Machine Learning: Yoshinari Fujinuma

University of Colorado Boulder

LECTURE 21

Slides adapted from Chenhao Tan, Jordan Boyd-Graber, Chris Ketelsen

Logistics

- Please check whether your final project team is in the shared spreadsheet
- Homework 4 is due on Wednesday

Learning objects

- Learn about Adaboost
- Understand the math behind boosting

Overview

Recap of Boosting Intuition

Adaboost

(Bonus) The underlying math

Outline

Boosting intuition

Boosting is an ensemble method, but with a different twist.
Idea:

- Build a sequence of dumb models
- Modify training data along the way to focus on difficult to classify examples
- Predict based on weighted majority vote of all the models

Challenges:

- What do we mean by dumb?
- How do we promote difficult examples?
- Which models get more say in vote?

Outline

Adaboost

- Training set $\{(\mathbf{x}_i, y_i)\}_{i=1}^m$: m training example
- Feature vector $\mathbf{x} \in \mathbb{R}^d$: d-dimensional feature vector
- Labels are $y_i \in \{-1, 1\}$
- α_k: Weights associcated to the kth weak classifier
- w_i: Weights associcated to the ith training example

Adaboost

1 . Initialize data weights to $w_i = \frac{1}{m}$, $i = 1, \ldots, m$

2 . For $k = 1$ to K:

(a) Fit classifier $h_k(\mathbf{x})$ to training data with weights w_i

(b) Compute weighted error $\text{err}_k = \frac{\sum_{i=1}^{m} w_i I(y_i \neq h_k(\mathbf{x}_i))}{\sum_{i=1}^{m} w_i}$

(c) Compute $\alpha_k = \frac{1}{2} \log((1 - \text{err}_k)/\text{err}_k)$

(d) Set $w_i \leftarrow \frac{w_i}{Z_k} \cdot \exp[-\alpha_k y_i h_k(\mathbf{x}_i)]$, $i = 1, \ldots, m$

3 . Output $H(\mathbf{x}) = \text{sign}\left[\sum_{k=1}^{K} \alpha_k h_k(\mathbf{x})\right]$

Adaboost

1 . Initialize data weights to $w_i = \frac{1}{m}$, $i = 1, \ldots, m$
Weights are initialized to equal weights. Every training example counts equally on first iteration.

Adaboost

2a. Fit classifier $h_k(\mathbf{x})$ to training data D where each examples are weighted by w_i
Decide split based on information gain IG with weighted entropy $H(D)$ with respect to feature f:

$$H(D) = -p \log_2 p - (1-p) \log_2 (1-p)$$

$$IG(D,f) = H(D) - \frac{|D_{f,\text{left}}|}{|D|} H(D_{\text{left}}) - \frac{|D_{f,\text{right}}|}{|D|} H(D_{\text{right}})$$

Note: p is the fraction of examples in positive class i.e., $p = \dfrac{\sum_{i=1}^{m} w_i \cdot I(y_i = 1)}{m}$

Adaboost

2b. Compute weighted error

$$\text{err}_k = \frac{\sum_{i=1}^{m} w_i I(y_i \neq h_k(\mathbf{x}_i))}{\sum_{i=1}^{m} w_i}$$

Adaboost

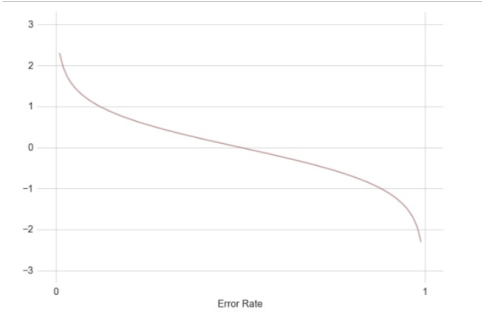
2b. Compute weighted error

$$\text{err}_k = \frac{\sum_{i=1}^{m} w_i I(y_i \neq h_k(\mathbf{x}_i))}{\sum_{i=1}^{m} w_i}$$

