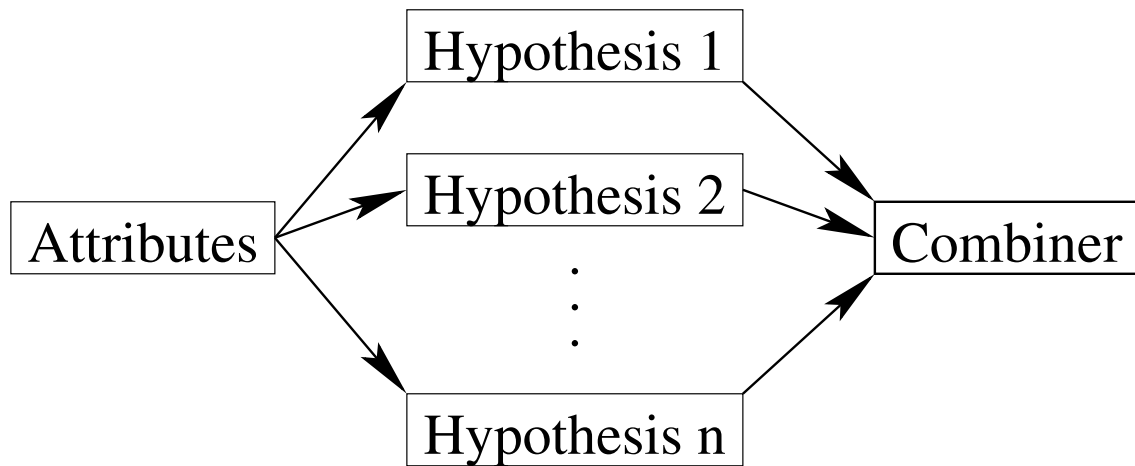


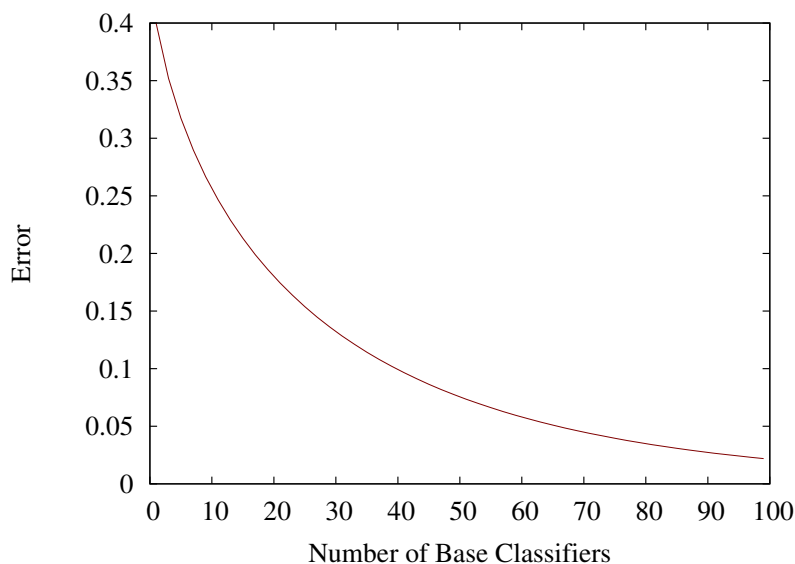
Ensemble Learning Algorithms

An ensemble learning algorithm learns a hypothesis by training a number of base hypotheses and combining their predictions.



Ensembles Can Decrease Error

Suppose each base classifier has a 40% chance of error on each example. More base classifiers make it more likely a majority will be correct.



Will Ensembles Decrease Error?

Assuming that base classifiers have independent errors is unrealistic.

However, for examples where base classifiers predict with $>50\%$ accuracy, ensembles can help.

