Bagging and Boosting Methods

- Motivation for combining learning machines

. Suppose you have many "easy rules": combining them may be a good idea.
. Parameter estimation: combine many machines with different parameters?
. Bootstrap: may helps with "variance"?

- Voting classification

. Methods for voting classification algorithms have been shown to be very successful in improving the accuracy.

. Voting algorithms can be divided into two types:

- change the distribution of the training set based on the performance of previous classifiers
  eg. Boosting.

- those that do not
  eg. Bagging.

- Strong and weak learning models

. Strong learning models:

have classification rate 1 - δ,

where δ is small positive number.

. Weak learning models:

have classification rate on slightly better than 1/2.

- Bagging methods

. Bagging = bootstrap agregation.

. Training data $Z = (x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$,

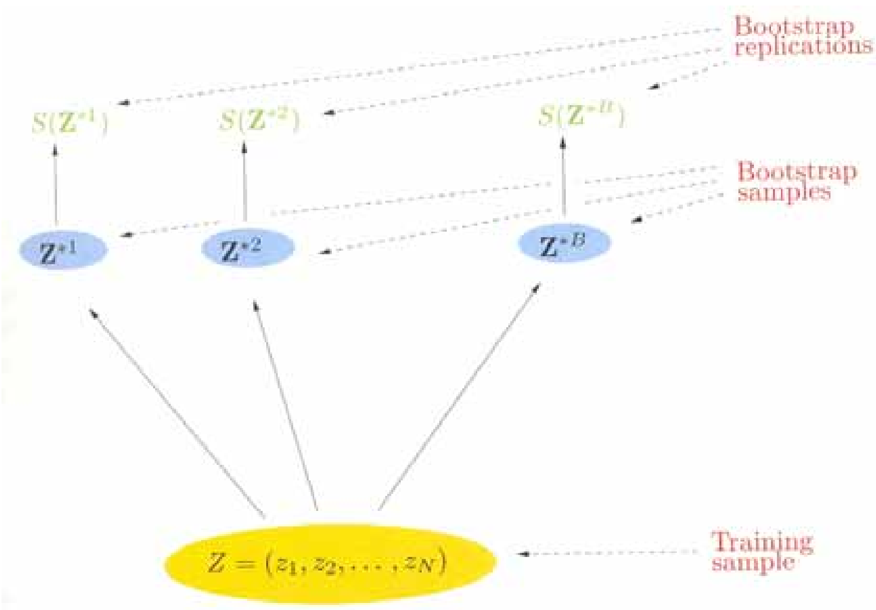
obtaining the prediction $\hat{f}(x)$ at input x.

. For each bootstrap sample Z^{*b}, $b = 1, 2, \ldots, B$.

fit a model, giving prediction $\widehat{f^{*b}}(x)$.

. The bagging estimate: $\widehat{f_{bag}}(x) = \frac{1}{B} \sum_{b=1}^{B} \widehat{f^{*b}}(x)$

. bootstrap process



. Bagging average this prediction over collection of bootstrap samples, thereby reducing its variance.

. Denote by $\hat{P}$ the empirical distribution putting equal probability $1/N$ on each of the data points (x_i, y_i).

. Let "true" bagging estimate: $E_{\hat{P}}\widehat{f^*}(x)$,

where $Z^* = (x_1^*, y_1^*), (x_2^*, y_2^*), \ldots, (x_N^*, y_N^*)$ and each $(x_i^*, y_i^*) \sim \hat{P}$.

. $\widehat{f_{bag}}(x)$ is a Monte Carlo estimate of the true Bagging estimate, approaching it as $B \rightarrow \infty$.

. If perturbing the learning set can cause significant change in the predictor constructed, then bagging can improve accuracy.

- Bagging (Bootstrap AGGregatING)

. Given a training set $D=\{(x_1,y_1),\cdots,(x_l,y_l)\}$,

> Sample N sets of l elements from D with replacement (bootstraping procedure), that is, $D_1,\cdots,D_N$ (N quasi replica training sets).

> Train a machine on each D_i, $i=1,\cdots,N$ and obtain a sequence of N outputs $f_1(x),\cdots,f_N(x)$.

> The final aggregate classifier can be

(1) for regression

$$\bar{f}(x)=E\{f_i(x)\},$$

that is, the average of f_i for $i=1,\cdots,N$.

