Ensemble Methods: Bagging and Boosting

Ying-Chao Hung
Department of Statistics
National Chengchi University
Email: hungy@nccu.edu.tw

Ensemble methods

- So far, we have covered several learning methods: LDA, QDA, Logistic regression, NN, Decision tree.

- Question: how to improve results?

- Solution: generating and combining multiple predictors
  - Bagging: Bootstrap aggregating
  - Boosting
  - …

Bootstrap Aggregating (Bagging)

- Bagging (Breiman, 1996) is a machine-learning method designed to improve the **stability** (in terms of variance) and **prediction accuracy** (avoid overfitting) of the classifier.

- It can be used with any types of classifiers (though it is usually used by the "classification tree").

- It belongs to the class of **model averaging approaches**.

Rationale of Bootstrapping

- Question: What's the average price of house prices?
- From the price distribution F, get a sample $x = (x_1, x_2, \ldots, x_n)$, and calculate the average $\bar{x}$.
- Question: How reliable is $\bar{x}$ when used to estimate a population parameter θ ?
- Difficulty: In some cases where the sampling distribution of $\bar{x}$ can not be obtained, how to obtain a good estimate of θ ?
- Possibility: Can get "several samples" from F, but, it is often impossible or too expensive to get multiple samples.

Idea of Bootstrap Sampling

Let the original sample be $x = (x_1, x_2, \ldots, x_n)$

- Generate a sample x^* of size n from x by sampling <u>with replacement.</u>
- Compute the statistic of interest $\hat{\theta}^*$ for the bootstrap sample x^*.

➔ Repeat the sampling B times, now we obtain the bootstrap values

$$\hat{\theta}^* = (\hat{\theta}_1^*, \ldots, \hat{\theta}_B^*)$$

- Use these bootstrap values (or distribution) for calculating the quantities of interest (e.g., standard deviation, confidence intervals, etc)

More on bootstrapping

- Introduced by Bradley Efron in 1979
- Named from the phrase “to pull oneself up by one’s bootstraps”, which is sometimes attributed to a British story/ movie “The Surprising Adventures of *Baron Munchausen*”- (終極天將).
- Popularized in 1980s due to the introduction of computers in statistical practice.
- It has a strong mathematical background.
- It is well known as a method for estimating standard errors, bias, and constructing confidence intervals for parameters.

What are Bootstraps?

Bootstrap Distribution

- Different from the "sampling distribution" based on multiple samples, it is free to develop the bootstrap distribution based on a number of bootstrap samples originated from merely "one sample".

- The bootstrap distribution is centered at the sample statistic, e.g., all bootstrap sample means $\bar{x}^*$ have the center/mean at $\bar{x}$, while the sample statistics (if there are a lot) have the center/mean at the population parameter θ, it is then natural to make inference about θ by utilizing the bootstrap distribution (though there always exists a bias).

$$(\bar{x}^* \rightarrow \bar{x} \rightarrow \mu)$$

Cases where bootstrap does not apply

- Small data sets: the original sample is not a good approximation of the population (i.e., n is small)

- Dirty data: There exist outliers that increase variability in the estimates (so data cleaning/screening is important).

- Dependence structures (e.g., time series, spatial problems): Bootstrap is based on the assumption of "independence between sampled observations" (i.e., random sample).

  → In this case, the wild/block bootstrap can be used.

How many bootstrap samples?

The number of necessary bootstrap samples depends on:

- Computer availability
- Type of the problem: standard errors, confidence intervals, …
- Complexity of the problem

Other Resampling methods

- Estimating the precision of sample statistics:
  - Bootstrap: (sampling with replacement)
  - Jackknife: (using subset of data - ignore one observation at each time)
- Significance tests, Permutation/exact tests, Randomization tests: exchanging labels on data points
- Validating models by using random subsets
  - Bootstrap
  - Cross Validation

Idea of Bagging

- Randomly generate *L* set of size *n* from the original set ***Z*** with replacement (generate *L* bootstrap samples).

- Each bootstrap sample is used to train a different component of base classifier.

  ➔ So there are *L* classifiers in total.

- Aggregating classifiers: Classification is done by "plurality voting" based on the obtained *L* classifiers.

Algorithm of Bagging

BAGGING

Training phase

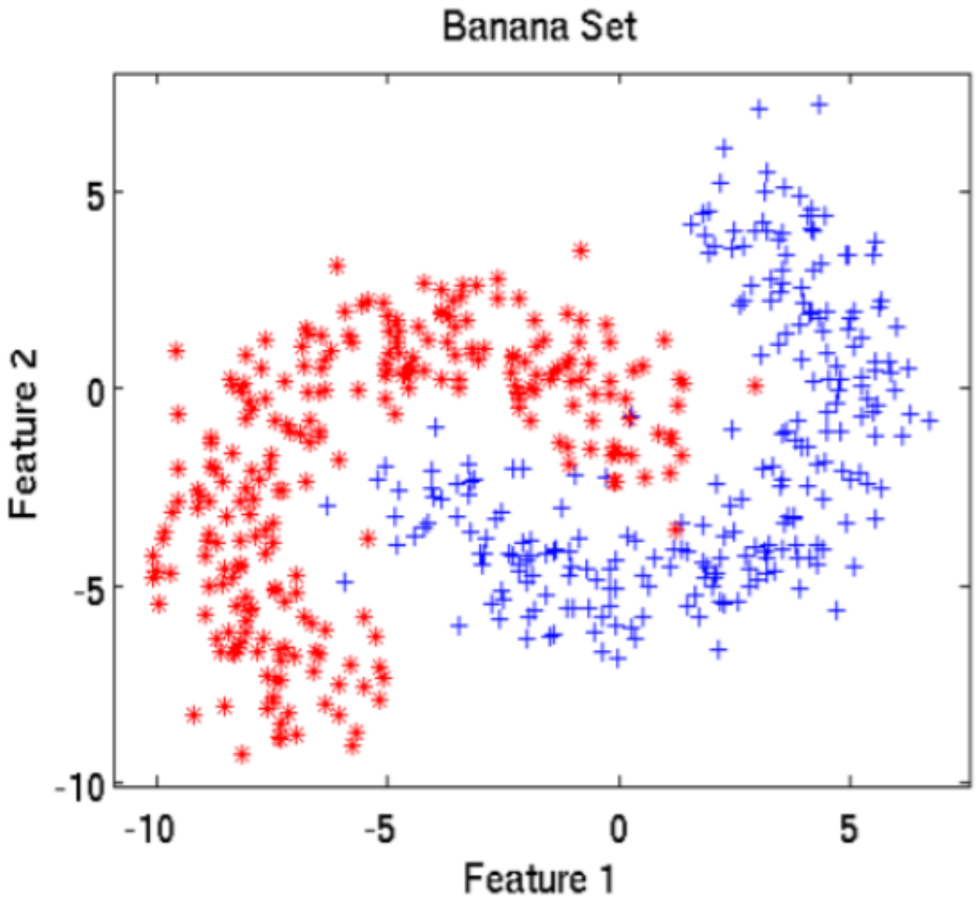
1. Initialize the parameters
   - $\mathcal{D} = \emptyset$, the ensemble.
   - L, the number of classifiers to train.
2. For $k = 1, \ldots, L$
   - Take a bootstrap sample S_k from X.
   - Build a classifier D_k using S_k as the training set.
   - Add the classifier to the current ensemble, $\mathcal{D} = \mathcal{D} \cup D_k$.
3. Return $\mathcal{D}$.

