Bias/Variance Tradeoff

Preliminaries

$$\overline{A+B} = \overline{A} + \overline{B}$$

$$VAR(A) = \left\langle \left(A_i - \overline{A}\right)^2 \right\rangle_i$$

mean squared error to the mean, variation of samples about mean

If A and B independent (not correlated):

$$VAR(A+B) = VAR(A) + VAR(B)$$

Supervised Learning Notation

$x_i =$ d-dimensional vector of input attributes

$T(x) =$ true, unknown ideal function mapping inputs to targets

$t_i = T(x_i) =$ ideal (no noise) target for input vector x_i

$T'(x_i) = T(x_i) + noise^2$ (because god is perverse!)

$t_i' = T(x_i) + noise^2 =$ target with noise for input x_i

$D =$ sample of N points from distribution $p(t,x)$

$L(x,D) =$ function learned from train set D that predicts target $y_i = t_i' = L(x_i,D)$ for input x_i

Model Loss (Error)

- Squared loss of model on test case i:

$$\left(Learn(x_i,D) - Truth(x_i)\right)^2$$

- Expected prediction error (total model variance):

$$\left\langle \left(Learn(x,D) - Truth(x)\right)^2 \right\rangle_D$$

- Mean squared error on a test set is an estimate of this expected error for one sample D

Sources of Squared Error

- training sample D
- noise in targets or input attributes
- bias (model mismatch)
- randomness in learning algorithm
  - neural net weight initialization
- randomized subsetting of train set:
  - cross validation, train and early stopping set

Bias/Variance Decomposition

$$\left\langle \left(L(x,D) - T'(x)\right)^2 \right\rangle_D = Noise^2 + Bias^2 + Variance$$

$Noise^2$ = lower bound on performance

$Bias^2 = \left(\text{expected error due to model mismatch}\right)^2$

$Variance$ = variation due to train sample and randomization

Bias/Variance Decomposition

$$\left\langle \left(L(x,D) - T'(x)\right)^2 \right\rangle_D = Noise^2 + Bias^2 + Variance$$

$$Noise^2 = \left\langle \left(\mathrm{T'(x) - T(x)}\right)^2 \right\rangle$$

$$= \text{Bayes Optimal Rate}$$

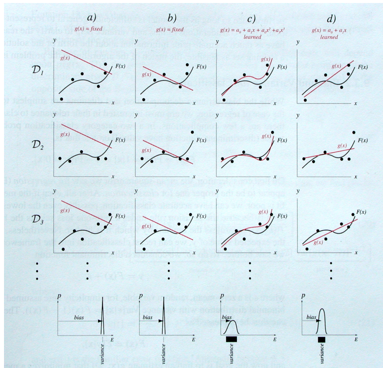
$$Bias^2 = \left(\left\langle L(x,D) - T(x)\right\rangle_D\right)^2$$

$$Variance = \left\langle \left(L(x,D) - \left\langle L(x,D)\right\rangle_D\right)^2 \right\rangle_D$$

Bias/Variance Tradeoff

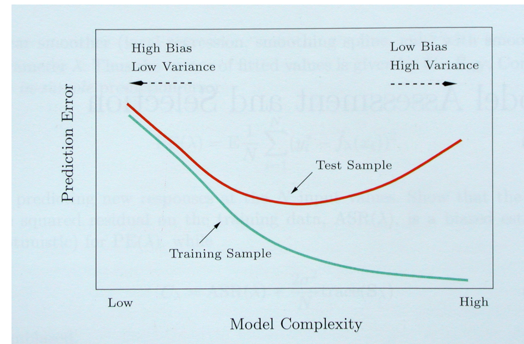
- (bias^2+variance) is what counts for prediction
- Often:
  - low bias => high variance
  - low variance => high bias
- Tradeoff:
  - bias^2 vs. variance

Bias/Variance Tradeoff



Duda, Hart, Stork "Pattern Classification", 2nd edition, 2001

Bias/Variance Tradeoff



Hastie, Tibshirani, Friedman "Elements of Statistical Learning" 2001

Bias^2

- bias$^2 \approx 0$
  - linear regression applied to linear data
  - 2nd degree polynomial applied to quadratic data
  - ANN with many hidden units trained to completion
- bias$^2 >> 0$
  - constant function
  - linear regression applied to very non-linear data
  - ANN with few hidden units applied to non-linear data

