LECTURE 22

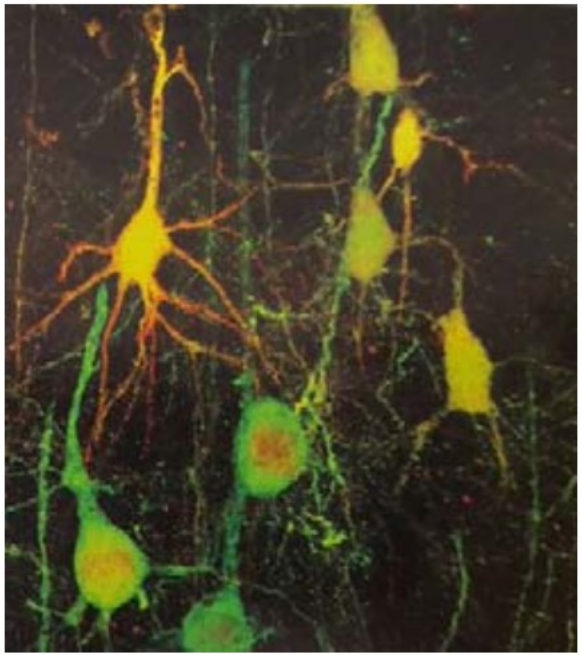
Boosting – Part I

A technique for combining a number of “weak” classifiers to make a “strong” classifier

Agenda

- Basic algorithm: introducing the boosting procedure
- Ensemble Methods: Boosting and Bagging
- Different versions:
    - Adaboost, RealBoost, and LogitBoost optimizing different design of loss functions.
- A statistical framework: discussing its relation to Maximum likelihood estimation (MLE learning).

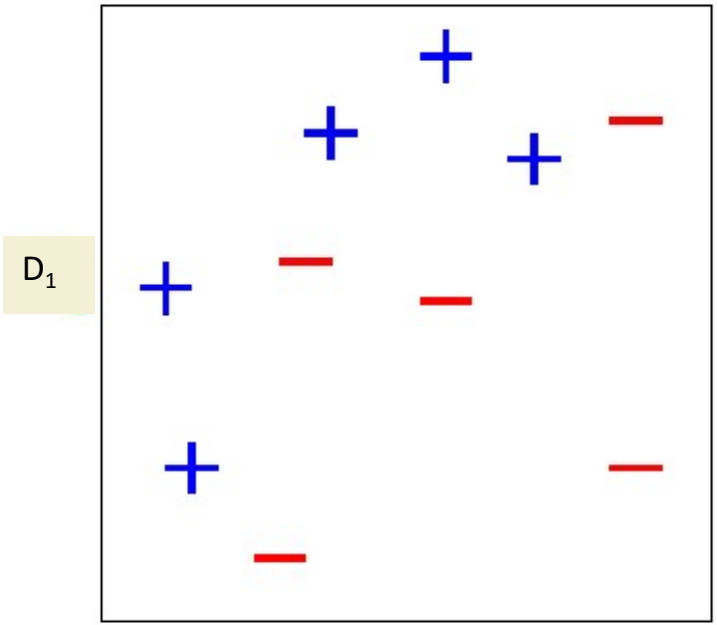
Background



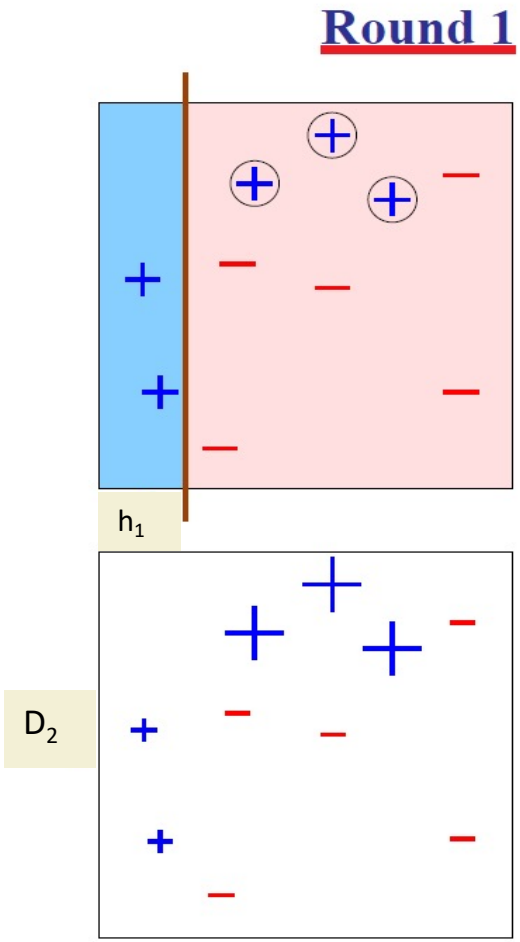
- Boosting is related to the early ideas in neural network, more specifically the perceptron proposed by Frank Rosenblatt 1962 as a model of neurons.
- Each neuron is modeled by a linear product of the input feature vector and a weight vector, which is then followed by a threshold.

