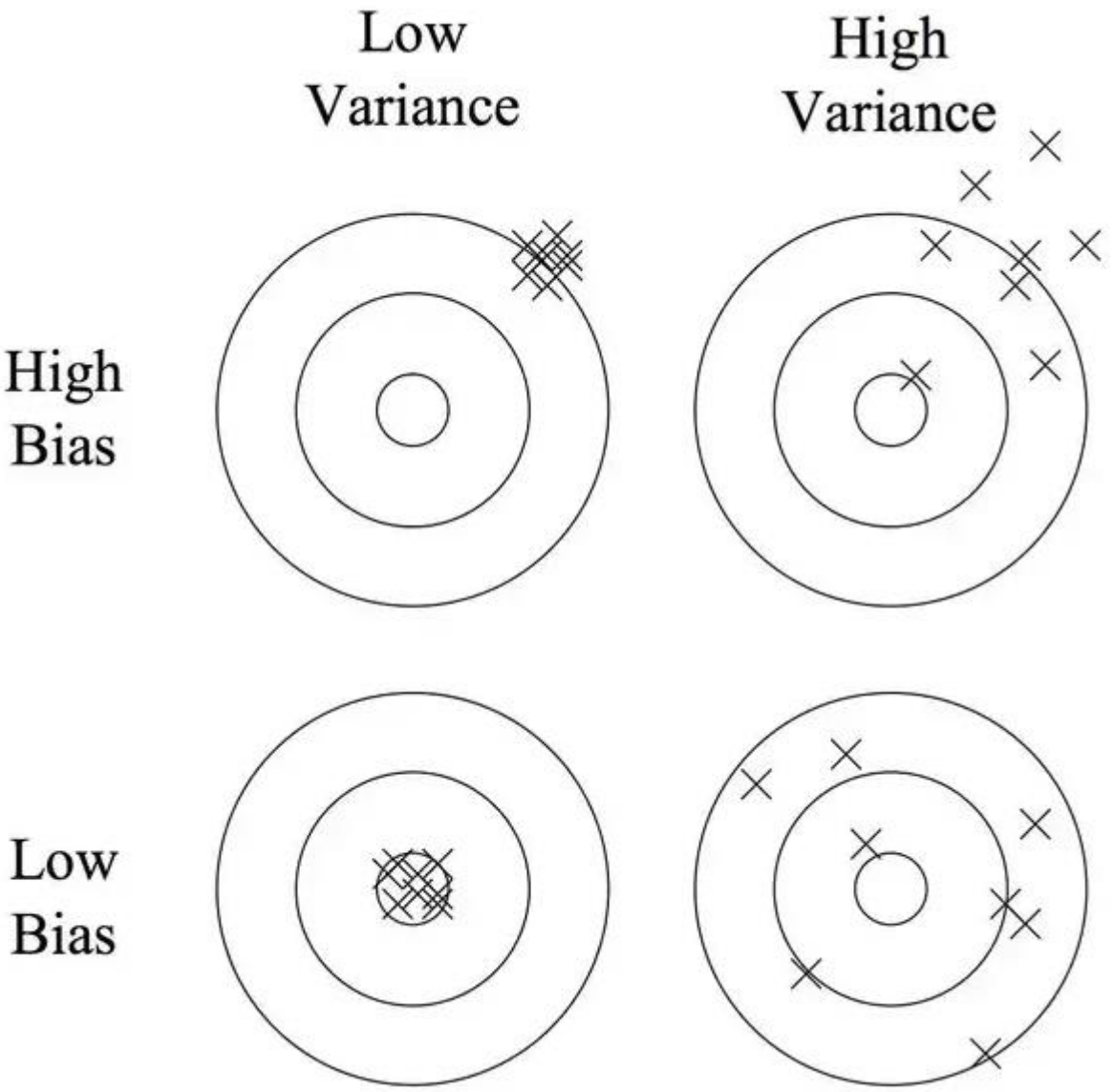
VBM683
Machine Learning

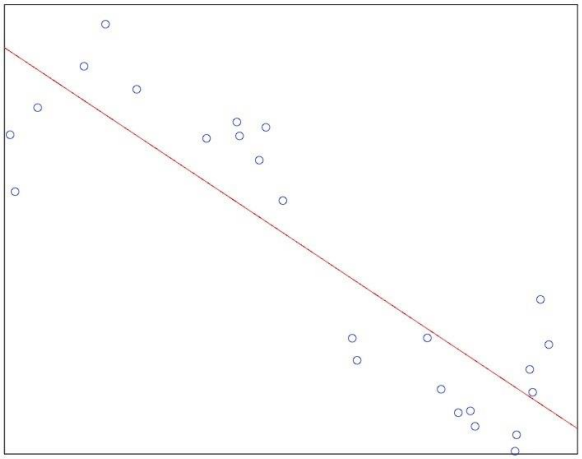
Pinar Duygulu

Slides are adapted from
Dhruv Batra

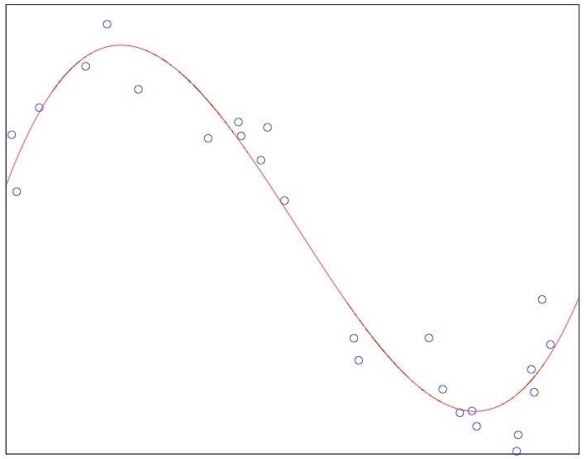


Bias is the algorithm's tendency to consistently learn the wrong thing by not taking into account all the information in the data (**underfitting**).

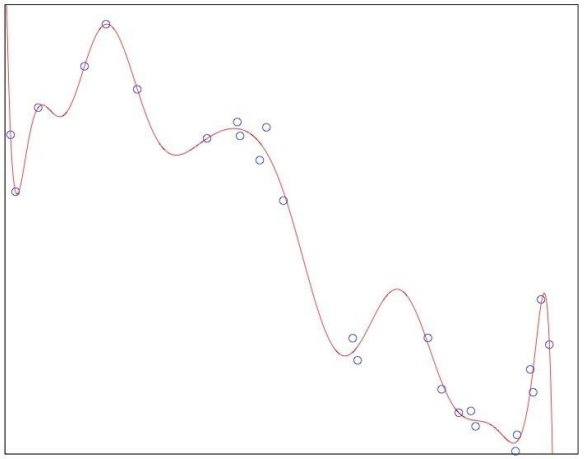
Variance is the algorithm's tendency to learn random things irrespective of the real signal by fitting highly flexible models that follow the error/noise in the data too closely (**overfitting**).



underfit
(degree = 1)



ideal fit
(degree = 3)



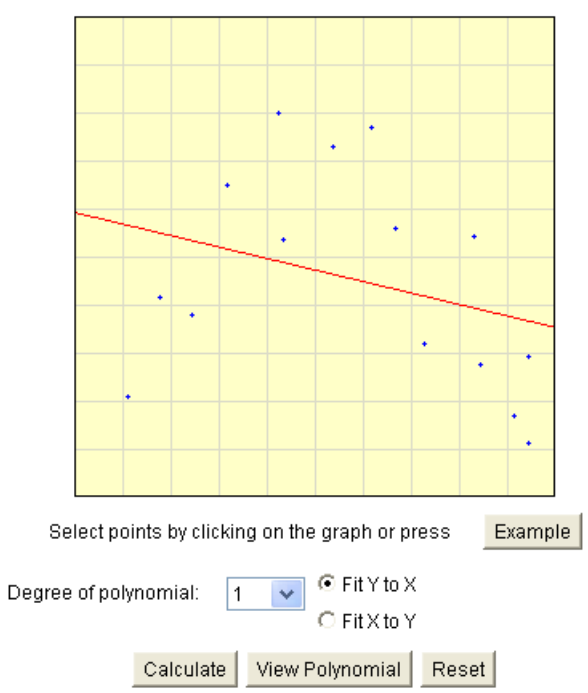
overfit
(degree = 20)

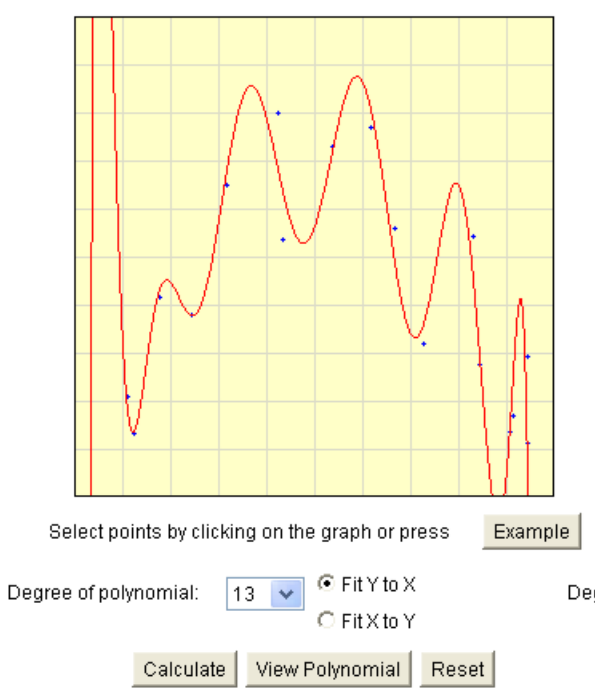
We see that the linear (degree = 1) fit is an *under-fit*:
1) It does not take into account all the information in the data (high bias), but
2) It will not change much in the face of a new set of points from the same source (low variance).

