Ensembles

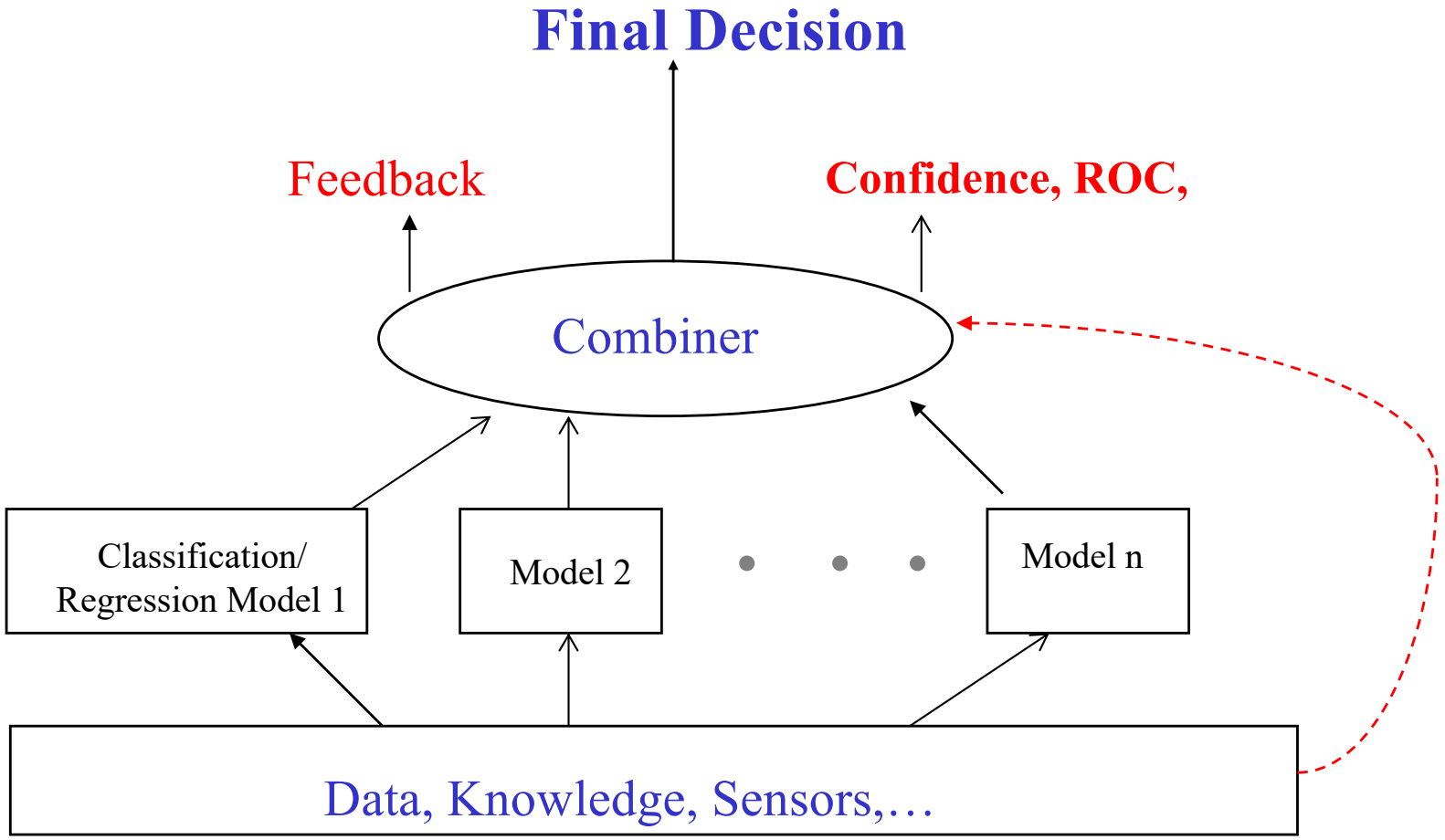
- Goal
  - How to use multiple "learners" to solve (parts of) the same problem
    - Regression/ Function approximation
    - Classification
  - The AMAZING benefits of Ensembles

  Readings: **HTF Ch 8.7, 10.1-10.11** (EA Ch 18, AG Ch 7, *KM ch 18)

- Competing learners: ENSEMBLES
  - Multiple looks at same problem
  - Random Forests, XGBoost,…

  Vs.
- Cooperative learners:
  - Divide and conquer
  - Mixture of experts; modular networks

Generic Multi-learner System



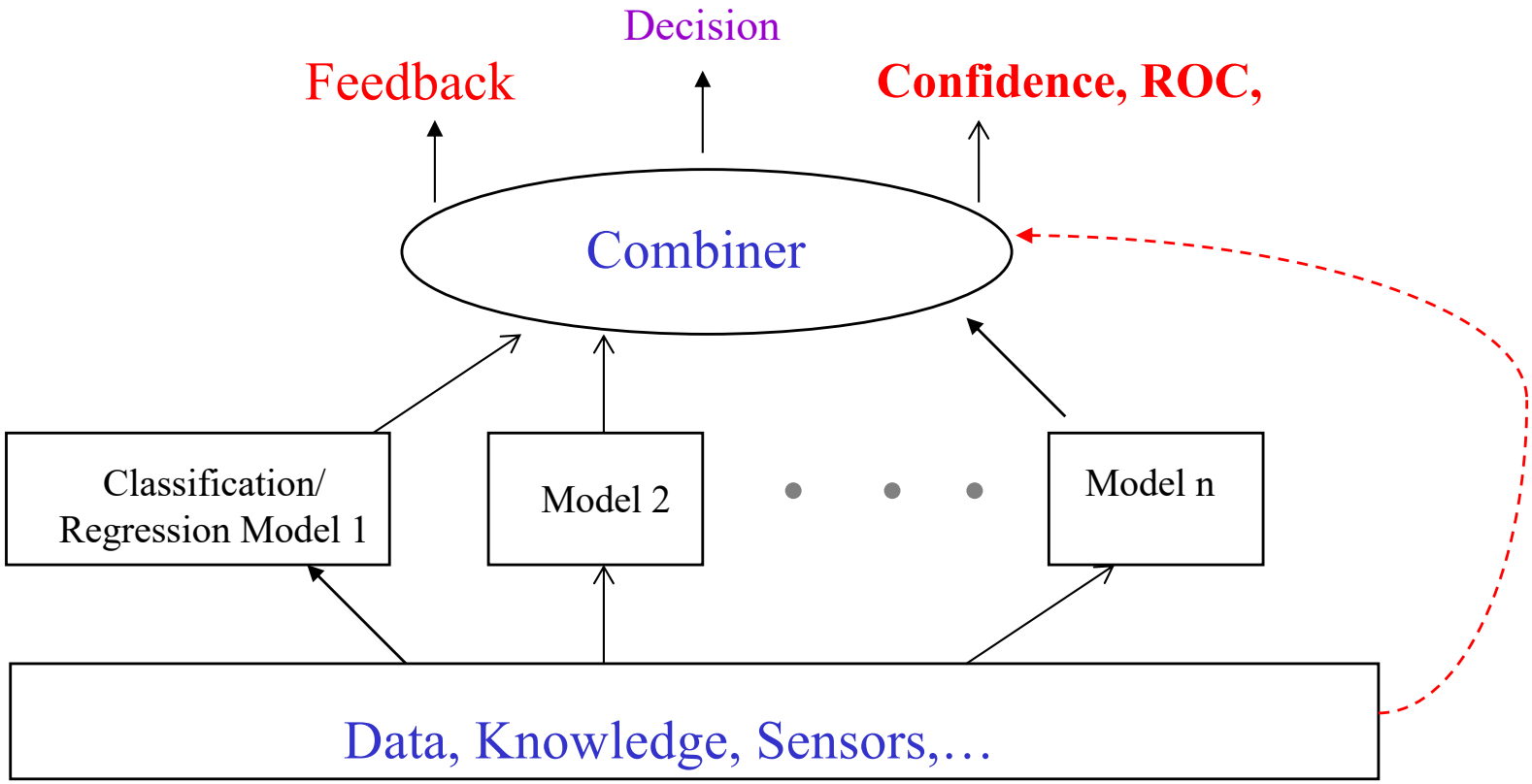
Motivation: Ensembles

- Different learners have different "inductive bias"
  - generalize differently from same training set
- Different properties
  - local vs. global
  - computation time / memory
  - susceptibility to outliers
- In designing/choosing one learner, several get created anyway!

