Ensemble Learning

Introduction

Ensemble learning involves the combining of mulitple learning models into a single model that has higher accuracy than any of its constituent models. Often times each constituent is considered as a **weak learner**, meaning that its accuracy is only slighly better than some minimum requirement (such as 50%). Ensemble learning has become an important tool for improving the overall accuracy of learning models, especially in machine-learning competitions, such as KDD Cup, and Kaggle, where contestants look for any means to boost their accuracy scores. Ensemble learning becomes an attractive option, because it may be used in conjunction with constituent learning models that may already exhibit high degrees of accuracy that can be further improved with ensemble methods. Perhaps the most famous example of is Belkor's Pragmatic Chaos winning of the famous Netflix prize in 2009. In addition to winning competitions, ensemble learning has been shown to obtains improvements some important learning tasks, such as object and face recognition, object tracking, lung-cancer cell identification, Alzheimer's diagnosis from EEG readings, and detection of computer viruses.

This lecture introduces perhaps the two most important ensemble-learning algorithms: Adaboost, and Bagging.

The Exponential Loss Function

Let X be a data universe and $f : X \to \{-1, +1\}$ be a target function. Let P denote a probability distribution over X, where $P(x)$ represents the likelihood of witnessing data point x when a data point is randomly selected from the universe. Consider a learned classifier $h : X \to \{-1, +1\}$. In this section we use the **exponential loss function**

$$L(f(x), h(x)) = \exp(-f(x)h(x)).$$

Thus, the expected risk for $h(x)$ is

$$R(h) = E[\exp(-f(X)h(X))],$$

where the expectation is computed using distribution $P(x)$. We now show that minimizing this loss function over possible classifiers h yields the best possible classifier

For the moment, assume that $f(x)$ represents a probability distribution over $\{-1, +1\}$. Now, for given x, we compute the value $h = h(x)$ for which $E[L(f(x), h)]$ is minimized, where now the expectation is with respect to $f(x)$. In other words, we want to minimize

$$e^{-h}P(f(x) = 1|x) + e^{h}P(f(x) = -1|x).$$

Differentiating the above expression with respect to h and setting the resullt to zero yields the equation

$$-e^{-h}P(f(x) = 1|x) + e^{h}P(f(x) = -1|x) = 0.$$

Soving this equation for h yields

$$h = \frac{1}{2}\ln\left(\frac{P(f(x) = 1|x)}{P(f(x) = -1|x)}\right).$$

Thus, the exponential loss is minimized by an $h(x)$ for which the sign of $h(x)$ is positive (i.e. choose class $+1$) iff $P(f(x) = 1|x) > P(f(x) = -1|x)$. Moreover, the classification error of h is equal to

$$\int \min(P(f(x) = 1|x), P(f(x) = -1|x))dP(x),$$

called the **Bayes error**, and is the minimum possible error that a classifier can achieve (why?). Therefore, choosing a classifier based on minimizing exponential loss leads to an optmal classifier. So why not just develop a single classifier based on minimizing this loss? Because we usually do not know the underlying distribution $P(x)$. For this reason we need learning algorithms that hopefully get close to the Bayesian error.

Adaboost Algorithm

The Adaboost algorithm involves developing a sequence $h_1, h_2, \ldots, h_T$ of learning models, called **base learners**, and weights $\alpha_1, \alpha_2, \ldots, \alpha_T$, that producte the ensemble classifier

$$H(x) = \text{sign}(\sum_{i=1}^{T} \alpha_i h_i(x)),$$

where the sign function returns -1 or $+1$, depending on the sign of $H(x)$.

To understand the main ideas behind the algorithm, suppose we have an algorithm A for constructing a base learning model h over a training data set $D = \{(x_1, y_1) \ldots, (x_l, y_l)\}$. We only require that h has a training error of less than 0.5.

For the base case, we assume initial ensemble classifier $H_0(x) = 0$, and uniform probability distribution $P_0(x) = \frac{1}{l}$ over the training data. Notice that all training points are equally misclassified by

H_0, and thus, assuming $P_0(x)$ should favor points misclassified by H_0, the uniform distribution is the most appropriate choice for P_0.

For the inductive step, suppose ensemble classifier

$$H_{t-1}(x) = \text{sign}(\sum_{i=1}^{t-1} \alpha_i h_i(x))$$

has been defined, along with a finite probability distribution $P_{t-1}(x)$ over the training data. Moreover, assume that $P_{t-1}(x)$ weights more heavily those points misclassified by H_{t-1}. Now suppose that on inputs D and P_{t-1}, A returns classifier h_t, with

$$\epsilon_t = \sum_{i=1}^{l} P_{t-1}(x_i) I(y_i \neq h_t(x_i)),$$

where $I(y_i \neq h_t(x_i)) = 1$ if h_t misclassifies x_i, and 0 otherwise.