Still gives error rate in $[0,\ 1]$

Adaboost

2c. Compute $\alpha_k = \frac{1}{2}\log((1-\text{err}_k)/\text{err}_k)$
Models with small err get promoted (exponentially)
Models with large err get demoted (exponentially)



Adaboost

2d. Set $w_i \leftarrow \frac{w_i}{Z_k} \cdot \exp[-\alpha_k y_i h_k(\mathbf{x}_i)]$, $i = 1, \ldots, m$

- If example was misclassified weight w_i goes up
- If example was classified correctly weight w_i goes down

How big of a jump depends on accuracy of model
Z_k is just a normalizing constant to ensure that w_i is probability

Adaboost

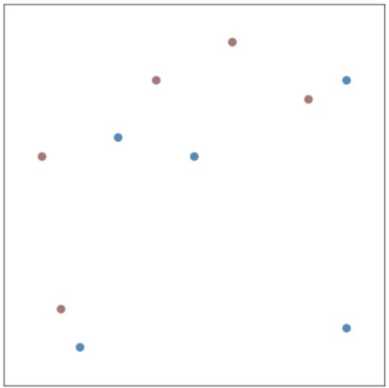
3. Output $H(\mathbf{x}) = \text{sign}\left[\sum_{k=1}^{K} \alpha_k h_k(\mathbf{x})\right]$

Sum up weighted votes from each model

Classify $y = 1$ if positive and $y = -1$ if negative

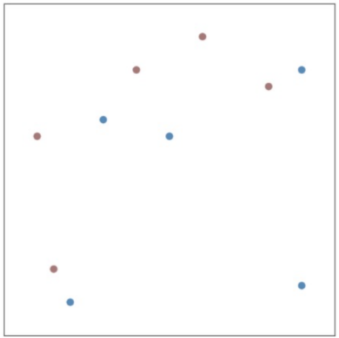
Adaboost example

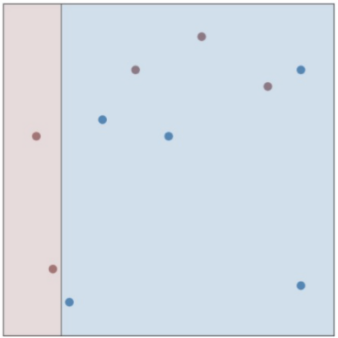
Suppose we have the following training data and train Adaboost with $K = 3$ weak learners



Adaboost example

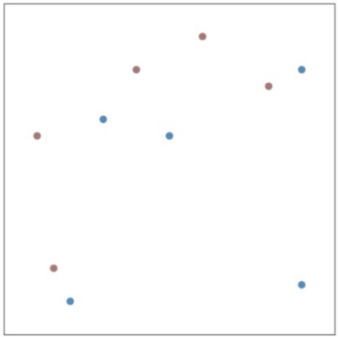
First decision stump

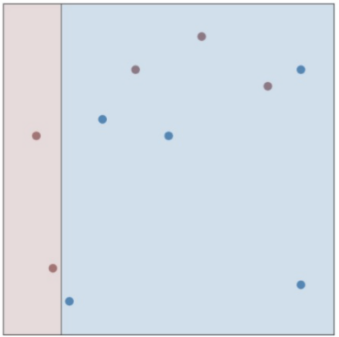




Adaboost example

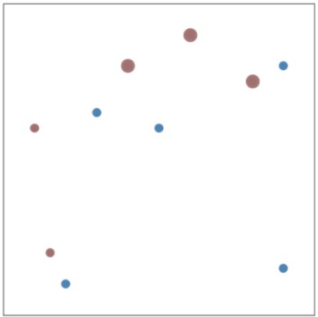
First decision stump , $err_1 = 0.3, \alpha_1 = 0.42$ (recall $\alpha_k = \frac{1}{2}\log((1 - \text{err}_k)/\text{err}_k)$





