

Statistical Machine Learning (BE4M33SSU)

Lecture 12: Ensembling

Jan Drchal

Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Computer Science

Topics covered in the lecture:

- ◆ Ensemble Methods
- ◆ Bias-Variance Decomposition
- ◆ Bagging
- ◆ Random Forests
- ◆ Boosting and Gradient Boosting
- ◆ Gradient Boosted Trees

Ensemble Methods

- ◆ Inspired in *Wisdom of the crowd*
 - (weighted) averaging or taking majority vote
 - cancelling effect of noise of individual opinions,
 - examples: politics, trial by jury (vs. trial by judge), sports (figure skating, gymnastics), Wikipedia, Quora, Stack Overflow, . . .
- ◆ Learning and aggregating multiple predictors
- ◆ Ensemble may be built using single or different types of predictors



Wikimedia Commons

Prediction Problem: Expected Risk and Error Decomposition

Expected risk for data generated by $p(x, y)$:

$$R(h) = \mathbb{E}_{(x,y) \sim p} \left(\ell(y, h(x)) \right)$$

- ◆ The best attainable (Bayes) risk is $R^* = \inf_{h \in \mathcal{Y}^{\mathcal{X}}} R(h)$
- ◆ The best predictor in \mathcal{H} is $h_{\mathcal{H}} \in \text{Argmin}_{h \in \mathcal{H}} R(h)$
- ◆ The predictor $h_m = A(\mathcal{T}^m)$ learned from \mathcal{T}^m has risk $R(h_m)$

Excess error measures deviation of the learned predictor from the best one:

$$\underbrace{\left(R(h_m) - R^* \right)}_{\text{excess error}} = \underbrace{\left(R(h_m) - R(h_{\mathcal{H}}) \right)}_{\text{estimation error}} + \underbrace{\left(R(h_{\mathcal{H}}) - R^* \right)}_{\text{approximation error}}$$

Risk Averaged over Datasets

- ◆ How will our predictor behave when sampling different training sets?
- ◆ We can define the errors considering average over all possible datasets \mathcal{T}^m , i.e., $\mathbb{E}_{\mathcal{T}^m}(R(h_m))$
- ◆ The errors can be redefined as:

$$\underbrace{\left(\mathbb{E}_{\mathcal{T}^m}(R(h_m)) - R^* \right)}_{\text{excess error}} = \underbrace{\left(\mathbb{E}_{\mathcal{T}^m}(R(h_m)) - R(h_{\mathcal{H}}) \right)}_{\text{estimation error}} + \underbrace{\left(R(h_{\mathcal{H}}) - R^* \right)}_{\text{approximation error}}$$

Predictors Averaged over Datasets

- ◆ Let us also define a model averaged over all possible datasets:

$$g_m(x) = \mathbb{E}_{\mathcal{T}^m} \left(h_m(x) \right)$$

- ◆ Unlike individual h_m models, g_m has an access to the whole $p(x, y)$
- ◆ g_m can't be actually evaluated for infinite number of \mathcal{T}^m datasets

Bias-Variance Decomposition for Regression

- ◆ Consider a regression problem with data generated as follows:

$$y = h^*(x) + \epsilon$$

where ϵ is noise: $\mathbb{E}(\epsilon) = 0$ and $\text{Var}(\epsilon) = \sigma^2$, e.g., $\epsilon \sim \mathcal{N}(0, \sigma^2)$

- ◆ Use squared loss:

$$\ell(y, h(x)) = (h(x) - y)^2$$

- ◆ The optimal predictor $h^*(x)$ has a nonzero risk (for $\sigma^2 > 0$):

$$R^* = \mathbb{E}_{x,y} \left((h^*(x) - y)^2 \right) = \mathbb{E}_{\epsilon} (\epsilon^2) = \text{Var}(\epsilon) = \sigma^2$$

Bias-Variance Decomposition for Regression 2

- ◆ The expected risk for h_m can be decomposed:

$$\begin{aligned}
 \mathbb{E}_{\mathcal{T}^m} \left(R(h_m) \right) &= \mathbb{E}_{x,y,\mathcal{T}^m} \left(\left(h_m(x) - y \right)^2 \right) \\
 &= \dots \\
 &= \underbrace{\mathbb{E}_{x,y,\mathcal{T}^m} \left(\left(h_m(x) - g_m(x) \right)^2 \right)}_{\text{variance}} + \\
 &\quad + \underbrace{\mathbb{E}_{x,y} \left(\left(g_m(x) - h^*(x) \right)^2 \right)}_{\text{bias}^2} + \underbrace{\sigma^2}_{\text{noise}}
 \end{aligned}$$

- ◆ The error splits into three terms
 - **variance**: difference of h_m from the averaged predictor g_m ,
 - **bias**²: difference of the averaged predictor g_m from the optimal one,
 - **noise**: irreducible determined by data

Excess Error vs. Bias and Variance

- ◆ The excess error is defined as:

$$\mathbb{E}_{\mathcal{T}^m}(R(h_m)) - R^*$$

- ◆ As $R^* = \sigma^2$ we get:

$$\begin{aligned} \mathbb{E}_{\mathcal{T}^m}(R(h_m)) - R^* &= \underbrace{\mathbb{E}_{x,y} \left(\left(g_m(x) - h^*(x) \right)^2 \right)}_{\text{bias}^2} \\ &\quad + \underbrace{\mathbb{E}_{x,y,\mathcal{T}^m} \left(\left(h_m(x) - g_m(x) \right)^2 \right)}_{\text{variance}} \end{aligned}$$

- ◆ We have
 - **bias**² \approx approximation error,
 - **variance** \approx estimation error

Derivation of the Bias-Variance Decomposition



$$\begin{aligned}\mathbb{E}_{\mathcal{T}^m} \left(R(h_m) \right) &= \mathbb{E}_{x,y,\mathcal{T}^m} \left(\left(h_m(x) - y \right)^2 \right) \\ &= \mathbb{E}_{x,y,\mathcal{T}^m} \left(\left(h_m(x) - g_m(x) + g_m(x) - y \right)^2 \right) \\ &= \mathbb{E}_{x,y,\mathcal{T}^m} \left(\left(h_m(x) - g_m(x) \right)^2 + \left(g_m(x) - y \right)^2 \right. \\ &\quad \left. + 2 \left(h_m(x) - g_m(x) \right) \left(g_m(x) - y \right) \right) \\ &= \mathbb{E}_{x,y,\mathcal{T}^m} \left(\left(h_m(x) - g_m(x) \right)^2 \right) + \mathbb{E}_{x,y,\mathcal{T}^m} \left(\left(g_m(x) - y \right)^2 \right) \\ &\quad + \mathbb{E}_{x,y} \left(2 \left(\underbrace{\mathbb{E}_{\mathcal{T}^m} \left(h_m(x) \right)}_{g_m(x)} - g_m(x) \right) \left(g_m(x) - y \right) \right)\end{aligned}$$

