AI and Machine Learning

Zhiyun Lin

南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

Ensemble Methods

- Bagging
- Boosting
- Random/Decision Forest
- Mixture of Experts

Ensemble Methods

- **Ensemble** of classifiers is a set of classifiers whose individual decisions are combined in some way to classify new examples
- Simplest approach:
  i. Generate multiple classifiers
  ii. Each votes on test instance
  iii. Take majority as classification
- Classifiers are different due to different sampling of training data, or randomized parameters within the classification algorithm
- **Aim:** Take simple mediocre algorithm and transform it into a super classifier without requiring any fancy new algorithm

Ensemble Methods: Summary

- Differ in training strategy, and combination method
  - **Parallel training** with different training sets

    Bagging (bootstrap aggregation) - train separate models on overlapping training sets, average their predictions
  - **Sequential training**, iteratively re-weighting training examples so current classifier focuses on hard examples: **boosting**
  - Parallel training with objective encouraging division of labor: **mixture of experts**
- Notes:
  - Also known as meta-learning
  - **Typically applied to weak models,** such as decision stumps (single-node decision trees), or linear classifiers

Variance-bias Tradeoff

- Minimize two sets of errors:
  i. **Variance:** Error from sensitivity to small fluctuations in the training set
  ii. **Bias:** Erroneous assumptions in the model

Wy do Ensemble Methods Work?

- Based on one of two basic observations:

  i. **Variance reduction:** If the training sets are completely independent, it will always help to average an ensemble because this will reduce variance without affecting bias (e.g., bagging)

  > reduce sensitivity to individual data points

  ii. **Bias reduction:** For simple models, average of models has much greater capacity than single model (e.g., hyperplane classifiers, Gaussian. densities)

  > Averaging models can reduce bias substantially by increasing capacity, and control variance by fitting one component at a time (e.g., boosting)

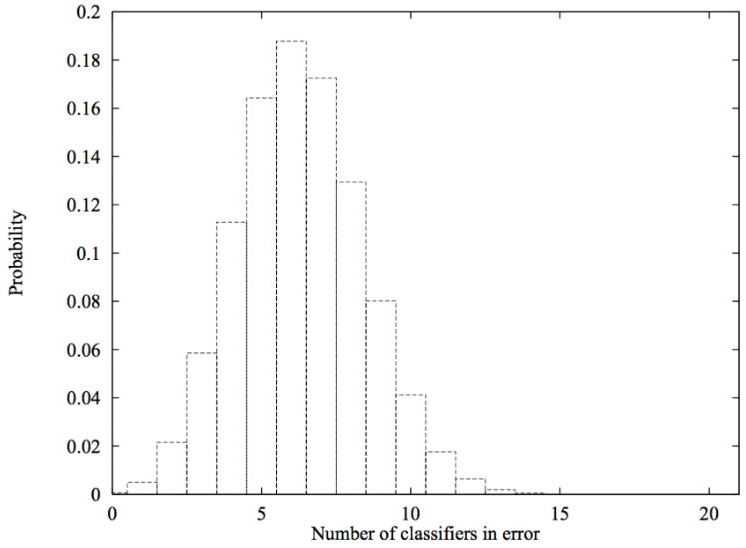
Ensemble Methods: Justification

- Ensemble methods **more accurate** than any individual members if:
  - Accurate (better than guessing)
  - Diverse (different errors on new examples)
- Why?
- Independent errors: Prob k of N classifiers (independent error rate ϵ) wrong:

$$P(\text{num errors} = k) = \binom{N}{k} \epsilon^k (1-\epsilon)^{N-k}$$

- Probability that majority vote wrong: **error under distribution where more than $N/2$ wrong**

Ensemble Methods: Justification



- Example: The probability that exactly K (out of 21) classifiers will make an error assuming each classifier has an error rate of $\epsilon = 0.3$ and makes its errors independently of the other classifier. The area under the curve for 11 or more classifiers being simultaneously wrong is 0.026 (much less than e).