Classification phase

4. Run $D_1, \ldots, D_L$ on the input $\mathbf{x}$.
5. The class with the maximum number of votes is chosen as the label for $\mathbf{x}$.

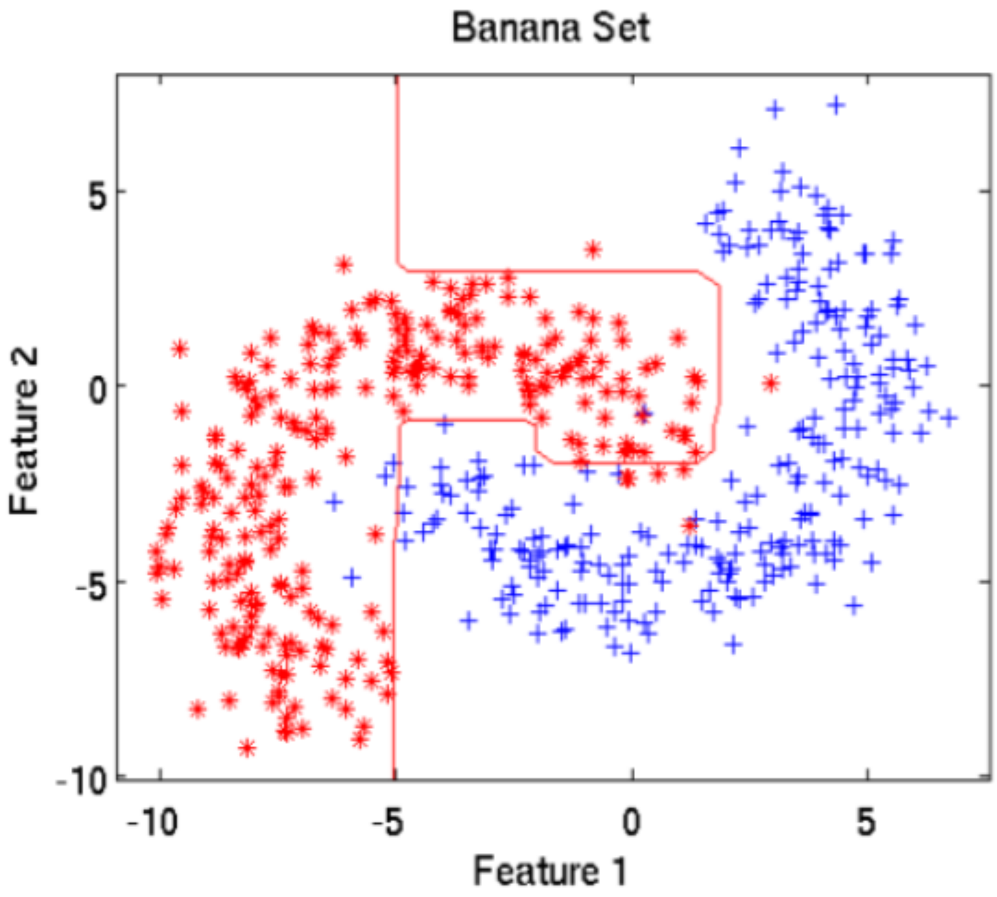
Example: Bagging for Decision Trees

Training Data:



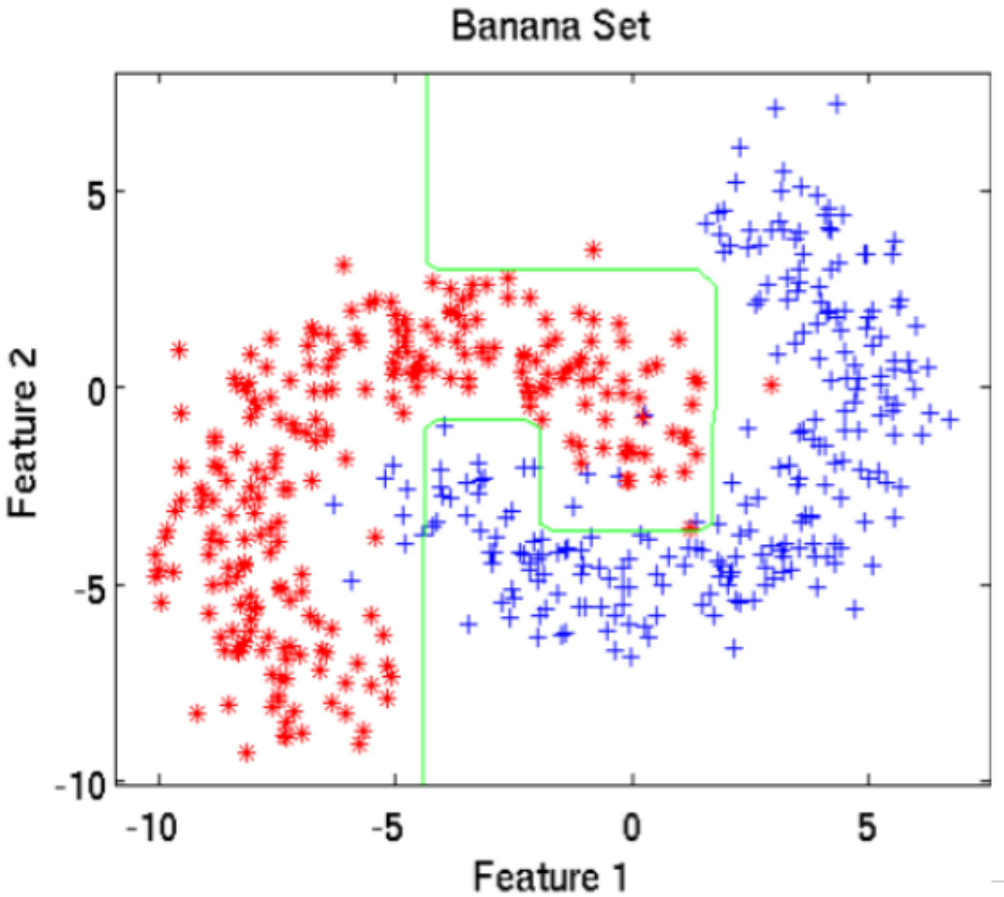
Example: Bagging for Decision Trees

Decision Tree 1:



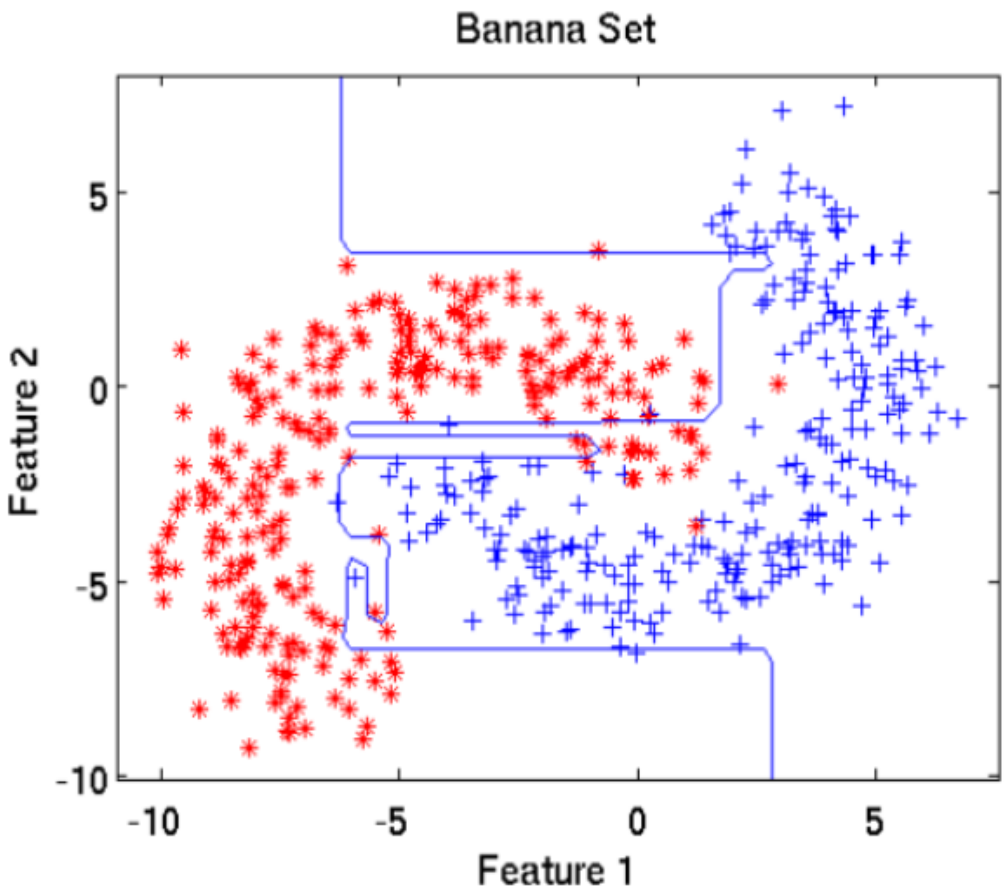
Example: Bagging for Decision Trees

Decision Tree 2:



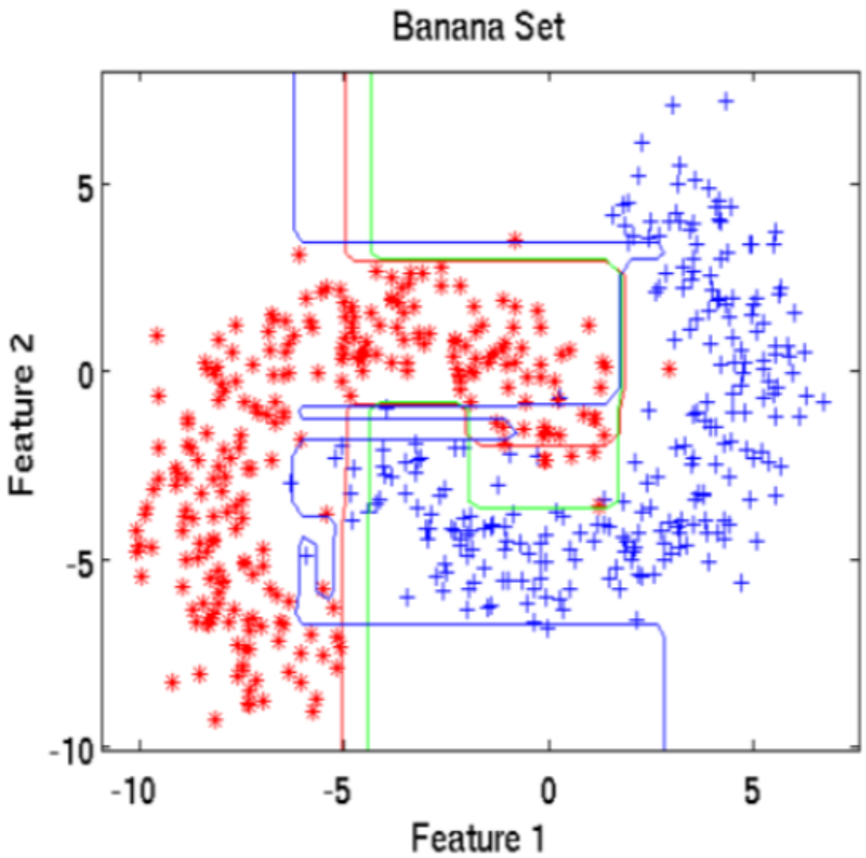
Example: Bagging for Decision Trees

Decision Tree 3:

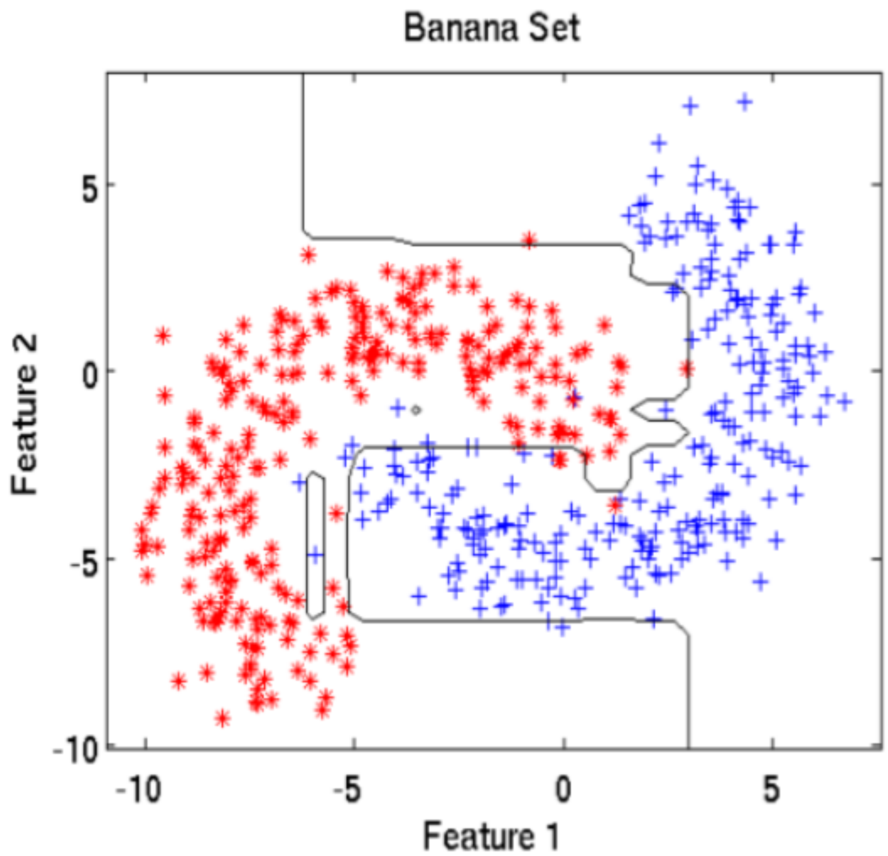


Example: Bagging for Decision Trees

3 Trees:



Result of Bagging:



Why does Bagging work?

- Main errors for classification:
  - Noise: error by the target function
  - Bias: where the classifier can not learn the target
  - Variance: comes from sampling

- Averaging over bootstrap samples can reduce "**error from variance**", thus minimizing the classification error.

- What types of classifiers benefit most from bagging?

  ➔ **Unstable learning methods** (such as decision trees).

Idea of Boosting

- A technique for combining multiple weak learners/ classifiers whose combined performance is significantly better than that of any "single" learner/classifier

  (i.e. get "boost" in accuracy)

- Sequential training of "weak learners" – Each learner/ classifier is trained on data where each point is "re-weighted" based on the performance/accuracy of the classifier.

- Each classifier votes (with a weight) to obtain a final outcome.

Boosting

- (Schapire, 1989): Construct a “weak” learner based on the training data, run it multiple times by re-weighting the training data, then let all the learned classifiers vote.

- Consider a simple classification rule:

$$C : X \rightarrow Y = \{-1, \ +1\}$$

Suppose there are *M* resulting “weak” learners:

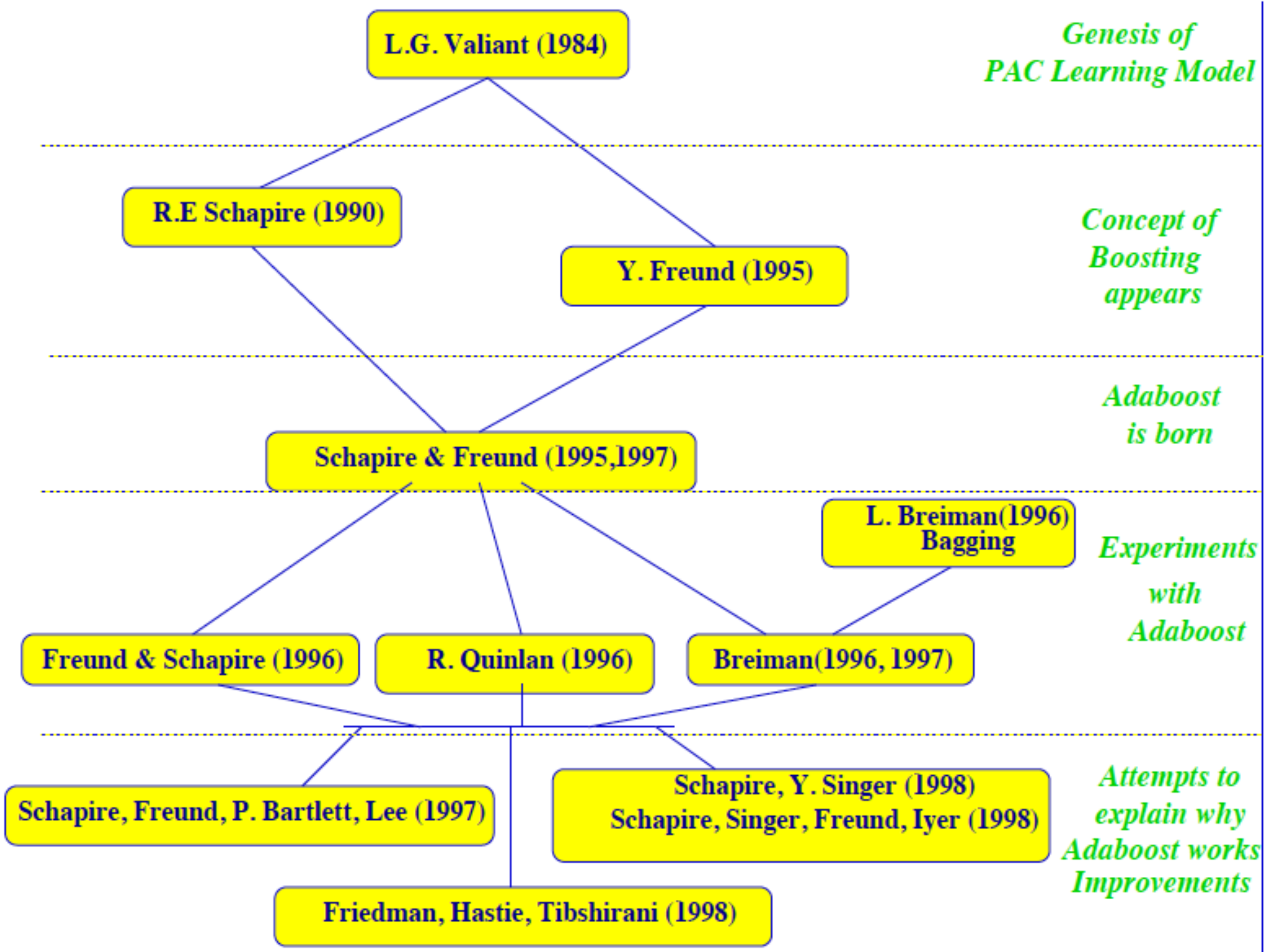
$$C_1(X), C_2(X), \ \ldots, \ C_M(X)$$

The final classifier is:

weight of each leaner at final stage

$$C_{\text{final}}(X) = \text{sign} \sum_{m=1}^{M} \alpha_m C_m(X)$$

History of Boosting



AdaBoost (Freund & Shapire, 1996)

Consider the training data $(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$.

(1) At the initial stage we assign equal weights to all data points, say, $w_1(i) = 1/N$, $i = 1, 2, \ldots, N$.

(2) At stage t, find the best (but still weak) classifier $C_t(x)$ based on the data with weights $w_t(i)$.

(3) Compute the error rate for $C_t(x)$ based on the weighted data:

$$\varepsilon_t = P_{w_t}\left(C_t(x_i) \neq y_i\right) = \frac{\sum_{i=1}^{N} w_t(i) I\{C_t(x_i) \neq y_i\}}{\sum_{i=1}^{N} w_t(i)}$$

See later

Assign the weight to $C_t(x)$: $\alpha_t = \frac{1}{2}\log\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right) > 0$ (since $\varepsilon_t < 0.5$)

AdaBoosting (continued)

(4) Update the data weights by

$$w_{t+1}(i) = \frac{w_t(i)}{Z_t} \cdot \begin{cases} e^{-\alpha_t} & \text{if } C_t(x_i) = y_i \quad \text{(correct decision)} \\ e^{\alpha_t} & \text{if } C_t(x_i) \neq y_i \quad \text{(wrong decision)} \end{cases}$$

$$= \frac{w_t(i)}{Z_t} \cdot \exp\{-\alpha_t y_i C_t(x_i)\},$$

where $Z_t = \sum_{i=1}^{N} w_t(i) \cdot \exp\{-\alpha_t y_i C_t(x_i)\}$ is a normalization constant s.t. $\sum_{i=1}^{N} w_{t+1}(i) = 1$.

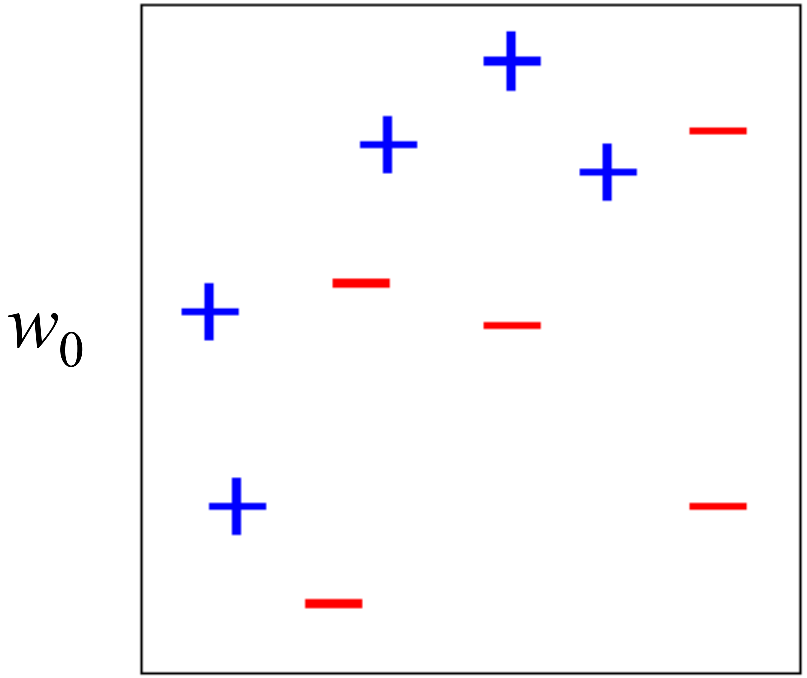
(5) Repeat (2)-(4) M times so that M (weak) classifiers are obtained.