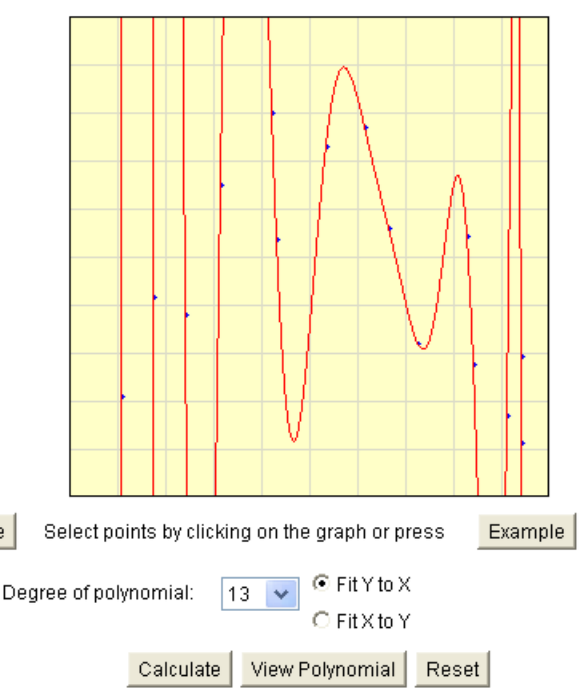
The high degree polynomial (degree = 20) fit, on the other hand, is an *over-fit*:
1) The curve fits the given data points very well (low bias), but
2) It will collapse in the face of subsets or new sets of points from the same source because it intimately takes all the data into account, thus losing generality (high variance).

Bias-Variance Tradeoff

- Choice of hypothesis class introduces learning bias
  - More complex class → less bias
  - More complex class → more variance







Fighting the bias-variance tradeoff

- **Simple (a.k.a. weak) learners**
  - e.g., naïve Bayes, logistic regression, decision stumps (or shallow decision trees)
  - **Good:** Low variance, don't usually overfit
  - **Bad:** High bias, can't solve hard learning problems

- **Sophisticated learners**
  - Kernel SVMs, Deep Neural Nets, Deep Decision Trees
  - **Good:** Low bias, have the potential to learn with Big Data
  - **Bad:** High variance, difficult to generalize

- Can we make combine these properties
  - **In general, No!!**
  - **But often yes…**

Ensemble Methods

Core Intuition: A combination of multiple classifiers will perform better than a single classifier.

Bagging

Boosting

Ensemble Methods

- Instead of learning a single predictor, learn **many predictors**
- **Output class:** (Weighted) combination of each predictor
- With sophisticated learners
  - Uncorrelated errors → expected error goes down
  - On average, do better than single classifier!
  - **Bagging**
- With weak learners
  - each one good at different parts of the input space
  - On average, do better than single classifier!
  - **Boosting**

Synonyms

- Ensemble Methods
- Learning Mixture of Experts/Committees
- Boosting types
  - AdaBoost
  - L2Boost
  - LogitBoost
  - <Your-Favorite-keyword>Boost

Bagging

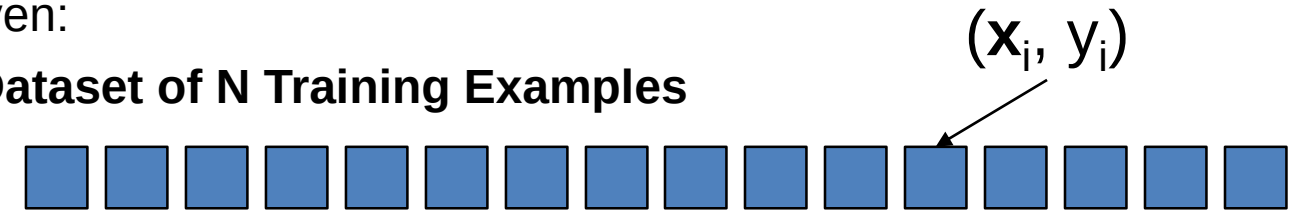
(**B**ootstrap **Ag**gregating / **B**ootstrap **A**veragin**g**)

Core Idea: Average multiple strong learners trained from resamples of your data to reduce variance and overfitting!

Bagging

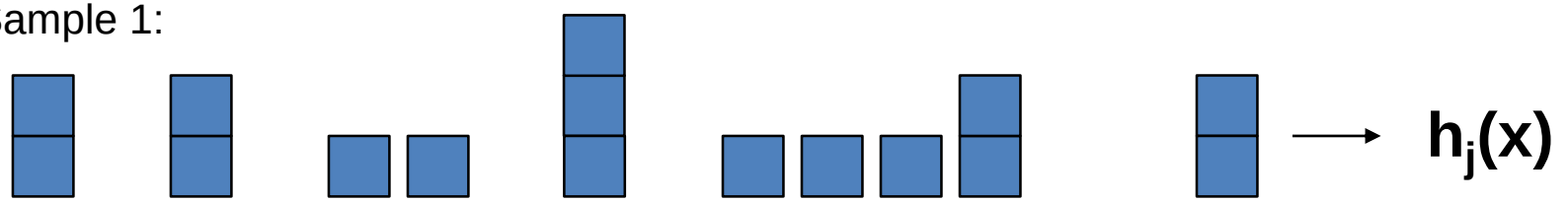
Given:

Dataset of N Training Examples

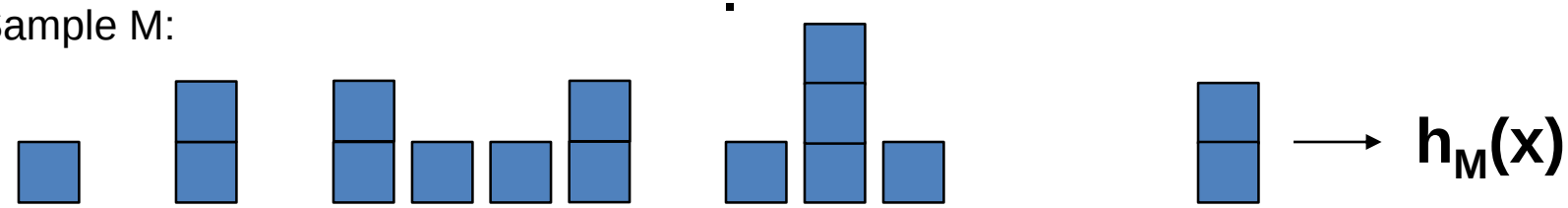


Sample N training points **with replacement** and train a predictor, repeat M times:

Sample 1:



Sample M:



At test time, output the (weighted) average output of these predictors.

Why Use Bagging

Let $\mathbf{e^m}$ be the error for the $\mathbf{m^{th}}$ predictor trained through bagging and e^{avg} be the error of the ensemble. If

$E[e^m] = 0$ (unbiased) and

$E[e^m e^k] = E[e^m]E[e^k]$ (uncorrelated) then..

$$E[e^{avg}] = \frac{1}{M}\frac{1}{M}\sum E[e^m]$$

The expected error of the average is a faction of the average expected error of the predictors!

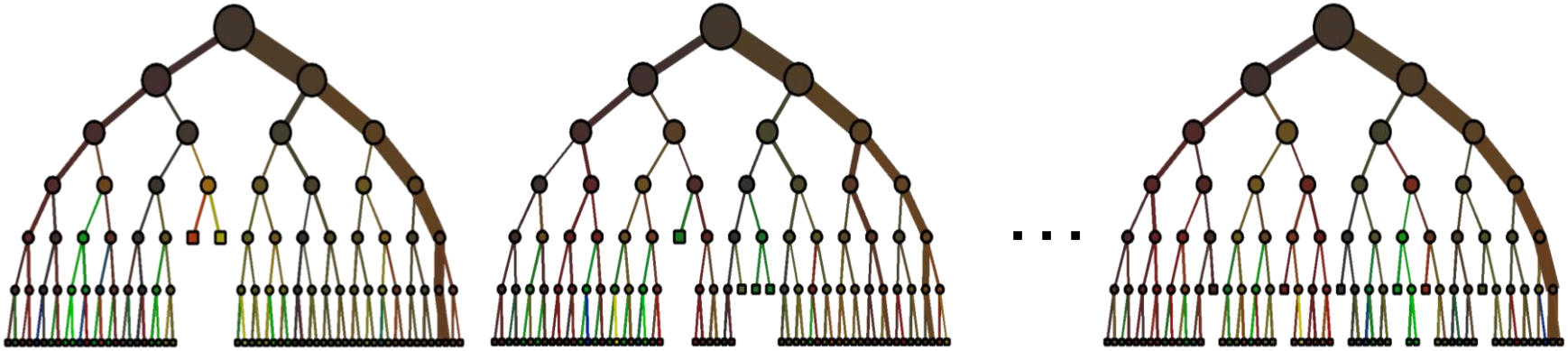
When To Use Bagging

In practice, completely uncorrelated predictors don't really happen, but there also wont likely be perfect correlation either, so bagging may still help!

Use bagging when…

- … you have overfit sophisticated learners (averaging lowers variance)
- … you have a somewhat reasonably sized dataset
- … you want an extra bit of performance from your models

Example: Decision Forests



We've seen that single decision trees can easily overfit!

- Train a M trees on different samples of the data and call it a forest.

Uncorrelated errors result in better ensemble performance. Can we force this?

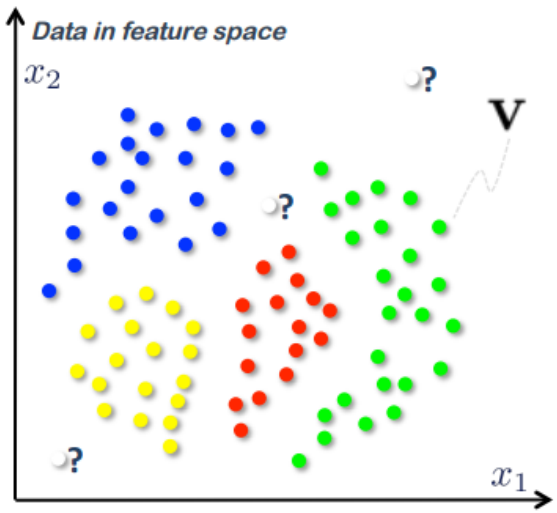
- Could assign trees random max depths
- Could only give each tree a random subset of the splits
- Some work to optimize for no correlation as part of the object!