(2) for classification

$$\bar{f}(x)=\theta(E\{f_i(x)\})$$

where θ represents the indicator function. In this case, $\bar{f}(x)$ will be the majority vote from $f_i(x)$.

- Bias and variance for regression

. Let

$$I[f] = \int (f(x) - y)^2 p(x,y)\,dx\,dy$$

be the expected risk and f_0 the regression function. With $\bar{f}(x) = E\{f_i(x)\}$, if we define the bias as

$$\int (f_0(x) - \bar{f}(x))^2 p(x)\,dx$$

and the variance as

$$E\left\{\int (f_i(x) - \bar{f}(x))^2 p(x)\,dx\right\},$$

we have the following decomposition:

$$E\{I[f_i]\} = I[f_0] + bias + variance.$$

- Bias and variance for classification

. No unique decomposition for classification exists.

In the binary case, with $\bar{f}(x) = \theta(E\{f_i(x)\})$, the decomposition suggetsed by Kong and Dietterich (1995) is

$$I[\bar{f}] - I[f_0]$$

for the bias, and

$$E\{I[f_i]\} - I[\bar{f}]$$

for the variance, which (again) gives

$$E\{I[f_i]\} = I[f_0] + bias + variance.$$

- Bagging reduces variance

. If each single classifier is unstable, that is, it has high variance, the aggregated classifier $\bar{f}$ has a smaller variance than a single original classifier.

. The aggregated classifier $\bar{f}$ can be thought of as an approximation to the true average f obtained by replacing the probability distribution p with the bootstrap approximation to p obtained concentrating mass $1/l$ at each point (x_i, y_i).

- Ensembles of kernel machines

. What happens when combining SVMs with kernels?

> different subsamples of training data (bagging)

> different kernels or different features

> different parameters, that is, regularization parameters

. Combination of SVMs

Let $f_1(x), \cdots, f_N(x)$ be SVM machines we want to combine and

$$f(x) = \sum_{i=1}^{N} c_n f_n(x)$$

for some fixed $c_n > 0$ with $\sum_n c_n = 1$.

- Leave-one-out error

. The leave-one-out error is computed in three steps

(1) Leave a training point out

(2) Train the remaining points and test the point left out

(3) Repeat for each training point and count "errors".

. Theorem (Luntz and Brailovski, 1969)

$$E\{I[f_l]\} = E\{CV\ error\ of\ f_{l+1}\}$$

where f_l represents the lth regression function.

. Leave-one-out bound for an SVM:

For SVM classification

$$\sum_{i=1}^{l} \theta(\alpha_i K(x_i, x_i) - y_i f(x_i)) \leqq \frac{r^2}{\rho^2}$$

where r is the radius of the smallest sphere containing the SVs and ρ is the true margin. (Jaakkola and Haussler, 1998)

. Leave-one-out bound for a kernel machine ensemble

The leave-one-out error of an SVM ensemble

$$f(x) = \sum_{i=1}^{N} c_i f_i(x)$$

is upper bounded by

$$\sum_{i=1}^{l}\theta(\sum_{n=1}^{N}(\alpha_i K^{(n)}(x_i,x_i)) - y_i f(x_i)) \leqq \sum_{i=1}^{N}\frac{r_{(n)}^2}{\rho_{(n)}^2}$$

where $r_{(n)}$ is the radius of the smallest sphere containing the SVs of machine n and $\rho_{(n)}$ the margin of SVM n.
This suggests that bagging SVMs can be a good idea!

. Trough a modified version of the notion of stability, it is possible to study conditions under which bagging should or shoud not improve performances. (Evgeniou et al, 2001)

- The original boosting (Schapire, 1990)

1. Train a first classifier f_1 on a training set drawn from a probability $p(x,y)$. Let ϵ_1 be the obtained performance.
2. Train a second classifier f_2 on a training set drawn from a probability $p_2(x,y)$ such that it has half its measure on the event that f_1 makes a mistake and half on the rest. Let ϵ_2 be the obtained performance.
3. Train a third classifier f_3 on disagreements of the first two, that is, drawn from a probability $p_3(x,y)$ which has its support on the event that f_1 and f_2 disagree. Let ϵ_3 be the obtained performance.

. Main result:

If $\epsilon_i < p$ for all i, the boosted hypothesis

$$f = Majority\,Vote(f_1, f_2, f_3)$$

has performance no worse than $\epsilon = 3p^2 - 2p^3$.
This implies that the boosting is effective when $p < 0.5$.