The next step is to compute α_t. Since h_t is already computed, and P_{t-1} is already defined, we have enough information to make use of the exponential loss function in order to choose α_t. Indeed, we know $f(x_i) = y_i$ for each training point x_i, and can compute $h_1(x_i)$. The only unknown is α_t. Now, to minimize expected exponential loss, we must minimize

$$\sum_{i=1}^{l} \exp(-f(x_i)\alpha_t h_t(x_i)) P_{t-1}(x_i) = \sum_{i=1}^{l} \exp(-\alpha_t y_i h_t(x_i)) P_{t-1}(x_i) = e^{-\alpha_t}(1-\epsilon_t) + e^{\alpha_t}\epsilon_t.$$

Notice that this expression is identical in form to the one minimized in the previous section, and will be minimized with

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right).$$

Thus, the new classifier is now

$$H_t(x) = \text{sign}(\sum_{i=1}^{t} \alpha_i h_i(x)).$$

The final step of the algorithm is to compute a new finite distribution $P_t(x)$ over the training set. This is accomplished by looking ahead to computing h_{t+1}. Ideally, h_{t+1} should be chosen so as to minimize the expected exponential loss of $H_t + h_{t+1}$, where the expectation is with respect to $P_{t-1}(x)$. In other words, it should be chosen to minimize

$$\sum_{i=1}^{l} \exp(-y_i(H_t(x_i) + h_{t+1}(x_i)) P_{t-1}(x_i) = \sum_{i=1}^{l} \exp(-y_i H_t(x_i) - y_i h_{t+1}(x_i)) P_{t-1}(x_i).$$

Using the 2nd-order Taylor expansion

$$e^{-x} \approx 1 - x + \frac{x^2}{2},$$

the exprected loss may be approximated as

$$\sum_{i=1}^{l} \exp(-y_i H_t(x_i))(1 - y_i h_{t+1}(x_i)) + \frac{y_i^2 h_{t+1}^2(x_i)}{2}) P_{t-1}(x_i) =$$

$$\sum_{i=1}^{l} \exp(-y_i H_t(x_i))(1 - y_i h_{t+1}(x_i)) + \frac{1}{2})P_{t-1}(x_i),$$

since $y_i^2 = h_{t+1}^2(x_i) = 1$, for all $i = 1, \ldots, l$. Thus, minimizing the expected loss reduces to maximizing

$$\sum_{i=1}^{l} \exp(-y_i H_t(x_i))y_i h_{t+1}(x_i))P_{t-1}(x_i) = \sum_{i=1}^{l} \exp(-y_i H_t(x_i))P_{t-1}(x_i)y_i h_{t+1}(x_i)).$$

Now let $L_t(H_t)$ be defined as

$$L_t(H_t) = \sum_{i=1}^{l} \exp(-y_i H_t(x_i))P_{t-1}(x_i).$$

Notice that $L_t(H_t)$ is just the expected exponential loss of H_t with respect to distribution P_{t-1}. Moreover the finite distribution $P_t(x)$ having domain $\{x_1, \ldots, x_l\}$, and defined by

$$P_t(x_i) = \frac{1}{L_t(H_t)} \exp(-y_i H_t(x_i))P_{t-1}(x_i),$$

for all $i = 1, \ldots, l$ is indeed a probability distribution. Finally, maximizing the sum

$$\sum_{i=1}^{l} \exp(-y_i H_t(x_i))P_{t-1}(x_i)y_i h_{t+1}(x_i)$$

is equivalent to minimizing

$$\sum_{i=1}^{l} P_t(x_i) I(y_i \neq h_{t+1}(x_i)).$$

Therefore, h_{t+1} should be computed so as to achieve the Bayesian error rate with respect to distribution $P_t(x)$. But since the goal of learning algorithm A is to do just this (or at least to do its best at achieving the Bayesian error rate) on inputs D and $P_t(x)$, it follows that $P_t(x)$ is the next desired distribution. To summarize, we have

$$P_t(x_i) = (\exp(-y_i H_t(x_i))P_{t-1}(x_i))/L_t(H_t),$$

assigning higher weight to points that are misclassified by H_t.

