THE UNIVERSITY of ADELAIDE

Ensemble Learning

Lingqiao Liu
University of Adelaide

Some slides borrowed from Rama Ramakrishnan and Rob Schapire etc.

Outlines

- Ensemble methods overview
- Random forest
- Bagging
- Boosting

Outlines

- Ensemble methods overview
- Random forest
- Bagging
- Boosting

What is Ensemble learning?

- According to the dictionary, the word ‘ensemble’ means

  ‘a group of items viewed as a whole rather than individually’

- In machine learning, ensemble learning is a learning technique to learn and combine multiple predictive models
  - The models can be anything
  - More diverse the better.
  - Combine the predictions:
    - Majority voting/Weighted Majority voting
    - Average/weighted average
    - …

Ensemble learning is powerful

- Ensemble approach is often the essential strategy for wining machine learning competition!

Machine learning competition with a $1 million prize

Leaderboard

Display top 20 leaders.

Rank	Team Name	Best Score	% Improvement	Last Submit Time
1	The Ensemble	0.8553	10.10	2009-07-26 18:38:22
2	BellKor's Pragmatic Chaos	0.8554	10.09	2009-07-26 18:18:28
Grand Prize - RMSE <= 0.8563				
3	Grand Prize Team	0.8571	9.91	2009-07-24 13:07:49
4	Opera Solutions and Vandelay United	0.8573	9.89	2009-07-25 20:05:52
5	Vandelay Industries !	0.8579	9.83	2009-07-26 02:49:53
6	PragmaticTheory	0.8582	9.80	2009-07-12 15:09:53
7	BellKor in BigChaos	0.8590	9.71	2009-07-26 12:57:25
8	Dace_	0.8603	9.58	2009-07-24 17:18:43
9	Opera Solutions	0.8611	9.49	2009-07-26 18:02:08
10	BellKor	0.8612	9.48	2009-07-26 17:19:11
11	BigChaos	0.8613	9.47	2009-06-23 23:06:52
12	Feeds2	0.8613	9.47	2009-07-24 20:06:46
Progress Prize 2008 - RMSE = 0.8616 - Winning Team: BellKor in BigChaos				
13	xiangliang	0.8633	9.26	2009-07-21 02:04:40
14	Gravity	0.8634	9.25	2009-07-26 15:58:34
15	Ces	0.8642	9.17	2009-07-25 17:42:38
16	Invisible Ideas	0.8644	9.14	2009-07-20 03:26:12
17	Just a guy in a garage	0.8650	9.08	2009-07-22 14:10:42
18	Craig Carmichael	0.8656	9.02	2009-07-25 16:00:54
19	J Dennis Su	0.8658	9.00	2009-03-11 09:41:54
20	acmehill	0.8659	8.99	2009-04-16 06:29:35
Progress Prize 2007 - RMSE = 0.8712 - Winning Team: KorBell				
Cinematch score on quiz subset - RMSE = 0.9514				

Ensemble learning

- The key is to ensure the diversity of predictive method
- Let's consider an extreme case
  - Binary classification problem
  - We ensemble 1000 classifiers, the accuracy of each classifier is 51%, that is, slightly better than random guessing
  - Those classifiers are statistically independent
  - We use majority voting to make final prediction
    - If most classifier predict class 1, then the prediction is class 1, vice versa

What is the classification accuracy of the final system?

Ensemble learning example

- If the prediction is correct, it means that more than 500 classifiers should give correct prediction
- This probability can be calculated by binomial cumulative distribution
  - [What is a cumulative Binomial probability? - Cross Validated (stackexchange.com)]

$$P = \text{binocdf}(499,1000,1-0.51) = 0.73$$

- After calculation, the final prediction accuracy can go from 51% to over 70%!
  - If the individual classifier got slightly higher accuracy, say 55%, the ensembled classifier can reach 100% accuracy

Ensemble learning

- We can see that by using a rather weak classifier, the ensemble classifier can significantly outperform the original classifier
- The key is the independence assumption
  - In practice, ensuring independence is challenging
  - we usually use different methods to ensure the individual predictors are diverse/independent as much as possible.
- How to ensure the diversity of classifiers
  - The predictor/classifier, e.g., different types of classifiers, or classifiers with different parameters
  - The dataset to train the model, e.g., classifier trained on different subset of data

Outlines

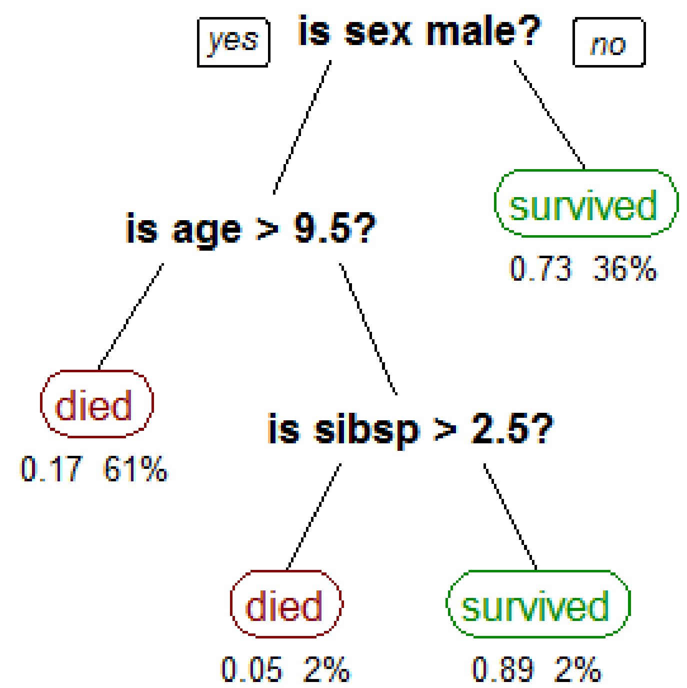
- Ensemble methods overview
- Random forest
- Bagging
- Boosting

Random forest

- Basic idea: ensemble a set of simple classifier, called decision tree, as the final classifier
- Simple but effective approach
  - Especially good for features with diverse meanings
  - Efficient in training and testing

