# Bias/Variance Tradeoff

# Model Loss (Error)

- Squared loss of model on test case i:

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

- Expected prediction error:

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

# 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$^2$

- Low bias
  - linear regression applied to linear data
  - 2nd degree polynomial applied to quadratic data
  - ANN with many hidden units trained to completion
- High bias
  - constant function
  - linear regression applied to non-linear data
  - ANN with few hidden units applied to non-linear data

# Variance

- Low variance
  - constant function
  - model independent of training data
  - model depends on stable measures of data
    - mean
    - median
- High variance
  - high degree polynomial
  - ANN with many hidden units trained to completion

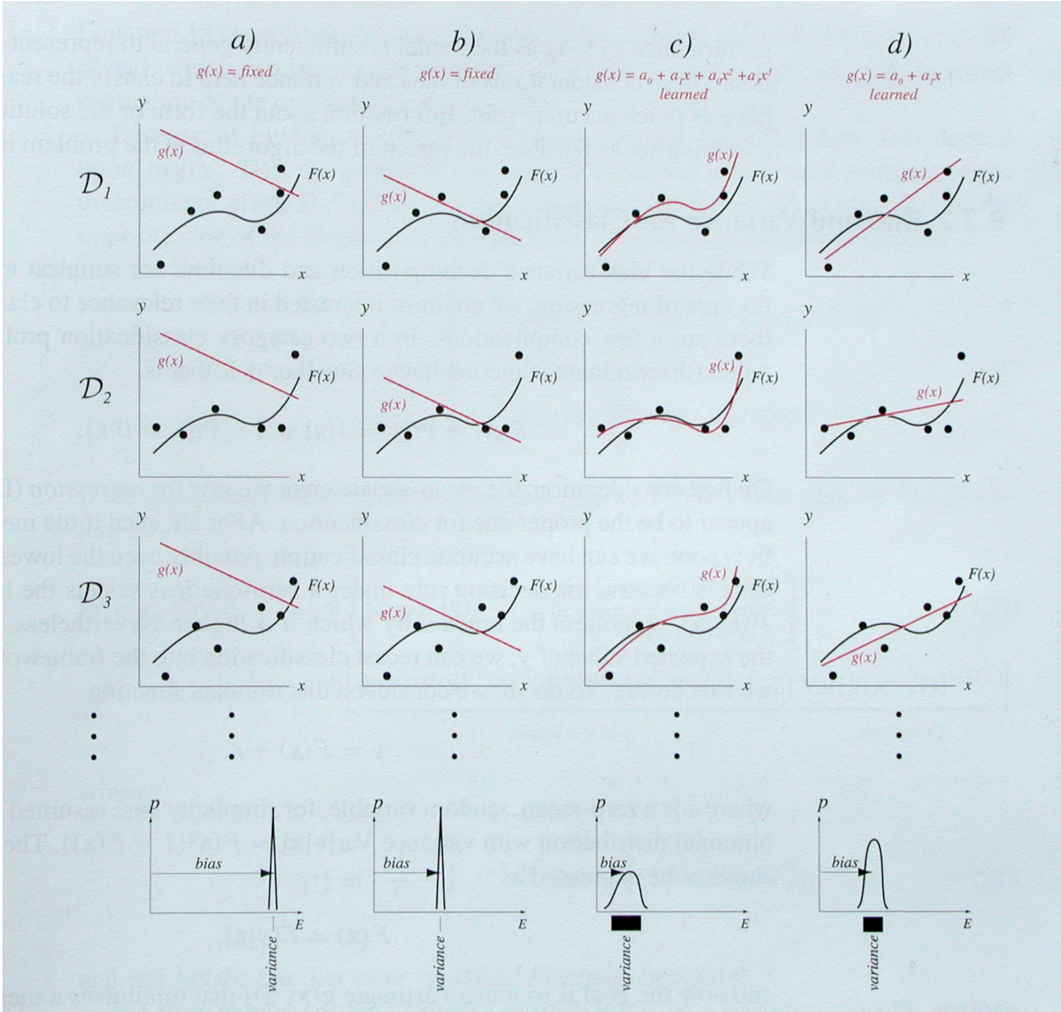
# Sources of Variance in Supervised Learning

