

Machine Learning

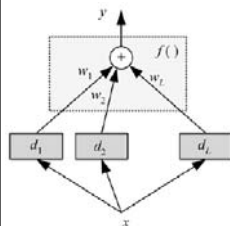
10-701/15-781, Fall 2011

Ensemble methods Boosting from Weak Learners

Eric Xing

Lecture 22, November 30, 2011

Reading: Chap. 14.3 C.B book



NB
LR
SVM
DT
kNN
ANN

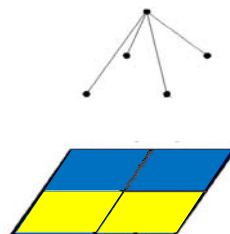
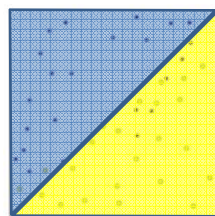


© Eric Xing @ CMU, 2006-2011

1

Weak Learners: Fighting the bias-variance tradeoff

- **Simple (a.k.a. weak) learners** e.g., naïve Bayes, logistic regression, decision stumps (or shallow decision trees)



Are good 😊 - Low variance, don't usually overfit

Are bad ☹️ - High bias, can't solve hard learning problems

- **Can we make weak learners always good???**

• **No!!!**

But often yes...

© Eric Xing @ CMU, 2006-2011

2

Why boost weak learners?

Goal: Automatically categorize type of call requested
(Collect, Calling card, Person-to-person, etc.)

```
• yes I'd like to place a collect call long  
distance please (Collect)  
• operator I need to make a call but I need to  
bill it to my office (ThirdNumber)  
• yes I'd like to place a call on my master card  
please (CallingCard)
```

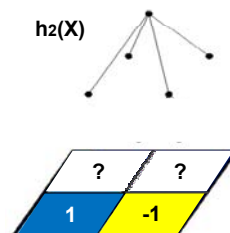
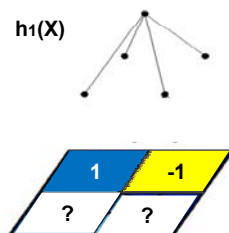
- **Easy to find “rules of thumb” that are “often” correct.**
E.g. If ‘card’ occurs in utterance, then predict ‘calling card’
- **Hard to find single highly accurate prediction rule.**

© Eric Xing @ CMU, 2006-2011

3

Voting (Ensemble Methods)

- 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!



$H: X \rightarrow Y \{-1, 1\}$

$H(X) = h_1(X) + h_2(X)$

$H(X) = \text{sign}(\sum_t \alpha_t h_t(X))$

\downarrow
weights

© Eric Xing @ CMU, 2006-2011

4

Voting (Ensemble Methods)



- 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 weights
 - 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 h_t to learn about different parts of the input space?
 - weigh the votes of different classifiers? α_t

© Eric Xing @ CMU, 2006-2011

5

Rationale: Combination of methods



- There is no algorithm that is always the most accurate
- We can select simple “weak” classification or regression methods and combine them into a single “strong” method
- Different learners use different
 - Algorithms
 - Parameters
 - Representations (Modalities)
 - Training sets
 - Subproblems
- The problem: how to combine them

© Eric Xing @ CMU, 2006-2011

6

Boosting [Schapire'89]

$$D = \left\{ \frac{x_i}{w_i} \mid y_i = 1 \right\}$$

- **Idea:** given a weak learner, run it multiple times on (reweighted) training data, then let learned classifiers vote
- On each iteration t :
 - weight each training example by how incorrectly it was classified w_i
 - Learn a weak hypothesis h_t
 - A strength for this hypothesis α_t
- Final classifier:

$H(X) = \text{sign}(\sum \alpha_t h_t(X))$
- **Practically useful, and theoretically interesting**
- Important issues:
 - what is the criterion that we are optimizing? (measure of loss)
 - we would like to estimate each new component classifier in the same manner (modularity)

$$h^t = \sum_{t'=1}^t \alpha_{t'} h^{t'}$$

© Eric Xing @ CMU, 2006-2011

7

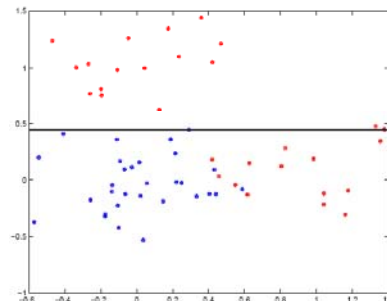
Combination of classifiers

- Suppose we have a family of component classifiers (generating ± 1 labels) such as decision stumps:

$$h(x; \theta) = \text{sign}(wx_k + b)$$

where $\theta = \{k, w, b\}$

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



© Eric Xing @ CMU, 2006-2011

8

Combination of classifiers con'd



- We'd like to combine the simple classifiers additively so that the final classifier is the sign of

