Machine Learning

CSCI 4622

Fall 2019

Prof. Claire Monteleoni

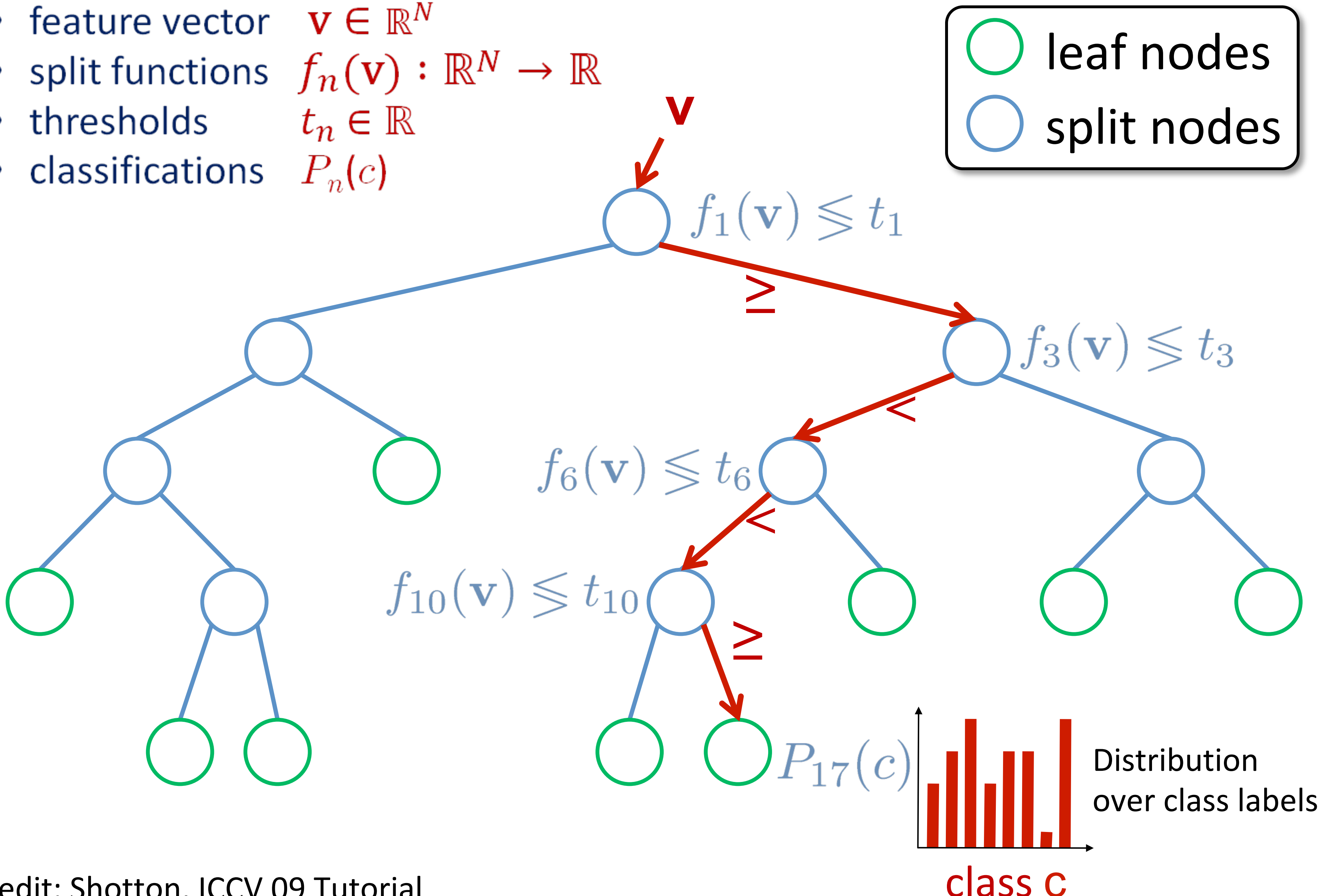
Today: Lecture 10

- Introduction to Ensemble Methods
  - Random Forests
  - Voted Perceptron
  - Boosting (if time)

With slide credits: S. Dasgupta, T. Jaakkola, J. Shotton, C. Tan, and A. Torralba

Review: (Binary) Decision Trees

- feature vector $\mathbf{v} \in \mathbb{R}^N$
- split functions $f_n(\mathbf{v}) : \mathbb{R}^N \rightarrow \mathbb{R}$
- thresholds $t_n \in \mathbb{R}$
- classifications $P_n(c)$



Credit: Shotton, ICCV 09 Tutorial

Speeding up Decision Tree Learning

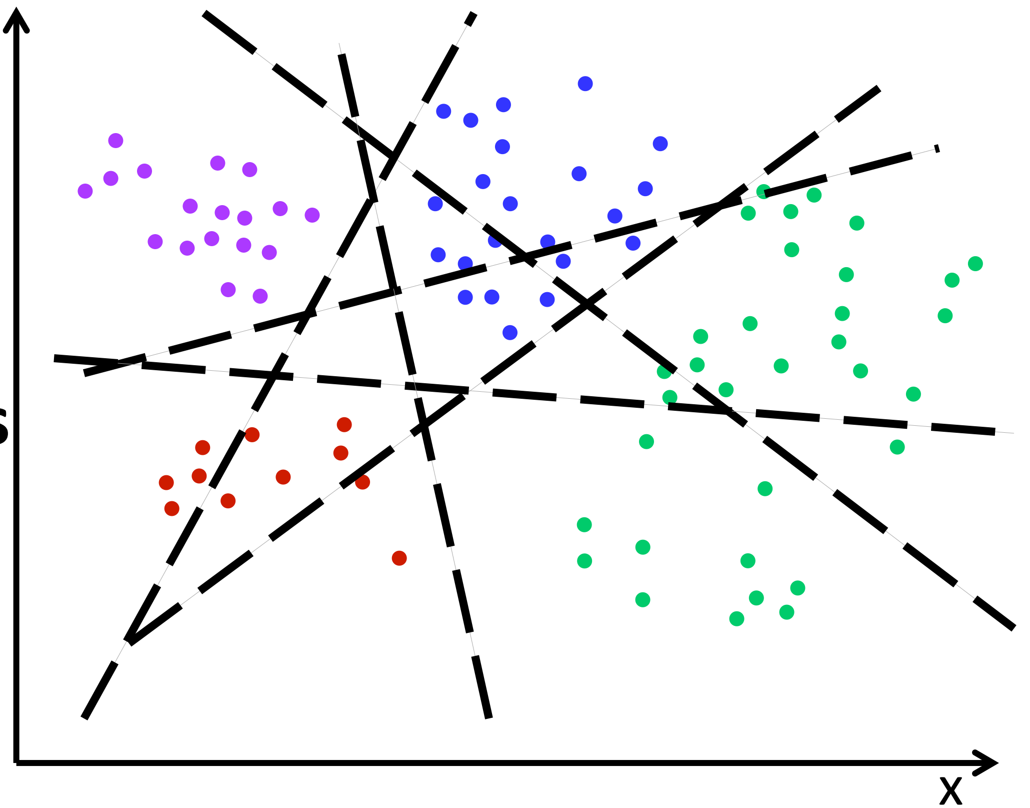
It is cumbersome to test all possible splits

Try several random splits

Keep the split that best separates data

- Reduces uncertainty

Recurse



Animation: Shotton, ICCV 09 Tutorial

Speeding up Decision Tree Learning

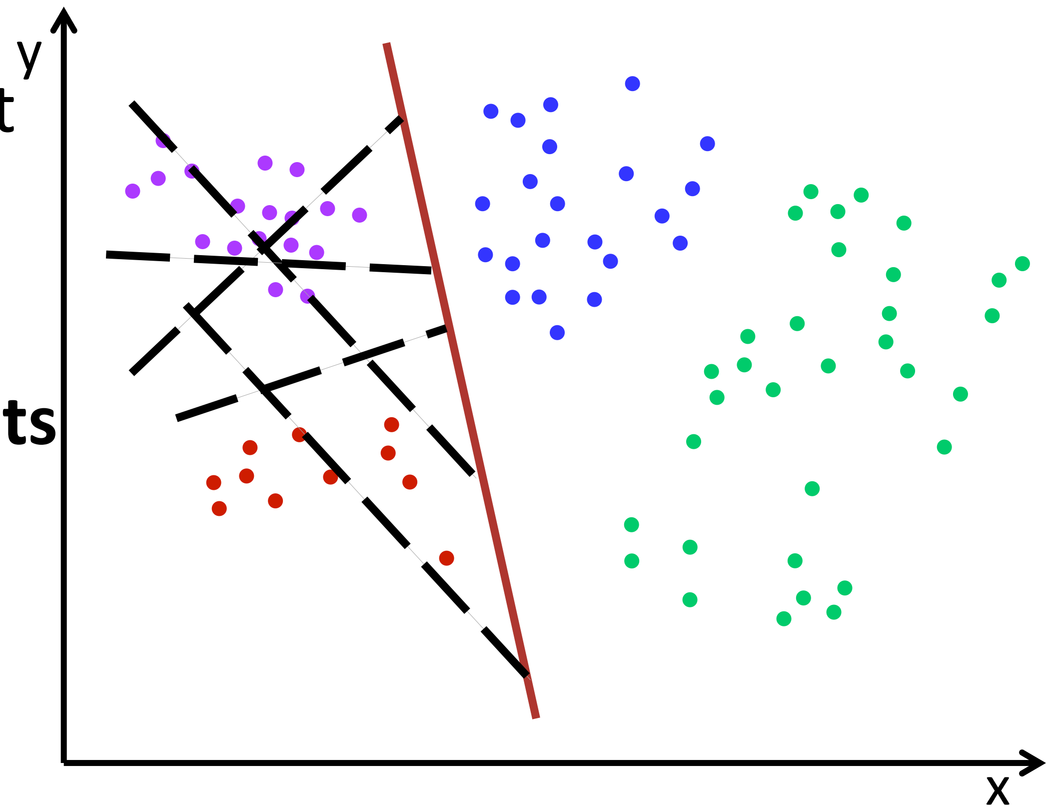
It is cumbersome to test all possible splits