Bias^2

- to decrease bias^2
  - add more parameters to model
  - train model further (towards overfitting)
  - use larger/richer model class
- to increase bias^2
  - reduce model complexity
  - use smaller model class

Variance

- Variance ≈ 0
  - constant function
  - model independent of training data
  - model depends on stable measures of data
    - mean
    - median
- Variance >> 0
  - high degree polynomial
  - ANN with many hidden units trained to completion

Variance

- to decrease variance
  - use simpler models (fewer parameters)
  - eliminate outliers before fitting model
  - train model less (early stopping)
  - more pruning
  - use larger training set
- to increase variance
  - use larger/richer model class
  - use more complex models
  - train model further

Bias/Variance Tradeoff

Variance

Bias²	HIGH	MEDIUM	LOW
LOW	1	2	3
MEDIUM	4	5	6
HIGH	7	8	9

Measuring Bias and Variance

$$MSE = \boxed{\left\langle \left(L(x,D) - T(x)\right)^2 \right\rangle_D} = Noise^2 + Bias^2 + Variance$$

- Random train set (fixed size)
- Random test set (any size)
- Train on train
- Compute variance of error on test set
- Repeat many times (use cross validation)
- This is the usual error measure reported

Measuring Bias and Variance

$$\left\langle \left(L(x,D) - T'(x)\right)^2 \right\rangle_D = Noise^2 + Bias^2 + \boxed{\left\langle \left(L(x,D) - \langle L(x,D)\rangle_D\right)^2 \right\rangle_D}$$

- Random test set (preferably large)
- Random train set (fixed size)
- Train on train
- Evaluate on test, saving predictions
- Resample train set, retrain, test on original test set
- Average predictions on test sets
- Calculate variance of predictions to averages

Measuring Bias and Variance

$$\boxed{Noise^2 + Bias^2} = \left\langle \left(L(x,D) - T'(x)\right)^2 \right\rangle - Variance$$

$$Noise^2 = \left\langle \left(\mathrm{T'(x)} - \mathrm{T(x)}\right)^2 \right\rangle$$

$$Bias^2 = \left(\langle L(x,D) - T(x) \rangle_D \right)^2$$

- Can't measure bias^2 separate from noise^2
- Lump (bias^2 + noise^2) together
- Calculate by subtracting variance from MSE (mean squared error)

Measuring Bias and Variance

- Calculate MSE usual way
- Calculate average test set predictions by averaging predictions from many train samples
- Calculate variance of predictions: $\left\langle (y - \bar{y})^2 \right\rangle$

$$MSE = Bias^2 + Variance$$

Model Complexity

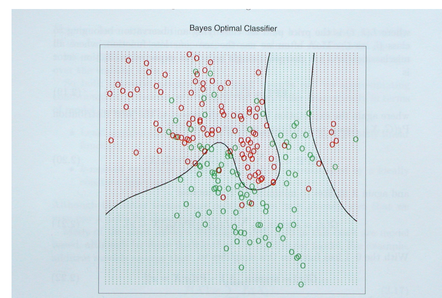


FIGURE 2.5. The optimal Bayes decision boundary for the simulation example of Figures 2.1, 2.2 and 2.3. Since the generating density is known for each class, this boundary can be calculated exactly (Exercise 2.2).

Hastie, Tibshirani, Friedman "Elements of Statistical Learning" 2001

Model Complexity

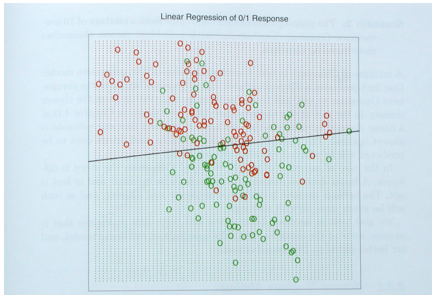


FIGURE 2.1. *A classification example in two dimensions. The classes are coded as a binary variable*—GREEN $= 0$, RED $= 1$—*and then fit by linear regression. The line is the decision boundary defined by* $x^T\hat{\beta} = 0.5$. *The red shaded region denotes that part of input space classified as* RED, *while the green region is classified as* GREEN.

