Discussion 3
Probabilistic Machine Learning, Spring 2018

1 Random Forests vs Tree Bagging

Bootstrap aggregation, also called bagging, is a technique for reducing the variance of an estimated prediction function. It works well for high-variance, low-bias prediction functions, like trees. The general procedure of bagging is to estimate the prediction function on M bootstrapped samples of the training data and then compile the M prediction functions into a single prediction (for example by taking the majority vote in a classification setting). We will compare bagging applied to decision trees ("tree bagging") and random forests. Let M be the number of trees, p be the number of features, and $K \leq p$ be a number of randomly selected features.

(a) True/False: When taking a bootstrap sample, we sample *with* replacement.

Solution: True.

(b) True/False: In tree bagging, we grow each tree on $K \leq p$ features that are randomly selected once before growing each tree.

Solution: False. In tree bagging we fix $K = p$.

(c) True/False: In random forests, we grow each tree on $K \leq p$ features that are randomly selected once before growing each tree.

Solution: False. We randomly choose $K \leq p$ features *separately for each split.*

(d) The variance of the average of M identically distributed random variables with variance σ^2 and positive pairwise correlation ρ is given by:

$$\rho\sigma^2 + \frac{1-\rho}{M}\sigma^2. \tag{1}$$

Suppose the M random variables represent decision trees. What happens as $M \to \infty$? What does this say about the advantages of random forests versus tree bagging?

Solution: As $M \to \infty$ the variance of the average of the M trees will approach $\rho\sigma^2$. The variance of the average of M bagged trees is limited by the first term, $\rho\sigma^2$. The idea of random forests is to improve upon tree bagging by constructing de-correlated trees (*i.e.*, reducing ρ).

(e) True/False: In random forests, as K increases we should expect the correlation between trees to increase (you can answer based on intuition, you do not need to prove anything).

Solution: True. As K increases, on average the trees will be more similar and so their correlation will increase. Note that because of the random selection of features and random selection (with replacement) of observations, we can only answer this question in expectation.

2 Boosting

A strong classifier is one that has an error rate close to zero. A weak classifier is one that has an error rate just below $\frac{1}{2}$, producing answers just a little better than a random guessing. Freund and Schapire discovered that you can construct a strong classifier from weak classifiers such that the strong classifier will correctly classify all samples in a sample set:

$$H(x) = sign \sum_t \alpha_t h_t(x)$$

At the t^{th} Adaboost step, you find the weak classifier, $h_t(x)$, that produces the lowest error rate with the samples reweighted to emphasize previously misclassified samples. Then you find the corresponding multiplier, α_t. You continue taking steps until the classifier $H(x)$ correctly classifies all samples or you cannot find any weak classifier that decreases error at the next step.

Algorithm 1 AdaBoost

```
Given training data $D = \{(x_i, y_i)\}_{i=1}^n$ and maximum number of iterations $T$
Initialize weights: $d_{1,i} = \frac{1}{n}$
for t=1,...,T do
    train a weak classifier: $h_t = \arg\min_h \sum_{i=1}^n d_{t,i} \times \mathbb{1}_{[h(x_i) \neq y_i]}$
    compute its weighted error: $\epsilon_t = \sum_{i=1}^n d_{t,i} \times \mathbb{1}_{[h_t(x_i) \neq y_i]}$
    compute coefficient: $\alpha_t = \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$
    update weights: $d_{t+1,i} = \begin{cases} \frac{d_{t,i} \times e^{-\alpha_t}}{Z_t}, & \text{if } x_i \text{ is correctly classified}, y_i = h_t(x_i) \\ \frac{d_{t,i} \times e^{\alpha_t}}{Z_t}, & \text{if } x_i \text{ is missclassified}, y_i \neq h_t(x_i) \end{cases}$
    where $Z_t$ is normalization constant for the discrete distribution, $Z_t = \sum_{i=1}^n d_{t+1,i} = 1$
end for
```

1. Prove that weights of AdaBoost given can be calculated as:

$$d_{t+1,i} = \begin{cases} \frac{1}{2} \frac{d_{t,i}}{1-\epsilon_t}, & \text{if } x_i \text{ is correctly classified}, y_i = h_t(x_i) \\ \frac{1}{2} \frac{d_{t,i}}{\epsilon_t}, & \text{if } x_i \text{ is missclassified}, y_i \neq h_t(x_i) \end{cases}$$