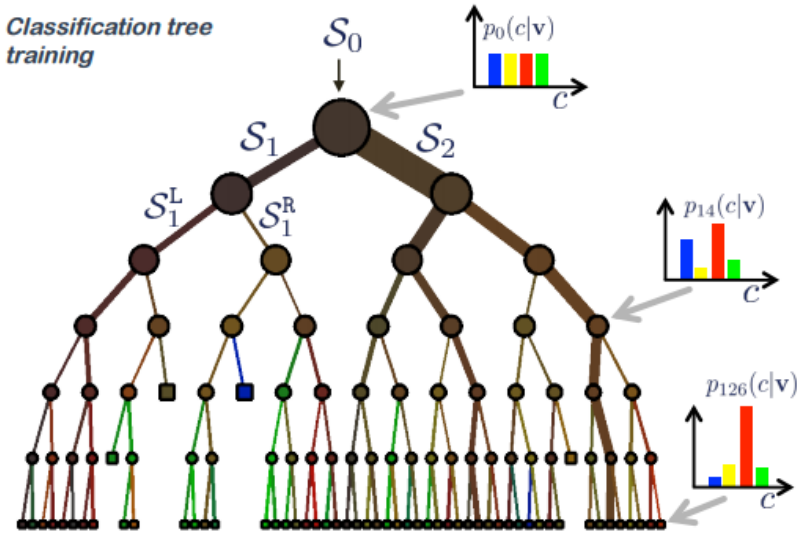
Random Forests

- Ensemble method specifically designed for decision tree classifiers
- Introduce two sources of randomness: "Bagging" and "Random input vectors"
  - **Bagging method:** each tree is grown using a bootstrap sample of training data
  - **Random vector method:** At each node, best split is chosen from a random sample of m attributes instead of all attributes

slide by David Sontag

Classification tree





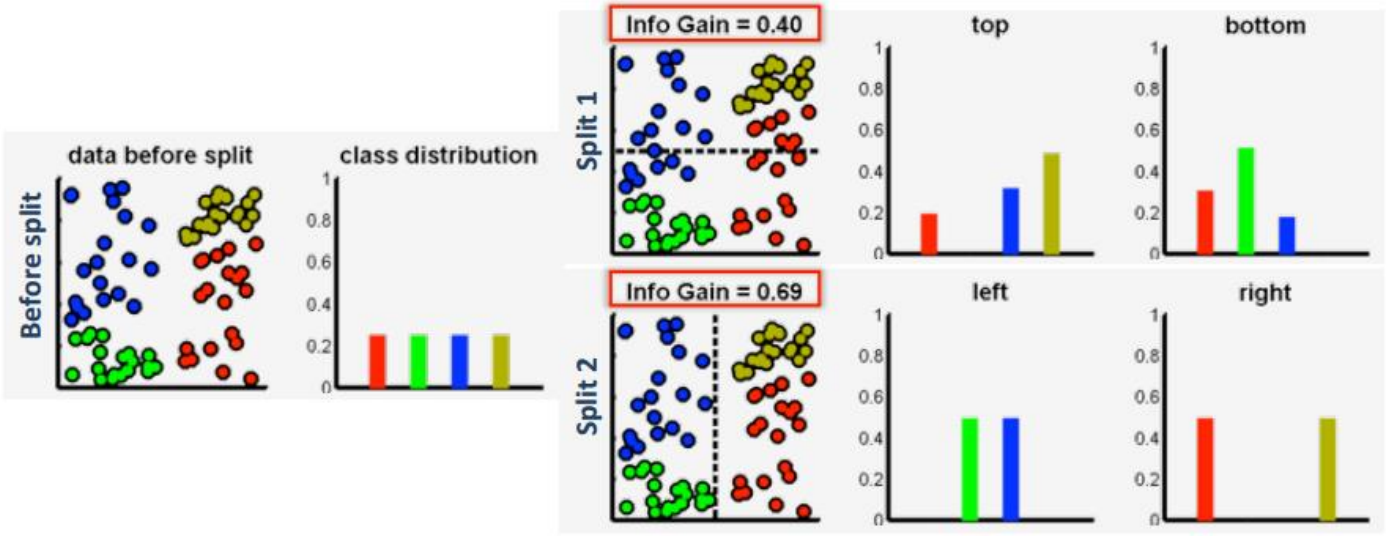
A generic data point is denoted by a vector $\mathbf{v} = (x_1, x_2, \cdots, x_d)$

$\mathcal{S}_j = \mathcal{S}_j^{\mathrm{L}} \cup \mathcal{S}_j^{\mathrm{R}}$

[Criminisi et al., 2011]

slide by Nando de Freitas

Use information gain to decide splits

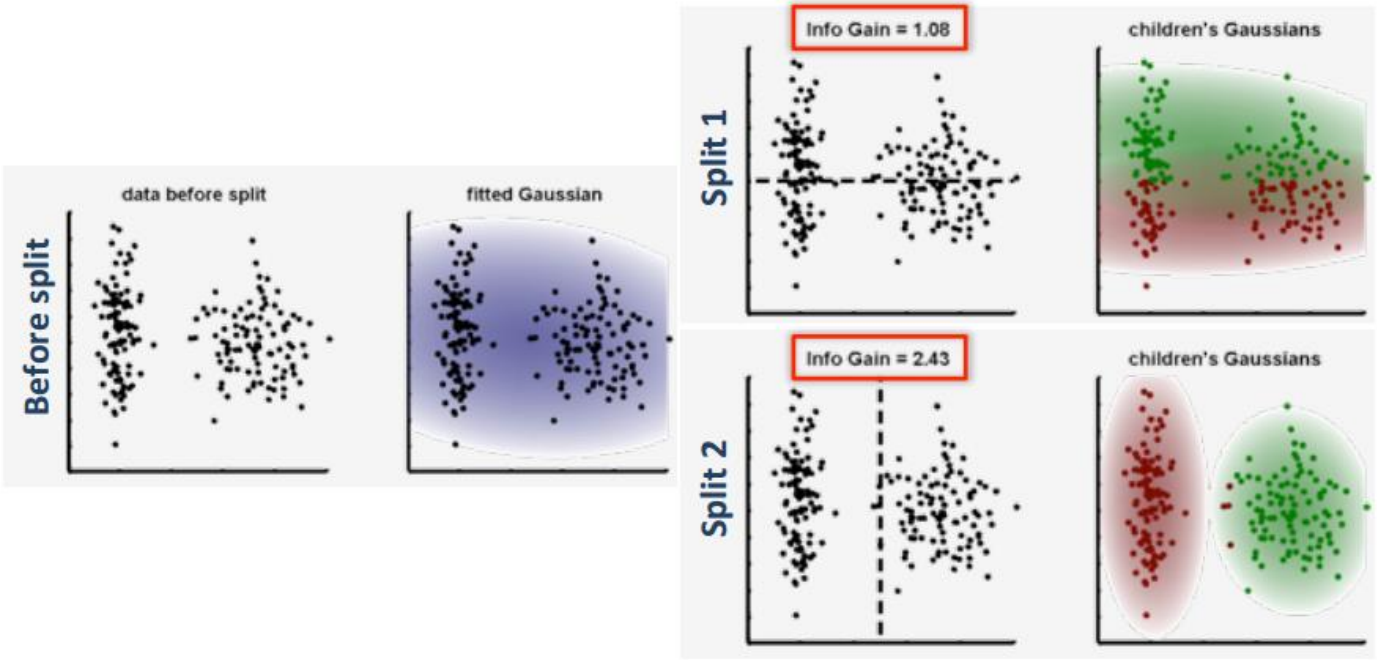


$$I_j = H(\mathcal{S}_j) - \sum_{i \in \{L,R\}} \frac{|\mathcal{S}_j^i|}{|\mathcal{S}_j|} H(\mathcal{S}_j^i)$$

[Criminisi et al., 2011]

slide by Nando de Freitas

Advanced: Gaussian information gain to decide splits



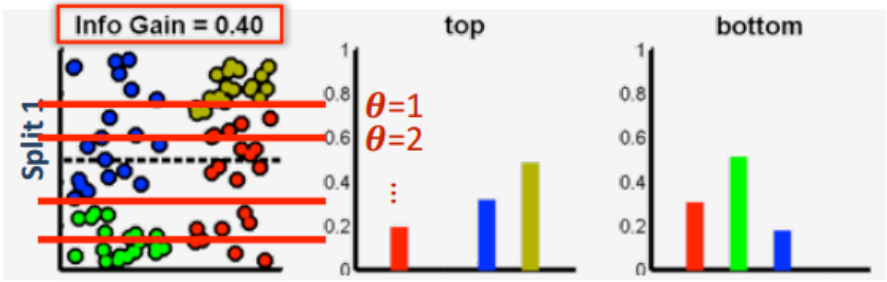
$$H(\mathcal{S}) = \frac{1}{2}\log\left((2\pi e)^d |\Lambda(\mathcal{S})|\right)$$

[Criminisi et al., 2011]

slide by Nando de Freitas

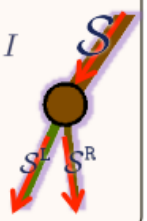
Each split node j is associated with a binary split function

$$h(\mathbf{v}, \boldsymbol{\theta}) \in \{\texttt{true}, \texttt{false}\}$$



Split node (train)

$$\boldsymbol{\theta}_j = \arg\max_{\boldsymbol{\theta} \in \mathcal{T}_j} I$$

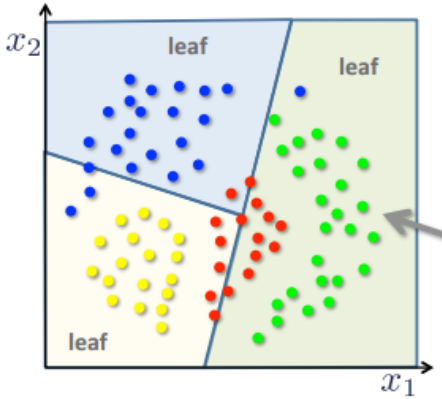


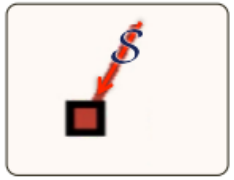
Split node (test)