Hastie, Tibshirani, Friedman "Elements of Statistical Learning" 2001

Model Complexity

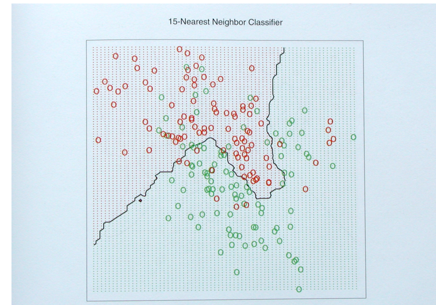


FIGURE 2.2. *The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable* (GREEN $= 0$, RED $= 1$) *and then fit by 15-nearest-neighbor averaging as in (2.8). The predicted class is hence chosen by majority vote amongst the 15-nearest neighbors.*

Hastie, Tibshirani, Friedman "Elements of Statistical Learning" 2001

Model Complexity

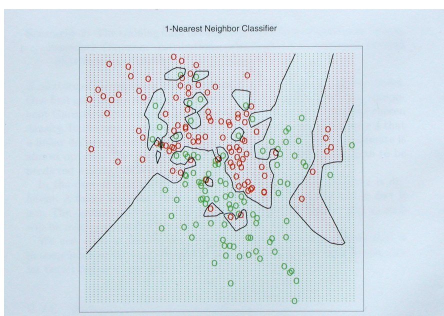
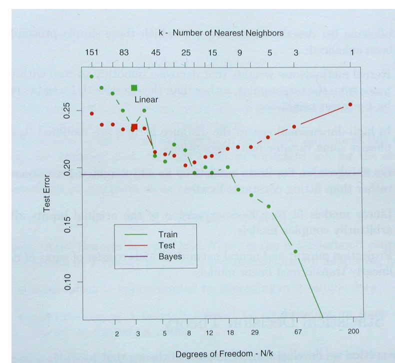


FIGURE 2.3. *The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable* (GREEN $= 0$, RED $= 1$), *and then predicted by 1-nearest-neighbor classification.*

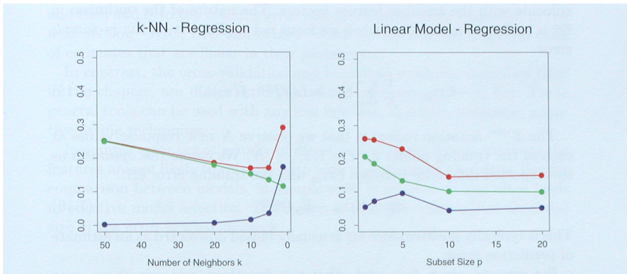
Hastie, Tibshirani, Friedman "Elements of Statistical Learning" 2001

Model Complexity



Hastie, Tibshirani, Friedman "Elements of Statistical Learning" 2001

Empirical Bias/Variance Decomposition



Hastie, Tibshirani, Friedman “Elements of Statistical Learning” 2001

Bias/Var Decomposition for Classification

- classification is 0/1 loss, not squared loss
- B/V decomposition not so straightforward
- active research (3 papers = 3 different approaches)

Bias/Var Decomposition for Classification

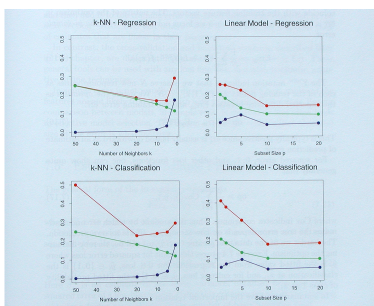


FIGURE 7.3. *Prediction error (red), squared bias (green) and variance (blue) for a simulated example. The top row is regression with squared error loss; the bottom row is classification with 0–1 loss. The models are k-nearest neighbors (left) and best subset regression of size p (right). The variance and bias curves are the same in regression and classification, but the prediction error curve is different.*

Hastie, Tibshirani, Friedman “Elements of Statistical Learning” 2001

So What?