Solution. For the coefficient we have $\alpha_t = \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t} = \ln \sqrt{\frac{1-\epsilon_t}{\epsilon_t}}$. Then weight can be updates as:

$$d_{t+1,i} = \begin{cases} \frac{d_{t,i}}{Z_t} \sqrt{\frac{\epsilon_t}{1-\epsilon_t}}, & \text{if } x_i \text{ is correctly classified}, y_i = h_t(x_i) \\ \frac{d_{t,i}}{Z_t} \sqrt{\frac{1-\epsilon_t}{\epsilon_t}}, & \text{if } x_i \text{ is missclassified}, y_i \neq h_t(x_i) \end{cases}$$

where

$$Z_t = \sqrt{\frac{1-\epsilon_t}{\epsilon_t}} \sum_{y_i \neq h_t(x_i)} d_{t,i} + \sqrt{\frac{\epsilon_t}{1-\epsilon_t}} \sum_{y_i = h_t(x_i)} d_{t,i}$$

Sum of the weights over all misclassified points is the error rate $\sum_{y_i \neq h_t(x_i)} d_{t,i} = \epsilon_t$. Then sum of the rest of the weights is $1 - \epsilon_t$. Therefore we have that

$$Z_t = 2\sqrt{\epsilon_t(1-\epsilon_t)}$$

And the weights update procedure simplifies to:

$$d_{t+1,i} = \begin{cases} \frac{d_{t,i}}{2\sqrt{\epsilon_t(1-\epsilon_t)}}\sqrt{\frac{\epsilon_t}{1-\epsilon_t}} = \frac{1}{2}\frac{d_{t,i}}{1-\epsilon_t}, & \text{if } x_i \text{ is correctly classified}, y_i = h_t(x_i) \\ \frac{d_{t,i}}{2\sqrt{\epsilon_t(1-\epsilon_t)}}\sqrt{\frac{1-\epsilon_t}{\epsilon_t}} = \frac{1}{2}\frac{d_{t,i}}{\epsilon_t}, & \text{if } x_i \text{ is missclassified}, y_i \neq h_t(x_i) \end{cases}$$

2. Consider a training dataset described on Figure 1. We want to use AdaBoost to construct a classifier, where weak classifiers are decision stumps of the form:

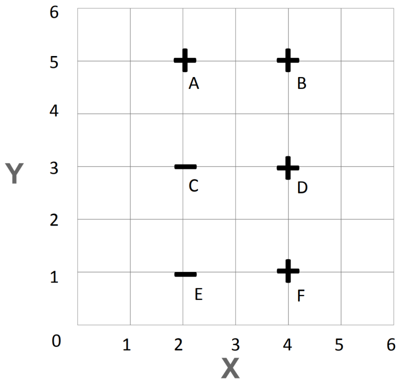

Figure 1: AdaBooost training dataset.

$$h(x,y) = \begin{cases} +1, & \text{if } x \geq T \\ -1, & \text{if } x < T \end{cases}; \quad h(x,y) = \begin{cases} +1, & \text{if } y \geq T \\ -1, & \text{if } y < T \end{cases}$$

Let's consider following list of classifiers: $X \geq 3$; $X \geq 5$; $Y \geq 4$; $Y \geq 6$.
NOTE: In a binary classification problem, usually a classifier does not misclassify over 50 percent of the data, since we can always flip the sign. For this problem, we allow the classifiers to make more than 50 percent of mistakes.

(a) List all the training points (A, B, C, D, E, F) that each classifier misclassifies.
Solution.

Classifier	Classifier	Misclassified Training Points
h_1	$X \geq 3$	A
h_2	$X \geq 5$	A, B, D, F
h_3	$Y \geq 4$	D, F
h_4	$Y \geq 6$	A, B, D, F

(b) Perform three iterations of boosting using the training points and the four classifiers. In case of a tie, use whichever classifier comes first in the list.

Solution.

		Iteration 1	Iteration 2	Iteration 3
weight A	d_A	1/6	5/10	5/16
weight B	d_B	1/6	1/10	1/16
weight C	d_C	1/6	1/10	1/16
weight D	d_D	1/6	1/10	4/16
weight E	d_E	1/6	1/10	1/16
weight F	d_E	1/6	1/10	4/16
Error rate h_1	ϵ^1	1/6	5/10	5/16
Error rate h_2	ϵ^2	4/6	8/10	14/16
Error rate h_3	ϵ^3	2/6	2/10	8/16
Error rate h_4	ϵ^4	4/6	8/10	14/16
Min error	ϵ	1/6	2/10	5/16
weak classifier	h	$X \geq 3$	$Y \geq 4$	$X \geq 3$
misclassified points		A	D, F	A
coefficient	α	$1/2 \ln 5$	$1/2 \ln 4$	$1/2 \ln 11/5$

(c) Consider the total classifier H which you produce after three rounds of boosting. How would $H(x)$ classify following points?

p	$signH(p)$
$(0, 0)$	-1
$(0, 7)$	-1
$(7, 7)$	1

3. Consider the following 2-class binary problem in Fig. 2. Using the update rules of the discrete AdaBoost, answer the following question

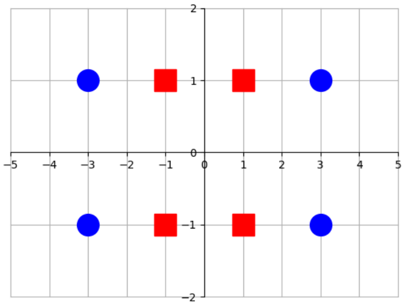

Figure 2: Toy data.