Derivation of the Bias-Variance Decomposition 2

We get:

$$\mathbb{E}_{\mathcal{T}^m} \left(R(h_m) \right) = \underbrace{\mathbb{E}_{x,y,\mathcal{T}^m} \left(\left(h_m(x) - g_m(x) \right)^2 \right)}_{\text{variance}} + \mathbb{E}_{x,y,\mathcal{T}^m} \left(\left(g_m(x) - y \right)^2 \right)$$

Note that the second term does not depend on \mathcal{T}^m :

$$\mathbb{E}_{x,y,\mathcal{T}^m} \left(\left(g_m(x) - y \right)^2 \right) = \mathbb{E}_{x,y} \left(\left(g_m(x) - y \right)^2 \right)$$

Derivation of the Bias-Variance Decomposition 3

Let us continue with the second term:

$$\begin{aligned}\mathbb{E}_{x,y} \left(\left(g_m(x) - y \right)^2 \right) &= \mathbb{E}_{x,\epsilon} \left(\left(g_m(x) - h^*(x) - \epsilon \right)^2 \right) \\ &= \mathbb{E}_{x,\epsilon} \left(\left(g_m(x) - h^*(x) \right)^2 + \epsilon^2 - 2\epsilon \left(g_m(x) - h^*(x) \right) \right) \\ &= \mathbb{E}_x \left(\left(g_m(x) - h^*(x) \right)^2 \right) + \mathbb{E}_\epsilon \left(\epsilon^2 \right) \\ &\quad - 2\mathbb{E}_{x,\epsilon} \left(\epsilon \left(g_m(x) - h^*(x) \right) \right) \\ &= \underbrace{\mathbb{E}_x \left(\left(g_m(x) - h^*(x) \right)^2 \right)}_{\text{bias}^2} + \underbrace{\sigma^2}_{\text{noise}}\end{aligned}$$

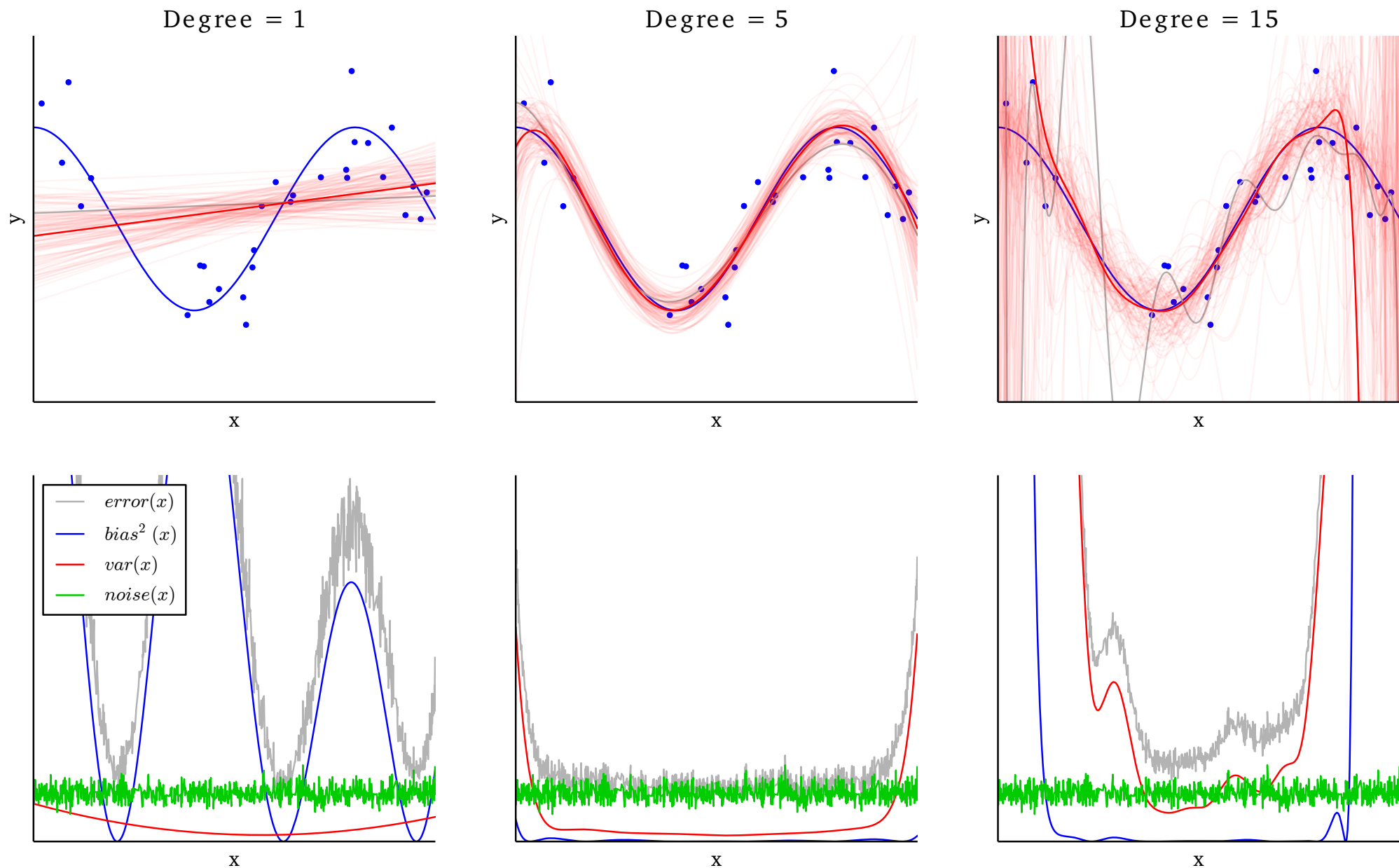
Pointwise Bias-Variance

We can express the bias and variance as function of x by not integrating over in expected values

$$\begin{aligned}
 \mathbb{E}_{y|x, \mathcal{T}^m} \left(\ell(y, h_m(x)) \right) &= \mathbb{E}_{y|x, \mathcal{T}^m} \left(\left(h_m(x) - y \right)^2 \right) \\
 &= \underbrace{\text{Var}_{y|x, \mathcal{T}^m} \left(h_m(x) \right)}_{\text{variance}(x)} + \\
 &\quad + \underbrace{\mathbb{E}_{y|x} \left(\left(g_m(x) - h^*(x) \right)^2 \right)}_{\text{bias}(x)^2} + \underbrace{\sigma^2}_{\text{noise}}
 \end{aligned}$$

