



# Boosting, Surrogate Losses, and Ensemble Methods

GU 4241/GR 5241

Statistical Machine Learning

Xiaofei Shi



# Types of classifiers

- Discriminative classifiers:
  - Directly estimate a decision rule/boundary
  - e.g. decision tree, SVM
- Instance based classifiers:
  - Use observation directly
  - e.g. K nearest neighborhood
- Generative classifiers:
  - Build a generative statistical model
  - e.g. Bayesian Network



# Pros and cons for simple classifiers

- Typical simple/weak classifiers
  - Shallow decision tree, SVM, logistic regression, naive Bayes
- Don't usually overfit
- Cannot solve complicated learning tasks

Can we make simple learners smarter?



# What about several simple model together?

## Buckets of models

- Input:
  - A dataset  $D$
  - your top  $T$  favorite learners:  $L_1, \dots, L_T$
- Learning algorithm:
  - Estimate the error of learners:  $L_1, \dots, L_T$
  - Pick the best learner  $L^*$
  - Train  $L^*$  on  $D$  and return results





# Pros and cons of a “bucket of models”

- Pros:
  - simple
  - not much worse than the best of the “base learners”
- Cons:
  - what if there's not a single best learner?



# Stack learners: first attempt

- Input:
  - A dataset  $D$
  - your top  $T$  favorite learners:  $L_1, \dots, L_T$
- Learning algorithm:
  - Train  $L_1, \dots, L_T$  on dataset to  $h_1, \dots, h_T$
  - Create a new dataset  $D'$  containing  $(x', y'), \dots$ 
    - $x'$  is a vector of the  $T$  predictions  $h_1(x), \dots, h_T(x)$
    - $y$  is the label for  $x$
  - Train new classifier on  $D'$  to get  $h'$  -- **which combines the predictions!**





# Pro and cons of stacking

- Pros:
  - Fairly simple
  - Slow, but easy to parallelize
- Cons:
  - What if there's not a single best combination scheme?
  - E.g.: for movie recommendation sometimes L1 is best for users with many ratings and L2 is best for users with few ratings



# 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!

**But how do you ???**

- force classifiers to learn about different parts of the input space?
- weigh the votes of different classifiers?





# Boosting (Schapire, 1989)

- **Practically useful**
- **Theoretically interesting**

- Idea: given a weak learner, run it multiple times on (reweighted) training data, then let the learned classifiers vote
- On each iteration  $t$ :
  - weight each training example by how incorrectly it was classified
  - Learn a hypothesis –  $h_t$
  - A strength for this hypothesis –  $s_t$
- Final classifier:
  - A linear combination of the votes of the different classifiers weighted by their strength.





# Learning from weighted data

**Sometimes not all data points are equal**

- Some data points are more equal than others

**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., MLE for Naïve Bayes, redefine  $Count(Y=y)$  to be weighted count





# Ada Boost

Given:  $(x_1, y_1), \dots, (x_m, y_m)$  where  $x_i \in X, y_i \in Y = \{-1, +1\}$

Initialize  $D_1(i) = 1/m$ .

For  $t = 1, \dots, T$ :

- Train weak learner using distribution  $D_t$ .
- Get weak classifier  $h_t : X \rightarrow \mathbb{R}$ .
- Choose  $\alpha_t \in \mathbb{R}$ .
- Update:

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

where  $Z_t$  is a normalization factor

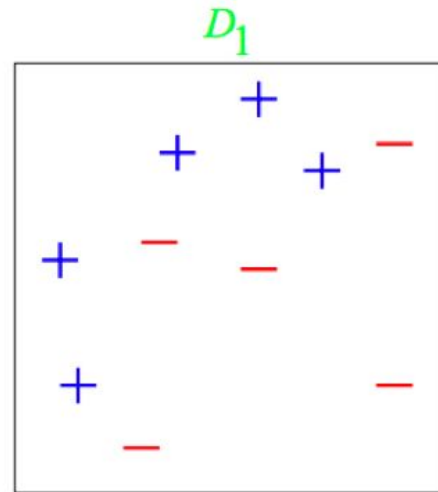
$$Z_t = \sum_{i=1}^m D_t(i) \exp(-\alpha_t y_i h_t(x_i))$$

Output the final classifier:

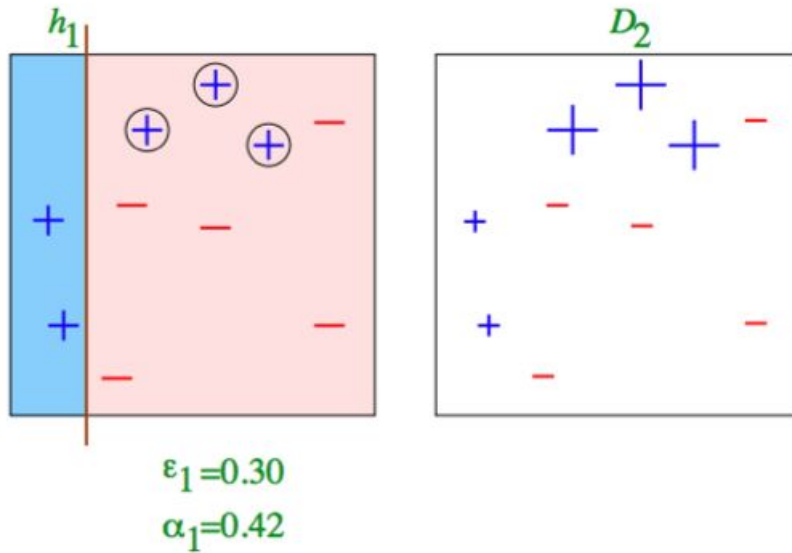
$$H(x) = \text{sign} \left( \sum_{t=1}^T \alpha_t h_t(x) \right).$$



