Ensembles

CSC411/2515: Machine Learning and Data Mining, Winter 2018

Michael Guerzhoy and Lisa Zhang

Slides based on those of Raquel Urtasun, Rich Zemel, Sanja Fidler

Netflix Challenge (2007-2009)

- Predict movie ratings better than Netflix
- Winning method combined 107 different models!

Netflix Challenge (2007-2009)

- “Our experience is that most efforts should be concentrated in deriving substantially different approaches, rather than refining a simple technique.”
- “We strongly believe that the success of an ensemble approach depends on the ability of its various predictors to expose different complementing aspects of the data. Experience shows that this is very different than optimizing the accuracy of each individual predictor.”

Ensembles

- **Ensemble of predictors** is a set of predictors whose individual decisions are **combined** in some way to classify new examples
- Simplest approach:
  - Generate multiple classifiers
  - Each vote on test instance
  - Take majority / average as prediction
- Predictors are different due to
  - Different sampling of training data
  - Randomized parameters within predictor algorithm
- Aim: mediocre algorithms ➔ super classifier

Ensembles: Overview

- Bagging:
  - Train separate models on overlapping training sets
  - Average the predictions
- Boosting:
  - Iteratively re-weight training examples to focus on harder examples
- Mixture of Experts
  - Parallel training with objective encouraging division of labour
- Plus Bayesian methods

Bias-Variance Decomposition

- Assume that $Y = f(X) + \epsilon$ with $\epsilon \sim N(0, \sigma^2)$
- Assume we have a regressor model $\hat{f}(X)$
- Break down the expected Mean Squared Error of $\hat{f}(X)$ into portions:
- $E\left[\left(Y - \hat{f}(X)\right)^2\right]$
  - $= E\left[\left(Y - f(x)\right)^2\right] + E\left[\left(f - \hat{f}(x)\right)^2\right]$
  - $= \sigma^2 + E\left[\left(f - E[\hat{f}(x)]\right)^2\right] + E\left[\left(E[\hat{f}(x)] - \hat{f}(x)\right)^2\right]$

Can prove this!

Irreducible Error

Bias^2

Variance

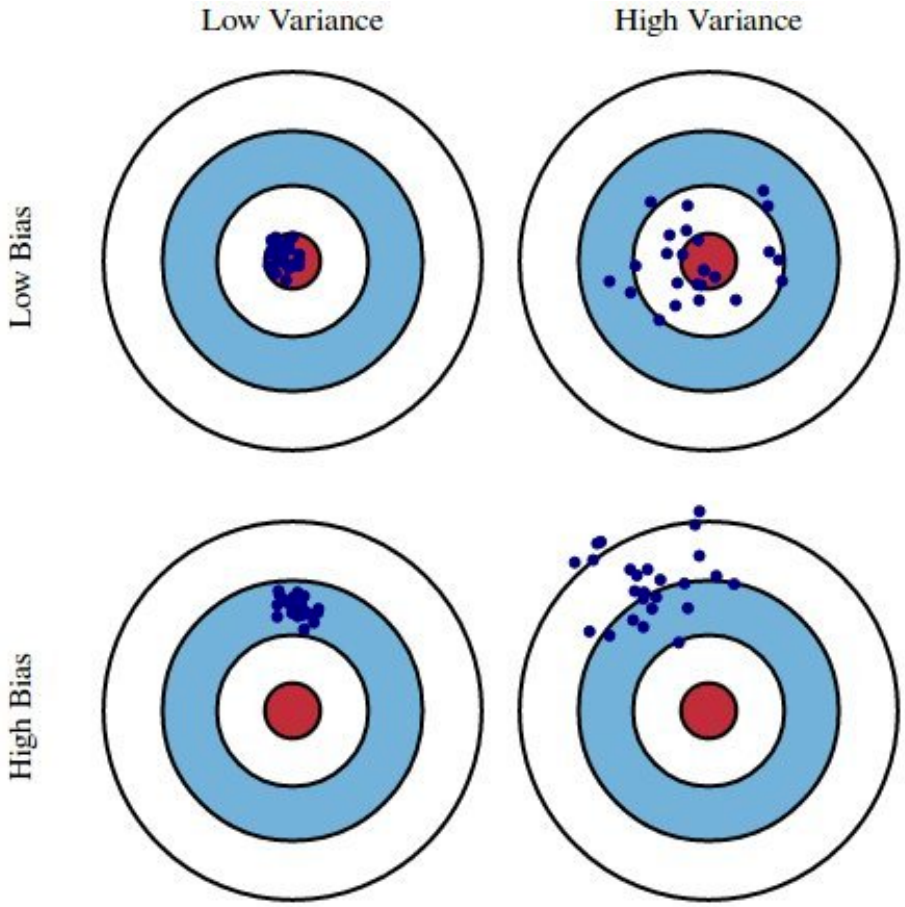
$$E\left[\left(Y-\hat{f}(X)\right)^2\right]$$

- The expectation is with respect to sampling data from the all the possible data
- Every time we have a new training set, we have a new $\hat{f}$
- $E\left[\left(Y-\hat{f}(X)\right)^2\right]$ is the average training error, when you keep trying new training sets

$$E\left[\left(Y-\hat{f}(X)\right)^2\right]=\sigma^2+E\left[\left(f(x)-E\left[\hat{f}(x)\right]\right)^2\right]+E\left[\left(E\left[\hat{f}(x)\right]-\hat{f}(x)\right)^2\right]$$

- $E\left[\left(Y-\hat{f}(X)\right)^2\right]$
  - The expected training error
- $E\left[f(x)-E\left[\hat{f}(x)\right]\right]^2$
  - Bias: the average difference between the actual f and the expectation of the prediction $\hat{f}$ that you'll get.
- $E\left[\left(E\left[\hat{f}(x)\right]-\hat{f}(x)\right)^2\right]$
  - Variance: how the average prediction differs from the prediction in a particular case. If the training set is all the possible training examples, $E\left[\hat{f}(x)\right]=\hat{f}(x)$ and the variance is zero