A toy example

A Toy Example

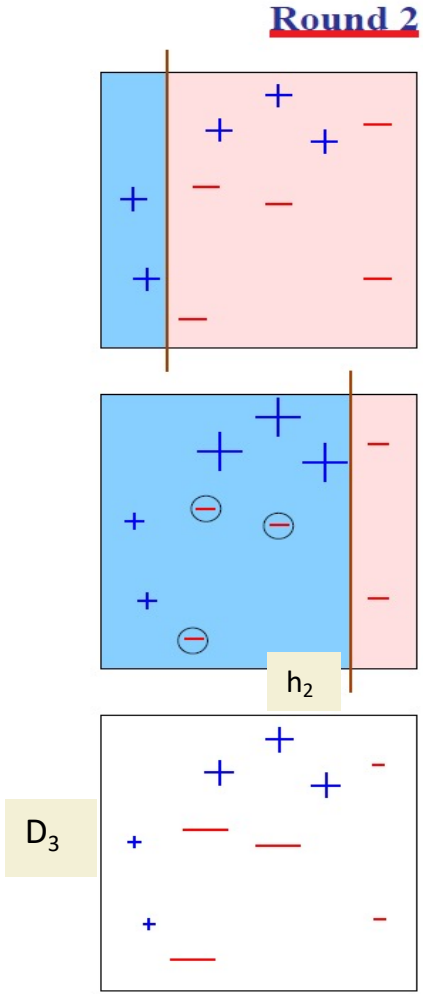


A Toy Example



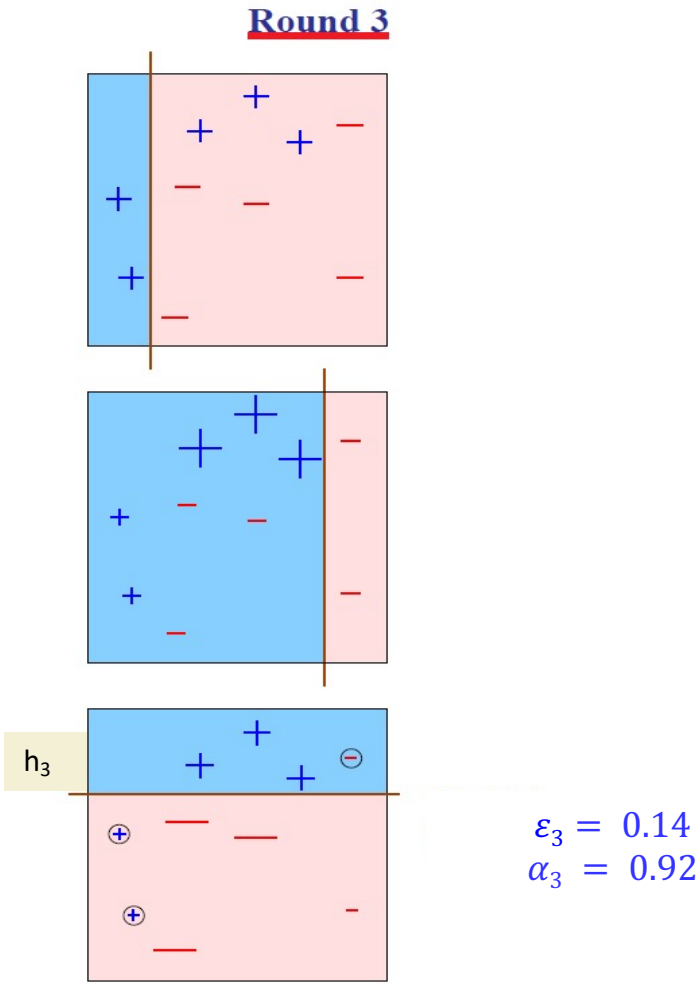