Reasons against ensembles include:

- increase training and prediction complexity (but only linear with the number of base hypotheses)
- more difficult to explain prediction

Bagging

Bagging = Bootstrap AGGREGatING

B is the number of “bags” or base hypotheses

L is the base learning algorithm

Bagging(*examples*, B , L)

1. **for** $i \leftarrow 1$ to B
2. $examples_i \leftarrow$ a bootstrap sample of *examples*
3. $h_i \leftarrow$ apply L to $examples_i$
4. **return** h_1, h_2, \dots, h_B

Bagging Details

- A bootstrap sample is formed by sampling with replacement.
The size is the same as the number of exs.
Some examples are sampled more than once.
About $1/e \approx 0.368$ of the exs. are omitted.
 - Classification is by plurality vote.
Regression is by averaging.
 - The “out-of-bag” exs. can be used as a validation set or to estimate generalization error.
-

Why Bagging Works

Error in learning is due to noise, bias, and variance.

- Noise is error by the target function.
- Bias is where the algorithm cannot learn the target.
- Variance comes from sampling, and how it affects the algorithm.

Bootstrap sampling simulates sampling variance.

Averaging over bootstrap samples can reduce error from variance, especially when small differences in training sets can produce big differences between hypotheses.

Ensembles Reduce Variance

Let $f(\mathbf{x})$ be the target value for \mathbf{x} .

Let h_1, \dots, h_B be the base hypotheses.

Let h_{avg} be the average prediction of h_1, \dots, h_B .

Let $e(h, \mathbf{x}) = (f(\mathbf{x}) - h(\mathbf{x}))^2$

$$e(h_{avg}, \mathbf{x}) = \frac{\sum_{i=1}^B e(h_i, \mathbf{x})}{B} - \frac{\sum_{i=1}^B (h_i(\mathbf{x}) - h_{avg}(\mathbf{x}))^2}{B}$$

The squared error of the average prediction equals the average squared error of the base hypotheses minus the variance of the base hypotheses.

A. Krogh and J. Vedelsby (1995). Neural network ensembles, cross validation and active learning. In D. S. Touretzky G. Tesauro and T. K. Leen, eds., *Advances in Neural Information Processing Systems*, pp. 231-238, MIT Press.

Boosting

The original motivation for boosting was to convert a “weak” learning algorithm L (error close to but $<50\%$) into a “strong” one (error close to 0%).

Idea: Repeatedly run L , but focus more attention on misclassified examples.

Implementation:

- Provide a weight for each example, misclassified examples get higher weight.
- Each base classifier gets a weighted vote depending on its accuracy.
- L does not have to be “weak”.

Adaboost Algorithm

Adaboost = ADAptive BOOSTing

Adaboost(*examples*, L , T)

1. $D_1(i) \leftarrow 1/m$ for each of m *examples*
 2. **for** $t \leftarrow 1$ to T
 3. $h_t \leftarrow$ apply L to *examples* using distribution D_t
 4. $\epsilon_t \leftarrow$ sum of $D_t(i)$ for *examples* misclassified by h_t
 5. $\beta_t \leftarrow (1 - \epsilon_t)/\epsilon_t$
 6. $D_{t+1}(i) \leftarrow \begin{cases} D_t(i)\beta_t & \text{if } h_t \text{ misclassifies example } i \\ D_t(i) & \text{otherwise} \end{cases}$
 7. normalize D_{t+1} so it sums to 1
 8. **return** h_1, \dots, h_T and β_1, \dots, β_T
-

Adaboost Details

- D_t is a distribution, an assignment of a probability to each example.
- L should be sensitive to the distribution. If not, we can take a bootstrap sample using the distribution.
- If $\epsilon_t \geq .5$, then set T to $t - 1$ and break from loop. What if $\epsilon_t = 0$?
- After line 7, sum of D_{t+1} for misclassified exs. = sum for correctly classified examples = $1/2$. This forces h_{t+1} to be different from h_t .
- Classification is by plurality voting with base classifier h_t getting $\ln \beta_t$ votes.

What Adaboost Does

Let $w_t(i)$ be a weight for example i , where $w_1(i) = 1$ and:

$$w_{t+1}(i) = \begin{cases} w_t(i)\sqrt{\beta_t} & \text{if } h_t \text{ misclassified } i \\ w_t(i)/\sqrt{\beta_t} & \text{if } h_t \text{ correctly classifies } i \end{cases}$$

w_t is used to relate D_t to the voting.

Let $s_t = \sum_{i=1}^m w_t(i)$

s_t is minimized by Adaboost.

Let $r_t = 1/2 - \epsilon_t$

r_t measures the goodness of h_t . It is 0 if $\epsilon_t = 1/2$, and $1/2$ if $\epsilon_t = 0$.