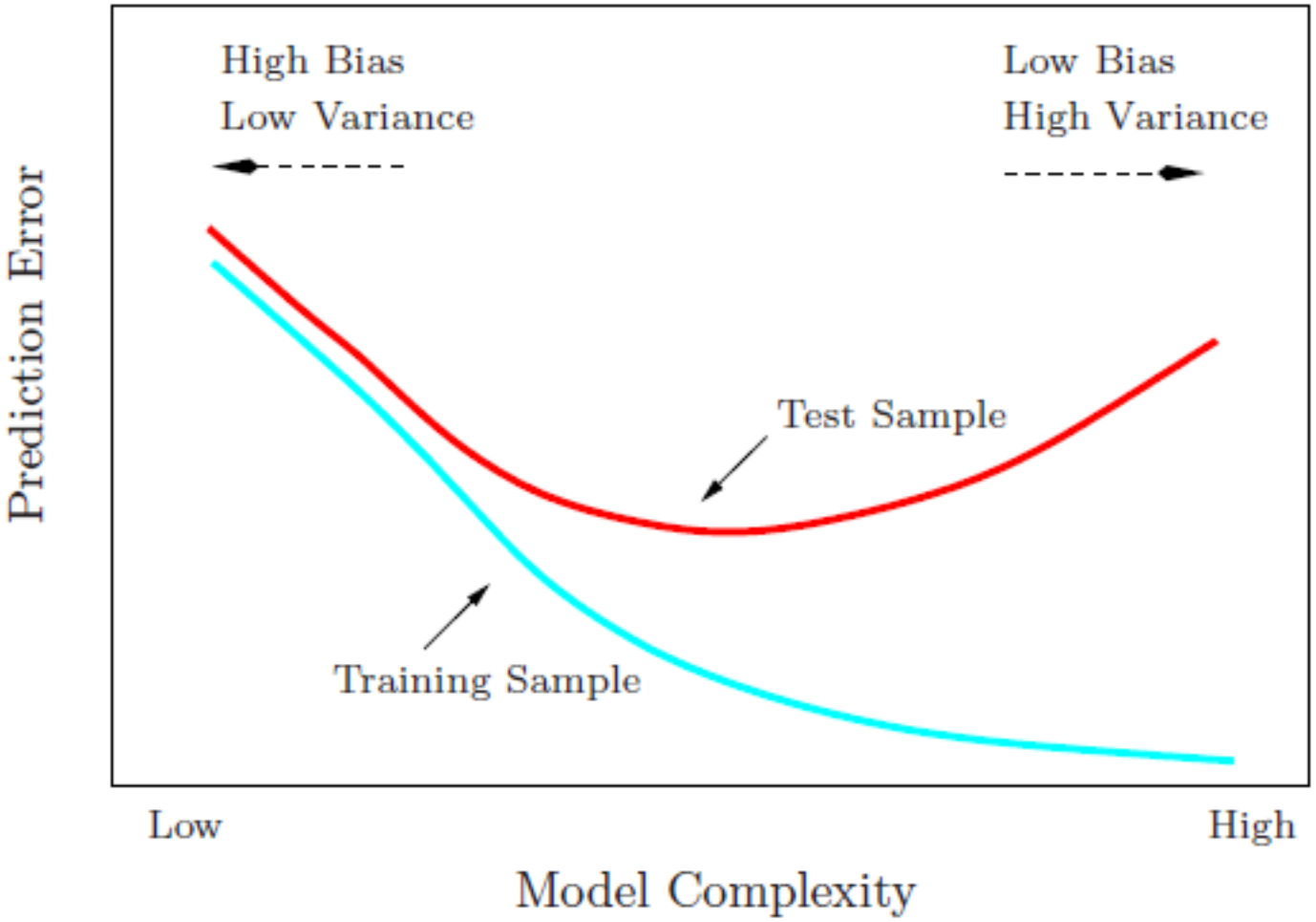
Bias-Variance Decomposition



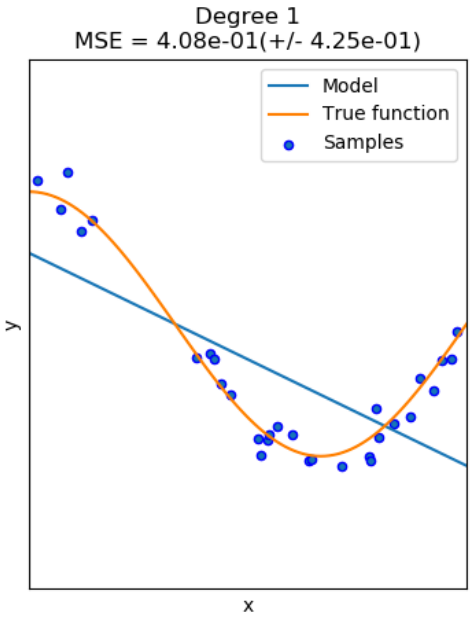
- **Bias**: Erroneous assumptions in learning algorithm
- **Variance**: Sensitivity to small changes in training data

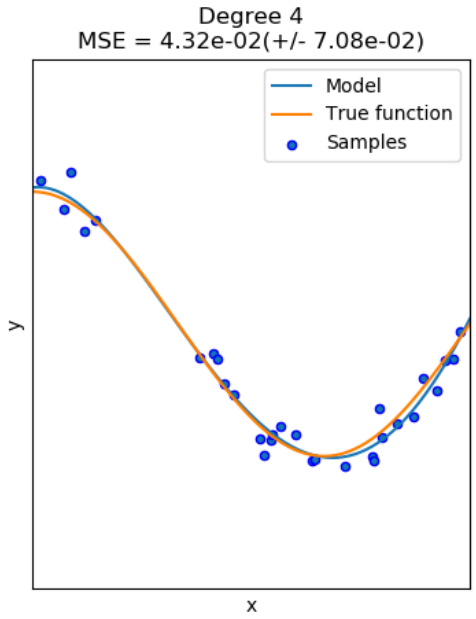
$$\sigma^2 + E\left[\left(Y - E\left[\hat{f}(x)\right]\right)^2\right] + E\left[\left(E\left[\hat{f}(x)\right] - \hat{f}(x)\right)^2\right]$$

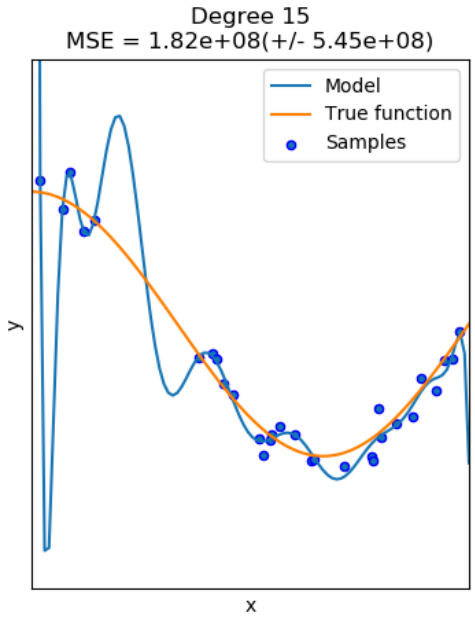
Bias-Variance Decomposition



High Bias vs High Variance







Example

- Linear regression typically have **high bias**.
- Decision trees (with many levels) have **low bias** and **high variance**.
- Any model that overfits easily has **high variance** (and hopefully low bias)

- One way to understand the variance of a predictor is to **sample different training datasets** and observe the empirical variance in prediction

Why do Ensemble Methods work?

- **Variance reduction**: if training sets are completely independent, it will always help to average an ensemble because this will reduce variance without affecting bias
- **Bias reduction**: average of models have greater capacity than a single model

Why do Ensemble Methods work?

- If each (independent) predictor has 30% chance of being wrong, averaging 61 predictors will give a very good classifier!