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

# Bias/Variance Tradeoff

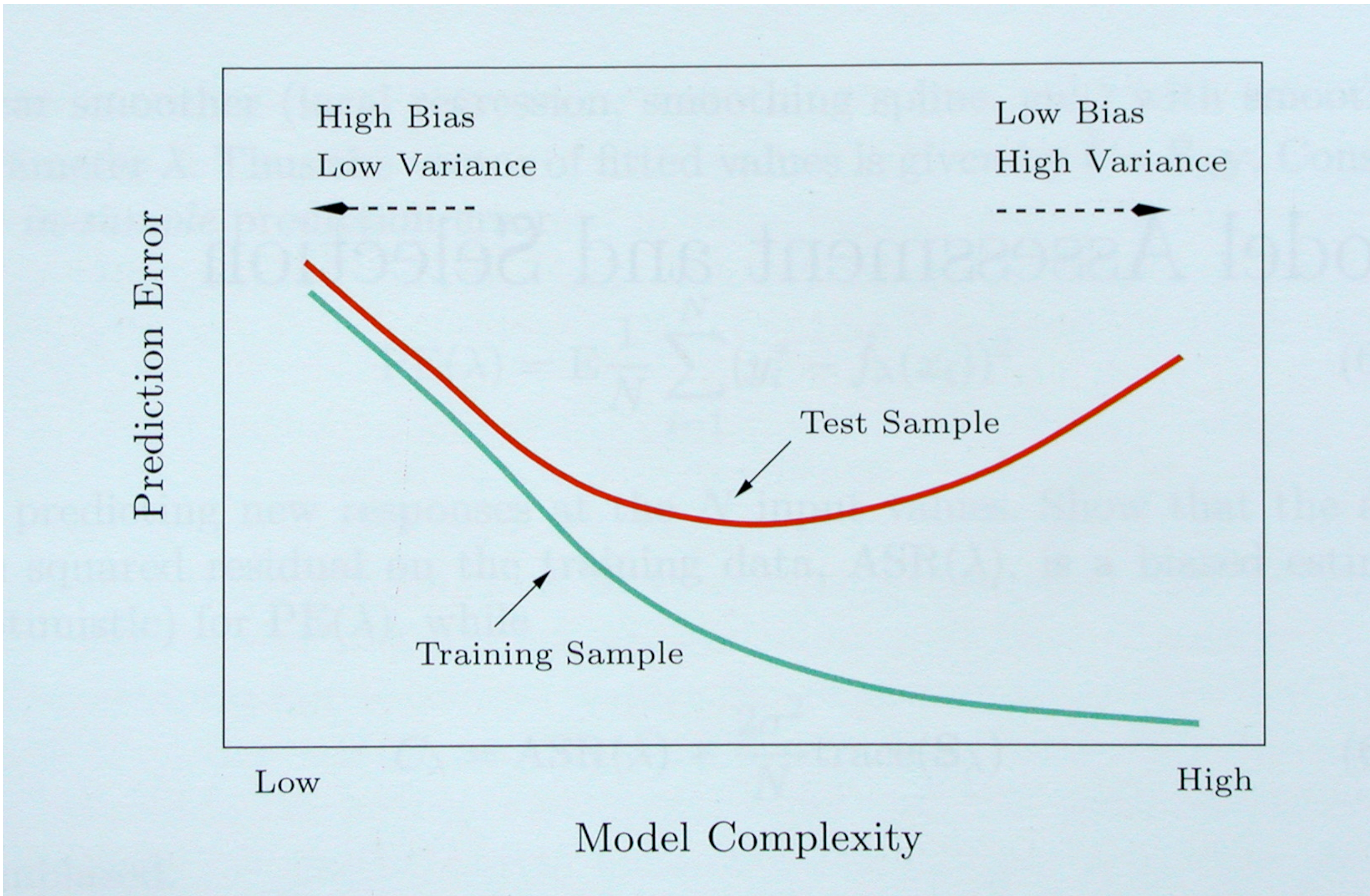
- ($bias^2$+variance) is what counts for prediction
- Often:
  - low bias => high variance
  - low variance => high bias
- Tradeoff:
  - $bias^2$ vs. variance

# Bias/Variance Tradeoff




Duda, Hart, Stork “Pattern Classification”, 2nd edition, 2001

# Bias/Variance Tradeoff




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

# Reduce Variance Without Increasing Bias

- Averaging reduces variance:

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

- 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

# 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

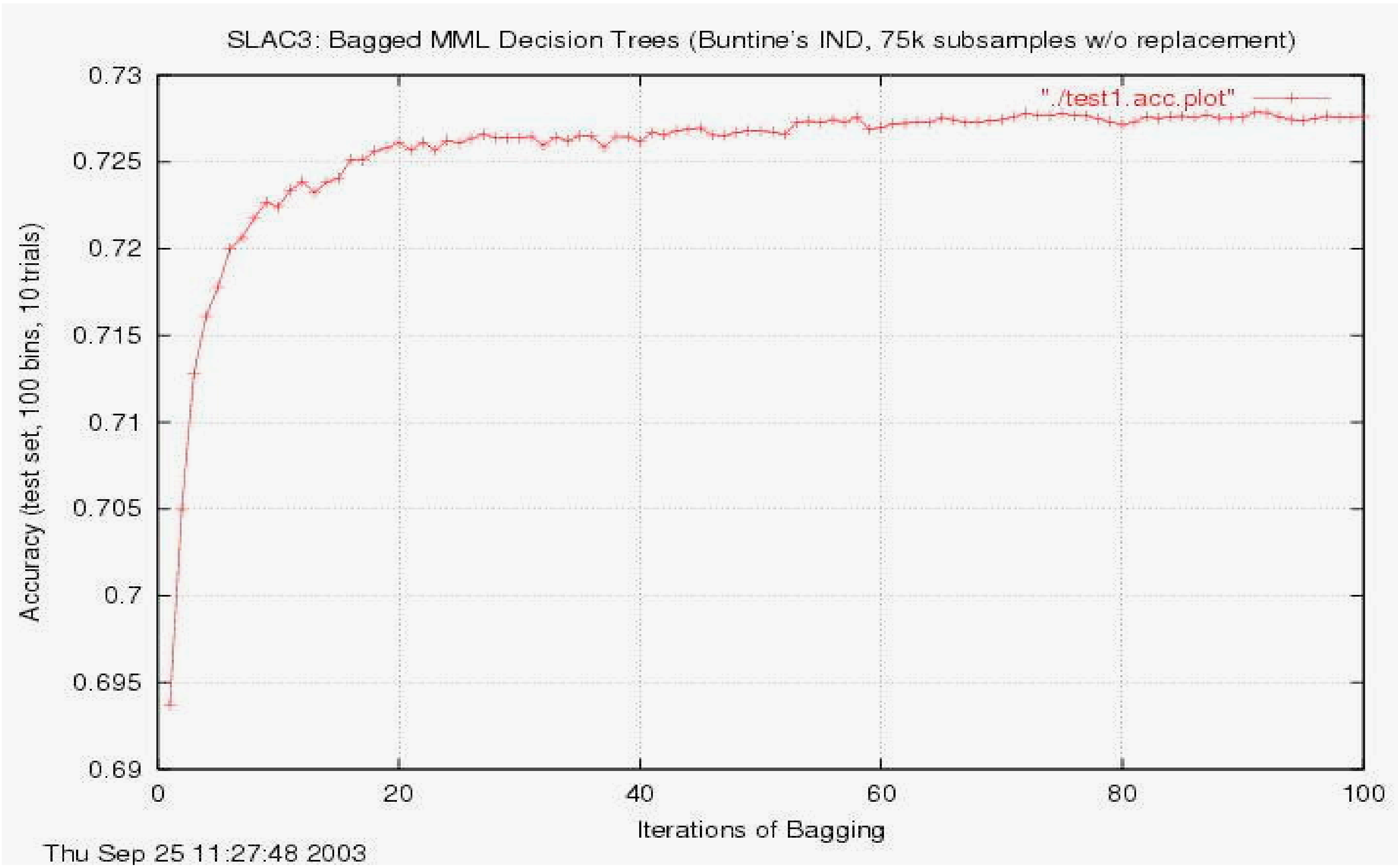
# How Many Bootstrap Samples?