Ensemble Methods: Justification

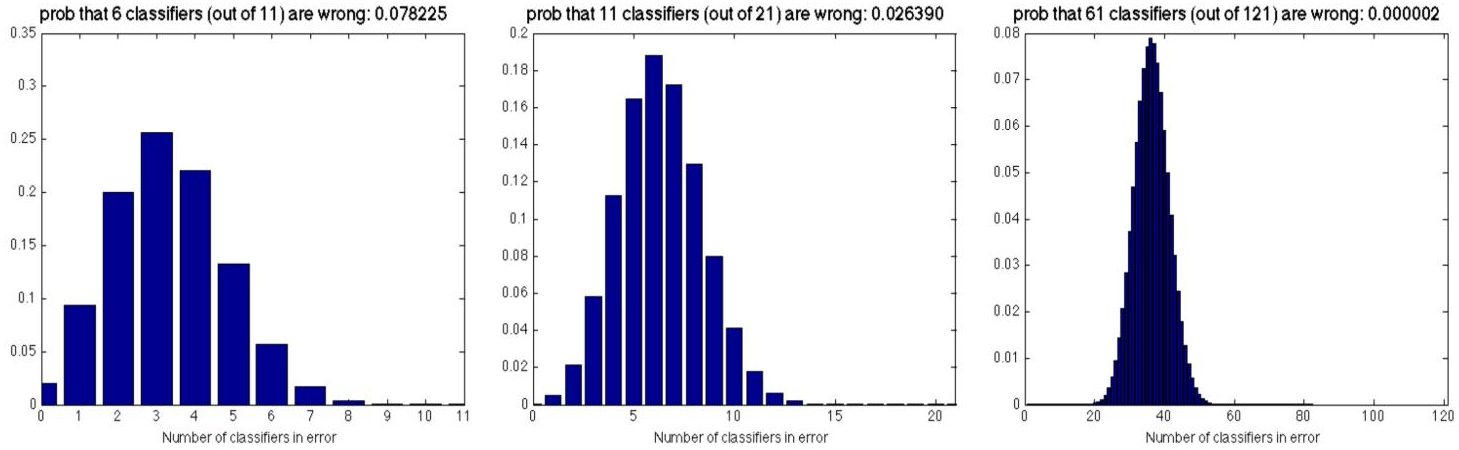


Figure : $\epsilon = 0.3$: (**left**) $N = 11$ classifiers, (**middle**) $N = 21$, (**right**) $N = 121$.

Bootstrap Estimation

- Repeatedly draw n samples from N
- For each set of samples, estimate a statistic
- The bootstrap estimate is **the mean of the individual estimates**
- Used to estimate a statistic (parameter) and its variance

> **Bagging**: **b**ootstrap **agg**regation (Breiman 1994)

Bagging

- **Simple idea:** Generate M bootstrap samples from your original training set.
- Train on each one to get y_m, and average them

$$y_{bag}^{M}(x) = \frac{1}{M} \sum_{m=1}^{M} y_m(x)$$

- **For regression:** average predictions
- **For classification:**
  - average class probabilities
  - or take the majority vote if only hard outputs available

Bagging

- Bagging approximates the Bayesian posterior mean. The more bootstraps the better, so use as many as you have time for
- Each bootstrap sample is drawn with replacement, so each one contains some **duplicates of certain training points** and leaves out other training points completely

Boosting (AdaBoost): Summary

- Also works by manipulating training set, but **classifiers trained sequentially**
- Each classifier trained given knowledge of the performance of previously trained classifiers: **focus on hard examples**
- Final classifier: **weighted sum** of component classifiers

Making Weak Learners Stronger

- Suppose you have a weak learning module (a **base classifier**) that can always get $(0.5 + \epsilon)$ correct when given a two-way classification task
  - That seems like a weak assumption but beware!
- Can you apply this learning module many times to get a **strong learner** that can get close to zero error rate on the training data?
  - Theorists showed how to do this and it actually led to an effective new learning procedure (Freund & Shapire, 1996)

Boosting (ADAboost)

- First train the base classifier on all the training data with **equal importance weights** on each case.
- Then **re-weight the training data** to emphasize the hard cases and train a second model.
  - How do we re-weight the data?
- Keep training new models **on re-weighted data**
- Finally, use a **weighted committee** of all the models for the test data.
  - How do we weight the models in the committee?