- Hopes: better accuracy and better reliability

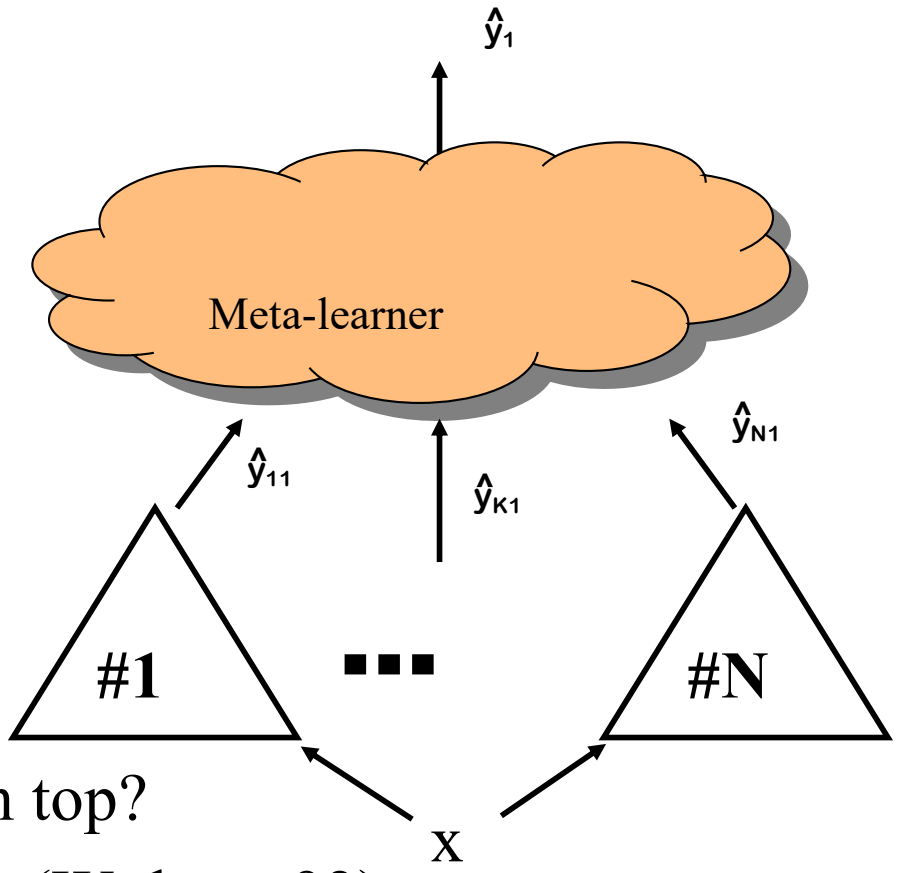
Common Issues

- Designing meta-learner or combiner
- Selecting the experts / learners / classifiers
- Determining training sets (same? automatic decomposition?)
- Estimating the gains, reliability



Output Combining for Function Approximation

- Meta-learners (trained)



- 1) Put a linear regressor or MLP on top?
- 2) Stacking :Stacked Generalization (Wolpert 92)
  - train second layer model on leave-one-out samples

Problems?

Gains from the Averaging Combiner

- **Simple combination: Just Average!**

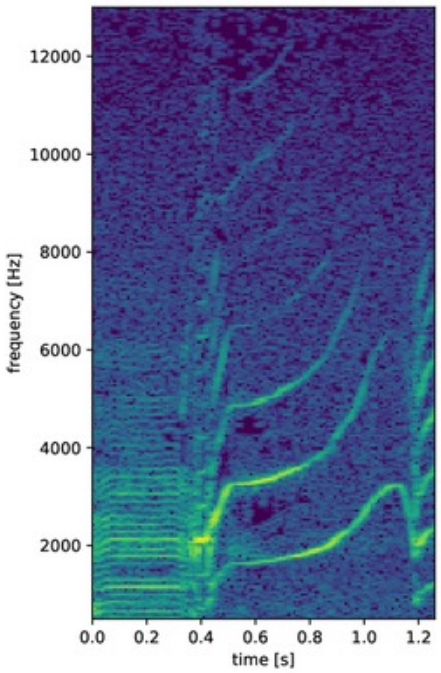
Analysis of Ensemble based on Simple Averaging:

- MSE of the "averaging" ensemble =
  average value of MSE over models in the ensemble – average value of ambiguity of the individual models.

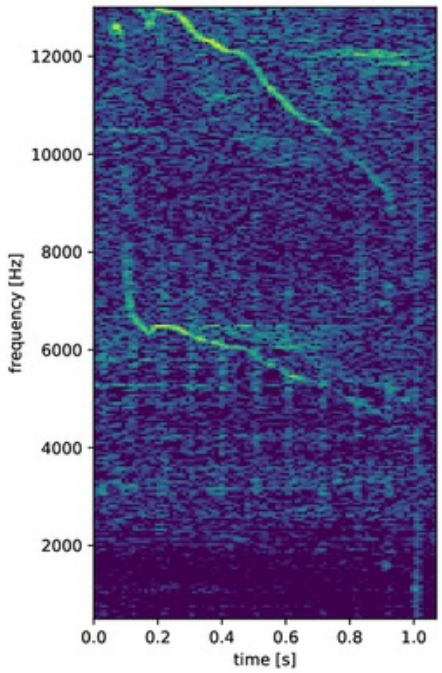
- Ambiguity= variance among the responses of different models for the same "x".

- So, need **good but diverse models**
  - rather overtrain than undertrain !!