Decision Tree: An introduction

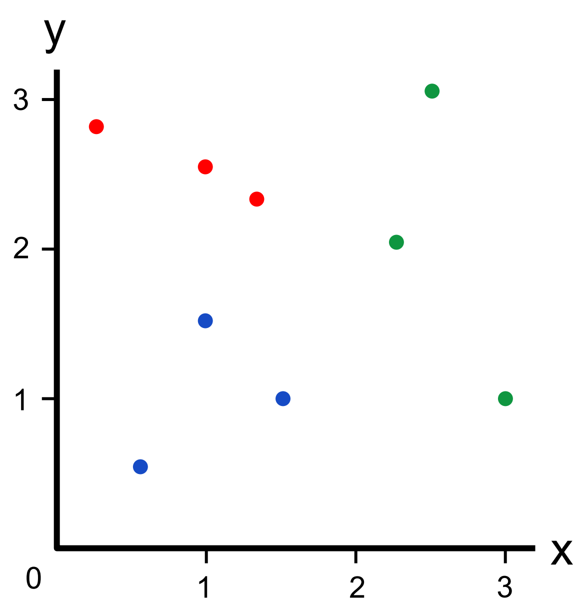
- A classic predictive model
  - A decision tree is drawn upside down with its root at the top.
  - Each non-leaf node represents a test, usually it involves only one feature
  - The leaf node represents the decision/prediction outcome
  - Example

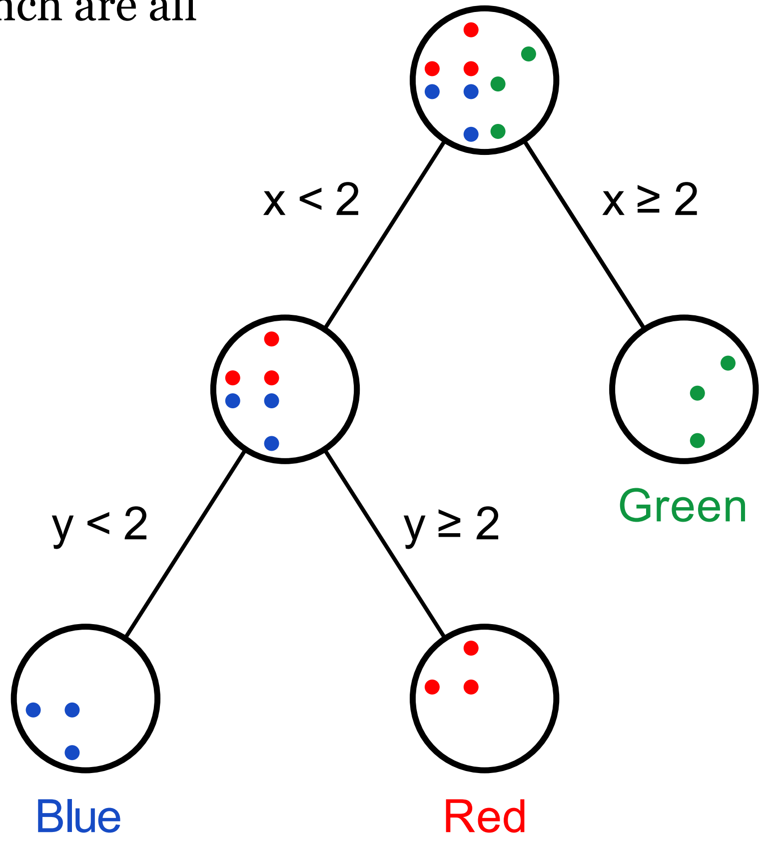


For this let's consider a very basic example that uses titanic data set for predicting whether a passenger will survive or not. Below model uses 3 features/attributes/columns from the data set, namely sex, age and sibsp (number of spouses or children along).

Decision Tree: an example

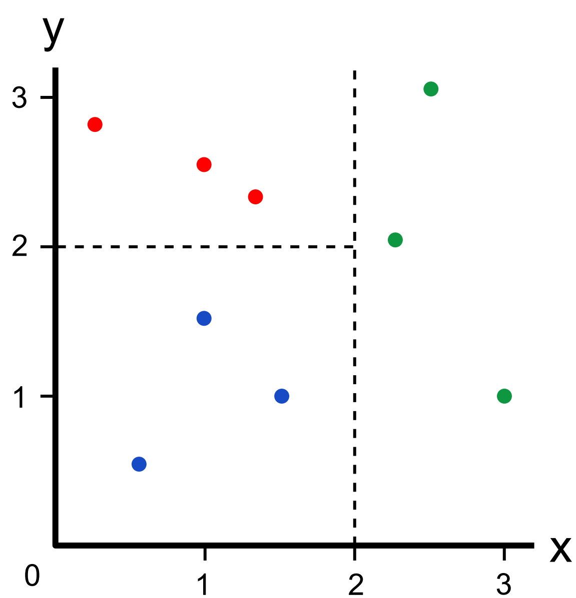
- In each node, decision tree chooses one dimension and one threshold to perform a test. Data are divided according to the outcome of the test
- The test ends when samples in one branch are all belong the same class.

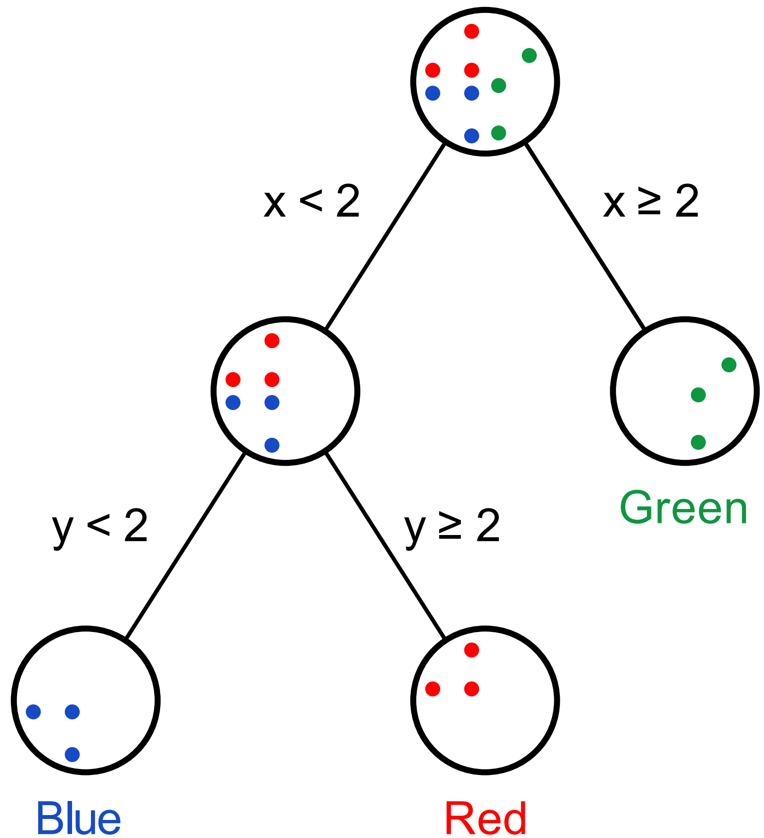




Decision Tree: an example

It is equivalent to recursively partition the feature spaces. (by vertical or horizontal lines in the 2-D case)





