# Machine Learning

CSCI 5622 Fall 2020

Prof. Claire Monteleoni

# Today

- Hyper-Parameter tuning (continued)
  – Picture of Knee in Curve
- Ensemble Methods (continued)
  – [Finish] Decision Forests, Random Forests
  – Bagging
  – Voted Perceptron
  – Boosting
- Loss functions

# Ensemble methods

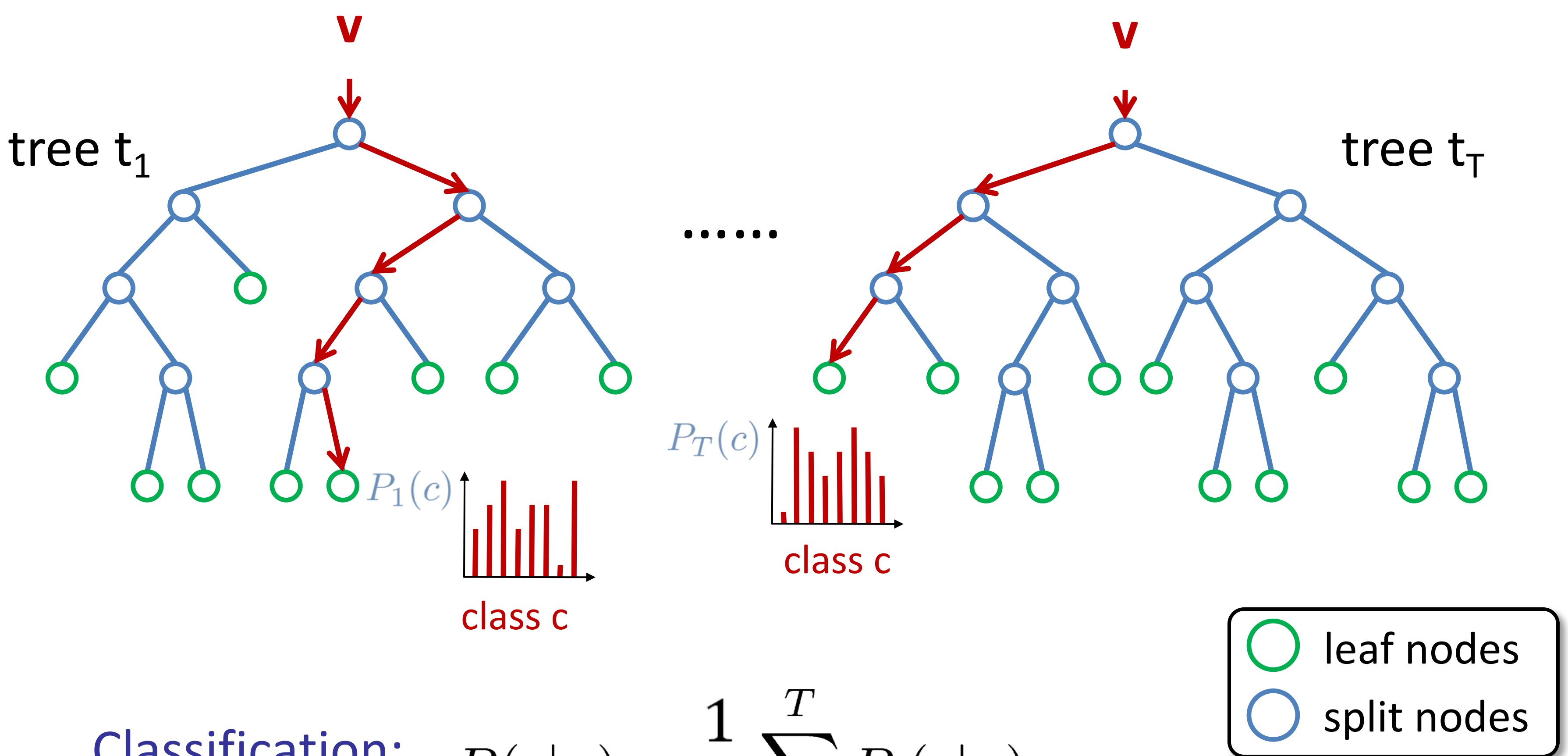
An ensemble classifier combines a set of weak “base” classifiers into a “strong” ensemble classifier.

- “boosted” performance
- more robust against overfitting

- Decision Forests, Random Forests [Breiman ‘01], Bagging
- Voted-Perceptron
- Boosting
- Learning with expert advice
- ….

# Decision Forests

- A forest is an ensemble of several decision trees




Classification: $$P(c|\mathbf{v}) = \frac{1}{T}\sum_{t=1}^{T} P_t(c|\mathbf{v})$$

Shotton, ICCV 09 Tutorial

# Learning a Forest

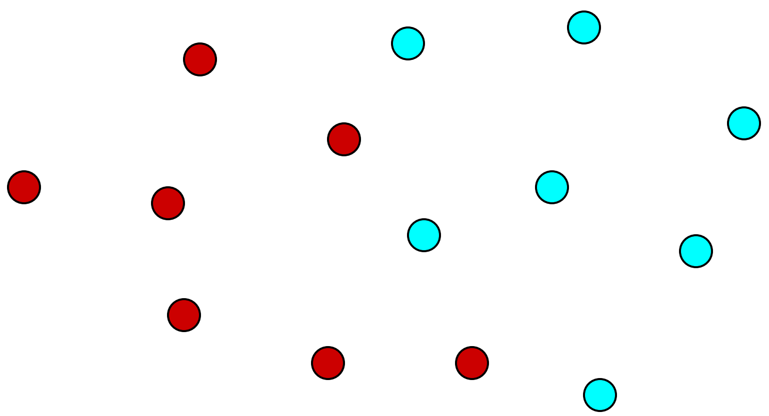
- Divide training examples into T subsets $S_t$
  - improves generalization
  - reduces memory requirements & training time
- Train each decision tree, *t,* on subset $S_t$
  - same decision tree learning as before
- Easy to parallelize

# Bagging

- Sample T subsets $S_t$ – i.i.d. sampling with replacement
  - improves generalization
  - reduces memory requirements & training time
- Train each decision tree, *t,* on subset $S_t$
  - same decision tree learning as before
- Easy to parallelize
- NOTE: Do not confuse a Decision Forest, nor even a Bagged Decision Forest with a Random Forest!
  - RF also uses randomness in choosing the tree splits!

# Perceptron: nonseparable data

What if data is not linearly separable?



In this case: almost linearly separable… how will the perceptron perform?

# Batch perceptron

Batch algorithm:

```
w = 0
while some (x_i ,y_i) is misclassified:
    w = w + y_i x_i
```

Nonseparable data: this algorithm will never converge. How can this be fixed?

Dream: somehow find the separator that misclassifies the fewest points… **but this is NP-hard** (in fact, eve NP-hard to approximately solve).

# Fixing the batch perceptron

Idea 1: only go through the data once, or a fixed number of times, K

```
w = 0
for k = 1 to K
    for i = 1 to m
        if (x_i, y_i) is misclassified:
            w = w + y_i x_i
```

At least this stops!

Problem: the final *w* might not be good.

Eg. right before terminating, the algorithm might perform an update on an outlier!

# Voted-perceptron

Idea 2: keep around intermediate hypotheses, and have them "vote" [Freund and Schapire, 1998]

```
n = 1
w_1 = 0
c_1 = 0
for k = 1 to K
    for i = 1 to m
        if (x_i ,y_i) is misclassified:
            w_{n+1} = w_n + y_i x_i
            c_{n+1} = 1
            n = n + 1
        else
            c_n = c_n + 1
```

At the end, a collection of linear separators $w_0$, $w_1$, $w_2$, …, along with survival times: $c_n$ = amount of time that $w_n$ survived.

# Voted-perceptron

Idea 2: keep around intermediate hypotheses, and have them "vote" [Freund and Schapire, 1998]

At the end, a collection of linear separators $w_0$, $w_1$, $w_2$, …, along with survival times: $c_n$ = amount of time that $w_n$ survived.

This $c_n$ is a good measure of the reliability (or confidence) of $w_n$.

To classify a test point x, use a weighted majority vote:

$$\mathrm{sgn}\left\{\sum_{n=0}^{N} c_n \; \mathrm{sgn}(w_n \cdot x)\right\}$$

# Voted-perceptron

Problem: may need to keep around a lot of $w_n$ vectors.

