Lecture 7-9. AdaBoost, RealBoost, and Logistic Regression

Boosting: One of a few popular machine learning algorithms in last 15 years (the others are Support Vector Machines, and DeepLearning).

It has several versions:

Adaboost, RealBoost, and LogitBoost optimizing different design of loss functions.

Intuitively, it is a technique for combining a number of “weak” classifiers to make a “strong” classifier.

We will describe it for binary classification, but it can be extended to N-class classification.

Outline

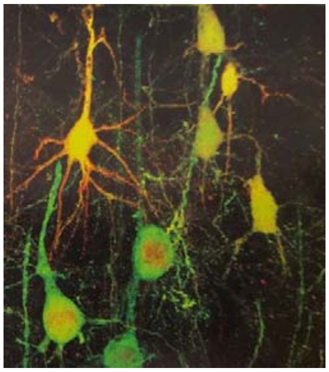
This part will be covered in three lectures, using face detection
as an example.

Lect 1. Basic algorithm: introducing the boosting procedure

Lect 2. Convergence analysis: showing each step is minimizing a function
which is an upper bound of the classification error:
Adaboost, RealBoost, Logitboost as 3 examples

Lect 3. A statistical framework: discussing its relation to the
Logitboost and Maximum likelihood estimation (MLE learning).

Background



This 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.

Some reasons against the idea:

- XOR challenge;
- Rationale for separating three levels of thinking: representation, algorithm and implementation

Background

This idea didn’t work, and inspired three lines of remarkable research.

1, Multi-layered neural networks, which were trained by Back-propagation (Hinton et al 1986) and found not working well. But recently it has been expanded to many more layers in Deep Learning with Convolutional Neural Networks, and made to work with huge training data. It becomes popular, again, since 2012.

2, Boosting, which adds a hidden layer of neurons one by one until the classification error on the training images goes to zero, by sequentially minimizing the upper-bound of the empirical error.

3, Support Vector Machines, which stay with one layer perceptron, but mapping the features to higher-dimensions.

All these methods minimize the empirical loss or its upper bounds, without explicitly modeling the data.

Boosting

We are given

(1) a set of labeled training data from 2 classes

$$\Omega = \{(x_i, y_i) : x_i \in \Omega^d,\ y_i \in \{-1, 1\},\ i = 1, \dots, m.\}$$

(2) a set of *weak classifiers*, a pool constructed for a task

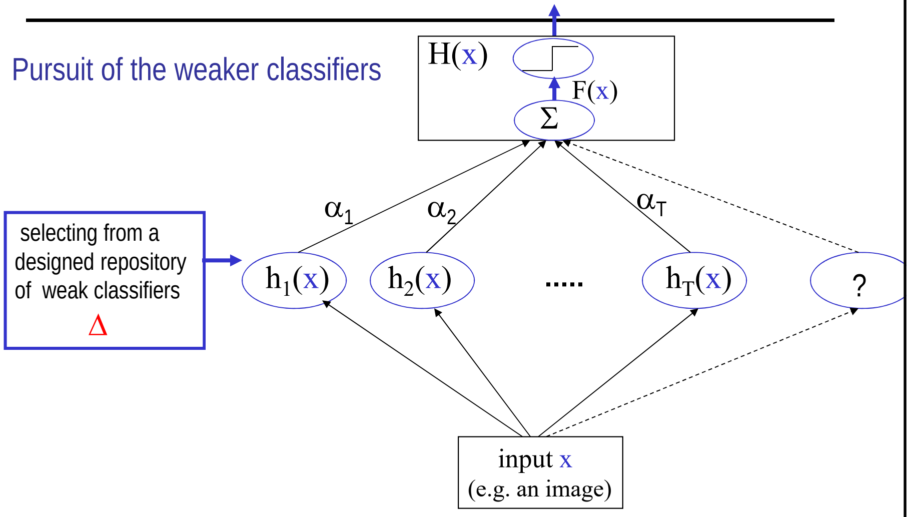
$$\Delta = \{h_t(x) : t = 1, 2, \dots T, ...\}$$

$$h_t(x) : \Omega^d \mapsto \{-1, 1\}$$

Intuition: suppose we have two weak classifiers h_1 and h_2 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.

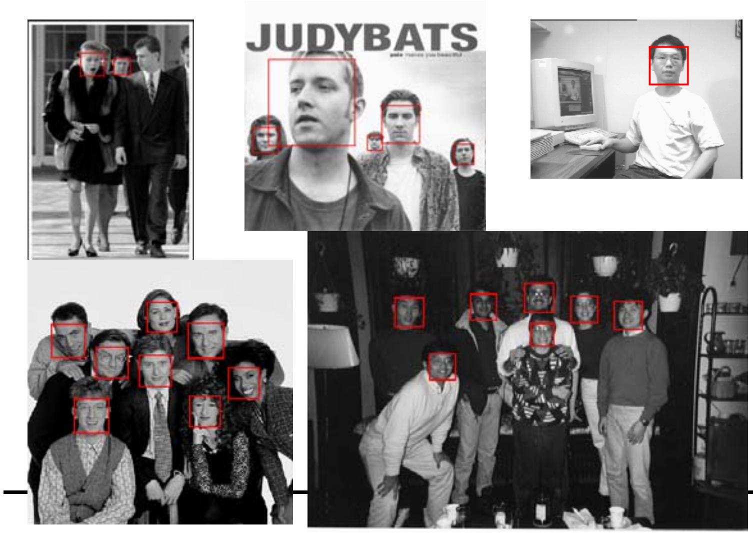
Illustration of the boost classifier:

Pursuit of the weaker classifiers



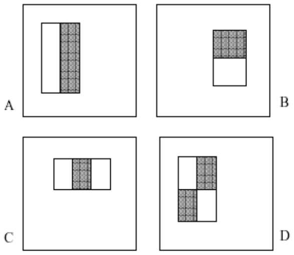
Example on face detection

Boosting is made popular by this face detection example as it works very well.



Example of weak classifier

For example, the 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.



Then it has a threshold θ and a parity (+/-) p to flip the sign for <50% error.

$$h(x) = \mathrm{sign}(<\omega, x>)$$

This is also called a "tree stump", as it is a 1 step decision tree.

F is a filter (see A,B,C,D).
Note that our binary classifiers have value in {-1, +1} rather than {0,1}.
But they are essentially the same except different thresholds.

Boosting

Relations to a layered neural network:

We denote these classifiers by $h_t(x; \omega_t)$, where ω_t is the parameter --- a vector of weights. By pre-designing ω_t which are also called “features”, we largely reduce the problem of learning large number of parameters in the multi-layer neural net.

Here we introduced the history of Neural Network and learning by back-propagation on blackboard.

The downside is: these manually design may not be effective or sufficient for complex data domains. The recent DeepLearning approaches go back to learn these features.

Boosting

The error rate of a weak classifier h is calculated empirically over the training set

$$\epsilon(h) = \frac{1}{m}\sum_{i=1}^{m} 1(h(x_i) \neq y_i) < \frac{1}{2}$$

where 1() is an indicator function.
It equals to 1 if the condition is true, and 0 otherwise.

A *weak classifier* works better than chance. For binary classifiers, it has less than 50% errors. If it is larger than 50%, we can put a minus sign to make it less than 50%. So we just don’t want classifiers with exactly 50% error. Such classifiers have no information.

Basic Boost

A strong classifier is a combination of a number of weaker classifiers *:*

$$H(x) = \text{sign}\{\alpha_1 h_1(x) + \cdots + \alpha_T h_T(x)\}$$

We denote by

$$h = (h_1, ..., h_T)$$

$$\alpha = (\alpha_1, ..., \alpha_T)$$

$$F(x) = \alpha_1 h_1(x) + \cdots + \alpha_T h_T(x) =< \alpha, h(x) >$$

So our objective is to choose h and parameters α to minimize the empirical error of the strong classifier $\text{Err}(H) = \frac{1}{m}\sum_{i=1}^{m} 1(H(x_i) \neq y_i)$

$$(\alpha, h)^* = \arg\min \text{Err}(H)$$

Basic AdaBoost Algorithm

Characteristics of Adaboost

1. AdaBoost is a *sequential algorithm* that minimizes an upper bound of the classification error Err(H) by selecting the *weak classifiers* h and their weights a one by one (pursuit). Each time, a weak classifier is selected to maximally reducing the upper bound of error.

2. AdaBoost assigns *weights to the data samples. The weights are summed to one.* These weights are updated when a new weak classifier h+ is added. *Data samples that are misclassified by* h+ *are given more weight* so that they receive more "attention" in selecting the next weak classifier.

3. Each step, a new weak classifier is selected to minimize the *weighted error,* so pays greater attention to misclassified samples.

4. The empirical error will converge to zero at an exponential rate (subject to assumptions described later).

Basic AdaBoost algorithm

0. Initialize the data with uniform weight

$$D_o = \frac{1}{m}, \quad \forall X_i \in \Omega, \qquad \text{so} \sum_{i=1}^{m} D(X_i) = 1$$

set t =0

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

$$\epsilon_t(h) = \sum_{i=1}^{m} D_t(x_i) 1(h(x_i) \neq y_i) \qquad \forall h \in \Delta$$

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

$$h_t = \arg\min_{h \in \Delta} \epsilon_t(h)$$

3. Assign weight for the new classifier $\alpha_t = \frac{1}{2} \log \frac{1 - \epsilon_t(h_t)}{\epsilon_t(h_t)}$

4. 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 1-4 until three conditions (next page).