$\varepsilon_1 = 0.3$
$\alpha_1 = 0.42$

A Toy Example



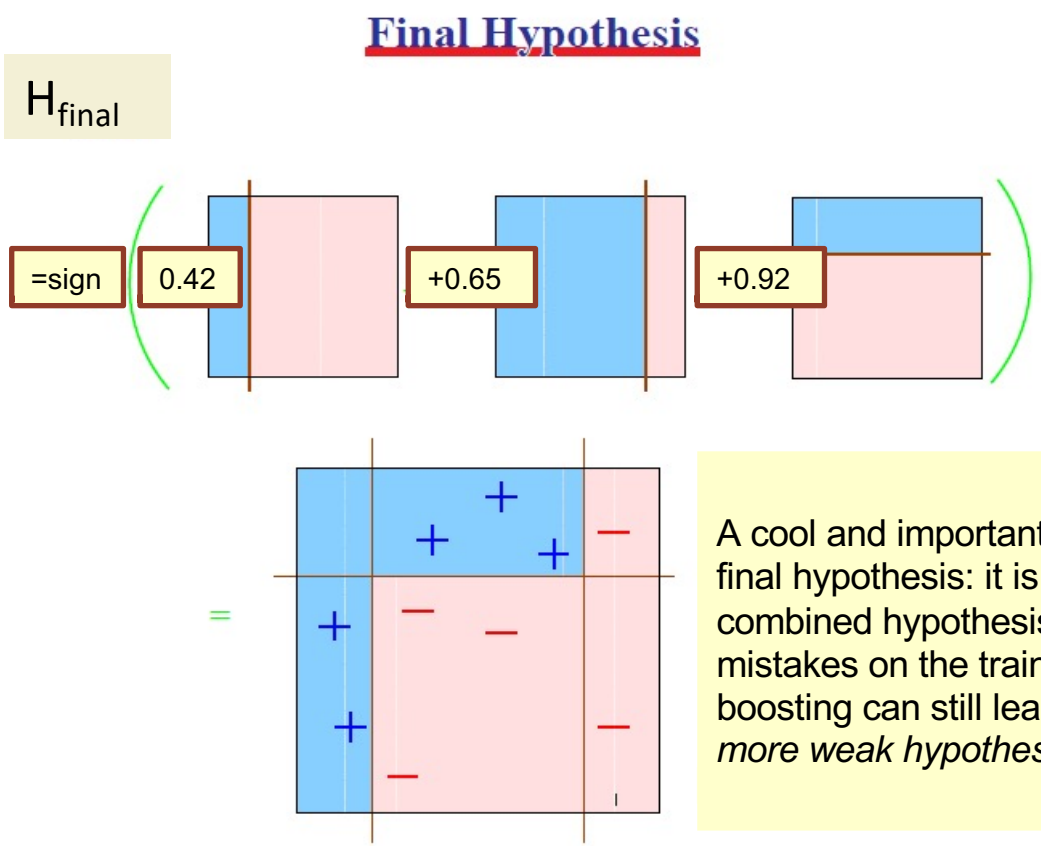
$\varepsilon_2 = 0.21$
$\alpha_2 = 0.65$