Solutions:

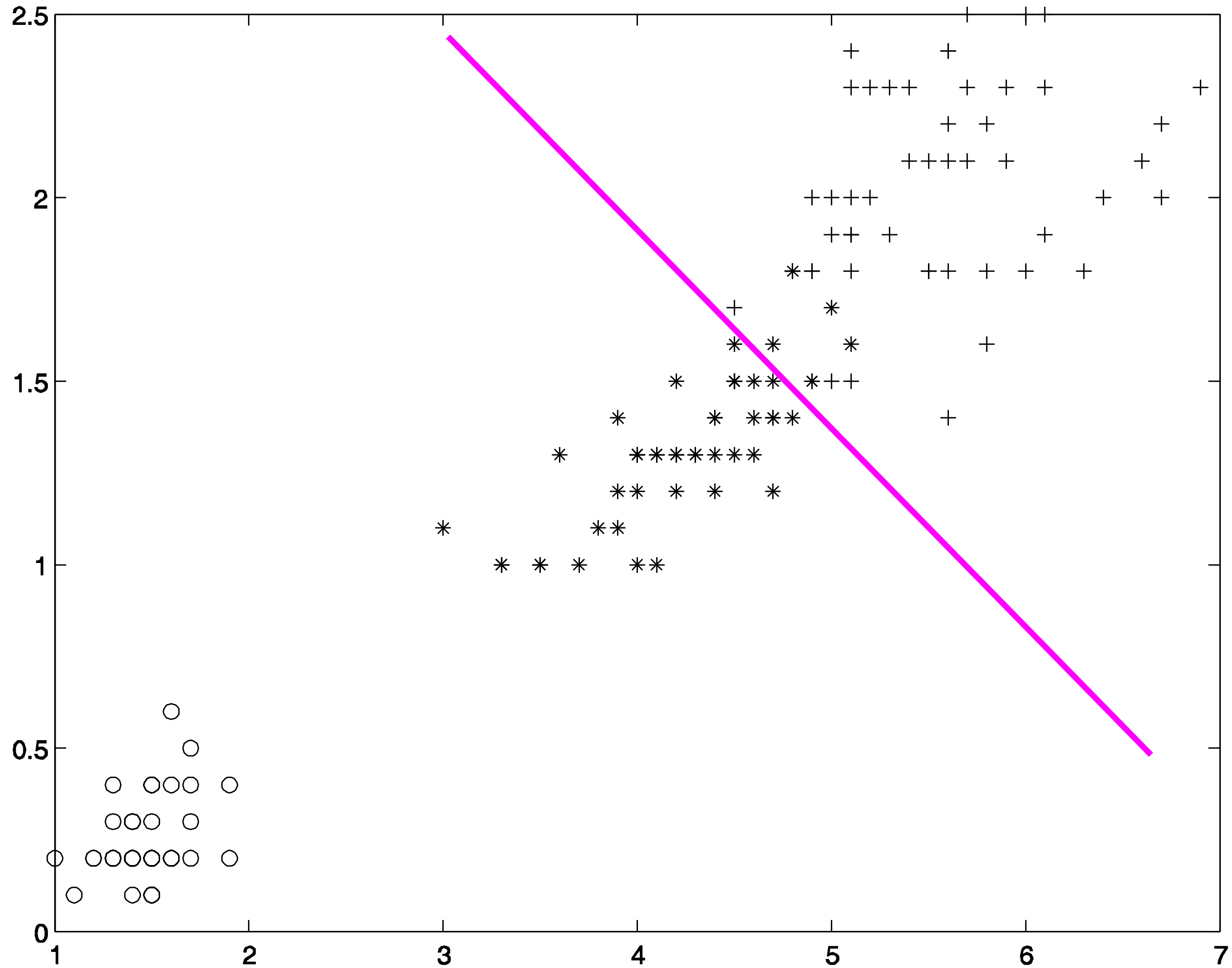
(i) Find "representatives" among the w vectors.

(ii) Alternative prediction rule: average vote:

$$\mathrm{sgn}\left\{\sum_{n=0}^{N} c_n \, (w_n \cdot x)\right\} = \mathrm{sgn}\left\{\left(\sum_{n=0}^{N} c_n \, w_n\right) \cdot x\right\}$$

Just keep track of a running average, $w_{avg}$

# IRIS: features 3 and 4; goal: separate + from o/x



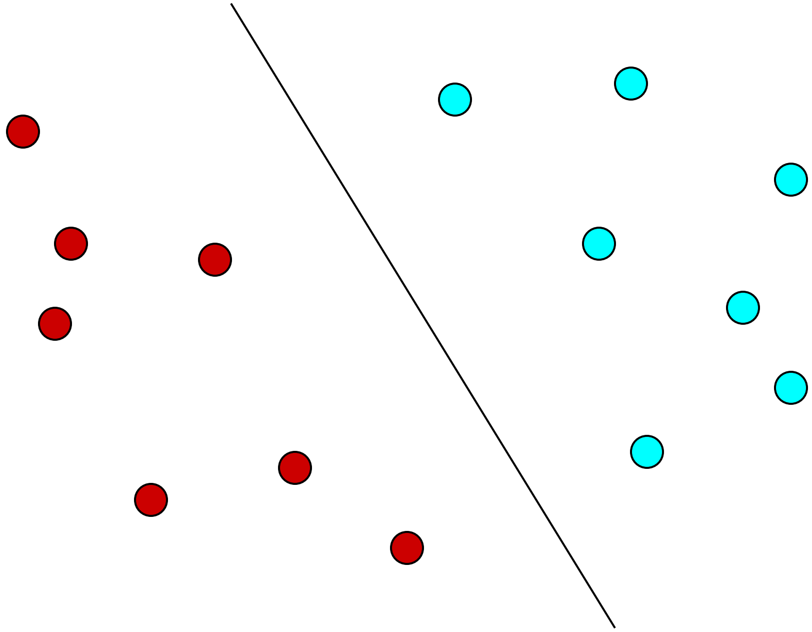

100 rounds, 1595 updates (5 errors)

Final hypothesis (makes 5 errors with voting, 6 with averaging)

# Interesting questions

Modify the (voted) perceptron algorithm to:

[1] Find a linear separator with large <u>margin</u>



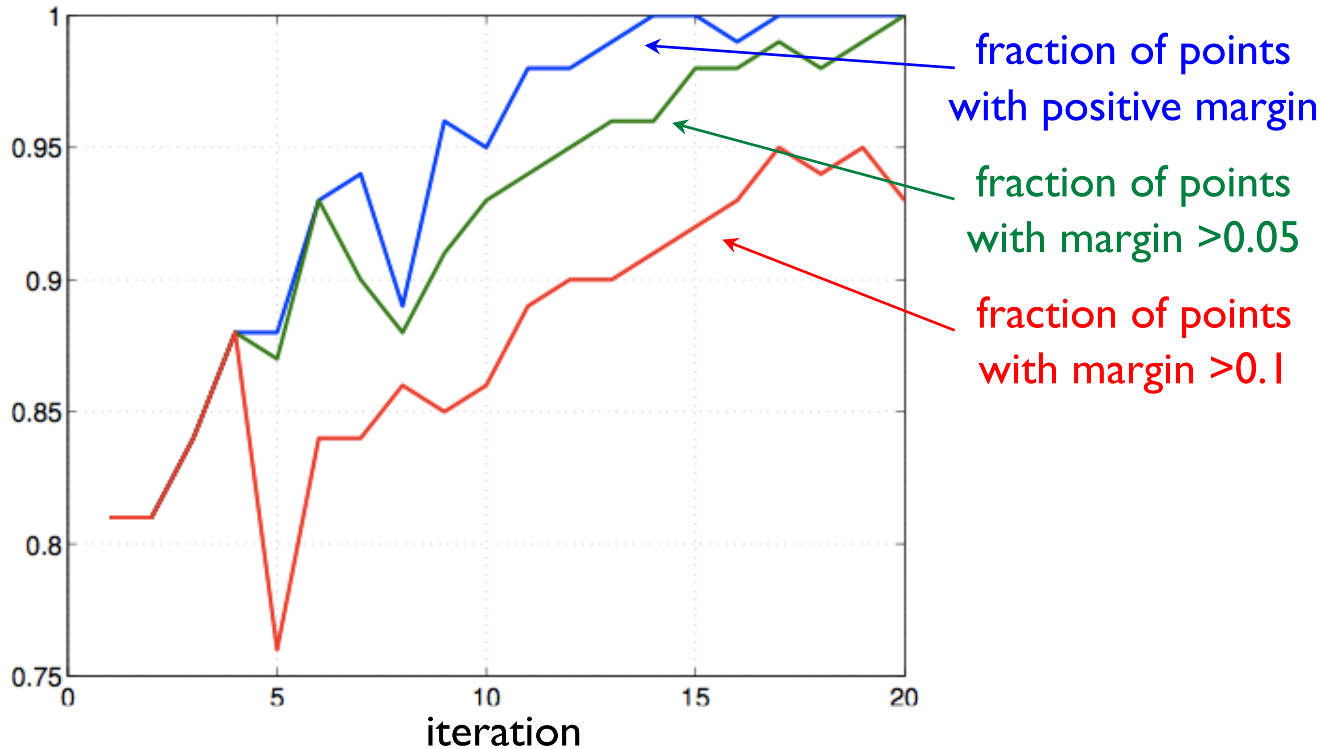
[2] “Give up” on troublesome points after a while

# Voting margin and generalization

- If we can obtain a large (positive) voting margin

$$\gamma_t = \frac{y_t h_m(\mathbf{x_t})}{\sum_{j=1}^m c_j} = \frac{y_t \sum_{j=1}^m c_j (\mathbf{w_j} \cdot \mathbf{x_t})}{c_1 + \cdots + c_m}$$

across the training examples, we will have better generalization guarantees (discussed later)



# Boosting

- A simple algorithm for learning robust ensemble classifiers
  - Freund & Shapire, 1995
  - Friedman, Hastie, Tibshhirani, 1998

- Easy to implement, no external optimization tools needed.

