

CSDS 440: Machine Learning

Soumya Ray (he/him, sray@case.edu)

Olin 516

Office hours T, Th 11:15-11:45 or by appointment

Ensemble Methods

- So far, for each task, we construct a single classifier
- Suppose we had a way of constructing *multiple* classifiers and *combining their output*
 - ~~Would this generalize any better than constructing a single classifier?~~
 - ~~Why or why not?~~
 - And how do we construct multiple classifiers in the first place?

Modifying the Training Set

- General idea:
 - Create multiple training sets, each different from the others in some way
 - Apply learning algorithm to each set
 - Resulting classifiers vote on new examples
- Works best for “unstable” algorithms
 - Small change to data can lead to large change in solution
- Two important methods
 - Bagging
 - Boosting

Boosting (FREUND and SCHAPIRE 1996)

- Technique arose from theoretical question: “Is it possible to “boost” a *weak learner* into a *strong learner*?”
 - WL: accuracy better than chance
 - SL: accuracy arbitrarily close to best possible
- Theoretically shown to be possible
- Resulted in a practical algorithm of enormous utility
 - Probably the best known ensemble approach

Adaboost (“Adaptive Boosting”)

- Adaboost is an iterative algorithm
- Maintains a “weight” for each training example (initially all equal)
- In each iteration, it constructs a classifier with the weighted training data (**base learner**)
 - The learner must be able to work with weighted data, usually easy to do this (later)
- Evaluate the resulting classifier on the weighted training data, suppose its error rate is ϵ
 - If $\epsilon=0$ or $\epsilon \geq \frac{1}{2}$, stop

Adaboost

- In the next iteration,
 - **Correctly** classified training examples have their weights **multiplied** by a factor proportional to ϵ
 - **Incorrectly** classified training examples have their weights **divided** by a factor proportional to ϵ

Adaboost

- After completion, the resulting classifiers are combined by a *weighted vote*
 - The weight of each classifier is *inversely proportional to its error rate*
 - (This weight is different from the example weights above)

Adaboost (Training)

- Initialize weights w_n to $1/N$, $n=1\dots N$ (N examples)
- Each iteration t
 - Train base learner h_t with *weighted* data
 - Calculate *weighted training error* of this classifier:

$$\varepsilon_t = \sum_{n=1}^N w_n^t I(y_n \neq h_t(x_n))$$

- Break if $\varepsilon_t=0$ or $\varepsilon_t \geq 0.5$

Adaboost (Training)

- Each iteration (continued)
 - Set *weight of this classifier* for new examples:

$$\alpha_t = \frac{1}{2} \log \frac{1 - \varepsilon_t}{\varepsilon_t}$$

- Update example weights:

$$w_n^{t+1} = \frac{1}{Z_t} w_n^t e^{-\alpha_t y_n h_t(x_n)}$$

- (Z is a normalization constant so all weights sum to 1, and we assume y is +1 and -1)

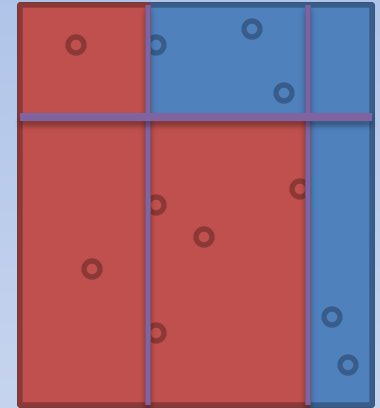
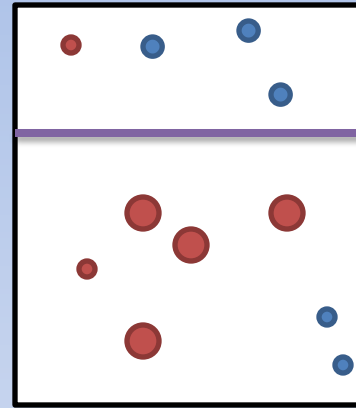
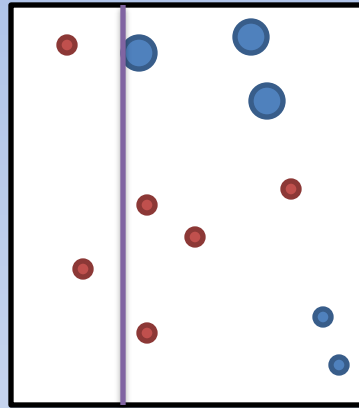
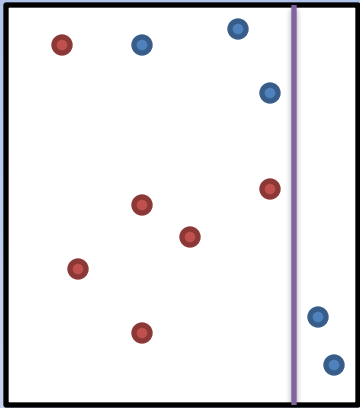
Adaboost (Classification)