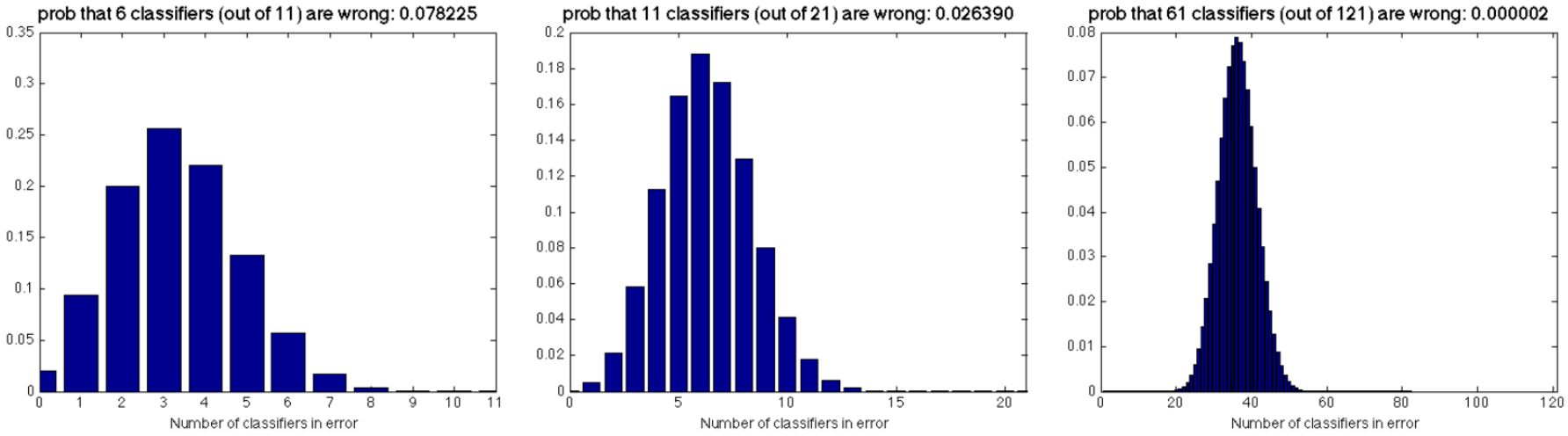


Figure: $\epsilon = 0.3$: (**left**) $N = 11$ classifiers, (**middle**) $N = 21$, (**right**) $N = 121$.

Bagging = **B**ootstrap **Ag**gregation

For each k=1..K:

- Sample (with replacement) *n* data points from the the training set D
- Train a model f_k using the *n* data points

At test time, use the average of the individual predictions of each f_k

- For regression: avg predictions
- For classification: avg class probabilities (or vote)

Bagging Example

Works well for any high variance classifiers.

Example: **Decision Trees**

- Decision trees overfit easily
- Fast to perform inference
- So, train lots of them!

➔ **Random Forest**

Random Forest

RandomForest = Decision Trees

+ Bagging

+ One more trick

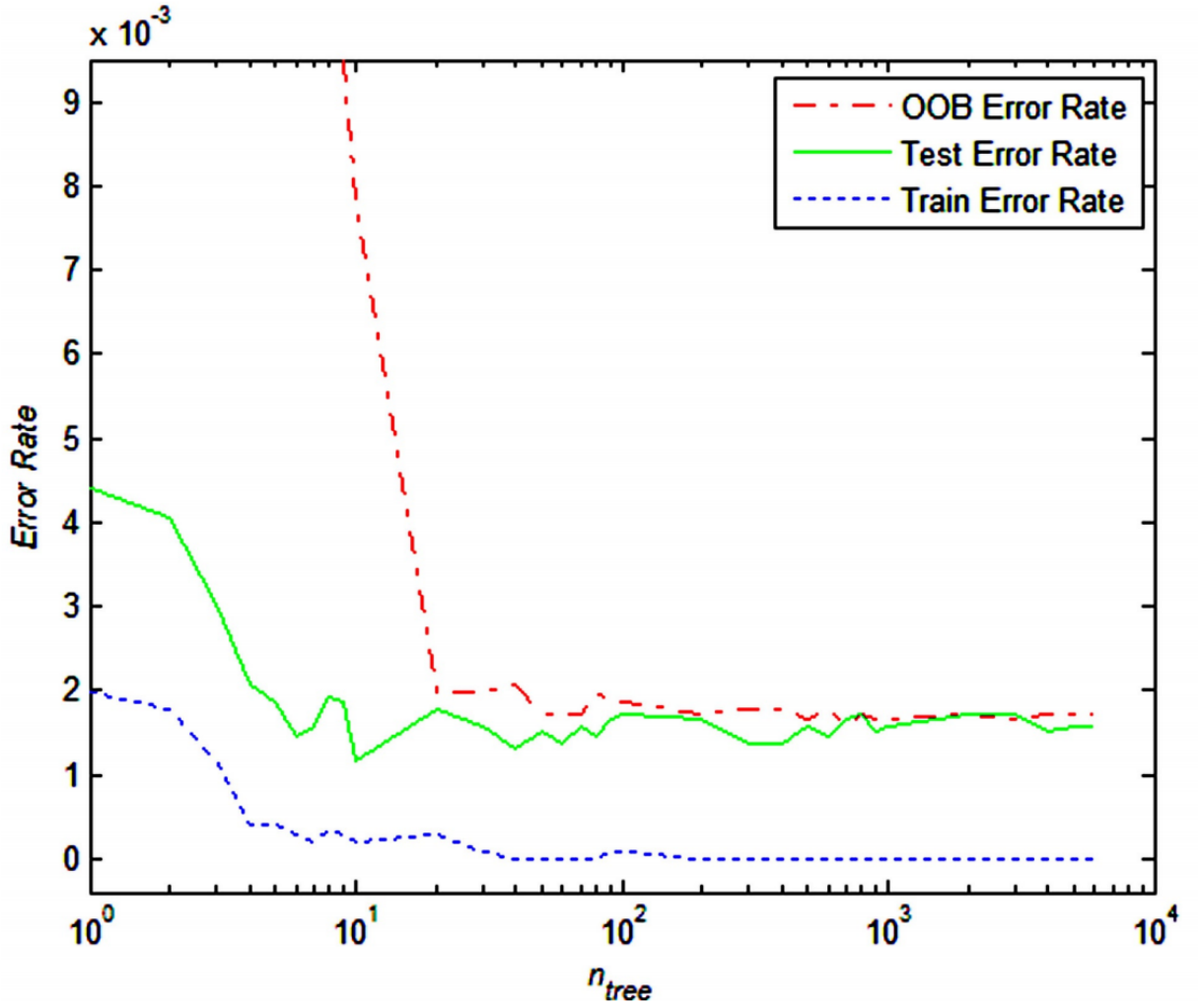
To reduce correlation, each split only considers a random subset of features (instead of all).

This help produce less correlated trees.

Out-of-bag estimation

- In bagging there is a nice way to cheaply estimate the test loss
- Each training example only appear in some of the "bagged" trees
- OOB estimation: we predict each training example using all the trees that **did not** contain it in their training data

Out-of-bag estimation



OOB Error is a good prediction of the test error

Aside: N-fold Cross-Validation

- Used when the available data set is small, so a train/valid/test split is not feasible
- Split the training data into N portions
- For each k=1..N:
  - Use all but the kth portion to train a predictor
  - Test the predictor on the kth portion
- Goal: estimate test accuracy for the predictor
- Then train a new predictor with the entire dataset to use at test time

Aside: Leave-one-out

- Same as cross-validation, but the size of each portion is 1 data point

- cross-validation and leave-one-out requires building many of the same classifier, which can be expensive.