Decision Tree: how to build

- Intuitively, we try to find a way to divide data such that the class labels are "purer" in the divided subsets
- For each node, scan all the dimensions and try all possible thresholds, find the best one
  - Best is measured by weighted average Gini impurity of the subset in the left child and right child , the lower the better

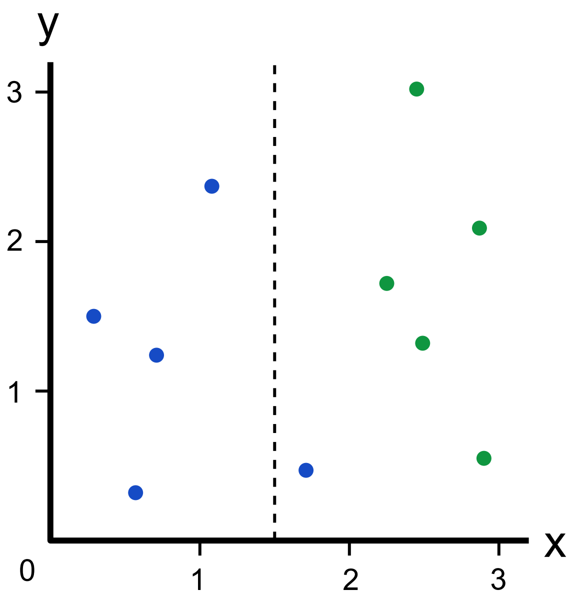
$$G = \sum_{i=1}^{C} p(i)(1 - p(i))$$

C: number of classes. $p(i)$ percentage of the i-th class samples.

Weight is calculated by the number of samples in each child

We can tell G = 0, if all samples belong to the same class

Example



$$G_{left} = 1 \times 0 = 0$$

$$G_{right} = \frac{1}{6} \times \frac{5}{6} + \frac{5}{6} \times \frac{1}{6} = 0.278$$

$$G_{ave} = 0 \times 0.4 + 0.278 * 0.6 = 0.167$$

Decision Tree

- By recursively apply the above rule, we could split nodes until the impurity can not be further reduced, i.e., all the samples in the children nodes belong to the same categories
- Decision Tree is fast to train and evaluate, because it only uses one feature each time
- Decision Tree can easily overfit the training data
  - Not stable since a mistake made in the parent node will affect the decision of its children.
  - Its classification accuracy is poor

Random forest

- Idea: build T trees instead of one, (that's why it is called forest), and introduce randomness for each tree
- Injecting randomness
  – Instead of trying all features every time we make a new decision node, we only try a random subset of the features.
- Random forest works surprisingly well in many applications
- Why use decision tree?
  – Easy to train
  – Fast to evaluate
  – Easy to inject randomness

Outlines

- Ensemble methods overview
- Random forest
- Bagging
- Boosting

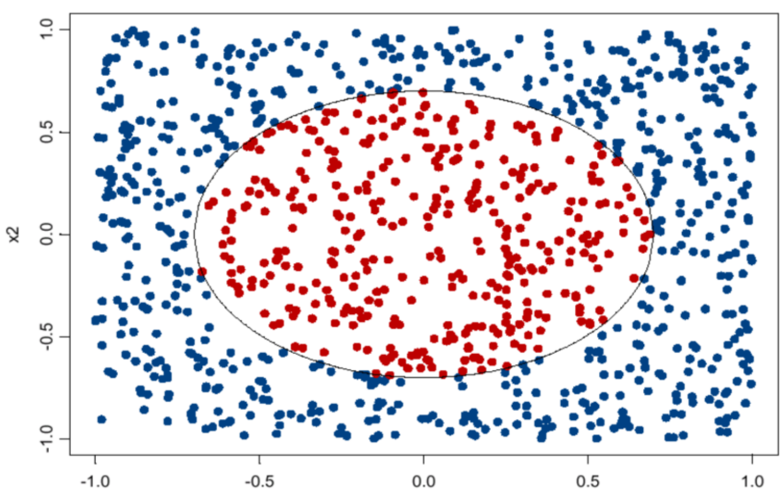
Bagging

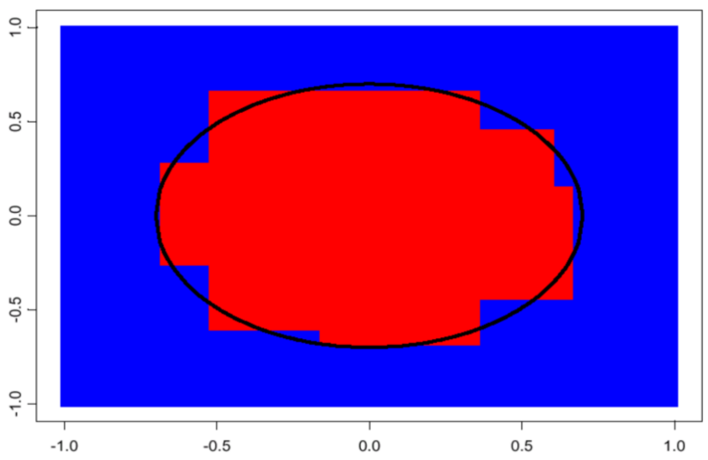
- A general method to train t diverse models
- Idea:
- given a dataset of n points:
  - Sample, **with replacement**, n training examples from the dataset.
  - Train a model on the n samples.
  - Repeat t times.