- For new example x , output

$$f(x) = \sum_{t=1}^T \frac{\alpha_t}{\sum_r \alpha_r} h_t(x)$$

$$\left(\alpha = \frac{1}{2} \log \frac{1 - \varepsilon}{\varepsilon} \right)$$

What is this doing?



What is Adaboost doing?

- In general, Adaboost:
 - Often helps accuracy significantly
 - In some cases hurts accuracy significantly
 - But, helps *much more often* than it hurts
- Key result: Adaboost exponentially decreases the loss on the training set as a function of the number of iterations it runs

Why Adaboost works (1)

- Still an active research area
- One explanation: boosting works by reducing “bias error”
 - Bias error is the error on a dataset due to choice of concept class
 - The ensemble classifier produced by boosting has lower bias error compared to any single member
 - Theoretically, this could increase the chance of overfitting
 - Rarely observed in practice, unless very noisy samples

Why Adaboost works (2)

- Adaboost can be viewed as a “margin maximization” algorithm (later;ask for paper)
 - Such algorithms have better generalization behavior
- Observation: The generalization error of the voted classifier improves *even after* its training set error goes to zero

Why Adaboost works (3)

- Sometimes, using a simple base classifier can prevent overfitting when there is noise
 - Rui Liu's thesis (MS 2016) on boosted linear classifiers (also in ICDM 2017)
 - (ask for copy)

Handling Weighted Data

- Naïve Bayes: Use weighted statistics

$$\Pr(X_i = 1 \mid Y = 1) = \frac{\sum_j w_j I(X_{ij} = 1 \wedge Y_j = 1)}{\sum_k w_k I(Y_k = 1)}$$

- Decision Trees: Use weighted entropy

$$p_w(X = v) = \frac{\sum_{\{i: X_i=v\}} w_i}{\sum_i w_i}$$

Handling Weighted Data

- SVMs: Use weighted objective function

$$\min_{\mathbf{w}, \mathbf{b}, \xi} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_i (weight_i \xi_i)$$

- Neural Nets, Logistic Regression: similar updates

Another view of Adaboost

- Consider a function that minimizes the objective:

$$F(\mathbf{a}, \mathbf{h}) = \sum_{i=1}^m e^{-y_i \sum_t \alpha_t h_t(x_i)}$$

Ensemble prediction for x_i

- In a *stagewise* manner: Given $\alpha_1 \dots \alpha_{t-1}, h_1 \dots h_{t-1}$, what is α_t and h_t ?
- Here h_t is the new “direction” and α_t the new “stepsize”
- Minimize using gradient descent