# Boosting

- Defines a classifier using an additive model:

$$F(x) = \alpha_1 f_1(x) + \alpha_2 f_2(x) + \alpha_3 f_3(x) + \ldots$$

Strong classifier

Weak classifier

Weight

Data point

# Boosting

- Defines a classifier using an additive model:

$$F(x) = \alpha_1 f_1(x) + \alpha_2 f_2(x) + \alpha_3 f_3(x) + \ldots$$

Strong classifier

Data point

Weight

Weak classifier

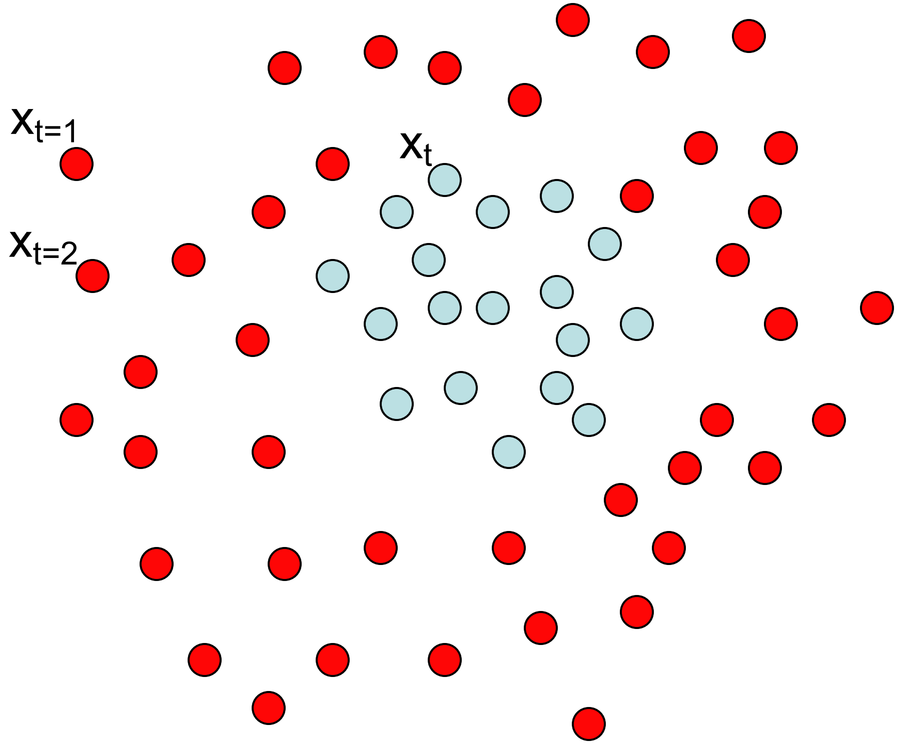
- We need to define a family of weak classifiers

$$f_k(x)$$

  - E.g. linear classifiers, decision trees, or even decision stumps (threshold on one axis-parallel dimension)

# Boosting

- Run sequentially on a batch of n data points




Each data point has a class label:

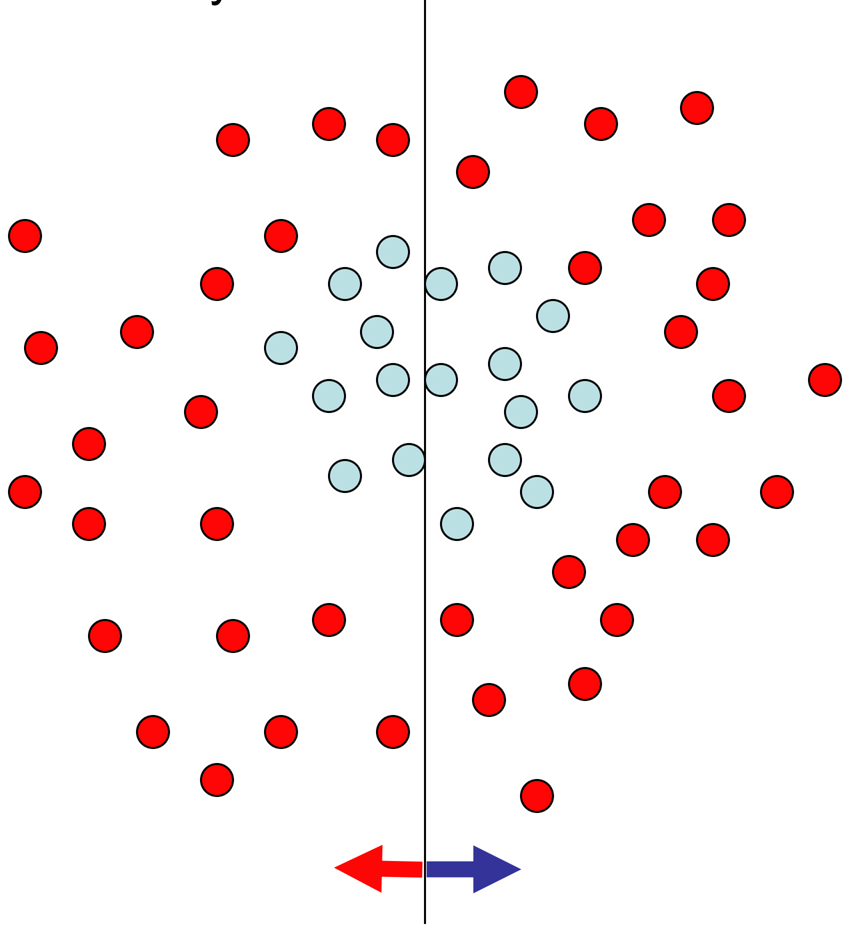
$$y_t = \begin{cases} +1 \; (\;) \\ -1 \; (\;) \end{cases}$$

and a weight, $w_t$.

- we initialize all $w_t = 1$

# Example using linear separators

Weak learners from the family of lines



Each data point has

a class label:

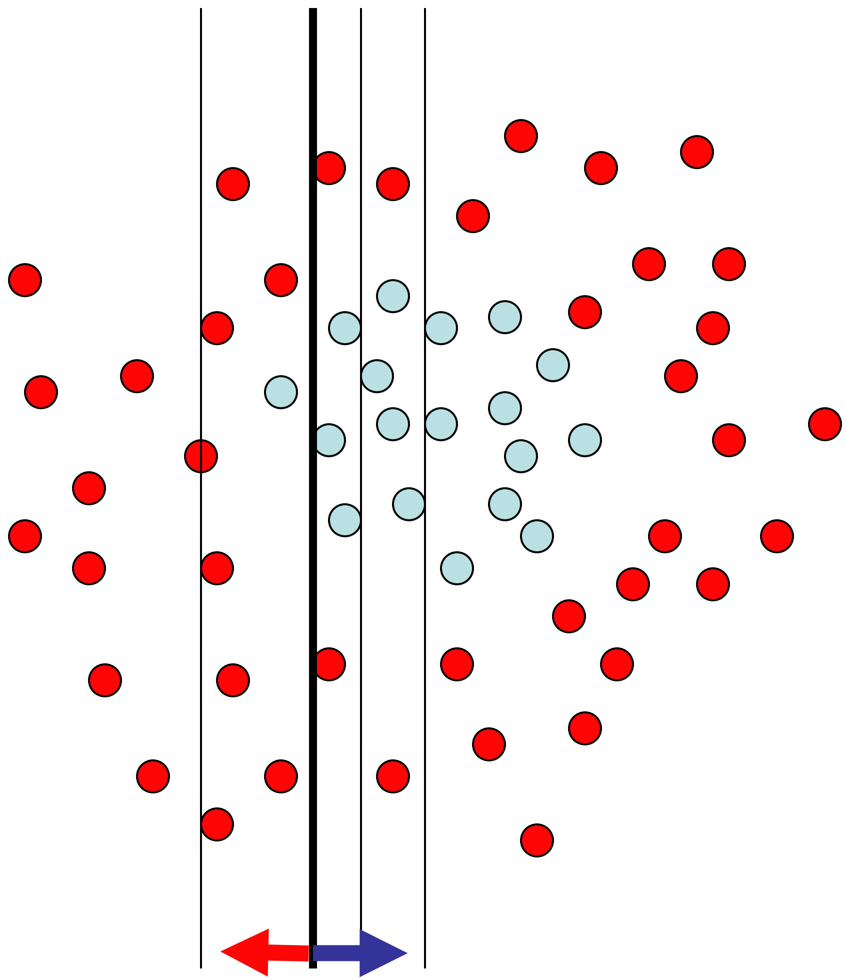
$$y_t = \begin{cases} +1 & (\text{red}) \\ -1 & (\text{blue}) \end{cases}$$

and a weight:

$w_t = 1$

This linear separator has error rate 50%

# Example using linear separators



Each data point has

a class label:

$$y_t = \begin{cases} +1 \ (\bullet) \\ -1 \ (\circ) \end{cases}$$

and a weight:

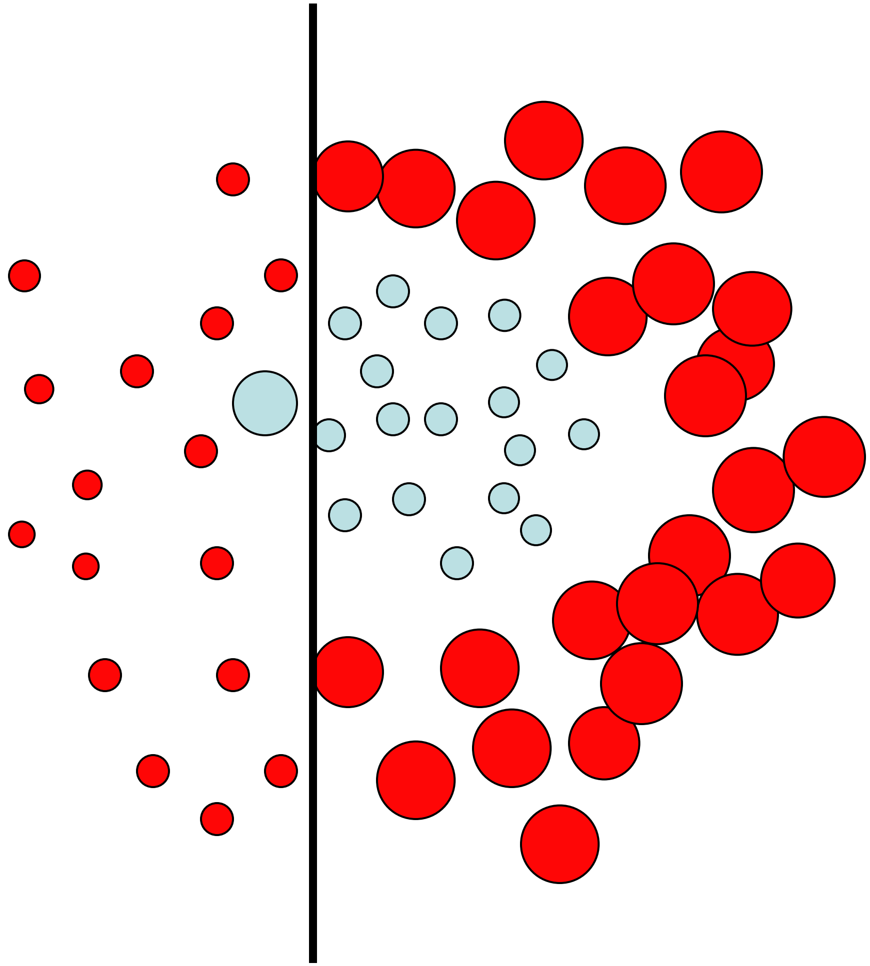
$w_t = 1$

This one seems to be the best, call it $f_1$

This is a ‘**weak classifier**’ : Its error rate is slightly less than 50%.

Torralba, ICCV 05 Short Course

# Example using linear separators



Each data point $x_i$ has a class label:

$$y_i = \begin{cases} +1 & ( \; ) \\ -1 & ( \; ) \end{cases}$$

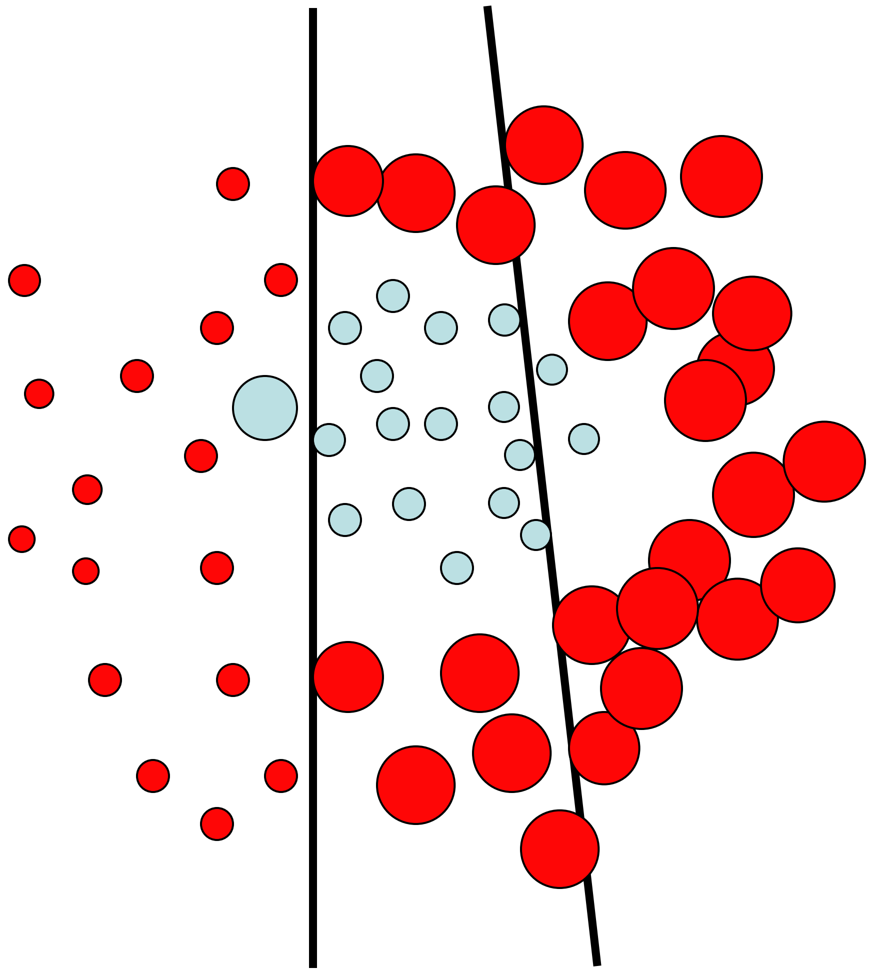
**We update the weights:**

$$w_t \leftarrow w_t \exp\{-y_t f_m(x_t)\}$$

- Re-weight the points such that the previous weak classifier now has 50% error
- Iterate: find a weak classifier for this new problem

Torralba, ICCV 05 Short Course

# Example using linear separators



Each data point $x_i$ has a class label:

$$y_i = \begin{cases} +1 \ (\bullet) \\ -1 \ (\circ) \end{cases}$$

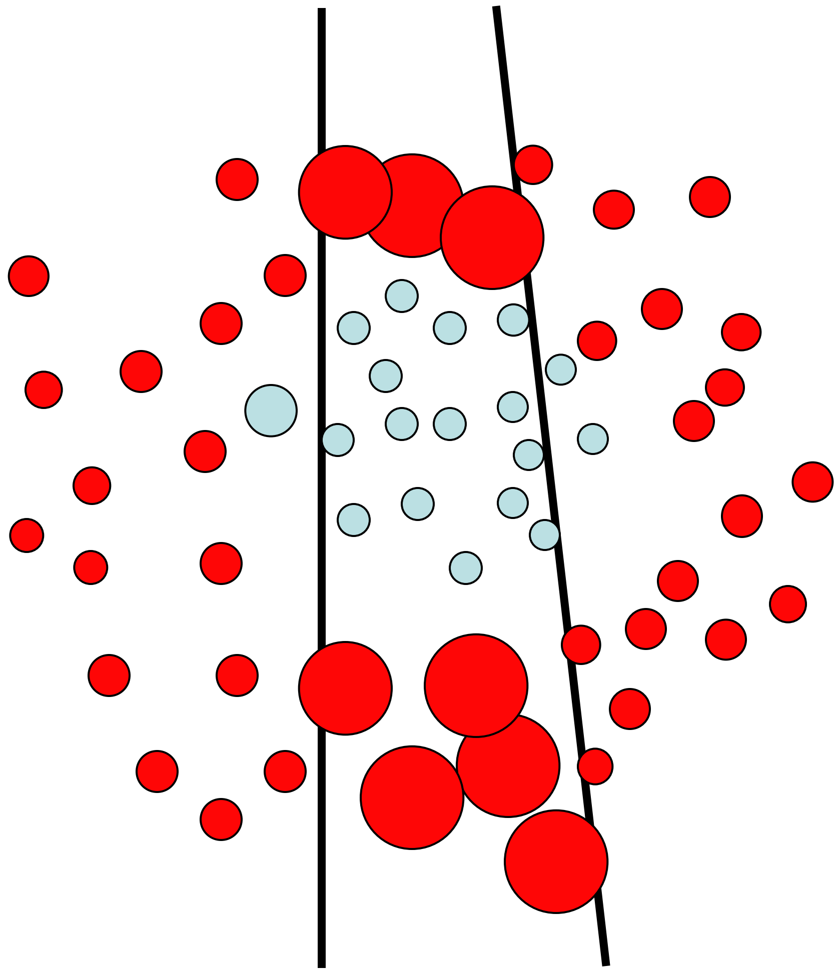
**We update the weights:**

$$w_t \leftarrow w_t \exp\{-y_t f_m(x_t)\}$$

- Re-weight the points such that the previous weak classifier now has 50% error
- Iterate: find a weak classifier for this new problem