Basic AdaBoost algorithm

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 and thus may have better performance on the test data..

- All the remaining weak classifiers have error close to 0.5 and thus redundant.

- A maximum number of weak classifier T is reach.

Facts about the weights

1. The weight for the new classifier is always positive.

The smaller classification error, and bigger weight, and stronger “voice” in the strong classifier.

$$\epsilon_t(h_t) < \frac{1}{2} \Rightarrow \alpha_t = \frac{1}{2}\log\frac{1-\epsilon_t(h_t)}{\epsilon_t(h_t)} > 0$$

$$\epsilon(h_A) < \epsilon(h_B) \Rightarrow \alpha(h_A) > \alpha(h_B)$$

2. The weights of the data points are multiplied by

$$\exp\{-y_i\alpha_t h_t(x_i)\} = \left(\begin{array}{ll} e^{-\alpha_t} < 1 & \text{if } h_t(x_i) = y_i, \\ e^{+\alpha_t} > 1 & \text{if } h_t(x_i) \neq y_i \end{array} \right.$$

The weights of the incorrectly classified data points increase, and the weights of the correctly classified data points decreases. So the incorrectly points receive more “attention” in the next run.

Facts about the normalizing functions

At each step, the weights of the data points are normalized by

$$Z_t = \sum_{x_i \in \Omega} D_{t-1}(x_i) \exp -y_i\alpha_t h_t(x_i)$$

$$= \sum_{x_i \in \mathrm{A}} D_{t-1}(x_i)\, \mathrm{e}^{-\alpha_t} + \sum_{x_i \in \mathrm{A}} D_{t-1}(x_i)\, \mathrm{e}^{+\alpha_t}.$$

A is the correctly classified data points by h_t. $Z_t = Z_t(\alpha_t)$ is a function of α_t.

The data weights can be computed recursively,

$$\begin{aligned} D_{t+1}(x_i) &= \frac{1}{Z_t} D_t(x_i) \exp\{-y_i\alpha_t h_t(x_i)\} \\ &\cdots \\ &= \frac{1}{Z_t \cdot Z_{t-1} \cdots Z_1} \frac{1}{m} \exp\{-y_i F(x_i)\} \end{aligned}$$

Facts about the normalizing functions

As the data weights have to be summed to one, we have

$$\sum_{i=1}^{m} D_t(x_i) = \frac{1}{Z_t \cdot Z_{t-1} \cdots Z_1} \frac{1}{m} \exp\{-y_i F(x_i)\} = 1,$$

Therefore

$$Z_t \cdot Z_{t-1} \cdots Z_1 = \frac{1}{m} \sum_{i=1}^{m} \exp\{-y_i F(x_i)\}.$$

We define a new normalizing function,

$$\begin{aligned} Z = Z_t \cdot Z_{t-1} \cdots Z_1 &= \frac{1}{m} \sum_{i=1}^{m} \exp\{-y_i F(x_i)\}, \\ &= \frac{1}{m} \sum_{i=1}^{m} \exp\{-y_i < \alpha, h(x_i) >\} \end{aligned}$$

Basic AdaBoost Algorithm

After t steps, the estimate of the strong classifier is:

$$H(x;\Theta) = \mathrm{sign}(\alpha_1 h_1(x) + \ \ldots + \alpha_t h_t(x))$$

Note that H(x) = H(x; Θ) with parameter Θ includes continuous weight and discrete index to the weaker classifiers. $\Theta = (\alpha_1, \ldots, \alpha_T;\ h_1, \ldots, h_T)$.

The data weights are calculated by recursion:

$$\begin{aligned} D_{t+1}(x_i) &= \frac{1}{Z_t(\alpha_t)} D_t(i) e^{-y_i \alpha_t h_t(x_i)} \\ &= \frac{1}{Z_t(\alpha_t) Z_{t-1}(\alpha_{t-1})} D_{t-1}(i) e^{-y_i\{\alpha_t h_t(x_i) + \alpha_{t-1} h_{t-1}(x_i)\}} \\ &= \ldots \\ &= \frac{1}{Z_t(\alpha_t) \cdots Z_1(\alpha_1)} D_1(i) e^{-y_i\{\alpha_t h_t(x_i) + \cdots + \alpha_1 h_1(x_i)\}} \end{aligned}$$

AdaBoost Convergence Analysis

1. Why, and when, does AdaBoost converge?

2. How fast does AdaBoost converge?

3. Why do we calculate the weight of the weak classifier as the following?

$$\alpha_t = \frac{1}{2} \log \frac{1 - \epsilon_t(h_t)}{\epsilon_t(h_t)}$$

4. Why do we choose the weak classifier that has minimum weighted error?

$$h_t = \arg \min_{h \in \Delta} \epsilon_t(h)$$

Boosting and margin

Empirical error for the strong classifier

$$\mathrm{Err}(H) = \frac{1}{m} \sum_{i=1}^{m} 1(H(x_i) \neq y_i)$$

We know, F(x)=0 is the decision boundary, F(x) measures how far away the point x is from the decision boundary (can be + or – depending on which side of F(x)=0.)

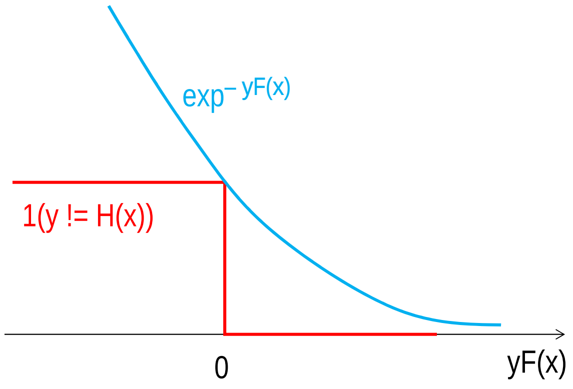
We define the “margin” of x to be y F(X)

If $y_iF(x_i) > 0$ then $H(x_i)=y_i$ is the correct classification.

Instead of minimizing the error Err(H) directly, Boost minimizes upper bounds of Err(H).

Boosting, margin and loss function

When we add new weaker classifiers, H(x; θ) will have less error and the upper bound goes to zero. *When this upper bound is minimized to zero, so is Err(H).*



AdaBoost Convergence

Now, we show the claim on the error bound is true, i.e.

$$\boxed{\mathrm{Err}(H) = \frac{1}{m}\sum_{i=1}^{m} 1(H(x_i) \neq y_i) \leq Z = \frac{1}{m}\sum_{i=1}^{m} \exp\{-y_i F(x_i)\}}$$

[Proof] As $F(x) = \mathrm{sign}(F(x))||F(x)|| = H(x_i)||F(x_i)||$

If $H(x_i) \neq y_i$ then $\mathrm{LHS} = 1 \leq \mathrm{RHS} = e^{+||F(x)||}$

If $H(x_i) = y_i$ $\mathrm{LHS} = 0 \leq \mathrm{RHS} = e^{-||F(x)||}$

So the inequality holds for each term,

$$1(H(x_i) \neq y_i) \leq \exp\{-y_i F(x_i)\}$$

Then the inequality is true.

AdaBoost Convergence

Claim: After t-steps, the error of the strong classifier is bounded above by a quantity Z. Z is the product of the weight normalization factors

$$\mathrm{Err}(H) \leq Z$$
$$Z = Z(\alpha, h) = Z_t(\alpha_t, h_t) \cdot Z_{t-1}(\alpha_{t-1}, h_{t-1}) \cdots Z_1(\alpha_1, h_1)$$

Adaboost is a greedy algorithm that minimizes the upper bound of the classification error,

$$(h, \alpha)^* = \arg\min Z(\alpha, h)$$

In each step t, it chooses the optimal h_t and α_t to minimize Z_t

$$(h_t, \alpha_t)^* = \arg\min Z_t(\alpha_t, h_t)$$

As Z goes to zero, the classification error (over the training set) goes to zero. So it converges.