Node weak learner

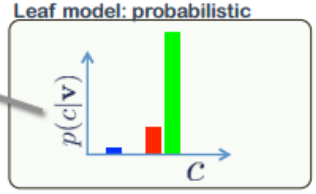
$$h(\mathbf{v}, \boldsymbol{\theta}_j)$$

$$\boldsymbol{\theta} \in \mathcal{T}_j$$





Leaf model: probabilistic



$$I_j = H(\mathcal{S}_j) - \sum_{i \in \{L,R\}} \frac{|\mathcal{S}_j^i|}{|\mathcal{S}_j|} H(\mathcal{S}_j^i)$$

[Criminisi et al., 2011]

slide by Nando de Freitas

Random Forests algorithm

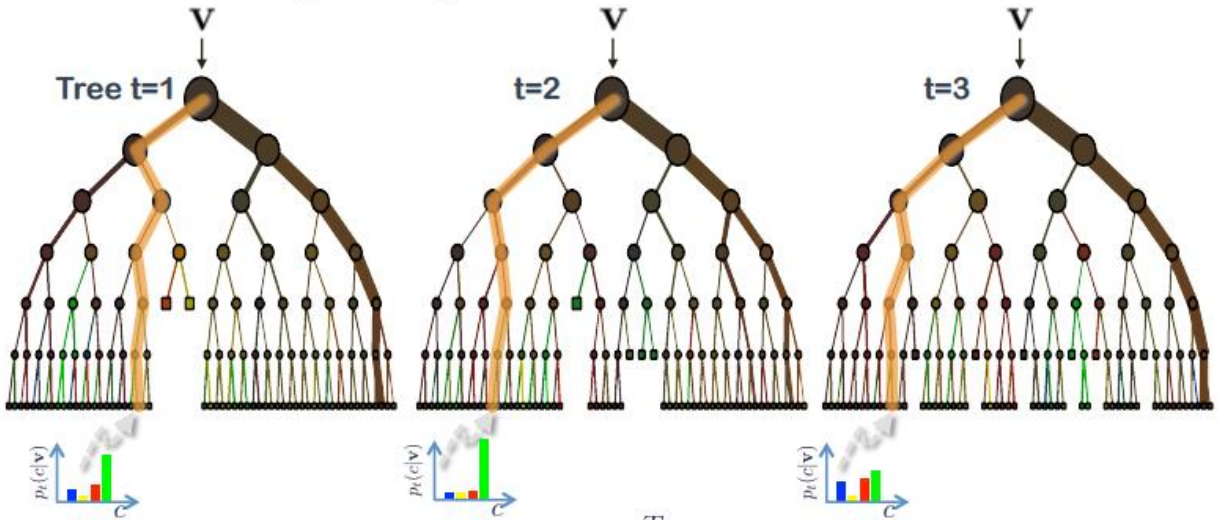
1. For $b = 1$ to B:
   (a) Draw a bootstrap sample $\mathbf{Z}^*$ of size N from the training data.
   (b) Grow a random-forest tree T_b to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} is reached.
      i. Select m variables at random from the p variables.
      ii. Pick the best variable/split-point among the m.
      iii. Split the node into two daughter nodes.
2. Output the ensemble of trees $\{T_b\}_1^B$.

slide by Nando de Freitas

[From the book of Hastie, Friedman and Tibshirani]

Building a forest (ensemble)

In a forest with T trees we have $t \in \{1, \cdots, T\}$. All trees are trained independently (and possibly in parallel). During testing, each test point $\mathbf{v}$ is simultaneously pushed through all trees (starting at the root) until it reaches the corresponding leaves.

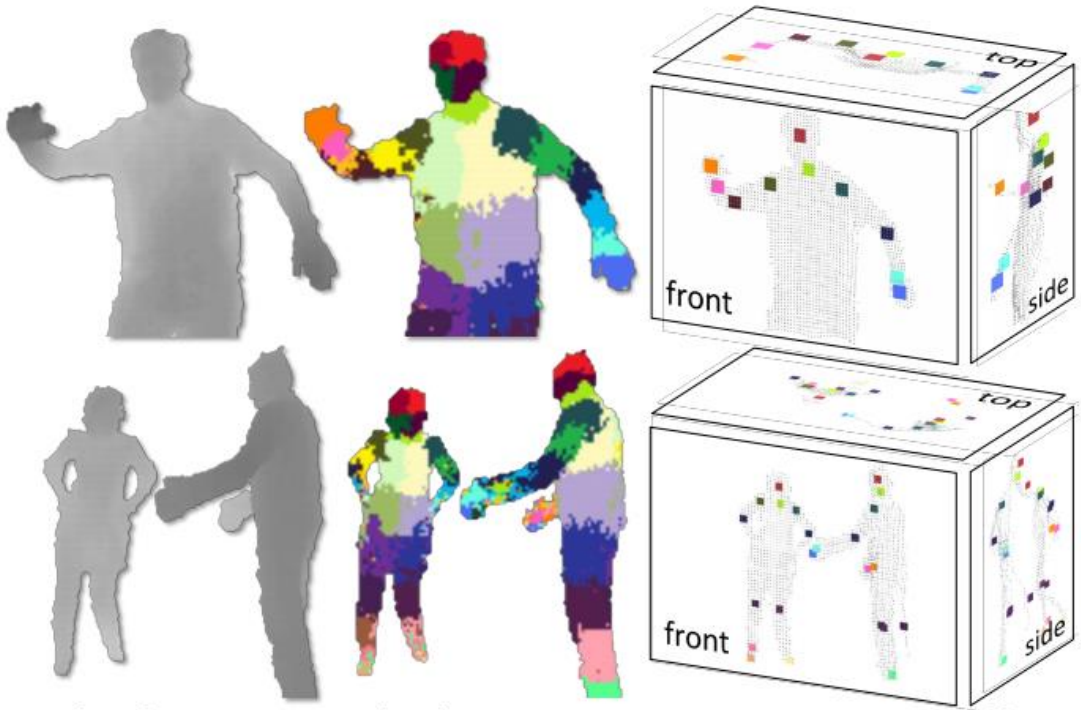


$$p(c|\mathbf{v}) = \frac{1}{T}\sum_{t}^{T} p_t(c|\mathbf{v})$$

slide by Nando de Freitas

Random Forests and the Kinec

Random Forests and the Kinect



[Jamie Shotton et al., 2011]

adopted from Nando de Freitas

Random Forests and the Kinect

- Use computer graphics to generate plenty of data

Real-Time Human Pose Recognition in Parts from Single Depth Images

CVPR 2011

Jamie Shotton, Andrew Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, Andrew Blake
Microsoft Research Cambridge & Xbox Incubation

adopted from Nando de Freitas

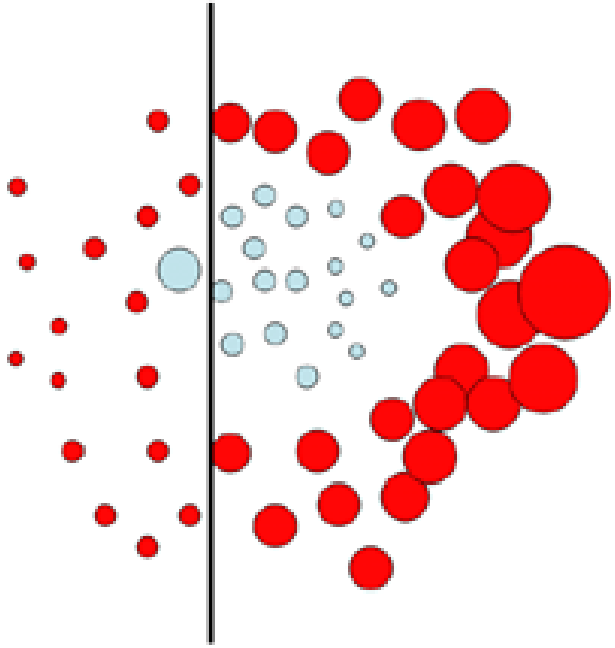
[Jamie Shotton et al., 2011]

A Note From Statistics

Bagging is a general method to reduce/estimate the variance of an estimator.

- Looking at the distribution of a estimator from multiple resamples of the data can give confidence intervals and bounds on that estimator.
- Typically just called Bootstrapping in this context.

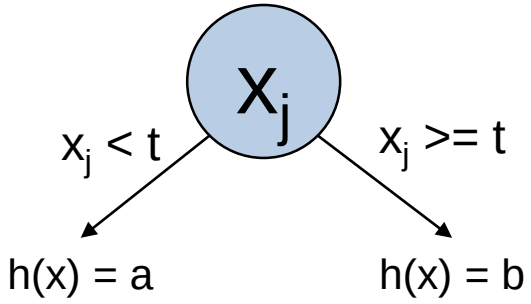
Boosting

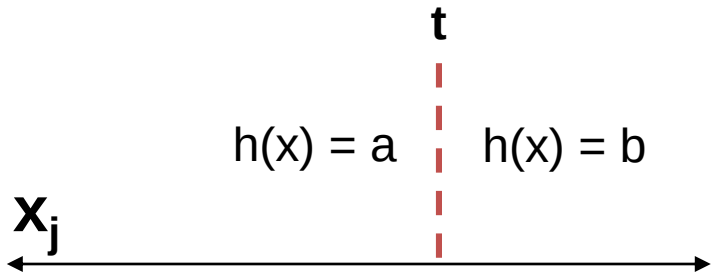


Core Idea: Combine multiple weak learners to reduce error/bias by reweighting hard examples!