Torralba, ICCV 05 Short Course

# Example using linear separators



Each data point $x_i$ has a class label:

$$y_i = \begin{cases} +1 \; (\;) \\ -1 \; (\;) \end{cases}$$

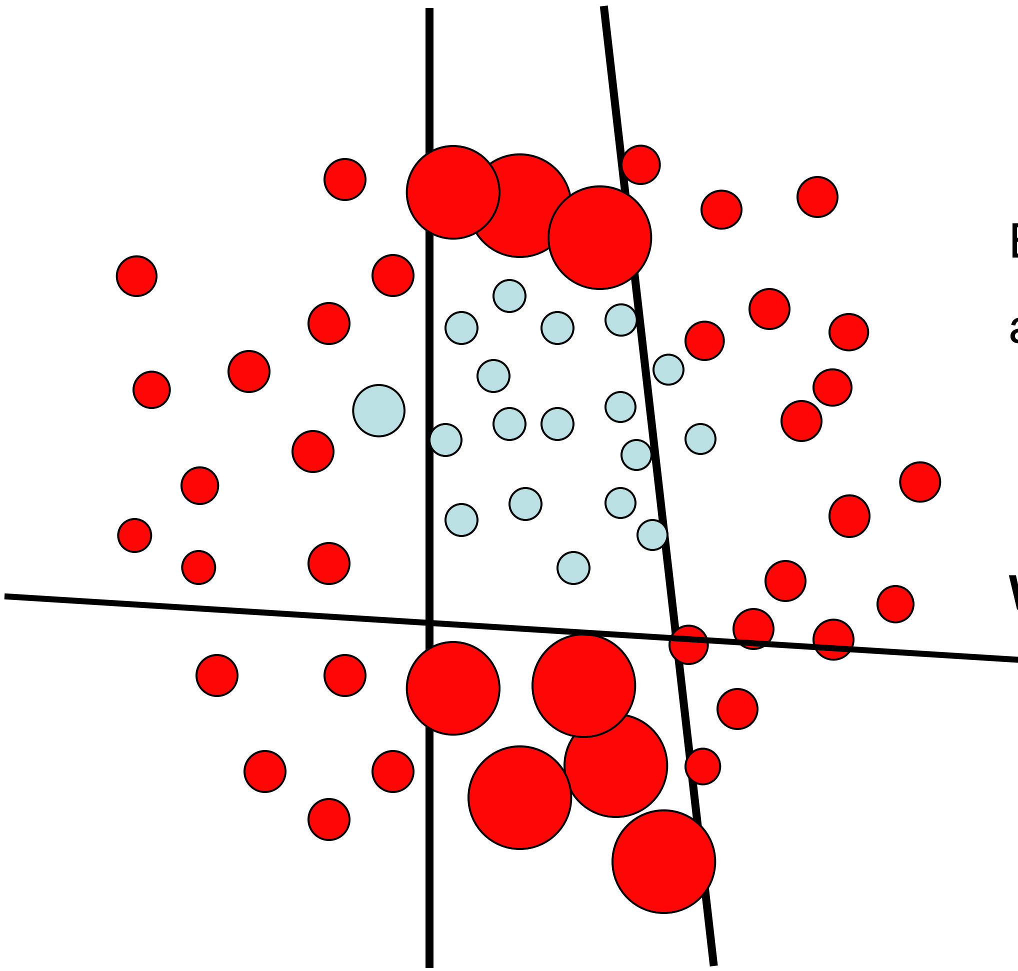
**We update the weights:**

$$w_t \leftarrow w_t \exp\{-y_t f_m(x_t)\}$$

- Re-weight the points such that the previous weak classifier now has 50% error
- Iterate: find a weak classifier for this new problem

Torralba, ICCV 05 Short Course

# Example using linear separators



Each data point $x_i$ has a class label:

$$y_i = \begin{cases} +1 & (\text{red}) \\ -1 & (\text{blue}) \end{cases}$$

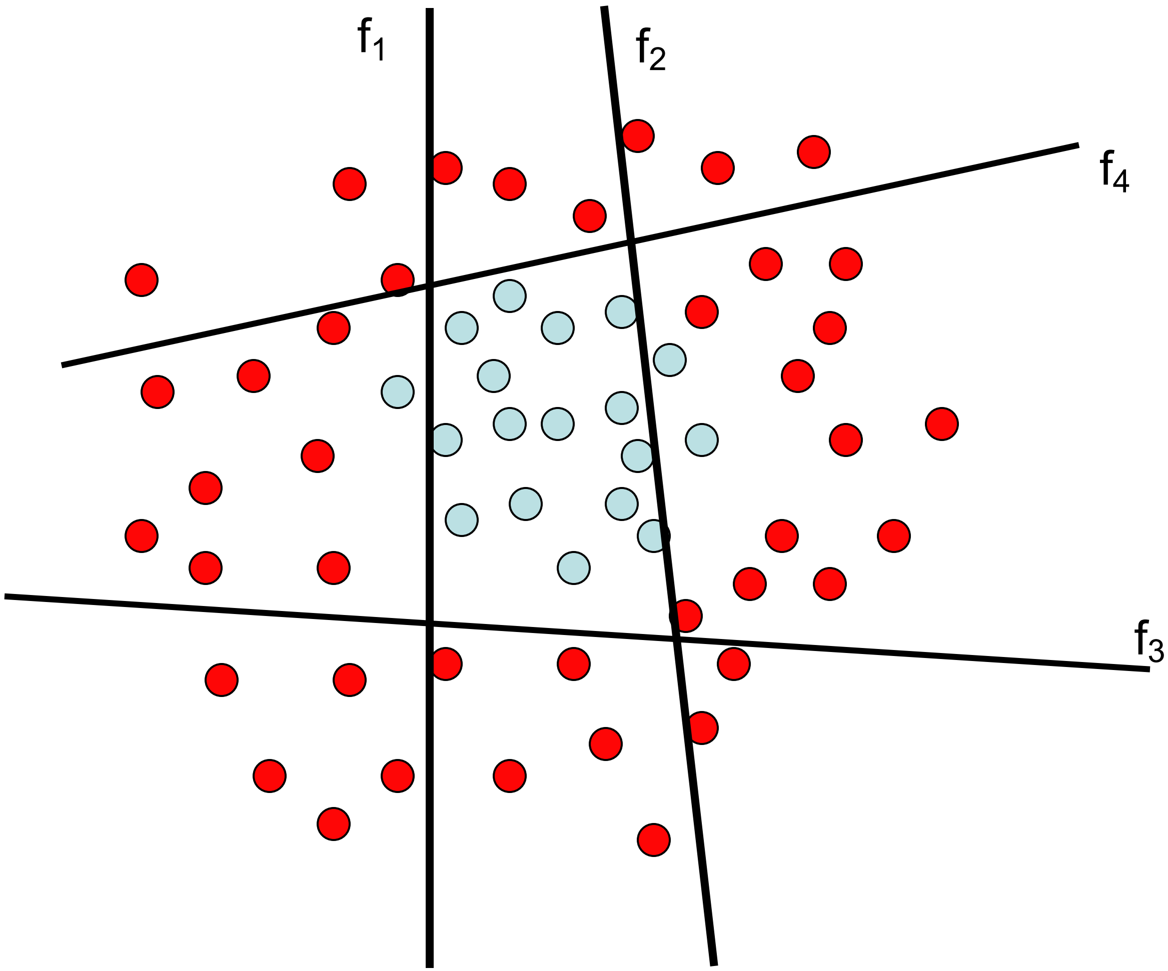
**We update the weights:**

$$w_t \leftarrow w_t \exp\{-y_t f_m(x_t)\}$$

- Re-weight the points such that the previous weak classifier now has 50% error
- Iterate: find a weak classifier for this new problem

Torralba, ICCV 05 Short Course

# Example using linear separators




The strong (non-linear) ensemble classifier is built as a weighted combination of all the weak (linear) classifiers.