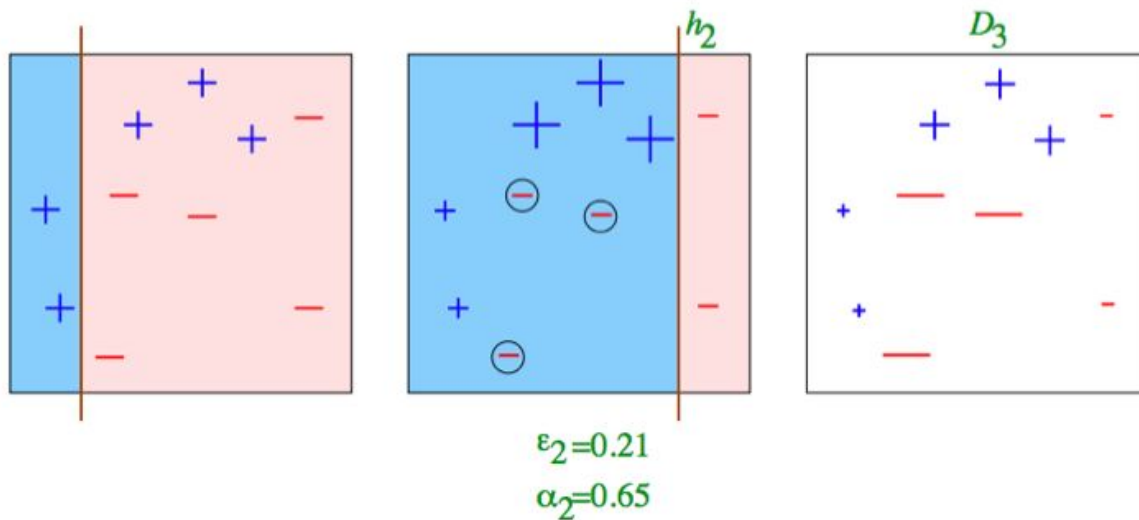
# Toy example



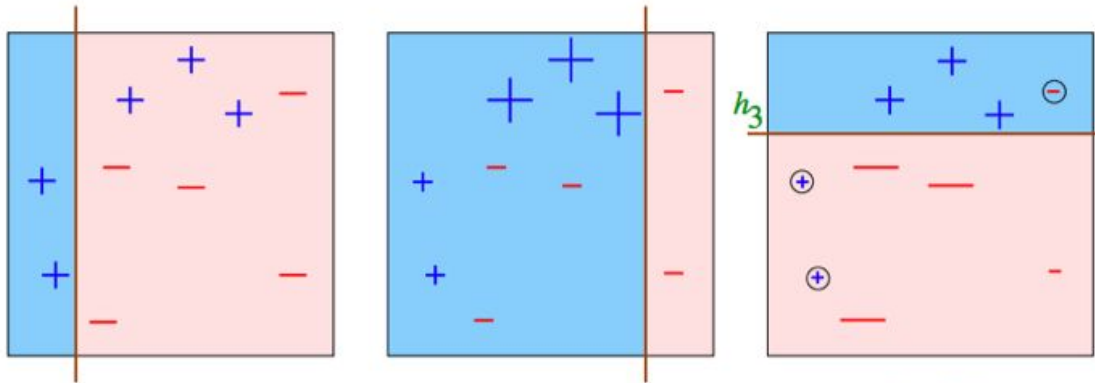
## Toy example: Round 1



## Toy example: Round 2



## Toy example: Round 3



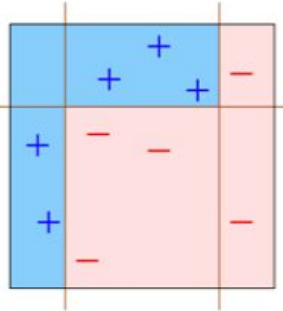
$$\epsilon_3 = 0.14$$

$$\alpha_3 = 0.92$$

## Toy example: Final classifier

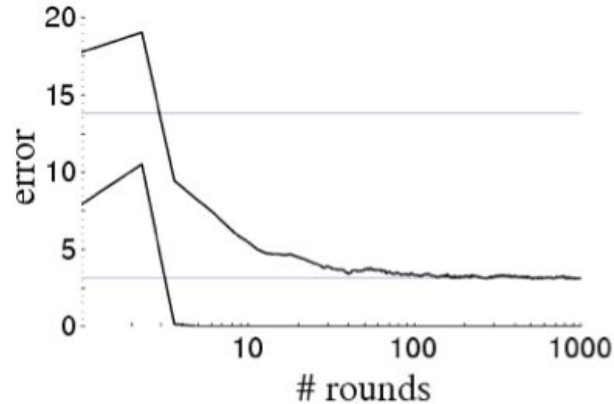
$$H_{\text{final}} = \text{sign} \left( 0.42 \begin{array}{|c|c|} \hline \text{blue} & \text{red} \\ \hline \end{array} + 0.65 \begin{array}{|c|c|} \hline \text{blue} & \text{red} \\ \hline \end{array} + 0.92 \begin{array}{|c|c|} \hline \text{blue} & \text{red} \\ \hline \end{array} \right)$$

=





# Boosting for handwritten digit recognition

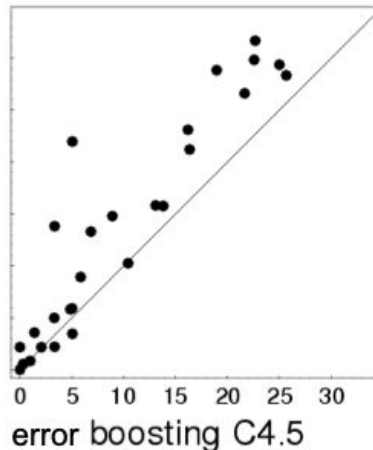
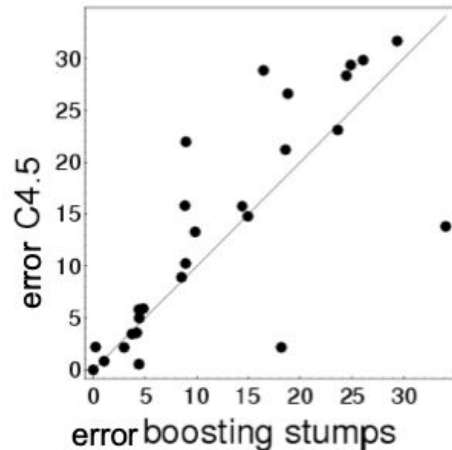