AdaBoost Convergence

Now we solve the minimization problem (pursuit of weak classifier)

$$(h_t, \alpha_t)^* = \arg\min Z(\alpha_t, h_t)$$

For any weak classifier h_t, recall the formula on page 15,

$$Z_t = \sum_{x_i \in \Omega} D_{t-1}(x_i) \exp -y_i \alpha_t h_t(x_i)$$
$$= \sum_{x_i \in A} D_{t-1}(x_i)\, e^{-\alpha_t} + \sum_{x_i \in A} D_{t-1}(x_i)\, e^{+\alpha_t}.$$

Where A is the set of correctly classified point by h_t

We take a derivative and set it to zero,

$$\frac{dZ_t(\alpha_t, h_t)}{d\alpha_t} = \sum_{x_i \in A} -D_t(x_i) e^{-\alpha_t} + \sum_{x_i \in \bar{A}} D_t(x_i) e^{\alpha_t} = 0$$
$$\sum_{x_i \in A} D_t(x_i) = \sum_{x_i \in \bar{A}} D_t(x_i) e^{2\alpha_t}$$

AdaBoost Convergence

$$\epsilon_t(h) = \sum_{i=1}^{m} D_t(x_i) 1(h(x_i) \neq y_i) = \sum_{x_i \in \bar{A}} D_t(x_i) \quad \forall h \in \Delta$$

Therefore we have the result used in the algorithm

$$\alpha_t = \frac{1}{2} \log \frac{1 - \epsilon_t(h_t)}{\epsilon_t(h_t)}$$

Then we plug in $Z_t(\alpha_t, h_t)$ to get the minimum

$$\begin{aligned} Z_t(\alpha_t, h_t) &= \sum_{x_i \in A} D_t(x_i) e^{-\alpha_t} + \sum_{x_i \in \bar{A}} D_t(x_i) e^{\alpha_t} \\ &= (1 - \epsilon_t(h_t)) \sqrt{\frac{\epsilon_t(h_t)}{1 - \epsilon_t(h_t)}} + \epsilon_t(h_t) \sqrt{\frac{1 - \epsilon_t(h_t)}{\epsilon_t(h_t)}} \\ &= 2\sqrt{\epsilon_t(h_t)(1 - \epsilon_t(h_t))} \end{aligned}$$

AdaBoost Convergence

Change a variable $\gamma_t = \frac{1}{2} - \epsilon_t(h_t), \quad \gamma_t \in (0, \frac{1}{2}]$

Then we have the minimum to be

$$\begin{aligned} Z_t(\alpha_t, h_t) &= 2\sqrt{\epsilon_t(h_t)(1 - \epsilon_t(h_t))} \\ &= \sqrt{1 - 4\gamma_t^2} \\ &\leq \exp\{-2\gamma_t^2\} \end{aligned}$$

Therefore, after t steps, the error rate of the strong classifier is bounded by

$$\begin{aligned} \mathrm{Err}(H) &\leq Z = Z_t(\alpha_t, h_t) \cdot Z_{t-1}(\alpha_{t-1}, h_{t-1}) \cdots Z_1(\alpha_1, h_1) \\ &\leq \exp\{-2 \sum_{t=1}^{T} \gamma_t^2\} \end{aligned}$$

It is clear that each step the upper bound of the error decrease exponentially. A weak classifier with small error rate will lead to faster descent.

AdaBoost Convergence

In summary, the objective of Adaboost is to minimize an upper bound of the classification error

$$\begin{aligned}(\alpha, \mathsf{h})^* &= \arg\min Z(\alpha, h) \\ &= \arg\min Z_t(\alpha_t, h_t) \cdot Z_{t-1}(\alpha_{t-1}, h_{t-1}) \cdots Z_1(\alpha_1, h_1) \\ &= \arg\min \sum_{i=1}^{m} \exp\{-y_i < \alpha, h(x_i) >\}.\end{aligned}$$

It takes a *stepwise* minimization scheme and it may not be optimal due to the greedy pursuit. When we calculate the parameter for the t-th weak classifier, we do not change the weights of the previous weak classifiers.

We should stop AdaBoost if all the weak classifiers have error rate 0.5. This will eventually happen as we update the weights.

AdaBoost algorithm with {0,1} output

Suppose the weak classifiers output in {0, 1} rather than in {-1, +1}

$$\Delta' = \{g_t(x) : \Omega^d \mapsto \{0, 1\}\}.$$

Then we get weak classifiers as

$$\Delta = \{h_t(x) : h_t(x) = 2g_t(x) - 1\}.$$

The updating functions will be

$$\epsilon_t(h) = \sum_{i=1}^{m} D_t(x_i) 1(g_t(x_i) \neq y_i) \quad \forall h \in \Delta$$

$$\alpha_t = \frac{1}{2} \log \frac{1 - \epsilon_t(h_t)}{\epsilon_t(h_t)}$$

$$D_{t+1}(x_i) = \frac{1}{Z_t} D_t(x_i) \exp\{-\alpha_t S(g_t(x_i) = y_i)\} \qquad S(c) = \left(\begin{array}{ll} 1 & c = \text{true} \\ 0 & c = \text{false} \end{array} \right.$$

$$H(x) = \text{sign}\{\textstyle\sum_{t=1}^{T} \alpha_t g_t(x) - \frac{1}{2} \sum_{t=1}^{T} \alpha_t\}$$

Multi-class Adaboost.M1

Suppose we have a set of weak classifiers for K-classification

$$\Delta = \{h_t(x) : \Omega^d \mapsto \{1, 2, ..., K\}\}.$$

The algorithm is the same as the binary Adaboost in calculating the weighted error of the weak classifiers and the data weights, except the last step becomes.

$$H(x) = \arg \max_{y \in \{1,...,K\}} \sum_{t=1}^{T} \alpha_t 1(h_t(x) = y)$$

Intuitively, the class that receives the highest weighted votes from the weak classifiers will win. The classification error is also upper bounded in a similar way as the binary case.

AdaBoost.M1 algorithm

0. Initialize the data with uniform weight

$$D_1(x_i) = \frac{1}{m}, \forall x_i \in \Omega \quad \text{so} \quad \sum_{i=1}^{m} D_0(x_i) = 1.$$

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

$$\epsilon_t(h) = \sum_{i=1}^{m} D_t(x_i) 1(h(x_i) \neq y_i) \quad \forall h \in \Delta$$

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

$$h_t = \arg \min_{h \in \Delta} \epsilon_t(h)$$

3. Assign weight for the new classifier $\alpha_t = \frac{1}{2} \log \frac{1 - \epsilon_t(h_t)}{\epsilon_t(h_t)}$

4. Update the weights of the data points

$$D_{t+1}(x_i) = \frac{1}{Z_t} D_t(x_i) \exp\{-\alpha_t S(h_t(x_i) = y_i)\}$$

Set t+1 → t, repeat 1-4 until maximum step t=T.

Variations of Boost: Logitboost and RealBoost

So far, we didn't derive the algorithm using probabilistic models as we did in Bayesian Decision Theory. The algorithms are derived by design:

1, Selecting a formula for the classifier

$$y = H(x;\Theta), \quad \text{or the form of } F(x;\Theta)$$

Θ includes both the parameters and structures of classifiers.

2, Optimizing some loss function

$$\Theta^* = \operatorname{argmax} L(\Theta)$$

Once we understand the Adaboost, we can see that there are many variations by different designs of F() and L().

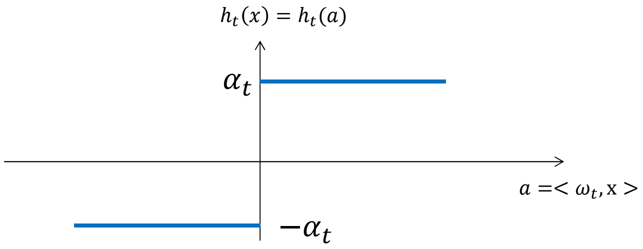
RealBoost

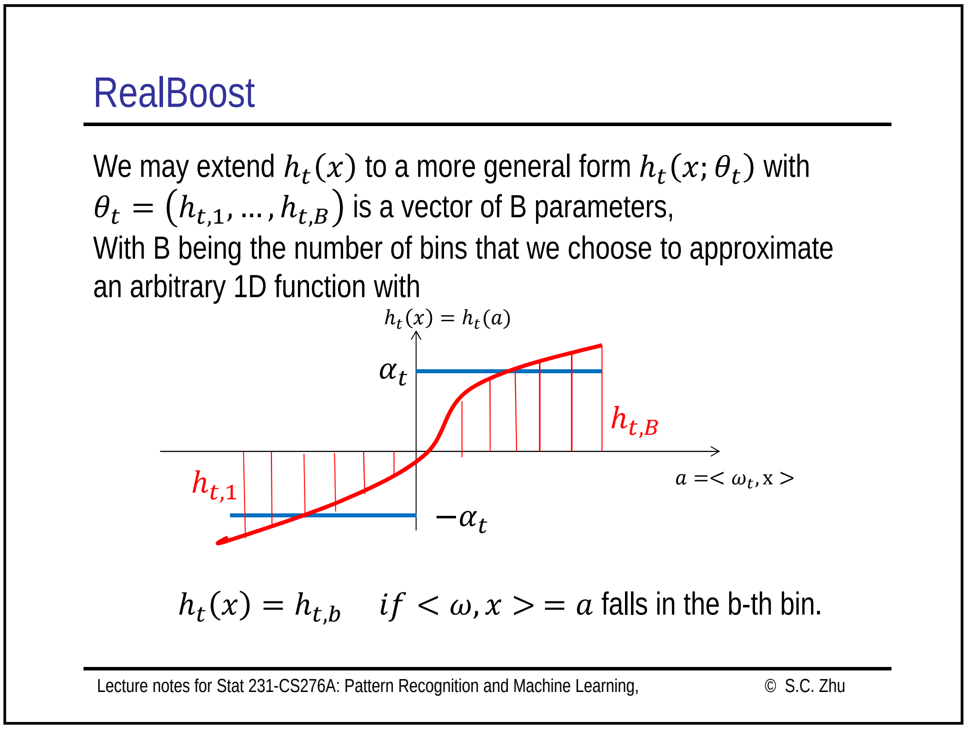
Realboost shares the same exponential loss function as Adaboost, But has a more general form of classifier.

