis of the results, it is clear that the best-performing algorithms are Naive Bayes and Logistic Regression. In many ways, this is unsurprising. Overall, determining the most accurate predictions will be a test in determining which features (individual experts) are the most accurate predictors. The Naïve Bayesian framework is ideal for this purpose and the Logistic Regression algorithm puts each of these features as a logistic parameter. SVMs performed nearly as well as the Naïve Bayesian classifier, while keeping the complexity of the classifier much lower and classification time much faster. The Bayesian Network algorithm performed similarly to Naïve Bayes for smaller datasets, but performed poorly on larger datasets, likely because it overfit the network structure in the training set.

Overall, the results are generally promising. Both Naïve Bayes and Logistic Regression substantially outperformed both baselines at all noise levels, and that their performance did not significantly drop off as noise increased (while baseline performance did). Even at high noise, our classifier was able to slightly outperform the CBS and ESPN experts, indicating that crowdsourcing expert predictions attached to topical tags (in this case the home and away teams) is a viable way to outperform any single expert.

References

[1] Pennock, David M. A Dynamic pari-mutuel market for hedging, wagering, and information aggregation. ACM Conference on Electronic Commerce. 2004.

[2] Lambert, Nicolas. Elicitation and Evaluation of Statistical Forecasts. Working paper. June 2011.