1

Contents

2

1 Boosting

Boosting: Motivation (1/2)

- Suppose we are to build an email filter that can distinguish spam (junk) emails from non-spam
  1. Collect as many examples as possible of spam and non-spam emails
  2. Use some machine learning approach to produce a classification or prediction rule
  3. Given a new email, apply the classification rule to predict if it is a junk mail or not

- Key observations
  - Building a highly accurate prediction rule is difficult
  - However, it is not hard to come up with rough rules of thumb that are only moderately accurate

3

Boosting: Motivation (2/2)

- Boosting is based on the observation that finding many rough rules of thumb can be a lot easier than finding a single, highly accurate prediction rule

- The steps of boosting
  1. Repeatedly call a base algorithm/method for finding the rules of thumb (called weak learners in boosting)
  2. Each time the base algorithm is called, with a different distribution/weighting of the training examples
  3. Each time your base algorithm is called, it generates a new weak prediction rule
  4. After many rounds, the boosting algorithm must combine these weak rules into a single prediction rule

4

Boosting: Two Fundamental Issues

- How should the distribution on the training examples be chosen on each round?
  - Focusing on the hardest examples

- How should the weak rules be combined into a single rule?
  - Taking a (weighted) majority vote of the predictions from these weak rules

5

Boosting: A Brief History about Boosting

- Kearn & Valiant (1988) pose the question whether a weak learning algorithm can be boosted into an arbitrarily accurate strong learning algorithm.
- Schapire (1989): The first provable polynomial-time boosting algorithm
- Freund & Schapire (1995): AdaBoost
  (2003 Gödel prize)

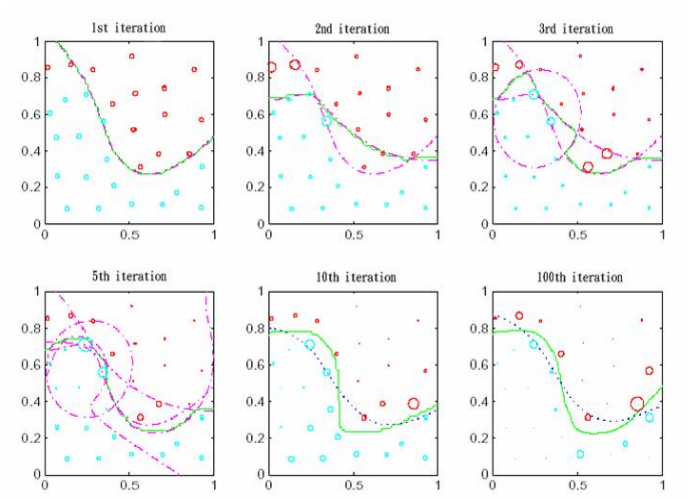
Boosting: AdaBoost Algorithm

Given: $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)$**,** $\mathbf{x}_i \in X$ **and** $y_i \in Y = \{-1, 1\}$ **Initialize the data weight distribution** $D_1(i) = 1/m$

- For $t = 1, \ldots, T$
  1. Call WeakLearn using distribution D_t and get back binary h_t
  2. Evaluate the weighted error of $h_t : X \to \{-1, +1\}$
  $$\varepsilon_t = \Pr_{i \sim D_t}[h_t(\mathbf{x}_i) \neq y_i]$$
  3. Choose $\alpha_t = \frac{1}{2} \ln \left(\frac{1-\varepsilon_t}{\varepsilon_t} \right)$
  4. Update data weight
  $$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(\mathbf{x}_i))}{Z_t}$$
  where Z_t is a normalization factor such that D_{t+1} is a distribution.

  Output final classifier: $H(\mathbf{x}) = \text{sign}(f(\mathbf{x})) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})\right)$

Boosting: Illustration



Boosting: Analyzing Training Error (1/5)

- The training error of the final classifier is bounded by

$$\frac{1}{m}|\{i : H(\mathbf{x}_i) \neq y_i\}| \leq \frac{1}{m}\sum_i \exp(-y_i f(\mathbf{x}_i)) = \prod_{t=1}^{T} Z_t$$

Since $D_{t+1}(i) = \frac{D_t(i)\exp(-\alpha_t y_i h_t(\mathbf{x}_i))}{Z_t}$, we have

$$\prod_{t=1}^{T} Z_t = \frac{D_1(i)\exp(-\alpha_1 y_i h_1(\mathbf{x}_i))}{D_2(i)} \times \cdots \times \frac{D_T(i)\exp(-\alpha_T y_i h_T(\mathbf{x}_i))}{D_{T+1}(i)}$$

$$\Longrightarrow D_{T+1}(i)\prod_{t=1}^{T} Z_t = D_1(i)\exp\left(-y_i\sum_{t=1}^{T}\alpha_t h_t(\mathbf{x}_i)\right)$$

$$\Longrightarrow \prod_{t=1}^{T} Z_t \sum_i D_{T+1}(i) = \frac{1}{m}\sum_i \exp(-y_i f(\mathbf{x}_i))$$

Boosting: Analyzing Training Error (2/5)

- Thus the idea is to minimize the error bound

$$\frac{1}{m}|\{i : H(\mathbf{x}_i) \neq y_i\}| \leq \frac{1}{m}\sum_i \exp(-y_i f(\mathbf{x}_i)) = \prod_{t=1}^{T} Z_t$$

However, since AdaBoost is an iterative algorithm, we instead greedily minimize Z_t at each iteration t.

