

# ITCS 6156/8156 Fall 2024

## Machine Learning

# Ensemble Methods

Instructor: Hongfei Xue

Email: [hongfei.xue@charlotte.edu](mailto:hongfei.xue@charlotte.edu)

Class Meeting: Tue & Thu, 4:00 PM – 5:15 PM, WWH 130



Some content in the slides is based on Dr. Raquel Urtasun's lecture

# Mid-term Exam Info

- Date: Oct. 08, Mon. (Next Tuesday)
- Duration: 75 minutes (4:00 – 5:15 PM)
- Location: This classroom (WWH 130)
- Content Scope: all lecture material presented to date (Lecture 1 - 14)
- Multi-choice question only.
- **Close-Book Exam**
  - But you can bring **1 piece of A4-sized cheat sheet.**
  - Bring your **calculator!**
- Accommodation: Send you request to DS Test Center

# An alternative derivation of AdaBoost

- Just write down the right cost function and optimize each parameter to minimize it
  - ▶ **stagewise additive modeling** (Friedman et. al. 2000)
- At each step employ the exponential loss function for classifier  $m$

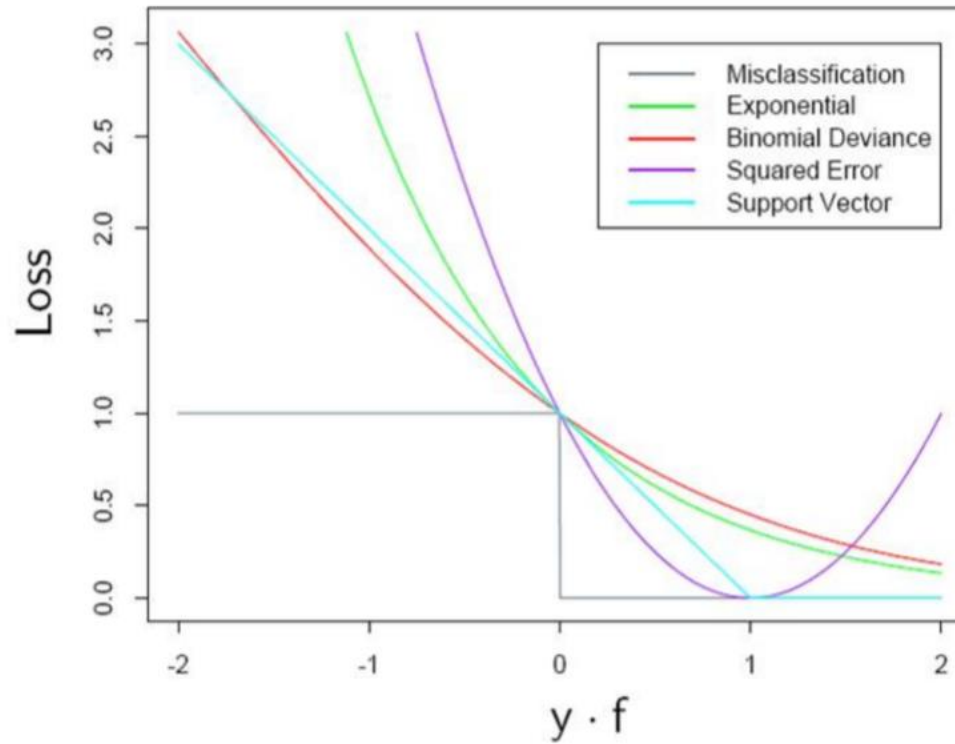
$$E = \sum_{n=1}^N \exp\{-t^{(n)} f_m(\mathbf{x}^{(n)})\}$$

- Real-valued prediction by committee of models up to  $m$

$$f_m(\mathbf{x}) = \frac{1}{2} \sum_{i=1}^m \alpha_i y_i(\mathbf{x})$$

- We want to minimize  $E$  w.r.t.  $\alpha_m$  and the parameters of the classifier  $y_m(\mathbf{x})$
- We do this in a sequential manner, one classifier at a time

# Loss Functions



- Misclassification: 0/1 loss
- Exponential loss:  $\exp(-t \cdot f(x))$  (AdaBoost)
- Squared error:  $(t - f(x))^2$
- Soft-margin support vector (hinge loss):  $\max(0, 1 - t \cdot y)$