Bias-Variance: Example

◆ Polynomial regression with a varying degree of polynomial

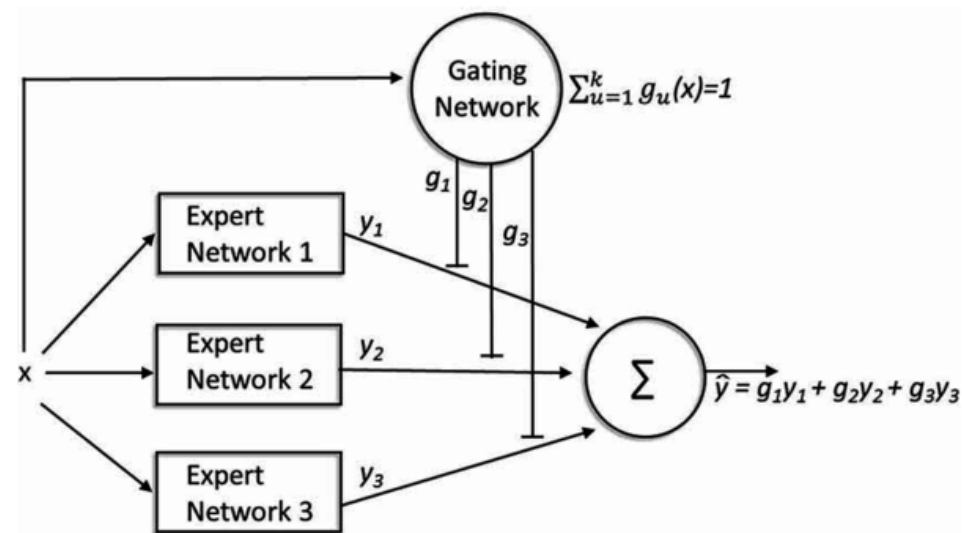
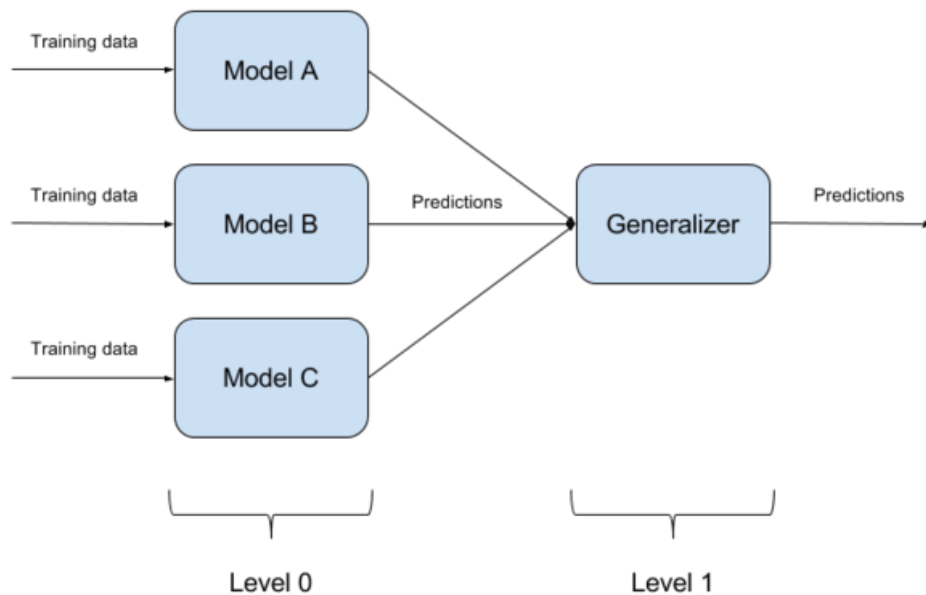


Ensembling Approaches

- ◆ Bagging (Bootstrap AGGREGatING):
 - sample different training sets from the original training set
 - train *high variance low bias* predictors based on these sets and average them
 - exploits independence between predictors
- ◆ Boosting:
 - sequentially train *low variance high bias* predictors
 - subsequent predictors learn from the mistakes of the previous ones
 - increase weights for mislabeled samples
 - exploits dependence between learners

Stacking and Mixture of Experts

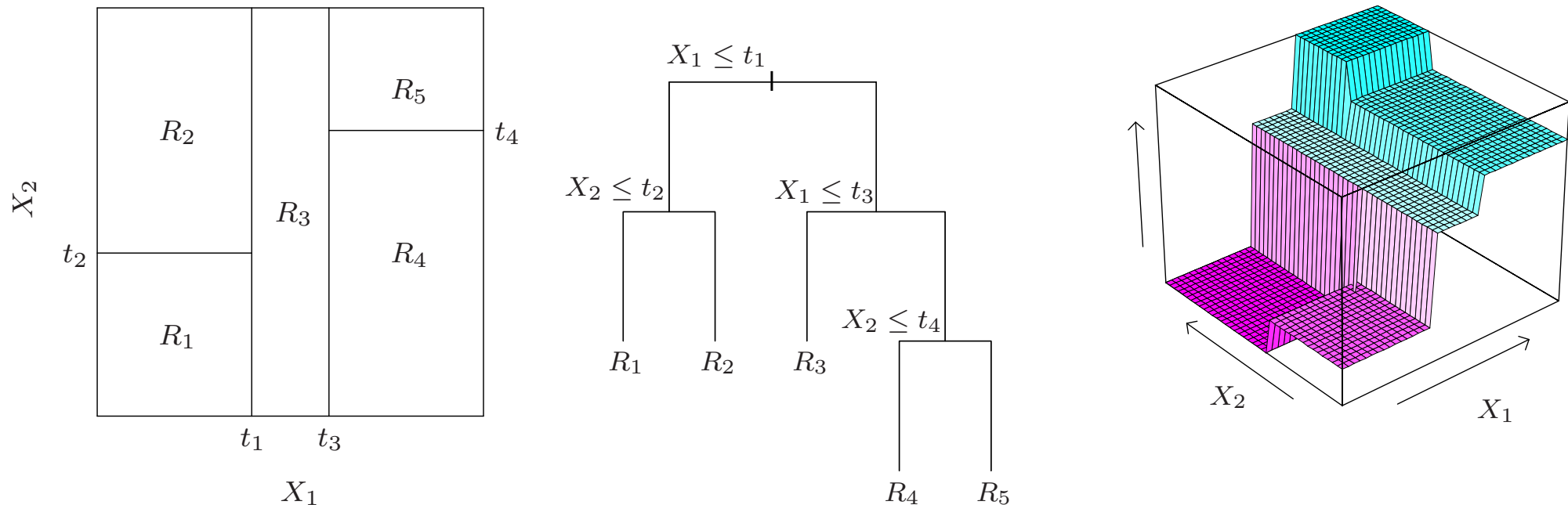
- ◆ Combine *base-learners* with *meta-learner*