Since $D_{t+1}(i) = \frac{D_t(i)\exp(-\alpha_t y_i h_t(\mathbf{x}_i))}{Z_t}$ and $\sum_i D_{t+1}(i) = 1$, we have

$$Z_t = \sum_i D_t(i)\exp(-\alpha_t y_i h_t(\mathbf{x}_i))$$

Boosting: Analyzing Training Error (3/5)

- Suppose weak learner h_t is chosen at iteration t. With

$$\begin{aligned} Z_t &= \textstyle\sum_i D_t(i)\exp(-\alpha_t y_i h_t(\mathbf{x}_i)) \\ \varepsilon_t &= \textstyle\sum_i D_t(i)\mathbf{1}[y_i \neq h_t(\mathbf{x}_i)] \end{aligned}$$

α_t can be decided by investigating the following:

$$\begin{aligned} \frac{\partial Z_t}{\partial \alpha_t} &= \textstyle\sum_i (-y_i h_t(\mathbf{x}_i)) D_t(i)\exp(-\alpha_t y_i h_t(\mathbf{x}_i)) \\ &= \textstyle\sum_{\times} D_t(i)\exp(\alpha_t) - \sum_{\surd} D_t(i)\exp(-\alpha_t) \\ &= \varepsilon_t \times \exp(\alpha_t) - (1-\varepsilon_t)\times\exp(-\alpha_t) = 0 \end{aligned}$$

$$\boxed{\alpha_t^* = \frac{1}{2}\ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)}$$

Boosting: Analyzing Training Error (4/5)

- With $\alpha_t^* = \frac{1}{2}\ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$,

$$\begin{aligned} Z_t^* &= \textstyle\sum_i D_t(i)\exp(-\alpha_t^* y_i h_t(\mathbf{x}_i)) \\ &= \textstyle\sum_{\surd} D_t(i)\exp(-\alpha_t^*) + \sum_{\times} D_t(i)\exp(\alpha_t^*) \\ &= \sqrt{\frac{\varepsilon_t}{1-\varepsilon_t}} \times \textstyle\sum_{\surd} D_t(i) + \sqrt{\frac{1-\varepsilon_t}{\varepsilon_t}} \times \sum_{\times} D_t(i) \\ &= \sqrt{\frac{\varepsilon_t}{1-\varepsilon_t}} \times (1-\varepsilon_t) + \sqrt{\frac{1-\varepsilon_t}{\varepsilon_t}} \times \varepsilon_t \\ &= 2\sqrt{\varepsilon_t(1-\varepsilon_t)} \end{aligned}$$

Boosting: Analyzing Training Error (5/5)

- Let $\gamma_t = \frac{1}{2} - \varepsilon_t$. From $Z_t^* = 2\sqrt{\varepsilon_t(1-\varepsilon_t)}$, we have

$$\prod_{t=1}^{T} Z_t^* = \prod_{t=1}^{T} 2\sqrt{\varepsilon_t(1-\varepsilon_t)} = \prod_{t=1}^{T}\sqrt{1-4\gamma_t^2} \le \exp\left(-2\sum_{t=1}^{T}\gamma_t^2\right)$$

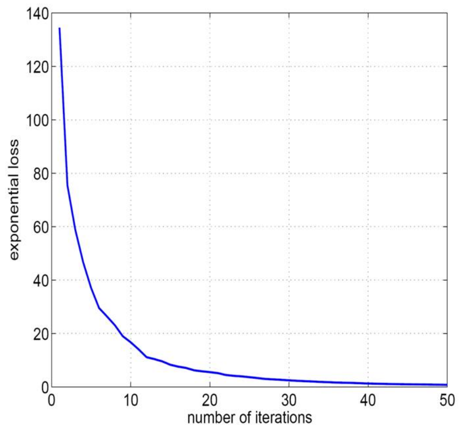
Thus, if each weak learner h_t is slightly better than random then $\exists\gamma > 0$ such that $\gamma_t \ge \gamma > 0$ for $t = 1, \dots, T$. Hence

$$\boxed{\prod_{t=1}^{T} Z_t^* \le e^{-2T\gamma^2} \to 0, \quad \text{as } T \uparrow}$$

13

Boosting: Exponential Loss

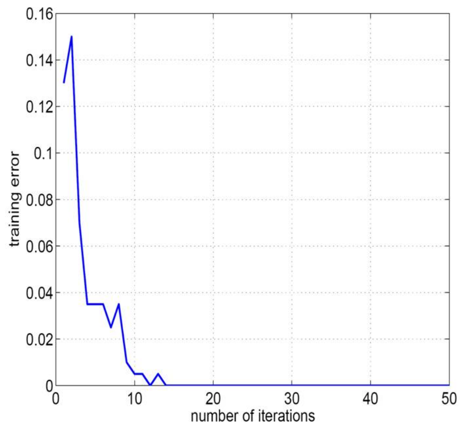
- Assuming all selected weak learners h_t satisfy $\varepsilon_t < \frac{1}{2}$, the ensemble classifier H is then guaranteed to have a lower exponential loss over the training examples.



14

Boosting: Training Error

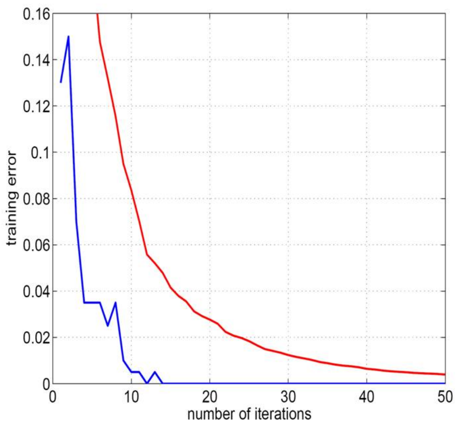
- Assuming all selected weak learners h_t satisfy $\varepsilon_t < \frac{1}{2}$, the boosting iterations also exponentially decrease the training error.



15

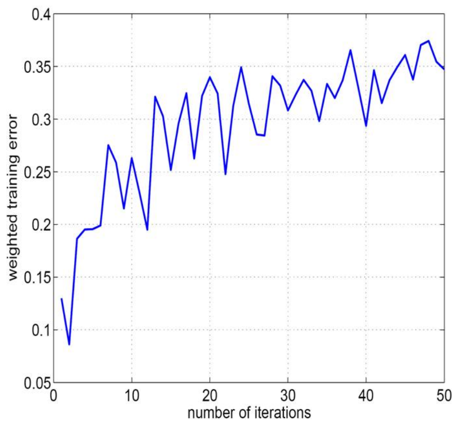
Boosting: Training Error Bounded by Exponential Loss

- Assuming all selected weak learners h_t satisfy $\varepsilon_t < \frac{1}{2}$, the boosting iterations also exponentially decrease the training error bounded by exponential loss.