With replacement -> chance to repeat sample same samples. So each “n-sample” training set will be different

Example of Bagging

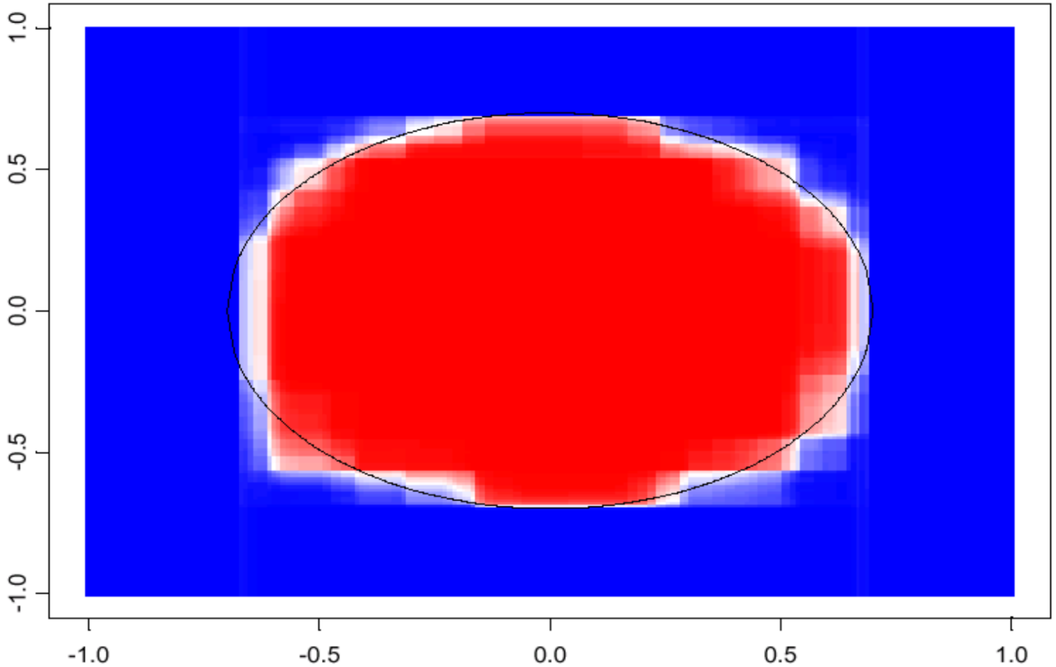
- Bagging a set of decision trees
  - Each tree is built by using decision tree algorithm, no randomness is introduced in the tree construction stage
- Decision boundary of individual trees





Example of Bagging

- Use T trees



shades of blue/red indicate strength of vote for particular classification

Outlines

- Ensemble methods overview
- Random forest
- Bagging
- Boosting

Boosting

- Bagging can be understood as a way to assign each sample different weights
- Recall that in general a machine learning algorithm try to minimize a loss function with the form

$$J = \sum_{i=1}^{N} l(\mathbf{x}_i, y_i)$$

$$\text{if } (\mathbf{x}_i, y_i) = (\mathbf{x}_j, y_j), \text{ then } l(\mathbf{x}_i, y_i) + l(\mathbf{x}_j, y_j) = 2l(\mathbf{x}_i, y_i)$$

If a sample is not selected, it is equivalent to set its weight to 0

- In bagging, the weights are randomly assigned and can only take limited number of values

Boosting

- Boosting follows a similar procedure to build a bundle of predictors
  - Build a model with a given predictor with initial weights
  - Adjust weights
  - Build a model with the updated weights
  - Adjust weights
  - Repeat T times
- The weighted is updated to make the classifier focuses on samples that are most often misclassified by previous rules
- The results are combined by weighted majority voting

Boosting

- Given a weak classifier with prediction accuracy slightly better than random, say, 55% in binary classification case, a boosting algorithm can provably construct single classifier with very high accuracy, say, 99%, given sufficient training data.
  - The weak classifier is also called weak learner
- Has many variants
  - Adaboost
  - Anyboost
  - Graident boost
  - ...

A formal definition of Adaboost (binary case)

- given training set $(x_1, y_1), \ldots, (x_m, y_m)$
- $y_i \in \{-1, +1\}$ correct label of instance $x_i \in X$
- for $t = 1, \ldots, T$:
  - construct distribution D_t on $\{1, \ldots, m\}$
  - find weak classifier ("rule of thumb")

    $$h_t : X \to \{-1, +1\}$$

    with error ϵ_t on D_t:

    $$\epsilon_t = \Pr_{i \sim D_t}[h_t(x_i) \neq y_i]$$
- output final/combined classifier H_{final}

Weight for each sample

Weighted error rate

$$Pr_{i \sim D_t}[h_t(x_i) \neq y_i] = \frac{1}{N} \sum_i D_t(i)[h_t(x_i) \neq y_i]$$

A formal definition of Adaboost (binary case)

- constructing D_{t+1}
  - $D_1(i) = 1/m$
  - given D_t and h_t:

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } y_i = h_t(x_i) \\ e^{\alpha_t} & \text{if } y_i \neq h_t(x_i) \end{cases}$$
$$= \frac{D_t(i)}{Z_t} \exp(-\alpha_t \, y_i \, h_t(x_i))$$

where $Z_t = \text{normalization factor}$
$$\alpha_t = \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right) > 0$$

- final classifier:
  - $H_{\text{final}}(x) = \text{sign}\left(\sum_t \alpha_t h_t(x)\right)$

A formal definition of Adaboost (binary case)

Weight is updated based on previous round error

Decrease

- given D_t and h_t:

$$
\begin{aligned}
D_{t+1}(i) &= \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } y_i = h_t(x_i) \\ e^{\alpha_t} & \text{if } y_i \neq h_t(x_i) \end{cases} \\
&= \frac{D_t(i)}{Z_t} \exp(-\alpha_t \, y_i \, h_t(x_i))
\end{aligned}
$$

Increase

where $Z_t = \text{normalization factor}$

$$\alpha_t = \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right) > 0$$

Putting together: an example procedure

- Start training with initial weight D_1
- Try to find a weak classifier minimizing

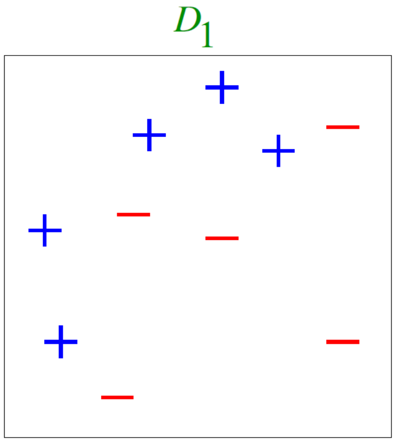
$$Pr_{i \sim D_t}[h_t(x_i) \neq y_i] = \frac{1}{N} \sum_i D_t(i)[h_t(x_i) \neq y_i]$$

- Calculate the weighted classification error ϵ_1 and calculate α_1
- Update weight to D_2 by using D_1 and α_1
- Try to find a weak classifier minimizing

$$Pr_{i \sim D_t}[h_t(x_i) \neq y_i] = \frac{1}{N} \sum_i D_t(i)[h_t(x_i) \neq y_i]$$