Example 1. Recall that a decision stump is a decision tree with a single query node $(x \geq a)$?. If a is limited to either 0.5 or -0.5, the two classes are $\{-1, +1\}$, and the two variables are x and y, then how many different decision stumps can be defined? Apply AdaBoost using this set of decision stumps as the pool of base learners, and a learning algorithm A that, on input D and P, selects the base learner that minimizes the expected training error. Assume D is the XOR data set $\{(0, 1, +1), (1, 0, +1), (0, 0, -1), (1, 1, -1)\}$.

Theoretical Results for AdaBoost

Theorem 1 (Freund and Schapire 1997). If ϵ is the training error of the final ensemble learner $H(x)$, then

$$\epsilon \leq \exp(-2\sum_{i=1}^{T}(1-\epsilon_i)^2).$$

Freund and Schapire also proved that ensemble leaarners have a VC-confidence terms of $\hat{\mathrm{O}}(\sqrt{\frac{dT}{l}})$, where d is the VC-dimension of the base-learner family, T is the number of base learners comprising H, and l is the amount of training data.

Since each round of the Adaboost algorithm targets data points that are misclassified in previous rounds, one might suspect that AdaBoost is prone to overfitting. However, empirical studies have suggested otherwise. One theory for why Adaboost seems more resistant to overfitting involves the concept of classifier margin. Given ensemble classiifer H, the **margin** of H on training instance (x, y) is defined as $yH(x)$. Then the following theorem holds.

Theorem 2 (Freund and Schapire 1998). Consider training data $D = \{(x_1, y_1) \ldots, (x_l, y_l)\}$, probability distribution P over $\{x_1, \ldots, x_l\}$, base learning algorithm A, and ensemble learner H constructed using D, P, and A. Given $\delta > 0$, and margin $\theta > 0$, then with probability at least $1 - \delta$, generalization error (i.e. expected risk of H) ϵ of $H(x)$ is bounded as

$$\epsilon \leq P(yH(x) \leq \theta) + \sqrt{\frac{d}{m\theta^2} + \ln\frac{1}{\delta}},$$

where d bounds the VC-dimension of all learning models that can be produced by A. Moreover, empirical studies have shown that an ensemble learner can improve on its margin even after 100% training accuracy has been achieved.

Bagging

Notice that the boosting approach to ensemble learning represents a *sequential* process in which the construction of the next base learner h_t depends on the learners constructed in steps $1, \ldots, t-1$. The term "bagging" was coined by L. Breiman, and refers to Bootstrap AGGregatING. Bagging represents a *parallel* approach in which the base learners are constructed "independently" over the same data set D. To accomplish this, for each $t = 1, \ldots, T$, a base learning algorithm A is applied using a bootstrap sample D_{bs} of D; i.e. D_{bs} is defined by randomly selecting $|D|$ elements of D *with* replacement. Thus, bagging tends to work well when there is more variation found in each of the base learners. For this reason, not all learning algorithms A are ideal for bagging. For example, linear classifiers tend to demonstrate less variance over bootstrap samples, and hence bagging will not be helpful in this case. On the other hand, unpruned decision trees tend to show more variance, and bagging has been shown to benefit these models. In fact, the **random forest** learning model is based on the use of bagging with decision trees.

When the learners are truly independent the following Chernoff bound can be used to demonstrate the power of aggregating decisions.

Theorem (Chernoff Bound). Let random variable X_i, $1 \leq i \leq n$, be the i th outcome (1=success, 0=fail) of a Binomial process $B(n, p)$. Let $\hat{p} = \frac{1}{n}\sum_{i=1}^{n} X_i$ denote the empirical probability of a successful experiment. Then

$$Pr(|p - \hat{p}| > \epsilon) < 2e^{-2\epsilon^2 n}.$$

Example 2. Suppose $h_1, \ldots, h_{99}$ are independent binary ± 1-classifiers. Suppose ensemble learner $H(x)$ classifies x based on the sign of $h_1(x) + \cdots + h_{99}(x)$. If each base learner has a generalization error that is bounded above by 0.40, then use the Chernoff bound to compute an upper bound for the generalization error of $H(x)$.

Information Theory

The previous section mentioned that the decision-tree learning model seems more appropriate as a base algorithm for bagging. In this section we review the ID3 learning algorithm for constructing a decision tree. The ID3 algorithm is a greedy algorithm that chooses the next query node based on one that maximizes the average gain in information obtained from the answer to the query. Here, "information" is based on the following definition provided by Claude Shannon in the 1930's.