- Reduce one without increasing the other?
- Better understand algorithms?
- Design better algorithms?

So What?

- Reduce one without increasing the other?
- Better understand algorithms?
- Design better algorithms?

Yes!

Better Understand Algorithms

Bias/Variance Decomposition for KNN

$$MSE = Noise^2 + Bias^2 + Variance$$

$$= Noise^2 + \left(\frac{\sum_{NN=1}^{k} T(x_{NN})}{k} - T(x) \right)^2 + \frac{Noise^2}{k}$$

- Note: This simplified analysis assumes points x_i are fixed (no variation from moving neighbors).

Better Algorithms by Reducing Variance

Reduce Variance Without Increasing Bias?

- Averaging reduces variance:

$$Var(\overline{X}) = \frac{Var(X)}{N}$$

- Average models to reduce model variance
- One problem:
  - only one train set
  - where do multiple models come from?

Bagging: Bootstrap Aggregation

- Leo Breiman (1994)
- Bootstrap Sample:
  - draw sample of size |D| with replacement from D

$$\text{Train } L_i\left(BootstrapSample_i(D)\right)$$

$$\text{Regression: } L_{bagging} = \overline{L_i}$$

$$\text{Classification: } L_{bagging} = Plurality\left(L_i\right)$$

Bagging

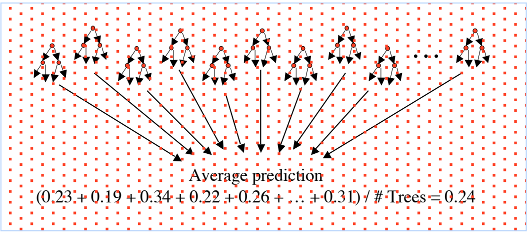
- Best case:

$$Var(Bagging(L(x,D))) = \frac{Variance(L(x,D))}{N}$$

- In practice:
  - models are correlated, so reduction is smaller than 1/N
  - variance of models trained on fewer training cases usually somewhat larger
  - stable learning methods have low variance to begin with, so bagging may not help much

Bagged Decision Trees

- Draw 100 bootstrap samples of data
- Train trees on each sample -> 100 trees
- Un-weighted average prediction of trees



- Highly under-rated!

Bagging Results

Table 1 Missclassification Rates (Percent)

Data Set	$\bar{e}_S$	$\bar{e}_B$	Decrease
waveform	29.0	19.4	33%
heart	10.0	5.3	47%
breast cancer	6.0	4.2	30%
ionosphere	11.2	8.6	23%
diabetes	23.4	18.8	20%
glass	32.0	24.9	22%
soybean	14.5	10.6	27%

Breiman "Bagging Predictors" Berkeley Statistics Department TR#421, 1994

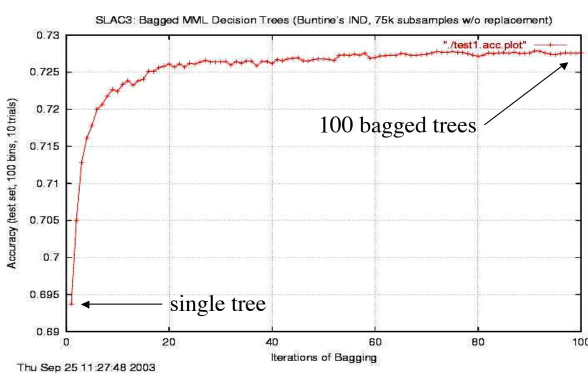
How Many Bootstrap Samples?

Table 5.1
Bagged Missclassification Rates (%)

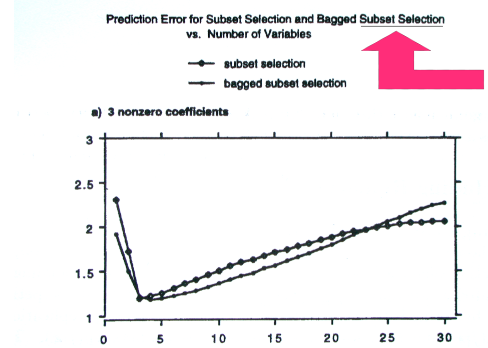
No. Bootstrap Replicates	Missclassification Rate
10	21.8
25	19.5
50	19.4
100	19.4

Breiman "Bagging Predictors" Berkeley Statistics Department TR#421, 1994

How Many Bootstrap Samples?