<https://www.commonlounge.com/discussion/9331c0d004704e89bd4d1da08fd7c7bc>

Decision/Regression Trees

- ◆ Nodes correspond to mutually exclusive subsets of the original training data as well as mutually exclusive subsets of the input space \mathcal{X}
- ◆ Inner node further splits its subset



Decision/Regression Trees (contd.)

- ◆ Training set: $\mathcal{T}^m = \{(\mathbf{x}_i, y_i) \mid i = 1, \dots, m\}$, $\mathbf{x}_i = (x_{i1}, x_{i2}, \dots, x_{ip})$
- ◆ Input space split into regions defined in leaves: R_r , $r \in \{1, \dots, M\}$
- ◆ We can model *region responses* by constants c_r , $r \in \{1, \dots, M\}$ but other possibilities, e.g., linear regression are possible

- ◆ Prediction:

$$h(\mathbf{x}) = \sum_{r=1}^M c_r [\mathbf{x} \in R_r]$$

- ◆ For sum of squares *loss function* $\sum_{i=1}^m (y_i - h(\mathbf{x}_i))^2$ we set the responses to be the averages over regions:

$$\hat{c}_r = \frac{1}{|S_r|} \sum_{\mathbf{x}_i \in R_r} y_i \quad (\text{see seminar})$$

where we define samples per region sets :

$$S_r = \{(\mathbf{x}_i, y_i) : (\mathbf{x}_i, y_i) \in \mathcal{T}^m \wedge \mathbf{x}_i \in R_r\}$$

Greedy Learning of Decision/Regression Trees

- ◆ How many distinct decision trees with n Boolean attributes for binary classification?
 - at least as many as boolean functions of p attributes
 - = number of distinct truth tables with 2^p rows: 2^{2^p}
 - For 6 Boolean attributes at least
18,446,744,073,709,551,616 trees!
- ◆ Learning is NP-complete: [Hyafil and Rivest 1976]
- ◆ We need heuristics \Rightarrow **greedy approach**
- ◆ Recursively choose the "most important" attribute to find a small tree consistent with the training data
- ◆ Split points:
 - **nominal attribute**: try all possibilities
 - **ordinal/continuous attribute**: try attribute values based on all training data samples or their subset

Regression Trees: Which Attribute to Split?

- ◆ The "most important" attribute for regression trees would be the one which will reduce the loss (sum of squared errors) by the greatest amount

- ◆ We have:

$$h(\mathbf{x}) = \sum_{r=1}^R c_r [\mathbf{x} \in R_r]$$

- ◆ Consider splitting attribute j and split point s , we split an original region R into a pair of half-planes for an ordinal (e.g., continuous) attribute:

$$R_L(j, s) = \{\mathbf{x} | \mathbf{x} \in R \wedge x_j \leq s\} \text{ and } R_R(j, s) = \{\mathbf{x} | \mathbf{x} \in R \wedge x_j > s\}$$

similarly for a nominal attribute:

$$R_L(j, s) = \{\mathbf{x} | \mathbf{x} \in R \wedge x_j = s\} \text{ and } R_R(j, s) = \{\mathbf{x} | \mathbf{x} \in R \wedge x_j \neq s\}$$

Regression Trees: Which Attribute to Split? (contd.)

- ◆ We seek for an attribute j and a split point s which minimize:

$$\min_{c_L} \sum_{\mathbf{x}_i \in R_L(j,s)} (y_i - c_L)^2 + \min_{c_R} \sum_{\mathbf{x}_i \in R_R(j,s)} (y_i - c_R)^2$$

for $(\mathbf{x}_i, y_i) \in S \subseteq \mathcal{T}^m$ ($S = \mathcal{T}^m$ for the root node) and $R = R_L \cup R_R$

- ◆ Inner minimizations (region response values) are solved by averaging tree outputs per region:

$$\hat{c}_L = \frac{1}{|S_L(j,s)|} \sum_{\mathbf{x}_i \in R_L(j,s)} y_i \quad \text{and} \quad \hat{c}_R = \frac{1}{|S_R(j,s)|} \sum_{\mathbf{x}_i \in R_R(j,s)} y_i$$

where $S_k(j,s) = \{(\mathbf{x}_i, y_i) \mid (\mathbf{x}_i, y_i) \in \mathcal{T} \wedge \mathbf{x}_i \in R_k(j,s)\}$

Tree Learning Algorithm

BUILD-TREE(S)

```

1   $i = \text{IMPURITY}(S)$  // e.g., the squared loss
2   $\hat{i}, \hat{j}, \hat{s}, \hat{S}_L, \hat{S}_R = 0, 0, 0, \emptyset, \emptyset$  // current best kept in these
3  for  $j \in \{1, \dots, p\}$  // iterate over attributes
4      for  $s \in \text{SPLIT-POINTS}(S, j)$  // iterate over all split points
5           $S_L, S_R = \text{SPLIT}(S, j, s)$ 
6           $i_L = \text{IMPURITY}(S_L)$ 
7           $i_R = \text{IMPURITY}(S_R)$ 
8          if  $i_L + i_R < \hat{i}$  and  $|S_L| > 0$  and  $|S_R| > 0$ 
9               $\hat{i}, \hat{j}, \hat{s}, \hat{S}_L, \hat{S}_R = (i_L + i_R), j, s, S_L, S_R$ 
10 if  $\hat{i} > i$ 
11      $N_L = \text{BUILD-TREE}(\hat{S}_L)$ 
12      $N_R = \text{BUILD-TREE}(\hat{S}_R)$ 
13     return  $\text{DECISION-NODE}(\hat{j}, \hat{s}, N_L, N_R)$ 
14 else return  $\text{LEAF-NODE}(S)$ 