The final classifier is:

$$C_{\text{final}}(x) = \text{sign} \sum_{t=1}^{M} \alpha_t C_t(x)$$

Demo of AdaBoost

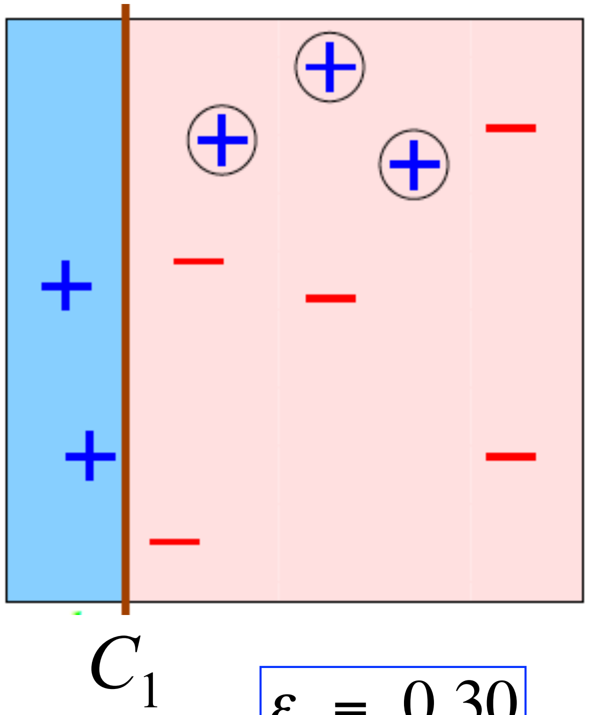
Toy Example



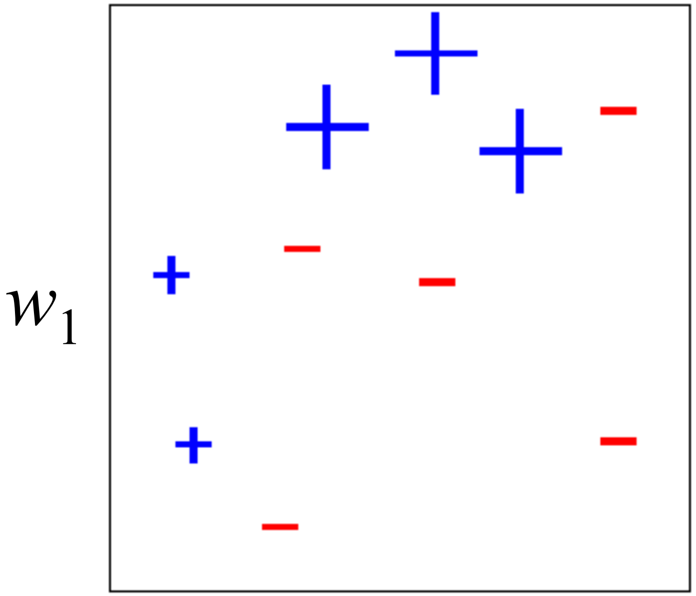
Demo of AdaBoost

Stage 1

給大的權重



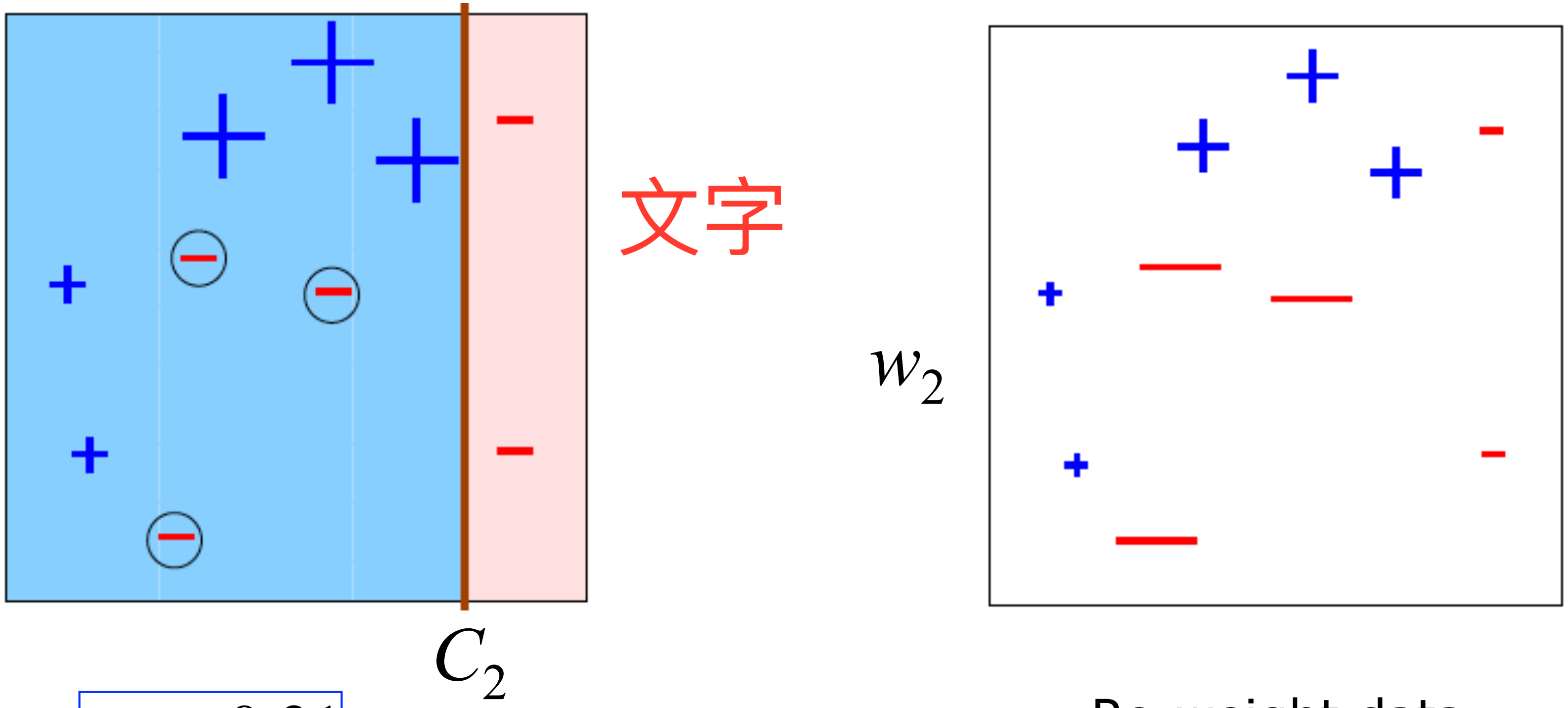
$$\varepsilon_1 = 0.30$$
$$\alpha_1 = 0.42$$