**Table 5.1**
**Bagged Missclassification Rates (%)**

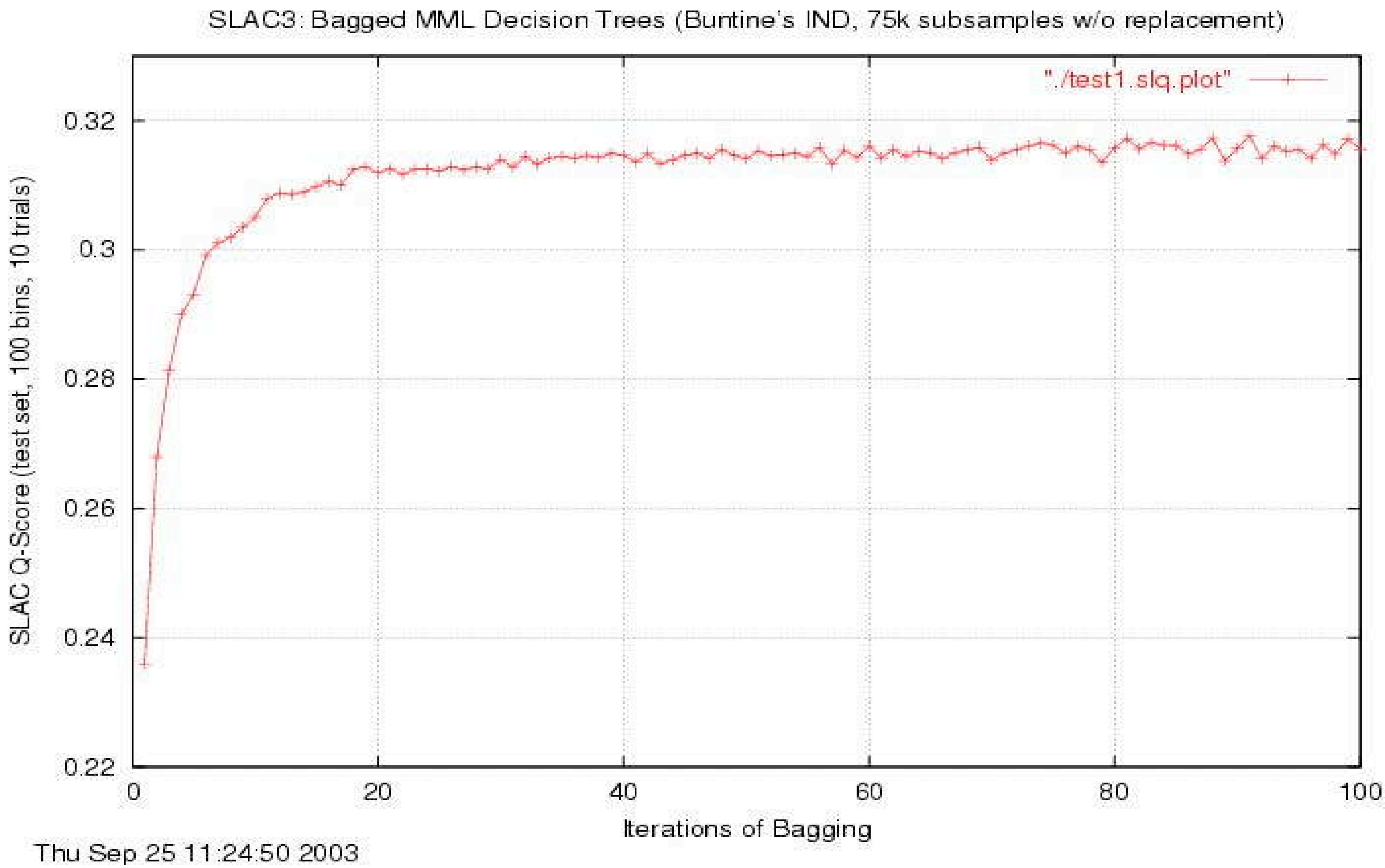
| No. Bootstrap Replicates | Missclassification Rate |
|---|---|
| 10 | 21.8 |
| 25 | 19.5 |
| 50 | 19.4 |
| 100 | 19.4 |

Breiman “Bagging Predictors” Berkeley Statistics Department TR#421, 1994

# More bagging results



# More bagging results



# Bagging with cross validation

- Train neural networks using 4-fold CV
  - Train on 3 folds earlystop on the fourth
  - At the end you have 4 neural nets

- How to make predictions on new examples?