Torralba, ICCV 05 Short Course

# Boosting

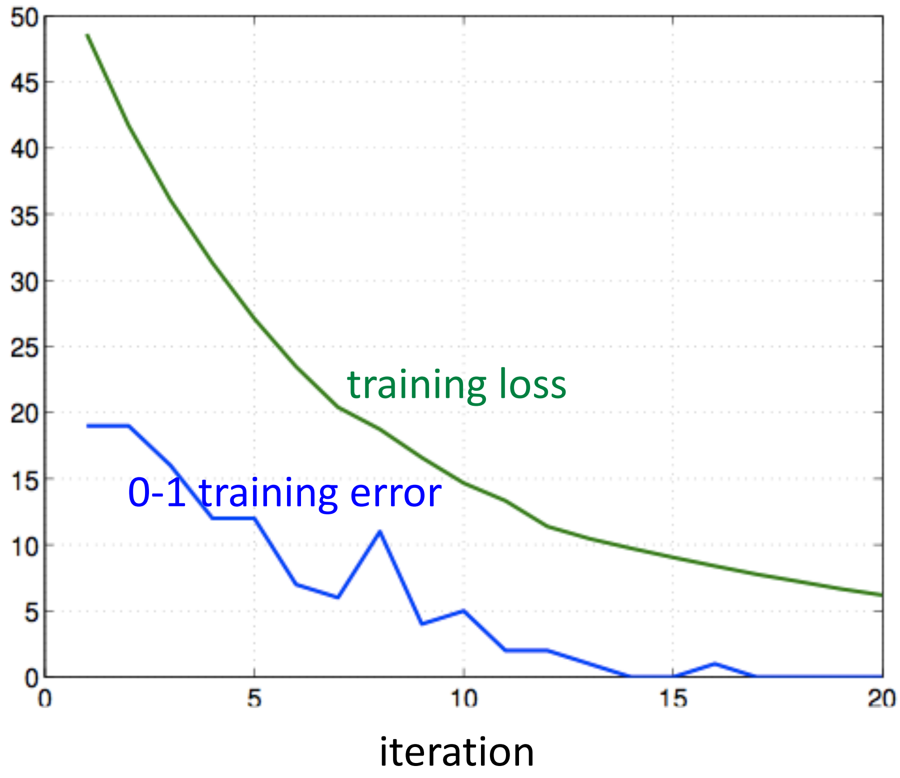
- AdaBoost (Freund and Shapire, 1995)
- Real AdaBoost (Friedman et al, 1998)
- LogitBoost (Friedman et al, 1998)
- Gentle AdaBoost (Friedman et al, 1998)
- BrownBoosting (Freund, 2000)
- FloatBoost (Li et al, 2002)
- …

Mostly differ in choice of loss function and how it is minimized.

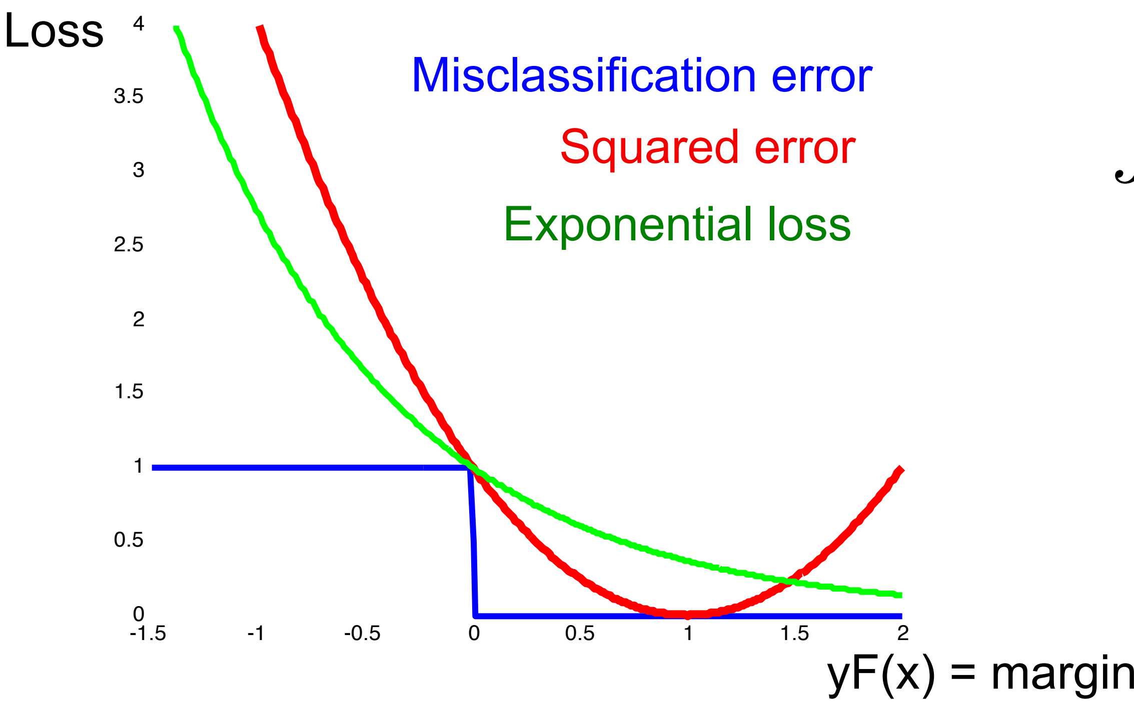
Torralba, ICCV 05 Short Course

# Loss functions: motivation

- Want a smooth upper bound on 0-1 training error.



# Loss functions




Squared error

$$J = \sum_{t=1}^{N} [y_t - F(x_t)]^2$$

Exponential loss

$$J = \sum_{t=1}^{N} e^{-y_t F(x_t)}$$

Torralba, ICCV 05 Short Course

# Boosting

Sequential procedure. At each step we add

$$F(x) \leftarrow F(x) + f_m(x)$$

to minimize the residual loss

$$(\phi_m) = \arg\min_{\phi} \sum_{t=1}^{N} J\left(y_i, F(x_t) + f(x_t; \phi)\right)$$

**Parameters of weak classifier**

**Desired output**

**input**

For more details: Friedman, Hastie, Tibshirani. "Additive Logistic Regression: a Statistical View of Boosting" (1998)

# How to set the ensemble weights?

- Prediction on a new data point *x* is typically of the form:

$$F(x) = \sum_{m=1}^{k} \alpha_m f_m(x)$$

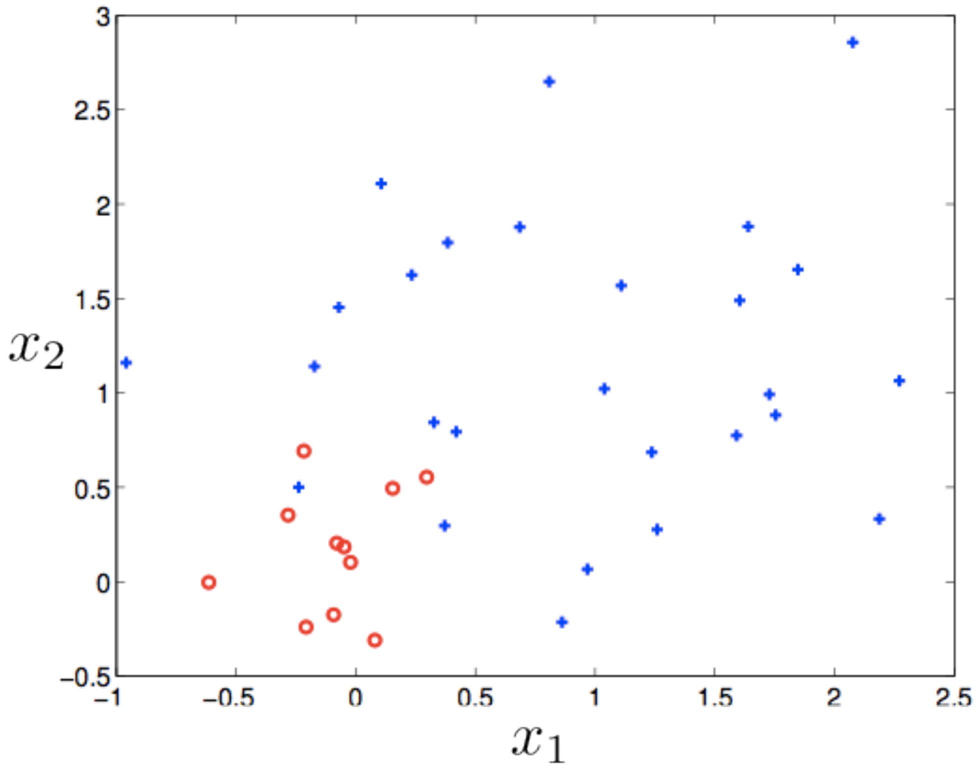
- How to set the $\alpha_m$ values?
- Depends on the algorithm (due to different loss functions, etc.) E.g. in AdaBoost:

$$\alpha_m = \frac{1}{2} \ln \frac{1 - \epsilon_m}{\epsilon_m}$$

- Where $\epsilon_m$ is the training error of $f_m$ on the (currently) weighted data set.