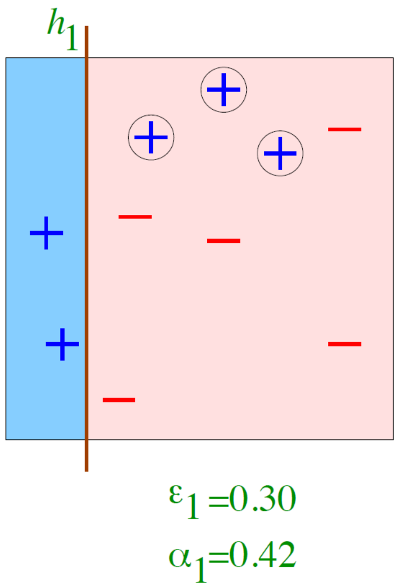
- Calculate the weighted classification error ϵ_2 and calculate α_2
- Repeat…

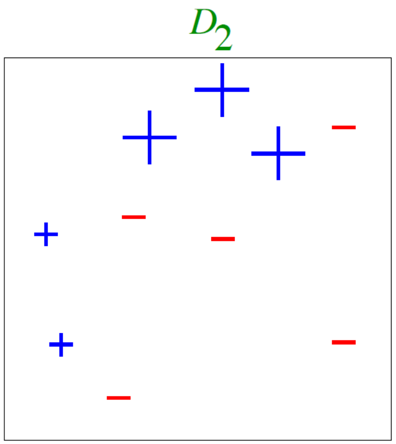
Toy example



weak classifiers = vertical or horizontal half-planes

Round 1





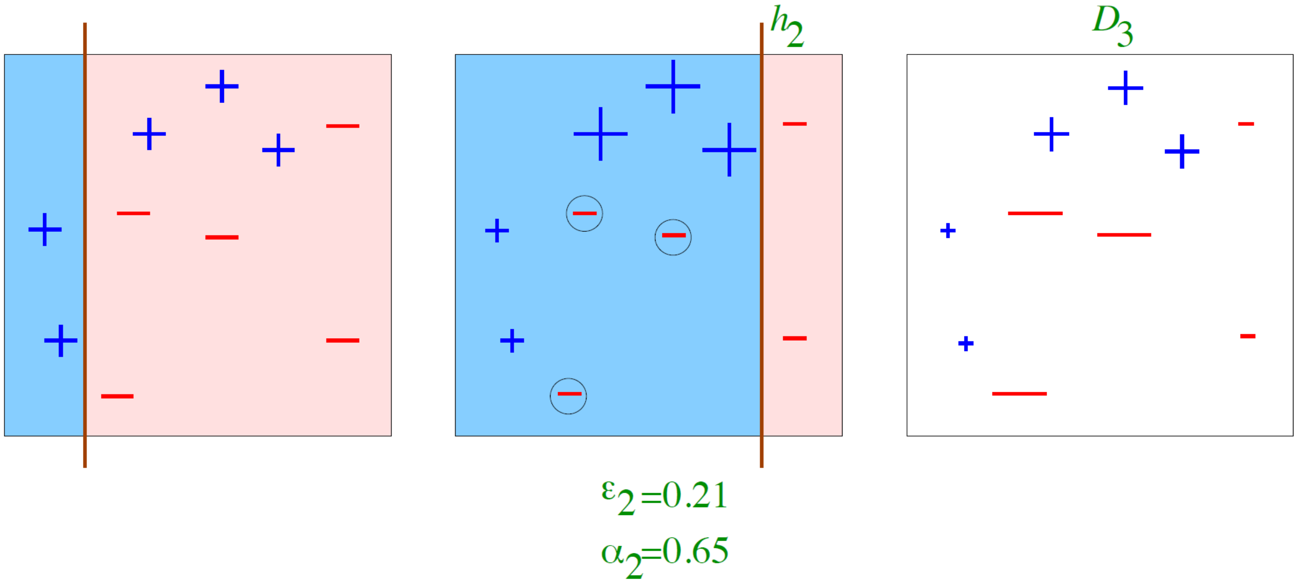
$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } y_i = h_t(x_i) \\ e^{\alpha_t} & \text{if } y_i \neq h_t(x_i) \end{cases}$$

$$= \frac{D_t(i)}{Z_t} \exp(-\alpha_t \, y_i \, h_t(x_i))$$

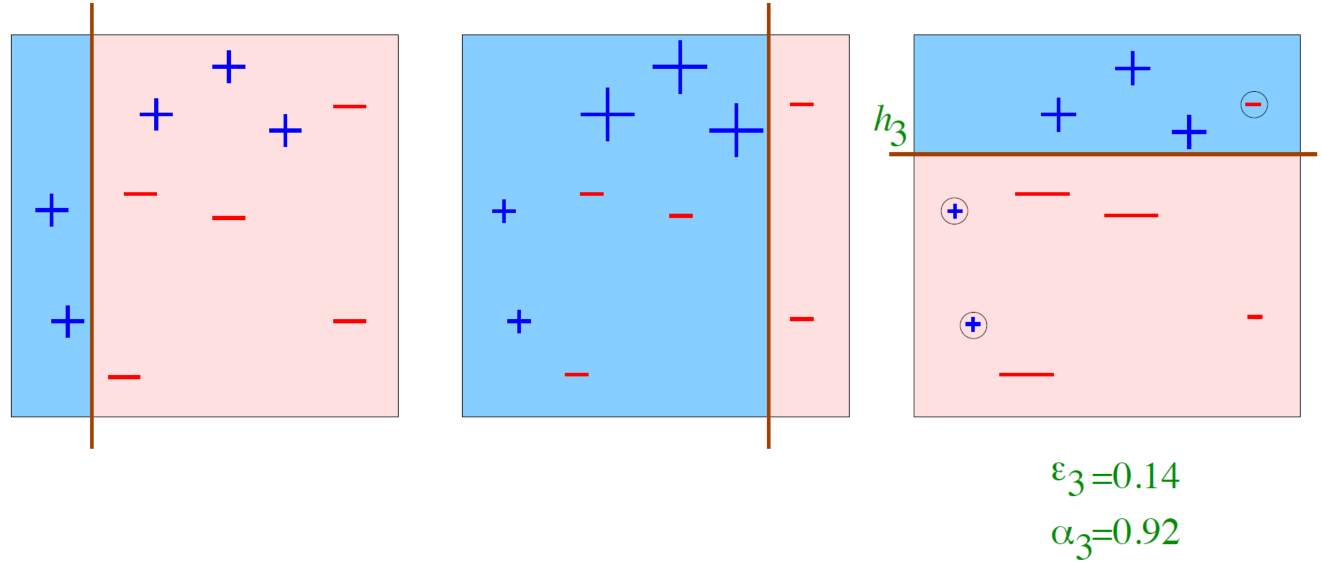
where $Z_t = \text{normalization factor}$

