CSDS 440: Machine Learning

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

Olin 516

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

Zoom Link

Recap

- In an ensemble, if we assume all classifiers are u_____, the distribution of wrong classifiers is b_____.
- Many algorithms can get stuck in l___ o____. One reason ensembles do well is because they a____ these l___ o___ classifiers.
- An ensemble has a simpler/ more complex decision boundary than its constituents.
- The optimal way to classify a new example given some training data is given by B____ m_____ a_____. An ensemble is an a_____ of this procedure.
- What are some downsides of using an ensemble?
- We can construct an ensemble by m____ the t___ s___.

Today

- Part 2: Ensemble Methods

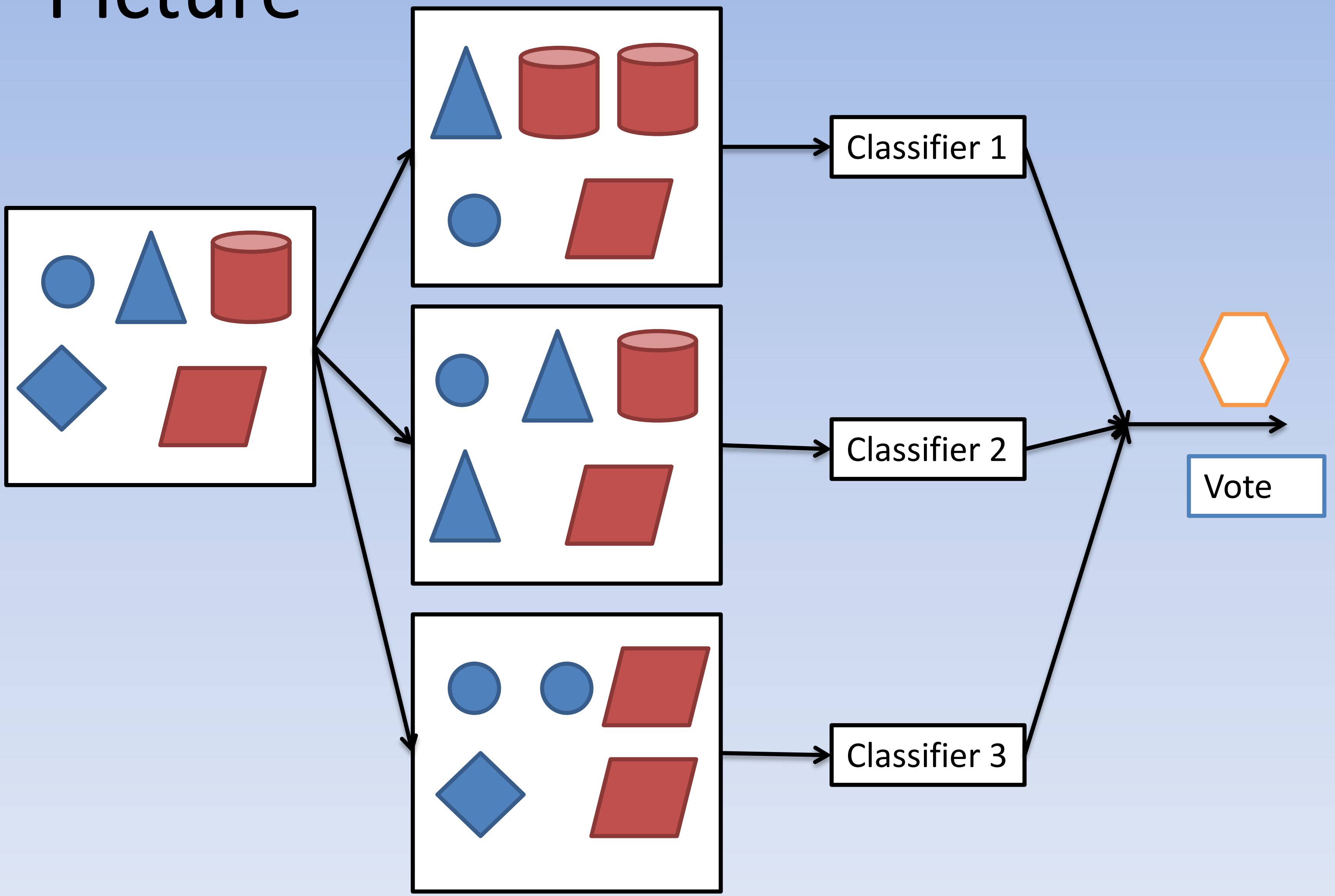
Modifying the Training Set

- 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

Bagging (Breiman 96)

- "**B**ootstrap **Agg**regation"
- Each training sample is a bootstrap replicate of the initial set
  - If the set has size m, sample m examples *uniformly with replacement* from it
- To classify a new example, use majority voting

Picture



Bagging

- In practice, Bagging:
  - Rarely or never hurts accuracy
  - But improvements in accuracy are likewise small
- Voting classifiers constructed with bootstrap replicates can result in "averaging out" the effect of noise