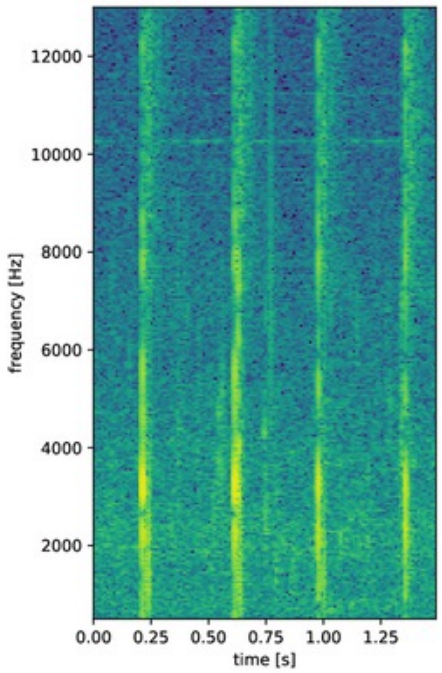
DARPA Sonar Challenge 1989-91



a) Killer whale pulsed call



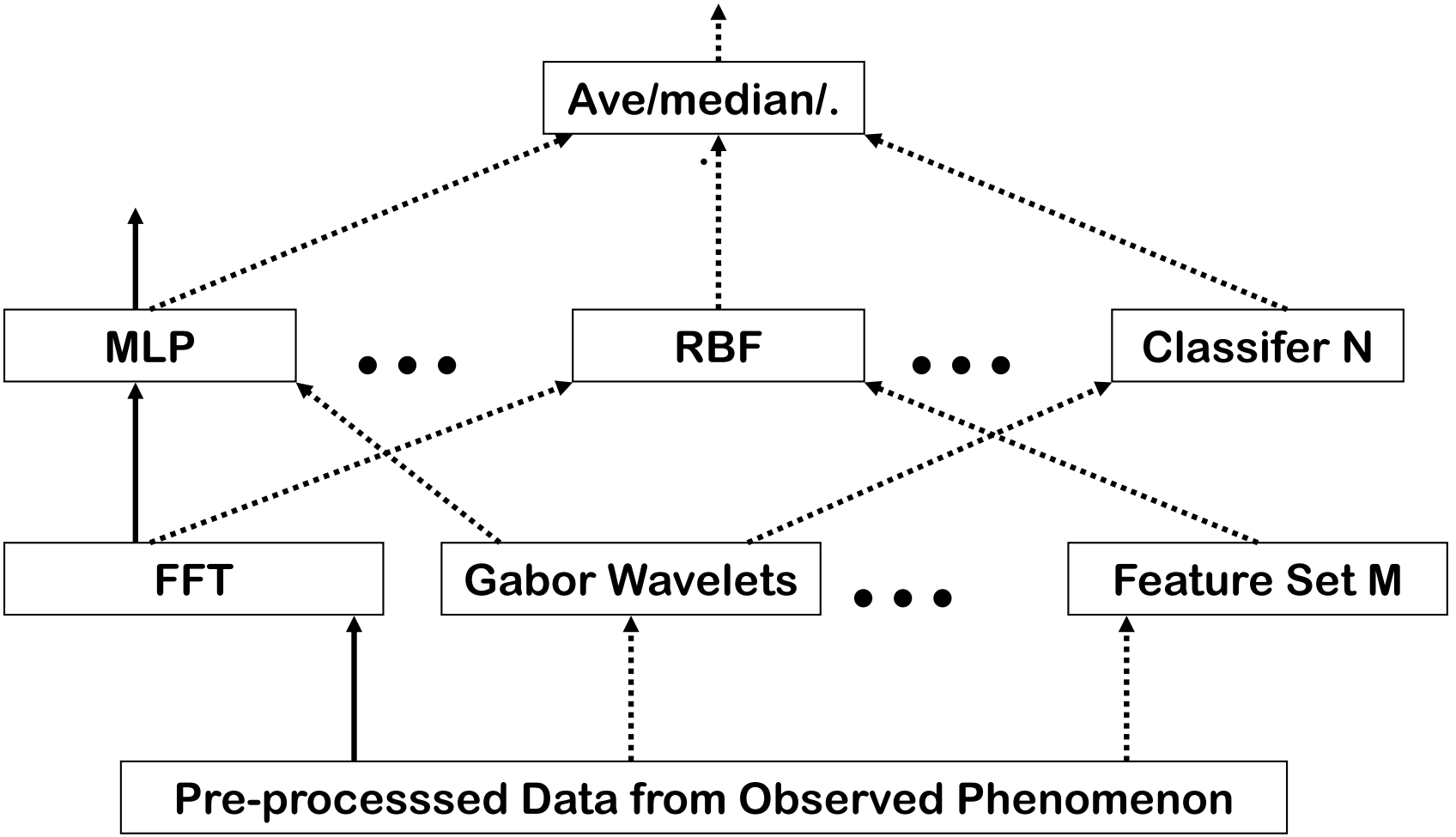
b) Killer whale whistle



c) Killer whale echolocation click

DARPA Sonar Transients Classification Program (1989-)

J. Ghosh, S. Beck and L. Deuser, *IEEE Jl. of Ocean Engineering*, Vol 17, No. 4, October 1992, pp. 351-363.



Selecting the Inputs

- Bagging (Breiman, 92)
  - Gains where sensitivity is high, i.e. base models are less stable
- Boosting (Schapire, 90)
  - Spl. Case of Adaptive Reweighting and Combining (ARC) methods

  - ***Read Ch 10 of HTF** for
    - Boosting and connections with Additive logistic regression
    - Gradient Boosted Decision Trees (GBDT)

Bagging (Bootstrap Aggregating)

- Variance reduction technique
- Method:
  - create bootstrap (sampling with replacement) replicates of dataset
  - fit a model to each replicate
  - combine predictions by averaging or voting
- Properties
  - stabilizes unstable models
    - **Decision trees**, neural nets
    - Prefer deeper trees (why??)
  - easily parallelizable; implementable
- Ref: www.stat.berkeley.edu/users/breiman/

Bagged Trees example from HTF Sec 8.7

- A variance reduction technique
- http://scikit-learn.org/stable/auto_examples/ensemble/plot_bias_variance.html

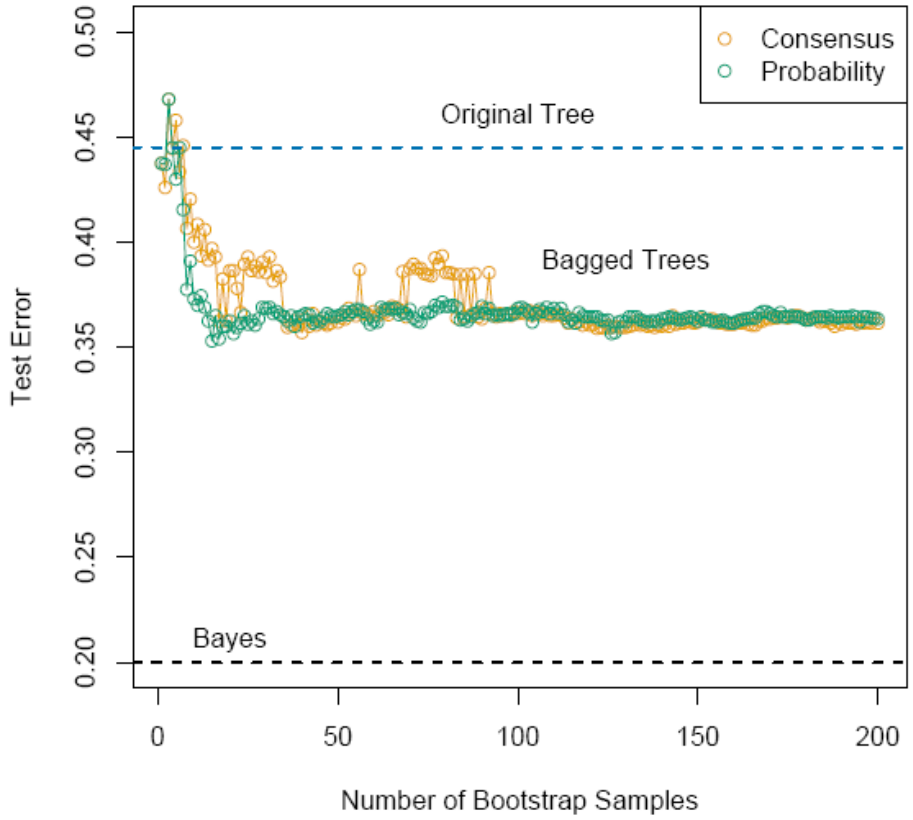


FIGURE 8.10. *Error curves for the bagging example of Figure 8.9. Shown is the test error of the original tree and bagged trees as a function of the number of bootstrap samples. The orange points correspond to the consensus vote, while the green points average the probabilities.*

Random Forests

- Bagging decision trees with additional randomization
  - generate bootstrap samples.
  - build one tree per bootstrap sample
  - increase diversity via additional randomization: randomly pick a subset of features of size m << d to split at each node
    - Goal: Decrease correlation among trees without affecting bias much
    - Should determine *m* using out of bag (OOB) samples.
  - take equally weighted average of the results from each tree
  - ----------
- ***Theory**: Variance of average of B (identically distributed, but not independent) variables = $\rho\sigma^2 + \frac{1-\rho}{B}\sigma^2$

  Where ρ = pair-wise correlation and σ^2 is variance of each variable.

Random Forests

- R: randomForest
- Reduce correlation among bagged trees
  - Consider only subset of variables at each split

Algorithm 15.1 *Random Forest for Regression or Classification.*

1. For $b = 1$ to B:
   (a) Draw a bootstrap sample $\mathbf{Z}^*$ of size N from the training data.
   (b) Grow a random-forest tree T_b to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} is reached.
      i. Select m variables at random from the p variables.
      ii. Pick the best variable/split-point among the m.
      iii. Split the node into two daughter nodes.
2. Output the ensemble of trees $\{T_b\}_1^B$.

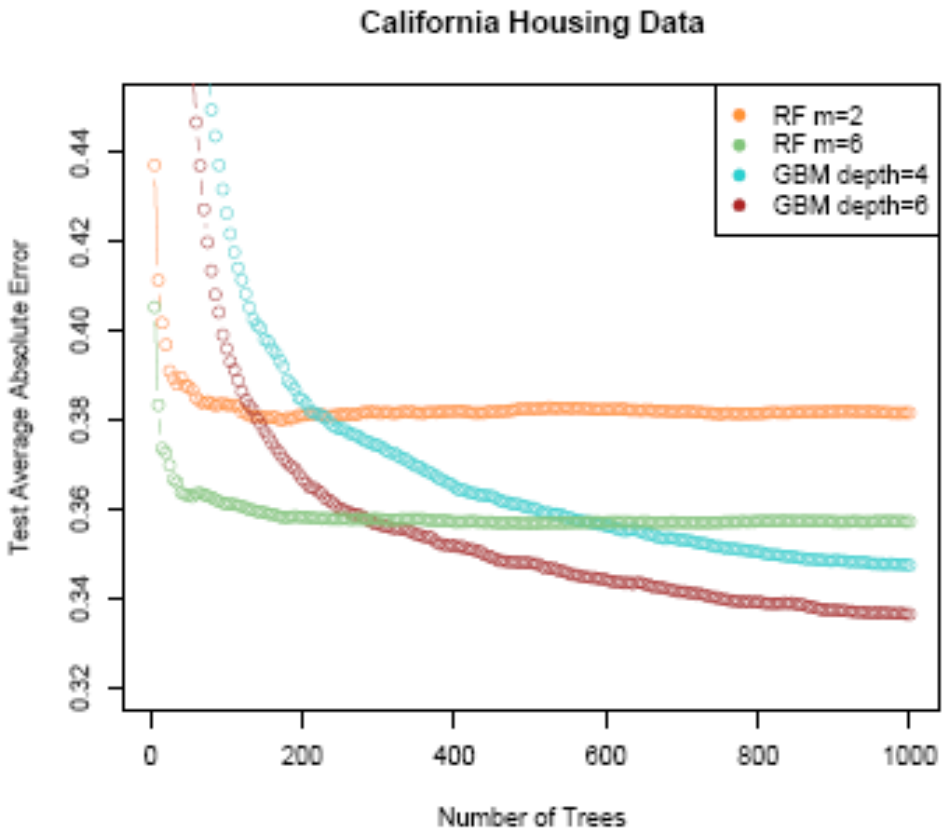
To make a prediction at a new point x:

Regression: $\hat{f}_{\text{rf}}^B(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x)$.