Re-weight data

Demo of AdaBoost

Stage 2

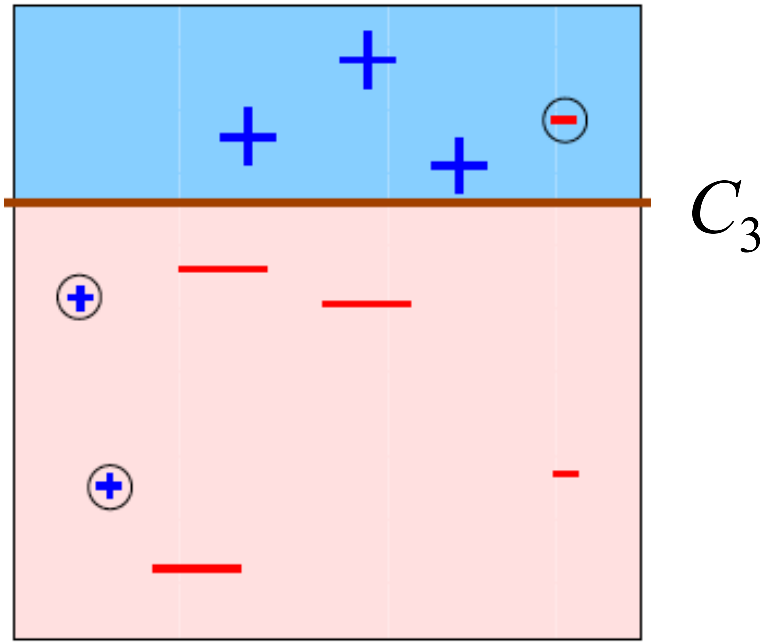


$$\varepsilon_2 = 0.21$$
$$\alpha_2 = 0.65$$

Re-weight data

Demo of AdaBoost

Stage 3

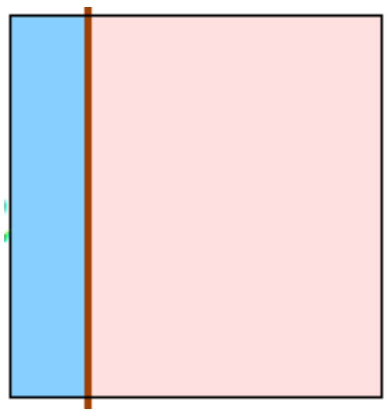
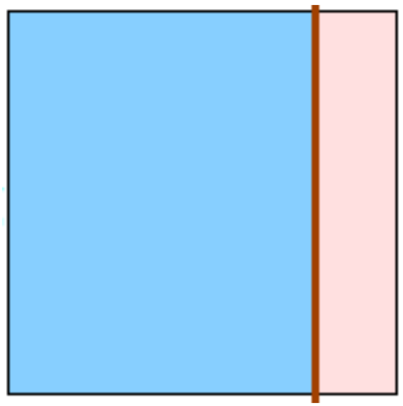
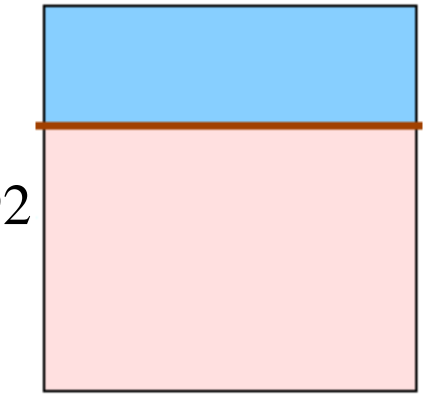


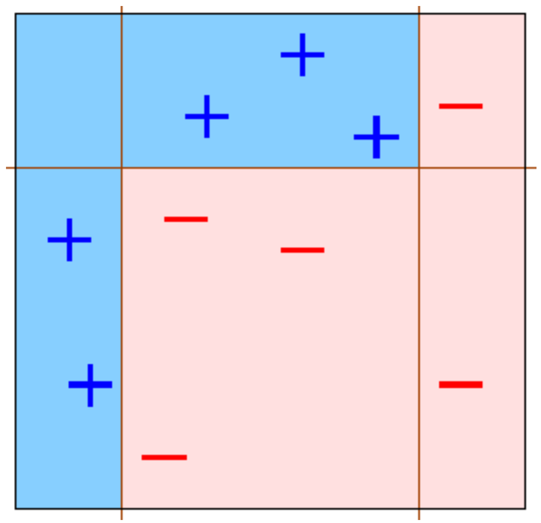
$$\varepsilon_3 = 0.14$$

$$\alpha_3 = 0.92$$

Demo of AdaBoost

Final classifier $C_{\text{final}}(x)$

= (0.42  + 0.65  + 0.92 )

= 

www.research.att.com/~yoav/adaboost

The Training Error of AdaBoost

Let us first focus on the binary classification problem.

Theorem 1.

$$\text{Training Error } (C_{\text{final}}(x)) \le \prod_{t=1}^{M} Z_t \,.$$

越小越好

<***Proof***> Let $f(x) = \sum_{t=1}^{M} \alpha_t C_t(x)$, then, $C_{\text{final}}(x) = \text{sign}(f(x))$.

Note that the weight of x_i at the final stage can be written as:

$$w_M(i) = \frac{1}{N} \cdot \frac{\exp\left\{-y_i \sum_t \alpha_t C_t(x_i)\right\}}{\prod_t Z_t} = \frac{1}{N} \cdot \frac{e^{-y_i f(x_i)}}{\prod_t Z_t}$$

The Training Error of AdaBoost

<*Proof*> (continued)

Since $C_{\text{final}}(x) \neq y \Rightarrow y f(x) \leq 0 \Rightarrow e^{-yf(x)} \geq 1$, we have:

$$
\begin{aligned}
\text{Training Error }(C_{\text{final}}(x)) &= \frac{1}{N} \cdot \sum_{i=1}^{N} \underbrace{I\left\{y_i \neq C_{\text{final}}(x_i)\right\}}_{\boxed{\leq 1}} \\
&\leq \frac{1}{N} \cdot \sum_{i=1}^{N} e^{-y_i f(x_i)} \\
&= \sum_{i=1}^{N} w_M(i) \cdot \left(\prod_t Z_t \right) \\
&= \prod_t Z_t
\end{aligned}
$$

$\square$

How did we choose α_t ?