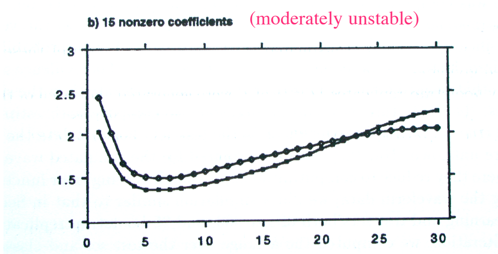
Bagging Stable vs. Unstable Models



linear regression is stable,but forward stepwise variable selection is unstable if there are many non-zero coefficients

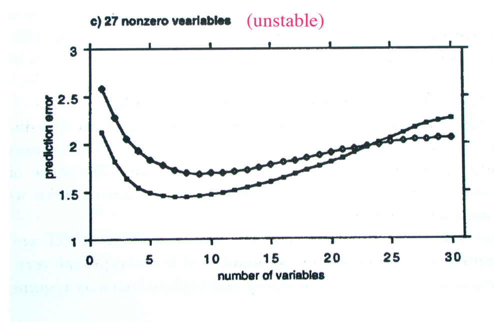
Breiman "Bagging Predictors" Berkeley Statistics Department TR#421, 1994

Bagging Stable vs. Unstable Models



Breiman "Bagging Predictors" Berkeley Statistics Department TR#421, 1994

Bagging Stable vs. Unstable Models



Breiman "Bagging Predictors" Berkeley Statistics Department TR#421, 1994

Can Bagging Hurt?

Can Bagging Hurt?

- Each base classifier is trained on less data
  - Only about 63.2% of the data points are in any bootstrap sample
- However the final model has seen all the data
  - On average a point will be in >50% of the bootstrap samples

Reduce $Bias^2$ and Decrease Variance?

- Bagging reduces variance by averaging
- Bagging has little effect on bias
- Can we average *and* reduce bias?
- Yes:

Boosting

Boosting

- Freund & Schapire:
  - theory for “weak learners” in late 80’s
- Weak Learner: performance on ***any*** train set is slightly better than chance prediction
- intended to answer a theoretical question, not as a practical way to improve learning
- tested in mid 90’s using not-so-weak learners
- works anyway!

The Potential Power of Weak Learning

Train Case	C_1	C_2	C_3	C_4	C_5
T(x) (true y)	1	0	0	1	0
Weak L_1	1	0	1	0	0
Weak L_2	1	1	0	1	1
Weak L_3	0	0	0	1	1
Weak L_4	0	0	1	1	0
…					
Weak L_n	1	1	0	0	0
<Weak L_n>	1	0	0	1	0

Boosting

- Weight all training samples equally
- Train model on train set
- Compute error of model on train set
- Increase weights on train cases model gets wrong
- Train new model on re-weighted train set
- Re-compute errors on weighted train set
- Increase weights again on cases model gets wrong
- Repeat until tired (100+ iteraations)
- Final model: weighted prediction of each model

Boosting

Initialization

Iteration

Final Model

Algorithm AdaBoost.M1
Input: sequence of m examples $\langle (x_1, y_1), \ldots, (x_m, y_m) \rangle$
with labels $y_i \in Y = \{1, \ldots, k\}$
weak learning algorithm **WeakLearn**
integer T specifying number of iterations
Initialize $D_1(i) = 1/m$ for all i.
Do for $t = 1, 2, \ldots, T$:
1. Call **WeakLearn**, providing it with the distribution D_t.
2. Get back a hypothesis $h_t : X \to Y$.
3. Calculate the error of h_t: $\epsilon_t = \sum_{i: h_t(x_i) \neq y_i} D_t(i)$.
   If $\epsilon_t > 1/2$, then set $T = t - 1$ and abort loop.
4. Set $\beta_t = \epsilon_t / (1 - \epsilon_t)$.
5. Update distribution D_t:
   $$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} \beta_t & \text{if } h_t(x_i) = y_i \\ 1 & \text{otherwise} \end{cases}$$
   where Z_t is a normalization constant (chosen so that D_{t+1} will be a distribution).