Classification: Let $\hat{C}_b(x)$ be the class prediction of the bth random-forest tree. Then $\hat{C}_{\text{rf}}^B(x) = \textit{majority vote}\ \{\hat{C}_b(x)\}_1^B$.

Performance

- Easy to train/tune
  - And can get variable importance like GBDT
- Typically better than bagging, often comparable to boosting, not as good as GBDT



Boosting

- Goal: improving classification, esp. with weak models
- Method:
  - sequentially fit models
  - later models see more of the samples mispredicted by earlier ones (input reweighting)
  - combine using weighted average (later models get more weight)
- Properties
  - reduces both bias and variance
  - slow to overfit
  - works well with "stumps", naïve Bayes,…
- Several Variations
- Danger: Sensitive to outliers

AdaBoost Algorithm

- **Note:** both outputs y_i and hypothesis (classifier) h_i are **+/- 1**

Input: Training set $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)$

Algorithm: Initialize $D_1(i) = 1/m$

For $t = 1, \ldots, T$

Weighted error

- Train a weak learner using distribution D_t
- Get weak hypothesis h_t with error $\epsilon_t = \Pr_{\mathbf{x} \sim D_t}[h_t(\mathbf{x}) \neq y]$
- Choose $\alpha_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$
- Update

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(\mathbf{x}_i))}{Z_t}$$

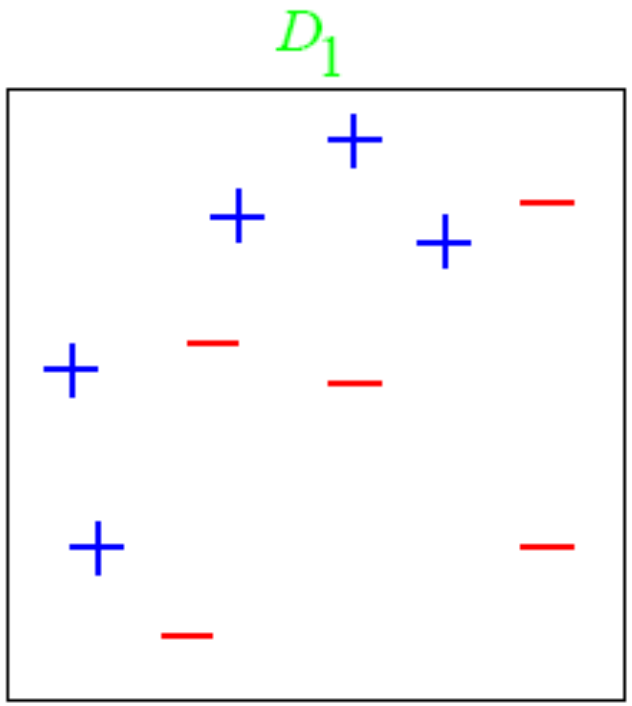
where Z_t is the normalization factor

$$\text{Output:} \quad h(\mathbf{x}) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})\right)$$

Toy Example from Schapire's NIPS 2007 tute

- Weak classifier: single horizontal or vertical half-plane

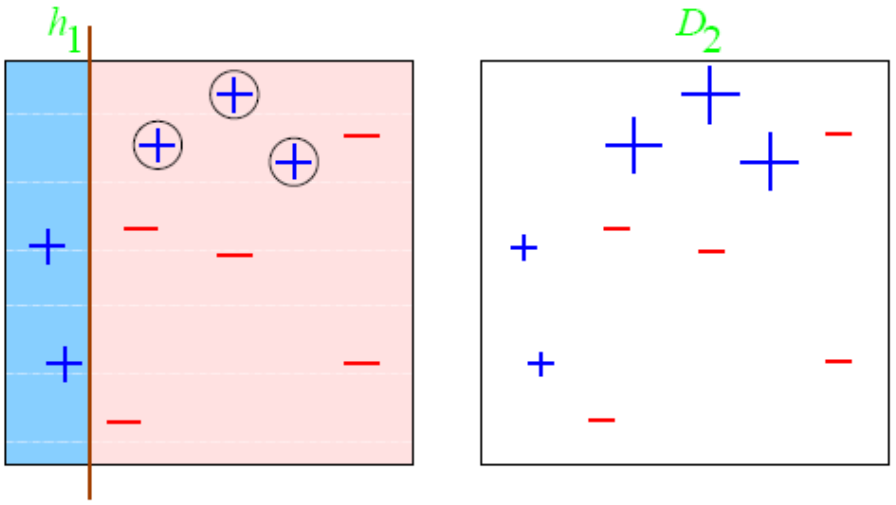
- Also see this fancy version



http://media.nips.cc/Conferences/2007/Tutorials/Slides/schapire-NIPS-07-tutorial.pdf