A Toy Example



$\varepsilon_3 = 0.14$
$\alpha_3 = 0.92$

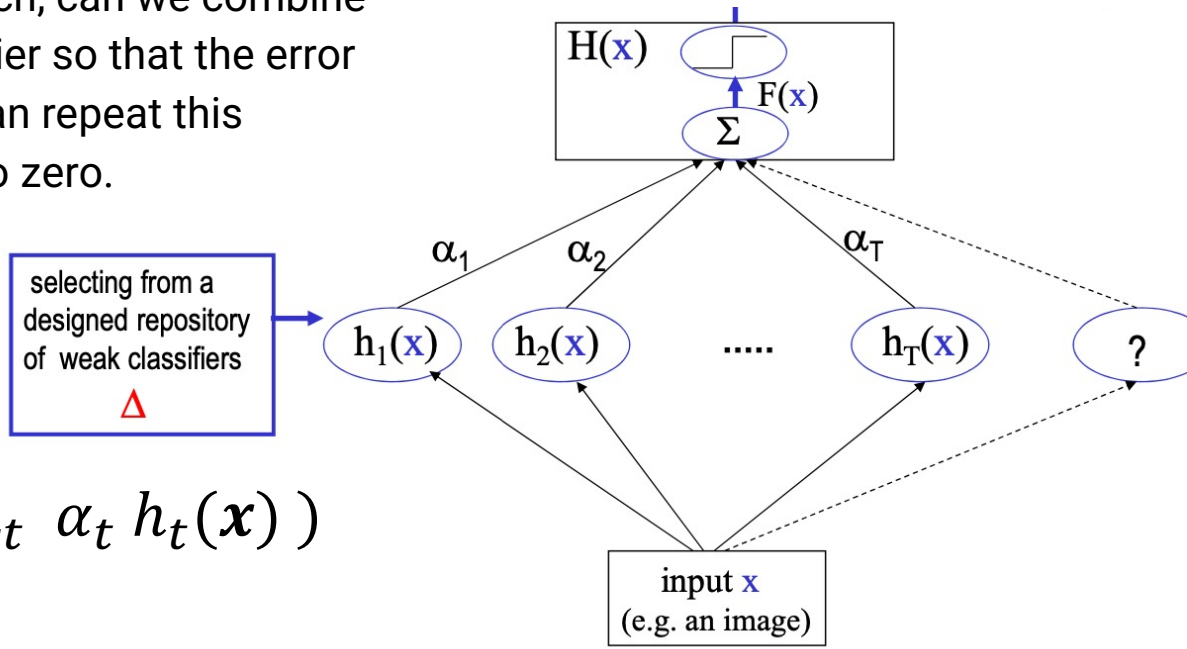
A Toy Example



A cool and important note about the final hypothesis: it is possible that the combined hypothesis makes no mistakes on the training data, but boosting can still learn, *by adding more weak hypotheses.*

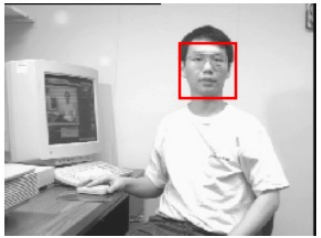
Intuition

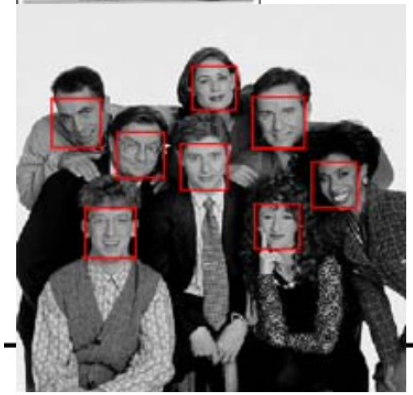
- Suppose we have two weak classifiers h1 and h2 which have 49% error each, can we combine them to make a new classifier so that the error becomes lower? If so, we can repeat this process to make the error to zero.