How to Train Each Classifier

- Input: x, Output: $y(x) \in \{1, -1\}$
- Target $t \in —1, 1$
- Weight on example n for classifier m: w_n^m
- Cost function for classifier m:

$$J_m = \sum_{n=1}^{N} w_n^m \underbrace{[y_m(x^n) \neq t^{(n)}]}_{\text{1 if error, 0 o.w.}} = \sum \text{weighted errors}$$

How to weight each training case for classifier m

- Recall cost function:

$$J_m = \sum_{n=1}^{N} w_n^m \underbrace{[y_m(x^n) \neq t^{(n)}]}_{\text{1 if error, 0 o.w.}} = \sum \text{weighted errors}$$

- **Weighted error rate** of a classfier

$$\epsilon_m = \frac{J_m}{\sum_n w_n^m}$$

- The **quality of the classifier** is

$$\alpha_m = \ln\left(\frac{1-\epsilon_m}{\epsilon_m}\right)$$

> It is zero if the classifier has weighted error rate of 0.5 and infinity if the classifier is perfect

How to weight each training case for classifier m

The **weights for the next round** are then

$$w_n^{m+1} = \exp\left(-\frac{1}{2}t^{(n)}\sum_{i=1}^{m}\alpha_i y_i(x^{(n)})\right) = w_n^m \exp\left(-\frac{1}{2}t^{(n)}\alpha_m y_m(x^{(n)})\right)$$

> If at this round, the weighted error rate is lower than 0.5, then the quality α_i of the classifier will be postive.

- If the prediction of example n is correct, then $t^{(n)}y_m^{(n)} > 0$. Then the weight for this example decays.
- On the other hand, if the prediction of example n is wrong, then $t^{(n)}y_m^{(n)} < 0$. Then the weight for this example grows.

How to make predictions using a committee of classifiers

- Weight the binary prediction of each classifier by the quality of that classifier:

$$y(x) = \text{sign}\left(\sum_{m=1}^{M} \alpha_m y_m(x)\right)$$

- This is how to do inference, i.e.,how to compute the prediction for each new example.

AdaBoost Algorithm

- Input: $\{x^{(n)}, t^{(n)}\}_{n=1}^{N}$, and **WeakLearn:** learning procedure, produces classfier $y(x)$
- Initialize example weights: $w_n^1(x) = \frac{1}{N}$
- For $m = 1 : M$
  - $y_m(x) = WeakLearn(x, t, w)$, fit classifier by minimizing

$$J_m = \sum_{n=1}^{N} w_n^m [y_m(x^n) \neq t^{(n)}]$$

  - Compute unnormalized error rate

$$\epsilon_m = \frac{J_m}{\sum_n w_n^m}$$

AdaBoost Algorithm

- ◦ Compute clasifie quality

$$\alpha_m = \ln\left(\frac{1-\epsilon_m}{\epsilon_m}\right)$$

  - ◦ Update data weights

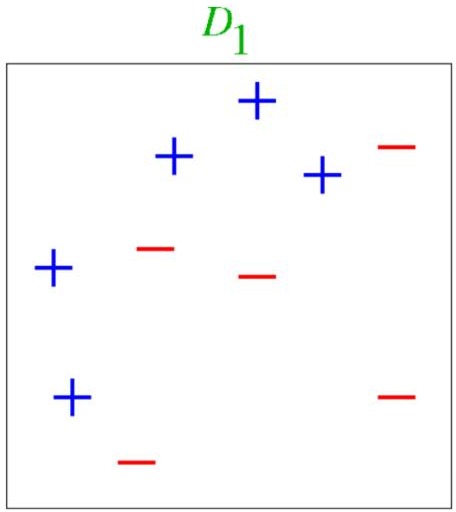
$$w_n^{m+1} = w_n^m \exp\left(-\frac{1}{2} t^{(n)} \alpha_m y_m(x^{(n)})\right)$$