Another view of Adaboost

- It turns out that $h_t = \operatorname{argmin}_h \varepsilon_t$ and

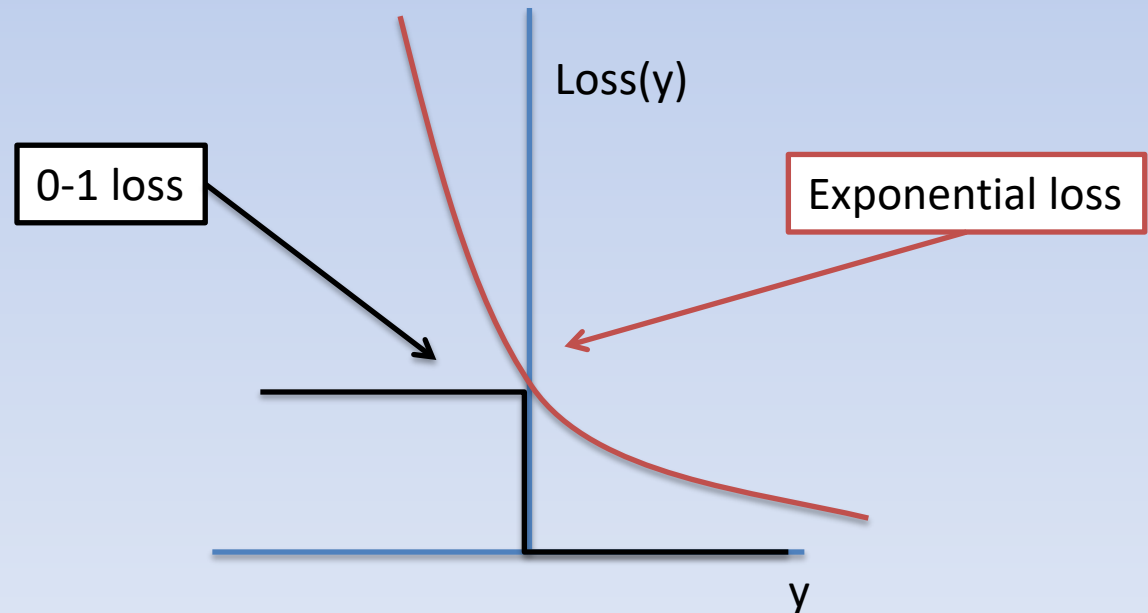
$$\alpha_t = \frac{1}{2} \log \frac{1 - \varepsilon_t}{\varepsilon_t}$$

- Exactly as done by Adaboost

Another view of Adaboost

- The function: $F(\mathbf{a}, \mathbf{h}) = \sum_{i=1}^m e^{-y_i \sum_t \alpha_t h_t(x_i)}$

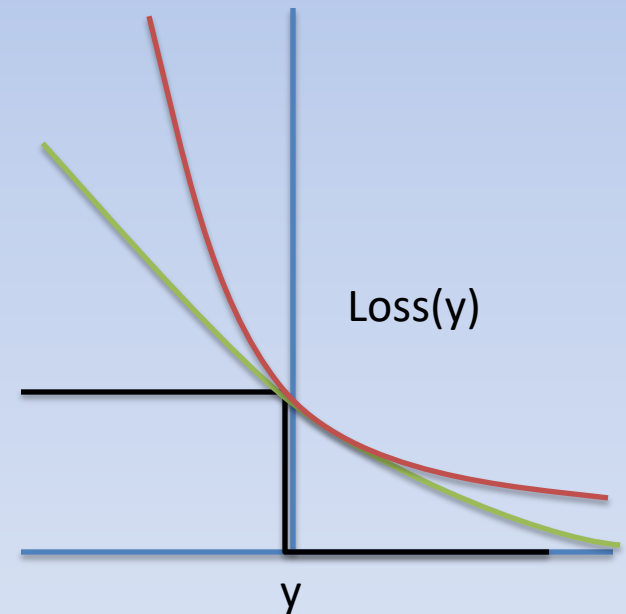
Ensemble prediction for x_i
- Is an *exponential* loss function



Connections to other algorithms

- What if we *replace* the exponential loss with other functions?

$$G(\mathbf{a}, \mathbf{h}) = \sum_{i=1}^m \log \left(1 + e^{-y_i \sum_t \alpha_t h_t(x_i)} \right)$$