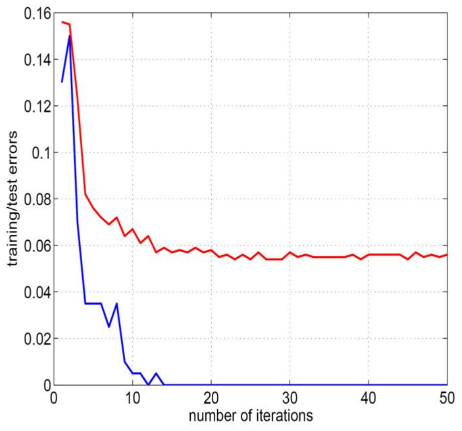
Boosting: Weighted Error of a Weak Learner

- Weighted error $\varepsilon_t = \sum_{\times} D_t(i)$ by h_t tends to be increasing with respect to boosting iterations.



Boosting: Typical AdaBoost Performance

- It has been observed, quite often, the test error may still go down when the training error has already reached zero.



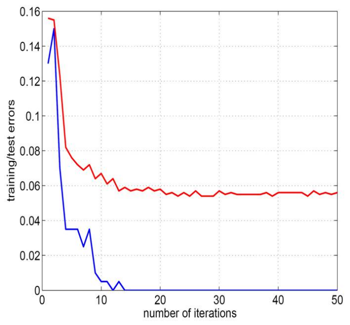
Boosting: Generalization Error (1/5)

- Assume training and testing samples are generated i.i.d. from some unknown distribution on $X \times X$.
- Also assume that all weak learners are binary.
- Generalization error here means the probability of misclassifying a new example, while the test error is the fraction of mistakes on a newly sampled test set (i.e., generalization error is the expected test error).

Boosting: Generalization Error (2/5)

- Freund and Schapire show that the generalization error, with high probability, is at most

$$\hat{\Pr}[H(\mathbf{x}) \neq y] + \tilde{O}\left(\sqrt{\frac{Td}{m}}\right)$$



d : VC-dimension of the weak learner space
T : # of boosting iterations
m : the size of training samples

Boosting: Generalization Error (3/5)

- For boosting, the margin of example $(\mathbf{x}, y)$ is defined to be

$$\text{margin}_f(\mathbf{x}, y) = \frac{yf(\mathbf{x})}{\|f\|} = \frac{yf(\mathbf{x})}{\sum_t |\alpha_t|} = \frac{y\sum_t \alpha_t h_t(\mathbf{x})}{\sum_t |\alpha_t|} \in [-1, +1]$$

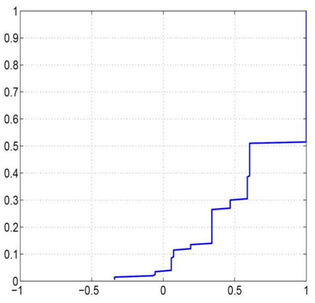
- Schapire et al. show that for any $\theta > 0$ the generalization error, with high probability, is at most

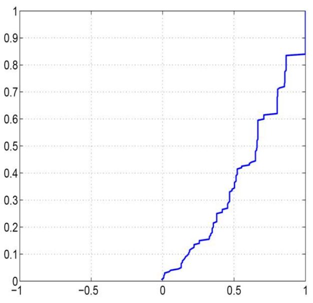
$$\hat{\Pr}[\text{margin}_f(\mathbf{x}, y) \leq \theta] + \tilde{O}\left(\sqrt{\frac{d}{m\theta^2}}\right)$$

Note that the above bound is now independent of T, the number of iterations of boosting.

Boosting: Generalization Error (4/5)

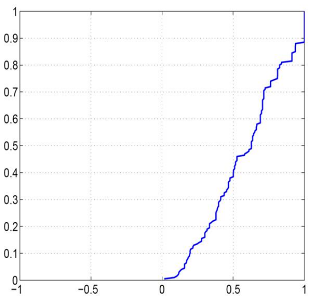
- Cumulative distribution of margin values versus the number of boosting iterations.

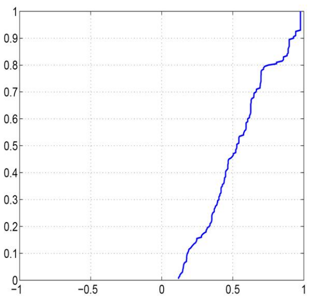

4 iterations


10 iterations

Boosting: Generalization Error (5/5)

- Cumulative distribution of margin values versus the number of boosting iterations.


20 iterations


50 iterations

2 Useful Notes

Basic Information Theory (1/6)

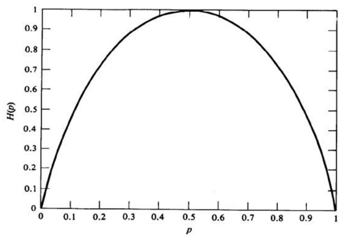
- The *entropy* $H(X)$ of a random variable X is a measure of *uncertainty* of a random variable, and is defined by

$$H(X) = -\sum_{x\in\mathscr{X}} p(x)\log p(x), \text{ where } \mathscr{X} \text{ is the alphabet of } X.$$

- Example:

$$X = \begin{cases} 1 & \text{with probability } p, \\ 0 & \text{with probability } 1-p. \end{cases}$$

$$H(X) = -p\times\log p - (1-p)\times\log(1-p) \equiv H(p)$$



Basic Information Theory (2/6)

- The *joint entropy* $H(X,Y)$ of a pair of random variables X and Y with a joint distribution $p(x,y)$ is defined by

$$H(X,Y) = -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x,y)\log p(x,y)$$