Entropy of a random variable. Let X be a random variable that takes on values from its **domain** $\{x_1, x_2, \ldots, x_n\}$ with respective probabilities $p_1, p_2, \ldots, p_n$, where $\sum_i p_i = 1$. Then the **entropy** of X, $H(X)$, represents the average amount of information contained in X and is defined by

$$H(X) = -\sum_{i=1}^{n} p_i \log_2 p_i = \sum_{i=1}^{n} p_i \log_2 \frac{1}{p_i}.$$

Note that entropy is measured in **bits**. Also, $H_n(p_1, \ldots, p_n)$ denotes another way of writing the entropy function.

The following are some arguments that justify the above definition of information

- The definition is consistent with the following extreme cases:
  1. If $n = 2$ and $p_1 = p_2 = \frac{1}{2}$, then $H(X) = 1$ bit; i.e. when an event (e.g. $X = x_1$) has an equally likely chance of occurring or not occurring, then its outcome possesses one bit of information. This is the maximum amount of information a binary outcome may possess.
  2. In the case when $p_i = 1$ for some $1 \leq i \leq n$, then $H(X) = 0$; i.e. any random variable whose outcome is certain possesses no information.
- The definition is the *only* one that satisfies the following three properties of information that seem desirable under any definition:
  - **Normalization:** $H_2(\frac{1}{2}, \frac{1}{2}) = 1$
  - **Continuity:** $H_2(p, 1-p)$ is a continuous function of p on the interval $(0, 1)$
  - **Grouping:**

$$H_m(p_1, \ldots, p_m) = H_{m-1}(p_1 + p_2, p_3, \ldots, p_m) + (p_1 + p_2) H_2\left(\frac{p_1}{p_1 + p_2}, \frac{p_2}{p_1 + p_2}\right)$$

Example 3. In a given region, each day there is one weather feature that is most salient for that day. The features are `mild`, `hot`, `rain`, `morning fog`, and `strong winds`. These features have the respective probabilities 1/2, 1/4, 1/8, 1/16, 1/16. On average how much information is conveyed by a weather report that announces the most salient feature?

Given two random variables X and Y, the **joint entropy** $H(X,Y)$ measures the average amount of information conveyed when jointly observing outcomes for both variables, and is defined as

$$H(X,Y) = \sum_{(x,y)\in X\times Y} p(x,y)\log\frac{1}{p(x,y)}.$$

In other words, $H(X,Y)$ equals the entropy of the random vector $Z=(X,Y)$.

On the other hand, $H(X|Y)$ measures the average amount of information that can be conveyed by X on condition that Y has been observed, and is defined as

$$H(X|Y) = \sum_y p(y)(\sum_x p(x|y)\log(\frac{1}{p(x|y)})).$$

Example 4. Compute $H(X,Y)$ and $H(X|Y)$ given the following probability table.

Y/X	1	2	3	4
1	$\frac{1}{8}$	$\frac{1}{16}$	$\frac{1}{32}$	$\frac{1}{32}$
2	$\frac{1}{16}$	$\frac{1}{8}$	$\frac{1}{32}$	$\frac{1}{32}$
3	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$
4	$\frac{1}{4}$	0	0	0

The ID3 Algorithm for Constructing Decision Trees

The concept of entropy may be applied to a training set D with class labels $\{1, \ldots, m\}$. Letting p_i, $i = 1, \ldots, m$, denote the fraction of training points labeled with i, then the **classification entropy** of D is defined as

$$H(D) = -\sum_{i=1}^{m} p_i \log_2 p_i.$$

We may interpret $H(D)$ as approximately the average amount of information that must be obtained in order to correctly classify a data point.

Example 5. Calculate the classification entropy for the following training set.

Weight	**color**	**texture**	**classification**
medium	orange	smooth	orange
heavy	green	smooth	melon
medium	green	smooth	apple
light	red	bumpy	berry
medium	orange	bumpy	orange
light	red	bumpy	berry
heavy	green	rough	melon
medium	red	smooth	apple
heavy	yellow	smooth	melon
medium	yellow	smooth	orange
medium	red	smooth	apple
medium	green	smooth	apple
medium	orange	rough	orange
light	orange	bumpy	berry
light	yellow	rough	berry
heavy	green	rough	melon

Now let A be an attribute/feature that is associated with each data point of D. For simplicity we assume that A has finite domain $\{a_1, \ldots, a_k\}$. Letting q_i, $i = 1, \ldots, k$, denote the fraction of training points whose A-attribute value equals a_i, then the **attribute entropy** of A is defined as

$$H(A) = -\sum_{i=1}^{k} q_i \log_2 q_i.$$

We may also define the conditional-entropy quantity

$$H(D|A) = \sum_{i=1}^{m} p_i \left(\sum_{j=1}^{k} r_{ji} \log \frac{1}{r_{ji}} \right),$$

where r_{ji} is the fraction of training points that have A-attribute value a_j, on condition that the training point is labeled with class i, for all $i = 1, \ldots, m$, and $j = 1, \ldots, k$.