## Boosting often

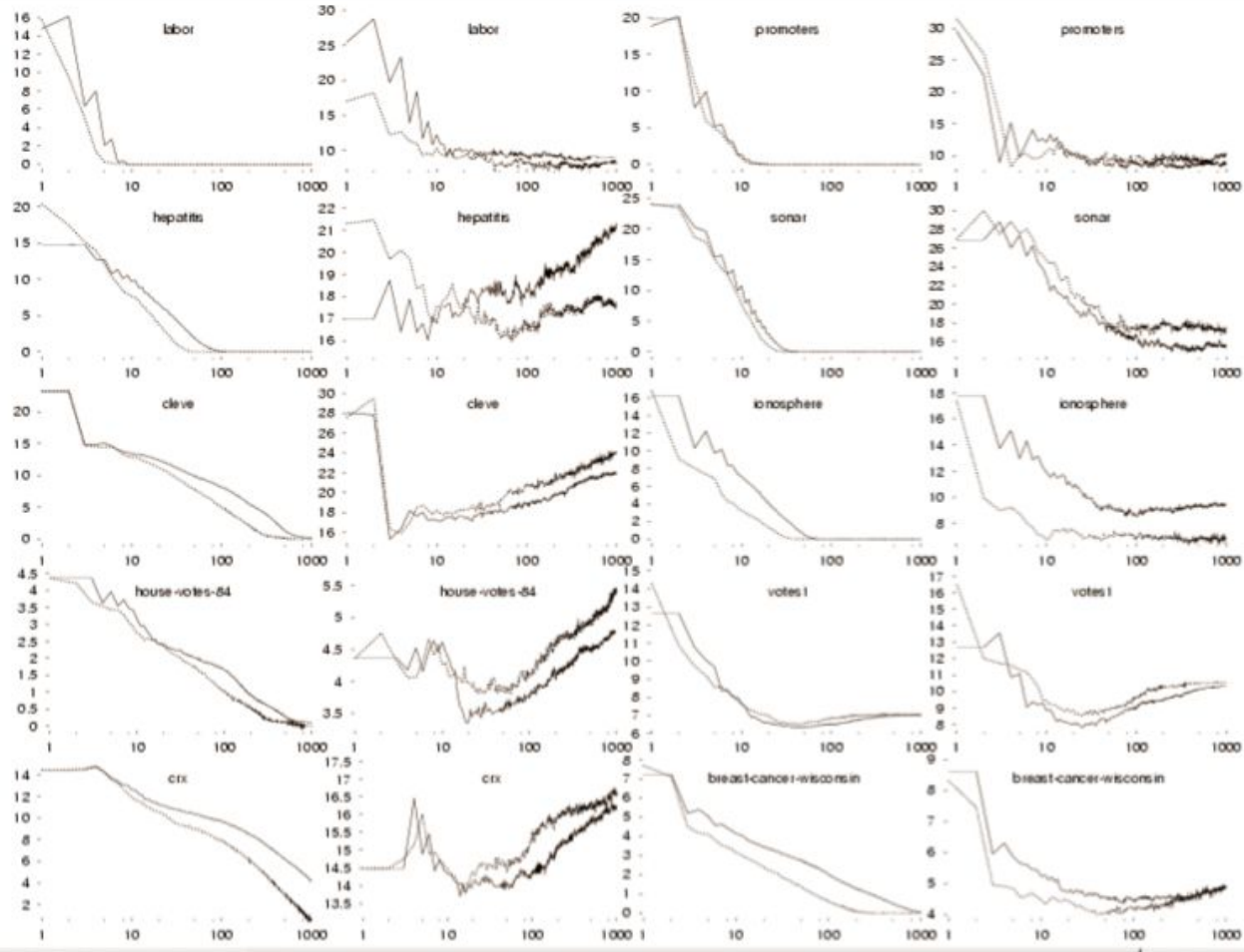
- Robust to overfitting
- Test set error decreases even after training error is zero

# Experimental results

Comparison of C4.5, Boosting C4.5, Boosting decision stumps  
(depth 1 trees), 27 benchmark datasets



AdaBoost and AdaBoost.MH on Train (left) and Test (right) data from Irvine repository. [Schapire and Singer, ML 1999]





# Takeaways

- Combine weak classifiers to obtain very strong classifier
  - Weak classifier – slightly better than random on training data
  - Resulting very strong classifier – can eventually provide zero training error
- AdaBoost algorithm
- Most popular application of Boosting:
  - Boosted shallow decision trees!
  - Very simple to implement, very effective classifier





# Boosting and logistic regression

Logistic regression assumes:

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

And tries to maximize data likelihood:

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

Equivalent to minimizing log loss

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

Boosting minimizes similar loss function!!

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

**Both smooth approximations of 0/1 loss!**



### Logistic regression:

- Minimize loss fn

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

- Define

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

where  $x_j$  predefined

### Boosting:

- Minimize loss fn

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

- Define

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

where  $h_t(x_j)$  defined  
dynamically to fit data  
(not a linear classifier)

- Weights  $\alpha_j$  learned incrementally



# Logistics for midterm

- Choose a time slots on Piazza
- Start the zoom lecture with video on during the time slot you choose
- Visit coursework and download the exam and start
- Submit your solution on coursework under Midterm



# References

- Trevor Hastie, Robert Tibshirani, Jerome Friedman: The Elements of Statistical Learning: Data Mining, Inference and Prediction, Chapter 10
- Ziv Bar-Joseph, Tom Mitchell, Pradeep Ravikumar and Aarti Singh: CMU 10-701