# Bagging with cross validation

- Train neural networks using 4-fold CV
  - Train on 3 folds earlystop on the fourth
  - At the end you have 4 neural nets
- How to make predictions on new examples?
  - Train a neural network until the mean earlystopping point
  - Average the predictions from the four neural networks

# 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!

# 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).

# Boosting: Prediction

**Output** the final hypothesis:

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

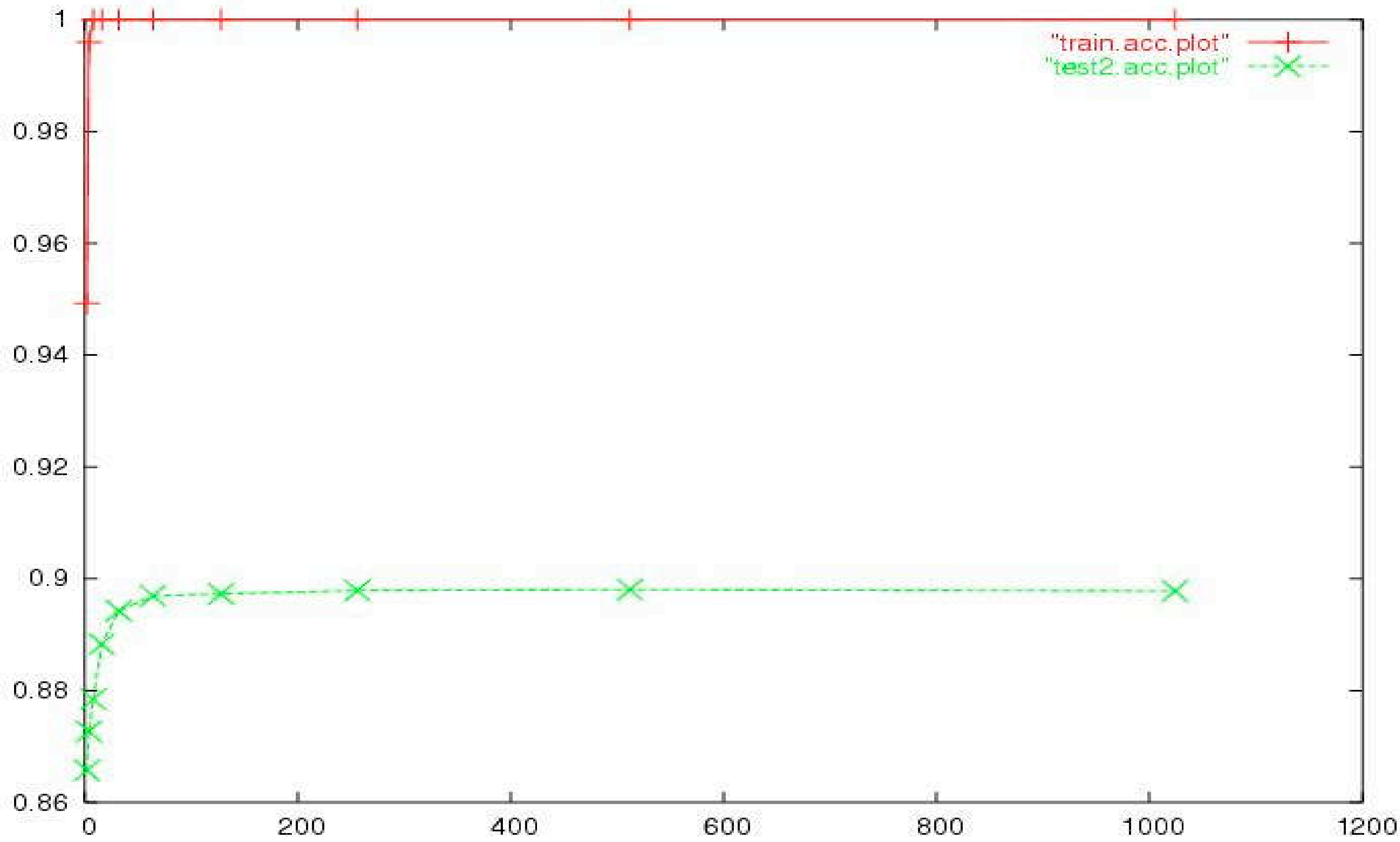
# Weight updates

- 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

# 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 Performance



# 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.