CS 5785 – Applied Machine Learning – Lec. 17

Prof. Nathan Kallus, Cornell Tech
Scribe: TBD

Oct. 31, 2017 (Under construction)

1 CART Recap

Recall the following pros and cons of Classification and Regression Trees.

Pros:

- Easy to interpret
- Readily handles mixed discrete and continuous inputs (example: in a survey)
- Insensitive to monotone transformations on input (since we are sweeping from minimum to maximum of the range.)
- Performs automatic variable selection
- Some robustness to outliers
- Can be modified to handle missing inputs

Cons:

- Accuracy is not as high as competing methods
- Instability in hierarchy: a bad split propagates the error down to all the splits below it

The worst of the cons is *instability*. Instability can be addressed through a technique called *bagging*, or bootstrap aggregating, which will lead us to the concept of *Random Decision Forests*.

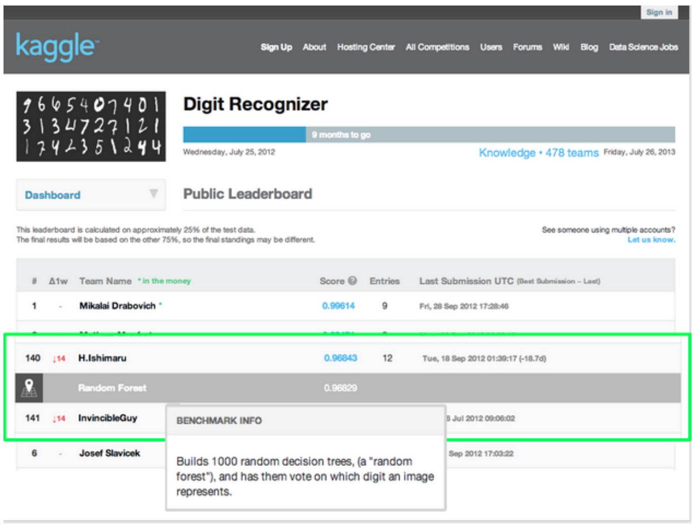
2 Random Decision Forests

Both in machine learning and in biology forests are defined in the same way:

Definition A forest is a set of trees.

Random Decision Forests (also called Random Forests) were first proposed as a technique for handwritten digit classification (see Fig. **??**). More recently they were used for human body pose estimation in Microsoft Kinect. The key idea is to use not one but many trees, each grown on different subsets of the data, to conquer the instability problem mentioned above.

The process of creating a random forest is as follows.



Digit Recognizer

Wednesday, July 25, 2012 — Knowledge • 478 teams — Friday, July 26, 2013

Public Leaderboard

This leaderboard is calculated on approximately 25% of the test data.
The final results will be based on the other 75%, so the final standings may be different.

See someone using multiple accounts?
Let us know.

#	Δ1w	Team Name * in the money	Score	Entries	Last Submission UTC (Best Submission – Last)
1	-	Mikalai Drabovich *	0.99614	9	Fri, 28 Sep 2012 17:28:46
140	↓14	H.Ishimaru	0.96843	12	Tue, 18 Sep 2012 01:39:17 (-18.7d)
		Random Forest	0.96829		
141	↓14	InvincibleGuy			5 Jul 2012 09:06:02
6	-	Josef Slavicek			Sep 2012 17:03:22

BENCHMARK INFO

Builds 1000 random decision trees, (a "random forest"), and has them vote on which digit an image represents.

Figure 1: Random Forest classification, shown here as a baseline approach for a digit recognition challenge, is a popular method on Kaggle.

- Train B different trees on different subsets of the data, sampling with replacement (allowing reuse of some data)
- Produce an ensemble of trees $\{T_b\}_1^B$
- To make a prediction at a new point x,
  - For regression use:

$$\hat{f}_{rf}^B = \frac{1}{B}\sum_{b=1}^{B} T_b(x)$$

  - For classification, let $\hat{C}_b(x)$ be the class prediction of the bth random forest tree. Then:

$$\hat{C}_{rf}^B = \text{majority vote}\{\hat{C}_b(x)\}_1^B$$

Up to this point this is just "bagging," or boostrap aggregation, simply rerunning the same learning algorithm on different subsets of the data. This can result in highly correlated predictors, limiting the amount of variance reduction. We need to decorrelate the base learners. We do this as follows. Before each split, select $m \leq p$ of the input variables at random as candidates for splitting. A typical value for m is $\sqrt{p}$. By choosing to get a sub-set of feature variables, we coerce the decision trees to become as diverse as possible. This helps us to obtain trees that have learnt some features better than the others, and avoid growing similar trees everytime.

Fig. **??** shows us the performance benefit of using Random Forests compared to Bagging.