In Adaboost: $F(x;\Theta) = \alpha_1 h_1(x) + \ \ldots + \alpha_t h_t(x)$

$h_t(x) = h_t(<\omega_t, x>)$ is defined as a 1D binary function

each weak classifier is limited to a step function of one parameter:

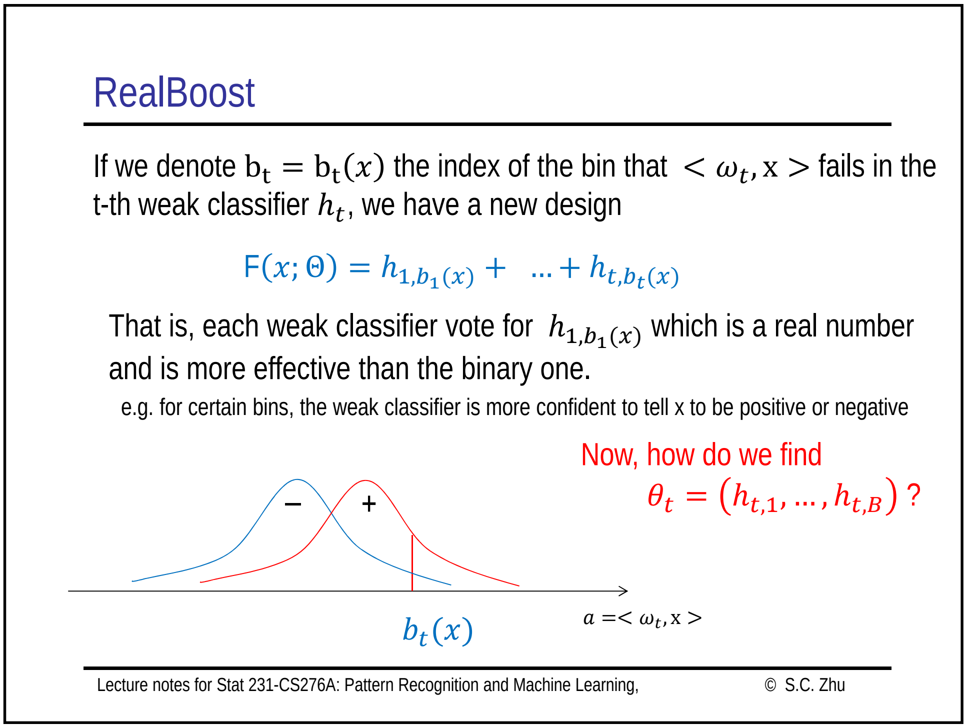




RealBoost

We may extend $h_t(x)$ to a more general form $h_t(x;\theta_t)$ with $\theta_t = \left(h_{t,1},\dots,h_{t,B}\right)$ is a vector of B parameters,
With B being the number of bins that we choose to approximate an arbitrary 1D function with

$h_t(x) = h_{t,b} \quad if < \omega, x > = a$ falls in the b-th bin.



RealBoost

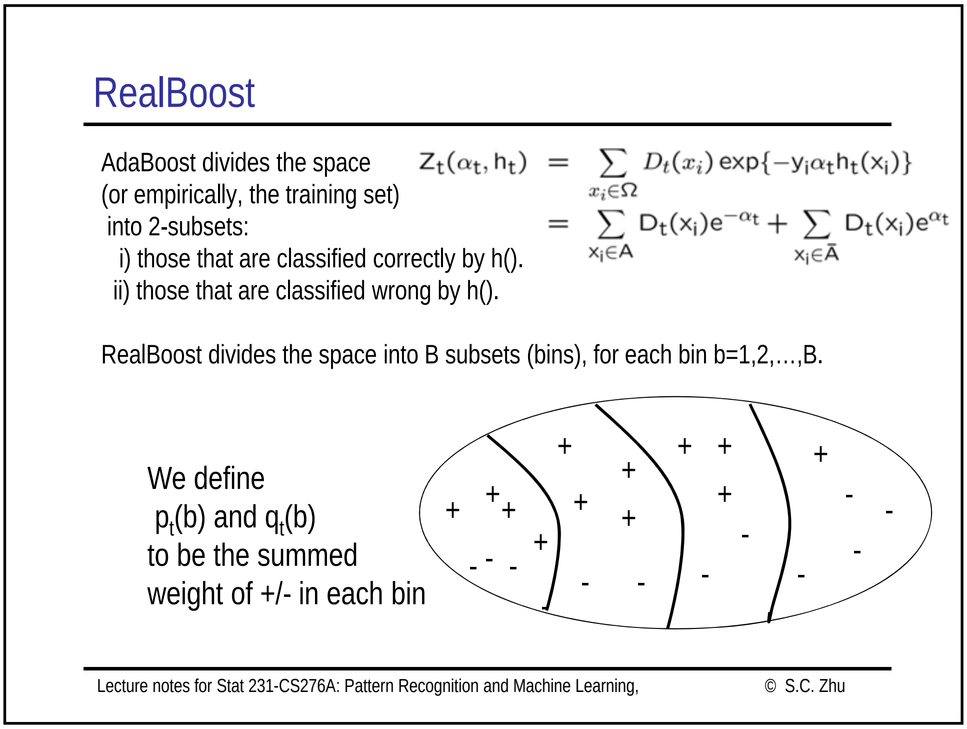
If we denote $\mathrm{b}_t = \mathrm{b}_t(x)$ the index of the bin that $< \omega_t, \mathrm{x} >$ fails in the t-th weak classifier h_t, we have a new design

$$\mathrm{F}(x;\Theta) = h_{1,b_1(x)} + \dots + h_{t,b_t(x)}$$

That is, each weak classifier vote for $h_{1,b_1(x)}$ which is a real number and is more effective than the binary one.

e.g. for certain bins, the weak classifier is more confident to tell x to be positive or negative

Now, how do we find $\theta_t = \left(h_{t,1},\dots,h_{t,B}\right)$?



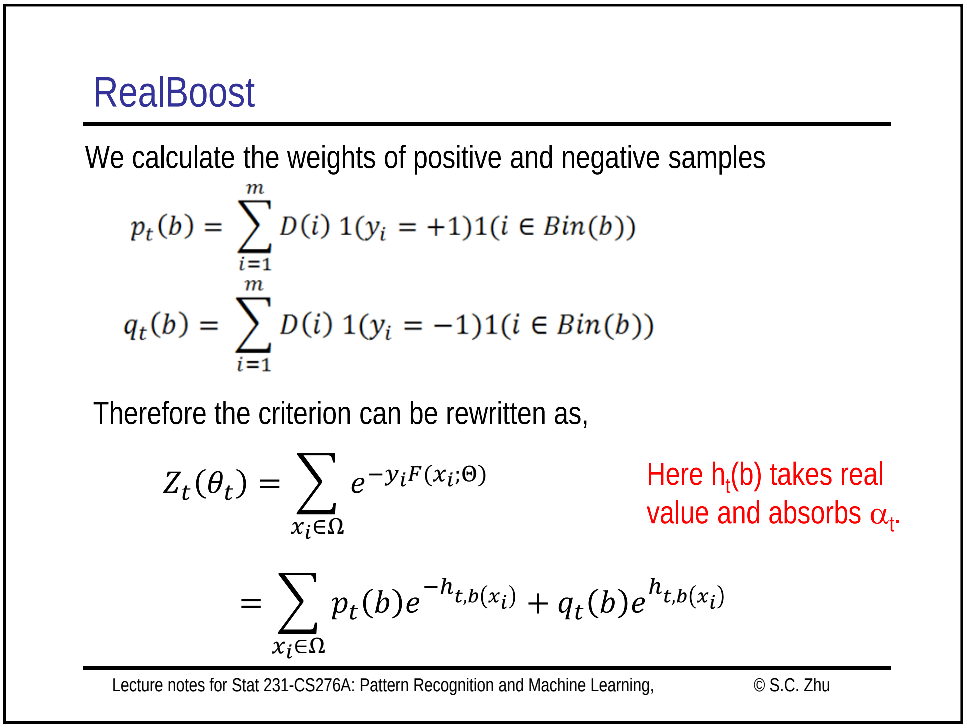
RealBoost

AdaBoost divides the space
(or empirically, the training set)
into 2-subsets:
i) those that are classified correctly by h().
ii) those that are classified wrong by h().

$$\begin{aligned} Z_t(\alpha_t, h_t) &= \sum_{x_i\in\Omega} D_t(x_i)\exp\{-y_i\alpha_t h_t(x_i)\} \\ &= \sum_{x_i\in A} D_t(x_i)e^{-\alpha_t} + \sum_{x_i\in\bar{A}} D_t(x_i)e^{\alpha_t} \end{aligned}$$

RealBoost divides the space into B subsets (bins), for each bin b=1,2,…,B.