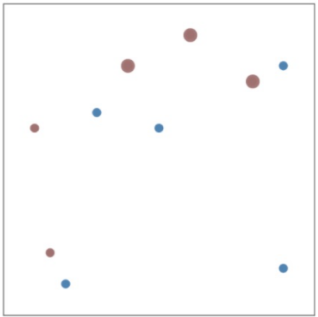
Adaboost example

Second decision stump



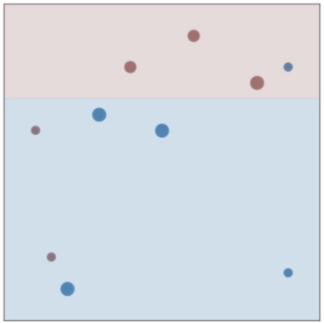
Adaboost example

Second decision stump , $err_2 = 0.21, \alpha_2 = 0.65$



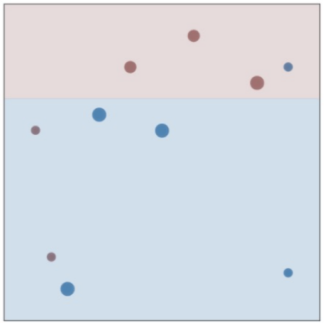
Adaboost example

Third decision stump

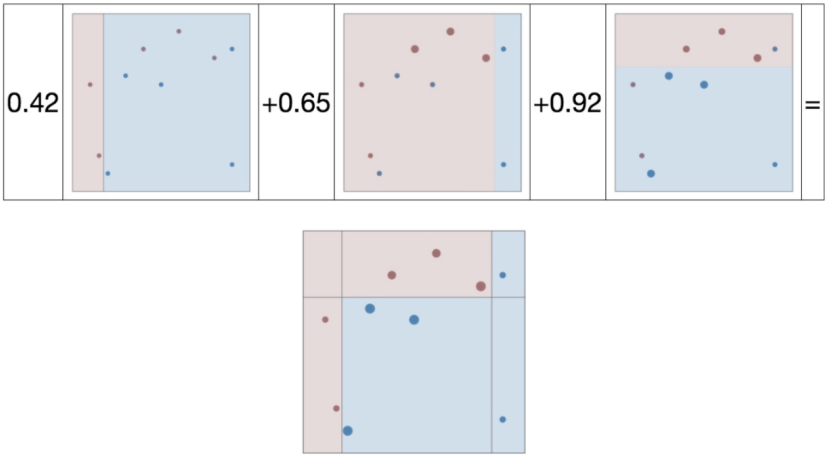


Adaboost example

Third decision stump , $err_3 = 0.14, \alpha_3 = 0.92$

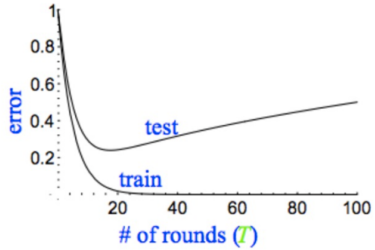


Adaboost example



Generalization performance

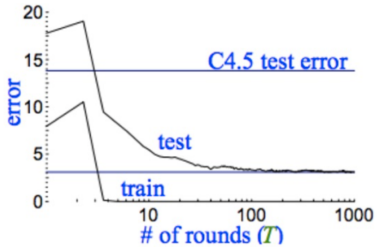
Recall the standard experiment of measuring test and training error vs. model complexity



Once overfitting begins, test error goes up

Generalization performance

Boosting has remarkably uncommon effect



Happens much slower with boosting

Outline

The underlying math

So far this looks like a reasonable thing that just worked out
But is there math behind it?