Output the final hypothesis:
$$h_{fin}(x) = \arg\max_{y \in Y} \sum_{t: h_t(x) = y} \log \frac{1}{\beta_t}.$$

Boosting: Initialization

Algorithm AdaBoost.M1
Input: sequence of m examples $\langle (x_1, y_1), \ldots, (x_m, y_m) \rangle$
with labels $y_i \in Y = \{1, \ldots, k\}$
weak learning algorithm **WeakLearn**
integer T specifying number of iterations
Initialize $D_1(i) = 1/m$ for all i.

Boosting: Iteration

Do for $t = 1, 2, \ldots, T$:
1. Call **WeakLearn**, providing it with the distribution D_t.
2. Get back a hypothesis $h_t : X \to Y$.
3. Calculate the error of h_t: $\epsilon_t = \sum_{i: h_t(x_i) \neq y_i} D_t(i)$.
   If $\epsilon_t > 1/2$, then set $T = t - 1$ and abort loop.
4. Set $\beta_t = \epsilon_t / (1 - \epsilon_t)$.
5. Update distribution D_t:
   $$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} \beta_t & \text{if } h_t(x_i) = y_i \\ 1 & \text{otherwise} \end{cases}$$
   where Z_t is a normalization constant (chosen so that D_{t+1} will be a distribution).

Weight updates

- Weights for incorrect instances are multiplied by 1/(2Error_i)
  - Small train set errors cause weights to grow by several orders of magnitude
- Total weight of misclassified examples is 0.5
- Total weight of correctly classified examples is 0.5

Boosting: Prediction

Output the final hypothesis:

$$h_{fin}(x) = \arg\max_{y \in Y} \sum_{t:h_t(x)=y} \log \frac{1}{\beta_t}.$$

Reweighting vs Resampling

- Example weights might be harder to deal with
  - Some learning methods can't use weights on examples
  - Many common packages don't support weighs on the train
- We can resample instead:
  - Draw a bootstrap sample from the data with the probability of drawing each example is proportional to it's weight
- Reweighting usually works better but resampling is easier to implement

Boosting Theorem

Then the following upper bound holds on the error of the final hypothesis h_{fin}:

$$\frac{|\{i : h_{fin}(x_i) \neq y_i\}|}{m} \leq \prod_{t=1}^{T} \sqrt{1 - 4\gamma_t^2} \leq \exp\left(-2\sum_{t=1}^{T} \gamma_t^2\right).$$

- training error of final weighted hypothesis is small
- does not say anything about error on test set
- if weak hypotheses are simple and T not too large, err(train) - err(test) can be bounded

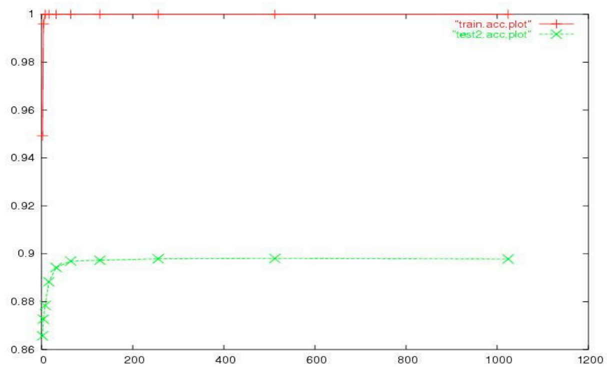
Boosting Theorem

Then the following upper bound holds on the error of the final hypothesis h_{fin}:

$$\frac{|\{i : h_{fin}(x_i) \neq y_i\}|}{m} \leq \prod_{t=1}^{T} \sqrt{1 - 4\gamma_t^2} \leq \exp\left(-2\sum_{t=1}^{T} \gamma_t^2\right).$$

- bound on training error is weak
- but bound is correct qualitatively
- performs better in practice than theory suggests!
- theoretical understanding is not correct/sufficient