We define
$p_t(b)$ and $q_t(b)$
to be the summed
weight of +/- in each bin



RealBoost

We calculate the weights of positive and negative samples

$$p_t(b) = \sum_{i=1}^{m} D(i)\ 1(y_i = +1)1(i \in Bin(b))$$

$$q_t(b) = \sum_{i=1}^{m} D(i)\ 1(y_i = -1)1(i \in Bin(b))$$

Therefore the criterion can be rewritten as,

$$Z_t(\theta_t) = \sum_{x_i\in\Omega} e^{-y_i F(x_i;\Theta)}$$

$$= \sum_{x_i\in\Omega} p_t(b)e^{-h_{t,b}(x_i)} + q_t(b)e^{h_{t,b}(x_i)}$$

Here $h_t(b)$ takes real value and absorbs α_t.

RealBoost

We rewrite the criterion as $Z_t(\theta_t) = Z_t(h_{t,1}, \dots, h_{t,B})$

$$= \sum_{b=1}^{B} p_t(b)e^{-h_{t,b}} + q_t(b)e^{+h_{t,b}}$$

Let $\frac{dZ}{dh_{t,b}} = 0$

We have $h_{t,b} = \frac{1}{2}\log\frac{p_t(b)}{q_t(b)}$

Plug in, we have $Z = 2\sum_{b=1}^{B}\sqrt{p_t(b)q_t(b)}$

More derivations on the blackboard.

RealBoost

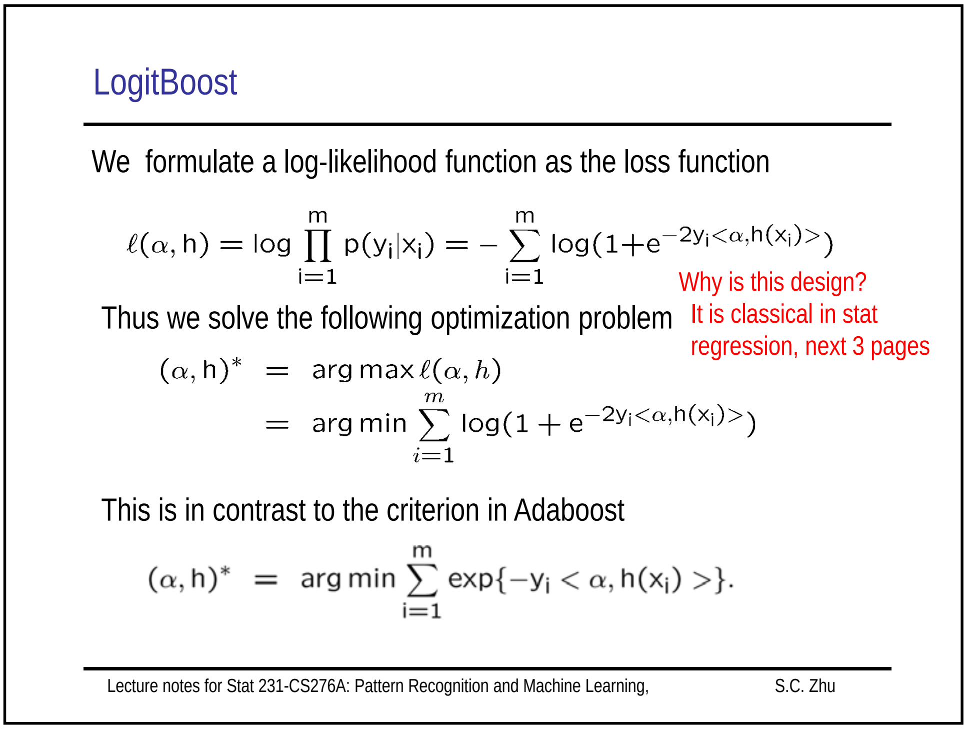
Then following the same procedure as in Adaboost.
At each iteration,

1, We choose the weak classifier to minimize

$$Z = 2\sum_{b=1}^{B}\sqrt{p_t(b)q_t(b)}$$

2, We update the weight of each data point by

$$D_t(x_i) = \frac{1}{Z_t}D_{t-1}(x_i)e^{-y_i h_{t,b(x_i)}}$$



LogitBoost

We formulate a log-likelihood function as the loss function

$$\ell(\alpha, h) = \log \prod_{i=1}^{m} p(y_i|x_i) = -\sum_{i=1}^{m} \log(1+e^{-2y_i<\alpha,h(x_i)>})$$

Why is this design?

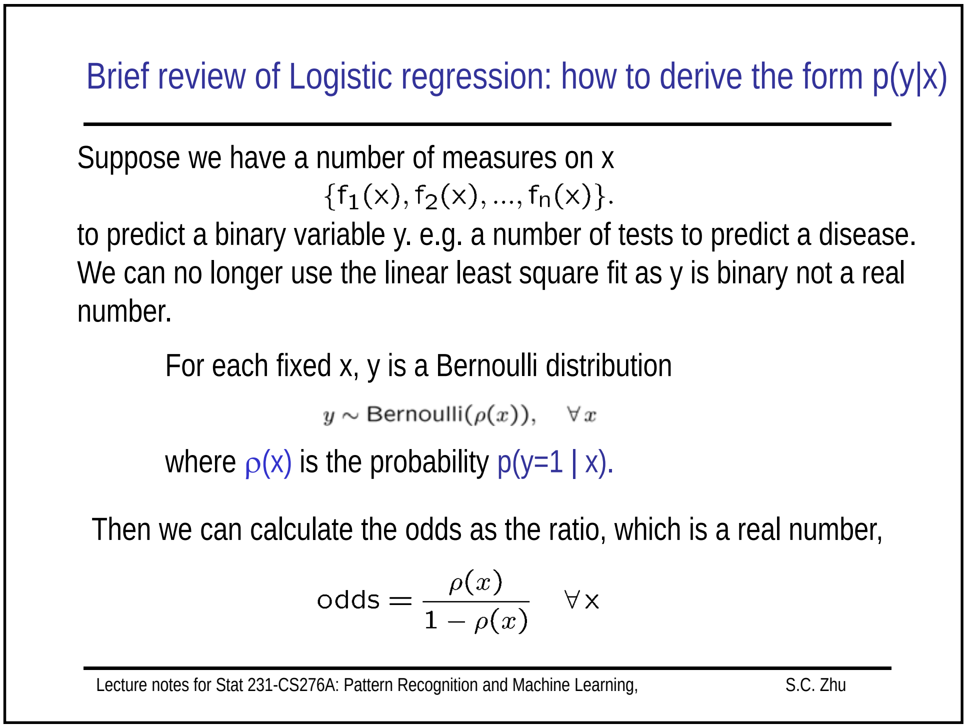
It is classical in stat regression, next 3 pages

Thus we solve the following optimization problem

$$\begin{aligned}(\alpha, h)^* &= \arg\max \ell(\alpha, h)\\ &= \arg\min \sum_{i=1}^{m} \log(1+e^{-2y_i<\alpha,h(x_i)>})\end{aligned}$$

This is in contrast to the criterion in Adaboost

$$(\alpha, h)^* = \arg\min \sum_{i=1}^{m} \exp\{-y_i < \alpha, h(x_i) >\}.$$



Brief review of Logistic regression: how to derive the form p(y|x)

Suppose we have a number of measures on x

$$\{f_1(x), f_2(x), ..., f_n(x)\}.$$

to predict a binary variable y. e.g. a number of tests to predict a disease. We can no longer use the linear least square fit as y is binary not a real number.

For each fixed x, y is a Bernoulli distribution

$$y \sim \text{Bernoulli}(\rho(x)), \quad \forall x$$

where ρ(x) is the probability p(y=1 | x).

Then we can calculate the odds as the ratio, which is a real number,

$$\text{odds} = \frac{\rho(x)}{1-\rho(x)} \quad \forall x$$

Brief review of Logistic regression

Then people fits the log-odd by a generalized linear regression,

$$\frac{1}{2}\log odds = \frac{1}{2}\log\frac{\rho(x)}{1-\rho(x)} = F(x;\Theta) = \alpha_1 h_1(x) + \cdots + \alpha_t h_t(x)$$

This fitting formula leads to the probability form used before

$$\rho(\mathrm{x}) = \mathrm{p}(\mathrm{y}=1|\mathrm{x}) = \frac{e^{<\alpha,h(x)>}}{e^{<\alpha,h(x)>} + e^{-<\alpha,h(x)>}} \quad \forall \mathrm{x}$$

$$\rho(x) = p(y=1|x) = \frac{\mathrm{e}^{\mathrm{F(x)}}}{\mathrm{e}^{\mathrm{F(x)}} + \mathrm{e}^{-\mathrm{F(x)}}} = \frac{1}{1+\mathrm{e}^{-2\mathrm{F(x)}}}$$