```

Bias and Variance of Decision Trees

- ◆ Small changes of training data lead to big differences in final trees
 - ◆ Decision trees grown deep enough have typically:
 - low bias
 - high variance
- ⇒ **overfitting**
- ◆ Idea: *average multiple models* to reduce variance while (happily) not increasing bias much

Averaging Models

- ◆ Define regression model b as an average of K models:

$$b(x) = \frac{1}{K} \sum_{i=1}^K h_m^{(i)}(x)$$

trained using a set of i.i.d. datasets of size m : $\mathcal{D}^m = \{\mathcal{T}_1^m, \dots, \mathcal{T}_K^m\}$

- ◆ Note that $b(x)$ approximates the averaging model

$$g_m(x) = \mathbb{E}_{\mathcal{T}^m} \left(h_m(x) \right)$$

- ◆ The need for K different training sets \mathcal{T}_i^m is still impractical – why not to train a single model using $\mathcal{T}_1^m \cup \mathcal{T}_2^m \cup \dots \mathcal{T}_K^m$ instead of $b(x)$?

Averaging Models: Bias

- ◆ Bias remains unchanged when compared to a single model:

$$\begin{aligned}
 \text{bias}(x)^2 &= \mathbb{E}_{y|x} \left((g_m(x) - h^*(x))^2 \right) \\
 &= \mathbb{E}_{y|x} \left((\mathbb{E}_{\mathcal{D}^m} (b(x)) - h^*(x))^2 \right) \\
 &= \mathbb{E}_{y|x} \left(\left(\mathbb{E}_{\mathcal{D}^m} \left(\frac{1}{K} \sum_{i=1}^K h_m^{(i)}(x) \right) - h^*(x) \right)^2 \right) \\
 &= \mathbb{E}_{y|x} \left(\left(\frac{1}{K} \sum_{i=1}^K \mathbb{E}_{\mathcal{T}_i^m} \left(h_m^{(i)}(x) \right) - h^*(x) \right)^2 \right) \\
 &= \mathbb{E}_{y|x} \left((\mathbb{E}_{\mathcal{T}^m} (h_m(x)) - h^*(x))^2 \right)
 \end{aligned}$$

where $\mathbb{E}_{\mathcal{T}^m} (h_m(x))$ was the $g_m(x)$ defined for a single model $h_m(x)$

Averaging Models: Variance

- ◆ For uncorrelated component models $h_m^{(i)}(x)$:

$$\begin{aligned}\text{Var}_{\mathcal{D}^m}(b(x)) &= \text{Var}_{\mathcal{D}^m} \left(\frac{1}{K} \sum_{i=1}^K h_m^{(i)}(x) \right) \\ &= \frac{1}{K^2} \sum_{i=1}^K \text{Var}_{\mathcal{T}_i^m} \left(h_m^{(i)}(x) \right) = \frac{1}{K} \text{Var}_{\mathcal{T}^m} (h_m(x))\end{aligned}$$

which is a great improvement based on the **strong** assumption

- ◆ There is no improvement for maximum correlation, i.e., all component models equal: $h_m^{(i)}(x) = h_m(x)$ for $i = 1, \dots, K$, we get:

$$\text{Var}_{\mathcal{D}^m}(b(x)) = \text{Var}_{\mathcal{D}^m} \left(\frac{1}{K} \sum_{i=1}^K h_m^{(i)}(x) \right) = \text{Var}_{\mathcal{T}^m}(h_m(x))$$

⇒ we need to train **uncorrelated** (diverse) component models while **keeping their bias reasonably low**

Bootstrapping

- ◆ In practice we have only a single training dataset \mathcal{T}^m
- ◆ Bootstrapping is a method producing datasets \mathcal{T}_i^m for $i = 1, \dots, K$ by sampling \mathcal{T}^m uniformly with *replacement*
- ◆ Bootstrap datasets have the same size as the original dataset
 $|\mathcal{T}_i^m| = |\mathcal{T}^m|$
- ◆ \mathcal{T}_i^m is expected to have the fraction $1 - \frac{1}{e} \approx 63.2\%$ of unique samples from \mathcal{T}^m , others are duplicates (see seminar)

Bagging

- ◆ Bagging = Bootstrap AGGREGating [Breiman 1994]:
 1. Use bootstrapping to generate K datasets
 2. Train a model $h_m^{(i)}$ on each dataset \mathcal{T}_i^m
 3. Average the models
- ◆ When decision trees are used as the models \Rightarrow **random forests**
- ◆ Low bias is achieved by growing the trees to maximal depth
- ◆ Trees are decorrelated by:
 - training each tree on a different bootstrap dataset
 - randomization of split attribute selection

Random Forest Algorithm

1. For $i = 1 \dots K$:
 - (a) draw a bootstrap dataset \mathcal{T}_i^m from \mathcal{T}^m , $|\mathcal{T}_i^m| = |\mathcal{T}^m| = m$
 - (b) grow a tree $h_m^{(i)}$ using \mathcal{T}_i^m by recursively repeating the following until the minimum node size n_{\min} is reached:
 - i. select k attributes at random from the p attributes
 - ii. pick the best attribute and split-point among the k
 - iii. split the node into two daughter nodes
2. Output ensemble of trees $b(x)$ averaging $h_m^{(i)}$ (regression) or selecting a majority vote (classification)
 - ◆ Node size n_{\min} is the number of dataset samples associated with the node, limits tree depth

Out-of-Bag (OOB) Error

- ◆ Cheap way of generalization error assessment for bagging
- ◆ Bagging produces bootstrapped sets $\mathcal{T}_1^m, \mathcal{T}_2^m, \dots, \mathcal{T}_K^m$
- ◆ For each $(x_i, y_i) \in \mathcal{T}^m$ select only trees which were not trained on this sample: $H_i = \{h_m^{(j)} \mid (x_i, y_i) \notin \mathcal{T}_j^m\}$
- ◆ Average only the OOB trees in H_i when evaluating error for (x_i, y_i)
- ◆ Replacement for K-fold cross-validation