Aside: Parametric vs Non-parametric models

- Parametric models make assumptions about distributions:
  - Naïve Bayes, GDA, Linear Regression, Logistic Regression…
  - They have a fixed set of parameters that are independent of the size of training set
- Non-parametric models do not:
  - kNN, Decision Trees, non-parametric
  - As size of training set increase, the number of parameters increases

Bagging: Summary

- Reduces variance (overfitting) by averaging predictions
- Even if a single model is great, a small ensemble usually helps
- Easy to parallelize
- Limitations:
  - Does not reduce bias
  - Correlation between classifiers

Boosting

- Suppose you have a weak learning module (a base classifier) that can always get slightly better than 50% correct on binary-classification
- Can you apply this learning module many times to get a strong learner?
- **YES!**

- Key idea: make each classifier focus on previous mistakes

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 at test time
  - How do we weight the models in the committee?

ADABoost: Single Classifier

- Input: x, Target: y = 1 or -1, Classifier: f(x) = 1 or -1
- Weight on training example i for classifier m: $w_m^{(i)}$
- Cost function for classifier m:

$$J_m = \sum_i w_m^{(i)} [f_m(x^{(i)}) \neq y^{(i)}]$$

- Weighted error rate of classifier:

$$\epsilon_m = \frac{J_m}{\sum_i w_m^{(i)}}$$

ADABoost

- Weighted error rate of classifier:

$$\epsilon_m = \frac{J_m}{\sum_i w_m^{(i)}}$$

- Quality of the classifier:

$$\alpha_m = \frac{1}{2}\ln\left(\frac{1-\epsilon_m}{\epsilon_m}\right)$$

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

ADABoost

- Weights for next round:

$$w_{m+1}^{(i)} = w_m^{(i)} \exp\{\alpha_m [f_m(x^{(i)}) \neq y^{(i)}]\}$$

To perform inference:

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

$$\mathrm{f(x)} = \mathrm{sign}\left(\sum_m \alpha_m f_m(x)\right)$$

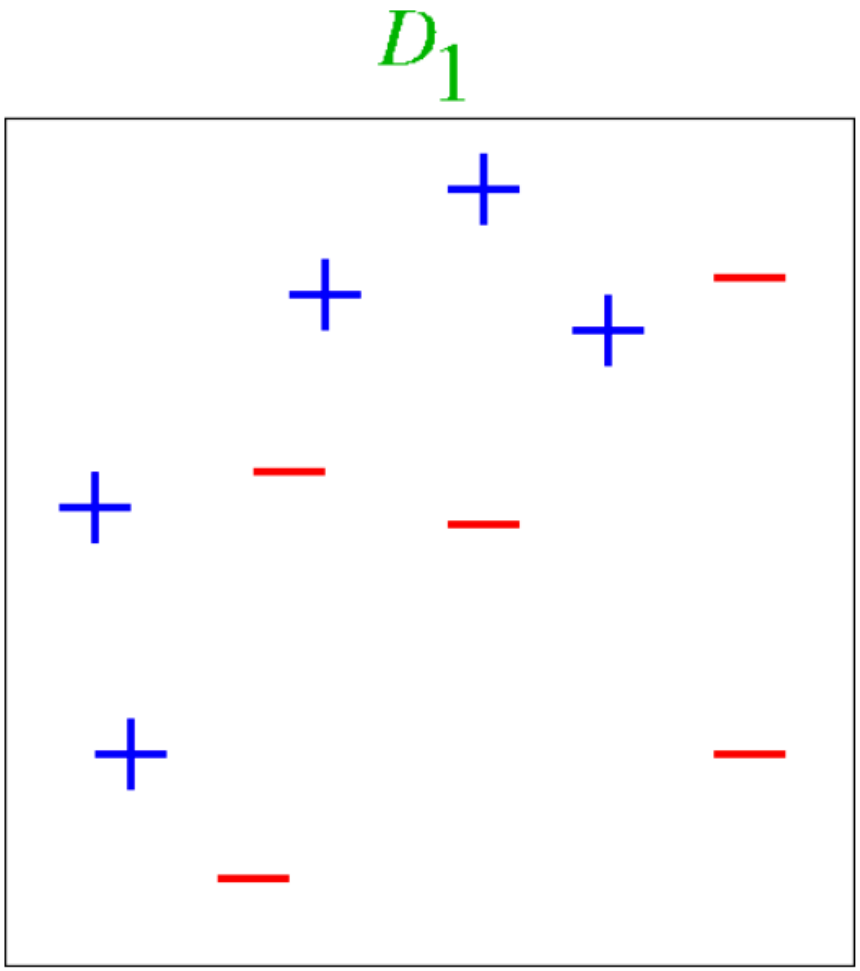
ADABoost: Summary

- Initialize example weights $w_m^{(i)} = \frac{1}{N}$
- For m = 1 to M
  - Fit classifier: min $J_m = \sum_i w_m^{(i)} [f_m(x^{(i)}) \neq y^{(i)}]$
  - Compute error: $\epsilon_m = \frac{J_m}{\sum_i w_m^{(i)}}$
  - Compute classifier coefficient: $\alpha_m = \frac{1}{2} \ln\left(\frac{1-\epsilon_m}{\epsilon_m}\right)$
  - Update data weights:

$$w_{m+1}^{(i)} = w_m^{(i)} \exp\{\alpha_m [f_m(x^{(i)}) \neq y^{(i)}]\}$$

- Final model: $\mathrm{f(x)} = \mathrm{sign}(\sum_m \alpha_m f_m(x))$

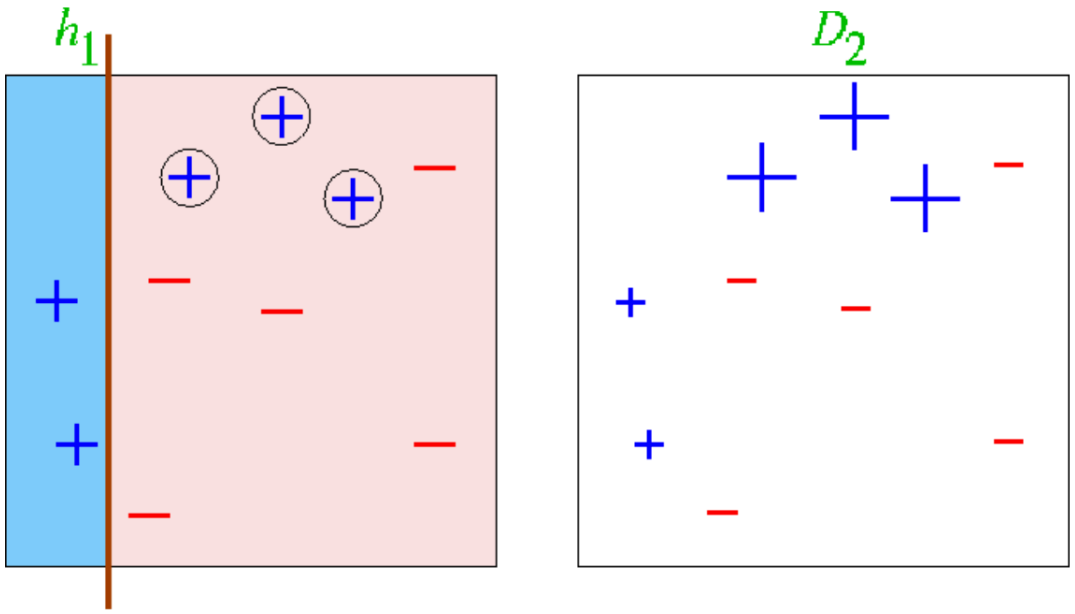
ADABoost: Example



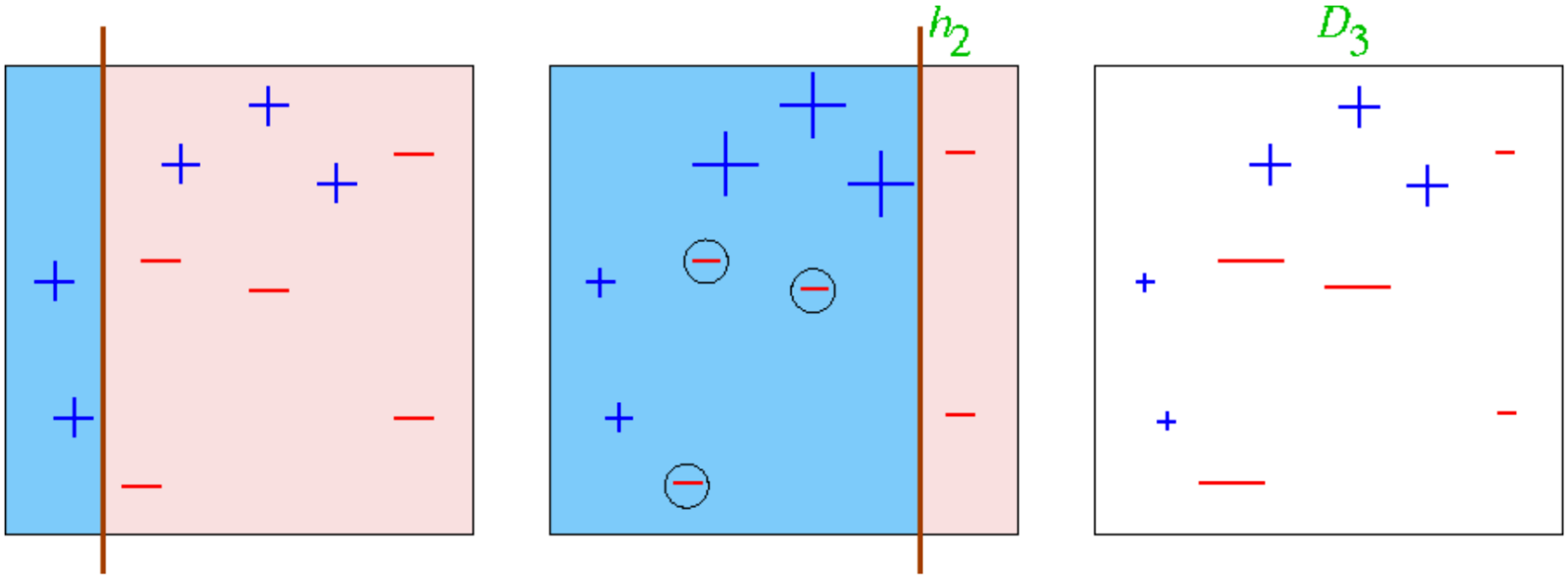
Weak Classifier:
a stump (a tree with just one split)

[Slide credit: Verma & Thrun]