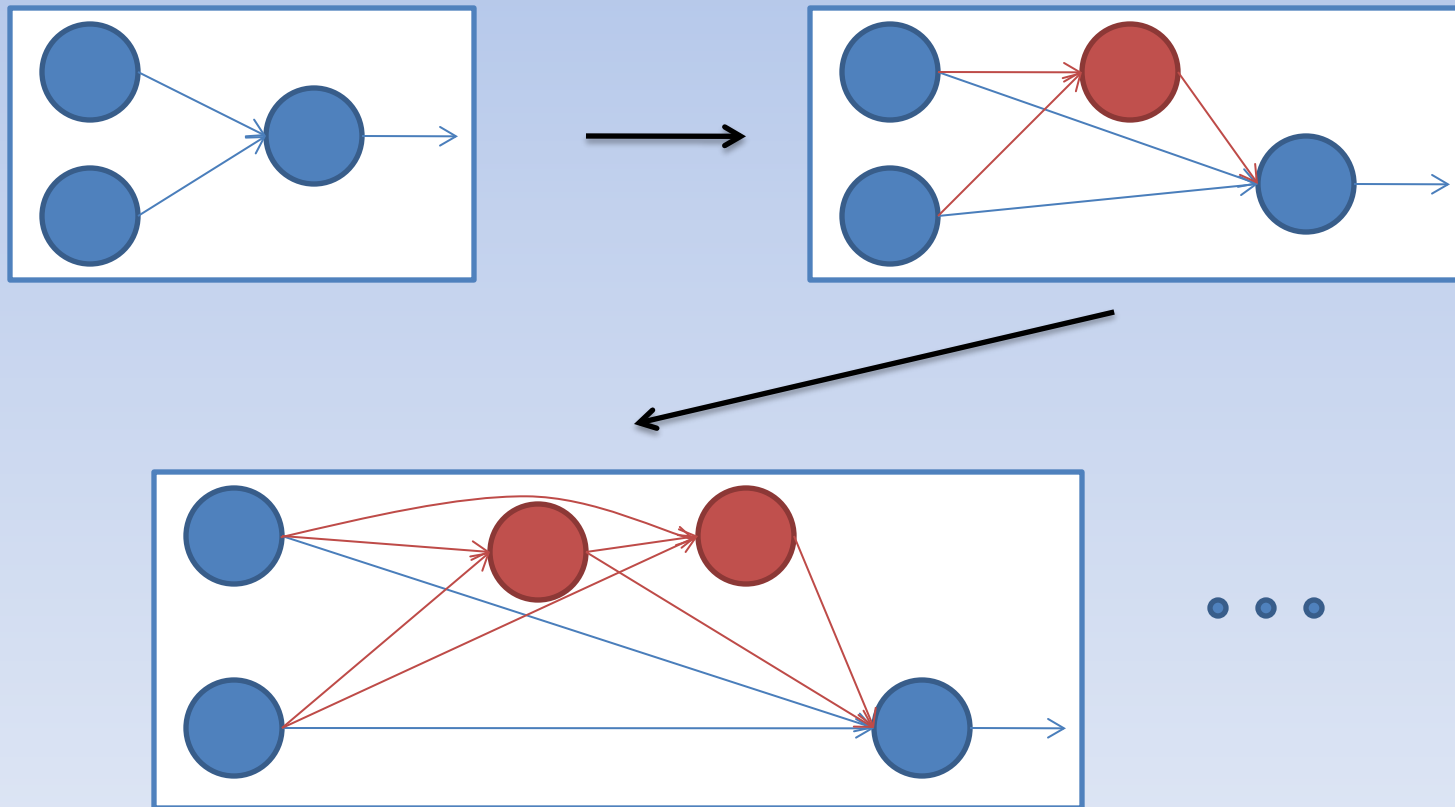
Gradient Boosting (FRIEDMAN 99)

- Like boosting, but optimize a different loss each iteration
- In each iteration find a new classifier that minimizes the *residual of the previous iteration's loss* on the training sample

Cascade Correlation for ANNs
Implements Gradient
Boosting specifically for the
perceptron

Learning the Structure

- Cascade Correlation



Other approaches

- Many other approaches to combining classifiers in the literature
 - Stacking, arcing, random forests, etc.
- Each has advantages and disadvantages
 - Generally empirical and not as well understood as boosting/bagging