- Final model

$$y(x) = \text{sign}\left(\sum_{m=1}^{M} \alpha_m y_m(x)\right)$$

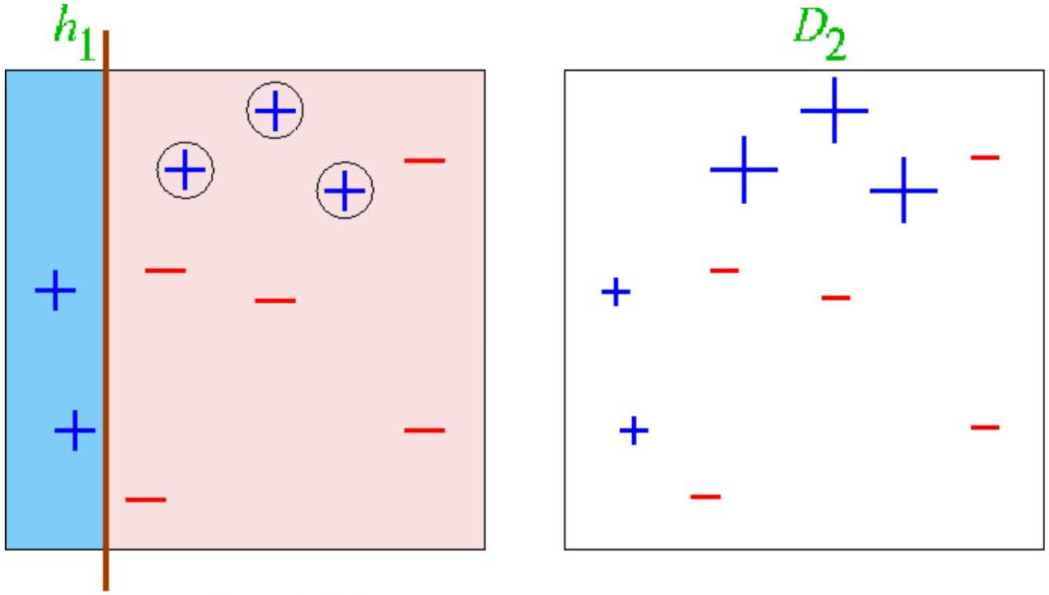
AdaBoost Example

- Training data



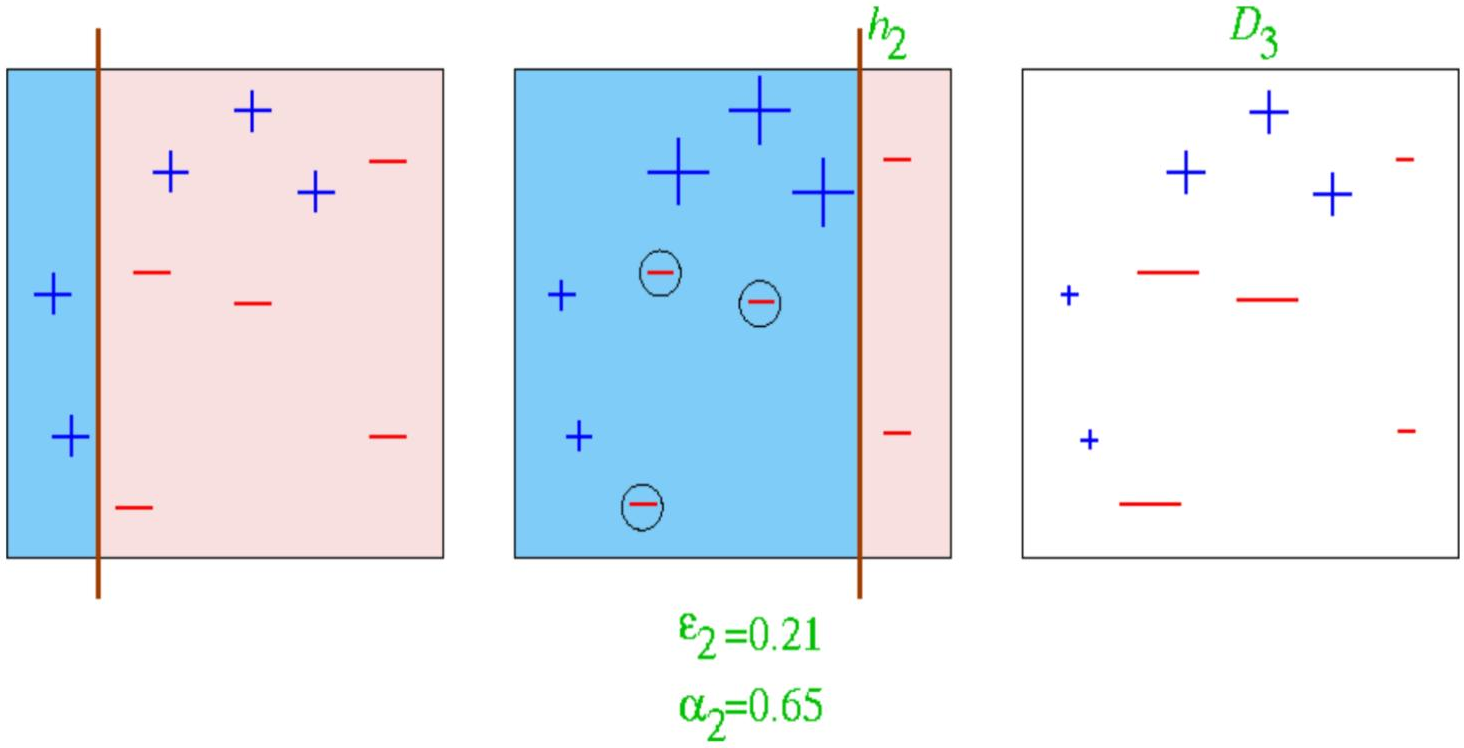
AdaBoost Example