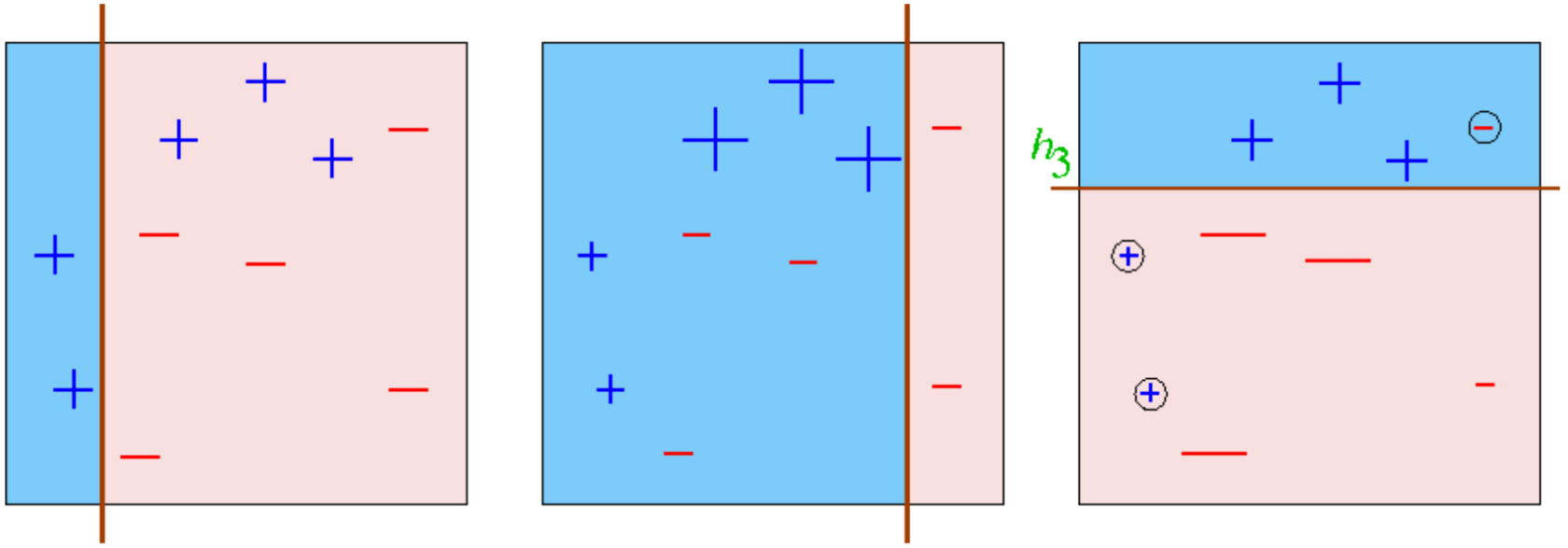
ADABoost: Example



ADABoost: Example



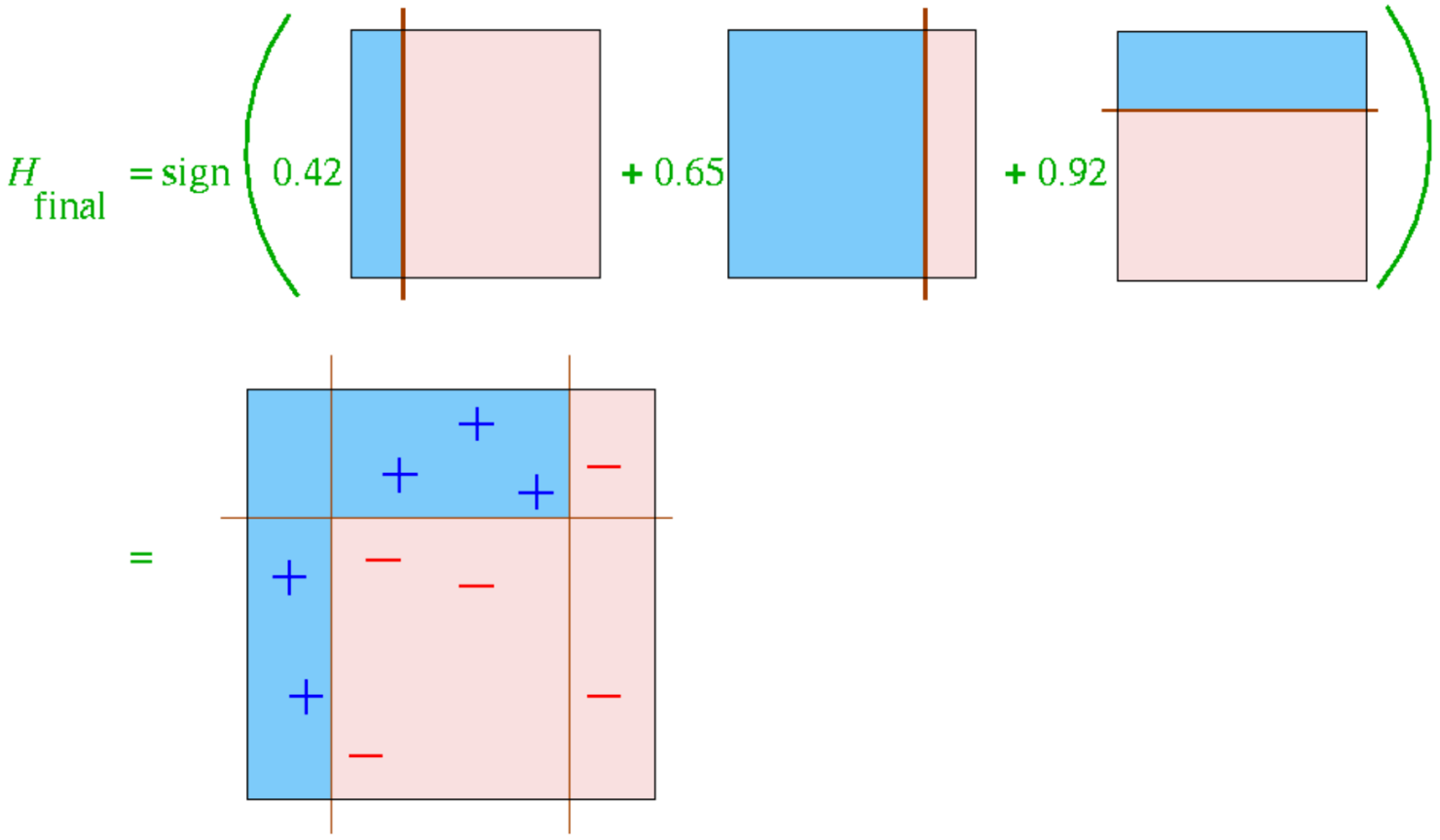
ADABoost: Example



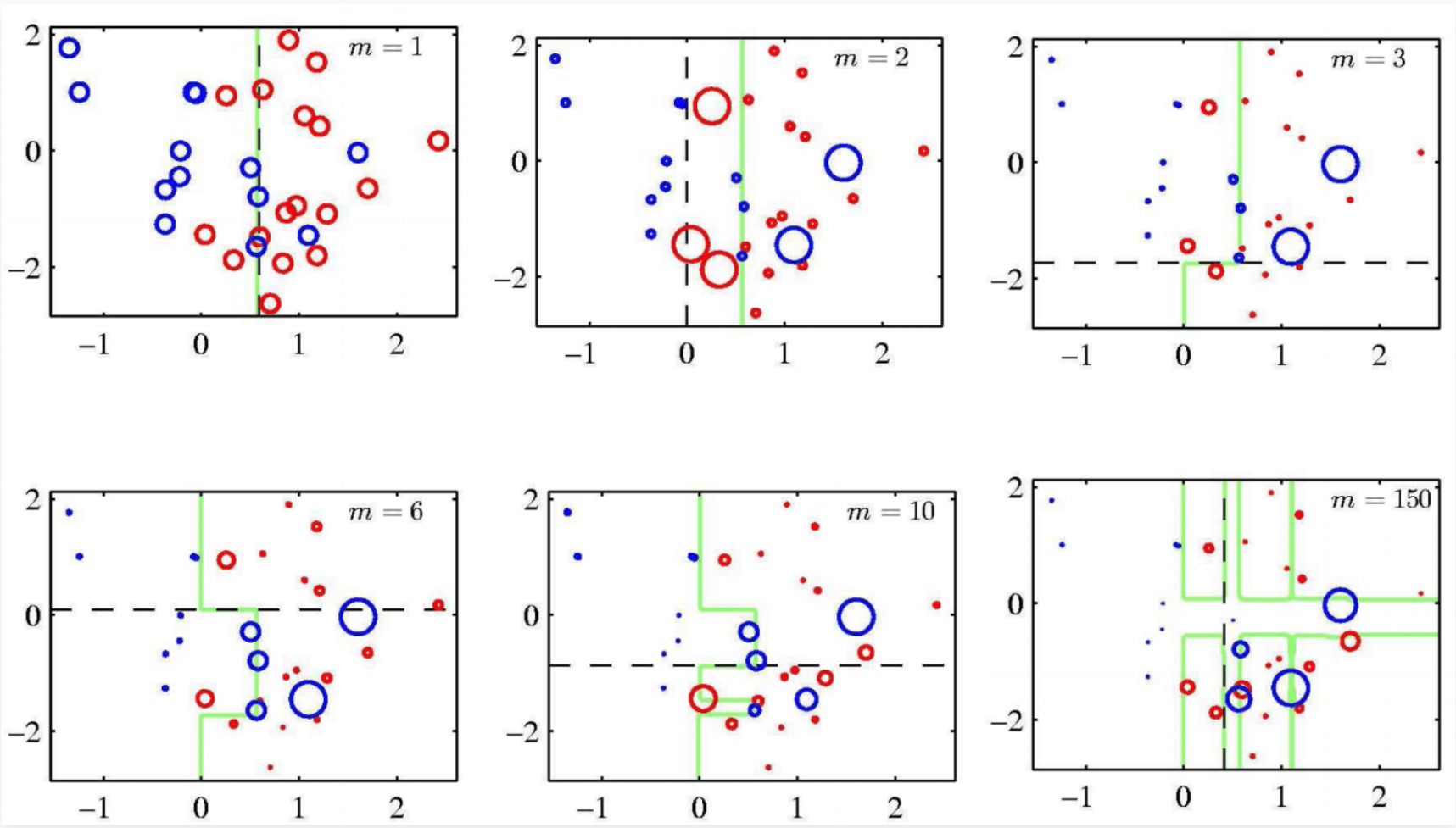
$\varepsilon_3 = 0.14$

$\alpha_3 = 0.92$

ADABoost: Example



ADABoost: Example

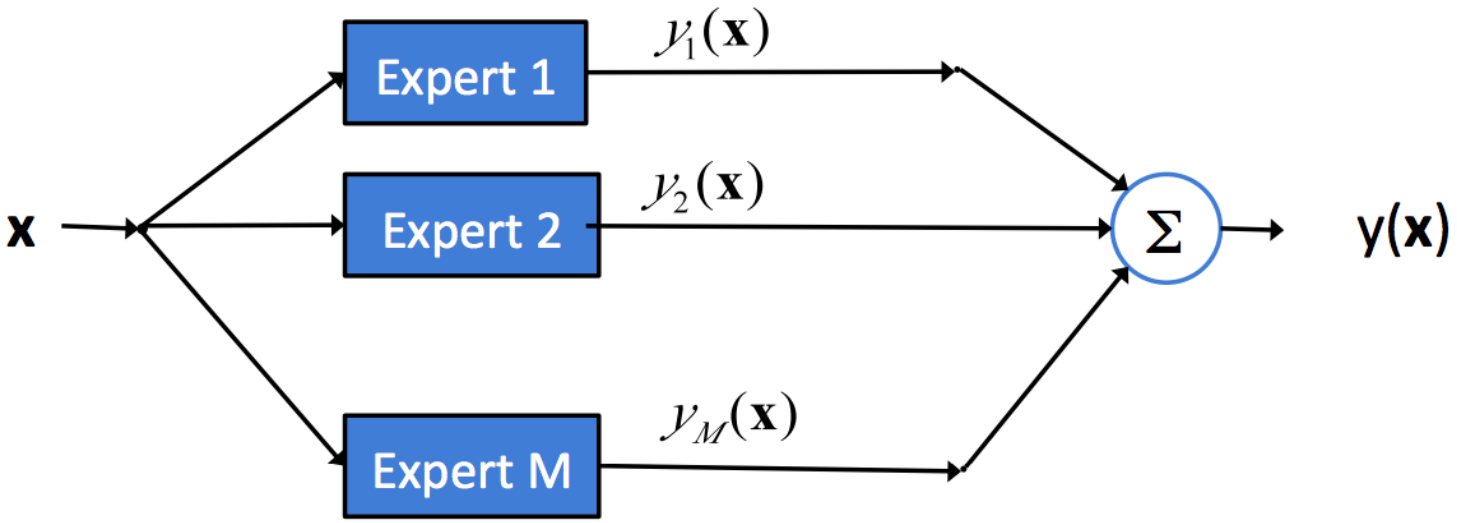


Boosting: Summary

- Reduces bias by iteratively re-weighting more difficult data points
- Even weak classifiers can be used to build strong classifiers

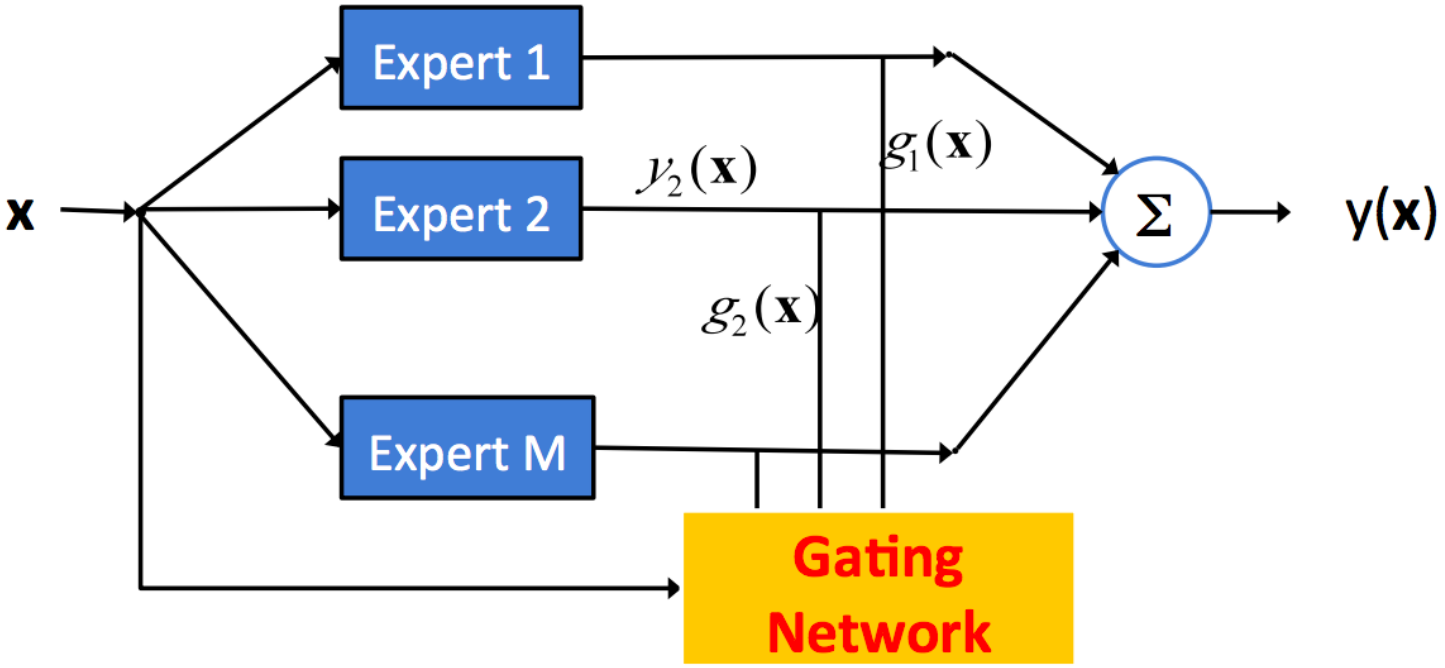
Mixture of Experts

Idea: have experts cooperate to predict output.



Mixture of Experts

But, add gating network so that weight of each expert depends on input x



Mixture of Experts

- This encourages specialization (local experts) instead of cooperation
- Each expert does well on a **subset** of data
- Gating network softmax over experts: stochastic selection of who is the true expert for a given input

- Used to be fairly popular, but not talked about as much any more.

Bayesian Methods