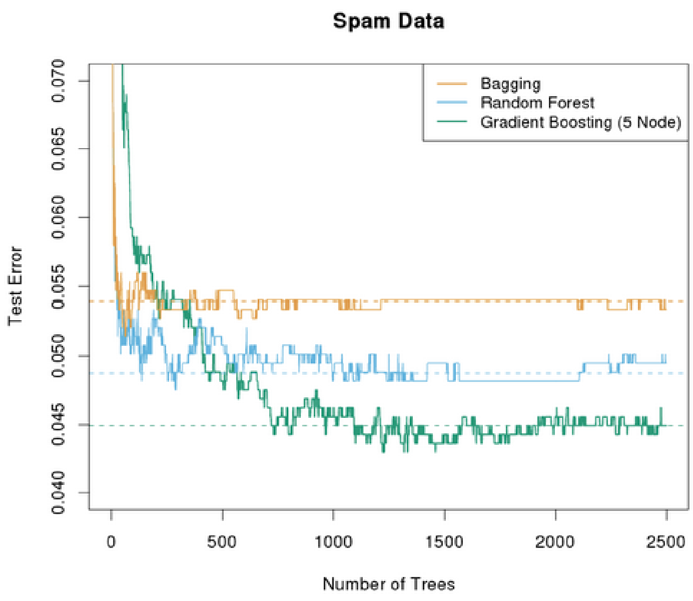


Figure 2: HTF Figure 15.1

3 Boosting

Boosting, one of the most influencial machine learning ideas of the last 20 years, was originally designed for classification but can be extended to regression. The basic idea behind boosting is to combine the outputs of many *weak* classifiers to produce a powerful *commitee*. Boosting is superficially related to bagging, since they both combine outputs from many classifiers, but is in fact fundamentally different.

Definition A weak classifier is a classifier with an error rate only slightly better than guessing.

The most popular implementation of a boosting algorithm is AdaBoost.M1 (Freund and Schapire 1997), see Fig. **??**.

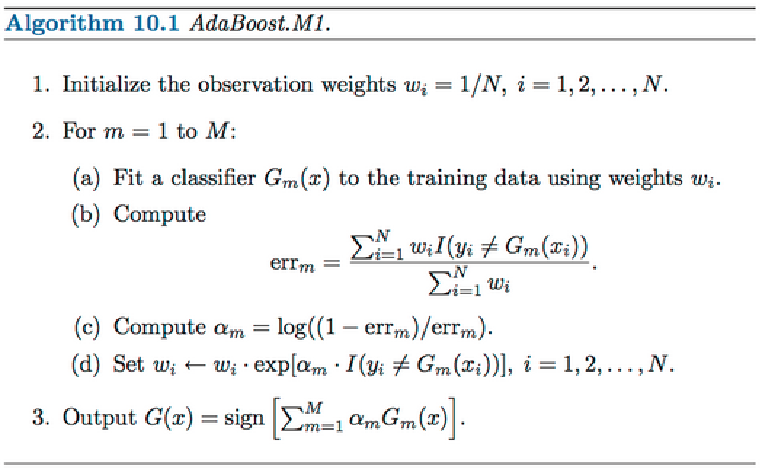
Consider the 2-class case with output variable coded $Y \in \{-1, 1\}$. Classifier $G(X)$ produces a prediction taking value 1 or -1. The error rate on the training data is defined as:

$$\overline{err} = \frac{1}{N} \sum_{i=1}^{N} I(y_i \neq G(x_i))$$

The expected error rate on future predictions (i.e., on the testing data) is defined as:

$$E_{XY} I(Y \neq G(X))$$

The idea behind boosting is to apply many weak classifiers sequentially to successively modified versions of the data, producing a sequence of weak classifiers $G_m(x)$, $m = 1, 2, \ldots, M$. We combine all of these predictions using a



Algorithm 10.1 *AdaBoost.M1.*

```
1. Initialize the observation weights $w_i = 1/N,\ i = 1, 2, \ldots, N$.
2. For $m = 1$ to $M$:
    (a) Fit a classifier $G_m(x)$ to the training data using weights $w_i$.
    (b) Compute
        $$\text{err}_m = \frac{\sum_{i=1}^N w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^N w_i}.$$
    (c) Compute $\alpha_m = \log((1 - \text{err}_m)/\text{err}_m)$.
    (d) Set $w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))],\ i = 1, 2, \ldots, N$.
3. Output $G(x) = \text{sign}\left[\sum_{m=1}^M \alpha_m G_m(x)\right]$.
```

Figure 3: The ADABoost.M1 Algorithm

weighted majority vote:

$$G(x) = \text{sign}\left(\sum_{m=1}^{M} \alpha_m G_m(x)\right)$$

The weights $\alpha_1, \alpha_2, \ldots, \alpha_M$ are computed by the boosting algorithm (AdaBoost.M1).

In Fig. **??** we see the pipeline of the boosting meta-algorithm.[1] The training sample gets successively reweighted, producing one weak classifier at a time. The pipeline is:

1. Initially we set the weights on the training observations (x_i, y_i) all to $w_i = \frac{1}{N}$, meaning the first step trains the classifier in the usual fashion with equal weights on all data points.