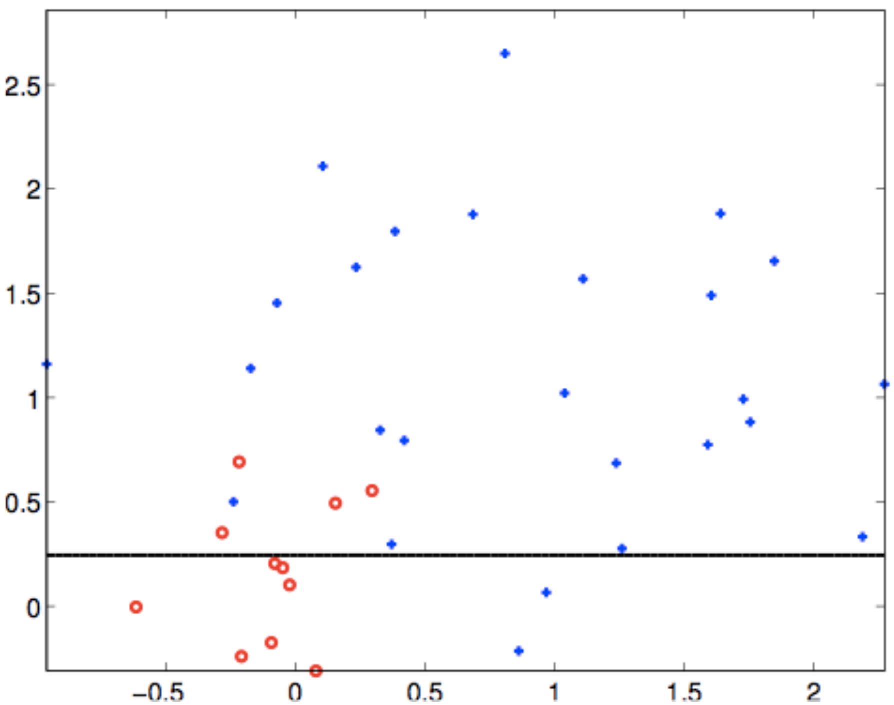
# Boosting example

Logistic loss $\text{Loss}(z) = \log(1 + \exp(-z))$

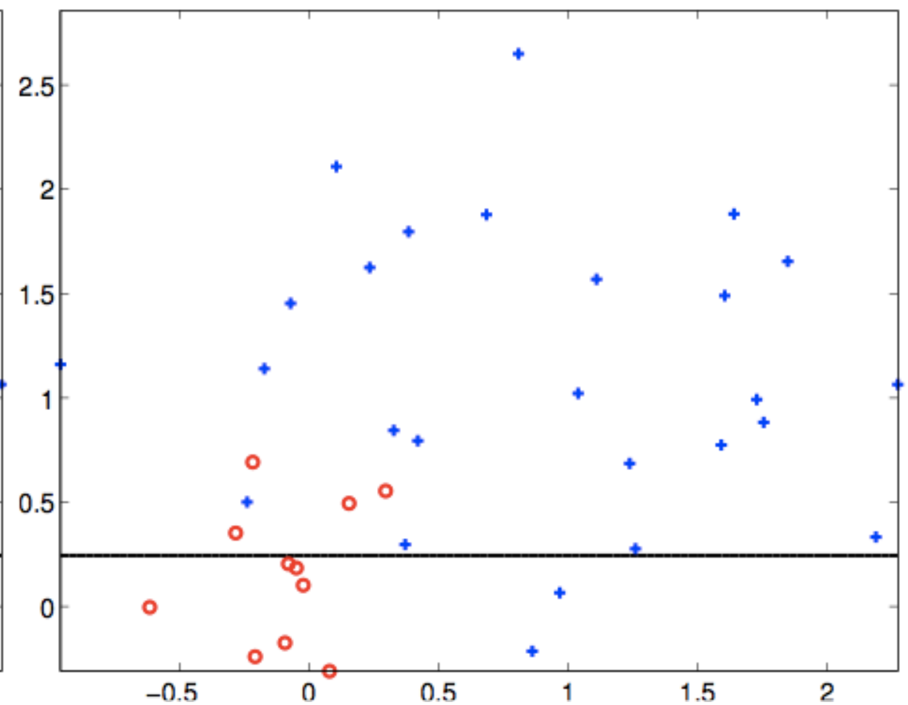


# Boosting example

$h(\underline{x};\hat{\underline{\theta}}_1)$

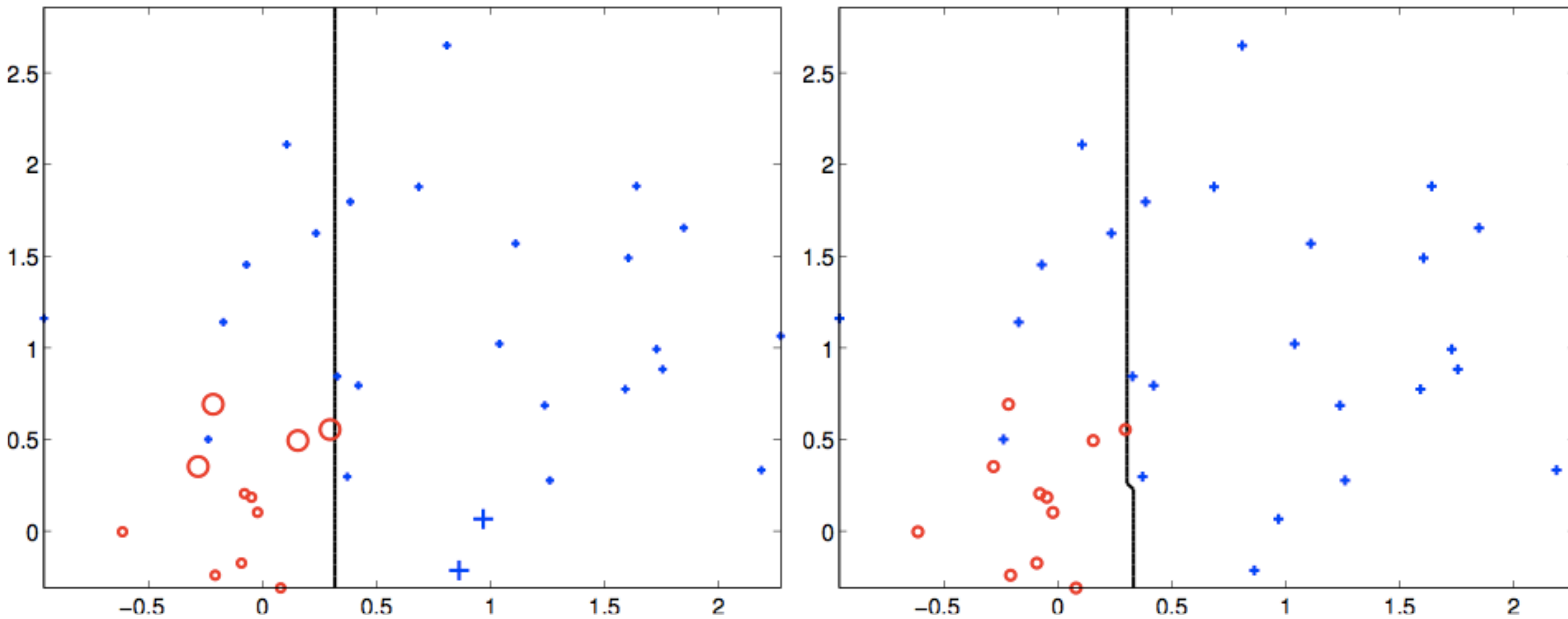



$h_1(\underline{x}) = \hat{\alpha}_1 h(\underline{x};\hat{\underline{\theta}}_1)$