Recall estimating $P_\theta(cancer|t)$ from Generative Classifiers slides:

- $P(cancer) \approx \frac{count(cancer)}{N}$
- $P(t|cancer) = N(t|\mu_{cancer}, \sigma^2_{cancer})$
- $P(t|\neg cancer) = N(t|\mu_{\neg cancer}, \sigma^2_{\neg cancer})$
  - $\theta = \{\mu_{cancer}, \mu_{\neg cancer}, \sigma_{cancer}, \sigma_{\neg cancer}, \dots\}$
- Maximum Likelihood:
  - $argmax_{\mu_c, \sigma_c^2} \prod_i N(t^{(i)}|\mu_c, \sigma_c^2), i \in cancer$
  - Solution (show with calculus!):
    - $\widehat{\mu_c} = \overline{t^{(i)}}, i \in cancer$
    - $\widehat{\sigma_c^2} = \sum_i \frac{\left(t^{(i)} - \widehat{\mu_c}\right)^2}{\#ncancer}, i \in cancer$
- **Slide #13**: What is $P(cancer|t, D)$? It is *not* $P_{\theta_{MAP}}(cancer|t, D)!$

t = test data point

D = training data

Bayesian Method

- **Ensembles**: we don't have to choose one θ, we can learn several θ and weight them
- **Bayesian Method**: why not choose all possible θ and weight them? using a distribution?
  - Treat the parameters as random variables