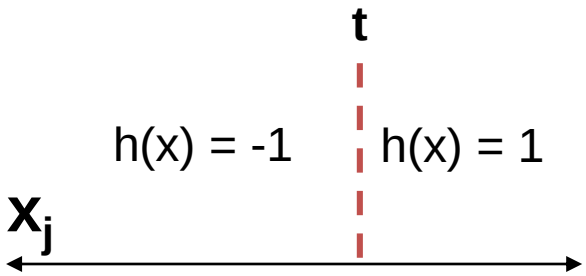
Some Intuition About Boosting

Consider a weak learner h(x), for example a decision stump:

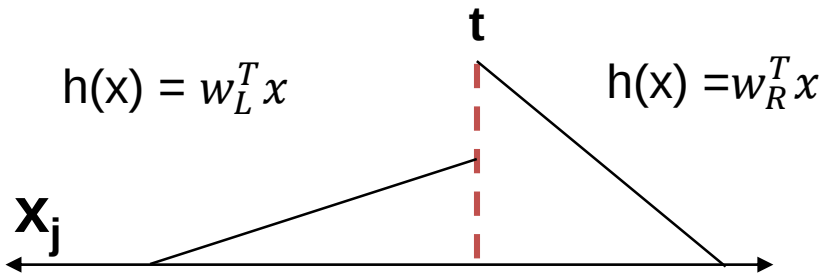




Example for binary classification:

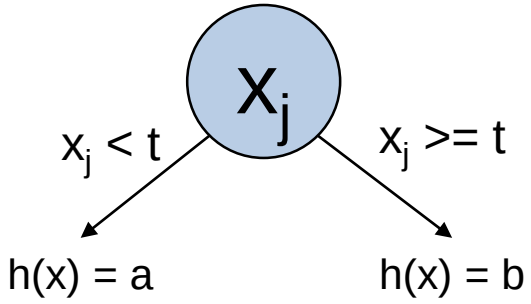


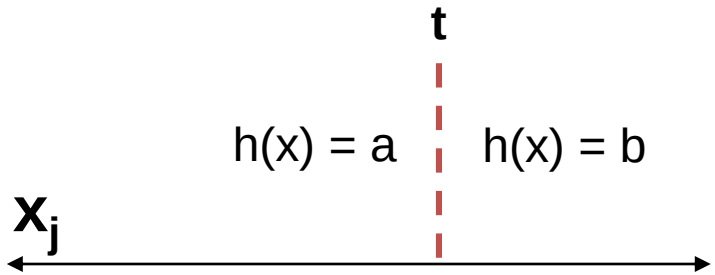
Example for regression:



Some Intuition About Boosting

Consider a weak learner h(x), for example a decision stump:





This learner will make mistakes often but what if we combine multiple to combat these errors such that our final predictor is:

$$f(x) = \alpha_1 h_1(x) + \alpha_2 h_2(x) + \cdots + \alpha_{M-1} h_{m-1}(x) + \alpha_M h_M(x)$$

This is a big optimization problem now!!

$$\min_{\substack{\alpha_{1,\ldots,\alpha_M} \\ h_i \in \boldsymbol{H}}} \frac{1}{N} \sum_{\boldsymbol{i}} \boldsymbol{L}(y_i, \alpha_1 h_1(x) + \alpha_2 h_2(x) + \cdots + \alpha_{M-1} h_{m-1}(x) + \alpha_M h_M(x))$$

Boosting will do this greedily, training one classifier at a time to correct the errors of the existing ensemble

Boosting Algorithm [Schapire, 1989]

- Pick a class of weak learners $\mathcal{H} = \{h \mid h : X \to Y\}$

- You have a black box that picks best weak learning
  - unweighted sum $h^* = \underset{h \in \mathcal{H}}{\operatorname{argmin}} \sum_i L\left(y_i, h(x_i)\right)$
  - weighted sum $h^* = \underset{h \in \mathcal{H}}{\operatorname{argmin}} \sum_i w_i L\left(y_i, h(x_i)\right)$

- On each iteration *t*
  - Compute error based on current ensemble $f_{t-1}(x_i) = \sum_{t'=1}^{t} \alpha_{t'} h_{t'}(x_i)$
  - Update weight of each training example based on it's error.
  - Learn a predictor h_t and strength for this predictor α_t

- Update ensemble: $f_t(x) = f_{t-1} + \alpha_t h_t(x)$

Boosting Demo

- Demo
  - Matlab demo by Antonio Torralba
  - http://people.csail.mit.edu/torralba/shortCourseRLOC/boosting/boosting.html

Boosting Ideas

- Main idea: use weak learner to create strong learner.
- Ensemble method: combine base classifiers returned by weak learner.
- Finding simple relatively accurate base classifiers often not hard.
- But, how should base classifiers be combined?

slide by Mehryar Mohri

Boosting: Intuition

- Instead of learning a single (weak) classifier, learn many weak classifiers that are good at different parts of the input space
- Output class: (Weighted) vote of each classifier
  - Classifiers that are most "sure" will vote with more conviction
  - Classifiers will be most "sure" about a particular part of the space
  - On average, do better than single classifier!
- But how do you???
  - force classifiers to learn about different parts of the input space?
  - weigh the votes of different classifiers?

slide by Aarti Singh & Barnabas Poczos

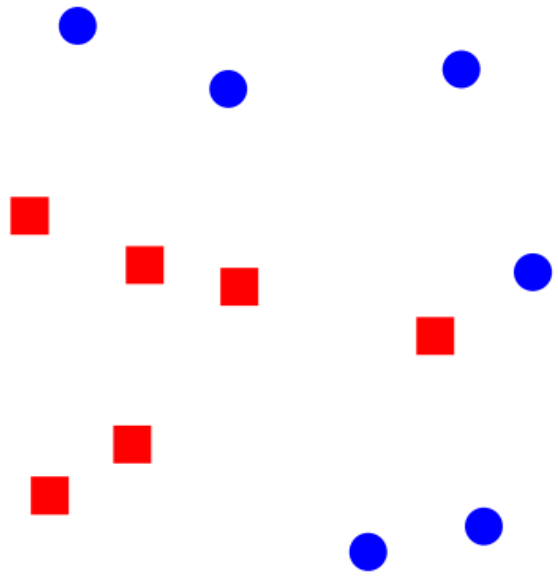
Boosting [Schapire, 1989]

- **Idea:** given a weak learner, run it multiple times on (reweighted) training data, then let the learned classifiers vote
- On each iteration t:
  - weight each training example by how incorrectly it was classifiec
  - Learn a hypothesis – h_t
  - A strength for this hypothesis – a_t
- Final classifier:
  - A linear combination of the votes of the different classifiers weighted by their strength $H(X) = \text{sign}\left(\sum \alpha_t h_t(X)\right)$
- **Practically useful**
- **Theoretically interesting**

slide by Aarti Singh & Barnabas Poczos

Boosting: Intuition

- Want to pick weak classifiers that contribute something to the ensemble



Greedy algorithm: for $m=1,...,M$

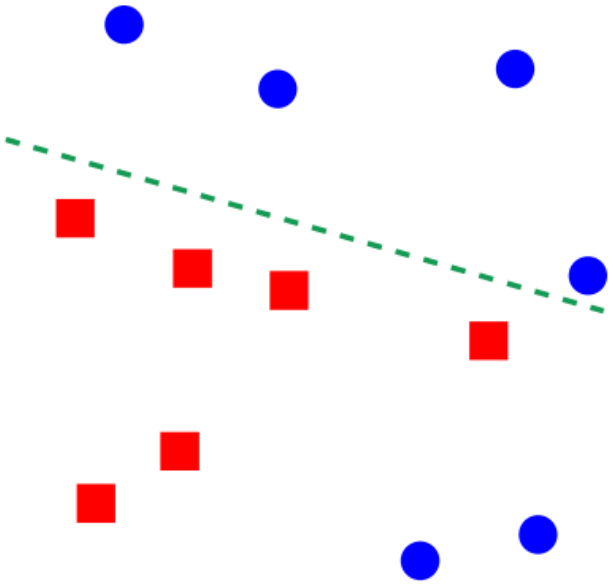
- Pick a weak classifier h_m
- Adjust weights: misclassified examples get "heavier"
- α_m set according to weighted error of h_m

[Source: G. Shakhnarovich]

slide by Raquel Urtasun

Boosting: Intuition

- Want to pick weak classifiers that contribute something to the ensemble



[Source: G. Shakhnarovich]

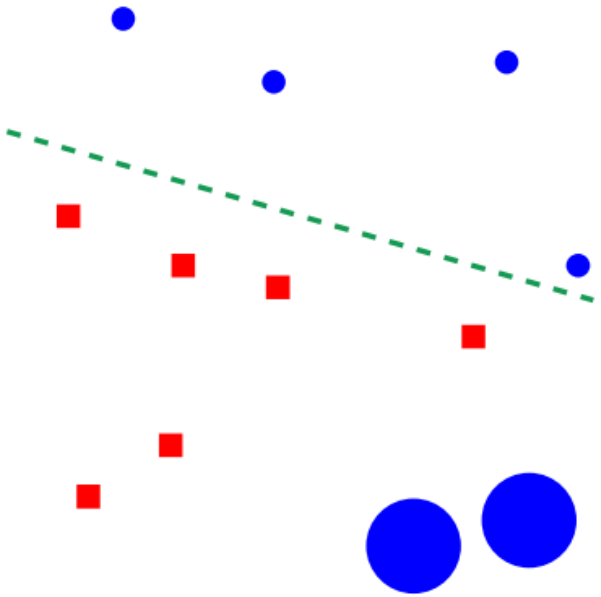
Greedy algorithm: for $m=1,...,M$

- Pick a weak classifier h_m
- Adjust weights: misclassified examples get "heavier"
- α_m set according to weighted error of h_m

slide by Raquel Urtasun

Boosting: Intuition

- Want to pick weak classifiers that contribute something to the ensemble



Greedy algorithm: for $m=1,...,M$

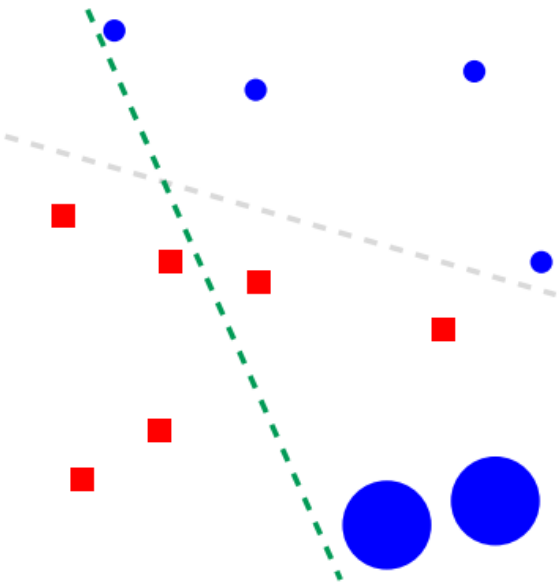
- Pick a weak classifier h_m
- Adjust weights: misclassified examples get "heavier"
- α_m set according to weighted error of h_m