Brief review of Logistic regression

As we have

$$p(y=1|x) = \frac{\mathrm{e}^{\mathrm{F(x)}}}{\mathrm{e}^{\mathrm{F(x)}} + \mathrm{e}^{-\mathrm{F(x)}}} = \frac{1}{1+\mathrm{e}^{-2\mathrm{F(x)}}}$$

$$p(y=-1|x) = \frac{\mathrm{e}^{-\mathrm{F(x)}}}{\mathrm{e}^{\mathrm{F(x)}} + \mathrm{e}^{-\mathrm{F(x)}}} = \frac{1}{1+\mathrm{e}^{+2\mathrm{F(x)}}}$$

We rewrite

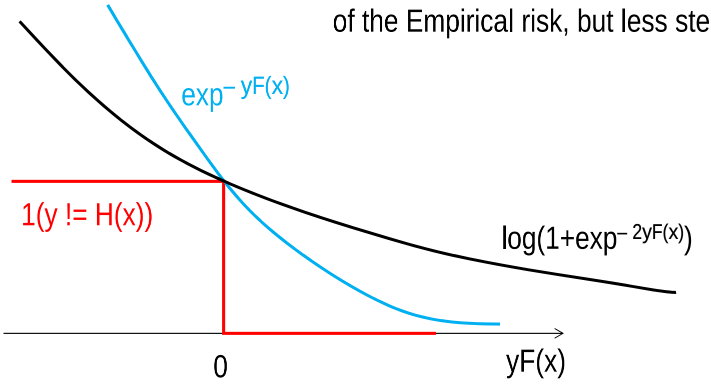
$$p(y|x) = \frac{1}{1+\mathrm{e}^{-2\mathrm{yF(x)}}}$$

$$\log p(y_i|x_i) = -\log(1+\mathrm{e}^{-2\mathrm{y_i F(x_i;\Theta)}})$$

LogitBoost

$$(\alpha, h)^* = \arg\min E_{p(x,y)}[\log(1 + e^{-2y<\alpha,h(x)>})]$$

This function is also an upper bound of the Empirical risk, but less steep.



Probabilistic interpretation of AdaBoost

Now we show a connection between Adaboost to Logistic regression

Given the labeled training data in supervised learning

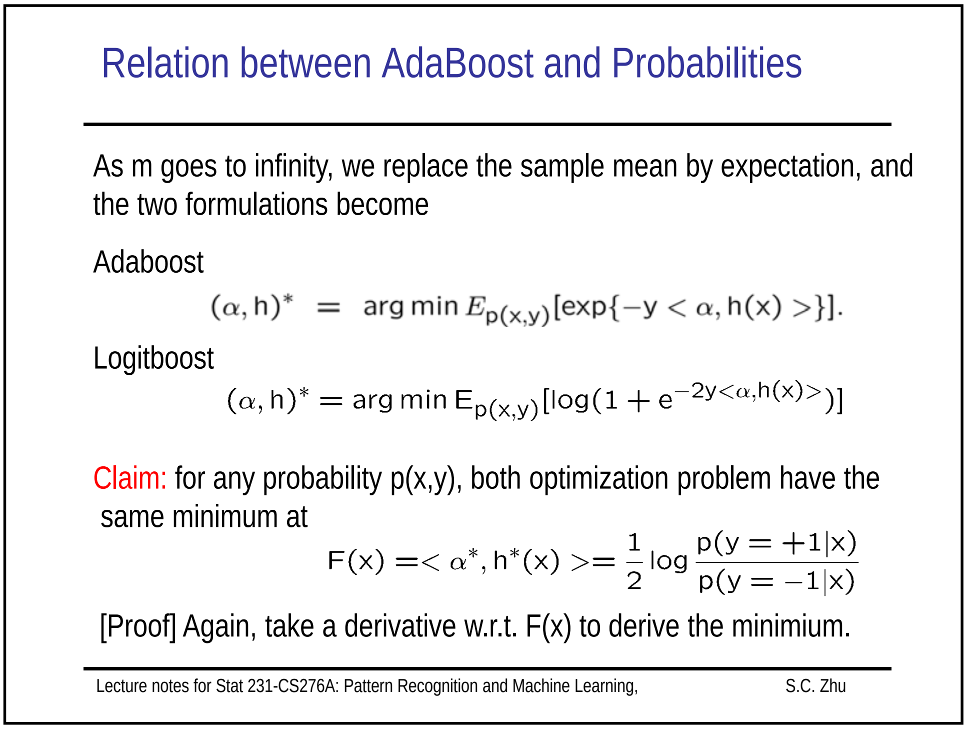
$$\Omega = \{(x_i, y_i) : x_i \in \Omega^d,\ y_i \in \{-1, 1\},\ i = 1, \ldots, m.\}$$

We can formulate a posterior probability

$$p(y|x) = \frac{1}{C}\exp\{-y < \alpha, h(x) >\}$$

$$p(y = 1|x) = \frac{e^{-<\alpha,h(x)>}}{e^{<\alpha h(x)>} + e^{-<\alpha,h(x)>}} = \frac{1}{1 + e^{-2<\alpha,h(x)>}}$$

$$p(y = -1|x) = \frac{e^{<\alpha,h(x)>}}{e^{<\alpha h(x)>} + e^{-<\alpha,h(x)>}} = \frac{1}{1 + e^{2<\alpha,h(x)>}}$$



Relation between AdaBoost and Probabilities

As m goes to infinity, we replace the sample mean by expectation, and the two formulations become

Adaboost

$$(\alpha, \mathrm{h})^* \;=\; \arg\min E_{\mathrm{p(x,y)}}[\exp\{-\mathrm{y} < \alpha, \mathrm{h(x)} >\}].$$

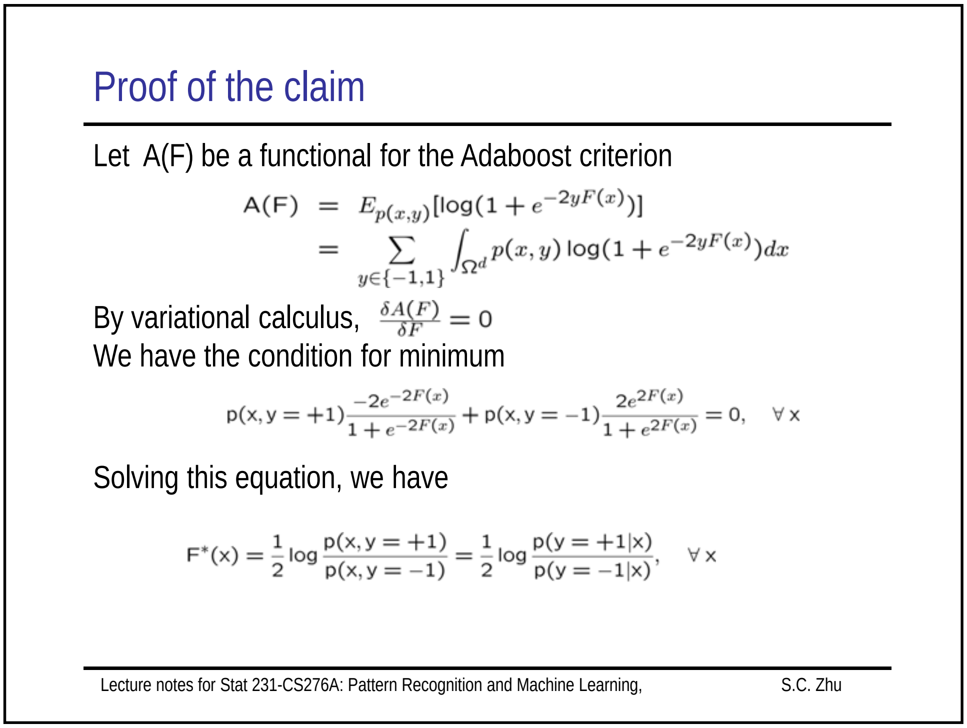
Logitboost

$$(\alpha, \mathrm{h})^* = \arg\min \mathrm{E}_{\mathrm{p(x,y)}}[\log(1 + \mathrm{e}^{-2\mathrm{y}<\alpha,\mathrm{h(x)}>})]$$

Claim: for any probability p(x,y), both optimization problem have the same minimum at

$$\mathrm{F(x)} =< \alpha^*, \mathrm{h}^*(\mathrm{x}) >= \frac{1}{2}\log\frac{\mathrm{p(y=+1|x)}}{\mathrm{p(y=-1|x)}}$$

[Proof] Again, take a derivative w.r.t. F(x) to derive the minimum.