What Adaboost Does

Theorem: $D_t(i) = w_t(i)/s_t$

Proof Sketch:

Base Case: $D_1(i) = 1/m = w_1(i)/s_1$

Inductive Case: Assume $D_t(i) = w_t(i)/s_t$

After line 6, whether h_t is correct or wrong:

$$D_{t+1}(i) = w_{t+1}(i)\sqrt{\beta_t}/s_t$$

so D_{t+1} and w_{t+1} differ by a factor $\sqrt{\beta_t}/s_t$.

Normalizing D_{t+1} (line 7) changes the factor to $1/s_{t+1}$.

What Adaboost Does (continued)

Theorem: If $w_{t+1}(i) < 1$, then h_1, \dots, h_t with β_1, \dots, β_t correctly classify example i .

Proof Sketch:

Let a = product of β_j where h_j is correct.

Let b = product of β_j where h_j is wrong.

Let c = sum of $\ln \beta_j$ where h_j is correct.

Let d = sum of $\ln \beta_j$ where h_j is wrong.

$w_{t+1}(i) = \sqrt{b/a}$, so $w_{t+1}(i) < 1$ implies $a > b$, which implies $c > d$, which implies a majority for the correct class.

Corollary: Number of mistakes on training set $\leq s_t$

What Adaboost Does (continued)

Theorem: $s_{t+1} = s_t \sqrt{1 - 4r_t^2} \leq s_t e^{-2r_t^2}$

Proof Sketch:

ϵ_t of s_t is multiplied by $\sqrt{\beta_t}$ and

$1 - \epsilon_t$ of s_t is divided by $\sqrt{\beta_t}$, so:

$$\begin{aligned} s_{t+1} &= \epsilon_t s_t \sqrt{\beta_t} + (1 - \epsilon_t) s_t / \sqrt{\beta_t} \\ &= \dots \\ &= s_t \sqrt{1 - 4r_t^2} \end{aligned}$$

Based on the inequality $1 + x \leq e^x$:

$$\sqrt{1 - 4r_t^2} \leq \sqrt{e^{-4r_t^2}} = e^{-2r_t^2}$$

What Adaboost Does (continued)

Theorem: $\sum_{t=1}^T 2r_t^2 > \ln m$ implies $s_{T+1} < 1$.

Proof Sketch: Note that $s_1 = m$.

$$\begin{aligned} s_{T+1} &\leq m \prod_{t=1}^T e^{-2r_t^2} \\ &= m \exp\left\{-\sum_{t=1}^T 2r_t^2\right\} \\ &< m \exp\{-\ln m\} = 1 \end{aligned}$$

With enough good hypotheses,
Adaboost will achieve zero training error.

It turns out increasing T usually does not increase overfitting.

Other Ensemble Algorithms: Stacking

M and L_1, \dots, L_N are learning algorithms.

Stacking($examples, M, L_1, \dots, L_N$)

1. $examples_M \leftarrow$ test outputs of L_1, \dots, L_N
on $examples$ using k -fold cross-validation
2. **for** $i \leftarrow 1$ to N
3. $h_i \leftarrow$ apply L_i to $examples$
4. $h_M \leftarrow$ apply M to $examples_M$
5. **return** h_M, h_1, \dots, h_N

M learns from outputs of base classifiers.

Outputs of h_1, \dots, h_N are input into h_M .

Other Ensemble Algorithms: Arcing

Arcing reweights the examples based on the number of mistakes. All base classifiers get one vote.

Arcing(*examples*, L , T)

1. $D_1(i) \leftarrow 1/m$ for each of m *examples*
2. $M_1(i) \leftarrow 0$ for each of m *examples*
3. **for** $t \leftarrow 1$ to T
4. $h_t \leftarrow$ apply L to *examples* using distribution D_t
5. $M_{t+1}(i) \leftarrow \begin{cases} 1 + M_t(i) & \text{if } h_t \text{ misclassifies example } i \\ M_t(i) & \text{otherwise} \end{cases}$
6. $D_{t+1}(i) \leftarrow 1 + (M_{t+1}(i))^4$ for all *examples*
7. normalize D_{t+1} so it sums to 1
8. **return** h_1, \dots, h_T