- Round 1



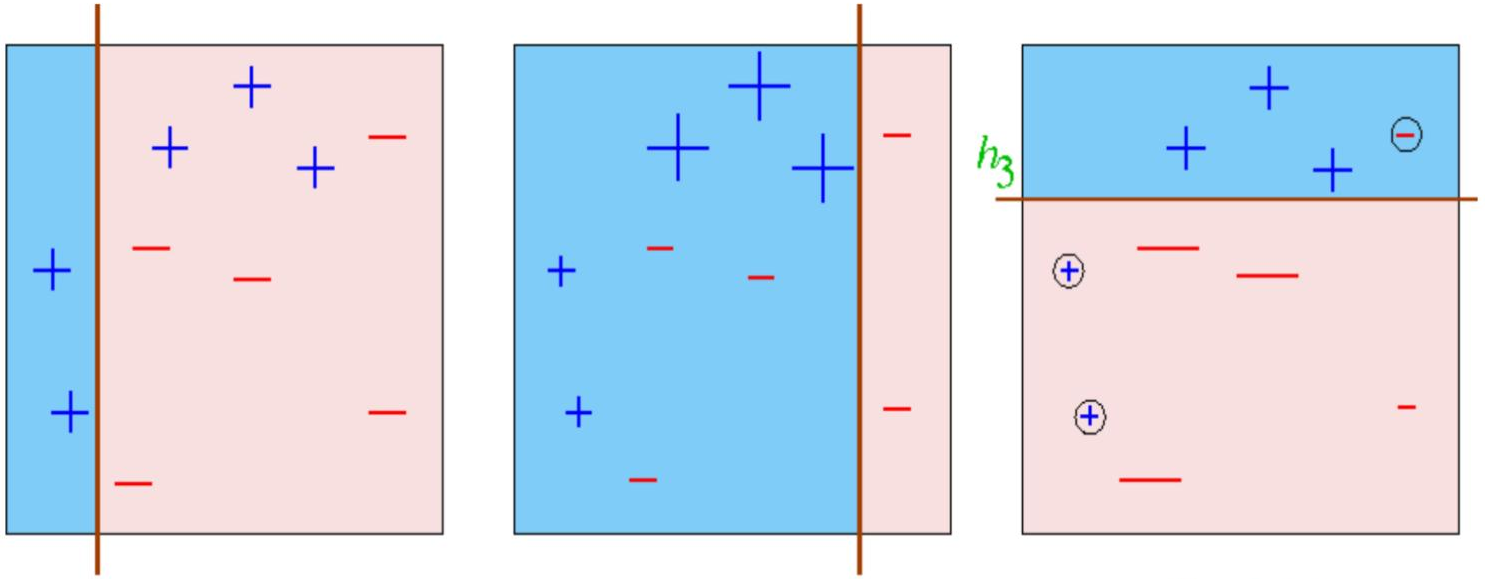
AdaBoost Example

- Round 2



AdaBoost Example

- Round 3

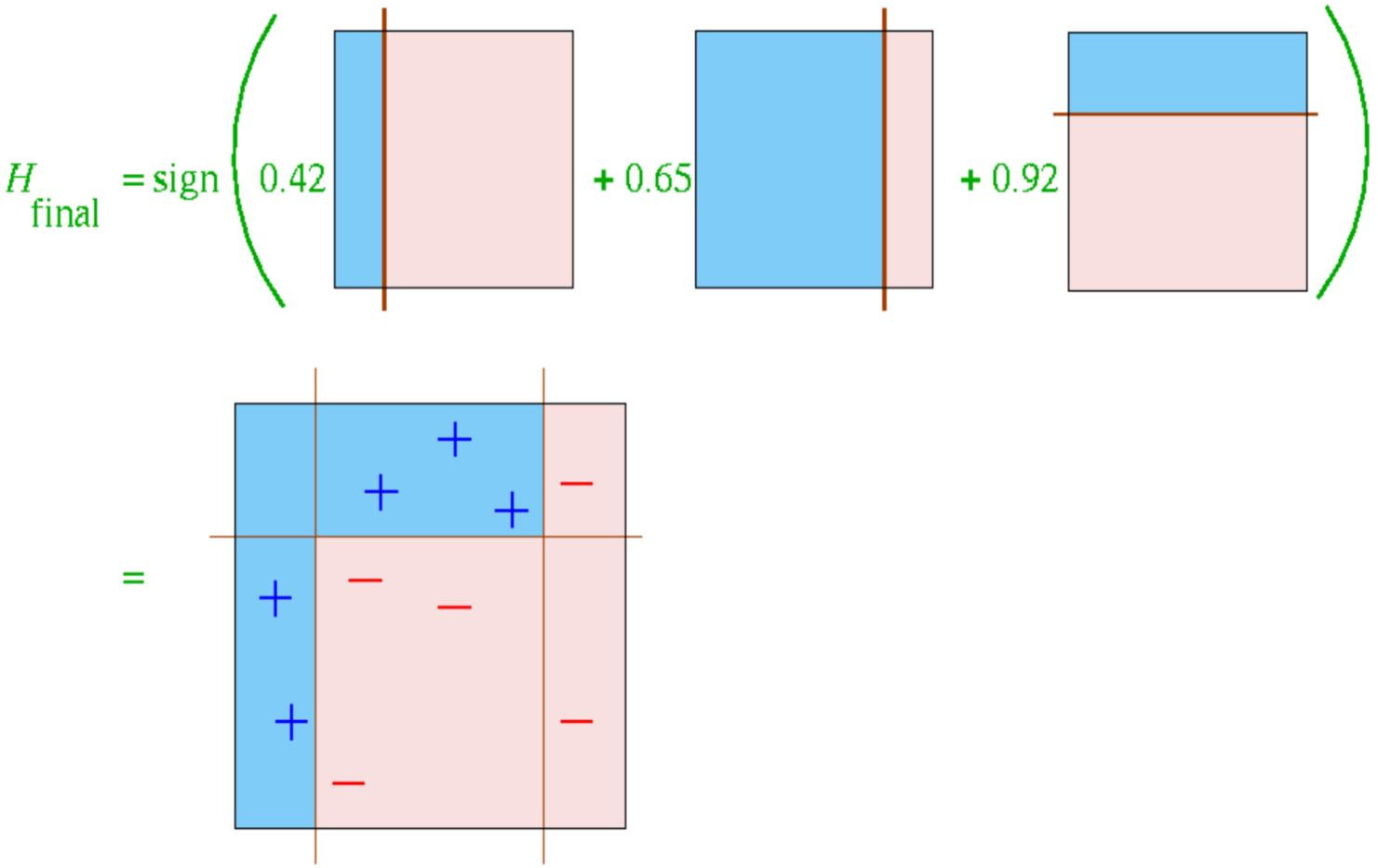


$\varepsilon_3 = 0.14$

$\alpha_3 = 0.92$