- The *conditional entropy* $H(Y|X)$ is defined as

$$\begin{aligned} H(Y|X) &= \sum_{x\in\mathcal{X}} p(x)H(Y|X=x) \\ &= -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x,y)\log p(y|x) \end{aligned}$$

- (Chain rule)

$$\begin{aligned} H(X,Y) &= H(X)+H(Y|X) \\ &= H(Y)+H(X|Y) \end{aligned}$$

Basic Information Theory (3/6)

- Example

$$H(X) = H\left(\frac{1}{2},\frac{1}{4},\frac{1}{8},\frac{1}{8}\right) = \frac{7}{4}\,\text{bits}$$

$$H(Y) = H\left(\frac{1}{4},\frac{1}{4},\frac{1}{4},\frac{1}{4}\right) = 2\,\text{bits}$$

$$H(X|Y) = \frac{11}{8}\,\text{bits},\quad H(Y|X) = \frac{13}{8}\,\text{bits},\quad H(X,Y) = \frac{27}{8}\,\text{bits}$$

Y X	1	2	3	4
1	$\frac{1}{8}$	$\frac{1}{16}$	$\frac{1}{32}$	$\frac{1}{32}$
2	$\frac{1}{16}$	$\frac{1}{8}$	$\frac{1}{32}$	$\frac{1}{32}$
3	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$
4	$\frac{1}{4}$	0	0	0

Basic Information Theory (4/6)

- The *relative entropy* or *Kullback-Leibler distance* between two probability mass functions $p(x)$ and $q(x)$ is defined as

$$D(p\|q) = \sum_{x\in\mathcal{X}} p(x)\log\frac{p(x)}{q(x)}$$

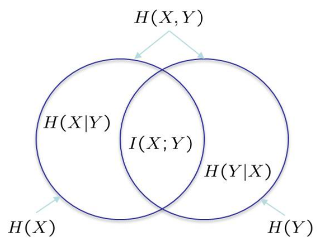
  $D(p\|q)$ is a measure of the inefficiency of assuming the distribution is q when the true distribution is p.

- The *mutual information* between two random variables X and Y is the relative entropy between the joint distribution $p(x,y)$ and the product distribution $p(x)p(y)$

$$\begin{aligned} I(X;Y) &= D(p(x,y)\|p(x)p(y)) \\ &= \sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x,y)\log\frac{p(x,y)}{p(x)p(y)} \end{aligned}$$

Basic Information Theory (5/6)

- Mutual information $I(X;Y)$: a measure of the amount of information that one random variable contains about another random variable. (i.e., the reduction in the uncertainty of one random variable due to the knowledge of the other)



$$\begin{aligned} &I(X;Y) \\ &= H(X) - H(X|Y) \\ &= H(Y) - H(Y|X) \\ &= H(X) + H(Y) - H(X,Y) \end{aligned}$$

28

Basic Information Theory (6/6)

- Bhattacharyya coefficient $\mathrm{BC}(p,q)$ is used in a measure of the distance between two discrete random variables with probability mass distributions $p(x)$ and $q(x)$, $x \in \mathscr{X}$.

$$\mathrm{BC}(p,q) = \sum_{x \in \mathscr{X}} \sqrt{p(x)q(x)}$$

$BC(p,q) \to 1 \Leftrightarrow p$ and q are more "similarly" distributed.

29

3 Boosting and Game Theory

Game Theory: Two-Person Zero-Sum Games (1/5)

- Two-person zero-sum games: games with only two players where one player wins what the other player loses.

- The strategic form of a two-person zero-sum game is given by a triplet (X,Y,A)
  1. X is a nonempty set, the set of strategies of Player I (the row player)
  2. Y is a nonempty set, the set of strategies of Player II (the column player)
  3. A is a real-valued function defined on $X \times Y$

  Player I chooses some $x \in X$ and Player II chooses $y \in Y$. $A(x,y)$ represents the winnings of I and the losses of II. (can be positive or negative)

30

Game Theory: Two-Person Zero-Sum Games (2/5)

- A two-person zero-sum game (X,Y,A) is said to be a finite game if both X and Y are finite sets.

- The Minmax Theorem

  For every finite two-person zero-sum game,
  1. There is a number V, called the value of the game
  2. There is a mixed strategy for Player I such that I's average gain is at least V no matter what II does
  3. There is a mixed strategy for Player II such that II's average loss is at most V no matter what I does.

31

Game Theory: Two-Person Zero-Sum Games (3/5)

- **Example**: (Odd or Even) Players I and II simultaneously call out one of the numbers one or two.

  If the sum of the two numbers is odd, Player I wins the same amount of reward. Otherwise, Player II wins.

$$\begin{array}{cc|c} & & \text{Player II (even)} \\ & & y \in Y = \{1,2\} \\ & & \begin{array}{cc} 1 & 2 \end{array} \\ \text{Player I (odd)} & 1 & \multirow{2}{*}{$\begin{pmatrix} -2 & +3 \\ +3 & -4 \end{pmatrix}$} \\ x \in X = \{1,2\} & 2 & \end{array}$$

- Let p be the portion of times Player I calls one.

$$\left.\begin{array}{rl} -2p+3(1-p) & \text{Player II calls one} \\ 3p-4(1-p) & \text{Player II calls two} \end{array}\right\} \begin{array}{l} p = \frac{7}{12} \text{ for Player I to call one is} \\ \text{the optimal strategy of Player I} \end{array}$$

  Note that $\left(-2 \times \frac{7}{12} + 3 \times \frac{5}{12}\right) = \frac{1}{12}$ is the value of the game.

32

Game Theory: Two-Person Zero-Sum Games (4/5)

- A finite two-person zero-sum game in strategy form, (X, Y, A), is sometimes called a matrix game because the payoff function can be represented as a matrix.