[Source: G. Shakhnarovich]

slide by Raquel Urta

Boosting: Intuition

- Want to pick weak classifiers that contribute something to the ensemble



Greedy algorithm: for $m=1,...,M$

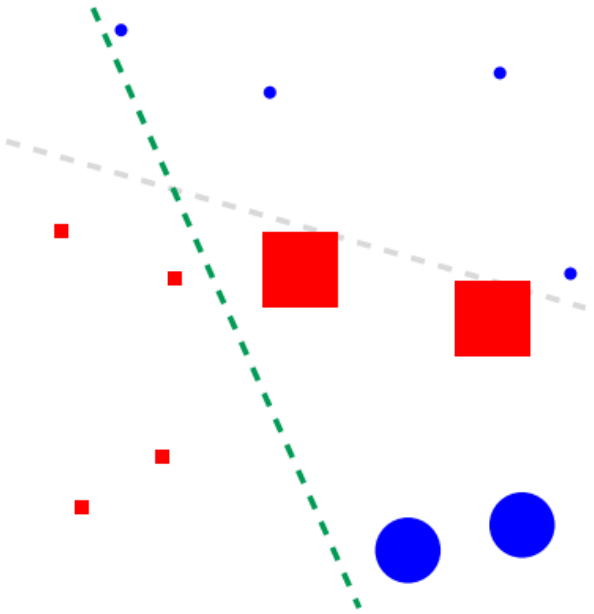
- Pick a weak classifier h_m
- Adjust weights: misclassified examples get "heavier"
- α_m set according to weighted error of h_m

[Source: G. Shakhnarovich]

slide by Raquel Urtasun

Boosting: Intuition

- Want to pick weak classifiers that contribute something to the ensemble



[Source: G. Shakhnarovich]

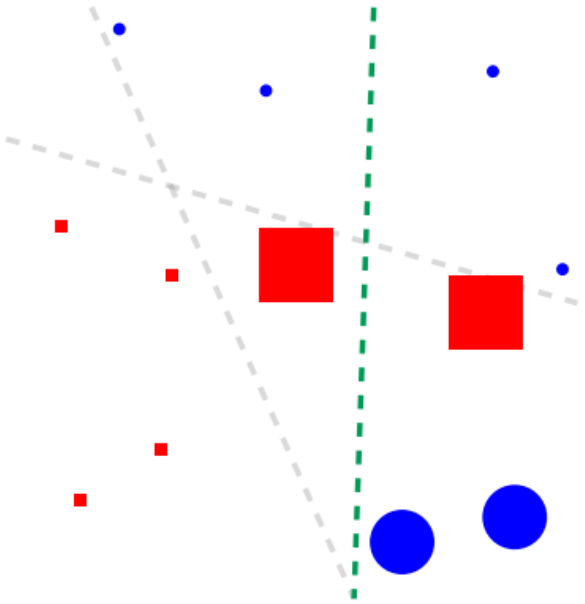
Greedy algorithm: for $m=1,...,M$

- Pick a weak classifier h_m
- Adjust weights: misclassifiec examples get "heavier"
- α_m set according to weighted error of h_m

slide by Raquel Urtasun

Boosting: Intuition

- Want to pick weak classifiers that contribute something to the ensemble



Greedy algorithm: for $m=1,...,M$

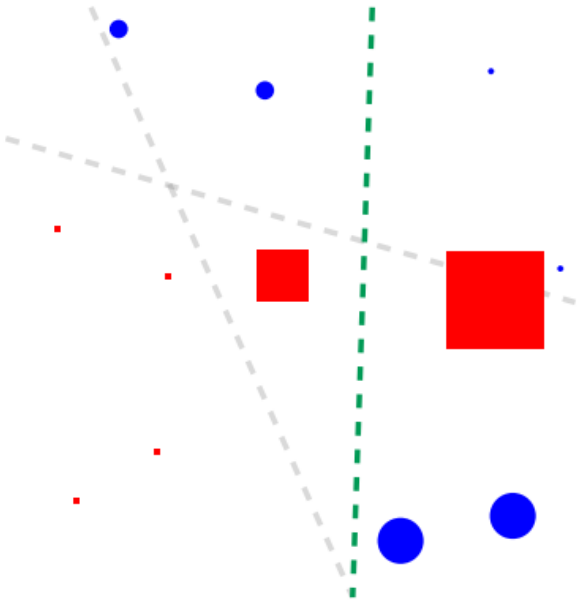
- Pick a weak classifier h_m
- Adjust weights: misclassified examples get "heavier"
- α_m set according to weighted error of h_m

[Source: G. Shakhnarovich]

slide by Raquel Urtasun

Boosting: Intuition

- Want to pick weak classifiers that contribute something to the ensemble

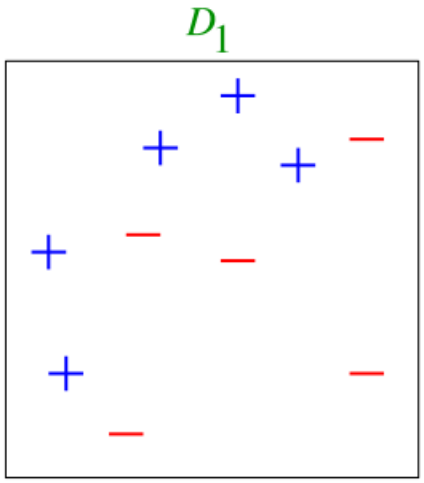


Greedy algorithm: for $m=1,...,M$

- Pick a weak classifier h_m
- Adjust weights: misclassified examples get "heavier"
- α_m set according to weighted error of h_m

[Source: G. Shakhnarovich]

Toy Example



Minimize the error

$$\epsilon_t = \Pr_{i \sim D_t}[h_t(x_i) \neq y_i]$$

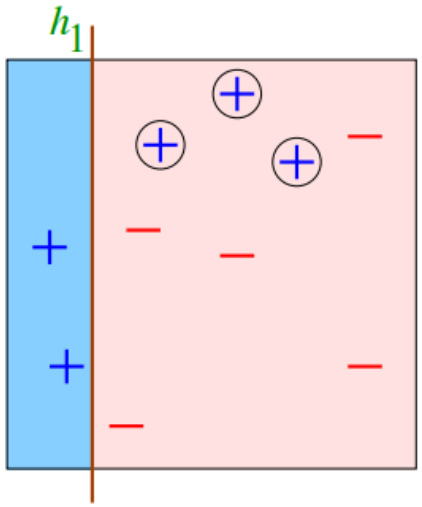
For binary h_t , typically use

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right)$$

slide by Rob Schapire

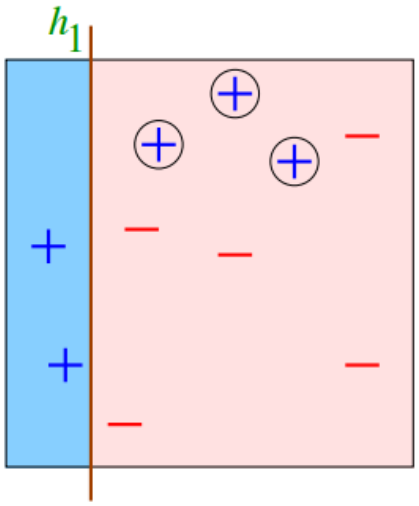
weak hypotheses = vertical or horizontal half-planes