Feature Importance

- ◆ Random forests allow easy evaluation of feature importances
- ◆ Mean Decrease Impurity (MDI):
 - set $f_j = 0$ for all attributes $j = 1, \dots, p$
 - traverse all trees processing all internal nodes
 - for each node having a split attribute j add its *impurity decrease* multiplied by the proportion of the *node size* to f_j
- ◆ Mean Decrease Accuracy (MDA), permutation importance:
 - evaluate the forest using OOB
 - do the same with permuted values of an attribute j
 - watch decrease in accuracy: low decrease means unimportant feature

Boosting

- ◆ Sequentially train weak learners/predictors *low variance high bias*
- ◆ Subsequent predictors fix the mistakes of the previous ones reducing bias
- ◆ Methods discussed here:
 - AdaBoost
 - Forward Stagewise Additive Modeling
 - Gradient Boosting Machine
 - Gradient Boosted Trees

AdaBoost M1

Binary classifier: $\mathcal{Y} = \{-1, 1\}$

1. Initialize the weights $w_i = 1/m$ for $i = 1, 2, \dots, m$

2. For $k = 1$ to K :

(a) Fit a classifier $f_k(x; \theta_k)$ to the training data using loss weighted by w_i :

$$\theta_k = \operatorname{argmin}_{\theta} \sum_{i=1}^m w_i [y_i \neq f_k(x_i; \theta)]$$

(b) Compute the weighted error rate

$$\epsilon_k = \frac{\sum_{i=1}^m w_i [y_i \neq f_k(x_i; \theta_k)]}{\sum_{i=1}^m w_i}$$

(c) Compute the weight $\alpha_k = \log((1 - \epsilon_k)/\epsilon_k)$

(d) Set $w_i \leftarrow w_i \cdot \exp(\alpha_k \cdot [y_i \neq f_k(x_i; \theta_k)])$ for $i = 1, 2, \dots, m$

3. Return $h_m(x) = \operatorname{sign} \left[\sum_{k=1}^K \alpha_k f_k(x; \theta_k) \right]$

Forward Stagewise Additive Modeling (FSAM)

1. Initialize $f_0(x) = 0$

2. For $k = 1$ to K :

(a) Find

$$(\beta_k, \theta_k) = \operatorname{argmin}_{\beta, \theta} \sum_{i=1}^m \ell\left(y_i, f_{k-1}(x_i) + \beta b(x_i; \theta)\right)$$

where $b(x_i; \theta_k)$ is the *basis function* and β_k the corresponding coefficient

(b) Set $f_k(x) = f_{k-1}(x) + \beta_k b(x; \theta_k)$

3. Return $h_m(x) = f_K(x)$

AdaBoost is FSAM

- ◆ Claim: AdaBoost is FSAM using the exponential loss

$$\ell(y, f(x)) = \exp(-yf(x))$$

- ◆ We get:

$$\begin{aligned} (\beta_k, \theta_k) &= \operatorname{argmin}_{\beta, \theta} \sum_{i=1}^m \ell\left(y_i, f_{k-1}(x_i) + \beta b(x_i; \theta)\right) \\ &= \operatorname{argmin}_{\beta, \theta} \sum_{i=1}^m \exp\left(-y_i\left(f_{k-1}(x_i) + \beta b(x_i; \theta)\right)\right) \\ &= \operatorname{argmin}_{\beta, \theta} \sum_{i=1}^m w_i^{(k)} \exp\left(-y_i \beta b(x_i; \theta)\right), \end{aligned}$$

where $w_i^{(k)} = \exp(-y_i f_{k-1}(x_i))$

AdaBoost is FSAM II

- ◆ We can rearrange further:

$$\begin{aligned}
 (\beta_k, \theta_k) &= \operatorname{argmin}_{\beta, \theta} \sum_{i=1}^m w_i^{(k)} \exp \left(-y_i \beta b(x_i; \theta) \right) \\
 &= \operatorname{argmin}_{\beta, \theta} \left[e^{-\beta} \sum_{y_i=b(x_i; \theta)} w_i^{(k)} + e^{\beta} \sum_{y_i \neq b(x_i; \theta)} w_i^{(k)} \right] \\
 &= \operatorname{argmin}_{\beta, \theta} \left[\underbrace{(e^{\beta} - e^{-\beta})}_{>0 \text{ for } \beta > 0} \sum_{i=1}^m w_i^{(k)} [y_i \neq b(x_i; \theta)] + e^{-\beta} \sum_{i=1}^m w_i^{(k)} \right]
 \end{aligned}$$

- ◆ For any $\beta > 0$ we can minimize θ separately:

$$\theta_k = \operatorname{argmin}_{\theta} \sum_{i=1}^m w_i^{(k)} [y_i \neq b(x_i; \theta)] \quad (\text{same as } \textit{AdaBoost 2(a)})$$

AdaBoost is FSAM III

- ◆ Let's minimize

$$(e^\beta - e^{-\beta}) \sum_{i=1}^m w_i^{(k)} [y_i \neq b(x_i; \theta_k)] + e^{-\beta} \sum_{i=1}^m w_i^{(k)}$$

with respect to β

$$(e^{\beta_k} + e^{-\beta_k}) \sum_{i=1}^m w_i^{(k)} [y_i \neq b(x_i; \theta_k)] - e^{-\beta_k} \sum_{i=1}^m w_i^{(k)} = 0$$

$$(e^{\beta_k} + e^{-\beta_k}) \epsilon_k - e^{-\beta_k} = 0$$

where $\epsilon_k = \frac{\sum_{i=1}^m w_i [y_i \neq b(x_i; \theta_k)]}{\sum_{i=1}^m w_i}$ as in *AdaBoost 2(b)*

- ◆ Solving for β_k :

$$\beta_k = \frac{1}{2} \log \frac{1 - \epsilon_k}{\epsilon_k}$$

- ◆ Define $\alpha_k \triangleq 2\beta_k$ and compare to *AdaBoost 2(c)*

AdaBoost is FSAM IV

- ◆ We have $w_i^{(k)} = e^{-y_i f_{k-1}(x_i)}$ and $f_k(x) = f_{k-1}(x) + \beta_k b(x; \theta_k)$ so:

$$w_i^{(k+1)} = e^{-y_i (f_{k-1}(x_i) + \beta_k b(x_i; \theta_k))} = w_i^{(k)} \cdot e^{-y_i \beta_k b(x_i; \theta_k)}$$