# Learning classifier $m$ using exponential loss

- At iteration  $m$ , the energy is computed as

$$E = \sum_{n=1}^N \exp\{-t^{(n)} f_m(\mathbf{x}^{(n)})\}$$

with

$$f_m(\mathbf{x}) = \frac{1}{2} \sum_{i=1}^m \alpha_i y_i(\mathbf{x}) = \frac{1}{2} \alpha_m y_m(\mathbf{x}) + \frac{1}{2} \sum_{i=1}^{m-1} \alpha_i y_i(\mathbf{x})$$

- We can compute the part that is relevant for the  $m$ -th classifier

$$\begin{aligned} E_{\text{relevant}} &= \sum_{n=1}^N \exp \left( -t^{(n)} f_{m-1}(\mathbf{x}^{(n)}) - \frac{1}{2} t^{(n)} \alpha_m y_m(\mathbf{x}^{(n)}) \right) \\ &= \sum_{n=1}^N w_n^m \exp \left( -\frac{1}{2} t^{(n)} \alpha_m y_m(\mathbf{x}^{(n)}) \right) \end{aligned}$$

with  $w_n^m = \exp(-t^{(n)} f_{m-1}(\mathbf{x}^{(n)}))$

# Continuing the derivation

$$\begin{aligned} E_{\text{relevant}} &= \sum_{n=1}^N w_n^m \exp\left(-t^{(n)} \frac{\alpha_m}{2} y_m(\mathbf{x}^{(n)})\right) \\ &= e^{-\frac{\alpha_m}{2}} \sum_{\text{right}} w_n^m + e^{\frac{\alpha_m}{2}} \sum_{\text{wrong}} w_n^m \\ &= \underbrace{\left(e^{\frac{\alpha_m}{2}} - e^{-\frac{\alpha_m}{2}}\right)}_{\text{multiplicative constant}} \underbrace{\sum_n w_n^m [t^{(n)} \neq y_m(\mathbf{x}^{(n)})]}_{\text{wrong cases}} + e^{-\frac{\alpha_m}{2}} \underbrace{\sum_n w_n^m}_{\text{unmodifiable}} \end{aligned}$$

- The second term is constant w.r.t.  $y_m(\mathbf{x})$
- Thus we minimize the weighted number of wrong examples



# AdaBoost Algorithm

- Input:  $\{\mathbf{x}^{(n)}, t^{(n)}\}_{n=1}^N$ , and **WeakLearn**: learning procedure, produces classifier  $y(\mathbf{x})$
- Initialize example weights:  $w_n^m(\mathbf{x}) = 1/N$
- For  $m=1:M$

- ▶  $y_m(\mathbf{x}) = \text{WeakLearn}(\{\mathbf{x}\}, \mathbf{t}, \mathbf{w})$ , fit classifier by minimizing

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

- ▶ Compute unnormalized error rate

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

- ▶ Compute classifier coefficient  $\alpha_m = \log \frac{1-\epsilon_m}{\epsilon_m}$
- ▶ Update data weights

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

- Final model

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

# An impressive example of boosting

- Viola and Jones created a very fast face detector that can be scanned across a large image to find the faces.