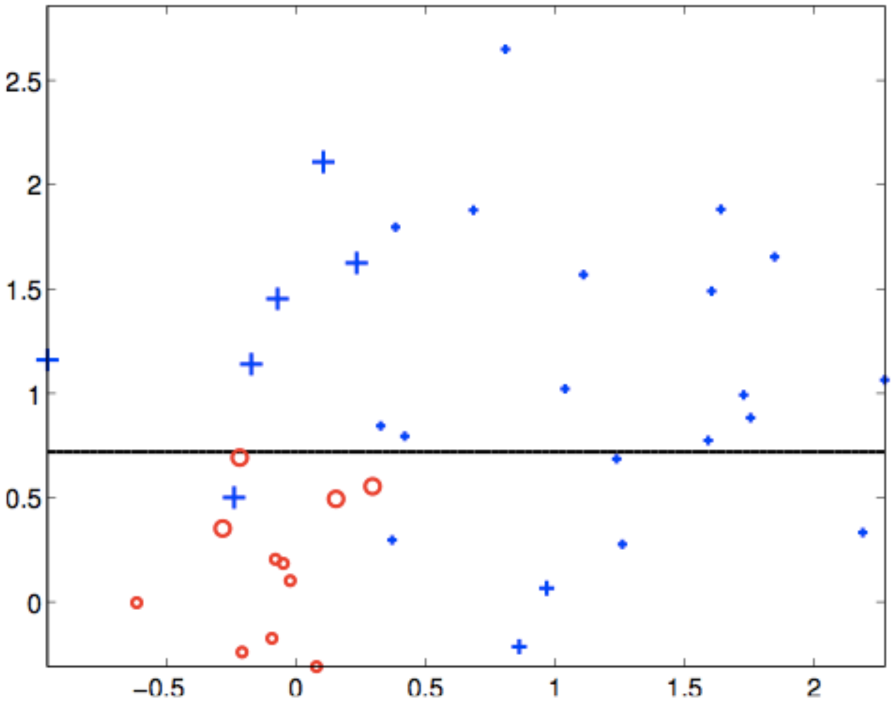
# Boosting example

$$h(\underline{x}; \hat{\underline{\theta}}_2) \qquad h_2(\underline{x}) = \hat{\alpha}_1 h(\underline{x}; \hat{\underline{\theta}}_1) + \hat{\alpha}_2 h(\underline{x}; \hat{\underline{\theta}}_2)$$

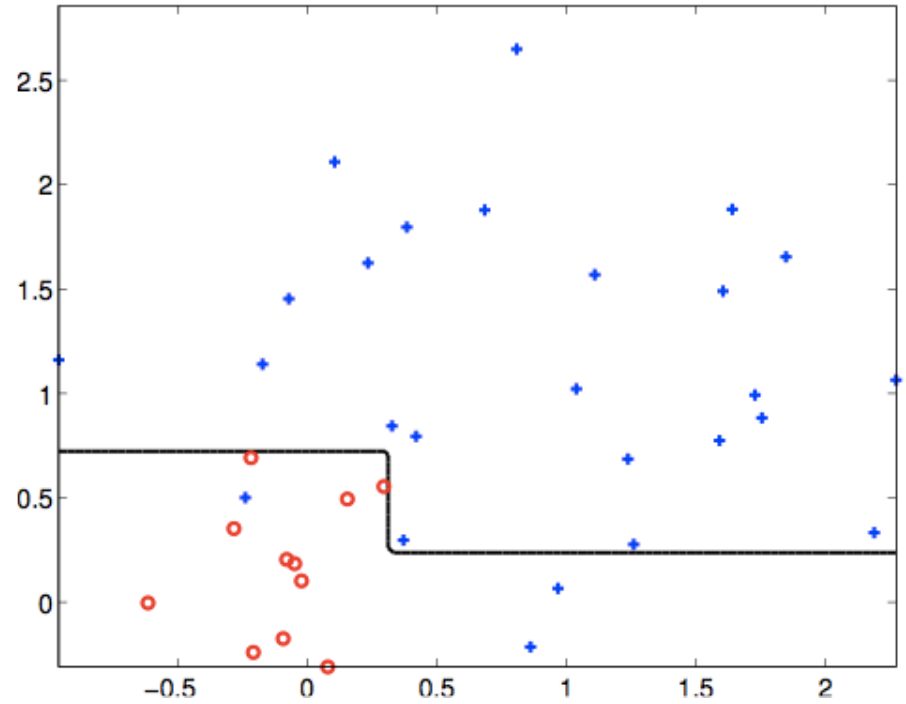


# Boosting example

$$h(\underline{x}; \underline{\hat{\theta}}_3)$$

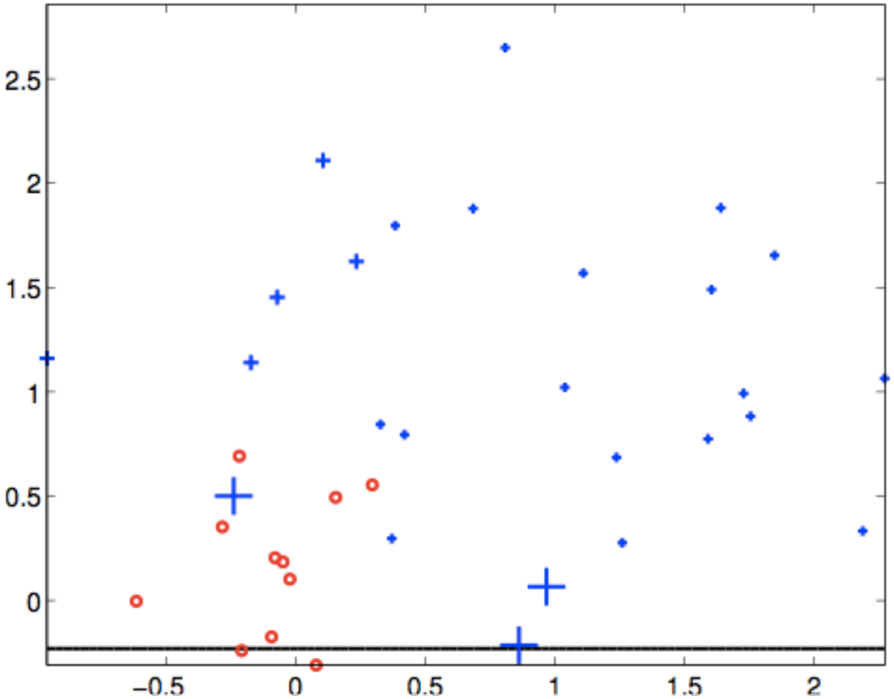



$$h_3(\underline{x}) = \hat{\alpha}_1 h(\underline{x}; \underline{\hat{\theta}}_1) + \cdots + \hat{\alpha}_3 h(\underline{x}; \underline{\hat{\theta}}_3)$$

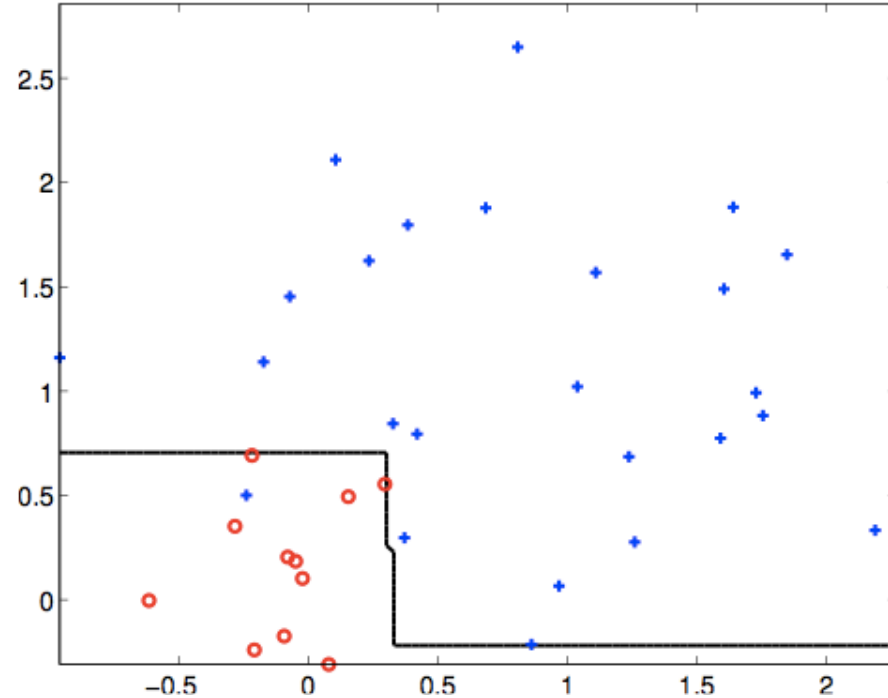


# Boosting example

$h(\underline{x};\hat{\theta}_4)$




$h_4(\underline{x}) = \hat{\alpha}_1 h(\underline{x};\hat{\theta}_1) + \cdots + \hat{\alpha}_4 h(\underline{x};\hat{\theta}_4)$



# Understanding boosting

- There are four different kinds of "error" in boosting:
  - weighted error that the base learner achieves at each iteration
  - weighted error of the base learner relative to just updated weights (i.e., trying the same base learner again)
  - training error of the ensemble as a function of the number of boosting iterations
  - generalization error of the ensemble