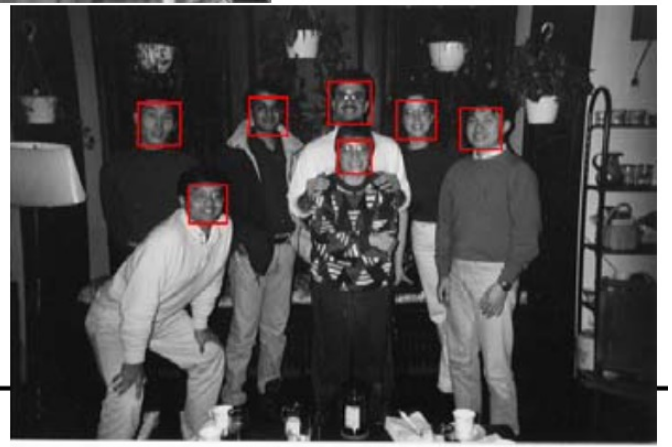


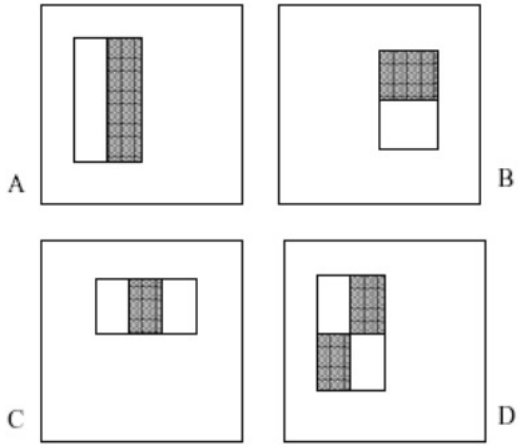
$$H_{final}(\boldsymbol{x}) = sign\left(\sum_t \alpha_t h_t(\boldsymbol{x})\right)$$

Example of weak classifier









Weak classifiers used for face detection in (Viola and Jones 01) are windowed features A,B,C,D on a 24x24 pixel image patches. The features are designed for easy computation using the integral image.

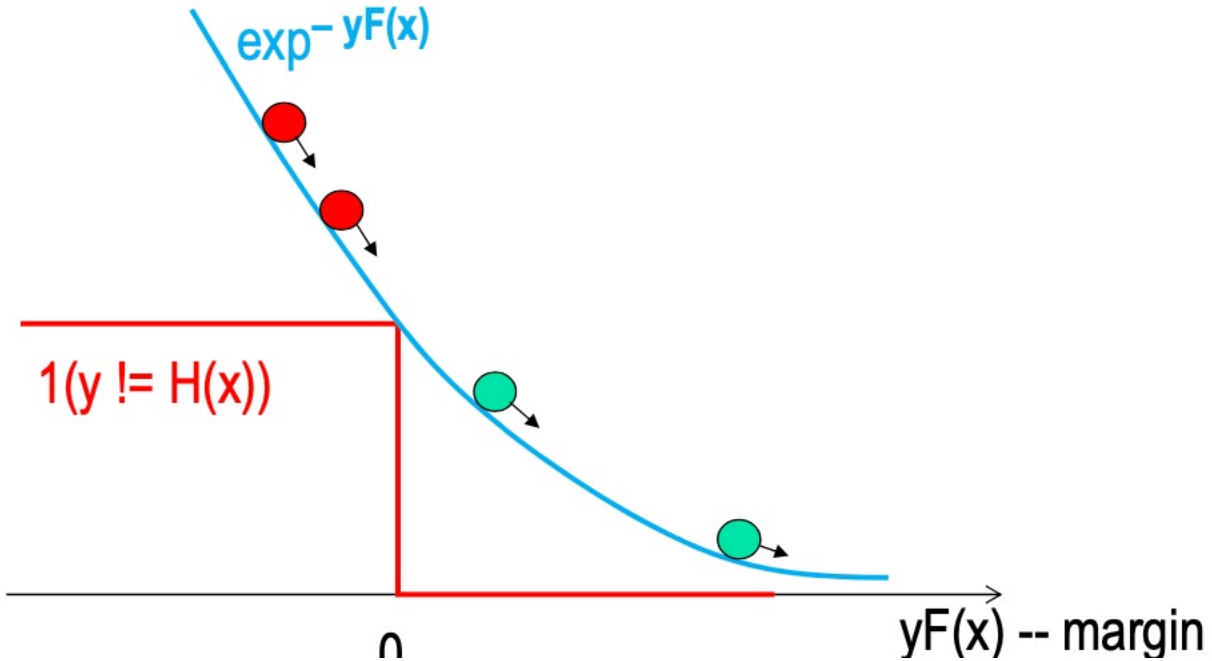
Basic Boost

- A strong classifier is a combination of a number of **weaker classifiers**:
  - $H(\boldsymbol{x}) = sign\left(\sum_t \alpha_t h_t(\boldsymbol{x})\right)$
- We denote by
  - $h = (h1, \ldots, h_T)$
  - $\alpha = (\alpha_1, \ldots, \alpha_T)$
  - F(x) = $\sum_t \alpha_t h_t(\boldsymbol{x}) = < \alpha, h >$
- So our objective is to choose h and parameters α to minimize the empirical error of the strong classifier
  - $Err(H) = \frac{1}{n}\sum_{i=1}^{n} 1(H(x_i) \neq y_i)$
  - $(\hat{\alpha}, h) = argmin\ Err(H)$