Quinlan's **ID3 algorithm** is a greedy algorithm that views the decision tree as a means for obtaining the requisite (classification) information needed to classify a data point. For example, a query node n that queries about attribute A will obtain an average of $I(D, A) = H(D) - H(D|A)$ bits of information, where $H(D)$ and $H(D|A)$ are computed with respect to only those training points that are capable of reaching n. The quantity $I(D, A) = H(D) - H(D, A)$ is referred to as the **mutual information** shared by D and A, or the **information gain** with respect to querying the data set in regards to attribute A.

Algorithm ID3 is greedy in the sense that it chooses the query attribute A as the one that minimizes $H(D|A)$, i.e., the one that minimizes the remaining information in D after the value of A has been observed for the data point being classified. Alternatively, A is chosen so as to maximize the information gain $I(D, A)$.

Example 6. Apply the ID3 algorithm to the data set from Example 5.

For attributes that take on many values, using the **gain ratio** $\frac{I(D,A)}{H(A)}$ instead of $I(D,A)$ can help avoid favoring many-valued attributes which may not perform well in classifying the data.

Example 7. Suppose attribute A has 8 possible values. If selecting A for the next node of a decision tree yields $I(D,A) = 2$ bits of information, then compute $\frac{I(D,A)}{SI(D,A)}$.

Exercises

1. For each x in data universe X, suppose $f(x)$ denotes a target probability distribution over $\{-1,+1\}$. Prove that if the chosen loss function is $\ln(1+\exp(-2f(x)h(x)))$, then the classifier that minimizes this loss function achieves the optimal Bayes error rate.

2. Assume training set D consists of positive training vectors $\overline{x}_1 = (-1,0)$ and $\overline{x}_2 = (1,0)$, and negative training vectors $\overline{x}_3 = (0,-1)$ and $\overline{x}_3 = (0,1)$. If $P_{t-1} = (0.4, 0.2, 0.3, 0.1)$ is a probability distribution over D, and classifier h_t assigns the label sequence $+1,-1,-1,+1$ to vectors $\overline{x}_1, \overline{x}_2, \overline{x}_3, \overline{x}_4$, then compute i) ϵ_t, the empirical risk of h_t, using 0-1 loss function, and distribution P_{t-1}, and ii) the α_t coefficient associated with h_t in a learning ensemble.

3. For the previous exercise (and assuming h_t is the classifier selected in round t of Adaboost), suppose $H_{t-1}(\overline{x}_1) = -2.1$, $H_{t-1}(\overline{x}_2) = 0.1$, $H_{t-1}(\overline{x}_3) = 0.4$, and $H_{t-1}(\overline{x}_4) = -1.3$. Then determine P_t, the probability distribution over D for round $t+1$.

4. State the criterion that is used for selecting the α_t coefficient of h_t in round t of the Adaboost algorithm.

5. Verify that the 2nd order Taylor expansion of e^{-x} about $a = 0$ is $1 - x + \frac{x^2}{2}$.

6. State the *initial* criterion that is used for describing the ideal next classifier h_{t+1} that should be obtained in round $t+1$ of Adaboost. After an appropriate Taylor expansion, what other criterion does this become equivalent to?

7. Consider the following sets of training examples:

Instance	Classification	a_1	a_2
1	+	T	T
2	+	T	T
3	-	T	F
4	+	F	F
5	-	F	T
6	-	F	T

   Calculate the entropy of the collection with respect to the classification, and determine which of the two attributes provides the most information gain.

8. Given data set D and two attributes A_1 and A_2, suppose $H(D) = 5$ $H(D|A_1) = 3.9$, $H(D|A_2) = 2.5$, $H(A_1) = 0.5$, while $H(A_2) = 2.1$. Which attribute is preferred during query-node onstruction in the ID3 algorithm if the greedy choice maximizes i) information gain? ii) gain ratio?

9. Create a decision tree using the ID3 Algorithm for the following table of data.

Vector	A_1	A_2	A_3	**Class**
v_1	1	0	0	0
v_2	1	0	1	0
v_3	0	1	0	0
v_4	1	1	1	1
v_5	1	1	0	1