- **Cancer prediction example:**

$$\mathrm{P}(\mathrm{cancer}|\mathrm{t},\mathrm{D}) = \int_{\theta} P(cancer|t, D, \theta)\,\mathrm{P}(\theta|D)d\theta$$

Bayesian Method

- **Cancer prediction example:**

$$\mathrm{P}(\mathrm{cancer}|\mathrm{t},\mathrm{D})$$

$$= \int_\theta P(cancer|t,D,\theta)\,\mathrm{P}(\theta|D)d\theta$$

$$= \int_\theta P_\theta(cancer|t,D)\,\mathrm{P}(\theta|D)d\theta$$

$$= \int_\theta \frac{P_\theta(t|cancer)P_\theta(cancer)}{P_\theta(t|cancer)P_\theta(cancer)+P_\theta(t|\neg cancer)P_\theta(\neg cancer)}\mathrm{P}(\theta|D)d\theta$$

- **Recall also that** $P(\theta|D) = \frac{P(D|\theta)P(\theta)}{P(D)}$

Bayesian Linear Regression

Prediction for new test data point x, training data D, weights w:

$$\mathrm{P}(\mathrm{y}|\mathrm{x},\mathrm{D}) = \int_w P(y|x,w,D)\,\mathrm{P}(w|x,D)d\theta = \int_w P(y|x,w)\,\mathrm{P}(w|x)d\theta = \int_w P(y|x,w)\frac{\mathrm{P}(x|w)P(w)}{P(x)}d\theta$$

This gives us a distribution over predicted values y

If model is “certain” about its predictions, the distribution will be narrow. If the model is “uncertain” the distribution can be wide.

Bayesian Linear Regression

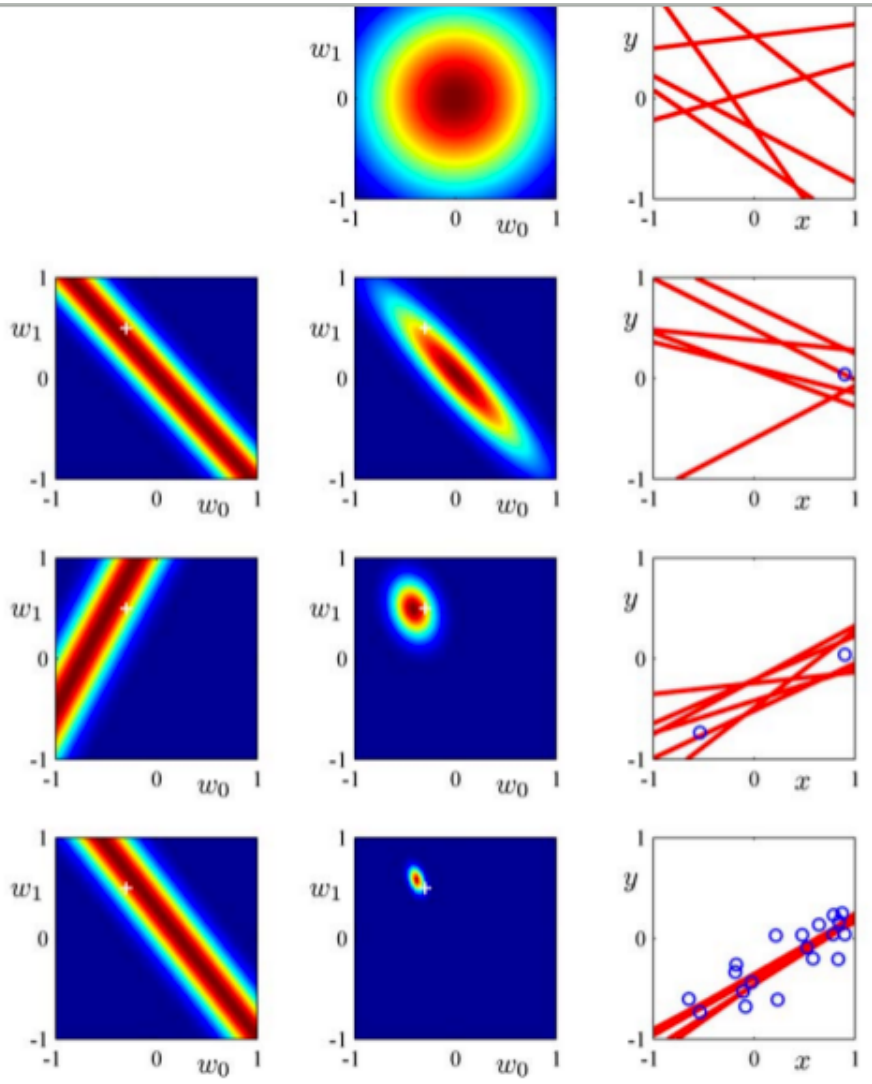
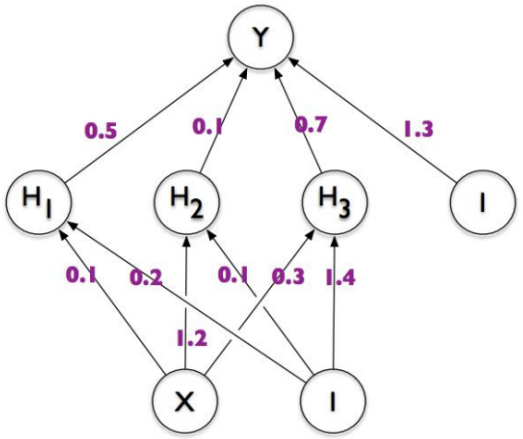


Figure 3.7 Illustration of sequential Bayesian learning for a simple linear model of the form $y(x, \mathbf{w}) = w_0 + w_1 x$. A detailed description of this figure is given in the text.

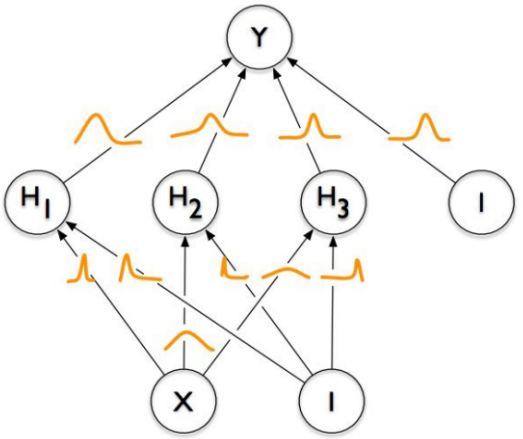
Bayesian Neural Networks

Standard Neural Net



- Parameters represented by *single, fixed values (point estimates)*
- Conventional approaches to training NNs can be interpreted as *approximations* to the full Bayesian method (equivalent to MLE or MAP estimation)

Bayesian Neural Net



- Parameters represented by *distributions*
- Introduce a prior distribution on the weights $P(\mathbf{w})$ and obtain the posterior $P(\mathbf{w} \mid \mathcal{D})$ through Bayesian learning
- Regularization arises naturally through the prior $P(\mathbf{w})$
- Enables principled model comparison

Slide credit: https://csc2541-f17.github.io/slides/lec03.pdf

Bayesian Methods Summary

- Maximum Likelihood is about optimization
- Bayesian parameter estimation is about integration
- Bayesian solution converges to maximum likelihood as we observe more data