AdaBoost Example

- Final



What are the base classifiers?

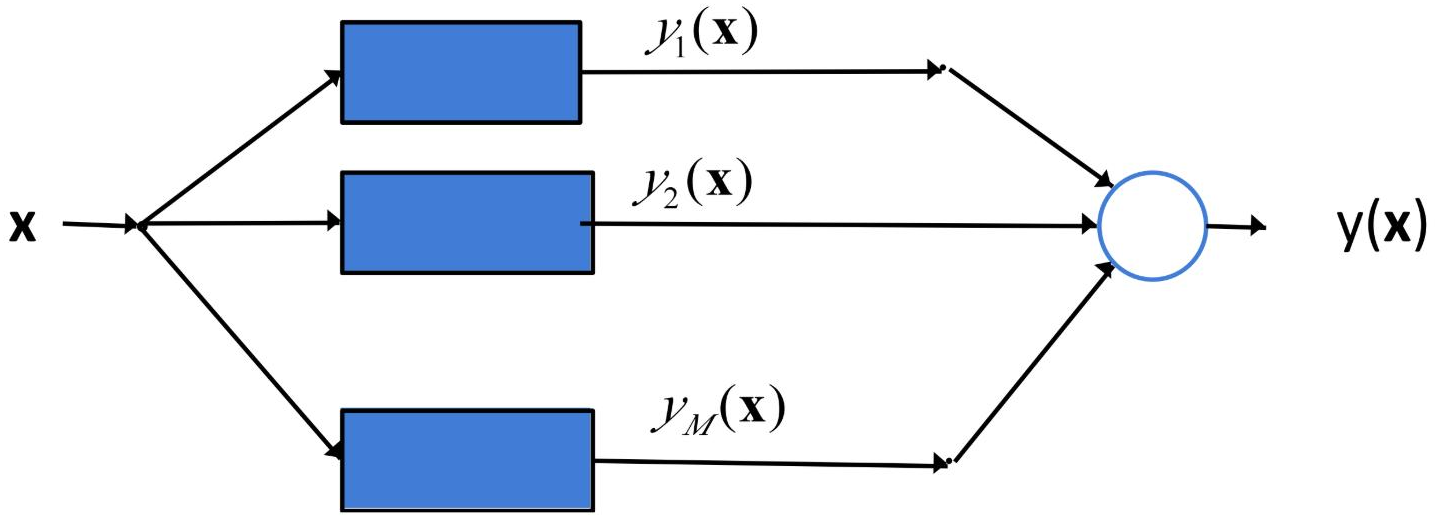
- Popular choices of base classifier for boosting and other ensemble methods:
  - Linear classifiers
  - Decision trees