  If $X = \{x_1, \ldots, x_m\}$ and $Y = \{y_1, \ldots, y_n\}$, then by the *game matrix* or *payoff matrix* we mean the matrix

$$A = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix} \quad \text{where } a_{ij} = A(x_i, y_j)$$

- A mixed strategy for Player I may be represented by an m-tuple, $\mathbf{p} = (p_1, p_2, \ldots, p_m)$ of probabilities that add to 1. If I uses the mixed strategy $\mathbf{p}$ and II chooses column j, then the (average) payoff to I is $\sum_{i=1}^{m} p_i a_{ij}$.

  On the other hand, if II uses $\mathbf{q} = (q_1, q_2, \ldots, q_n)$ and I uses row i, the payoff to I is $\sum_{j=1}^{n} a_{ij} q_j$.

33

Game Theory: Two-Person Zero-Sum Games (5/5)

- A finite two-person zero-sum game in strategy form, (X, Y, A), is sometimes called a matrix game because the payoff function can be represented as a matrix.

  If $X = \{x_1, \ldots, x_m\}$ and $Y = \{y_1, \ldots, y_n\}$, then by the *game matrix* or *payoff matrix* we mean the matrix

$$A = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix} \quad \text{where } a_{ij} = A(x_i, y_j)$$

- More generally, if I uses the mixed strategy $\mathbf{p}$ and II uses the mixed strategy $\mathbf{q}$, the (average) payoff to I is

$$\mathbf{p}^T A \mathbf{q} = \sum_{i=1}^{m} \sum_{j=1}^{n} p_i a_{ij} q_j$$

34

Pronounced differences: Story of a romance, from model to muddle

She pronounced “a Tupple”, he said always “Toople”,
But both knew that in life one has to be supple.
Before long they huddled while browsing Google or watching Ted Koppel,
And all thought there wasn’t more perfect a couple.
He was from just Seattle, her folks straight from the Shtetl,
Yet their love — so it seemed — could suffer not even a ripple,
For he kept to the rule: be nimble, don’t ogle her nipple,
And don’t ever discuss the phonemes of “Tuple”.
Each morning they parted for the city bustle,
He walking the poodle, she coding in Eiffel at Apple;
Alas! Nights at home saw the fights redouble
When she brought up tupples and he cried “It’s Tooooople!”.
From splits most subtle, a marriage can topple:
First rabble, then rubble, theirs soon wasn’t worth a ruble.
She stuck hard to Tupple, he pushed still for Toople;
Two tickets to Reno corrected the trouble.

— *quoted from Journal of Object Technology, 2003*

Boosting as Two-Person Zero-Sum Games

- Instead of working with a payoff matrix A, we now consider a loss matrix M, again, of the row player.
- Boosting with training samples $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$ and a set of binary weak classifiers $\mathscr{H} = \{h_1, \ldots, h_n\}$ can be viewed as repeated play of the game matrix $M_{m\times n}$ where

$$M_{ij} = \begin{cases} 1 & \text{if } h_j(\mathbf{x}_i) = y_i \\ 0 & \text{otherwise.} \end{cases}$$

- The row player is the boosting algorithm: to decide a distribution D_t over $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$ becomes a distribution P over rows of M, while the base learner’s choice of of a weak classifier h_t becomes the choice of a column j of M.

von Neumann’s Minimax Theorem

$$\max_Q \min_P P^T M Q = \min_P \max_Q P^T M Q$$

4 Boosting and Logistic Regression

Boosting and Logistic Regression (1/5)

- For estimating the probability of a label, Friedman et al. suggest using a logistic function, and estimating

$$P(y = 1|\mathbf{x}) = \frac{e^{f(\mathbf{x})}}{e^{f(\mathbf{x})} + e^{-f(\mathbf{x})}} = \frac{1}{1 + e^{-2f(\mathbf{x})}} \qquad (*)$$

- **Note 1**
  $E\left\{e^{-yf(\mathbf{x})}\right\}$ is minimized at $f(\mathbf{x}) = \frac{1}{2}\ln\frac{p(y=1|\mathbf{x})}{p(y=-1|\mathbf{x})}$.

$$P(y = 1|\mathbf{x}) = \frac{e^{f(\mathbf{x})}}{e^{f(\mathbf{x})} + e^{-f(\mathbf{x})}}, \quad P(y = -1|\mathbf{x}) = \frac{e^{-f(\mathbf{x})}}{e^{f(\mathbf{x})} + e^{-f(\mathbf{x})}}$$

Proof

$$E\left\{e^{-yf(\mathbf{x})}\right\} = P(y=1|\mathbf{x})e^{-f(\mathbf{x})} + P(y=-1|\mathbf{x})e^{f(\mathbf{x})}$$
$$\frac{\partial E\left\{e^{-yf(\mathbf{x})}\right\}}{\partial f(\mathbf{x})} = -P(y=1|\mathbf{x})e^{-f(\mathbf{x})} + P(y=-1|\mathbf{x})e^{f(\mathbf{x})} = 0$$

37

Boosting and Logistic Regression (2/5)

- **Note 2**

Data likelihood is

$$\prod_{y_i=+1} p(y_i=1 \mid \mathbf{x}_i) \prod_{y_i=-1} p(y_i=-1 \mid \mathbf{x}_i)$$

$$\begin{aligned} \text{log loss} &= \text{negative log likelihood} \\ &= -\ln\left(\prod_{y_i=+1}\frac{1}{1+e^{-2f(\mathbf{x}_i)}}\prod_{y_i=-1}\frac{1}{1+e^{2f(\mathbf{x}_i)}}\right) \\ &= \sum_{y_i=+1}\ln\left(1+e^{-2f(\mathbf{x}_i)}\right) + \sum_{y_i=-1}\ln\left(1+e^{2f(\mathbf{x}_i)}\right) \\ &= \sum_{i=1}^{m}\ln\left(1+e^{-2y_if(\mathbf{x}_i)}\right) \end{aligned}$$

38

Boosting and Logistic Regression (3/5)