Round 1



slide by Rob Schapire

Round 1

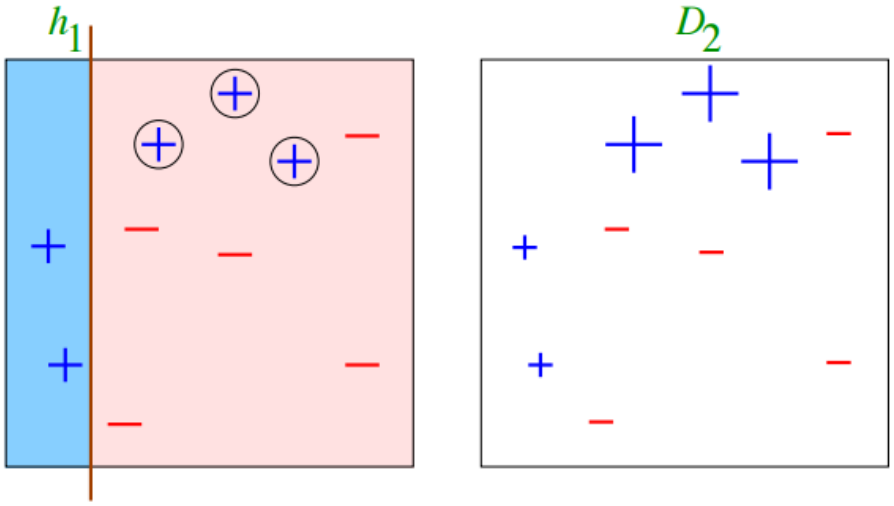


$\varepsilon_1 = 0.30$

$\alpha_1 = 0.42$

slide by Rob Schapire

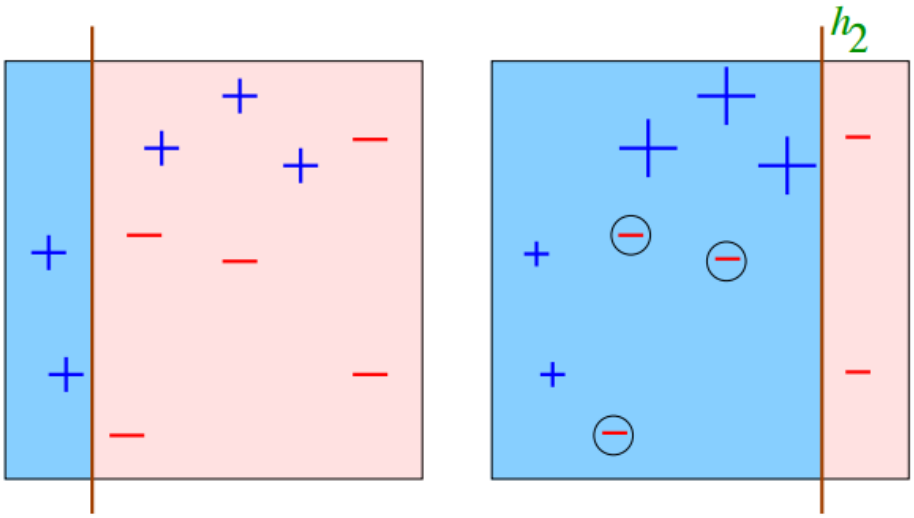
Round 1



$\varepsilon_1 = 0.30$

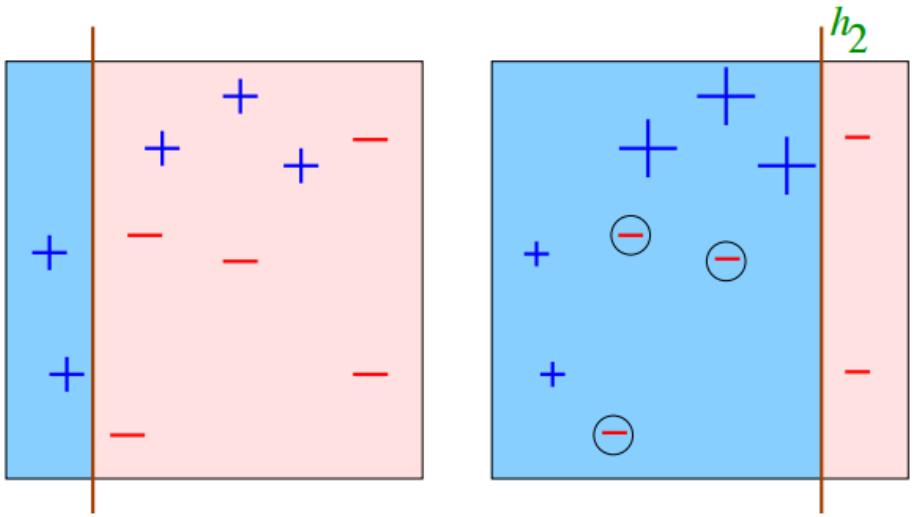
$\alpha_1 = 0.42$

Round 2



slide by Rob Schapire

Round 2

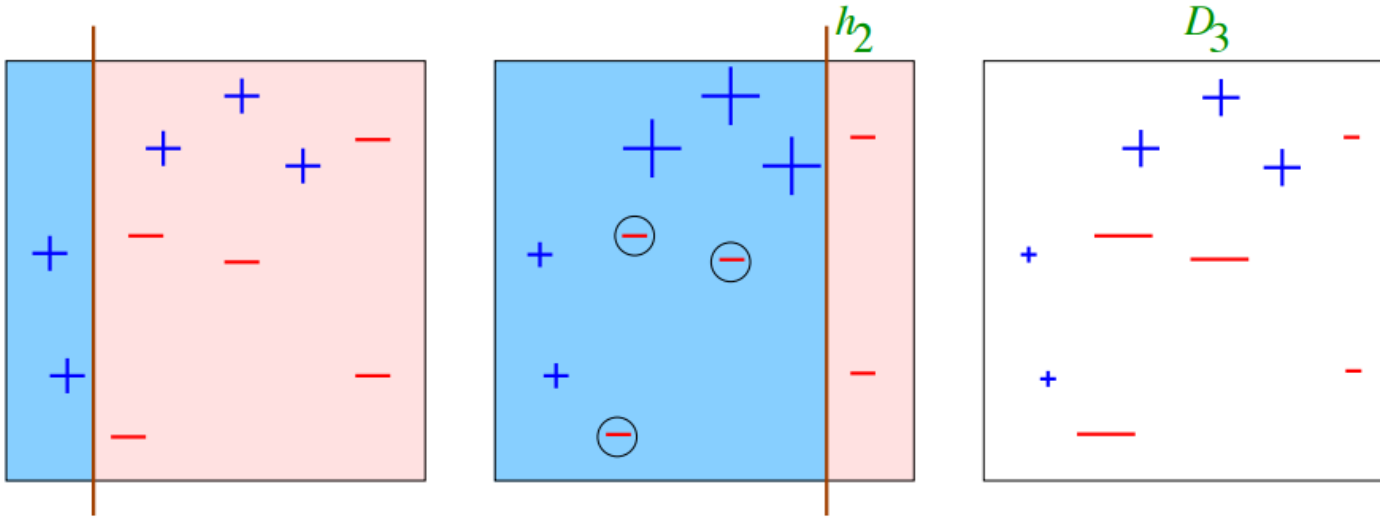


$\varepsilon_2 = 0.21$

$\alpha_2 = 0.65$

slide by Rob Schapire

Round 2

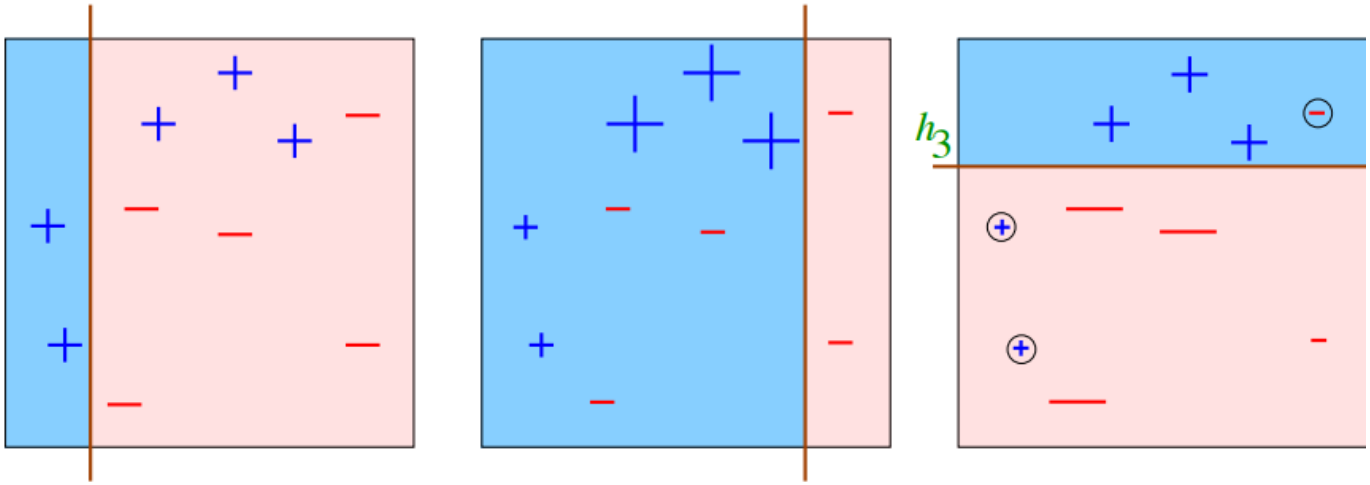


$\varepsilon_2 = 0.21$

$\alpha_2 = 0.65$

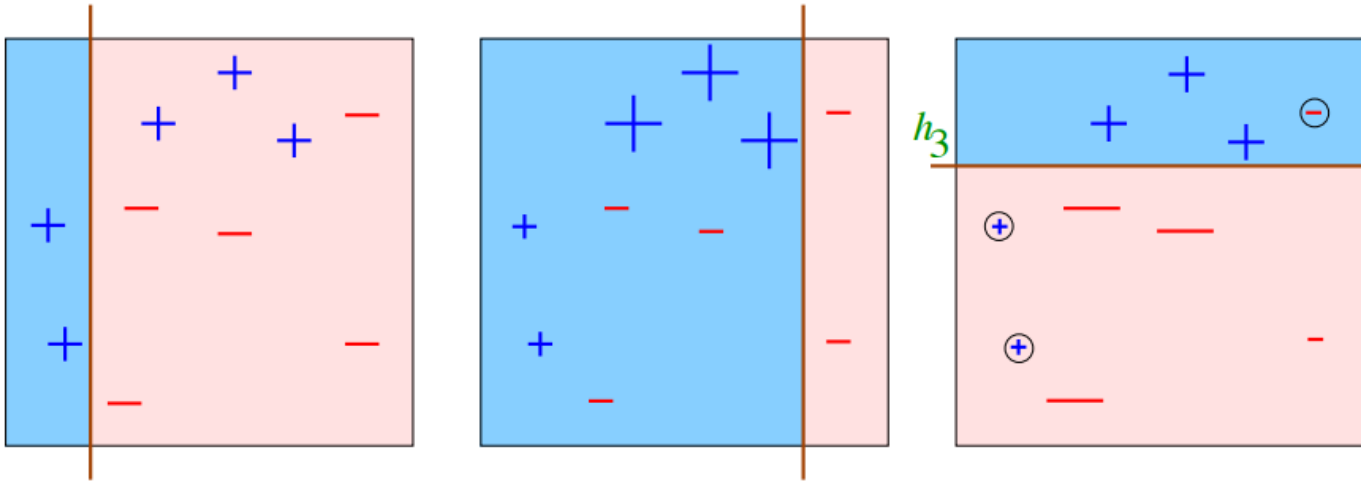
slide by Rob Schapire

Round 3



slide by Rob Schapire

Round 3



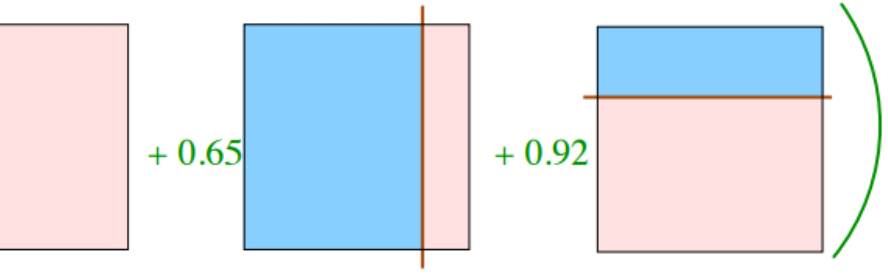
$\varepsilon_3 = 0.14$

$\alpha_3 = 0.92$

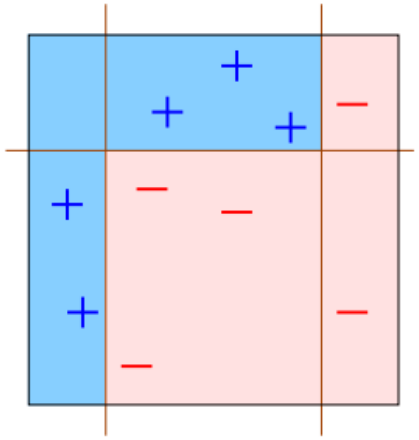
slide by Rob Schapire

Final Hypothesis

$H_{\text{final}} = \text{sign}\left(0.42 + 0.65 + 0.92\right)$



$=$



slide by Rob Schapire

Application: Detecting Faces

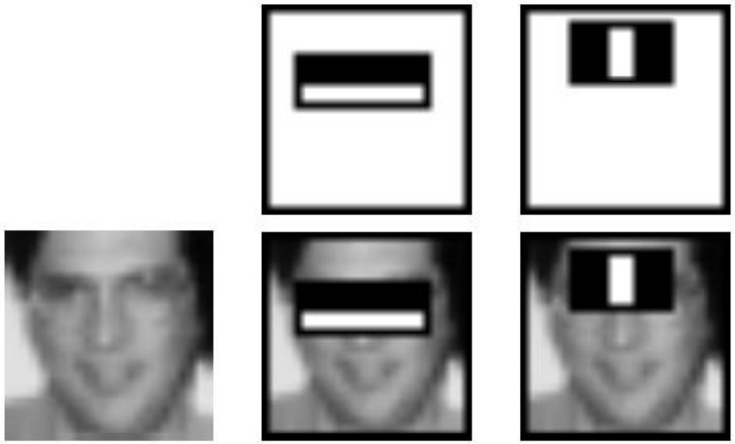
- Training Data
  - 5000 faces
    - All frontal
  - 300 million non-faces
    - 9500 non-face images

[Viola & Jones]

slide by Rob Scha

Application: Detecting Faces

- **Problem:** find faces in photograph or movie
- **Weak classifiers:** detect light/dark rectangle in image

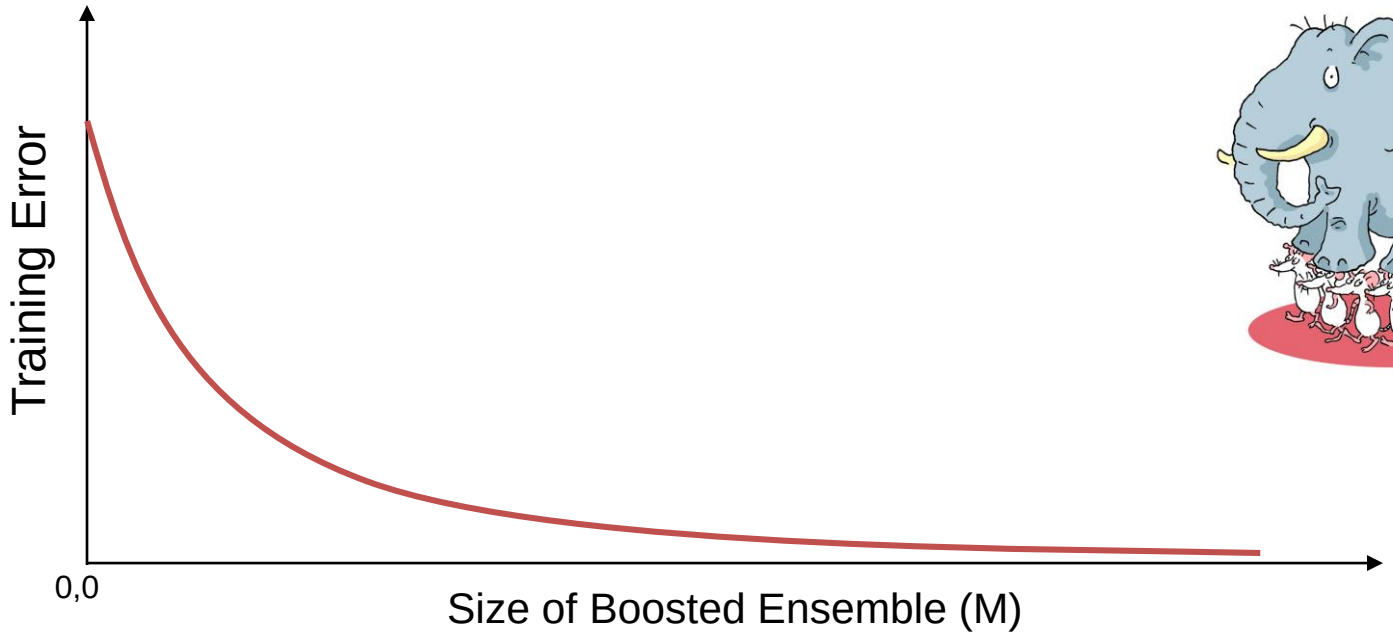


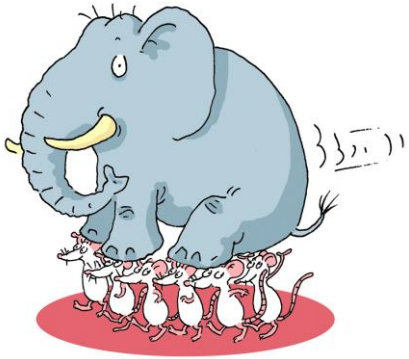
- Many clever tricks to make extremely fast and accurate

[Viola & Jones]

slide by Rob Schapire

Boosting: Weak to Strong





As we add more boosted learners to our ensemble, error approaches zero (in the limit)

- need to decided when to stop based on a validation set
- don’t use this on already overfit strong learners, will just become worse

Boosting Algorithm [Schapire, 1989]

- Pick a class of weak learners $\mathcal{H} = \{h \mid h : X \to Y\}$

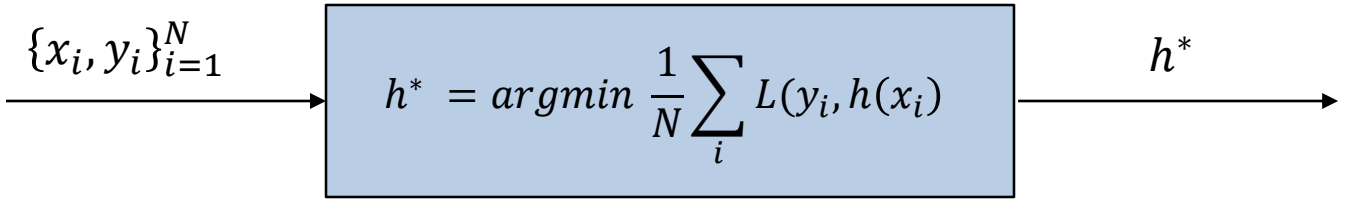