Try several random splits

Keep the split that best separates data

- Reduces uncertainty

Recurse



Shotton, ICCV 09 Tutorial

Speeding up Decision Tree Learning

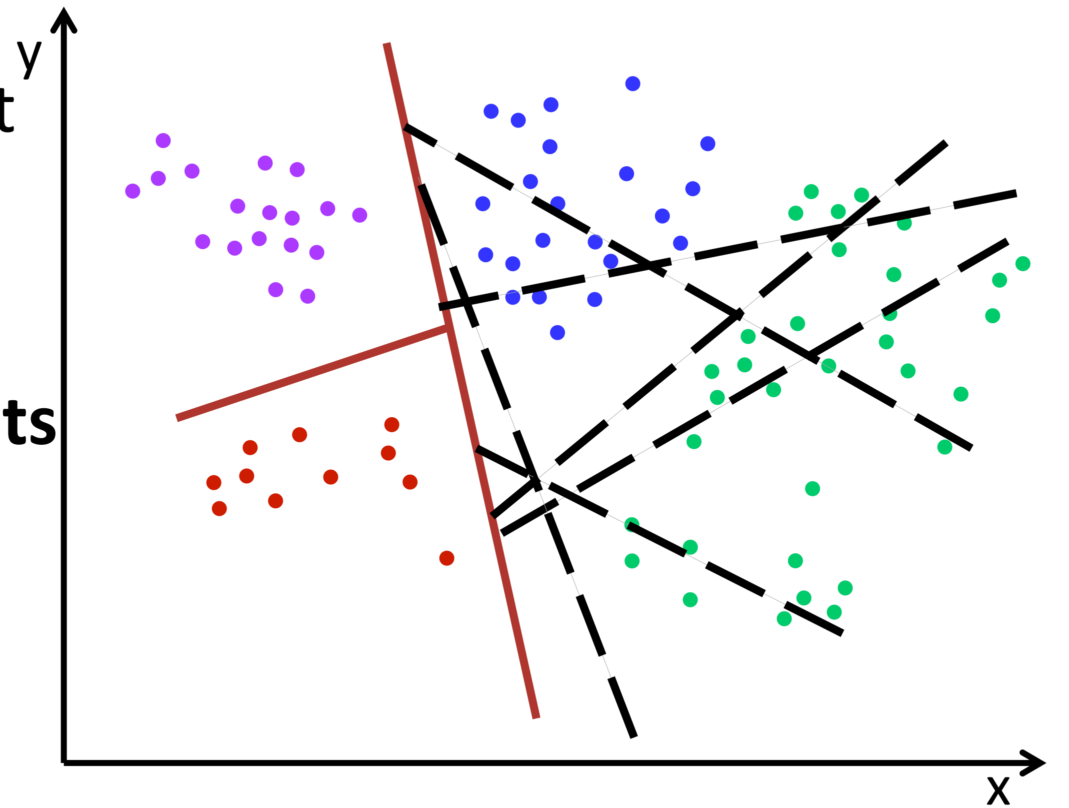
It is cumbersome to test all possible splits

Try several random splits

Keep the split that best separates data

- Reduces uncertainty

Recurse



Shotton, ICCV 09 Tutorial

Speeding up Decision Tree Learning

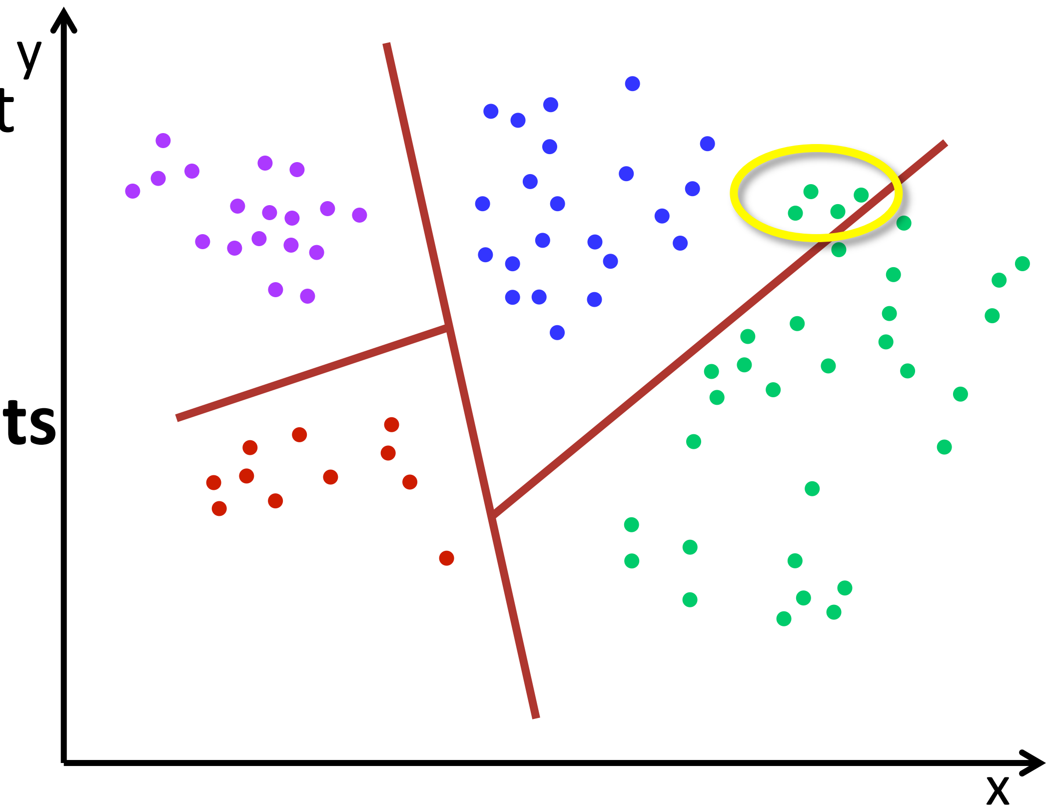
It is cumbersome to test all possible splits

Try several random splits

Keep the split that best separates data

- Reduces uncertainty

Recurse



Random Decision Tree

Shotton, ICCV 09 Tutorial

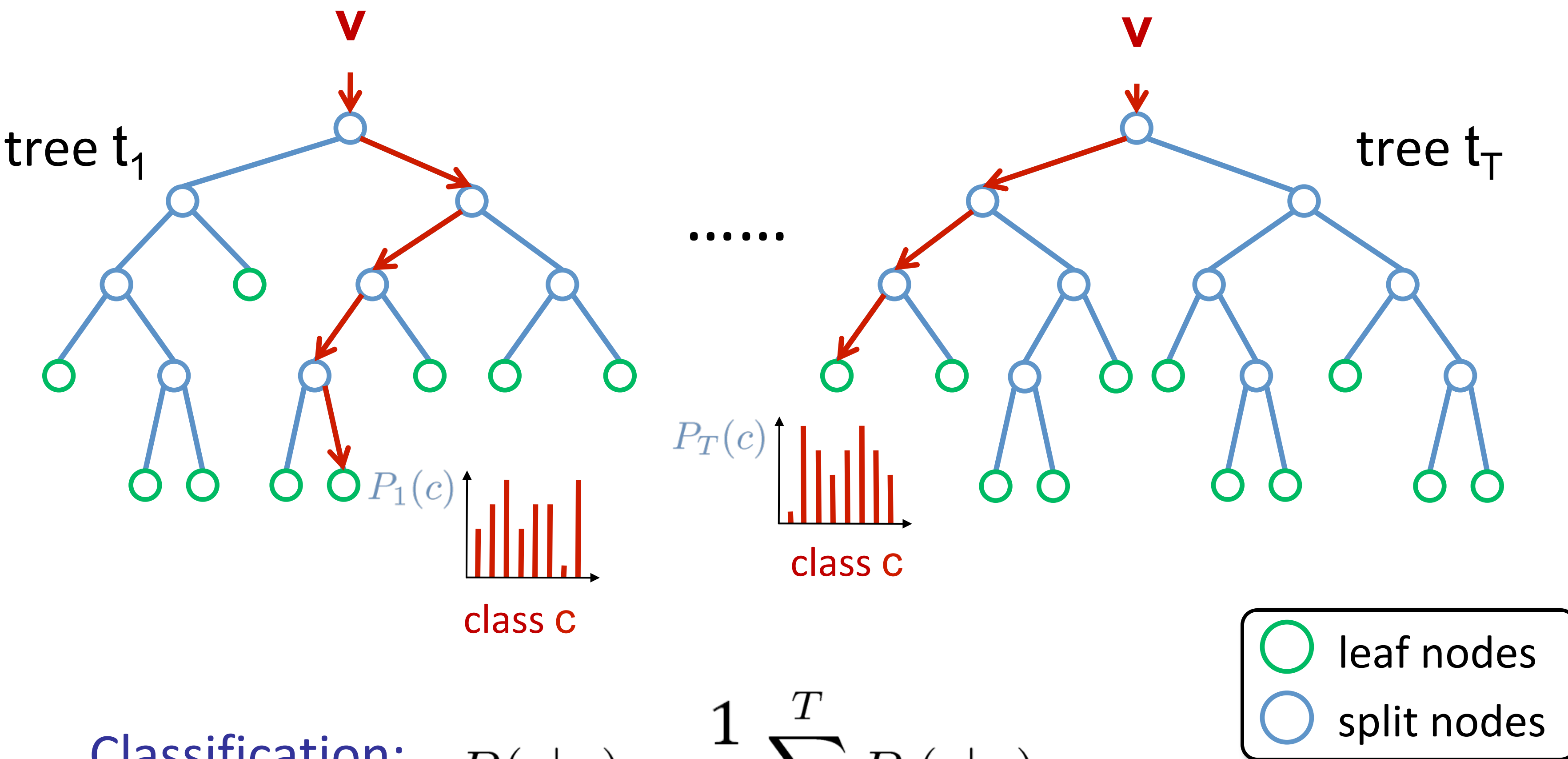
Random Decision Trees

- How many features and thresholds to try?
  - just one: "extremely randomized" [Geurts *et al.* 06]
  - few: fast training, may under-fit
  - many: slower training, may over-fit

- When to stop growing the tree?
  - maximum depth
  - minimum entropy gain
  - threshold changes in class distribution
  - pruning

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

Ensemble methods

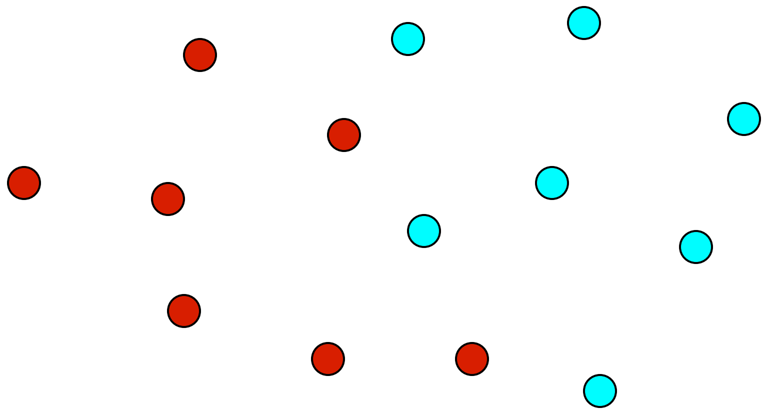
An ensemble classifier combines a set of weak “base” classifiers into a “strong” ensemble classifier.