(a) What are the first 2 decision stumps, and what are their corresponding thresholds?

Solution First decision stump: $\phi_1(x) = 1, \text{if } x > -2, \quad \text{otherwise} - 1$

Second decision stump: $\phi_2(x) = 1, \text{if } x < 2, \quad \text{otherwise} - 1$

(b) Are these sufficient to obtain perfect classification?

Solution Not with Discrete AdaBoost. The final vote is obtained by linear combination of the weak learners. Regardless of the choice of weights assigned to each learner, the vote for the negative (black) samples on one of the two sides will be positive (the classifier function will be equal to 0 if all the weak learner weights are equal for instance). You need at minimum 3 weak learners to correctly classify this problem so that the third weight can play the role of a bias (a decision stump with all the points on one side of it, with a weight equal to the bias).

4. AdaBoost can be used in two ways. The first way is to combine a set of weak classifiers (such as stumps) where the set is determined before we run AdaBoost. The second way to run AdaBoost is as a meta algorithm on top of a weak learning algorithm like CART or C4.5. In that case, we can never actually enumerate all the weak classifiers. Look through all the steps of the AdaBoost algorithm and show that no step in the algorithm requires us to actually enumerate the set of weak classifiers. This means we never need to evaluate the matrix of margins $\mathbf{M}$. You will need to know that we do not actually need to obtain the best possible weak classifier (the argmax) in each round of AdaBoost in practice. It is sufficient to get a good weak classifier.

   Solution: For a data point x, we have

$$
\begin{aligned}
f(x) &= \sum_{j=1}^{n} \lambda_{j,t} h_j(x) && (2)\\
&= \sum_{j=1}^{n} \left(\sum_{t=1}^{T} \alpha_t \mathbf{1}_{[j_t=j]} \right) h_j(x) = \sum_{t=1}^{T} \alpha_t \left(\sum_{j=1}^{n} \mathbf{1}_{[j_t=j]} h_j(x) \right) && (3)\\
&= \sum_{t=1}^{T} \alpha_t h_{j_t}(x) && (4)
\end{aligned}
$$

   We can calculate $f(x)$ based on either Eq. 2 or Eq. 4. If Eq. 2 is chosen, we have to compute $\mathbf{M}$. In practice, we can alternatively choose Eq. 4 to obtain $f(x)$, where we avoid computing $\mathbf{M}$.

5. Assume the weak learning assumption holds. Asymptotically, as AdaBoost iterates over rounds, it is possible to determine what values of $f(\mathbf{x}_i)$ it will converge to?

   Solution: Note that because of the weak learning assumption, the data is separable, and α_t is always bounded below by $\alpha_{\min}$, where $\alpha_{\min}$ is a function of the minimum edge $\gamma_{\min}$ that comes from the weak learning assumption. This means α_t is always fairly large. This means some of the λ's grow by at least a constant at every iteration.

   $f(x_i)$ is a sum over λ's of the h_j's, which take on values -1 or 1. This means that $f(x_i)$'s get larger and larger, and since the data are correctly classified, $f(x_i)$ for the positive examples goes to infinity and $f(x_i)$ for the negative examples goes to negative infinity.

   A simple proof can largely depend on the Theorem that *the training error of Adaboost decays exponentially fast.* For the weak learning assumption, the each classifier performs better than "random guessing": $\epsilon_t = \frac{1}{2} - \gamma_t$, with $\gamma_t > \gamma_{\text{WLA}}$, the Theorem shows that:

$$
LH \triangleq R_{\text{train}}(\lambda_T) \leq \exp(-2\sum_{t=1}^{T} \gamma_t^2) \leq \exp(-2\sum_{t=1}^{T} \gamma_{\text{WLA}}^2 T) \triangleq RH \ ,
$$

   where $R_{\text{train}}(\lambda_T) = \frac{1}{m}\sum_{i=1}^{m} \exp(-(\mathbf{M}\lambda_T)_i)$, and $(\mathbf{M}\lambda_t)_i = y_i f(x_i)$.

   When $T \to +\infty$, we have $RH \to 0$, therefore $LH \leq 0$. Note that every term in LH is nonnegative, we need $\exp(-(\mathbf{M}\lambda_t)_i) \to 0$ or $y_i f(x_i) \to +\infty$ to satisfy this. This means, for x_i with label $y_i = 1$, its $f(x_i) \to +\infty$, and for x_i with label $y_i = -1$, its $f(x_i) \to -\infty$.