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 data
  - 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 ε=0 or $\varepsilon \geq$ ½, stop

Adaboost

- In the next iteration,
  - Each correctly classified training example has its weight multiplied by a factor proportional to ε
  - Each incorrectly classified training example has its weight 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 Pseudocode (Training)

- Initialize weights w_n to $1/N$, n=1…N (N examples)
- Each iteration t
  - Train weak/base learner h_t with *weighted* sample
  - 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 ε_t=0 or $\varepsilon_t \geq 0.5$

Adaboost Pseudocode (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)

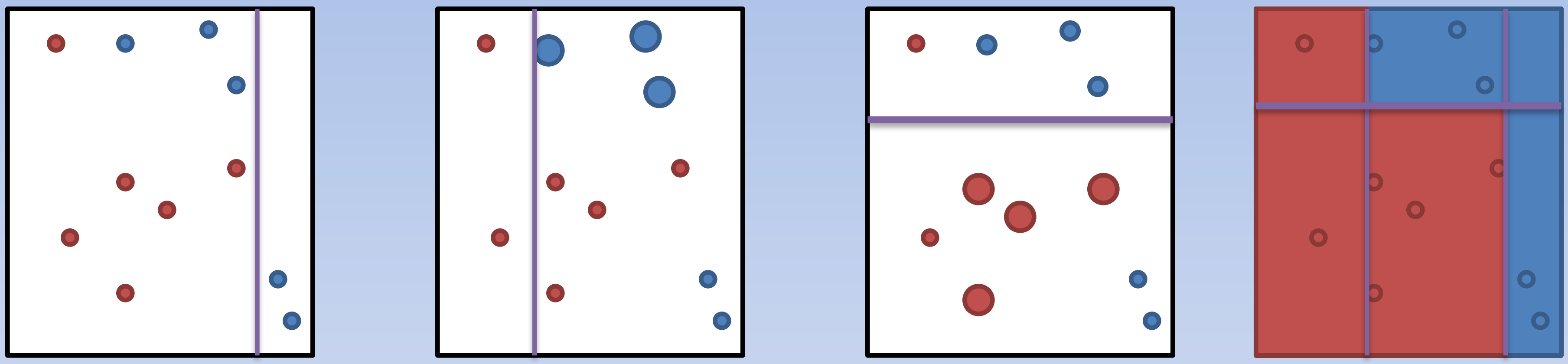
Adaboost Pseudocode (Classification)

- 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 (ask for paper)
- Increasing weights on the misclassified examples may force the learner to produce a classifier that has larger margins on all of the training data
- 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

$$p_w\left(X = v\right) = \sum_{\{i : X_i = v\}} w_i \Big/ \sum_i w_i$$

Handling Weighted Data

- SVMs: Use weighted objective function

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

- Neural Nets, Logistic Regression: similar updates

Another view of Adaboost

- Consider a function that minimizes the objective:

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

Ensemble prediction for x_i

- In a *stagewise* manner: Given $\alpha_1 \ldots \alpha_{t\text{-}1}$, $h_1 \ldots h_{t\text{-}1}$, 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=argmin$_h$ ε_t and

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

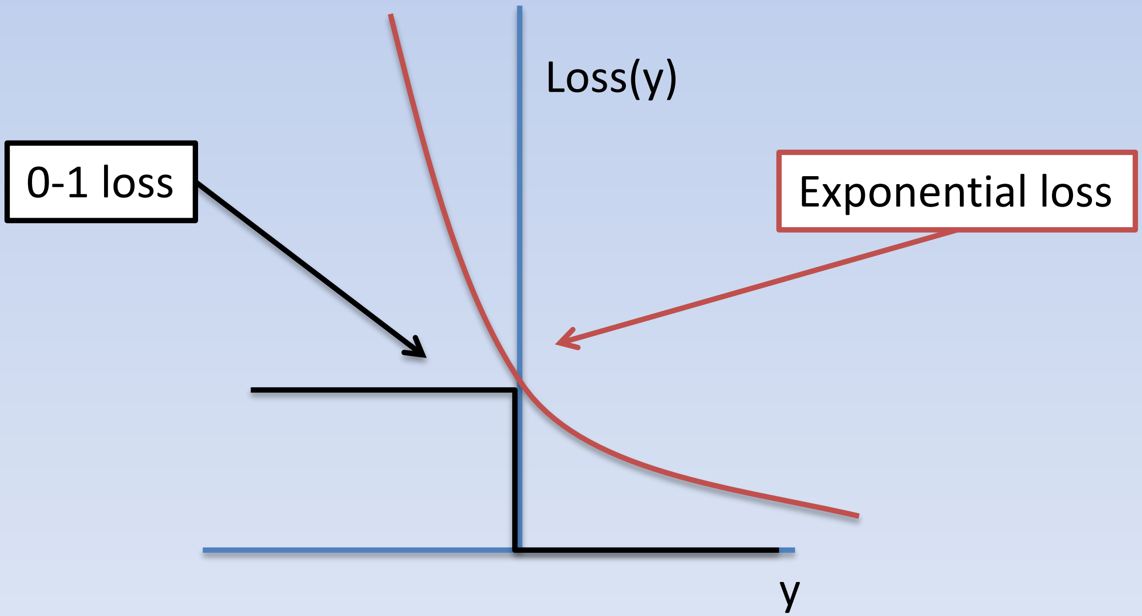
- Exactly as done by Adaboost

Another view of Adaboost