By Theorem 1, in order to "minimize" the upper bound of the final training error, at each iteration/stage we can choose the value of α_t so that Z_t is minimized.

To see this,

$$Z_t = \sum_{i=1}^{N} w_t(i) \cdot \exp\{-\alpha_t y_i C_t(x_i)\}$$

at 的選取可以影響Zt

$$= \sum_{i:y_i \neq C_t(x_i)} w_t(i) e^{\alpha_t} + \sum_{i:y_i = C_t(x_i)} w_t(i) e^{-\alpha_t} = \varepsilon_t e^{\alpha_t} + (1-\varepsilon_t) e^{-\alpha_t}$$

Thus,

$$\frac{\partial Z_t}{\partial \alpha_t} = 0 \Rightarrow \varepsilon_t e^{\alpha_t} - (1-\varepsilon_t) e^{-\alpha_t} = 0 \Rightarrow \boxed{\alpha_t = \frac{1}{2} \log\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)}.$$

The Training Error of AdaBoost

Let $\varepsilon_t = 1/2 - \gamma_t$ (recall that even a weak learner s.t. $\varepsilon_t < 0.5$).

<u>Theorem 2 (extension of Theorem 1):</u>

$$\text{Training Error } (C_{\text{final}}(x)) \le \prod_{t=1}^{M} \left\{ 2\sqrt{\varepsilon_t (1 - \varepsilon_t)} \right\}$$

$$= \prod_{t=1}^{M} \sqrt{1 - 4\gamma_t^2}$$

$$\le \exp\left\{ -2 \sum_{t=1}^{M} \gamma_t^2 \right\}$$

<u>Note:</u> If $\gamma_t \ge \gamma > 0 \ \forall\ t$, then:

$$\boxed{\text{Training Error } (C_{\text{final}}(x)) \le \exp\left\{ -2\gamma^2 M \right\}.}$$

The Training Error of AdaBoost

<Proof of Theorem 2>

Note that by choosing $\alpha_t = \frac{1}{2}\log\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$, Z_t has the minimum value $Z_t = 2\sqrt{\varepsilon_t(1-\varepsilon_t)}$.

Therefore, by Theorem 1 we have:

$$
\begin{aligned}
\text{Training Error } (C_{\text{final}}(x)) &\leq \prod_{t=1}^{M} Z_t \leq \prod_{t=1}^{M} \left\{2\sqrt{\varepsilon_t(1-\varepsilon_t)}\right\} \\
&= \prod_{t=1}^{M} \sqrt{1-4\gamma_t^2} \\
&\leq \exp\left\{-2\gamma_t^2\right\}
\end{aligned}
$$

□

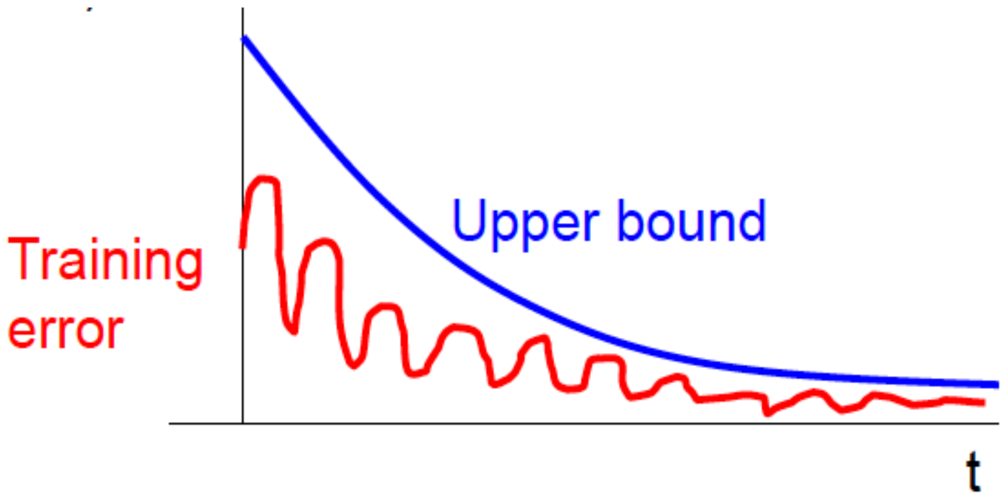
Upper Bound of the Training Error

Note that if $\gamma_t \geq \gamma > 0 \; \forall \; t$, then Theorem 2 implies:

$$\boxed{\text{Training Error } (C_{\text{final}}(x)) \leq \exp\left\{-2\gamma^2 M\right\}.}$$

However, in practice we <u>don't need to know the value of γ.</u>

Also, this shows that the bound goes down "exponentially" <u>to zero</u> as the number of boosting iteration/stage (t) becomes large.



Multiclass Problem

- For $y \in \{1, 2, ..., K\}$, the final classifier is given by

$$C_{\text{final}}(x) = \arg\max_{y} \sum_{t=1}^{M} \alpha_t \cdot I\{C_t(x) = y\}$$

Remarks:

- If the weights α_t are all the same, then $C_{\text{final}}(x)$ simply refers to the "**majority vote**".
- If $\varepsilon_t \leq ½$ for all t, then we can obtain the same upper bound for the training error (not working for "fairly weak" learners).
- The data points with most weight are potential outliers.

Summary-1

- Bagging (Breiman, 1996):

  Construct many classifiers based on the bootstrap-resampled versions of the training data, and classify by majority vote.

  ➔ The goal is "**variance reduction**".

- Boosting (Schapire, 1989; Freund & Shapire, 1996):

  Construct many weak learners to reweighted versions of the training data. Classify by weighted majority vote.

  ➔ Both "**bias**" and "**variance**" are reduced.

- In general, we have: $\boxed{\text{Boosting} \succ \text{Bagging} \succ \text{Single Classifier}}$

  Further, "AdaBoost $\cdots$ best off-the-shelf classifier in the world" — Leo Breiman, NIPS workshop, 1996.

Summary-2

- Boosting can of course "over-fit", stopping early is a good way to regularize boosting.
- Boosting fits "additive logistic models", where each base learner is simple. It considers a similar loss function to that of logistic regression.
- Modern versions of boosting use different loss functions (e.g. gradient boosting) ➔ can handle a wide variety of regression modeling scenarios.