The underlying math

Formulate the problem as finding a classifier $H(\mathbf{x})$ such that

$$H^* = \arg\min_H \sum_{i=1}^{m} L(y_i, H(\mathbf{x}_i))$$

where here we take the loss function L to be the following exponential function

$$L(y, H(\mathbf{x})) = \exp[-y\, H(\mathbf{x})]$$

Notice if $y \neq H(\mathbf{x})$ we get a positive exponent and a large loss.

The underlying math

Since we're doing this in an iterative way, we're going to assume a form of $H(\mathbf{x})$ that is amenable to iterative improvement. Specifically

$$H(\mathbf{x}) = \sum_{k=1}^{K} \alpha_k h_k(\mathbf{x})$$

So the problem becomes to choose the optimal weights, α, and optimal functions h_k.
For everything I will assume that we've already computed a good H_{k-1} and attempt to build a better H_k.

The underlying math

At step k we have loss L_k

$$L_k = \sum_{i=1}^{m} \exp\left[-y_i \left(H_{k-1}(\mathbf{x}_i) + \alpha h_k(\mathbf{x_i})\right)\right]$$

Taking the things we know out of the exponent gives

$$L_k = \sum_{i=1}^{m} w_{i,k} \exp\left[-\alpha y_i h_k(\mathbf{x_i})\right], \quad w_{i,k} = \exp\left[-y_i \, H_{k-1}(\mathbf{x}_i)\right]$$

Want to choose good α and h to reduce loss

The underlying math

Can rewrite as

$$L_k = e^{\alpha} \sum_{y_i \neq h(\mathbf{x}_i)} w_{i,k} + e^{-\alpha} \sum_{y_i = h_k(\mathbf{x}_i)} w_{i,k}$$

Add zero in a fancy way

$$L_k = (e^{\alpha} - e^{-\alpha}) \sum_{i=1}^{m} w_{i,k} I(y_i \neq h_k(\mathbf{x}_i)) + e^{-\alpha} \sum_{i=1}^{m} w_{i,k}$$

The underlying math

Choose h and α separately.
A good h would be one that minimizes weighted misclassifications

$$h_k = \arg\min_h w_{i,k} I(y_i \neq h(\mathbf{x}_i))$$

That's what our weak learner h_k is for

The underlying math

Plugging that into our Loss function gives

$$L_k = (e^{\alpha} - e^{-\alpha}) \sum_{i=1}^{m} w_{i,k} I(y_i \neq h_k(\mathbf{x}_i)) + e^{-\alpha} \sum_{i=1}^{m} w_{i,k}$$

Now we want to minimize w.r.t. α. Take derivative, set equal to zero

$$0 = \frac{dL_k}{d\alpha} = (e^{\alpha} + e^{-\alpha}) \sum_{i=1}^{m} w_{i,k} I(y_i \neq h_k(\mathbf{x}_i)) - e^{-\alpha} \sum_{i=1}^{m} w_{i,k}$$

which gives $e^{2\alpha} = \frac{\sum_i w_{i,k}}{\sum_i w_{i,k} I(y_i \neq h_k(\mathbf{x}))} - 1 = \frac{1}{\text{err}_k} - 1 = \frac{1-\text{err}_k}{\text{err}_k}$

And finally $\alpha_k = \frac{1}{2} \log\left(\frac{1-\text{err}_k}{\text{err}_k}\right)$

The underlying math

What about the weight update?
Remember we got the weights by pulling the already computed function out of L.
For the new weights, we have

$$w_{i,k+1} = \exp[-y_i H_k(\mathbf{x}_i)] = \exp[-y_i H_{k-1}(\mathbf{x}_i) - \alpha_k y_i h_k(\mathbf{x}_i)] = w_{i,k} \exp[-\alpha_k y_i h_k(\mathbf{x}_i)]$$

which gives the update

$$w_i \leftarrow w_i \exp[-\alpha_k y_i h_k(\mathbf{x}_i)]$$

And finally, $H_K(\mathbf{x}) = \text{sign}\left[\sum_{k=1}^{K} \alpha_k h_k(\mathbf{x})\right]$
This is exactly Adaboost