- **Note 3**

$E\left\{\ln\left(1+e^{-2yf(\mathbf{x})}\right)\right\}$ is minimized at $f(\mathbf{x}) = \frac{1}{2}\ln\frac{p(y=1|\mathbf{x})}{p(y=-1|\mathbf{x})}$

proof

$$E\left\{\ln\left(1+e^{-2yf(\mathbf{x})}\right)\right\}$$
$$= P(y=1|\mathbf{x})\ln\left(1+e^{-2f(\mathbf{x})}\right) + P(y=-1|\mathbf{x})\ln\left(1+e^{2f(\mathbf{x})}\right)$$

The claim will follow by evaluating

$$\frac{\partial E\left\{\ln\left(1+e^{-2yf(\mathbf{x})}\right)\right\}}{\partial f(\mathbf{x})} = 0$$

39

Boosting and Logistic Regression (4/5)

- **Note 4**

$\ln\left(1+e^{-2yf(\mathbf{x})}\right)$ and $e^{-yf(\mathbf{x})}$ have identical Taylor expansions around $f=0$ up to second order.

More details:

$$\begin{aligned} \ln\left(1+e^{-2yf(\mathbf{x})}\right) &= \ln 2 - yf(\mathbf{x}) + \frac{y^2}{2}f^2(\mathbf{x}) + \cdots \\ e^{-yf(\mathbf{x})} &= 1 - yf(\mathbf{x}) + \frac{y^2}{2}f^2(\mathbf{x}) + \cdots \end{aligned}$$

Conclusion:

Minimizing $\sum_i e^{-y_if(\mathbf{x}_i)}$, as is done by AdaBoost, can be viewed as a method of approximately minimizing the negative log likelihood. Thus we may expect equation $(*)$ to give a reasonable probability estimate.

40

Boosting and Logistic Regression (5/5)

- **Note 5**

  Indeed it can be shown that log loss is upper bounded by exponential loss

$$\sum_{i=1}^{m} \ln\left(1+e^{-2y_i f(\mathbf{x}_i)}\right) \leq \sum_{i=1}^{m} e^{-y_i f(\mathbf{x}_i)}$$

41

5 BH-Boost (Bhattacharyya Boost)

BHBoost: Introduction

- BHBoost (Lin et al., ECCV-2004) The main idea is to use efficient weak learners by analyzing 1-D projections of training samples.

- Notations

$$\begin{aligned} S &= \{(\mathbf{x}_i, y_i)\}_{i=1}^{m} = \{(\mathbf{x}_i, y_i) \mid y_i = 1\} \bigcup \{(\mathbf{x}_i, y_i) \mid y_i = -1\} \\ &= S^+ \bigcup S^- \quad \text{(two-class labeled training samples)} \end{aligned}$$

$\Phi = \{\phi_j\}_{j=1}^{n}$ (a set of n possible 1-D projections)

Range$(\Phi) = \{b_k\}_{k=1}^{B}$
(range of projected real values is divided into B bins)

42

Naive Weak Learners

- Example

  A classification function (weak learner) can be formed by a *rectangle feature*.

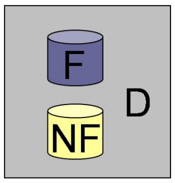
  A classification function h consists of a rectangle feature f, a threshold θ of f, and polarity p, i.e.,

$$h(\mathbf{x}) = \begin{cases} 1 & \text{if } pf(\mathbf{x}) \geq p\theta \\ -1 & \text{otherwise} \end{cases}$$

where $\mathbf{x}$ is a 24×24 image

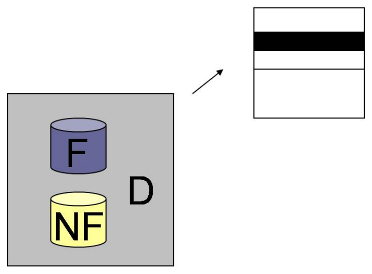
43

What Could Be Wrong?



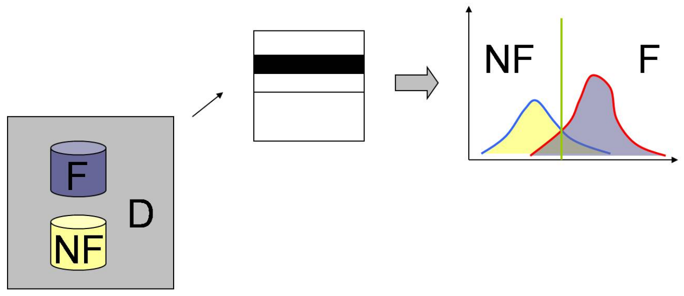
44

What Could Be Wrong?



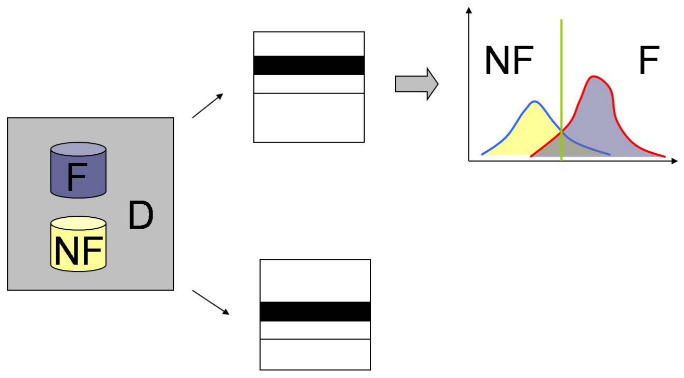
45

What Could Be Wrong?



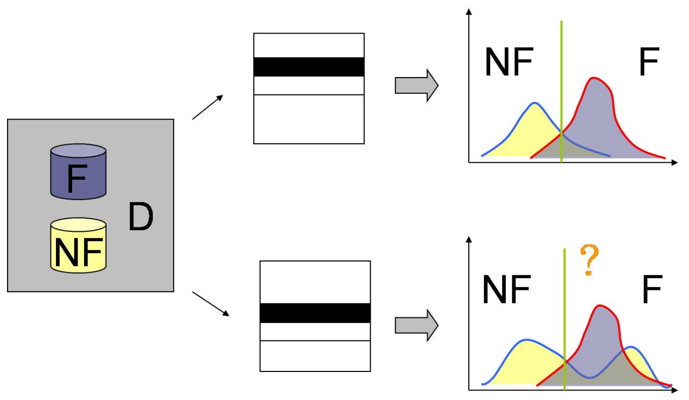
46

What Could Be Wrong?