Proof of the claim

Let A(F) be a functional for the Adaboost criterion

$$\begin{aligned}\mathrm{A(F)} &= E_{p(x,y)}[\log(1 + e^{-2yF(x)})] \\ &= \sum_{y\in\{-1,1\}} \int_{\Omega^d} p(x,y)\log(1 + e^{-2yF(x)})dx\end{aligned}$$

By variational calculus, $\frac{\delta A(F)}{\delta F} = 0$
We have the condition for minimum

$$\mathrm{p(x,y=+1)}\frac{-2e^{-2F(x)}}{1+e^{-2F(x)}} + \mathrm{p(x,y=-1)}\frac{2e^{2F(x)}}{1+e^{2F(x)}} = 0, \quad \forall \mathrm{x}$$

Solving this equation, we have

$$\mathrm{F}^*(\mathrm{x}) = \frac{1}{2}\log\frac{\mathrm{p(x,y=+1)}}{\mathrm{p(x,y=-1)}} = \frac{1}{2}\log\frac{\mathrm{p(y=+1|x)}}{\mathrm{p(y=-1|x)}}, \quad \forall \mathrm{x}$$

Proof of the claim

Similarly let B(F) be a functional for the Logitboost criterion

$$\begin{aligned} \mathrm{B(F)} &= E_{p(x,y)}[e^{-yF(x)}] \\ &= \sum_{y\in\{-1,1\}} \int_{\Omega^d} p(x,y)e^{-yF(x)}dx \end{aligned}$$

By variational calculus, $\frac{\delta B(F)}{\delta F} = 0$
We have the condition for minimum

$$-\mathrm{p(x, y = +1)e^{-F(x)} + p(x, y = -1)e^{F(x)} = 0}, \quad \forall\, \mathrm{x}$$

Solving this equation, we also have

$$\mathrm{F^*(x) = \frac{1}{2}\log\frac{p(x, y = +1)}{p(x, y = -1)} = \frac{1}{2}\log\frac{p(y = +1|x)}{p(y = -1|x)}}, \quad \forall\, \mathrm{x}$$

Relation between AdaBoost and Probabilities

Theorem

As m is large enough, we have

$$\mathrm{H(x) = sign(<\alpha^*, h^*(x)>) = sign(\log\frac{p(y = +1|x)}{p(y = -1|x)})}$$

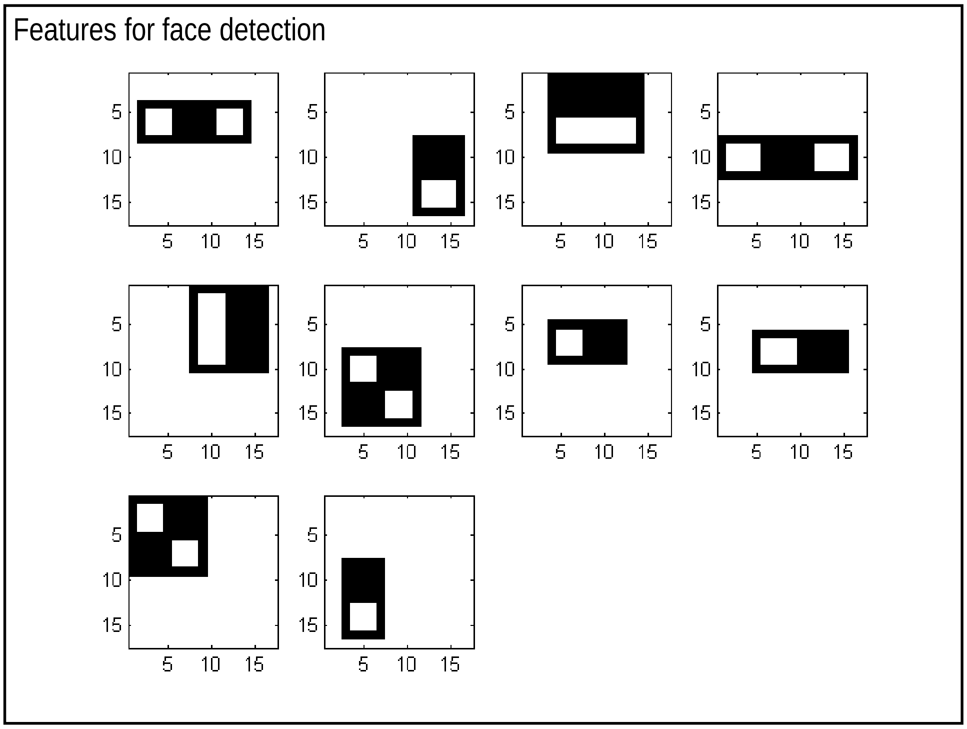
This again leads to the Bayesian decision and Bayes error, like the k-NN classifier leading to Bayes decision for large enough samples (asymptotically).

Note that this is different from the MLE learning. In MLE, we have to assume a joint probability

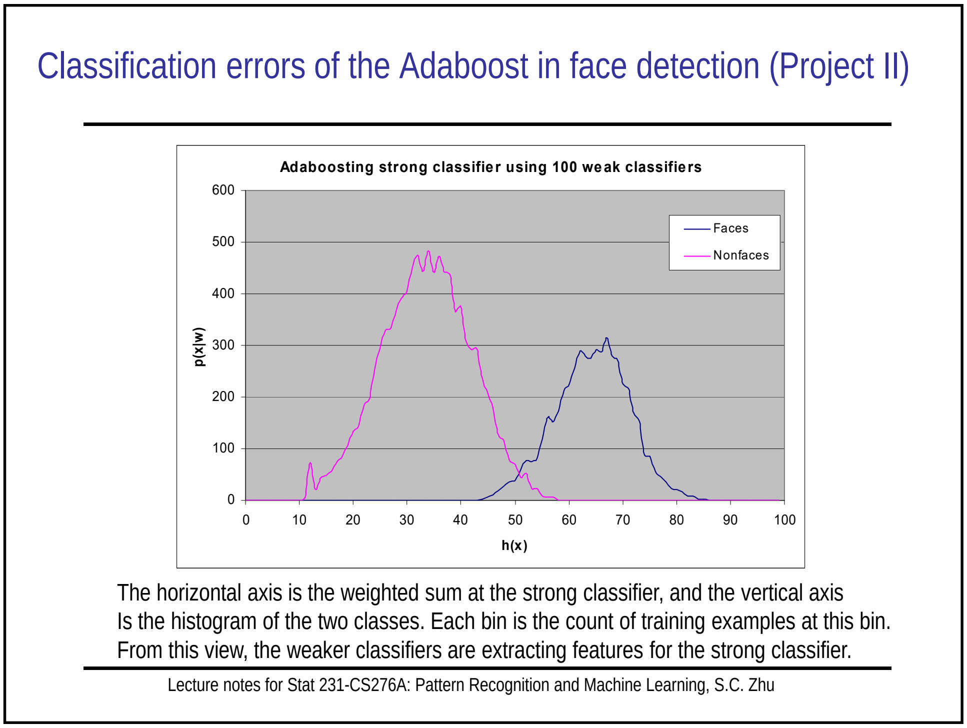
$$\mathrm{p(x, y) = \frac{1}{Z}\exp\{-y < \alpha, h(x) >\}}.$$

with $$\mathrm{Z} = \sum_{\mathrm{x}\in\Omega}\sum_{\mathrm{y}\in\{-1,1\}} \mathrm{\exp\{-y < \alpha, h(x) >\}}.$$

In the logitboost, the sum over x is dropped.



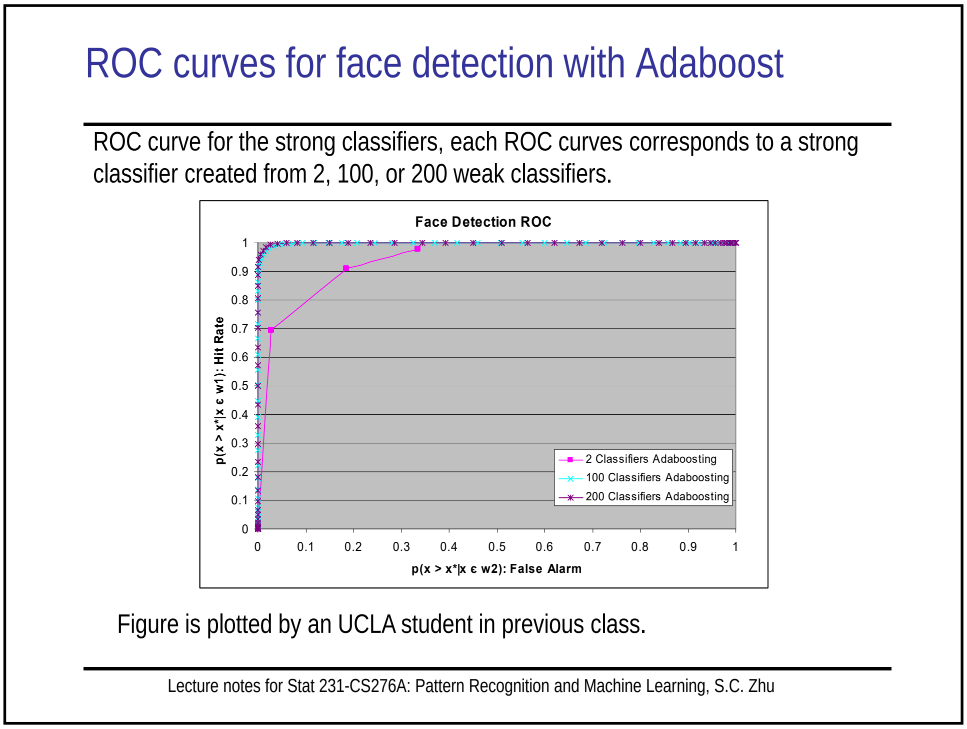
Features for face detection



Classification errors of the Adaboost in face detection (Project II)

The horizontal axis is the weighted sum at the strong classifier, and the vertical axis
Is the histogram of the two classes. Each bin is the count of training examples at this bin.
From this view, the weaker classifiers are extracting features for the strong classifier.