Random/Decision Forests

- **Definition:** Ensemble of decision trees
- Algorithm:
  - Divide training examples into multiple training sets (bagging)
  - Train a decision tree on each set (can randomly select subset of variables to consider)
  - Aggregate the predictions of each tree to make classification decision (e.g., can choose mode vote)

Ensemble Learning: Boosting and Bagging

- Experts cooperate to predict output

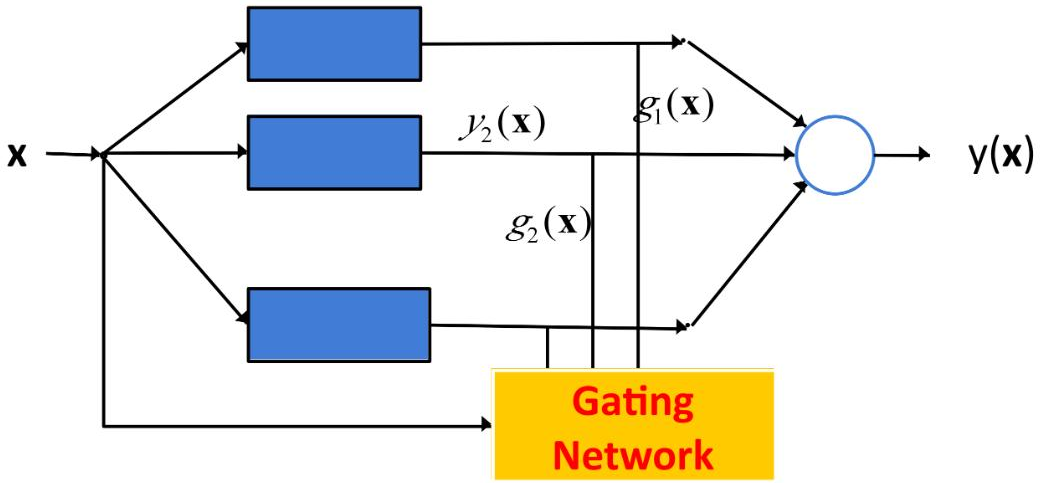


- Vote of each expert has **consistent weight** for each test example

$$y(x) = \sum_m \alpha_m y_m(x)$$

Mixture of Experts

- Weight of each expert is not constant – depends on input **x**



- Gating network **encourages specialization** (local experts) instead of cooperation

$$y(x) = \sum_m g_m(x) y_m(x)$$

> g_m refers the gating network

Mixture of Experts: Summary

1. Cost function designed to make each expert estimate desired output **independently**
2. **Gating network softmax over experts:** stochastic selection of who is the true expert for given input
3. Allow each expert to produce **distribution over outputs**

Cooperation vs. Specialization

- Consider a regression problem
- To encourage cooperation, we can train to reduce discrepancy between average of predictors with target

$$E = \left(t - \frac{1}{M} \sum_m y_m(x) \right)^2$$

- This can overfit badly. It makes the model much more powerful than training each predictor separately