Boosting

- Initialization:
  - Weigh all training samples equally
- Iteration Step:
  - Train model on (weighted) train set
    - Choose your favorite hypothesis space & learning algorithm
  - Compute error of model on train set
  - Update the distribution:
    - Increase/decrease weights on training cases model gets wrong/correct.
- Typically requires $100's$ to $1000's$ of iterations
- Return final model:
  - Carefully weighted prediction of each model

Adaboost



Intuitively, a margin measures how far away
a data point is away from the decision boundary.

Adaboost Loss function

- It is difficult to derive such a loss function, so the following function is used instead.
- $Err(H) = \frac{1}{n}\sum_{i=1}^{n} 1(H(x_i) \neq y_i) \leq \frac{1}{n}\sum_{i=1}^{n} e^{-y_i F(x_i)}$

  (minimize the exponential loss) , an upper bound on 0/1 loss
- $Loss(F) = \frac{1}{n}\sum_{i=1}^{n} e^{-y_i F(x_i)}$
- $(h_{t+1}, \alpha_{t+1}) = \text{argmin}_{h,\alpha}\left(Loss(F_t + \alpha h)\right) = \text{argmin}_{h,\alpha}\frac{1}{n}\sum_{i=1}^{n} e^{-y_i[F(x_i)+\alpha h]}$

  $= \text{argmin}_{h,\alpha}\frac{1}{n}\sum_{i=1}^{n} \omega_i e^{-y_i \alpha h(x_i)}$

Basic AdaBoost algorithm

1. Initialize the data with uniform weight

$$D_0(x_i) = \frac{1}{n} \text{ so, } \sum_{i=1}^{n} D_0(x_i) = 1$$

2. At step t, compute the weighted error for each weak classifier

$$\varepsilon_t(h) = \sum_{i=1}^{n} D_t(x_i)\, 1(h(x_i) \neq y_i)$$

3. Choose a new weak classifier which has the least weighted error

$$h_t = argmin\ \varepsilon_t(h)$$

4. Assign weight for the new classifier

$$\alpha_t = \frac{1}{2}\log\left(\frac{1-\varepsilon(h_t)}{\varepsilon(h_t)}\right)$$

5. Update the weights of the data points

$$D_t(x_i) = \frac{1}{Z_t} D_{t-1}(x_i) e^{-y_i \alpha_t h_t(x_i)}$$

Set t+1→t, repeat 2-5 until stopping conditions

The algorithm stops under three possible conditions:

- The training error of the strong classifier H(x) is below a threshold, or become zero.
  - In fact, people can continue to boost after the training error becomes zero, such that the positive and negative examples are separated by a bigger margin
- All the remaining weak classifiers have error close to 0.5 and thus redundant.
- A maximum number of weak classifier T is reach.

Summary of Ensemble Methods

- Boosting
- Bagging (Random Forests)

Boosting: Different Perspectives

- Boosting is a maximum-margin method (Schapire et al. 1998, Rosset et al. 2004)
  - Trades lower margin on easy cases for higher margin on harder cases
- Boosting is an additive logistic regression model (Friedman, Hastie and Tibshirani 2000)
  - Tries to fit the logit of the true conditional probabilities
- Boosting is a linear classifier, over an incrementally acquired “feature space”.

Bagging

- Bagging predictors is a method for generating multiple versions of a predictor and using these to get an aggregated predictor.
  - The aggregation averages over the versions when predicting a numerical outcome and does a plurality vote when predicting a class.
- The multiple versions are formed by making bootstrap replicates of the learning set and using these as new learning sets.
  - That is, use samples of the data, with repetition

- Tests on real and simulated data sets using classification and regression trees and subset selection in linear regression show that bagging can give substantial gains in accuracy.
- The vital element is the instability of the prediction method. If perturbing the learning set can cause significant changes in the predictor constructed, then bagging can improve accuracy.