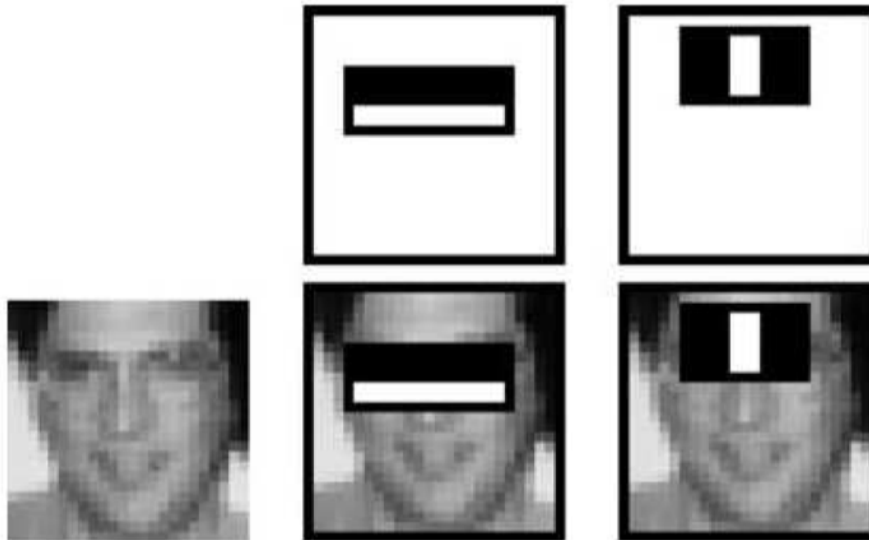


- The base classifier/weak learner just compares the total intensity in two rectangular pieces of the image.
  - ▶ There is a neat trick for computing the total intensity in a rectangle in a few operations.
    - ▶ So its easy to evaluate a huge number of base classifiers and they are very fast at runtime.
  - ▶ The algorithm adds classifiers greedily based on their quality on the weighted training cases.



# AdaBoost in Face Detection

- Famous application of boosting: detecting faces in images
- Two twists on standard algorithm
  - ▶ Pre-define weak classifiers, so optimization=selection
  - ▶ Change loss function for weak learners: false positives less costly than misses