- You have a black box that picks best weak learning
  - unweighted sum $h^* = \underset{h \in \mathcal{H}}{\operatorname{argmin}} \sum_i L\left(y_i, h(x_i)\right)$
  - weighted sum $h^* = \underset{h \in \mathcal{H}}{\operatorname{argmin}} \sum_i w_i L\left(y_i, h(x_i)\right)$

- On each iteration *t*
  - Compute error based on current ensemble $f_{t-1}(x_i) = \sum_{t'=1}^{t} \alpha_{t'} h_{t'}(x_i)$
  - **Update weight of each training example based on it's error.**
  - Learn a predictor h_t and strength for this predictor α_t

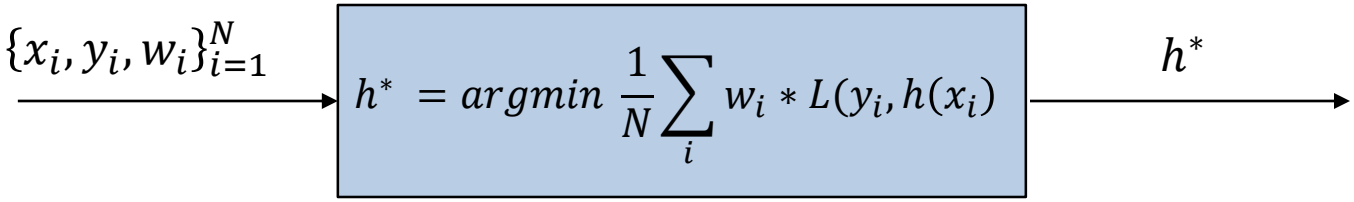
- Update ensemble: $f_t(x) = f_{t-1} + \alpha_t h_t(x)$

Boosting Algorithm [Schapire, 1989]

We've assumed we have some tools to find optimal learners, either



or



To train the t[th] predictor, our job is to express the optimization for the new predictor in one of these forms

$$\min_{\substack{\alpha_1,\ldots,\alpha_M \\ h_i \in \boldsymbol{H}}} \frac{1}{N} \sum_{\boldsymbol{i}} \boldsymbol{L}(y_i, f_{t-1}(x) + \alpha_t h_t(x))$$

Typically done by either changing y_i or w_i depending on L.

Types of Boosting

Loss Name	Loss Formula	Boosting Name
Regression: Squared Loss	$(y - f(x))^2$	L2Boosting
Regression: Absolute Loss	$\lvert y - f(x) \rvert$	Gradient Boosting
Classification: Exponential Loss	$e^{-yf(x)}$	AdaBoost
Classification: Log/Logistic Loss	$\log\left(1 + e^{-yf(x)}\right)$	LogitBoost

L2 Boosting

Loss Name	Loss Formula	Boosting Name
Regression: Squared Loss	$(y - f(x))^2$	L2Boosting

- Algorithm
  - On Board

Adaboost

Loss Name	Loss Formula	Boosting Name
Classification: Exponential Loss	$e^{-yf(x)}$	AdaBoost

- Algorithm
  - You will derive in HW4!

What you should know

- Voting/Ensemble methods
- Bagging
  - How to sample
  - Under what conditions is error reduced
- Boosting
  - General algorithm
  - L2Boosting derivation
  - Adaboost derivation (from HW4)