47

What Could Be Wrong?



48

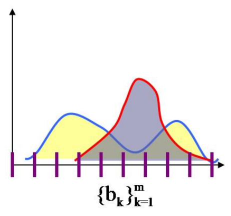
Two Weighted Histograms

- ϕ: rectangle feature/projection direction

$$i_k(\phi) = \{i | \mathbf{x}_i \in S, \phi(\mathbf{x}_i) \in b_k\}$$

$$i_k^+(\phi) = \{i | \mathbf{x}_i \in S^+, \phi(\mathbf{x}_i) \in b_k\}$$

$$i_k^-(\phi) = \{i | \mathbf{x}_i \in S^-, \phi(\mathbf{x}_i) \in b_k\}$$



Positive weighted histogram: $p_k^+(\phi) = \sum_{i_k^+(\phi)} D(i)$

Negative weighted histogram: $p_k^-(\phi) = \sum_{i_k^-(\phi)} D(i)$

49

AdaBoost Error Bound: Revisited

- **Data**

$$\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_m, y_m)\}$$

- **Classifier**

$$H(\mathbf{x}) = \text{sign}(f(\mathbf{x})) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})\right)$$

- **Error bound**

$$\frac{1}{m}\sum_{i=1}^{m}\frac{1}{2}|y_i - H(\mathbf{x}_i)| \le \frac{1}{m}\sum_{i=1}^{m}\exp(-y_i f(\mathbf{x}_i)) = \prod_{t=1}^{T} Z_t$$

where $Z_t = \sum_{i=1}^{m} D_t(i)\exp(-\alpha_t y_i h_t(\mathbf{x}_i))$

50

Projection & Weak Learners

- Each projection ϕ results in a weak learner h_ϕ

  For each ϕ define h_ϕ by $h_\phi(\mathbf{x}) = s_k$ if $\phi(\mathbf{x}) \in b_k$
  It follows that

$$Z = \sum_{i=1}^{m} D(i)\exp(-\alpha y_i h(\mathbf{x}_i)) = \sum_{k=1}^{B}\sum_{i_k(\phi)} D(i)\exp(-\alpha y_i s_k)$$

$$\begin{aligned}
&\frac{dZ}{ds_k} = 0 \\
&\implies \quad \sum_{i_k(\phi)} (-\alpha y_i) D(i) \exp(-\alpha y_i s_k) = 0 \\
&\implies \quad -\sum_{i_k^+(\phi)} \alpha D(i) \exp(-\alpha s_k) + \sum_{i_k^-(\phi)} \alpha D(i) \exp(\alpha s_k) = 0 \\
&\implies \quad \exp(-\alpha s_k) p_k^+(\phi) = \exp(\alpha s_k) p_k^-(\phi) \\
&\implies \quad s_k = \frac{1}{\alpha} \ln \sqrt{\frac{p_k^+(\phi)}{p_k^-(\phi)}}
\end{aligned}$$

51

Bhattacharyya Weak Learner

- How to measure the *goodness* of h_ϕ

$$\begin{aligned}
Z &= \sum_{k=1}^B \sum_{i_k(\phi)} D(i) \exp(-\alpha y_i s_k) \\
&= \sum_{k=1}^B \sum_{i_k(\phi)} D(i) \exp\left(-\alpha y_i \frac{1}{\alpha} \ln \sqrt{p_k^+(\phi)/p_k^-(\phi)}\right) \\
&= \sum_{k=1}^B \left[\sum_{i_k^+(\phi)} D(i) \exp\left(-\ln \sqrt{p_k^+(\phi)/p_k^-(\phi)}\right) \right. \\
&\qquad \left. + \sum_{i_k^-(\phi)} D(i) \exp\left(\ln \sqrt{p_k^+(\phi)/p_k^-(\phi)}\right) \right] \\
&= \sum_{k=1}^B \left[p_k^+(\phi) \sqrt{p_k^-(\phi)/p_k^+(\phi)} \times p_k^-(\phi) \sqrt{p_k^+(\phi)/p_k^-(\phi)} \right] \\
&= 2 \sum_{k=1}^B \sqrt{p_k^+(\phi) p_k^-(\phi)} \quad \text{(Bhattacharyya Coefficient)}
\end{aligned}$$

52

BHBoost Algorithm

- Given: $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)$, $\mathbf{x}_i \in X$ and $y_i \in Y = \{-1, 1\}$

  Initialize the data weight distribution $D_1(i) = 1/m$

  For $t = 1, \ldots, T$

  1. Call BH-WeakLearn using distribution D_t
  2. Find $h_t : X \to \mathbb{R}$ with minimum BC
  3. Update data weight:

     $$D_{t+1}(i) = \frac{D_t(i) \exp(-y_i h_t(\mathbf{x}_i))}{Z_t}$$

     where Z_t is a normalization factor such that D_{t+1} is a distribution

  Output the final BH-classifier:

$$\boxed{H(\mathbf{x}) = \operatorname{sign}(f(\mathbf{x})) = \operatorname{sign}\left(\sum_{t=1}^{T} h_t(\mathbf{x})\right)}$$

53

6 Boosting Algorithm as Gradient Descent

Boosting Algorithm as Gradient Descent: Notations

- Assume that training data $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$ are randomly generated from some unknown probability distribution $\mathscr{D}$ on $X \times Y$.

  We consider voted combination of (weak) classifiers of the following form:

$$H(\mathbf{x}) = \text{sign}(f(\mathbf{x})) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})\right)$$

  where $h_t : X \longrightarrow \{\pm 1\}$.