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

where the “votes” $\{\alpha_i\}$ emphasize component classifiers that make more reliable predictions than others

- Important issues:
 - what is the criterion that we are optimizing? (measure of loss)
 - we would like to estimate each new component classifier in the same manner (modularity)

AdaBoost



- **Input:**
 - N examples $S_N = \{(x_1, y_1), \dots, (x_N, y_N)\}$
 - a weak base learner $h = h(x, \theta)$
- **Initialize:** equal example weights $w_i = 1/N$ for all $i = 1..N$
- **Iterate for $t = 1..T$:**
 1. train base learner according to weighted example set (w_t, x) and obtain hypothesis $h_t = h(x, \theta_t)$
 2. compute hypothesis error ϵ_t
 3. compute hypothesis weight α_t
 4. update example weights for next iteration w_{t+1}
- **Output:** final hypothesis as a linear combination of h_t

AdaBoost

$$\sum_{i=1}^n I(y_i \neq h(\mathbf{x}_i))$$



- At the k th iteration we find (any) classifier $h(\mathbf{x}; \theta_k^*)$ for which the weighted classification error:

$$\varepsilon_k = \sum_{i=1}^n W_i^{k-1} I(y_i \neq h(\mathbf{x}_i; \theta_k^*)) / \sum_{i=1}^n W_i^{k-1}$$

is better than change.

- This is meant to be "easy" --- weak classifier
- Determine how many "votes" to assign to the new component classifier:

$$\alpha_k = 0.5 \log((1 - \varepsilon_k) / \varepsilon_k)$$

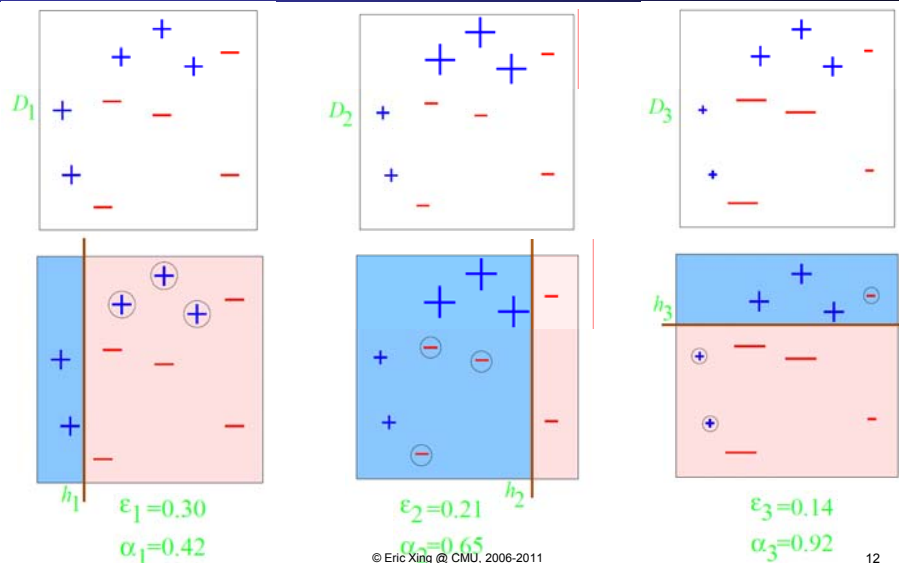
- stronger classifier gets more votes
- Update the weights on the training examples:

$$W_i^k = W_i^{k-1} \exp\{-y_i \alpha_k h(\mathbf{x}_i; \theta_k^*)\}$$

© Eric Xing @ CMU, 2006-2011

11

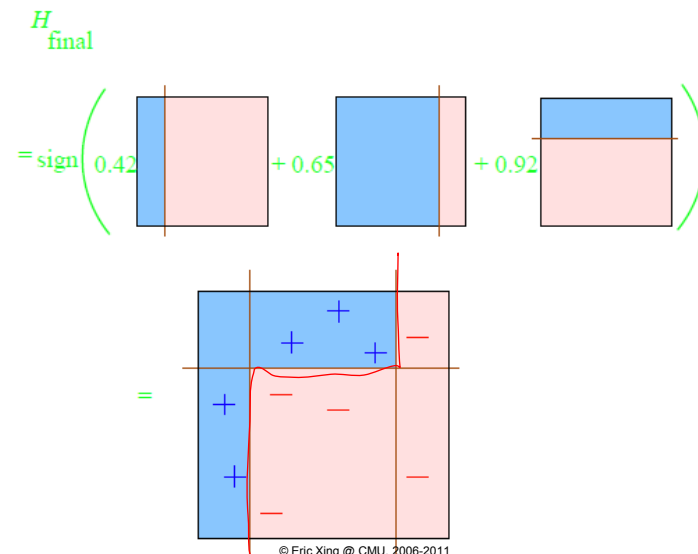
Boosting Example (Decision Stumps)



© Eric Xing @ CMU, 2006-2011

12

Boosting Example (Decision Stumps)



© Eric Xing @ CMU, 2006-2011

13

- What is the criterion that we are optimizing?
(measure of loss)



© Eric Xing @ CMU, 2006-2011

14

Measurement of error

- Loss function:

$$\lambda(y, h(\mathbf{x}))$$

$$(e.g. I(y \neq h(\mathbf{x})))$$

- Generalization error:

$$L(h) = E[\lambda(y, h(\mathbf{x}))]$$

- Objective: find h with minimum *generalization* error

- Main boosting idea: minimize the *empirical* error:

$$\hat{L}(h) = \frac{1}{N} \sum_{i=1}^N \lambda(y_i, h(\mathbf{x}_i))$$

Exponential Loss

- Empirical loss:

$$\hat{L}(h) = \frac{1}{N} \sum_{i=1}^N \lambda(y_i, \hat{h}_m(\mathbf{x}_i))$$

- Another possible measure of empirical loss is

$$\hat{L}(h) = \sum_{i=1}^n \exp\{-y_i \hat{h}_m(\mathbf{x}_i)\}$$

Exponential Loss

- One possible measure of empirical loss is

$$\hat{L}(h) = \sum_{i=1}^n \exp\{-y_i \hat{h}_m(\mathbf{x}_i)\}$$

Recall that:

$$\hat{h}_m(\mathbf{x}) = \alpha_1 h(\mathbf{x}; \theta_1) + \dots + \alpha_m h(\mathbf{x}; \theta_m)$$

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

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

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

$$W_i^{m-1} = \exp\{-y_i \hat{h}_{m-1}(\mathbf{x}_i)\}$$

- The combined classifier based on $m-1$ iterations defines a weighted loss criterion for the next simple classifier to add
- each training sample is weighted by its "classifiability" (or difficulty) seen by the classifier we have built so far

Handwritten notes:

$$\hat{h}_m = \hat{h}_{m-1} + \alpha_m h(\mathbf{x}; \theta_m)$$

$$\alpha_m^* = \arg\min_{\alpha} \hat{L}$$

© Eric Xing @ CMU, 2006-2011

17

Linearization of loss function

- We can simplify a bit the estimation criterion for the new component classifiers (assuming α is small)

$$\exp\{-y_i \alpha_m h(\mathbf{x}_i; \theta_m)\} \approx 1 - y_i \alpha_m h(\mathbf{x}_i; \theta_m)$$

- Now our empirical loss criterion reduces to

$$\sum_{i=1}^n \exp\{-y_i \hat{h}_m(\mathbf{x}_i)\}$$

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

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

$$W_i^{m-1} = \exp\{-y_i \hat{h}_{m-1}(\mathbf{x}_i)\}$$

Handwritten notes:

$$M-1 \rightarrow \mathcal{M}_i: \mathbf{x}_i, y_i, W_{m-1,i}$$

$$\downarrow$$

$$\alpha_m, h(\mathbf{x}_i; \theta_m), W_m$$

- We could choose a new component classifier to optimize this weighted agreement

© Eric Xing @ CMU, 2006-2011

18

A possible algorithm

- At stage m we find θ^* that maximize (or at least give a sufficiently high) weighted agreement:

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

- each sample is weighted by its "difficulty" under the previously combined $m-1$ classifiers,
- more "difficult" samples received heavier attention as they dominates the total loss