Rounds 1 and 2

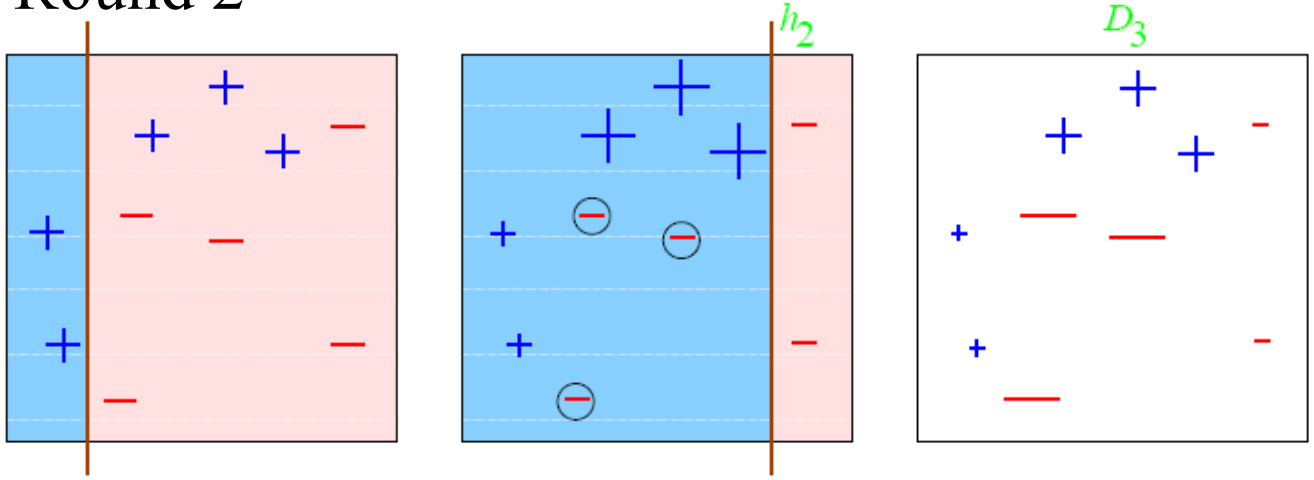
Round 1



$\varepsilon_1 = 0.30$

$\alpha_1 = 0.42$

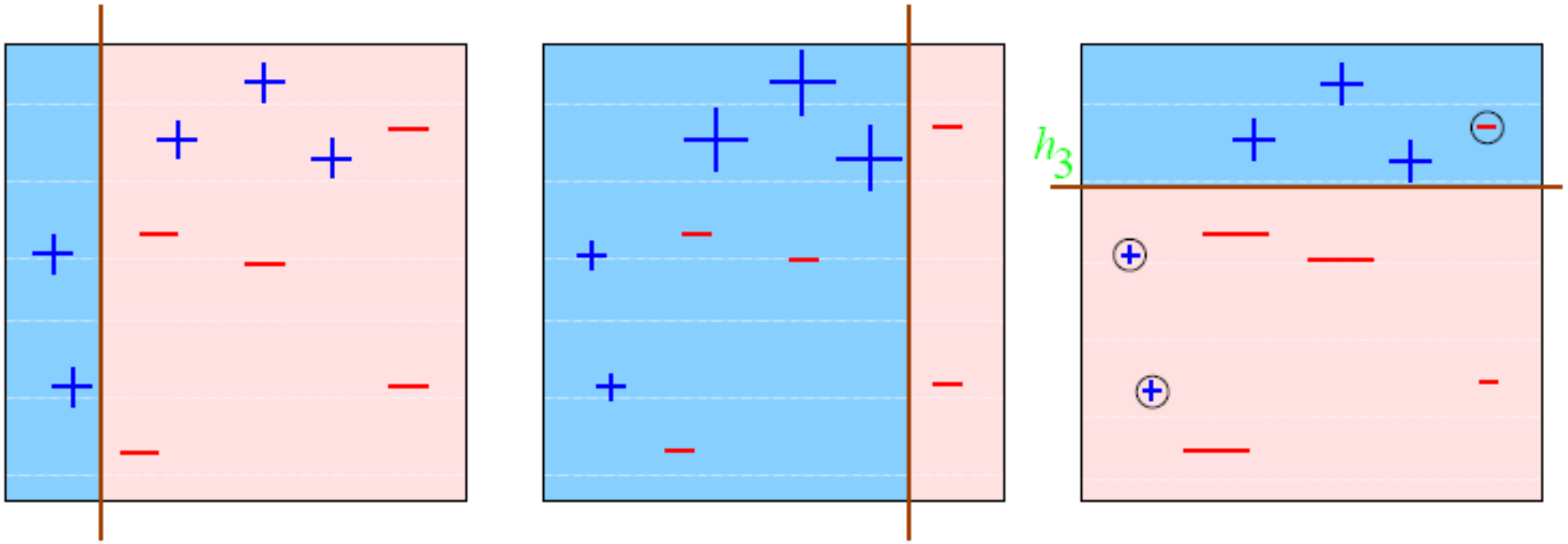
Round 2



$\varepsilon_2 = 0.21$

$\alpha_2 = 0.65$

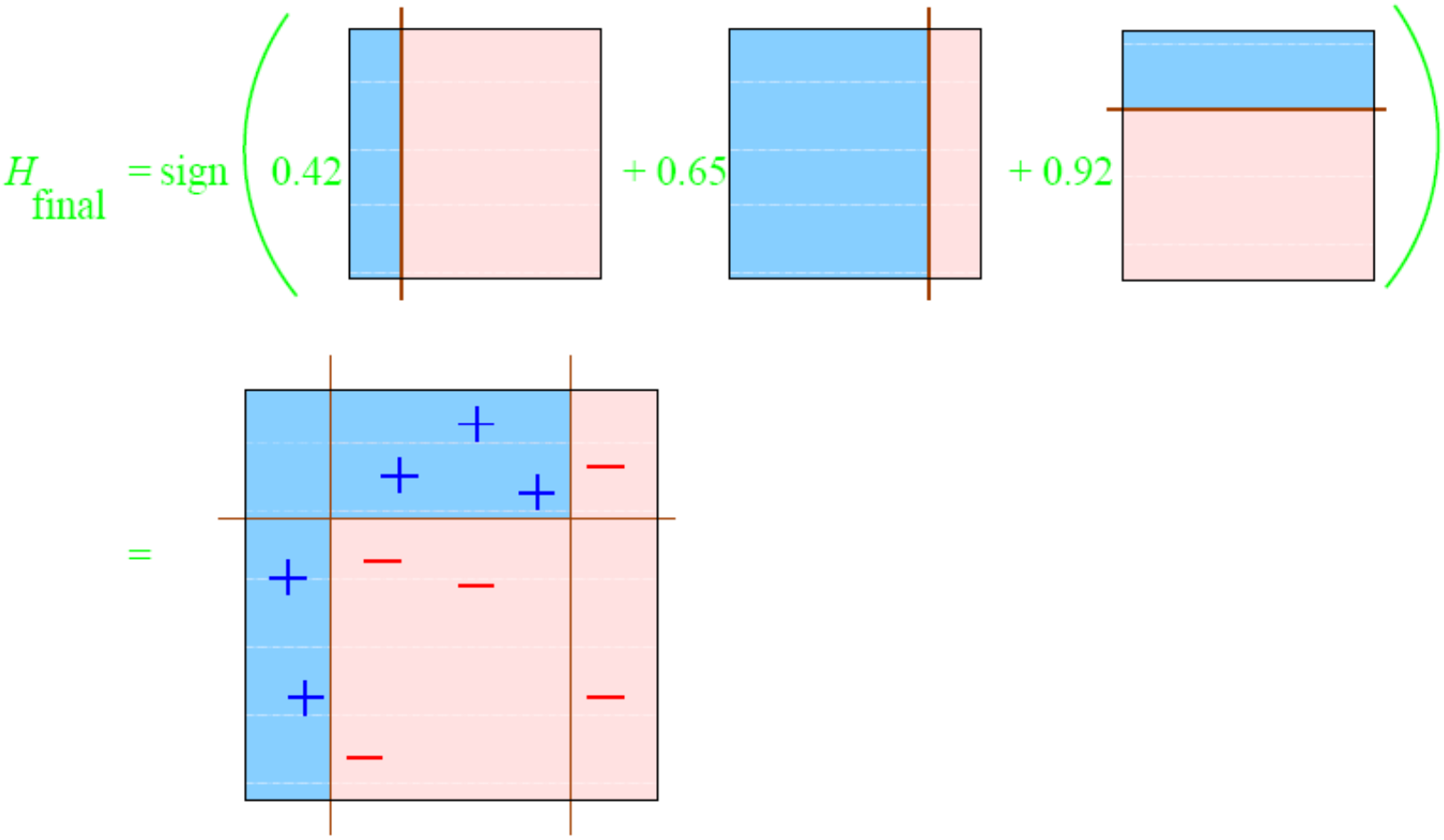
Round 3



$\varepsilon_3 = 0.14$

$\alpha_3 = 0.92$

Final Classifier



Many Stumps win in the long run

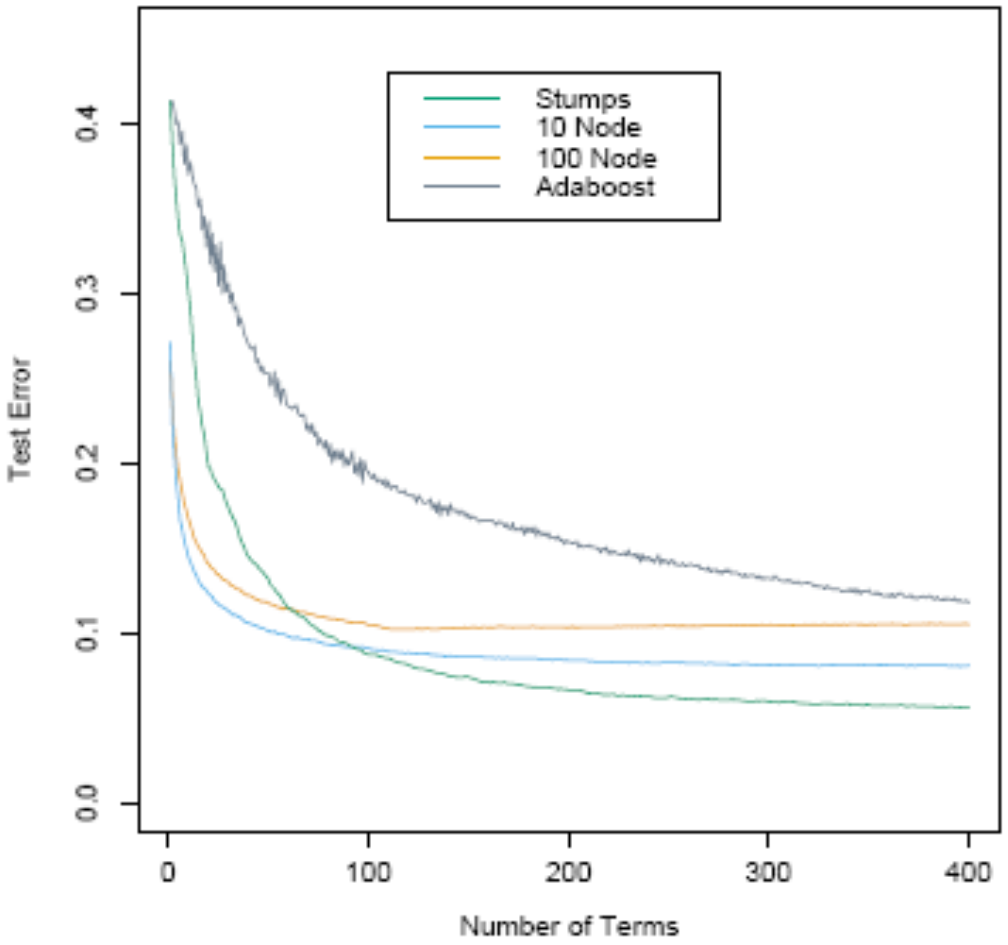


FIGURE 10.9. *Boosting with different sized trees, applied to the example (10.2) used in Figure 10.2. Since the generative model is additive, stumps perform the best. The boosting algorithm used the binomial deviance loss in Algorithm 10.3; shown for comparison is the AdaBoost Algorithm 10.1.*