- ◆ Finally $-y_i b(x_i; \theta_k) = 2 \cdot [y_i \neq b(x_i; \theta_k)] - 1$ gives the weight update:

$$w_i^{(k+1)} = w_i^{(k)} \cdot e^{\alpha_k [y_i \neq b(x_i; \theta_k)]} \cdot e^{-\beta_k}$$

corresponding to *AdaBoost* 2(*d*) up to the factor $e^{-\beta_k}$ which is same for all weights and hence has no effect

FSAM and Gradient Descent

- ◆ FSAM update looks very similar to the gradient descent one:

$$f_k(x) = f_{k-1}(x) + \beta_k b(x; \theta_k)$$

- ◆ Just think of
 - $\beta_k \approx$ step size (learning rate)
 - $b(x_i; \theta_k) \approx$ the negative of gradient

FSAM for Squared Loss

- ◆ Again consider regression with the squared loss:

$$\ell(y, f(x)) = (y - f(x))^2$$

- ◆ For FSAM we get:

$$\begin{aligned}\ell(y_i, f_k(x_i)) &= \ell(y_i, f_{k-1}(x_i) + \beta_k b(x_i; \theta_k)) \\ &= (y_i - f_{k-1}(x_i) - \beta_k b(x_i; \theta_k))^2 \\ &= (r_{ik} - \beta_k b(x_i; \theta_k))^2\end{aligned}$$

where $r_{ik} = y_i - f_{k-1}(x_i)$ is the *residual* of the current model for the i -th sample

- ◆ The task of FSAM is to fit the model $\beta_k b(x_i; \theta_k)$ to match the residuals
- ◆ The method is sometimes called the *least-squares boosting*

Gradient Boosting for Regression

- ◆ In case of regression with squared loss we minimize:

$$\mathcal{L} = \sum_{i=1}^m \ell(y_i, f(x_i)) = \sum_{i=1}^m \frac{1}{2} (y_i - f(x_i))^2,$$

which is same as minimization of the empirical risk

- ◆ We can treat $f(x_1), f(x_2), \dots, f(x_m)$ as parameters and take derivatives:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial f(x_i)} &= \frac{\partial \left(\sum_{j=1}^m \ell(y_j, f(x_j)) \right)}{\partial f(x_i)} = \frac{\partial \ell(y_i, f(x_i))}{\partial f(x_i)} \\ &= f(x_i) - y_i = -r_i \end{aligned}$$

- ◆ The *least-squares boosting* hence takes steps in the negative gradient direction where $r_i = -\frac{\partial \mathcal{L}}{\partial f(x_i)}$
- ◆ This approach can be generalized for any differentiable loss function!

Gradient Boosting Machine

1. Initialize $f_0(x) = 0$ or $f_0(x) = \operatorname{argmin}_{\gamma} \sum_{i=1}^m \ell(y_i, \gamma)$

2. For $k = 1$ to K :

(a) Compute:

$$\mathbf{g}_k = \left[\frac{\partial \ell(y_i, f_{k-1}(x_i))}{\partial f_{k-1}(x_i)} \right]_{i=1}^m$$

(b) Fit a regression model $b(\cdot; \theta)$ to $-\mathbf{g}_k$:

$$\theta_k = \operatorname{argmin}_{\theta} \sum_{i=1}^m [(-\mathbf{g}_k)_i - b(x_i; \theta)]^2$$

(c) Choose a fixed step size $\beta_k = \beta \in (0, 1]$ or use line search:

$$\beta_k = \operatorname{argmin}_{\beta > 0} \sum_{i=1}^m \ell\left(y_i, f_{k-1}(x_i) + \beta b(x_i; \theta_k)\right)$$

(d) Set $f_k(x) = f_{k-1}(x) + \beta_k b(x; \theta_k)$

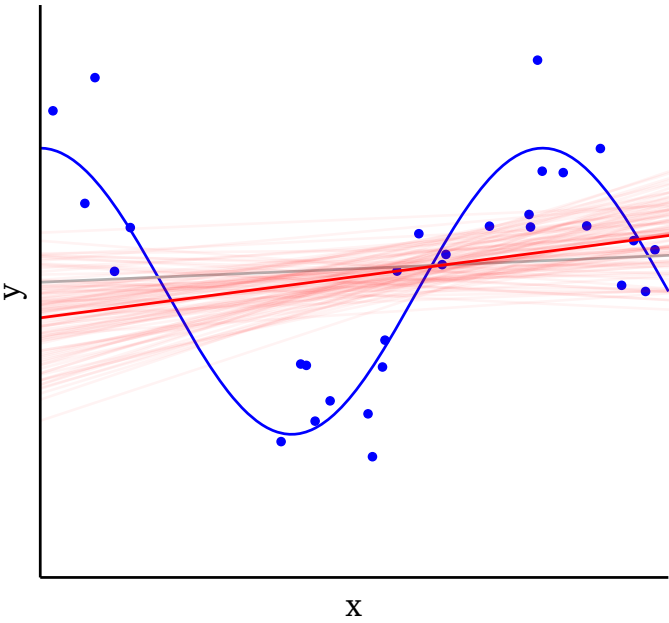
3. Return $h_m(x) = f_K(x)$

Gradient Boosted Trees

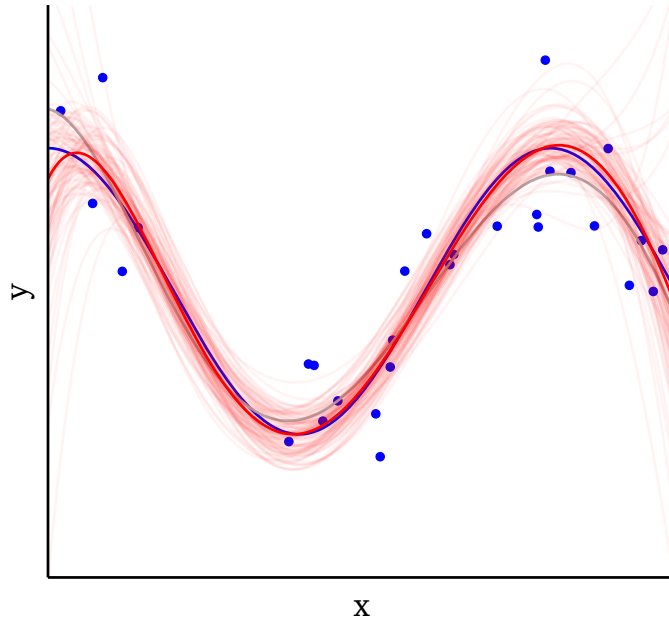
- ◆ Gradient Boosting Tree is GBM where all weak learners f_k are decision or regression trees
- ◆ Use limit on depth/number of leaves/node size for the weak learners \Rightarrow high bias
- ◆ Meta-parameters such as K (number of trees) and β (learning rate) have to be found using cross validation
- ◆ Model is built sequentially (unlike random forests)
- ◆ Highly optimized algorithms based on Gradient Boosting Trees:
 - XGBoost, LightGBM
 - parallelization, scalability, regularization