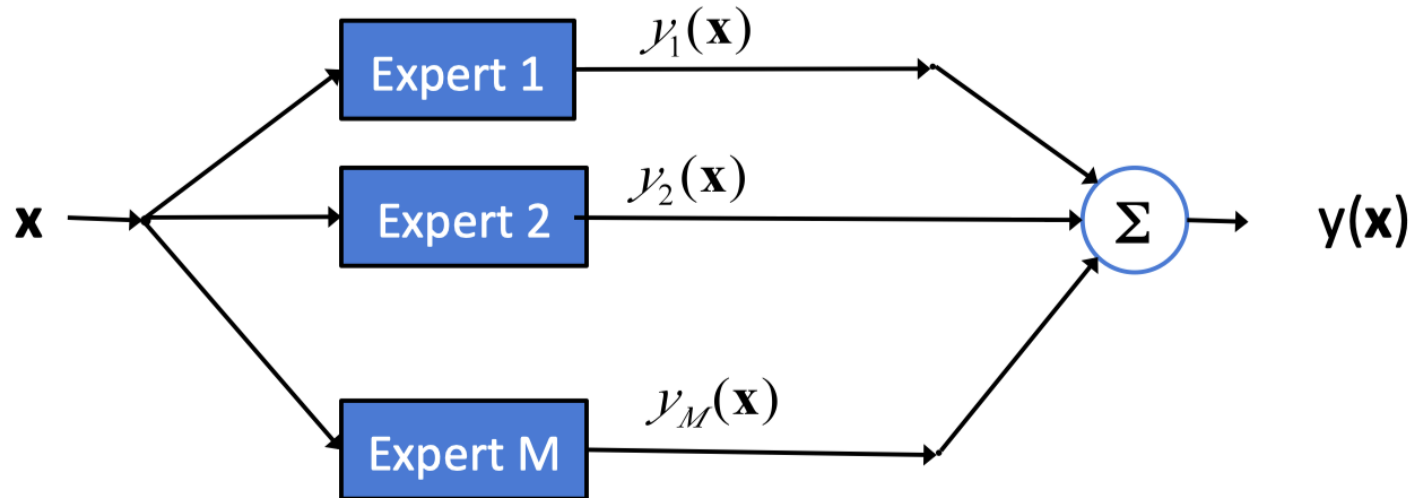


# AdaBoost Face Detection Results



# Ensemble Learning: Boosting and Bagging

- Experts cooperate to predict output

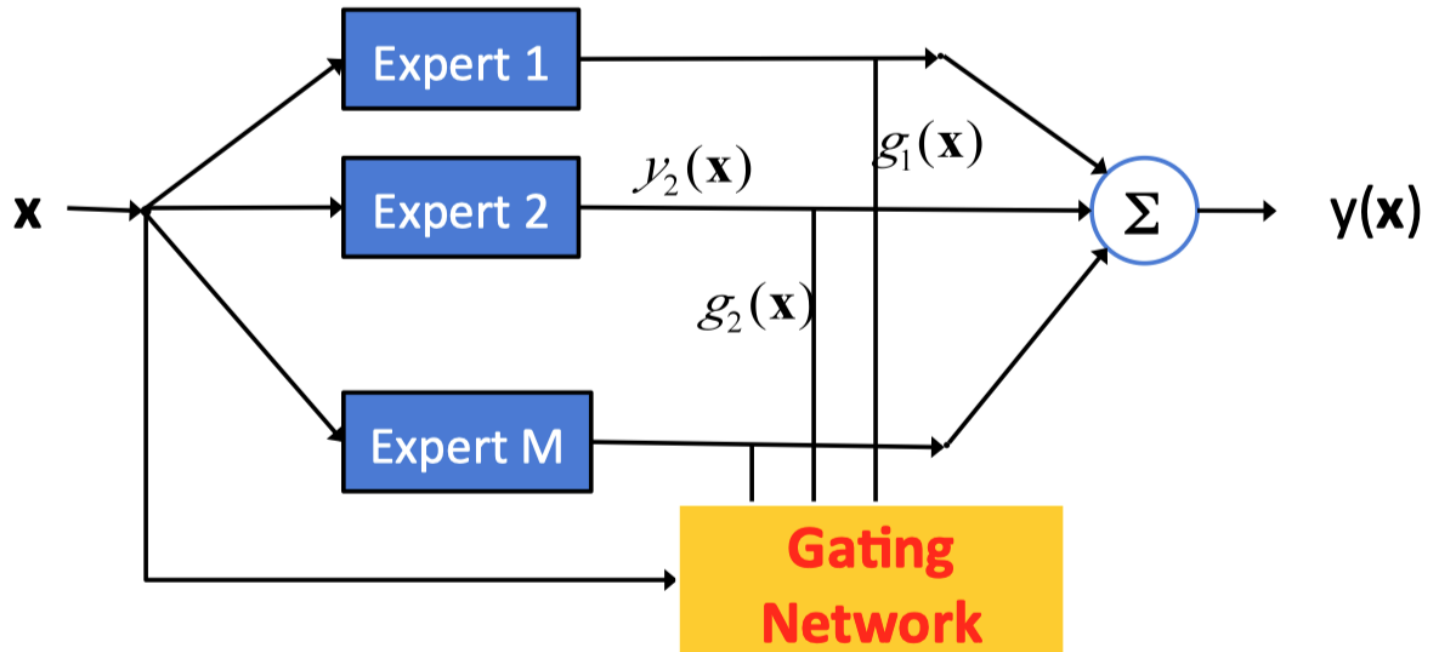


- Vote of each expert has consistent weight for each test example

$$y(\mathbf{x}) = \sum_m g_m y_m(\mathbf{x})$$

# Mixture of Experts

- Weight of each expert is not constant – depends on input  $\mathbf{x}$



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

$$y(\mathbf{x}) = \sum_m g_m(\mathbf{x}) y_m(\mathbf{x})$$

# 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

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

- This can overfit badly. It makes the model much more powerful than training each predictor separately
- Leads to odd objective: consider adding models/experts sequentially
  - ▶ if its estimate for  $t$  is too low, and the average of other models is too high, then model  $m$  encouraged to lower its prediction

# Cooperation vs. Specialization

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

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

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

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

- Gating output is softmax of  $z = U\mathbf{x}$

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

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



# Derivatives of Simple Cost Function

- Look at derivatives to see what cost function will do

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

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

$$\begin{aligned} \frac{\partial E}{\partial y_m} &= \frac{1}{M} g_m(\mathbf{x})(t - y_m(\mathbf{x})) \\ \frac{\partial E}{\partial z_m} &= \frac{1}{M} g_m(\mathbf{x}) [(t - y_m(\mathbf{x}))^2 - E] \end{aligned}$$

# Mixture of Experts: Final Cost Function

- Can improve cost function by allowing each expert to produce not just a single value estimate, but a distribution
- Result is a mixture of experts model:

$$p(t|MOE) = \sum_m g_m(\mathbf{x}) \mathcal{N}(t|y_m(\mathbf{x}), \Sigma)$$

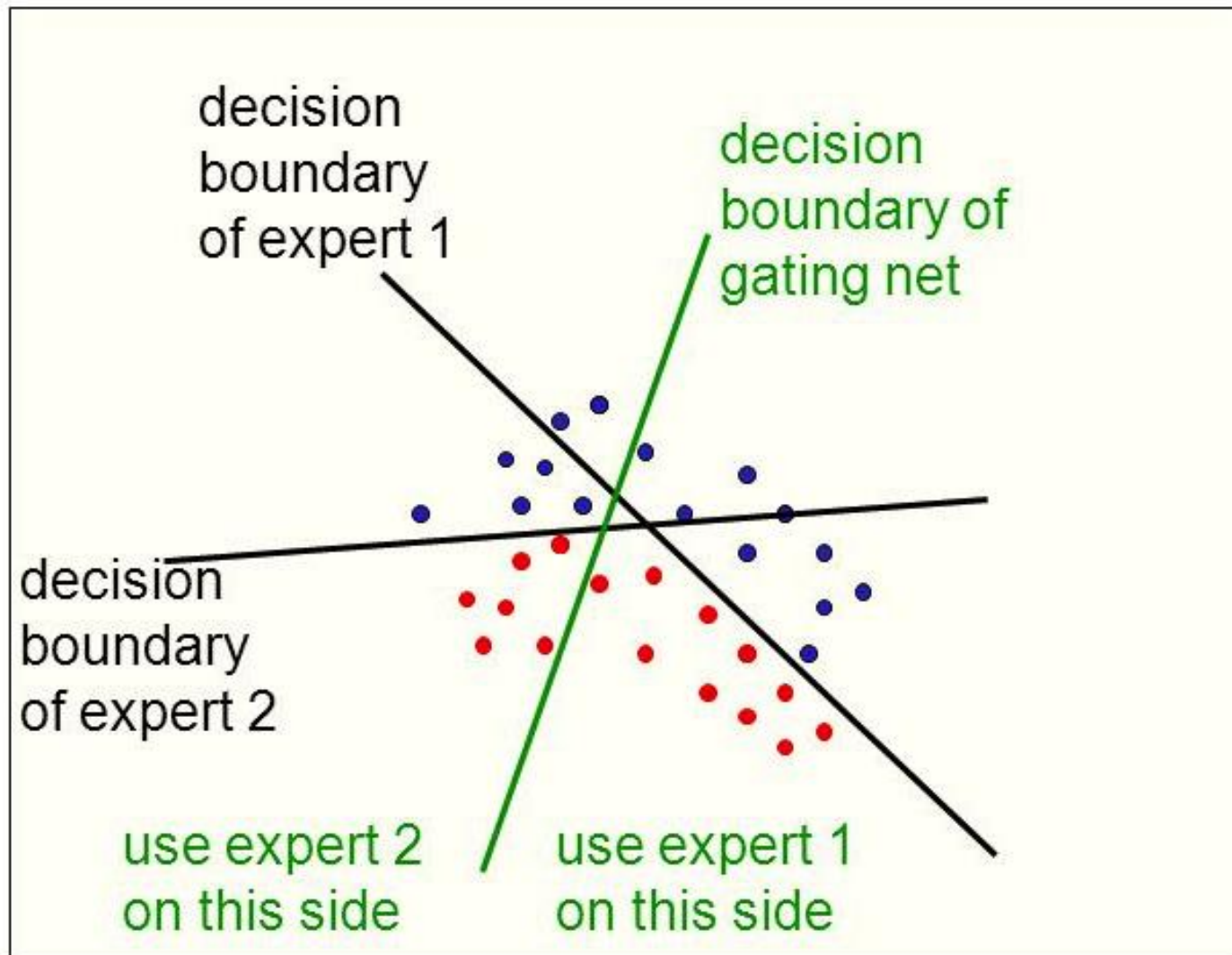
- Optimize minus log-likelihood:

$$-\log p(t|MOE) = -\log \sum_m g_m(\mathbf{x}) \exp \left( -\frac{1}{2} \|t - y_m(\mathbf{x})\|^2 \right)$$

- Gradient: Error weighted by posterior probability of the expert

$$\frac{\partial E}{\partial y_m} = -2 \frac{g_m(\mathbf{x}) \exp \left( -\frac{1}{2} \|t - y_m(\mathbf{x})\|^2 \right)}{\sum_i g_i(\mathbf{x}) \exp \left( -\frac{1}{2} \|t - y_i(\mathbf{x})\|^2 \right)} (t - y_m(\mathbf{x}))$$

# Mixture of Experts: Example



# Mixture of Experts: Summary

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

# 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

# Questions?