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

- Then we go back and find the "votes" α_m^* associated with the new classifier by minimizing the **original** weighted (exponential) loss

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

\Rightarrow

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

© Eric Xing @ CMU, 2006-2011

19

The AdaBoost algorithm

$$W_i^{m-1} = \exp\{-y_i \hat{h}_{m-1}(\mathbf{x}_i)\}$$

- At the k th iteration we find (any) classifier $h(\mathbf{x}; \theta_k^*)$ for which the weighted classification error:

$$\epsilon_k = \sum_{i=1}^n W_i^{k-1} I(y_i \neq h(\mathbf{x}_i; \theta_k^*)) / \sum_{i=1}^n W_i^{k-1}$$

weak learner
 $h_m(\mathbf{x}, \theta_m)$
 classifier
 $h_m(\quad)$

is better than change.

- This is meant to be "easy" --- weak classifier
- Determine how many "votes" to assign to the new component classifier:

$$\alpha_k = 0.5 \log((1 - \epsilon_k) / \epsilon_k)$$

- stronger classifier gets more votes
- Update the weights on the training examples:

$$W_i^k = W_i^{k-1} \exp\{-y_i \alpha_k h(\mathbf{x}_i; \theta_k)\}$$

© Eric Xing @ CMU, 2006-2011

20

The AdaBoost algorithm cont'd



- The final classifier after m boosting iterations is given by the sign of

$$\hat{h}(\mathbf{x}) = \frac{\alpha_1 h(\mathbf{x}; \theta_1) + \dots + \alpha_m h(\mathbf{x}; \theta_m)}{\alpha_1 + \dots + \alpha_m}$$

- the votes here are normalized for convenience

Boosting



- We have basically derived a Boosting algorithm that sequentially adds **new component classifiers**, each trained on reweighted training examples
 - each component classifier is presented with a slightly different problem
- AdaBoost preliminaries:
 - we work with *normalized weights* W_i on the training examples, initially uniform ($W_i = 1/n$)
 - the weight reflect the "*degree of difficulty*" of each datum on the latest classifier

AdaBoost: summary



- **Input:**
 - N examples $S_N = \{(x_1, y_1), \dots, (x_N, y_N)\}$
 - a weak base learner $h = h(x, \theta)$
- **Initialize:** equal example weights $w_i = 1/N$ for all $i = 1..N$
- **Iterate for $t = 1..T$:**
 1. train base learner according to weighted example set (w_t, x) and obtain hypothesis $h_t = h(x, \theta_t)$
 2. compute hypothesis error ε_t
 3. compute hypothesis weight α_t
 4. update example weights for next iteration w_{t+1}
- **Output:** final hypothesis as a linear combination of h_t

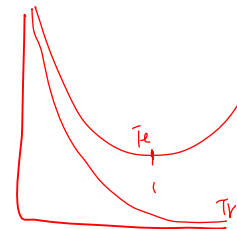
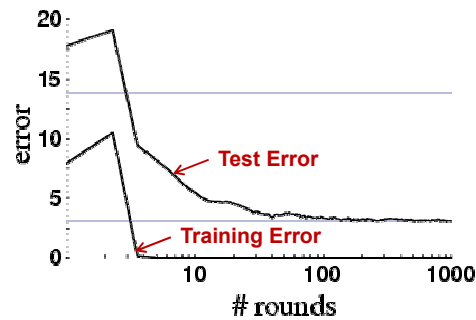
Base Learners



- Weak learners used in practice:
 - Decision stumps (axis parallel splits)
 - Decision trees (e.g. C4.5 by Quinlan 1996)
 - Multi-layer neural networks
 - Radial basis function networks
- Can base learners operate on weighted examples?
 - In many cases they can be modified to accept weights along with the examples
 - In general, we can sample the examples (with replacement) according to the distribution defined by the weights

Boosting results – Digit recognition

[Schapire, 1989]



- Boosting often, **but not always**
 - Robust to overfitting
 - Test set error decreases even after training error is zero

© Eric Xing @ CMU, 2006-2011

25

Generalization Error Bounds

[Freund & Schapire '95]



$$error_{true}(H) \leq error_{train}(H) + \tilde{O} \left(\sqrt{\frac{Td}{m}} \right)$$

	bias	variance	
tradeoff	large	small	T small
	small	large	T large

- T – number of boosting rounds
- d – VC dimension of weak learner, measures complexity of classifier
- m – number of training examples