Lecture notes for Stat 231-CS276A: Pattern Recognition and Machine Learning, S.C. Zhu

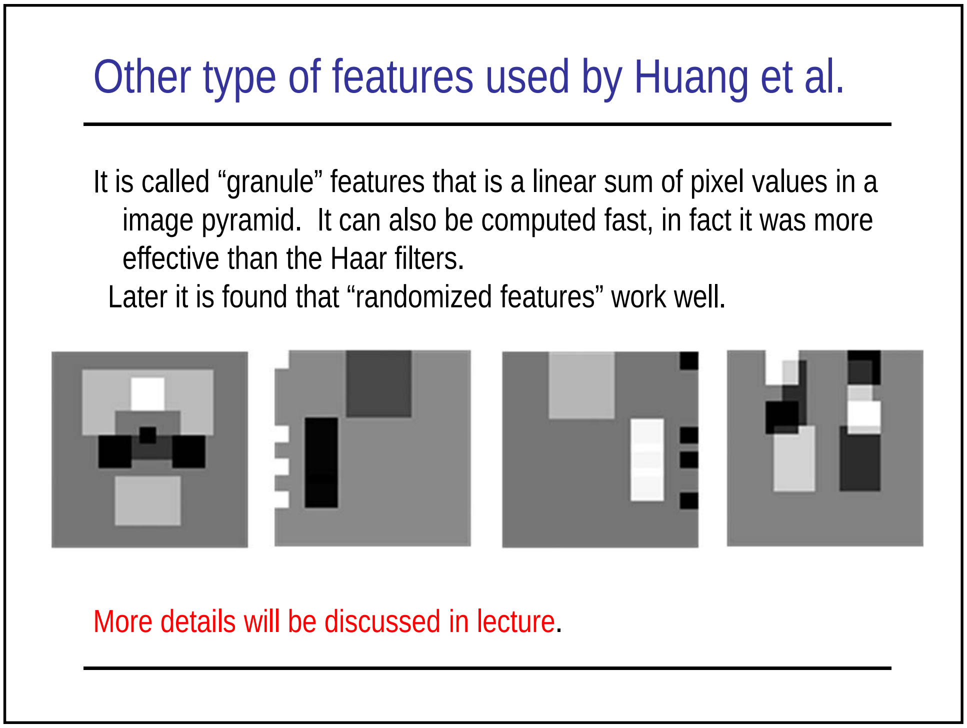


ROC curves for face detection with Adaboost

ROC curve for the strong classifiers, each ROC curves corresponds to a strong classifier created from 2, 100, or 200 weak classifiers.

Figure is plotted by an UCLA student in previous class.

Lecture notes for Stat 231-CS276A: Pattern Recognition and Machine Learning, S.C. Zhu



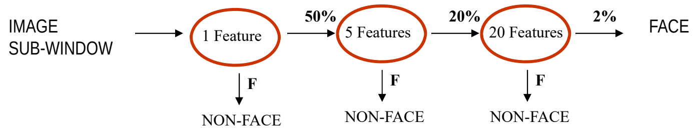
Other type of features used by Huang et al.

It is called “granule” features that is a linear sum of pixel values in a image pyramid. It can also be computed fast, in fact it was more effective than the Haar filters.

Later it is found that “randomized features” work well.

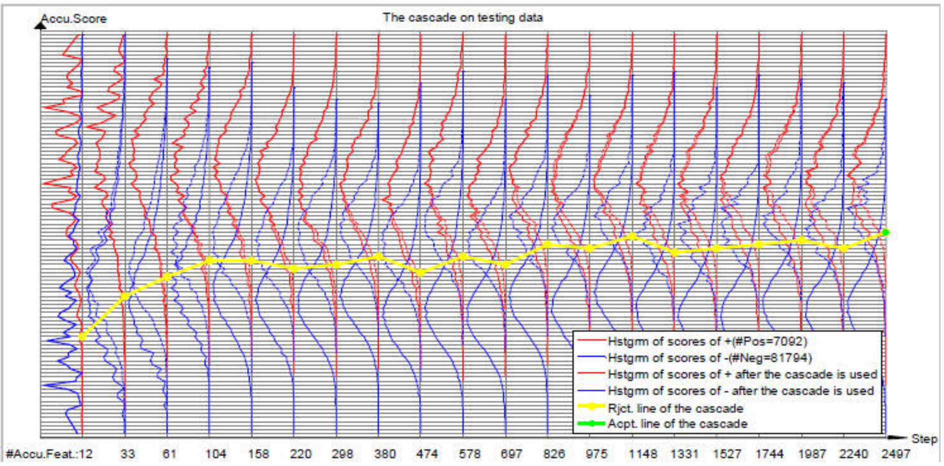
More details will be discussed in lecture.

Cascaded Classifier

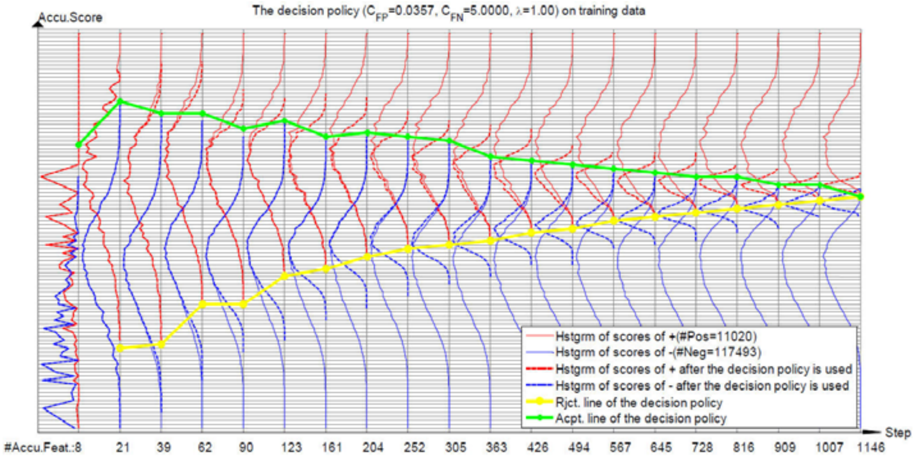


- A 1 feature classifier achieves 100% detection rate and about 50% false positive rate.
- A 5 feature classifier achieves 100% detection rate and 40% false positive rate (20% cumulative)
  - using data from previous stage.
- A 20 feature classifier achieve 100% detection rate with 10% false positive rate (2% cumulative)

Computational topics: Cascade and Decision Policy

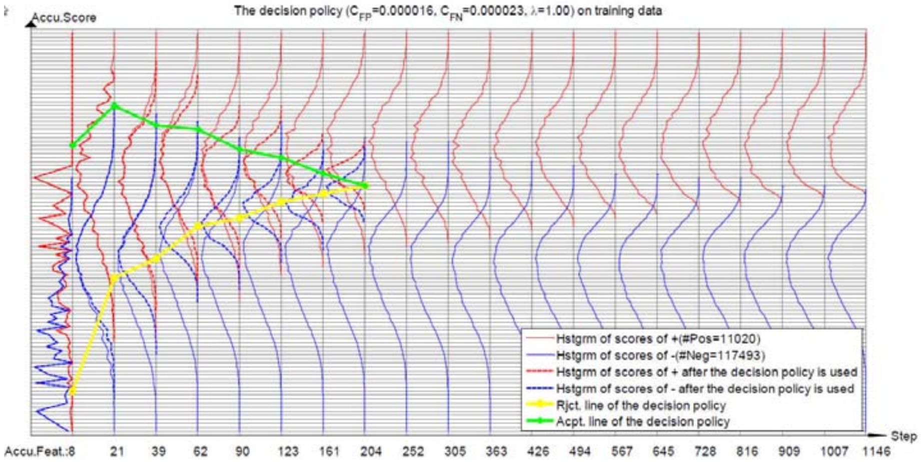


Computational topics: Cascade and Decision Policy

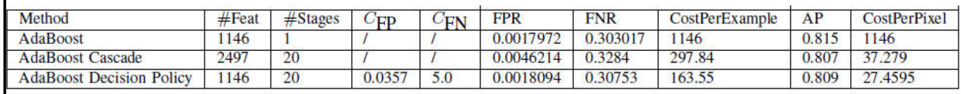


T.F. Wu and S.C. Zhu, “Learning Near-Optimal Cost-Sensitive Decision Policy for Object Detection,” Int'l Conf. on Computer vision, 2013.

Computational topics: Cascade and Decision Policy



Computational topics: Cascade and Decision Policy



Method	#Feat	#Stages	C_{FP}	C_{FN}	FPR	FNR	CostPerExample	AP	CostPerPixel
AdaBoost	1146	1	/	/	0.0017972	0.303017	1146	0.815	1146
AdaBoost Cascade	2497	20	/	/	0.0046214	0.3284	297.84	0.807	37.279
AdaBoost Decision Policy	1146	20	0.0357	5.0	0.0018094	0.30753	163.55	0.809	27.4595