- “boosted” performance
- more robust against overfitting

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

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, even 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:

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

Voted-perceptron

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

Solutions:

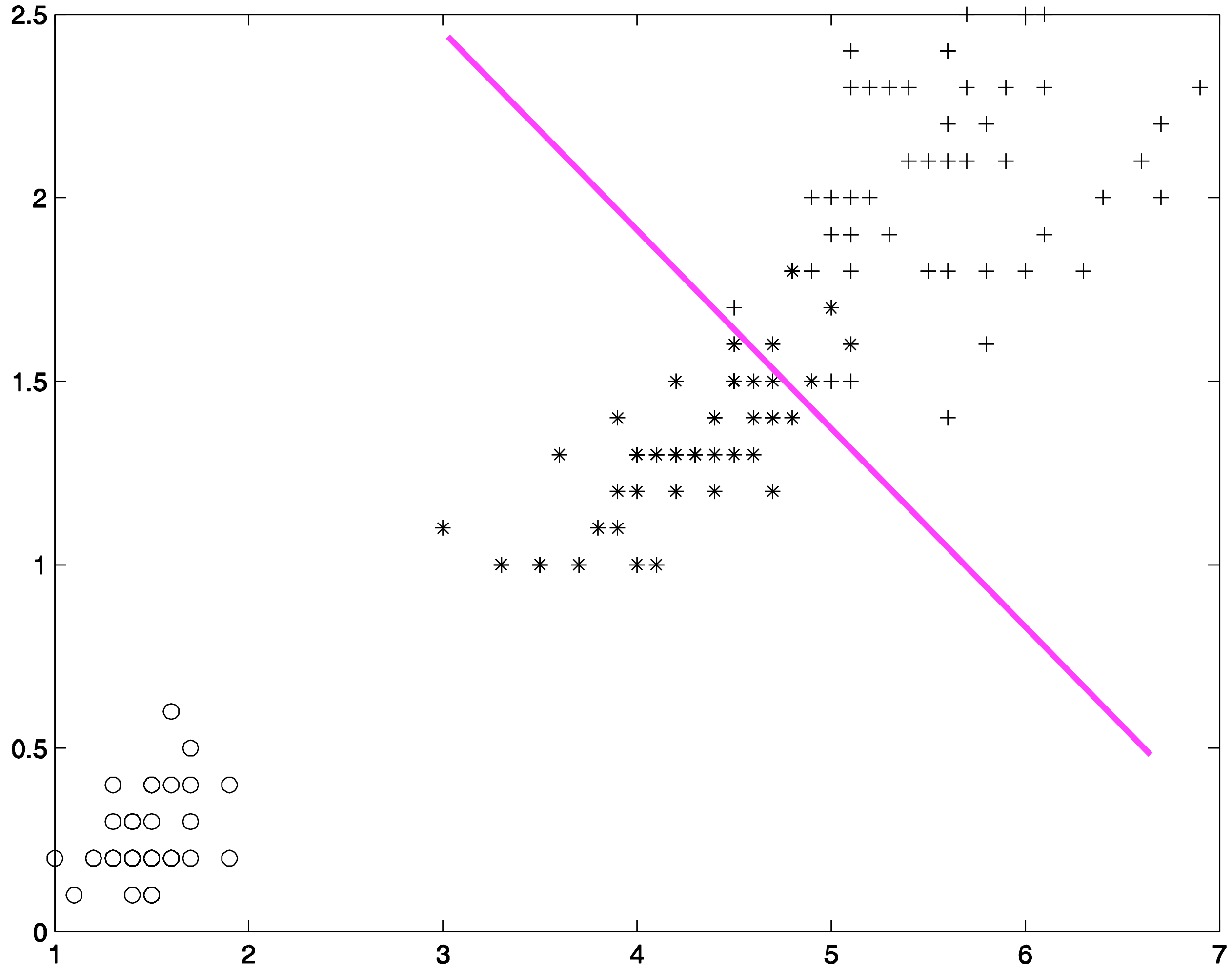
(i) Find "representatives" among the w vectors.

(ii) Alternative prediction rule:

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

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

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



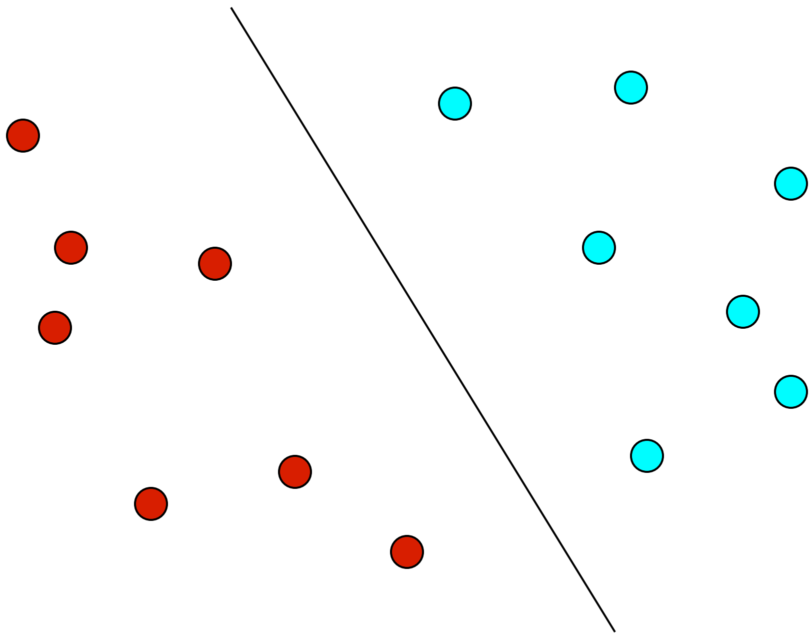
100 rounds, 1595 updates (5 errors)

Final hypothesis: makes 5 errors for voting

Interesting questions

Modify the (voted) perceptron algorithm to:

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



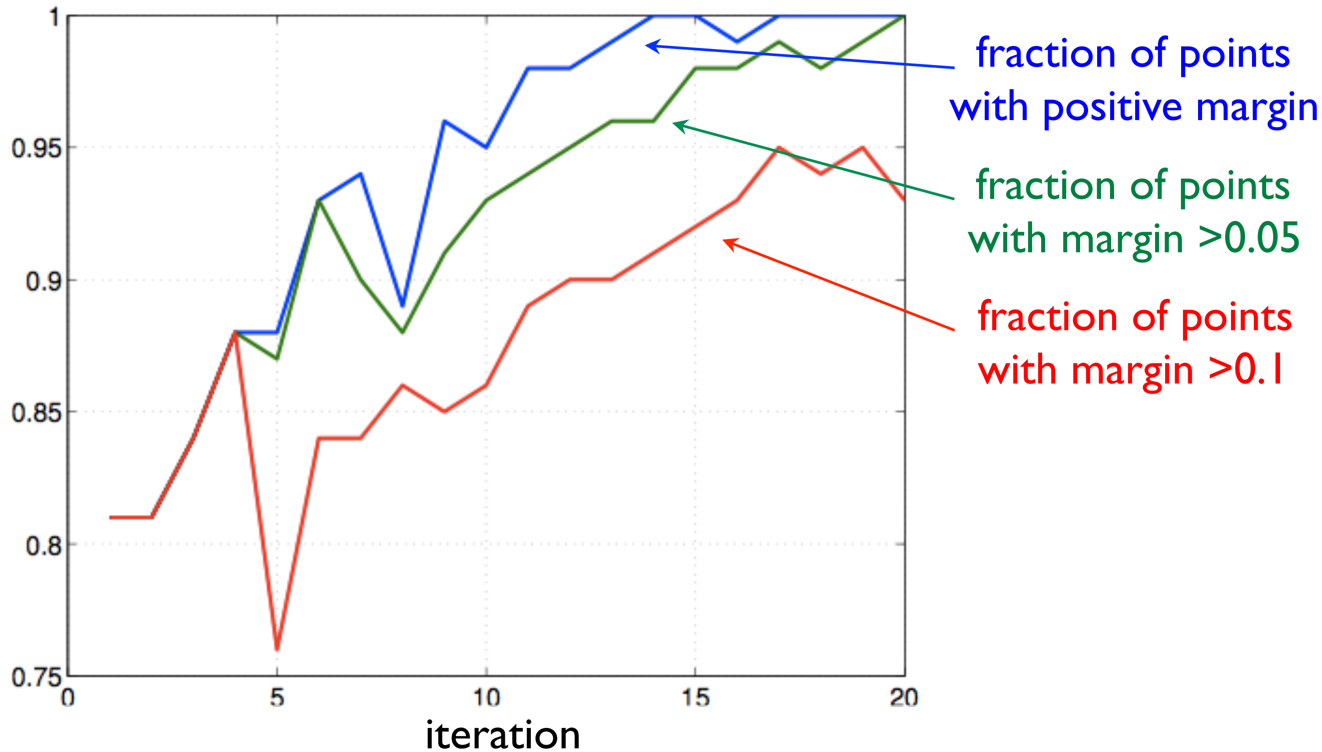
[2] "Give up" on troublesome points after a while