- Adaboost (Freund and Schapire, 1996)

The idea is adaptively resampling the data.

. Maintain a probability distribution over training set.
. Generate a sequence of classifier in which the next classifier focuses on sample where the previous classifier failed.
. Weigh machines according to their performance.
. Adaboost algorithm

Step 1. Initialize the distribution as $P_1(i) = 1/l$.

Step 2. For $i = 1, \cdots, N$ repeat the following procedure:

(1) Train a machine with weights $P_n(i)$ and get f_n.

(2) Compute the weighted error

$$\epsilon_n = \sum_{i=1}^{l} P_n(i)\theta(-y_i f_n(x_i)).$$

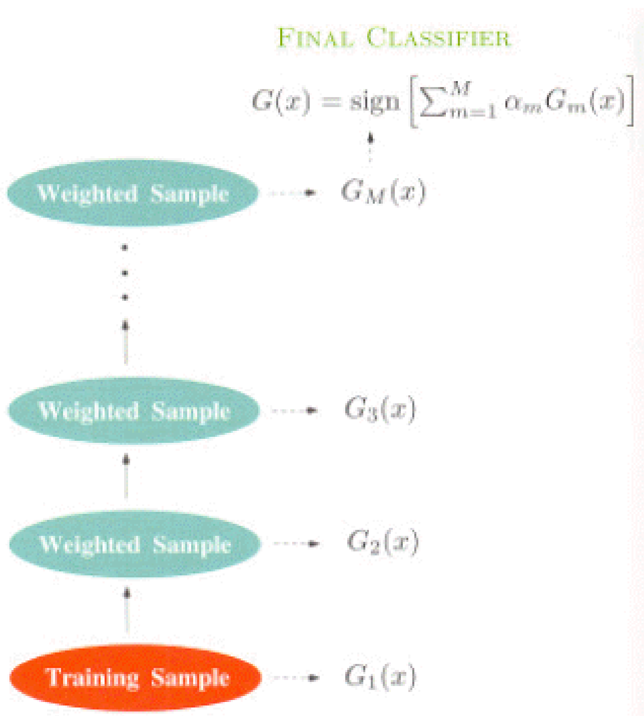
(3) Compute the importance of f_n as

$$\alpha_n = \frac{1}{2}\ln\left(\frac{1-\epsilon_n}{\epsilon_n}\right).$$

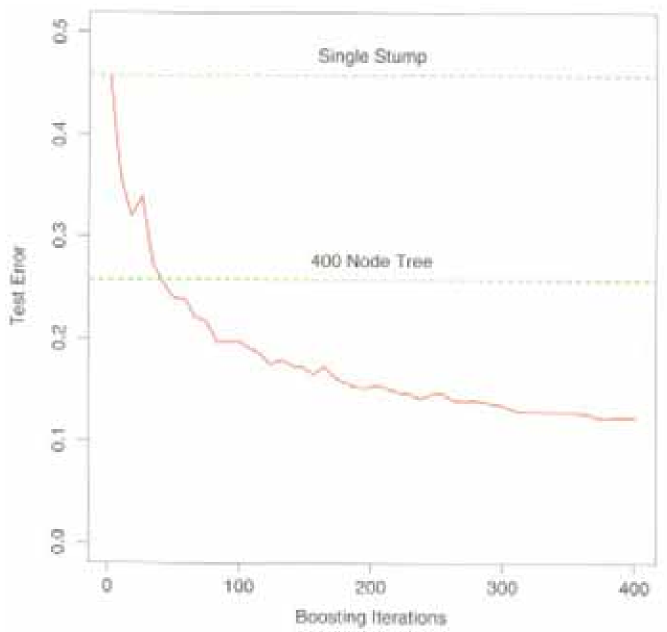
(4) Update the distribution $P_{n+1}(i) \propto P_n(i) e^{-\alpha_n y_i f_n(x_i)}$.

. The final hypothesis is given by

$$f(x) = sign\left(\sum_{n=1}^{N} \alpha_n f_n(x)\right).$$



- Example of Adaboost: decision tree learning



- Theory of boosting

. We define the margin of (x_i, y_i) according to the real-valued function f to be

$$margin(x_i, y_i) = y_i f(x_i).$$

Note that this notion of margin is different from the SVM margin. This defines a margin for each training sample.