$$\alpha_t = \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right) > 0$$

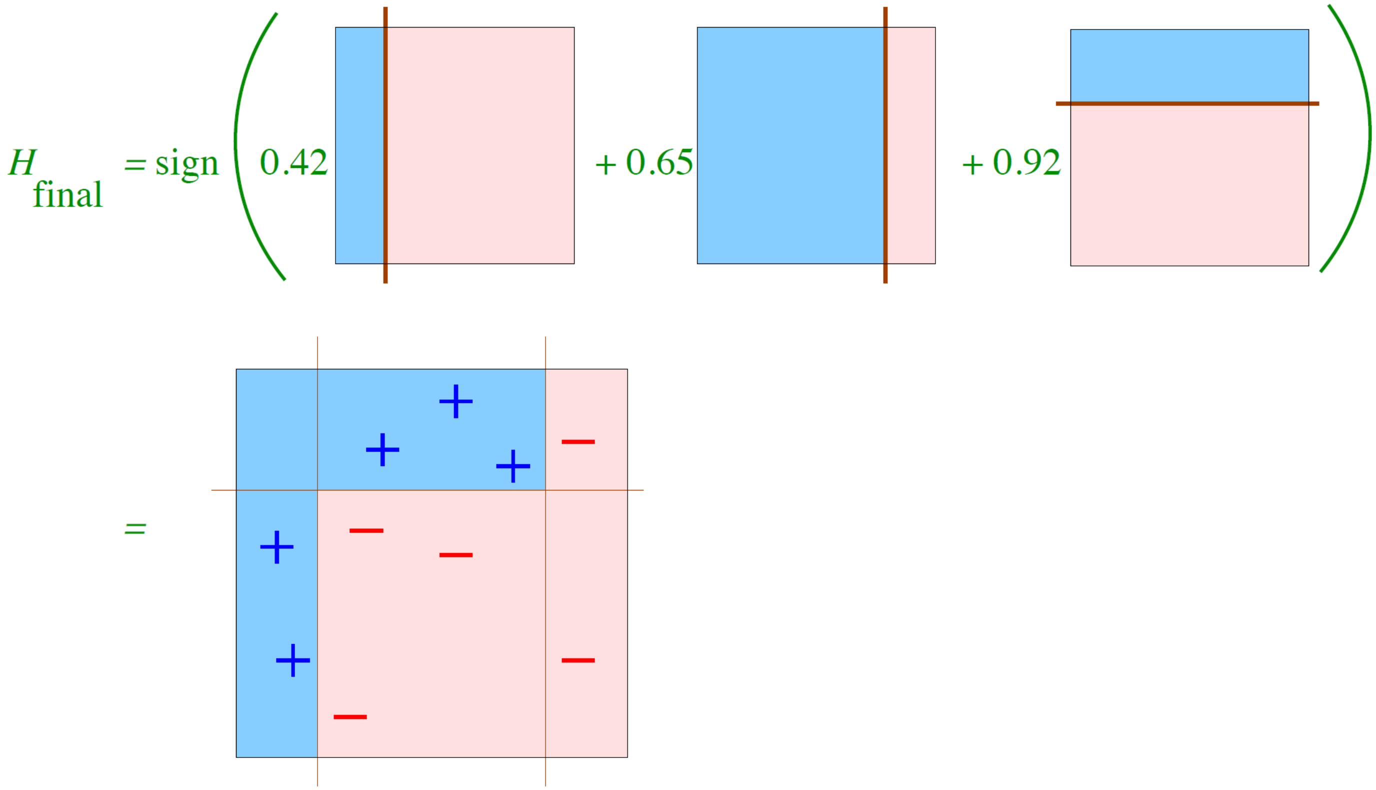
Round 2



Round 3



Final classifier



- final classifier:
  - $H_{\text{final}}(x) = \text{sign}\left(\sum_t \alpha_t h_t(x)\right)$

Voted combination of classifiers

- The general problem here is to try to combine many simple "weak" classifiers into a single "strong" classifier
- We consider voted combinations of simple binary ± 1 component classifiers

$$h_m(\mathbf{x}) = \alpha_1\, h(\mathbf{x};\theta_1) + \ldots + \alpha_m\, h(\mathbf{x};\theta_m)$$

where the (non-negative) votes α_i can be used to emphasize component classifiers that are more reliable than others

where $Z_t = \text{normalization factor}$

$$\alpha_t = \frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right) > 0$$

Smaller error, larger weight

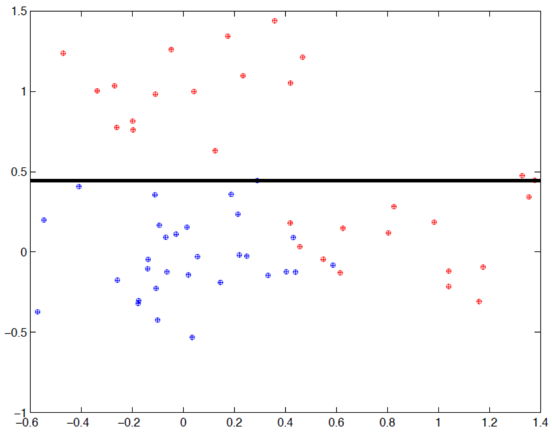
Example of weak learner

- Consider the following simple family of component classifiers generating ± 1 labels:

$$h(\mathbf{x};\theta) = \mathrm{sign}(\, w_1\, x_k - w_0\,)$$

  where $\theta = \{k, w_1, w_0\}$. These are called *decision stumps*.

- Each decision stump pays attention to only a single component of the input vector



In practice, it is OK to assume w_1 = 1 or -1

Discussion

- How to train a decision stump for a set of given weights?