Voting margin and generalization

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

$$\gamma_t = \frac{y_t h_m(\underline{x}_t)}{\sum_{j=1}^{m} \alpha_j} = \frac{\alpha_1 y_t h(\underline{x}_t; \underline{\theta}_1) + \ldots + \alpha_m y_m h(\underline{x}_t; \underline{\theta}_m)}{\alpha_1 + \ldots + \alpha_m} \in [-1, 1]$$

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

Weak classifier

Weight

Data point

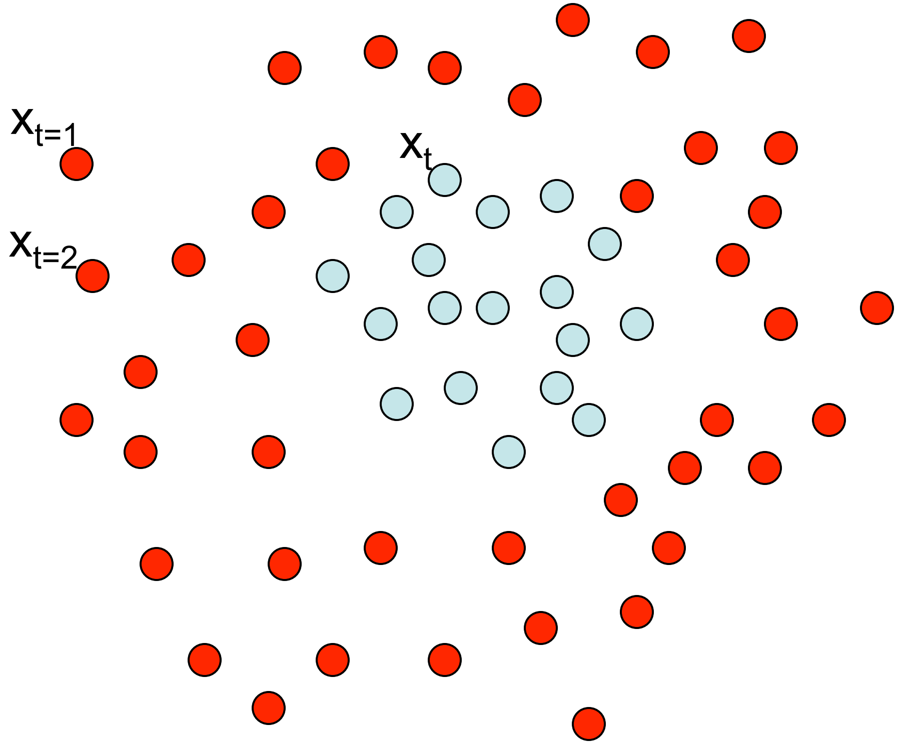
- We need to define a family of weak classifiers

$$f_k(x)$$

  – 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



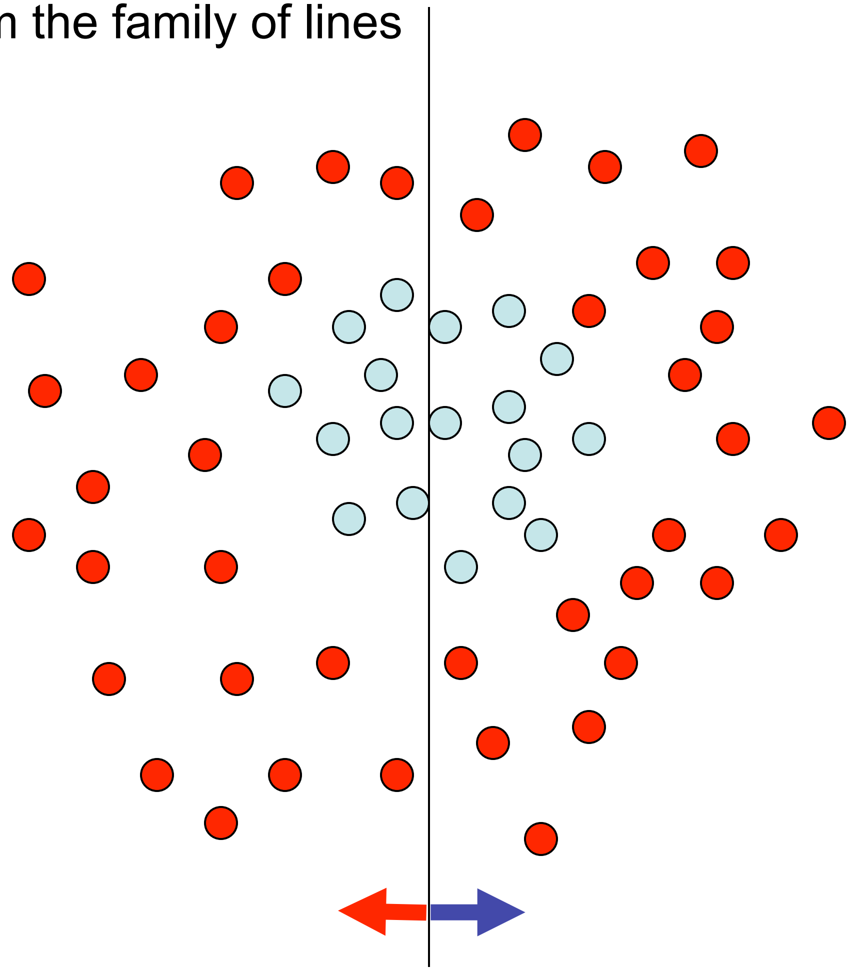
Each data point has a class label:

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

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:

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

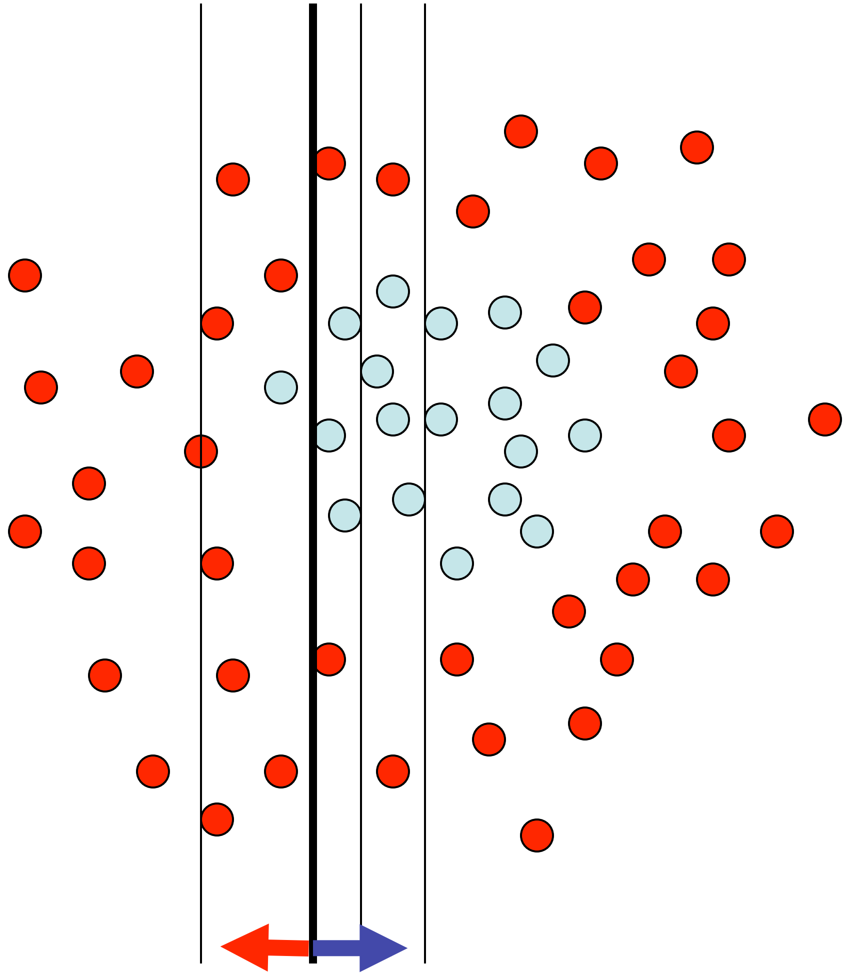
and a weight:

$w_t = 1$

This linear separator has error rate 50%

Torralba, ICCV 05 Short Course

Example using linear separators



Each data point has a class label:

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

and a weight:

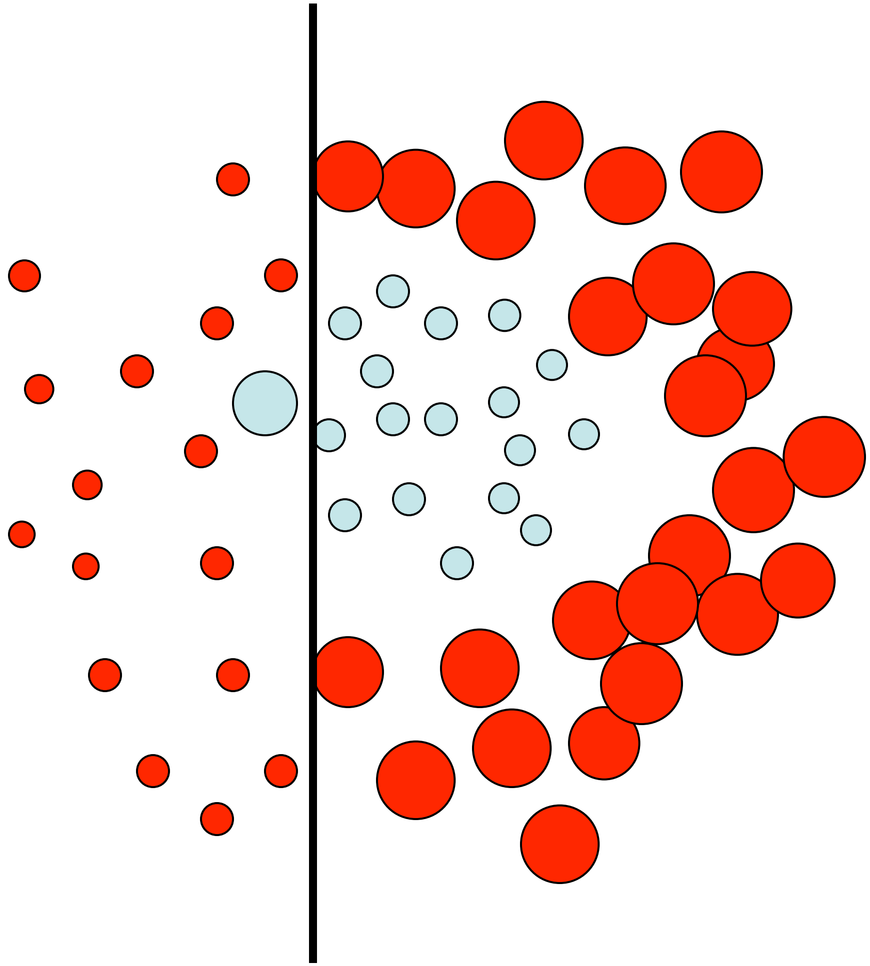
$w_t = 1$

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:

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

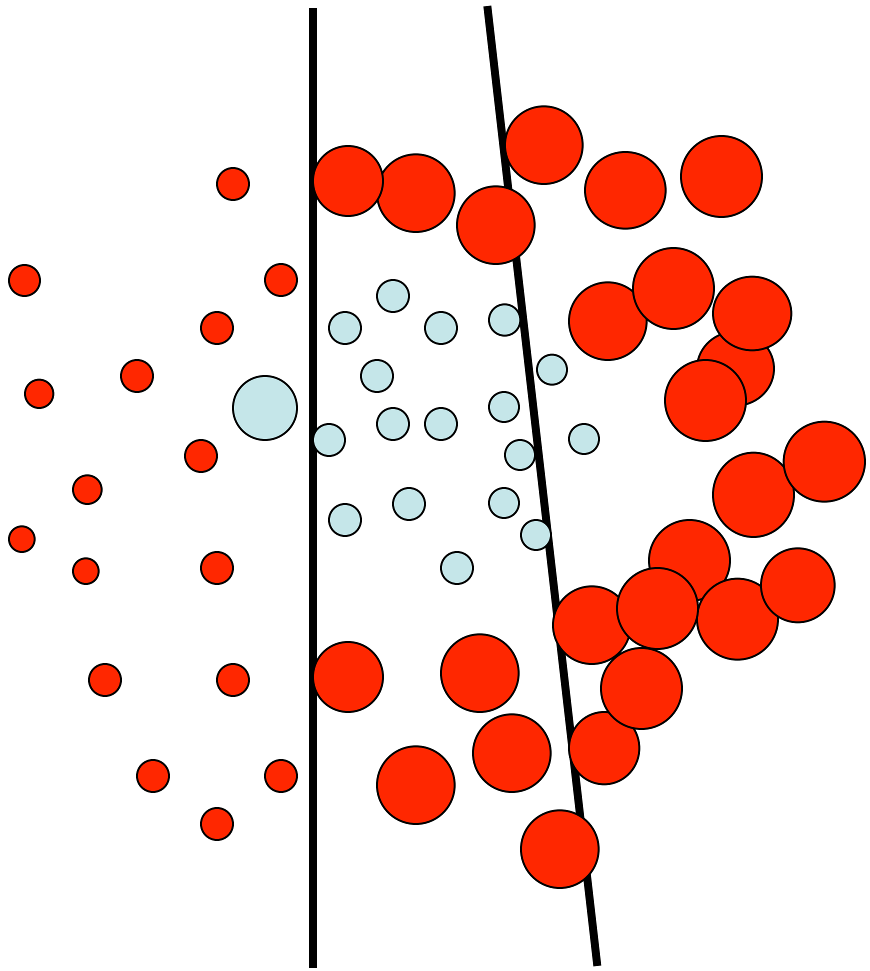
We update the weights:

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

- 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:

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

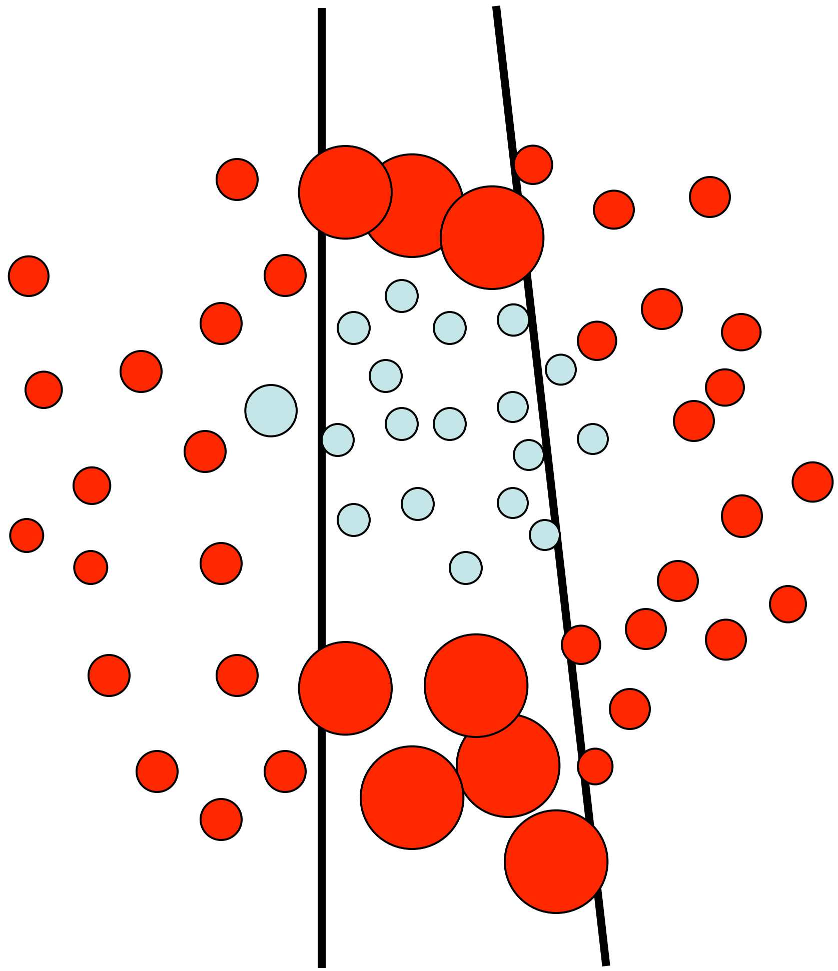
We update the weights:

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

- 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:

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

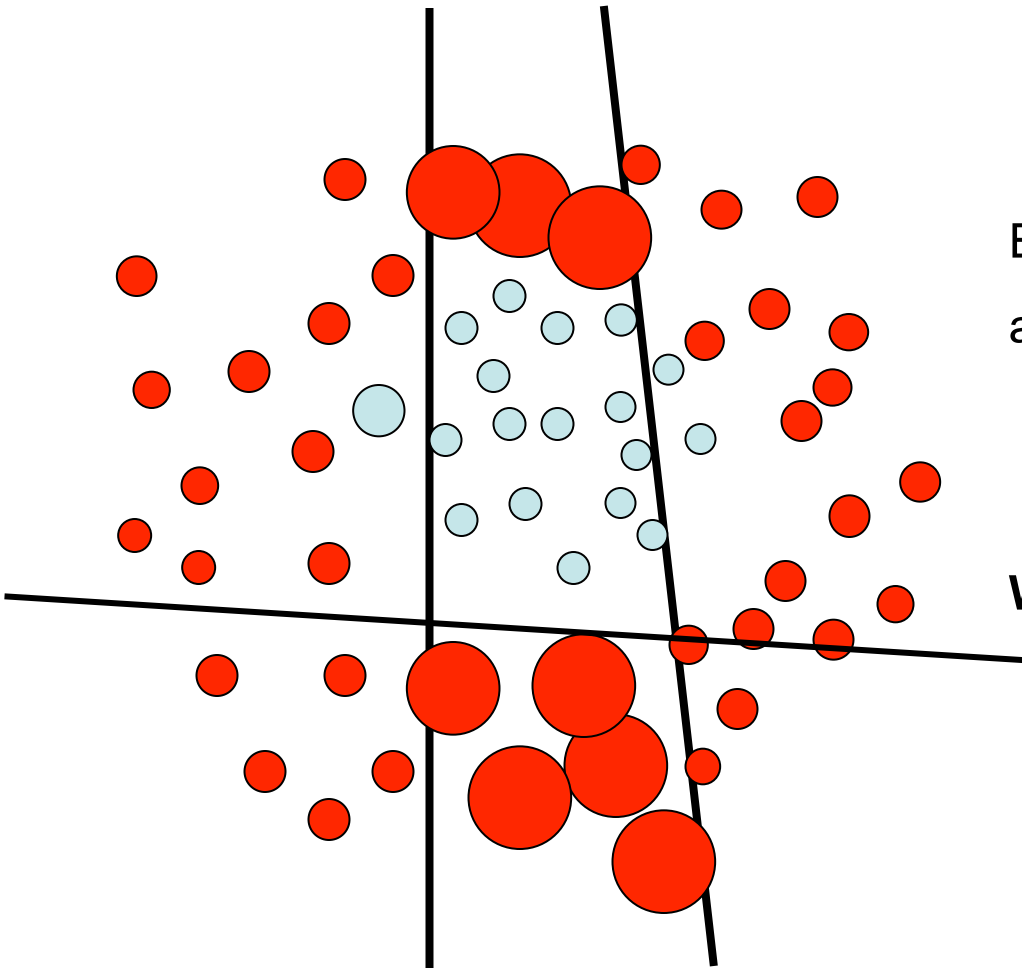
We update the weights:

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

- 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:

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

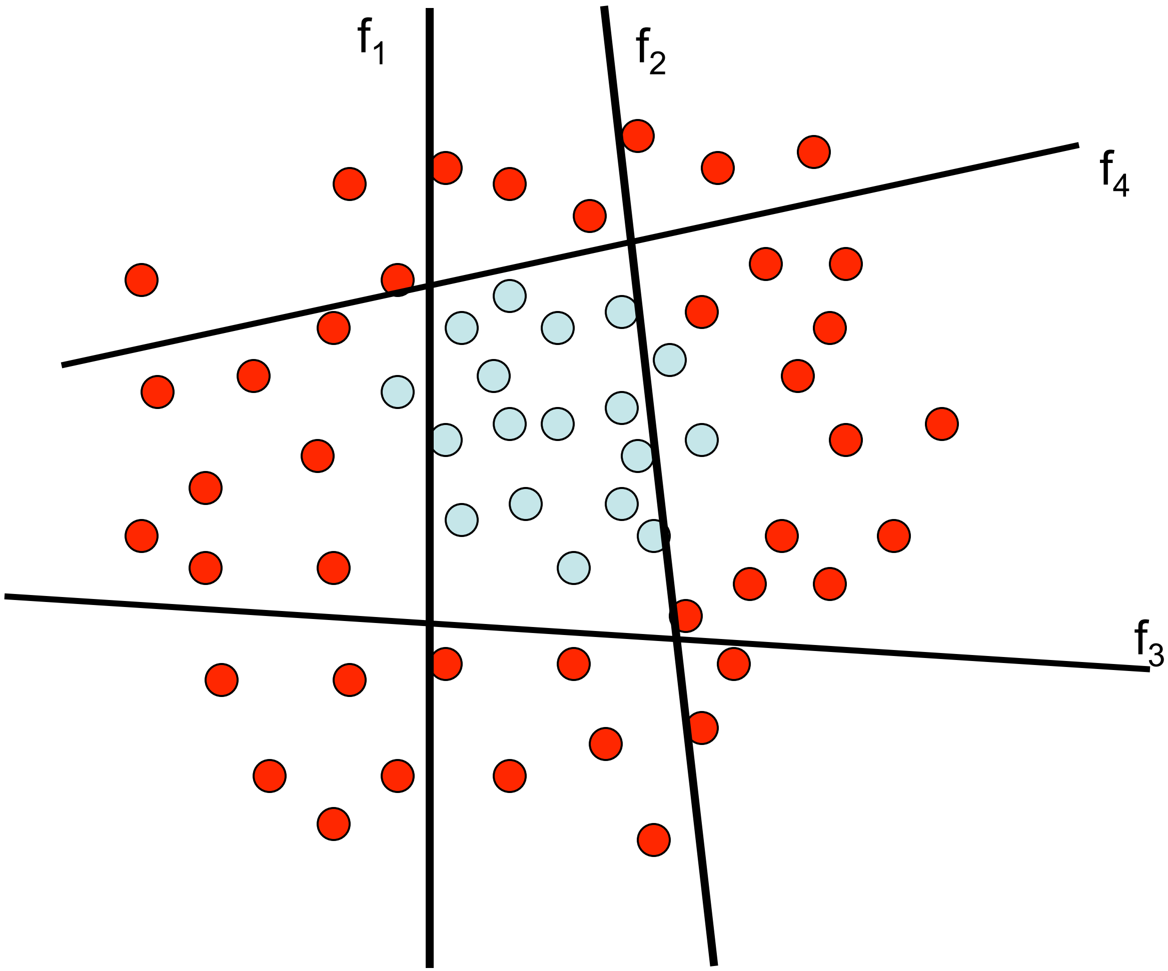
We update the weights:

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

- 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



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

Torralba, ICCV 05 Short Course

Boosting

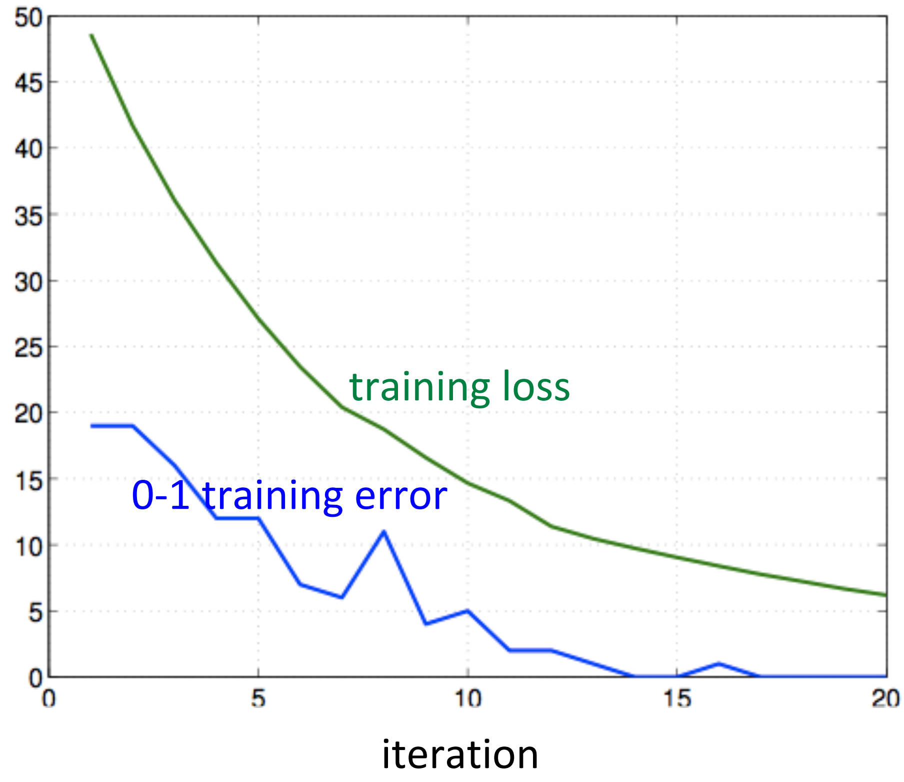
- AdaBoost (Freund and Shapire, 1995)
- Real AdaBoost (Friedman et al, 1998)
- LogitBoost (Friedman et al, 1998)
- Gentle AdaBoost (Friedman et al, 1998)
- BrownBoosting (Freund, 2000)
- FloatBoost (Li et al, 2002)
- …

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

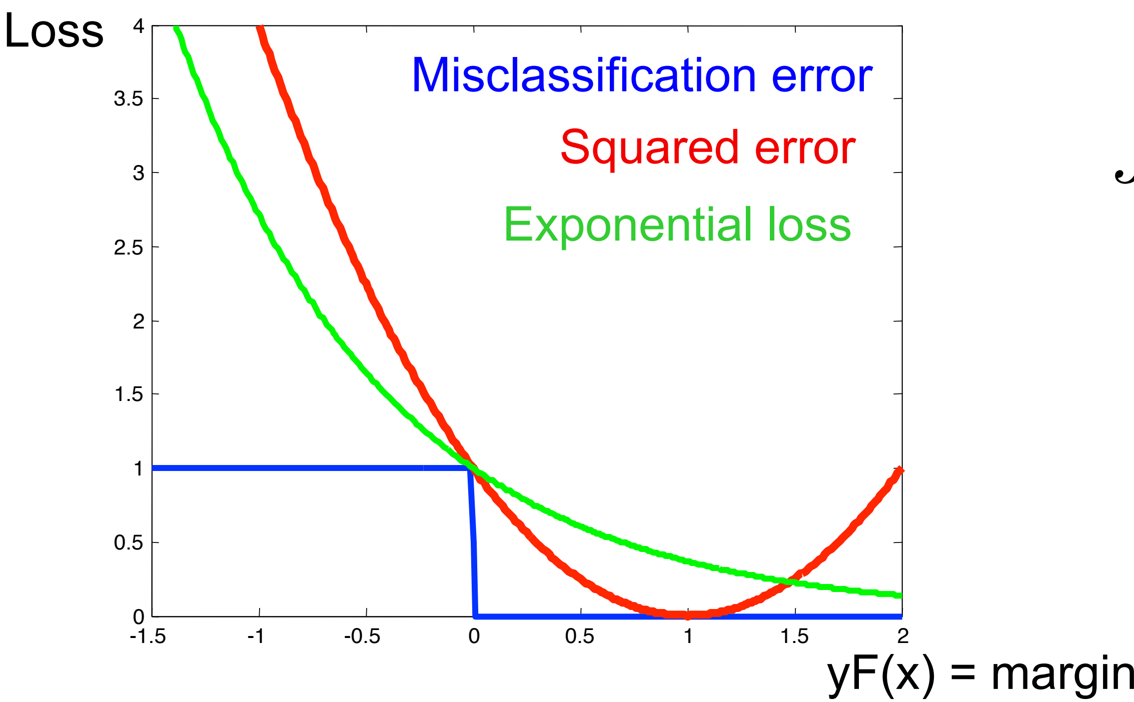
Torralba, ICCV 05 Short Course

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 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:

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

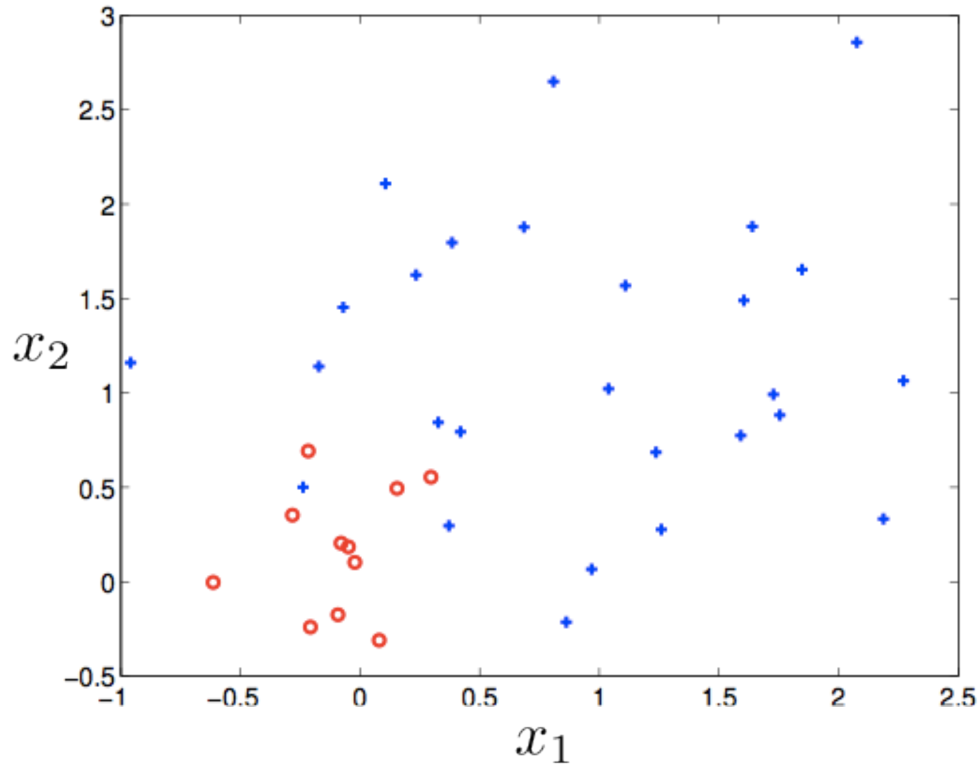
- How to set the α_m values?
- Depends on the algorithm. E.g. in AdaBoost:

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

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

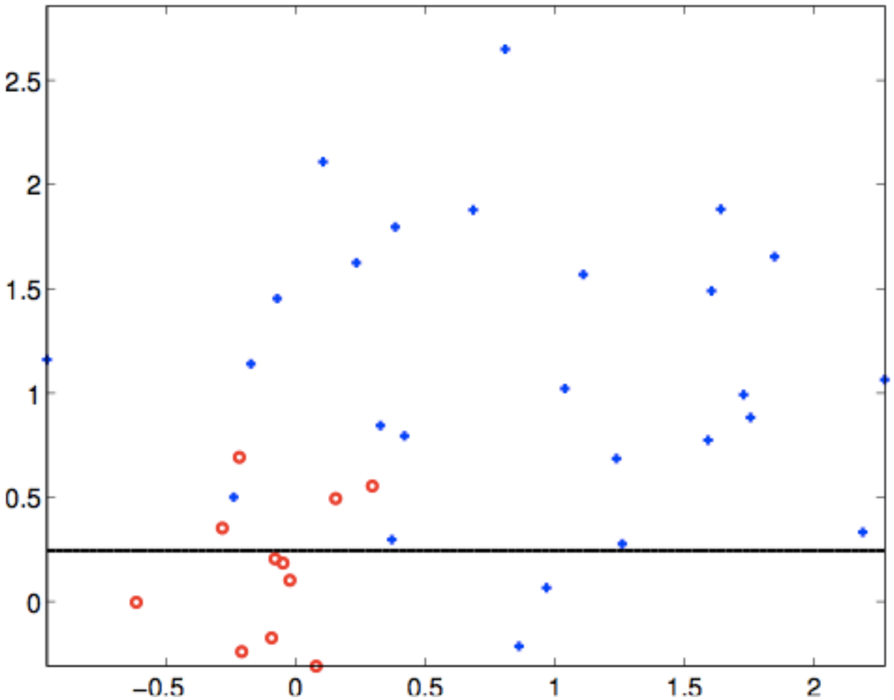
Boosting example

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

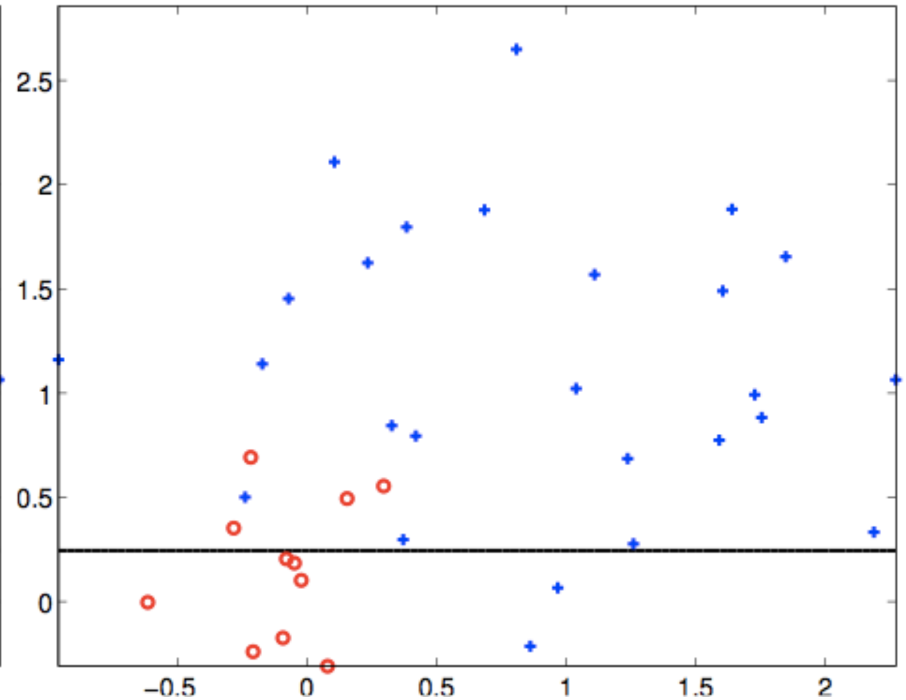


Boosting example

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

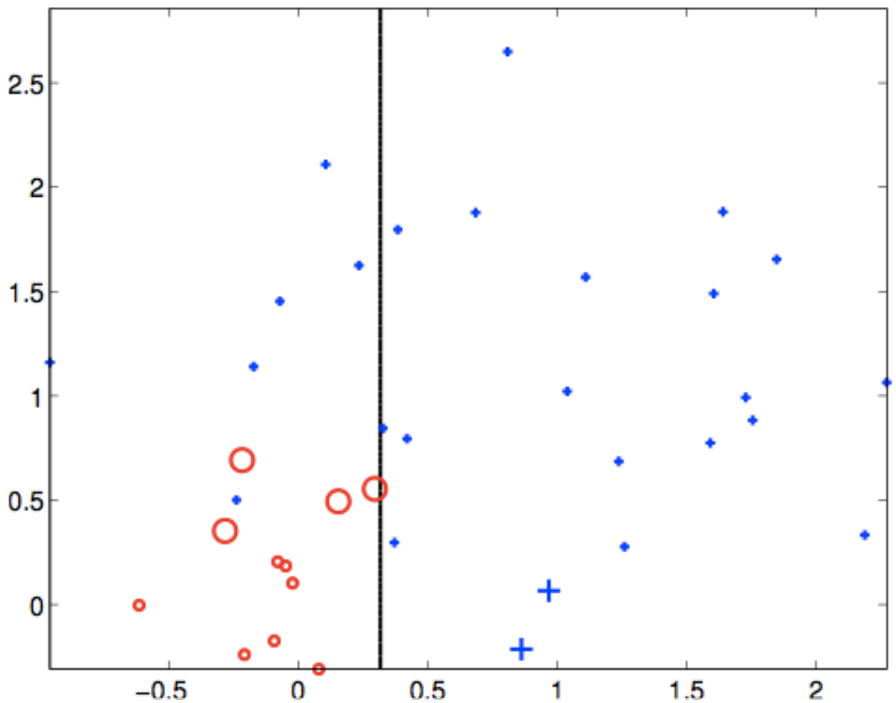


$h_1(\underline{x}) = \hat{\alpha}_1 h(\underline{x};\hat{\underline{\theta}}_1)$