Gradient Boosting

- See HTF Ch 10 for details and case studies
- **R Code: gbm** (Commercial: TreeNet)
  - Also in Scikit-learn (**GradientBoostingRegressor**)
  - https://www.youtube.com/watch?v=IXZKgIsZRm0
- **Gradient descent in function space**
  - Less Greedy, more accurate, robust and interpretable than Adaboost
  - Can apply a variety of loss functions
  - Importance of variables (**for trees**): net information gain across all splits

Regression Tree Example (Baseball Players Salaries)

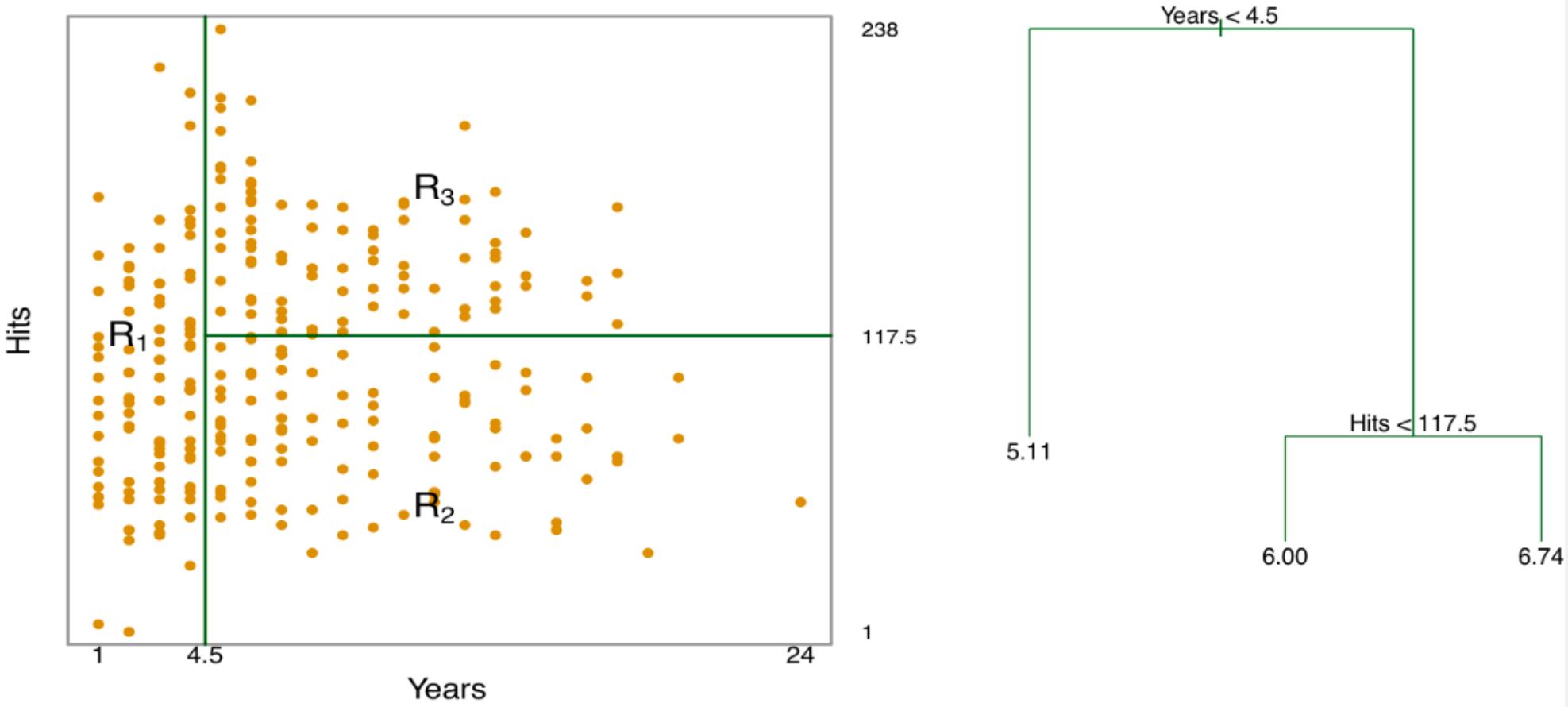
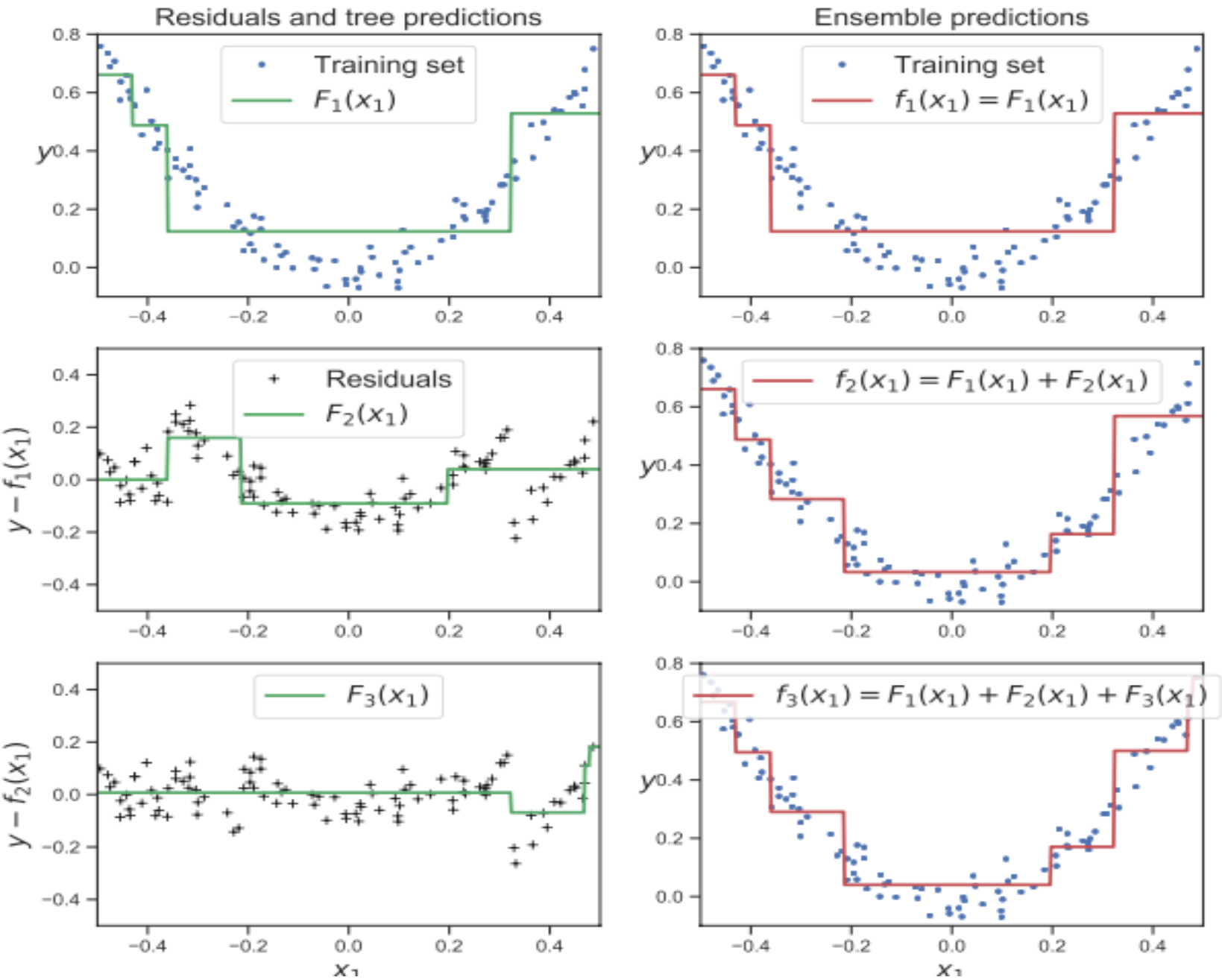


Figure: Regression Tree for Baseball Salaries (in '000, log transformed, so 6 means $402,834