Boosting Performance



AdaBoost.M1 can't handle hypothesis with $e_t \geq 0.5$

Algorithm AdaBoost.M2

Input: sequence of m examples $\langle(x_1, y_1), \ldots, (x_m, y_m)\rangle$ with labels $y_i \in Y = \{1, \ldots, k\}$

weak learning algorithm **WeakLearn**

integer T specifying number of iterations

Let $B = \{(i, y) : i \in \{1, \ldots, m\}, y \neq y_i\}$

Initialize $D_1(i, y) = 1/|B|$ for $(i, y) \in B$.

Do for $t = 1, 2, \ldots, T$

1. Call **WeakLearn**, providing it with mislabel distribution D_t.
2. Get back a hypothesis $h_t : X \times Y \to [0, 1]$.
3. Calculate the pseudo-loss of h_t:
   $$\epsilon_t = \frac{1}{2} \sum_{(i,y)\in B} D_t(i, y)\big(1 - h_t(x_i, y_i) + h_t(x_i, y)\big).$$
4. Set $\beta_t = \epsilon_t/(1 - \epsilon_t)$.
5. Update D_t:
   $$D_{t+1}(i, y) = \frac{D_t(i, y)}{Z_t} \cdot \beta_t^{(1/2)(1+h_t(x_i,y_i)-h_t(x_i,y))}$$
   where Z_t is a normalization constant (chosen so that D_{t+1} will be a distribution).

Output the final hypothesis:
$$h_{fin}(x) = \arg\max_{y\in Y} \sum_{t=1}^{T} \left(\log \frac{1}{\beta_t}\right) h_t(x, y).$$

AdaBoost.M2

Do for $t = 1, 2, \ldots, T$

1. Call **WeakLearn**, providing it with mislabel distribution D_t.
2. Get back a hypothesis $h_t : X \times Y \to [0, 1]$.
3. Calculate the pseudo-loss of h_t:
   $$\epsilon_t = \frac{1}{2} \sum_{(i,y)\in B} D_t(i, y)\big(1 - h_t(x_i, y_i) + h_t(x_i, y)\big).$$
4. Set $\beta_t = \epsilon_t/(1 - \epsilon_t)$.
5. Update D_t:
   $$D_{t+1}(i, y) = \frac{D_t(i, y)}{Z_t} \cdot \beta_t^{(1/2)(1+h_t(x_i,y_i)-h_t(x_i,y))}$$
   where Z_t is a normalization constant (chosen so that D_{t+1} will be a distribution).

AdaBoost.M2

Output the final hypothesis:
$$h_{fin}(x) = \arg\max_{y\in Y} \sum_{t=1}^{T} \left(\log \frac{1}{\beta_t}\right) h_t(x, y).$$

AdaBoost.M2

$$\frac{|\{i : h_{fin}(x_i) \neq y_i\}|}{m} \leq (k-1)\prod_{t=1}^{T}\sqrt{1-4\gamma_t^2} \leq (k-1)\exp\left(-2\sum_{t=1}^{T}\gamma_t^2\right)$$

where k is the number of classes.

Boosting vs. Bagging

- Bagging doesn't work so well with stable models. Boosting might still help.
- Boosting might hurt performance on noisy datasets. Bagging doesn't have this problem
- In practice bagging almost always helps.

Boosting vs. Bagging

- On average, boosting helps more than bagging, but it is also more common for boosting to hurt performance.
- The weights grow exponentially.
- Bagging is easier to parallelize.

Boosting Theorem

- Suppose weak learning method WeakLearn, called by AdaBoost, generates hypotheses with errors $e_1,\ldots,e_T$. Assume each $e_t \leq 0.5$, and let $\gamma_t = 0.5 - e_t$.

Then the following upper bound holds on the error of the final hypothesis h_{fin}:

$$\frac{|\{i : h_{fin}(x_i) \neq y_i\}|}{m} \leq \prod_{t=1}^{T}\sqrt{1-4\gamma_t^2} \leq \exp\left(-2\sum_{t=1}^{T}\gamma_t^2\right).$$