© Eric Xing @ CMU, 2006-2011

26

Generalization Error Bounds

[Freund & Schapire '95]



$$error_{true}(H) \leq error_{train}(H) + \tilde{O}\left(\sqrt{\frac{Td}{m}}\right) \quad \text{With high probability}$$

Boosting can overfit if T is large

Boosting often,

Contradicts experimental results

- Robust to overfitting
- Test set error decreases even after training error is zero

Need better analysis tools – margin based bounds

~~$\sum_i \alpha_i h_i(x_i)$~~ $\sum_i \alpha_i \exp\{-y_i \alpha_i h_i(x_i)\}$

© Eric Xing @ CMU, 2006-2011

27

Why it is working?



- You will need some learning theory (to be covered in the next two lectures) to understand this fully, but for now let's just go over some high level ideas
- Generalization Error:

With high probability, Generalization error is less than:

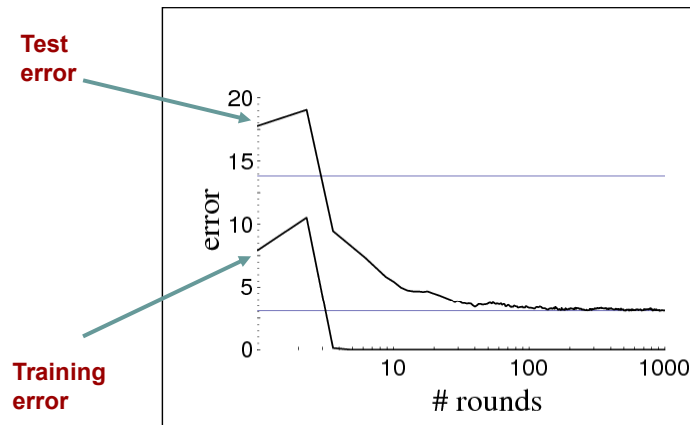
$$\hat{\Pr}[H(x) \neq y] + \tilde{O}\left(\sqrt{\frac{Td}{m}}\right)$$

As T goes up, our bound becomes worse,
Boosting should overfit!

© Eric Xing @ CMU, 2006-2011

28

Experiments



The Boosting Approach to Machine Learning, by Robert E. Schapire

© Eric Xing @ CMU, 2006-2011

29

Training Margins

- When a vote is taken, the more predictors agreeing, the more confident you are in your prediction.

- Margin for example:

$$\text{margin}_h(\mathbf{x}_i, y_i) = y_i \left[\frac{\alpha_1 h(\mathbf{x}_i; \theta_1) + \dots + \alpha_m h(\mathbf{x}_i; \theta_m)}{\alpha_1 + \dots + \alpha_m} \right]$$

The margin lies in $[-1, 1]$ and is negative for all misclassified examples.

- Successive boosting iterations improve the majority vote or margin for the training examples

© Eric Xing @ CMU, 2006-2011

30

A Margin Bound



- For any γ , the generalization error is less than:

$$\Pr(\text{margin}_h(\mathbf{x}, y) \leq \gamma) + O\left(\sqrt{\frac{6}{m\gamma^2}}\right)$$

Robert E. Schapire, Yoav Freund, Peter Bartlett and Wee Sun Lee.
Boosting the margin: A new explanation for the effectiveness of voting
methods. *The Annals of Statistics*, 26(5):1651-1686, 1998.

- It does not depend on T !!!

Boosting and Logistic Regression



Logistic regression assumes:

$$P(Y = 1|X) = \frac{1}{1 + \exp(f(x))} \quad f(x) = w_0 + \sum_j w_j x_j$$

And tries to maximize data likelihood:

$$P(\mathcal{D}|f) = \prod_{i=1}^m \frac{1}{1 + \exp(-y_i f(x_i))}$$

Equivalent to minimizing log loss

$$-\log P(\mathcal{D}|f) = \sum_{i=1}^m \ln(1 + \exp(-y_i f(x_i)))$$

Boosting and Logistic Regression



Logistic regression equivalent to minimizing log loss

$$\sum_{i=1}^m \ln(1 + \exp(-y_i f(x_i)))$$

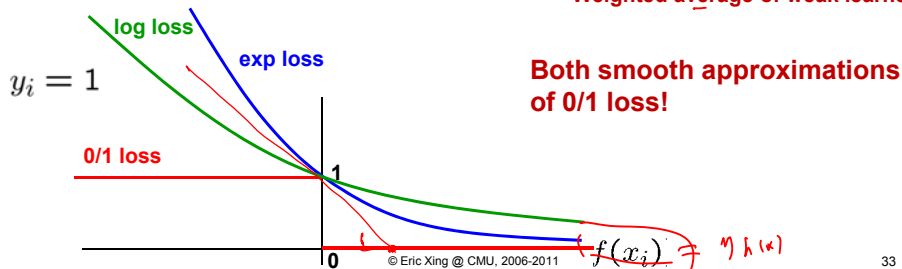
$$f(x) = w_0 + \sum_j w_j x_j$$

Boosting minimizes similar loss function!!

$$\frac{1}{m} \sum_{i=1}^m \exp(-y_i f(x_i)) = \prod_t Z_t$$

$$f(x) = \sum_t \alpha_t h_t(x)$$

Weighted average of weak learners



33

Boosting and Logistic Regression



Logistic regression:

- Minimize log loss

$$\sum_{i=1}^m \ln(1 + \exp(-y_i f(x_i)))$$

- Define

$$f(x) = \sum_j w_j x_j$$

where x_j predefined features

(linear classifier)

- Jointly optimize over all weights $w_0, w_1, w_2 \dots$

Boosting:

- Minimize exp loss

$$\sum_{i=1}^m \exp(-y_i f(x_i))$$

- Define

$$f(x) = \sum_t \alpha_t h_t(x)$$

where $h_t(x)$ defined dynamically to fit data

(not a linear classifier)

- Weights α_t learned per iteration t incrementally

© Eric Xing @ CMU, 2006-2011

34

Hard & Soft Decision



Weighted average of weak learners

$$f(x) = \sum_t \alpha_t h_t(x)$$

Hard Decision/Predicted label:

$$H(x) = \text{sign}(f(x))$$

Soft Decision:
(based on analogy with logistic regression)

$$P(Y = 1|X) = \frac{1}{1 + \exp(f(x))}$$

© Eric Xing @ CMU, 2006-2011

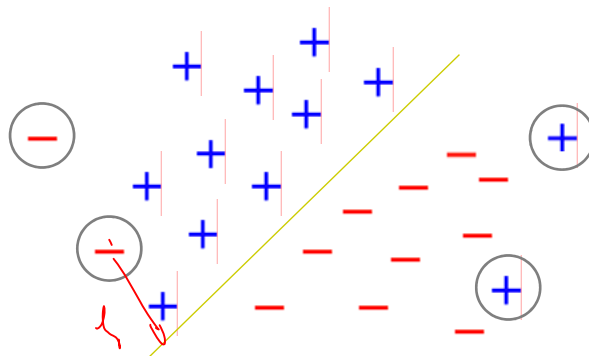
35

Effect of Outliers



Good ☺ : Can identify outliers since focuses on examples that are hard to categorize

Bad ☹ : Too many outliers can degrade classification performance dramatically increase time to convergence



© Eric Xing @ CMU, 2006-2011

36

Summary



- Boosting takes a weak learner and converts it to a strong one
- Works by asymptotically minimizing the empirical error
- Effectively maximizes the margin of the combined hypothesis

Learning from weighted data




- Consider a weighted dataset
 - $D(i)$ – weight of i th training example (\mathbf{x}^i, y^i)
 - Interpretations:
 - i th training example counts as $D(i)$ examples
 - If I were to “resample” data, I would get more samples of “heavier” data points
- Now, in all calculations, whenever used, i th training example counts as $D(i)$ “examples”
 - e.g., in MLE redefine $\text{Count}(Y=y)$ to be weighted count

Unweighted data


$$\text{Count}(Y=y) = \sum_{i=1}^m \mathbf{1}(Y^i=y)$$

Weights $D(i)$

$$\text{Count}(Y=y) = \sum_{i=1}^m D(i) \mathbf{1}(Y^i=y)$$



Method	Task	Input	Output	Loss / obj	hypothesis	opt algorithm	Gen/Dist
DT	classification	$X \in \mathbb{R}^m$ $X \in \{1, \dots, m\}^m$ $X \in \mathbb{R}^m \times \{1, \dots, k\}^{m'}$		$y \in \{1\}$ $\{1, \dots, k\}$	0/1? ϕ	greedy	D
NN	C	?	?	0/1? ϕ Bicubic	?	?	?



--	--	--	--	--	--	--	--