From AG pg. 205

Gradient descent in function space*

TABLE 10.2. *Gradients for commonly used loss functions.*

Setting	Loss Function	$-\partial L(y_i, f(x_i))/\partial f(x_i)$
Regression	$\frac{1}{2}[y_i - f(x_i)]^2$	$y_i - f(x_i)$
Regression	$\lvert y_i - f(x_i)\rvert$	$\text{sign}[y_i - f(x_i)]$
Regression	Huber	$y_i - f(x_i)$ for $\lvert y_i - f(x_i)\rvert \leq \delta_m$ $\delta_m \text{sign}[y_i - f(x_i)]$ for $\lvert y_i - f(x_i)\rvert > \delta_m$ where $\delta_m = \alpha$th-quantile$\{\lvert y_i - f(x_i)\rvert\}$
Classification	Deviance	kth component: $I(y_i = \mathcal{G}_k) - p_k(x_i)$

Gradient Boosted (Decision) Trees (regression algorithm)*

Algorithm 10.3 *Gradient Tree Boosting Algorithm.*

1. Initialize $f_0(x) = \arg\min_\gamma \sum_{i=1}^{N} L(y_i, \gamma)$.
2. For $m = 1$ to M:
   (a) For $i = 1, 2, \ldots, N$ compute
   $$r_{im} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f=f_{m-1}}$$
   (b) Fit a regression tree to the targets r_{im} giving terminal regions R_{jm}, $j = 1, 2, \ldots, J_m$.
   (c) For $j = 1, 2, \ldots, J_m$ compute
   $$\gamma_{jm} = \arg\min_\gamma \sum_{x_i \in R_{jm}} L\left(y_i, f_{m-1}(x_i) + \gamma\right).$$
   (d) Update $f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$.
3. Output $\hat{f}(x) = f_M(x)$.

Takeaway: Fits trees sequentially Using gradient boosting; Applies to regression/classification By using appropriate Loss fn.

GBDT can rank order features

- See HTF 10.14 for case studies

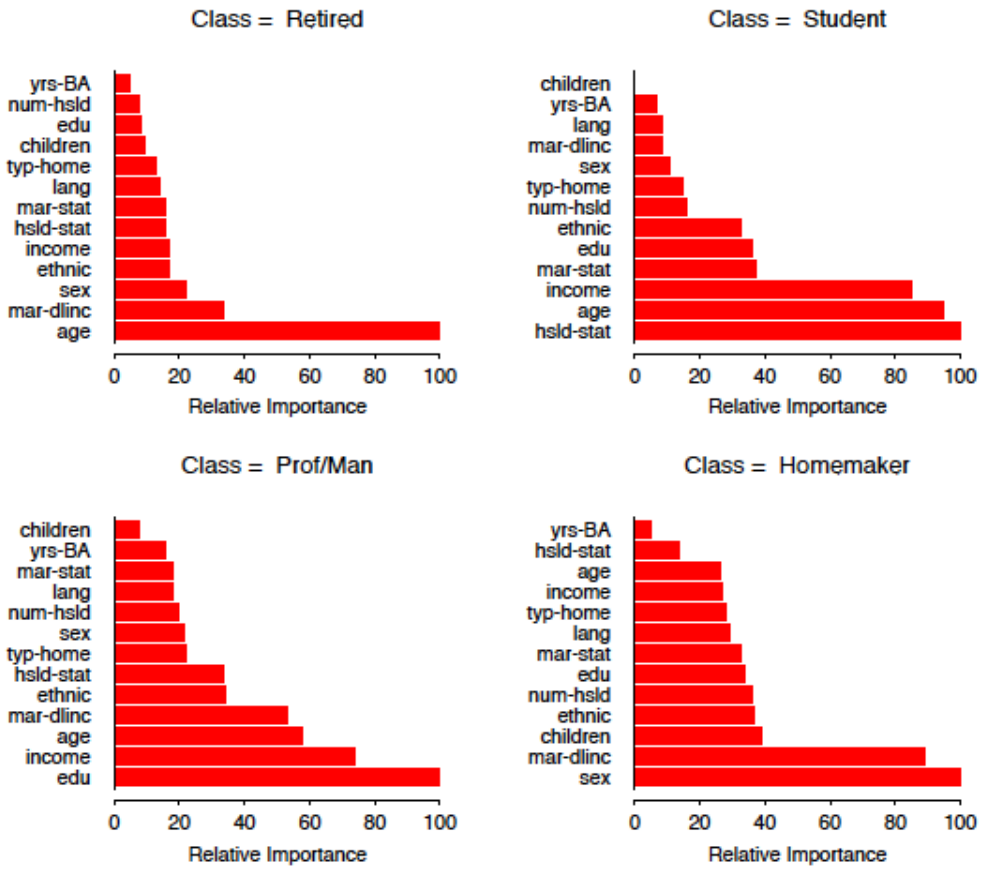
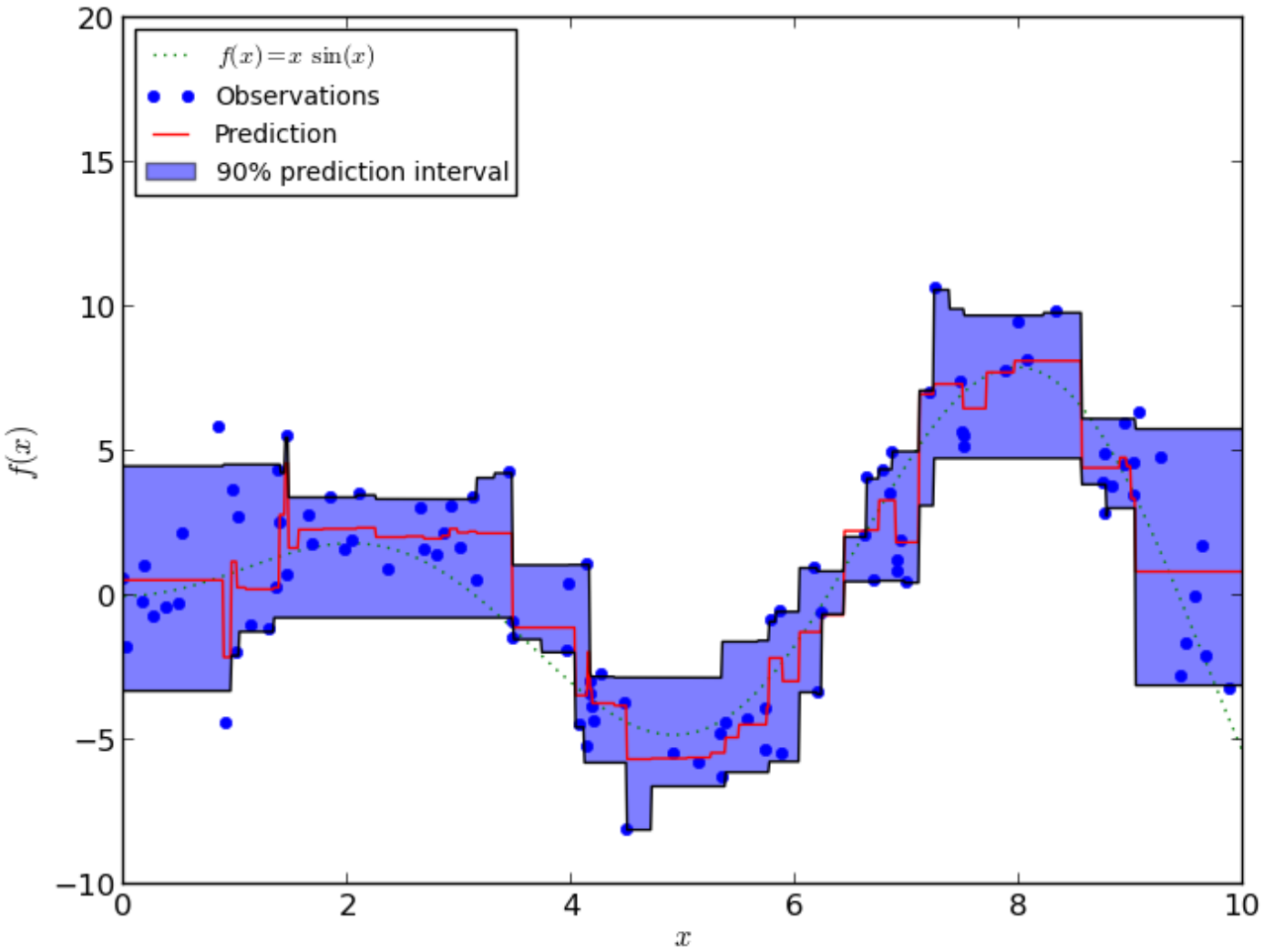


FIGURE 10.24. *Predictor variable importances separately for each of the four classes with lowest error rate for the demographics data.*