54

Optimizing Margin-based Cost Functions

- The (functional) margin of an example $(\mathbf{x}, y)$ with respect to the classifier H is defined as $yf(\mathbf{x})$.

  We aim to construct f such that the probability H misclassifies an example is low.

- The idea is to minimize the sample average of some cost function of the margin.

  That is, for a training set S, we want to find f such that cost functional

$$\mathrm{C}(f) = \frac{1}{m}\sum_{i=1}^{m} C(y_i f(\mathbf{x}_i))$$

  is minimized for some suitable cost function $C : \mathbb{R} \longrightarrow \mathbb{R}$.

55

AnyBoost: The Idea (1/2)

- At an abstract level, we can view the binary weak (or base) hypotheses $h \in \mathscr{H}$ and their combinations f as elements of an inner product space $(\mathscr{X}, \langle,\rangle)$.

  S is then a linear space of functions that contains $\text{lin}(\mathscr{H})$, the set of all linear combinations of functions in $\mathscr{H}$. And the inner product is defined by

$$\langle f, g\rangle = \frac{1}{m}\sum_{i=1}^{m} f(\mathbf{x}_i) g(\mathbf{x}_i)$$

  for all $f, g \in \text{lin}(\mathscr{H})$.

- That is, with training set S, a function f can be thought as the following vector in the inner product space:

$$f \Longleftrightarrow \langle f(\mathbf{x}_1), \ldots, f(\mathbf{x}_m)\rangle.$$

56

AnyBoost: The Idea (2/2)

- Suppose function $f \in \text{lin}(\mathscr{H})$ and we wish to find a new $h \in \mathscr{H}$ to add to f such that cost $\mathrm{C}(f + \varepsilon h)$ decreases.

- The idea is to choose h such that $\mathrm{C}(f + \varepsilon h)$ most rapidly decreases.

  However, since the choice of h is limited to the set $\mathscr{H}$, it may not be possible to have $h = -\nabla \mathrm{C}(f)$. Instead we choose h maximizing $-\langle \nabla \mathrm{C}(f), h\rangle$

  From $\mathrm{C}(f) = \frac{1}{m}\sum_{i=1}^{m} y_i f(\mathbf{x}_i)$, we denote $f_i = f(\mathbf{x}_i)$ and write $\mathrm{C}(f)$ as $\mathrm{C}(f_1, f_2, \ldots, f_m)$.

  Thus

$$\begin{aligned} \nabla \mathrm{C}(f) &= \left(\frac{\partial \mathrm{C}}{\partial f_1}, \ldots, \frac{\partial \mathrm{C}}{\partial f_m}\right) = \frac{1}{m}\langle y_1 C', \ldots, y_m C'\rangle \\ -\langle \nabla \mathrm{C}(f), h\rangle &= -\frac{1}{m^2}\sum_{i=1}^{m} y_i h(\mathbf{x}_i) C'(y_i f(\mathbf{x}_i)) \end{aligned}$$

57

AnyBoost: Algorithm

- Input
  - A class of base (weak) classifiers $\mathscr{H} \subseteq \mathscr{X}$.
  - A differentiable cost functional $\mathrm{C} : \mathrm{lin}(\mathscr{H}) \to \mathbb{R}$.
  - Call WeakLearn(f) will return $h \in \mathscr{H}$ with a large value of $-\langle \nabla \mathrm{C}(f), h \rangle$

- Algorithm

```
1. $f_0 = 0$
2. **for** $t := 0$ to $T$ **do**
3.     $h_{t+1} \leftarrow \mathrm{WeakLearn}(f_t = \sum_{\tau=1}^{t} \alpha_\tau h_\tau)$
4.     **if** $-\langle \nabla \mathrm{C}(f_t), h_{t+1} \rangle \leq 0$ **then** return $f \leftarrow f_t$
5.        Choose $\alpha_{t+1}$
6.        $f_{t+1} := f_t + \alpha_{t+1} h_{t+1}$
7. **end for**
8. return $f \leftarrow f_{t+1}$
```

Gradient Descent and Voting Methods (1/2)

- Consider the inner product:

$$-\langle \nabla \mathrm{C}(f), h \rangle = -\frac{1}{m^2} \sum_{i=1}^{m} y_i h(\mathbf{x}_i) C'(y_i f(\mathbf{x}_i))$$

- A reasonable cost function of the margin should be monotonically decreasing, i.e., $-C'(y_i f(\mathbf{x}_i)) > 0$.

  Let

$$D(i) = \frac{C'(y_i f(\mathbf{x}_i))}{-\frac{1}{m^2}\sum_{i=1}^{m} C'(y_i f(\mathbf{x}_i))} \quad \text{for } i = 1, \ldots, m.$$

$$\boxed{\max_h -\langle \nabla \mathrm{C}(f), h \rangle \Longleftrightarrow \min_h \sum_{i:h(\mathbf{x}_i) \neq y_i} D(i)}$$

Gradient Descent and Voting Methods (2/2)

- **Theorem** Let $\mathrm{C} : \mathrm{lin}(\mathscr{H}) \to \mathbb{R}$ be a bounded, Lipschitz differentiable cost functional. (That is, $\exists L > 0$ such that $\|\nabla \mathrm{C}(f) - \nabla \mathrm{C}(f')\| \leq L \|f - f'\|$ for all $f, f' \in \mathrm{lin}(\mathscr{H})$.) Let $f_0, f_1, \ldots,$ be the sequence of combined hypotheses generated by the AnyBoost algorithm, using step-sizes

$$\alpha_{t+1} := -\frac{\langle \nabla \mathrm{C}(f_t), h_{t+1} \rangle}{L \|h_{t+1}\|^2}.$$

  Then AnyBoost algorithm either halts on round T with $-\langle \nabla \mathrm{C}(f_T), h_{T+1} \rangle \leq 0$ or $\mathrm{C}(f_t)$ converges to some finite value C^*, in which case $\lim_{t \to \infty} \langle \nabla \mathrm{C}(f_t), h_{t+1} \rangle = 0$.