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

Ensemble prediction for x_i

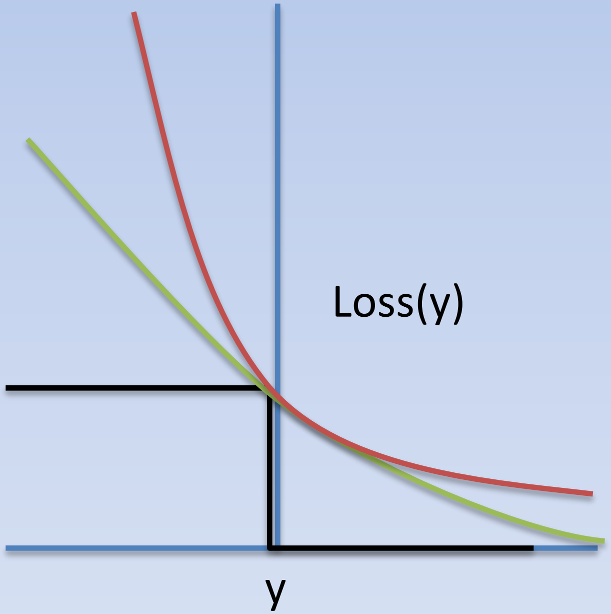
- Is an *exponential* loss function



Connections to other algorithms

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

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



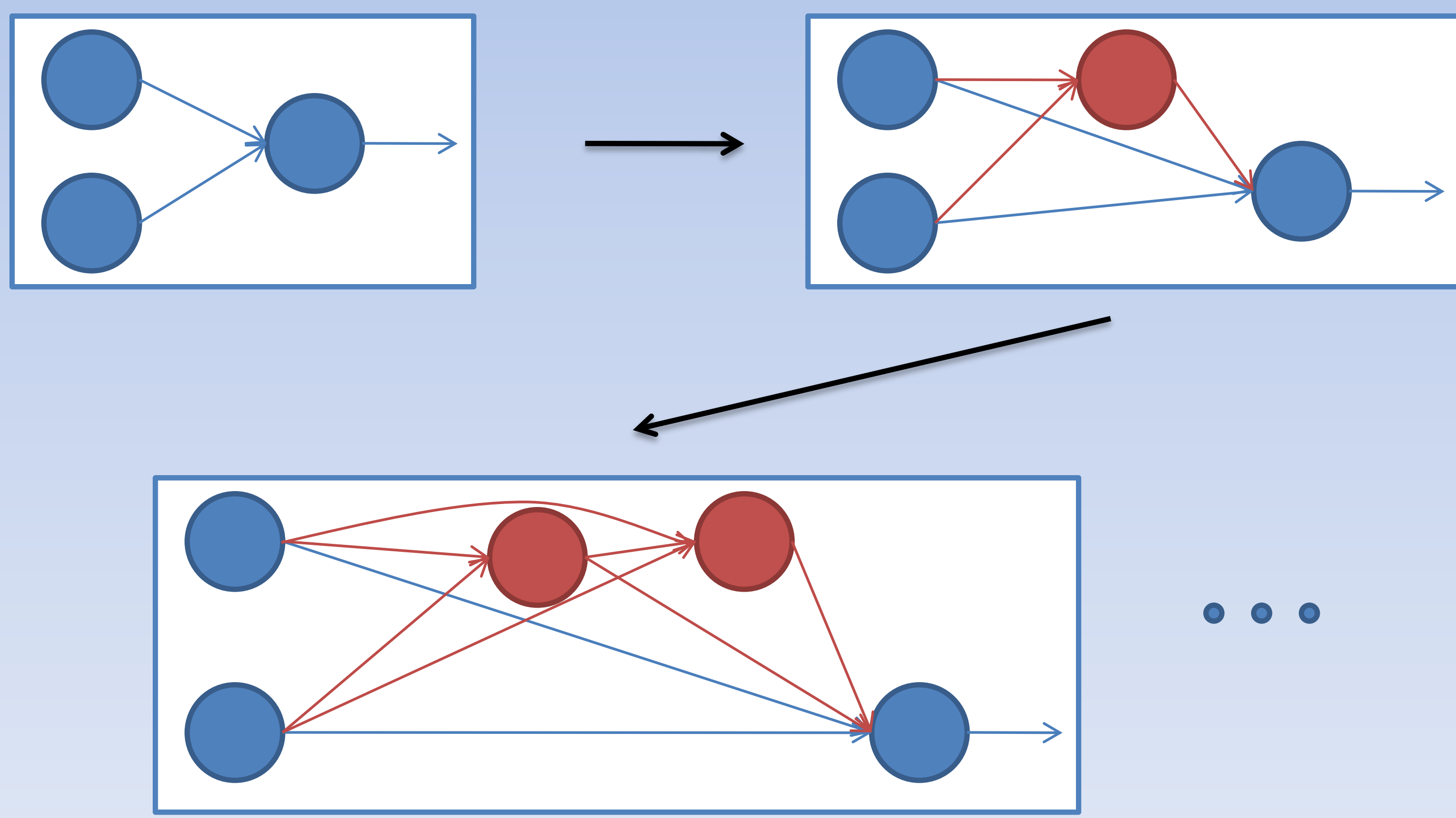
Gradient Boosting (FRIEDMAN 99)

- 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