Occupation Prediction Example from HTF

Prediction Intervals

- http://scikit-learn.org/stable/auto_examples/ensemble/plot_gradient_boosting_quantile.html



Large Scale Boosting

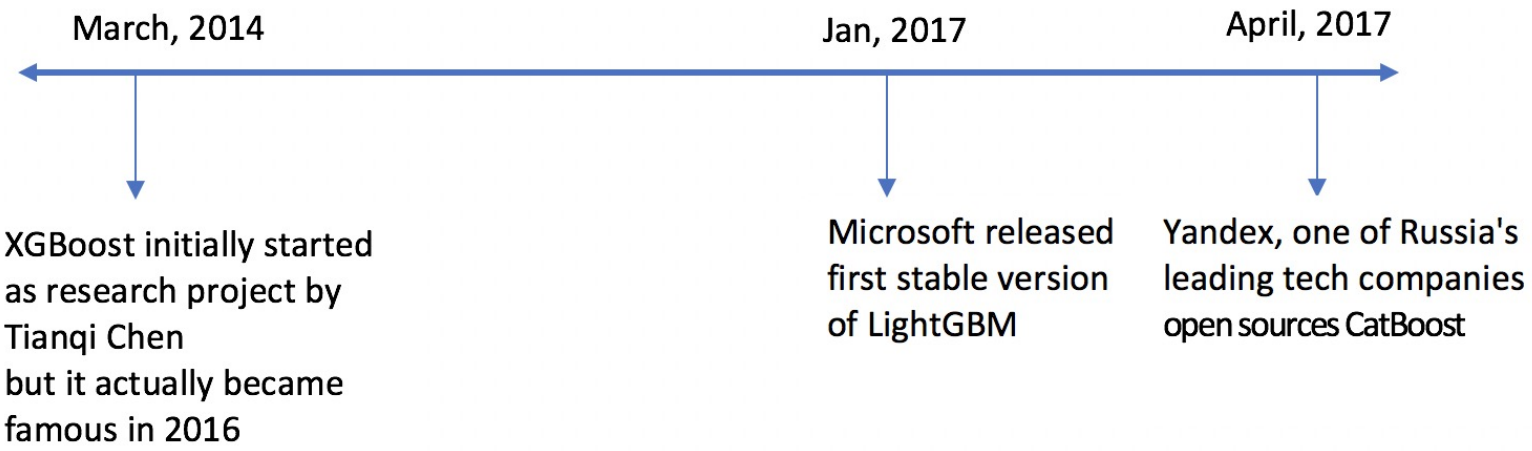
- XGBoost: extreme Boosting
  - Boosts DTs as well as GLMs
    - Supports regression, classification, ranking and user defined objectives
  - Portable: desktop or cloud
  - Distributed; R and Python versions

  - https://xgboost.readthedocs.org/en/latest/
  - https://github.com/dmlc/xgboost
  - Needs tuning on several hyperparameters, e.g. using a package suc as hyperopt, a specific example of Bayesian Hyperparameter Optimization
  - https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/

CatBoost (Yandex)

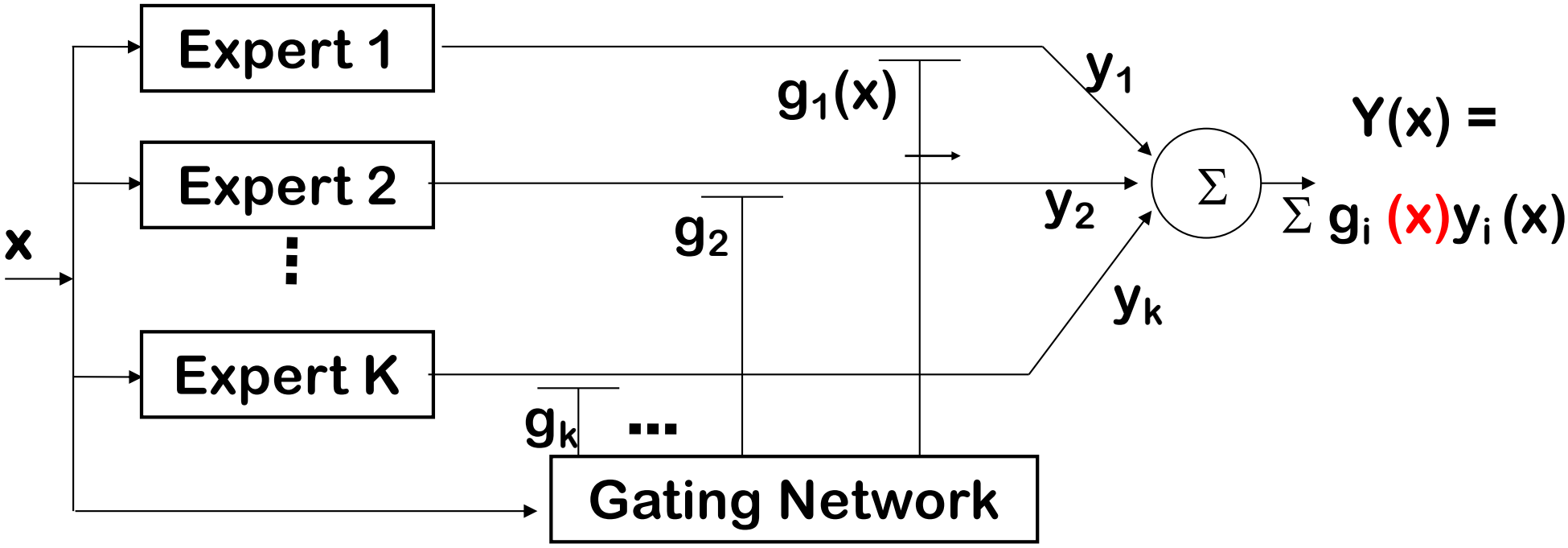
- Very competitive GBDT algorithm
- Handles categorical variables well
- Can be applied for ranking tasks
- https://tech.yandex.com/catboost/

- Also see this article comparing XGBoost, LightGBM and CatBoost



Cop-operative Approach: Mixtures of Experts (MoE)

- Both $g_i(x)$'s and expert parameters adapted during training.



- Hierarchical versions possible

MoE makes a comeback!

LIMoE: Learning Multiple Modalities with One Sparse Mixture-of-Experts Model

Thursday, June 9, 2022

Posted by Basil Mustafa, Research Software Engineer and Carlos Riquelme, Research Scientist, Google Research, Brain team

LiMoE Blog

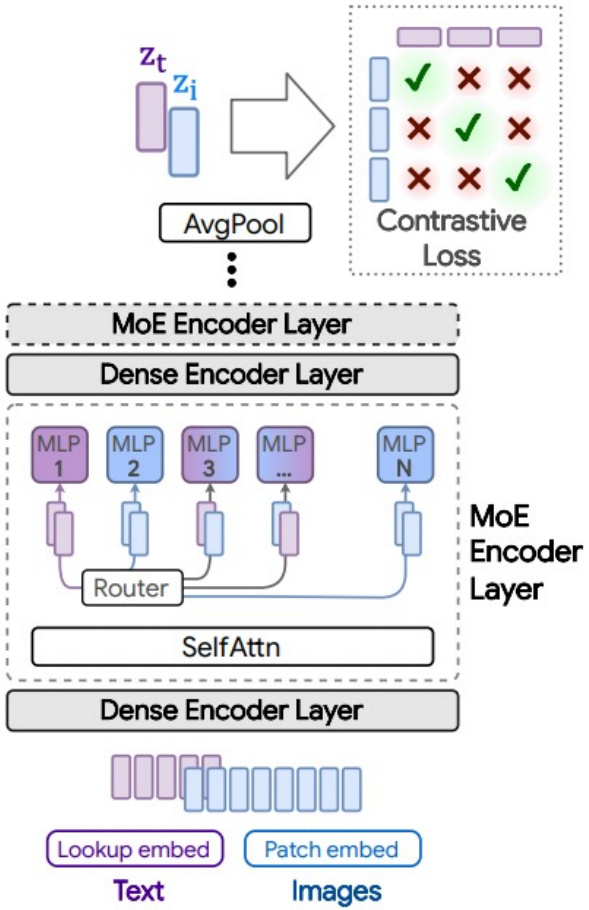


Figure 1: LIMoE, a sparsely activated multimodal model, processes both images and texts, utilising conditional computation to allocate tokens in a modality-agnostic fashion.

Summary of Committee Classifiers

- Variance reduction method
- Diversity is good (agree to disagree!)
- Good gains with little extra effort
- Provides estimate of decision confidence
  - Increased flexibility in accuracy-rejection rates
- "Standard"

Backups