Cooperation vs. Specialization

- To encourage specialization, train to reduce the average of each predictor's discrepancy with target

$$E = \frac{1}{M} \sum_m (t - y_m(x))^2$$

- Use a weighted average: weights are probabilities of picking that"expert" for the particular training case

$$E = \sum_m g_m(x)(t - y_m(x))^2$$

- Gating output is softmax of $z = Ux$

$$g_m(x) = \frac{\exp(z_m(x))}{\sum_i \exp(z_i(x))}$$

- We want to estimate the parameters of the gating as well as the classifier y_m

Derivatives of Simple Cost Functions

- Look at derivatives to see what cost function will do

$$E = \sum_m g_m(x)(t - y_m(x))^2$$

- For gating network, increase weight on expert when its error is less than average error of experts

$$\frac{\partial E}{\partial z_m} = g_m(x)\left[(t - y_m(x))^2 - E\right]$$

Derivatives of Simple Cost Functions

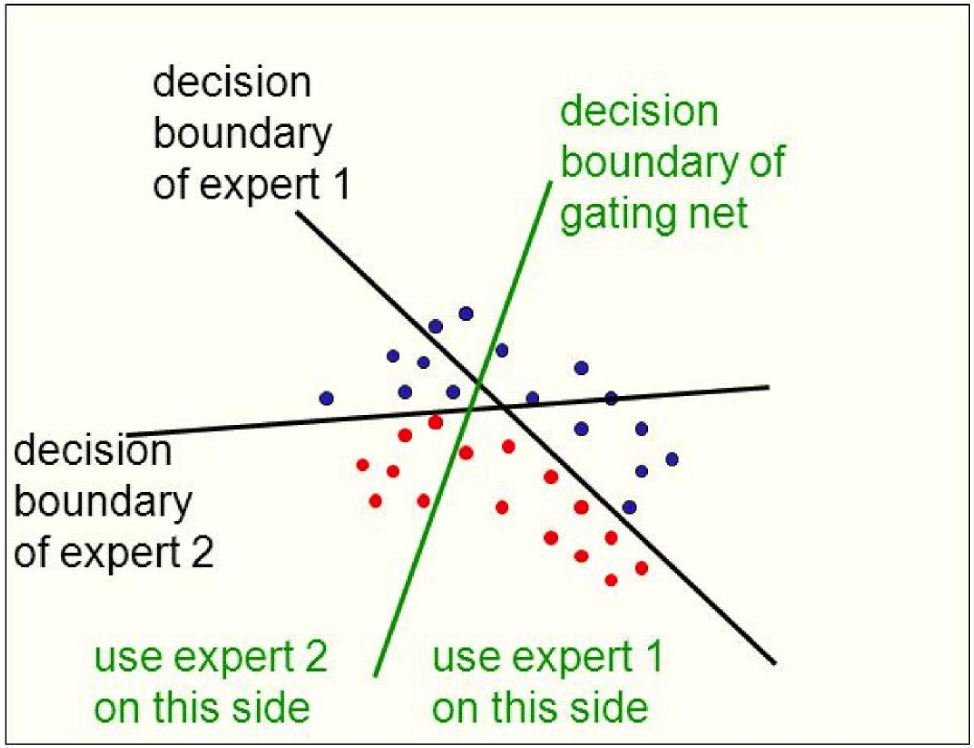
- To see this, recall that

$$\frac{\partial g_m}{\partial z_m} = g_m(1 - g_m)$$
$$\frac{\partial g_i}{\partial z_m} = -g_i g_m, \quad \forall i \neq m$$

- Thus, we have:

$$\begin{aligned}
\frac{\partial E}{\partial z_m} &= \frac{\partial E}{\partial y_m}\frac{\partial y_m}{\partial z_m} + \sum_{i \neq m} \frac{\partial E}{\partial g_i}\frac{\partial g_i}{\partial z_m} \\
&= \left[(t - y_m(x))^2 g_m(1 - g_m) - \sum_{i \neq m} (t - y_i(x))^2 g_i g_m \right] \\
&= g_m \left[(t - y_m(x))^2 - \sum_i (t - y_i(x))^2 g_i \right] \\
&= g_m(x) \left[(t - y_m(x))^2 - E \right]
\end{aligned}$$

Mixture of Experts: Example



Ensemble Methods: Summary

- Differ in training strategy, and combination method
  - Parallel training with different training sets

    Bagging (bootstrap aggregation) - train separate models on overlapping training sets, average their predictions
  - Sequential training, iteratively re-weighting training examples so current classifier focuses on hard examples: **boosting**
  - Parallel training with objective encouraging division of labor: **mixture of experts**
- Notes:
  - Differ in: training strategy; selection of examples; weighting of components in final classifier