Discussion

- How to train a decision stump for a set of given weights?
  - Try different model parameters and pick the one given the minimal weighted error

$$Pr_{i \sim D_t}[h_t(x_i) \neq y_i] = \frac{1}{N} \sum_i D_t(i)[h_t(x_i) \neq y_i]$$

  - Efficient method could be used to fast search the optimal parameter
  - Example: consider choosing the k-th dimension and $w_1 = 1$, how to choose w_0

    Feature values at the k-th dimension [0.5, 0.2, 0.3,-0.4, 0.1]

    Weights for each sample [0.2,0.1, 0.3, 0.2, 0.2]

Discussion

- How to train a decision stump for a set of given weights?
  - Try different model parameters and pick the one given the minimal weighted error

$$Pr_{i \sim D_t}[h_t(x_i) \neq y_i] = \frac{1}{N} \sum_i D_t(i)[h_t(x_i) \neq y_i]$$

  - Efficient method could be used to fast search the optimal parameter
  - Example: consider choosing the k-th dimension and $w_1 = 1$, how to choose w_0

    Feature values at the k-th dimension [0.5, 0.2, 0.3,-0.4, 0.1]

    Weights for each sample [0.2,0.1, 0.3, 0.2, 0.2]

Discussion

- Example: consider choosing the k-th dimension and $w_1 = 1$, how to choose w_0

  Feature values at the k-th dimension [0.5, 0.2, 0.3,-0.4, 0.1]

  Weights for each sample [0.2,0.1, 0.3, 0.2, 0.2]

- Rank the feature values and try threshold $\frac{(v_t + v_{t+1})}{2}$
- Calculate weighted classification error for each threshold and find the best threshold

  [0.5, 0.3, 0.2, 0.1, -0.4]

  [0.2, 0.3, 0.1, 0.2, 0.2]

Questions

- Although intuitively reasonable, why should we choose the update equation in Adaboost?
- What is the objective function of Adaboost?

Loss of Adaboost

- We need to define a loss function for the combination so we can determine which new component $h(\mathbf{x};\theta)$ to add and how many votes it should receive

$$h_m(\mathbf{x}) = \alpha_1 h(\mathbf{x};\theta_1) + \ldots + \alpha_m h(\mathbf{x};\theta_m)$$

- While there are many options for the loss function we consider here only a simple exponential loss

$$\exp\{\, -y\, h_m(\mathbf{x})\, \}$$

$$\text{Note that } y \in \{1,-1\}, h_m(\mathbf{x}) \in \{1,-1\}$$

For correct classification, $loss = \exp(-1)$;

incorrect classification $loss = \exp(1)$

$\sum_i exp(-y_i h_m(\mathbf{x}_i))$ is smaller if classification error is smaller

Adding a weak learner

- Consider adding the m^{th} component:

$$\sum_{i=1}^{n} \exp\{ -y_i[h_{m-1}(\mathbf{x}_i) + \alpha_m h(\mathbf{x}_i;\theta_m)] \}$$

$$= \sum_{i=1}^{n} \exp\{ -y_i h_{m-1}(\mathbf{x}_i) - y_i \alpha_m h(\mathbf{x}_i;\theta_m) \}$$

Note that Adaboost sequentially add weak learner, Adaboost can be seen as a greedy way of optimizing the loss function: fixed previously found classifiers and weights and only optimize the current round of classifier and weight

Adding a weak learner

- Consider adding the m^{th} component:

$$\sum_{i=1}^{n} \exp\{ -y_i[h_{m-1}(\mathbf{x}_i) + \alpha_m h(\mathbf{x}_i;\theta_m)] \}$$

$$= \sum_{i=1}^{n} \exp\{ -y_i h_{m-1}(\mathbf{x}_i) - y_i \boxed{\alpha_m} h(\mathbf{x}_i; \boxed{\theta_m}) \}$$

Variables to be optimized at the m round

Note that Adaboost sequentially add weak learner, Adaboost can be seen as a greedy way of optimizing the loss function: fixed previously found classifiers and weights and only optimize the current round of classifier and weight

Adding a weak learner

- Consider adding the m^{th} component:

$$
\begin{aligned}
&\sum_{i=1}^{n} \exp\{\, -y_i[h_{m-1}(\mathbf{x}_i) + \alpha_m h(\mathbf{x}_i;\theta_m)]\,\} \\
&= \sum_{i=1}^{n} \exp\{\, -y_i h_{m-1}(\mathbf{x}_i) - y_i \alpha_m h(\mathbf{x}_i;\theta_m)\,\} \\
&= \sum_{i=1}^{n} \underbrace{\exp\{\, -y_i h_{m-1}(\mathbf{x}_i)\}}_{\text{fixed at stage } m} \exp\{-y_i \alpha_m h(\mathbf{x}_i;\theta_m)\,\}
\end{aligned}
$$

Adding a weak learner

- Consider adding the m^{th} component:

$$\begin{aligned}
&\sum_{i=1}^{n} \exp\{\, -y_i[h_{m-1}(\mathbf{x}_i) + \alpha_m h(\mathbf{x}_i;\theta_m)]\,\} \\
&= \sum_{i=1}^{n} \exp\{\, -y_i h_{m-1}(\mathbf{x}_i) - y_i \alpha_m h(\mathbf{x}_i;\theta_m)\,\} \\
&= \sum_{i=1}^{n} \underbrace{\exp\{\, -y_i h_{m-1}(\mathbf{x}_i)\}}_{\text{fixed at stage } m} \exp\{-y_i \alpha_m h(\mathbf{x}_i;\theta_m)\,\} \\
&= \sum_{i=1}^{n} W_i^{(m-1)} \exp\{-y_i \alpha_m h(\mathbf{x}_i;\theta_m)\,\}
\end{aligned}$$

Solving α

$$\frac{\partial}{\partial \alpha_m} \sum_{i=1}^{n} W_i^{(m-1)} \exp\{-y_i \alpha_m h(\mathbf{x}_i;\theta_m)\} =$$

$$\left[\sum_{i=1}^{n} W_i^{(m-1)} \exp\{-y_i \alpha_m h(\mathbf{x}_i;\theta_m)\} \cdot \left(-y_i h(\mathbf{x}_i;\theta_m)\right)\right] = 0$$

Define $I_i = y_i h(\mathbf{x}_i;\theta_m) \in \{1,-1\}$

$$\exp(-y_i \alpha_m h(\mathbf{x}_i;\theta_m)) = \frac{(I_i+1)}{2}\exp(-\alpha_m) - \frac{(I_i-1)}{2}\exp(\alpha_m)$$

$$\begin{aligned} Above &= -\sum_i W_i^{m-1}\left(\frac{(I_i+1)}{2}\exp(-\alpha_m) - \frac{(I_i-1)}{2}\exp(\alpha_m)\right) I_i \\ &= -\sum_i W_i^{m-1}\left(\frac{(I_i^2+I_i)}{2}\exp(-\alpha_m) - \frac{(I_i^2-I_i)}{2}\exp(\alpha_m)\right) \\ &= -\sum_i W_i^{m-1}\left(\frac{(1+I_i)}{2}\exp(-\alpha_m) - \frac{(1-I_i)}{2}\exp(\alpha_m)\right) = 0 \end{aligned}$$