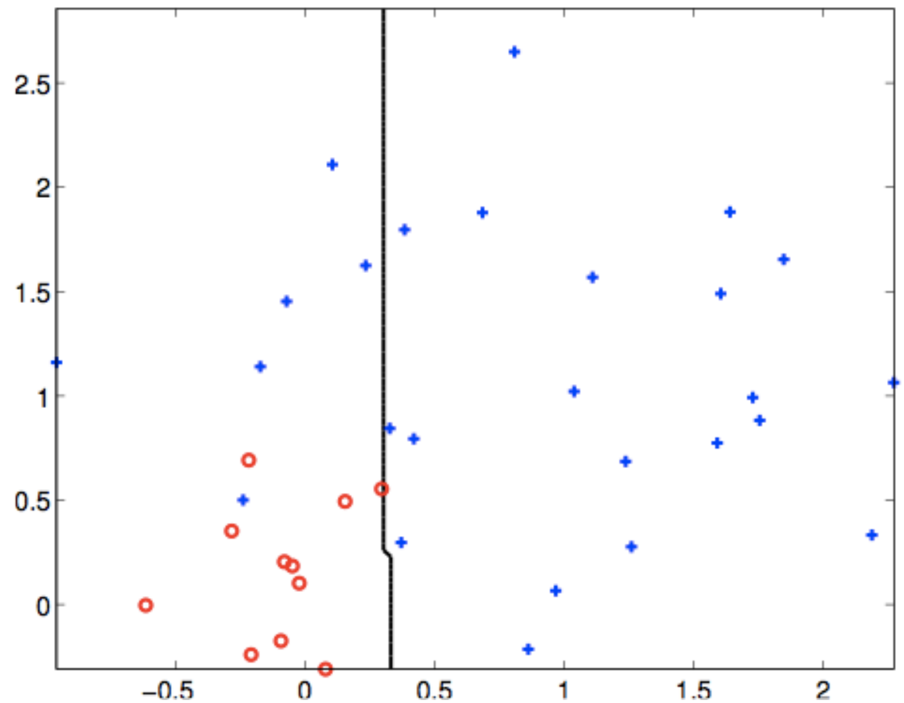


Boosting example

$$h(\underline{x};\hat{\theta}_2)$$

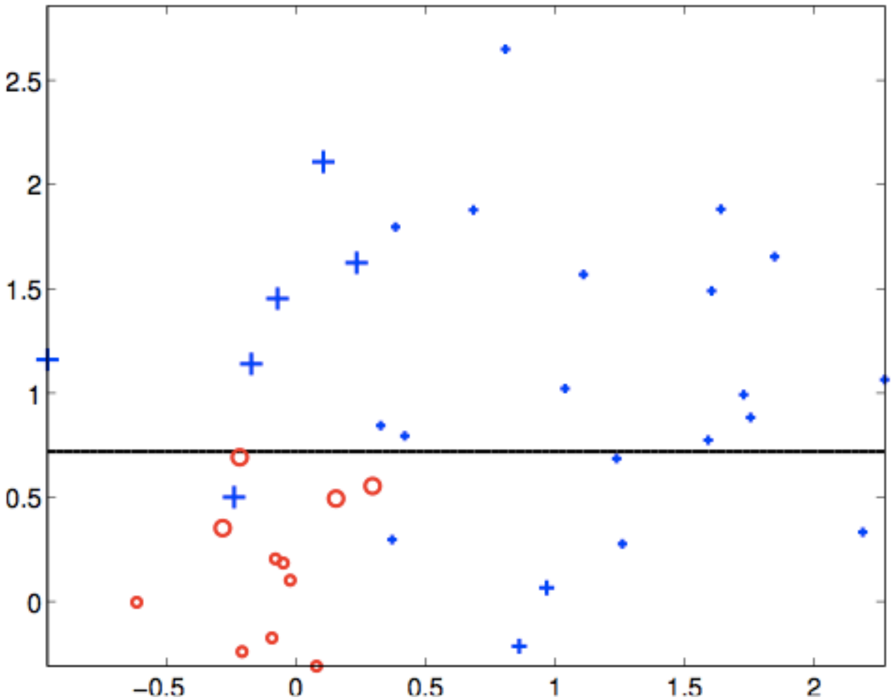


$$h_2(\underline{x}) = \hat{\alpha}_1 h(\underline{x};\hat{\underline{\theta}}_1) + \hat{\alpha}_2 h(\underline{x};\hat{\underline{\theta}}_2)$$

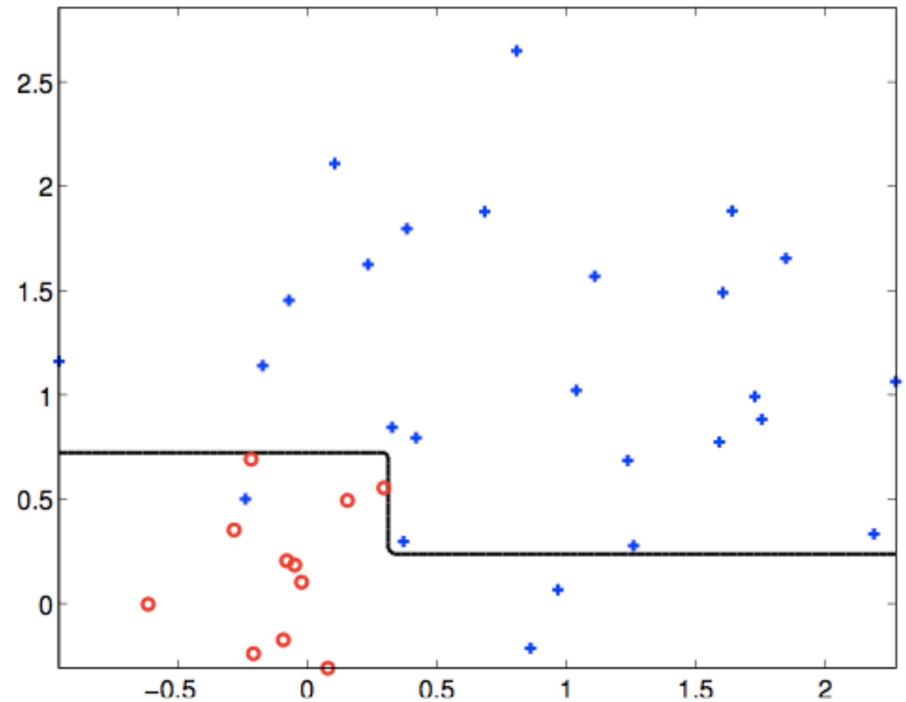


Boosting example

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

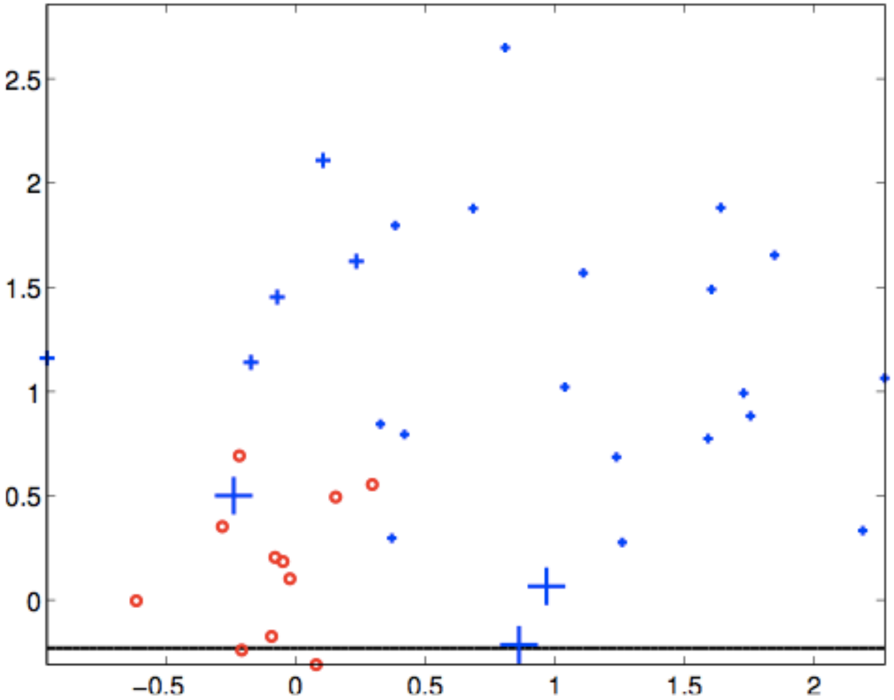


$h_3(\underline{x}) = \hat{\alpha}_1 h(\underline{x}; \underline{\hat{\theta}}_1) + \cdots + \hat{\alpha}_3 h(\underline{x}; \underline{\hat{\theta}}_3)$

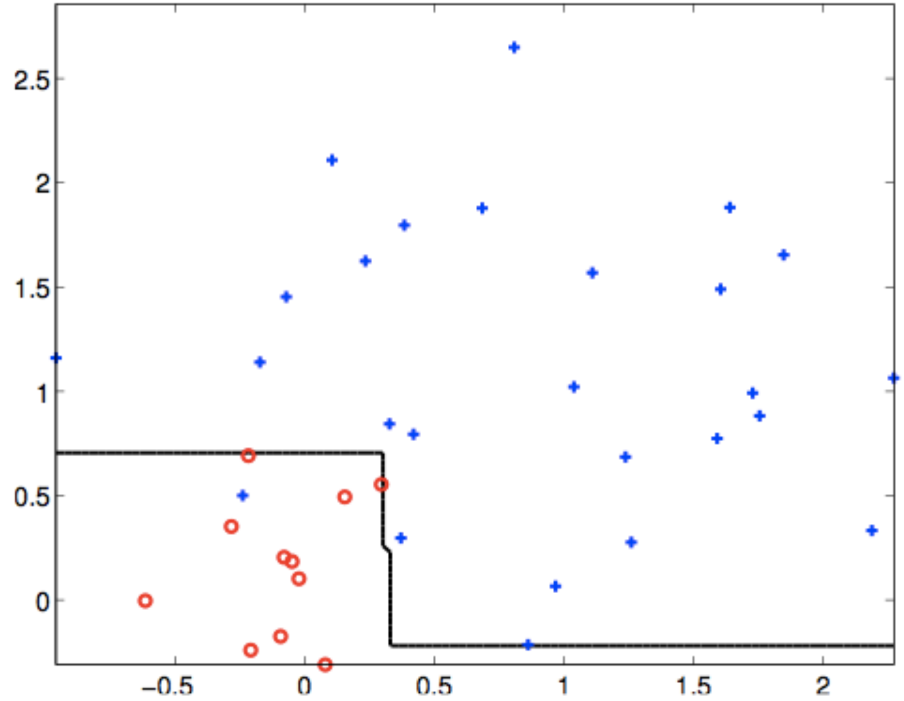


Boosting example

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



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



Bias-Variance Error Decomposition

Assume the data *(x,y)* is drawn i.i.d. from *D*, s.t.: $y = f(x) + \epsilon$

where: $\mathrm{E}[\epsilon] = 0$

$\mathrm{Var}(\epsilon) = \sigma^2$

Then, for any data point (x_0, y_0),

The image cannot be displayed. Your computer may not have enough memory to open the image, or the image may have been corrupted. Restart your computer, and then open the file again. If the red x still appears, you may have to delete the image and then insert it again.

$$= \boxed{\sigma^2} + \mathrm{Var}[h(x_0)] + (\mathrm{Bias}[h(x_0)])^2$$

Irreducible error (can't be changed by choice of h)

where: $\mathrm{Var}[h(x_0)] = \mathrm{E}[h(x_0)^2] - \mathrm{E}[h(x_0)]^2$

The image cannot be displayed. Your computer may not have enough memory to open the image, or the image may have been corrupted. Restart your computer, and then open the file again. If the red x still appears, you may have to delete the image and then insert it again.

NOTE: All expectations are w.r.t. the random training set *h* is learned from, NOT x_0.

Understanding boosting

- There are four different kinds of errors in the boosting procedure that we can try to understand
  - 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