- The first theorem on boosting

. Theorem (Schapire et al, 1997)

If running adaboost generates functions with errors

$\epsilon_1, \cdots, \epsilon_N$,

then $\forall \gamma$

$$\sum_{i=1}^{l} \theta(\gamma - y_i f(x_i)) \leqq \prod_{n=1}^{N} \sqrt{4\epsilon_n^{1-\gamma}(1-\epsilon_n)^{1+\gamma}}.$$

Thus, the running margin error drops exponentially fast if $\epsilon_n < 0.5$.

- The second theorem on boosting

. Theorem (Shapire et al, 1997)

Let H be an hypothesis space with VC-dimension d and C the convex hull of H, that is,

$$C = \left\{ f : f(x) = \sum_{h \in H} \alpha_h h(x) | \alpha_h \geqq 0, \sum_{h \in H} \alpha_h = 1 \right\}.$$

Then, $\forall f \in C$ and $\forall \gamma > 0$

$$I[f] \leqq \sum_{i=1}^{l} \theta(\gamma - y_i f(x_i)) + O\left(\frac{d/l}{\gamma}\right).$$

This holds for any voting method!

- Are these theorems really useful?

. The first theorem simply ensures that the training error goes to zero.

. The second theorem gives a loose bound which does not account for the success of boosting as a learning technique.

. More realistic bound accommodating the estimation function ensemble generated by boosting algorithm so that we can find the optimal boosting number N.

- Generalization error

. Let sample size m, the VC-dimension d of the weak hypothesis space and the number of boosting rounds T.

. The generalization error is at most

$$\hat{Pr}[H(x) \neq y] + \widetilde{O}(\sqrt{\frac{Td}{m}})$$

where $\hat{Pr}[\bullet]$ denotes empirical probability on the training sample.

. This bound suggests that boosting can have a over-fit for large T. In fact, over-fitting can happen in the boosting method.

. However, in general, over-fitting is not observed empirically even for large number of boosting rounds.

. Moreover, it was observed that AdaBoost would sometimes continue to drive down the generalization error long after the training error has reached zero, clearly contradicting the generalization bounds.

. Boosting is particularly aggressive at reducing the margin since it concentrates on the examples with the smallest margins.

- Generalization error with margin

. In response to theses empirical findings, gave an alternative analysis in terms of the margins of the training examples.

. The margin of example (x, y): $yf(x)$ or $y\sum_t \alpha_t h_t(x)$.

Margin is a number in [-1, +1].
Margin is positive $\Leftrightarrow$ H correctly classifies the example.

. The magnitude of the margin can be interpreted as a measure of confidence in the prediction.

. Larger margins on the training set translate into a superior upper bound on the generation error.

. The generation error is at most

$\hat{Pr}[margin(x,y) \leq \theta] + \widetilde{O}(\sqrt{\frac{d}{m\theta^2}})$ for any $\theta > 0$ with high probability.

. This bound is entirely independent of T, the number of boosting rounds. However, even in this case, the over-fitting in boosting can not be explained.

- Compare Bagging with Boosting

. Bagging

distribution :1/N.
always improve an learning system.
high computational complexity for learning.
unstable learning system → improve accuracy.

. Boosting

change the distribution.
medium computational complexity for learning.
in general, over-fitting does not occur.
sometimes over-fitting does occur.

- References.

. Breiman, L. “Bagging predictors.” Machine Learning 24, 123-140, 1996.

. Yoav Freund, "Boosting a weak learning algorithm by majority," Information and Computation, 121(2):256-285, 1995.

. Harris Drucker and Corinna Cortes, “Boosting decision trees;” In Advances in Neural Information Processing Systems 8, pages 479-485, 1996.

. Robert E. Schapire, Yoav Freund, Peter Bartlett, and Wee Sun Lee, "Boosting the margin: A new explanation for the effectiveness of voting methods," The annals of Statistics, 26(5):1651-1686, 1998.

. Ratsch, G., Onoda, T., and Muller, K. R. "Soft margins for adaboost," Machine Learning, 42, 287-320, 2000.

. Quinlan, J. R., "Bagging, boosting, and C4.5," in Proceedings of the Thirteenth Natural Conference on Artificial Intelligence, AAAI Press and MIT Press, pp. 725-730, 1996.

. Richard Maclin and David Opits, "An empirical evaluation of bagging and boosting," In Proceedings of the Fourteenth National Conference on Artificial Intelligence, pages 546-551, 1997.

. Maclin, R. and Opitz, D., "An empirical evaluation of bagging and boosting," In AAAI-97 (pp. 546-551), 1997.