Solving α

$$\begin{aligned} Above &= -\sum_i W_i^{m-1}\left(\frac{(I_i+1)}{2}\exp(-\alpha_m) - \frac{(I_i-1)}{2}\exp(\alpha_m)\right) I_i \\ &= -\sum_i W_i^{m-1}\left(\frac{(I_i^2+I_i)}{2}\exp(-\alpha_m) - \frac{(I_i^2-I_i)}{2}\exp(\alpha_m)\right) \\ &= -\sum_i W_i^{m-1}\left(\frac{(1+I_i)}{2}\exp(-\alpha_m) - \frac{(1-I_i)}{2}\exp(\alpha_m)\right) = 0 \end{aligned}$$

$$\exp(-\alpha_m)\left(\textstyle\sum_{i|I_i=1} W_i^{m-1}\right) = \exp(\alpha_m)\left(\textstyle\sum_{i|I_i=-1} W_i^{m-1}\right)$$

$$\alpha_m = \tfrac{1}{2}\ln\left(\frac{\sum_{i|I_i=1} W_i^{m-1}}{\sum_{i|I_i=-1} W_i^{m-1}}\right)$$

$$\text{If } \textstyle\sum_i W_i^{m-1} = 1, \text{ then } \alpha_m = \tfrac{1}{2}\ln\left(\tfrac{1-\epsilon_m}{\epsilon_m}\right)$$

Solving θ_m

$$\sum_{i=1}^{n} W_i^{m-1} \exp\{-y_i \alpha_m h(\mathbf{x}_i;\theta_m)\}$$

$$= \frac{1}{2}\sum_i W_i^{m-1} I_i(\exp(-\alpha_m) - \exp(\alpha_m)) + \frac{1}{2}\sum_i W_i^{m-1}(\exp(-\alpha_m) + \exp(\alpha_m))$$

$$= (\exp(-\alpha_m) - \exp(\alpha_m))\frac{1}{2}\sum_i W_i^{m-1} I_i + (\exp(-\alpha_m) + \exp(\alpha_m))\frac{1}{2}\sum_i W_i^{m-1}$$

$$= (\exp(-\alpha_m) - \exp(\alpha_m))\frac{1}{2}\sum_i W_i^{m-1} I_i + const$$

Equivalent to minimize the weighted error

Equivalent to minimize $-\sum_i W_i^{m-1} I_i$

$$\exp(-y_i \alpha_m h(\mathbf{x}_i;\theta_m)) = \tfrac{(I_i+1)}{2}\exp(-\alpha_m) - \tfrac{(I_i-1)}{2}\exp(\alpha_m)$$

Update W

From the definition of W_i^{m-1}

$$= \sum_{i=1}^{n} \underbrace{\exp\{-y_i h_{m-1}(\mathbf{x}_i)\}}_{\text{fixed at stage } m} \exp\{-y_i \alpha_m h(\mathbf{x}_i;\theta_m)\}$$

$$= \sum_{i=1}^{n} W_i^{(m-1)} \exp\{-y_i \alpha_m h(\mathbf{x}_i;\theta_m)\}$$

and the update equation of h_m

$$h_m(\mathbf{x}_i) = h_{m-1} + \alpha_m h(\mathbf{x}_i;\theta_m)$$

We get

$$W_i^m = W_i^{m-1} \times \exp(-y_i \alpha_m h(\mathbf{x}_i;\theta_m))$$

Note that since we update W greedily, it is OK to divide W by a constant or normalize W to make it sum to 1

Put the above derivation together

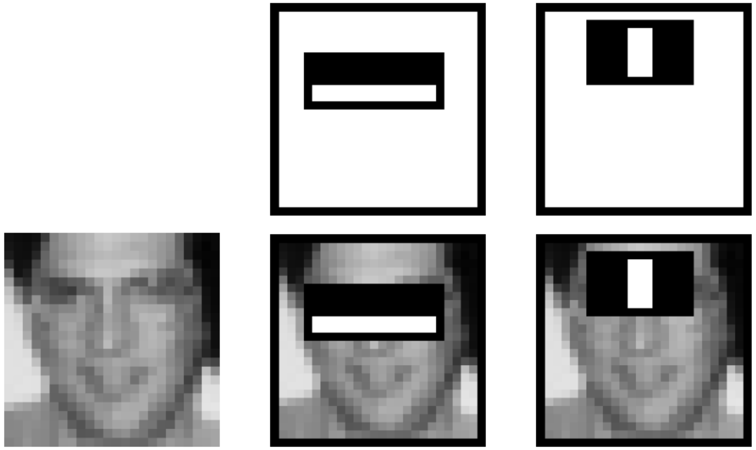
- We show that Adaboost can be seen as a greedy solution to the following optimization problem (the objective function of Adaboost)

$$\min_{\{\alpha_k\},\{\theta_k\}} \sum_i \exp(-y_i h_m(\mathbf{x}_i))$$

- It sequentially optimizes each α_k and θ_m

Application of Adaboost

- problem: find faces in photograph or movie
- weak classifiers: detect light/dark rectangles in image

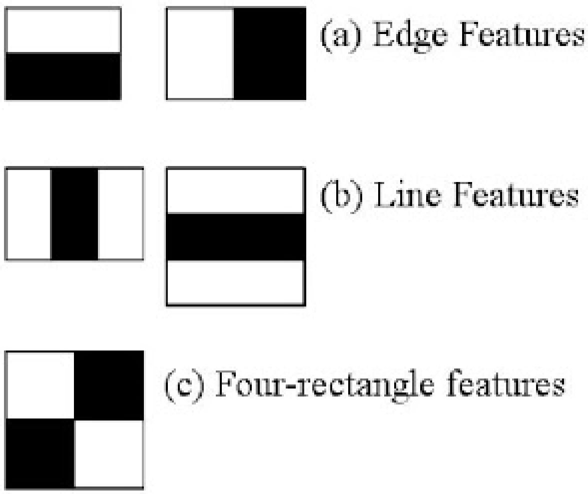


- many clever tricks to make extremely fast and accurate

P. Viola; M. Jones. Rapid object detection using a boosted cascade of simple features. CVPR 2001

Application of Adaboost: Face detection

- Face detection: scan the image with a bounding box and evaluate the decision at each candidate location
  - Very time consuming, need fast evaluation
- Haar features
  - The sum of pixel values in the black area – The sum of pixel values in the white area
  - Parameters:
    - Type of Haar features
    - Location of Haar template inside the bounding box



Combine results by Adaboost
Used in OpenCV

Summary

- Ensemble Learning
  - Main idea.
  - Why it works
- Random forest
  - Decision tree
  - Injecting randomness to decision tree to make a random forest
- Bagging
- Boosting
  - Adaboost: procedure
  - Weak classifier
  - The objective function of Adaboost