2. On iterations $m = 2, 3, \ldots, M$, the observations' weights are individually modified and the classification algorithm is reapplied. The modification is done such that at step m, the observations that were misclassified by G_{m-1} get increased weights and the opposite for observations correctly classified. In this sense, we focus our effort on observations that failed previously.

3. As iterations proceed, difficult observations recieve ever-increasing influence such that each successive classification is forced to concentrate on training observations missed by previous ones in the sequence.

In Fig. **??**, step 2a we fit a new classifier based on the current weights. In step 2b we compute the normalized error rate of the classifier. In step 2c we calculate α_m and in step 2d we recalculate the weights. Finally, the algorithm outputs the sign of the sum of all classifier results.

[1] We call it a meta-algorithm since it operates on top of our choice of $G(\cdot)$, which is itself an algorithm.

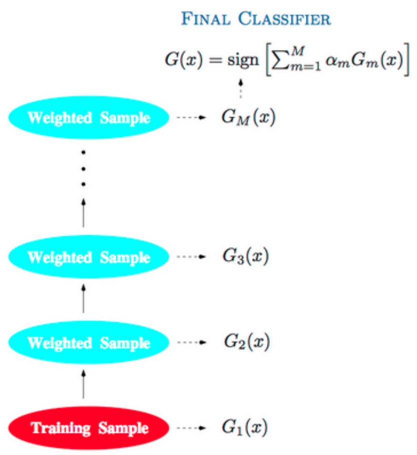


FIGURE 10.1. *Schematic of AdaBoost. Classifiers are trained on weighted versions of the dataset, and then combined to produce a final prediction.*

Figure 4:

3.1 Synthetic Example

To build intuition, consider the following synthetic example from HTF Sec. 10.1. Prepare a set of randomly sampled feature vectors $X \in \mathbb{R}^p$ with $X \sim \mathcal{N}(0, I)$ and $p = 10$. Define a deterministic target Y as follows:

$$Y = \begin{cases} 1 & \text{if} \quad \sum_{j=1}^{10} X_j^2 > \chi^2_{10}(0.5) \\ 0 & \qquad\qquad \text{otherwise} \end{cases}$$

Recall that the χ^2 random variable arises from the sum of standard normal random variables, the number of random variables in this sum is equal to the degrees of freedom. The value $\chi^2_{10}(0.5) = 9.34$ is the median of a χ^2 random variable with 10 degrees of freedom. For purposes of visualization, in the case of $p = 2$ the decision boundary would be a circle with the $Y = 1$ observations outside the circle and the $Y = 0$ observations inside.

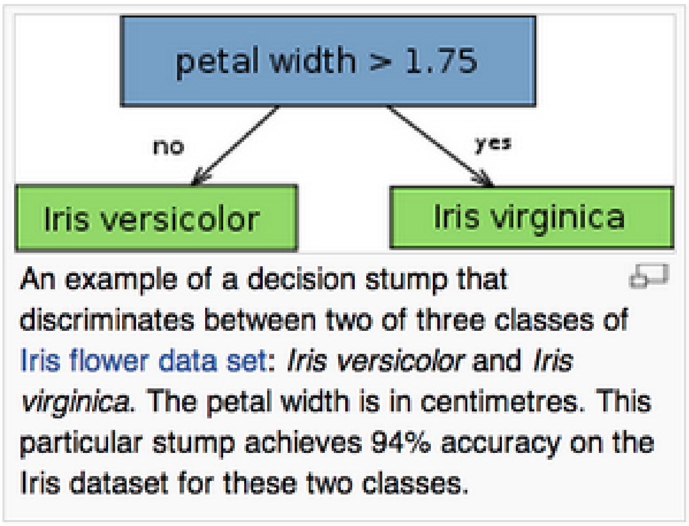
Assume there are 2000 training observations ($N = 2000$), with approximately 1000 observations in each class (evenly distributed), and 10,000 test observations. For the weak classifier we will use something called a *stump*.

Definition A stump is a special case of a decision tree that uses only one input dimension and only one split node (see Fig. **??**).

Recall that after we create the training data in this synthetic example, we pretend we know nothing about how it was made, we're just handed 2000 training observations in the form (x_i, y_i) with a 50/50 class split and we want to train a classifier on it. The stump we will use as a weak classifier has the form

$$G(x|j, \theta) = \begin{cases} 1 & \text{if} \quad x_j \leq \theta \\ -1 & \text{otherwise} \end{cases}$$

The parameters are j and θ. (We are free to flip the sign or use $x_j \geq \theta$.) Training the stump just amounts to stepping through $j = 1, \ldots, p$ and empirically



An example of a decision stump that discriminates between two of three classes of Iris flower data set: *Iris versicolor* and *Iris virginica*. The petal width is in centimetres. This particular stump achieves 94% accuracy on the Iris dataset for these two classes.

Figure 5: [Wikipedia]

choosing the best threshold θ based on the training error. Unsurprisingly, if we fit one stump to the training data we'll get an error rate close to chance (50%). In the next lecture we'll see how much the performance improves as we add more stumps using AdaBoost.