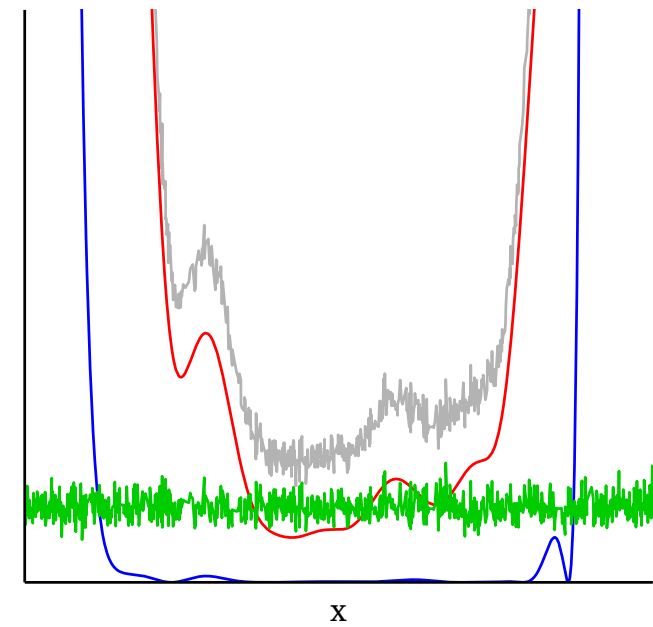
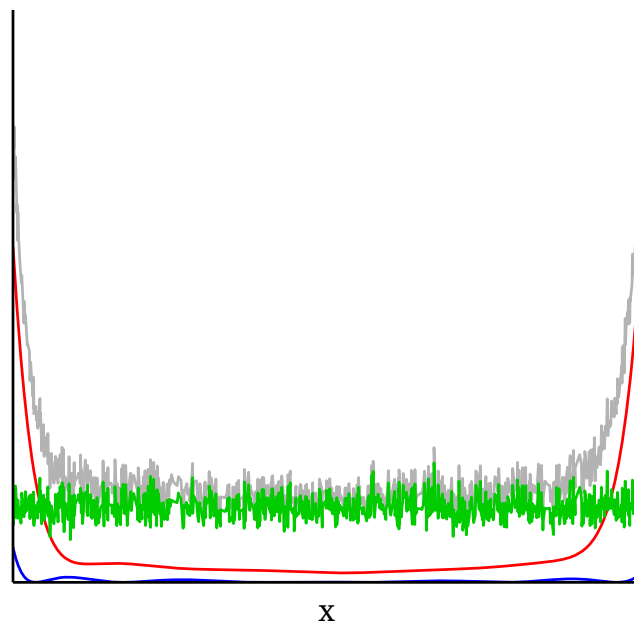
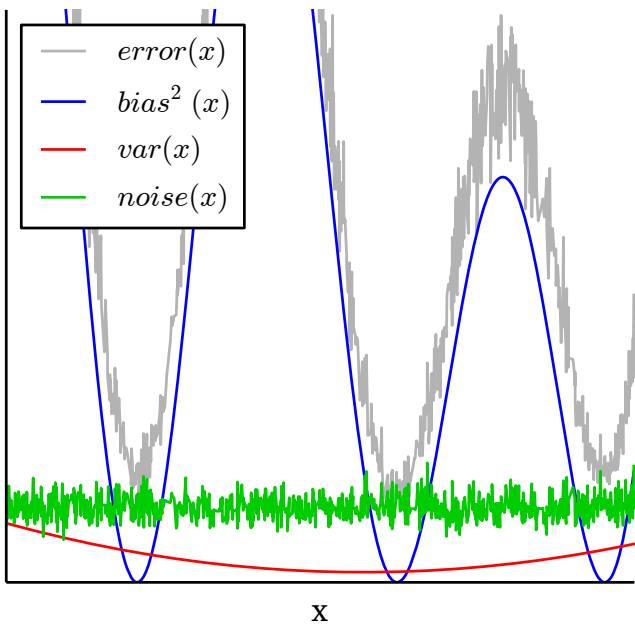
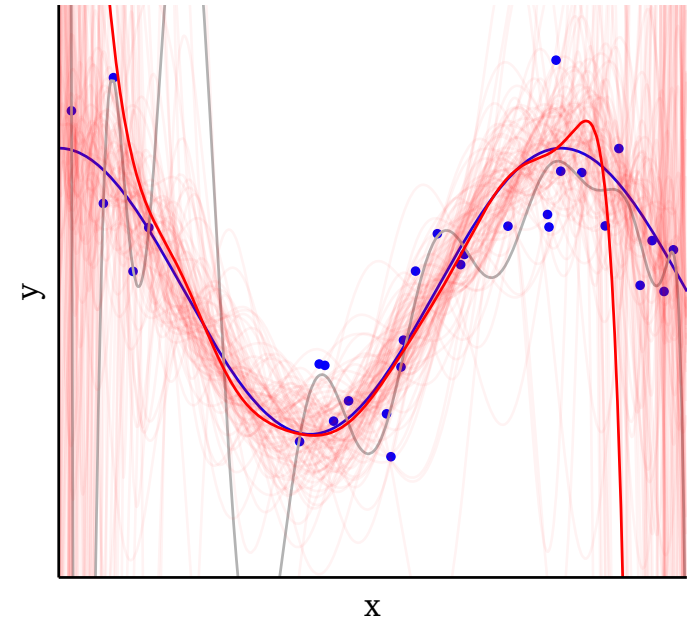
Degree = 1

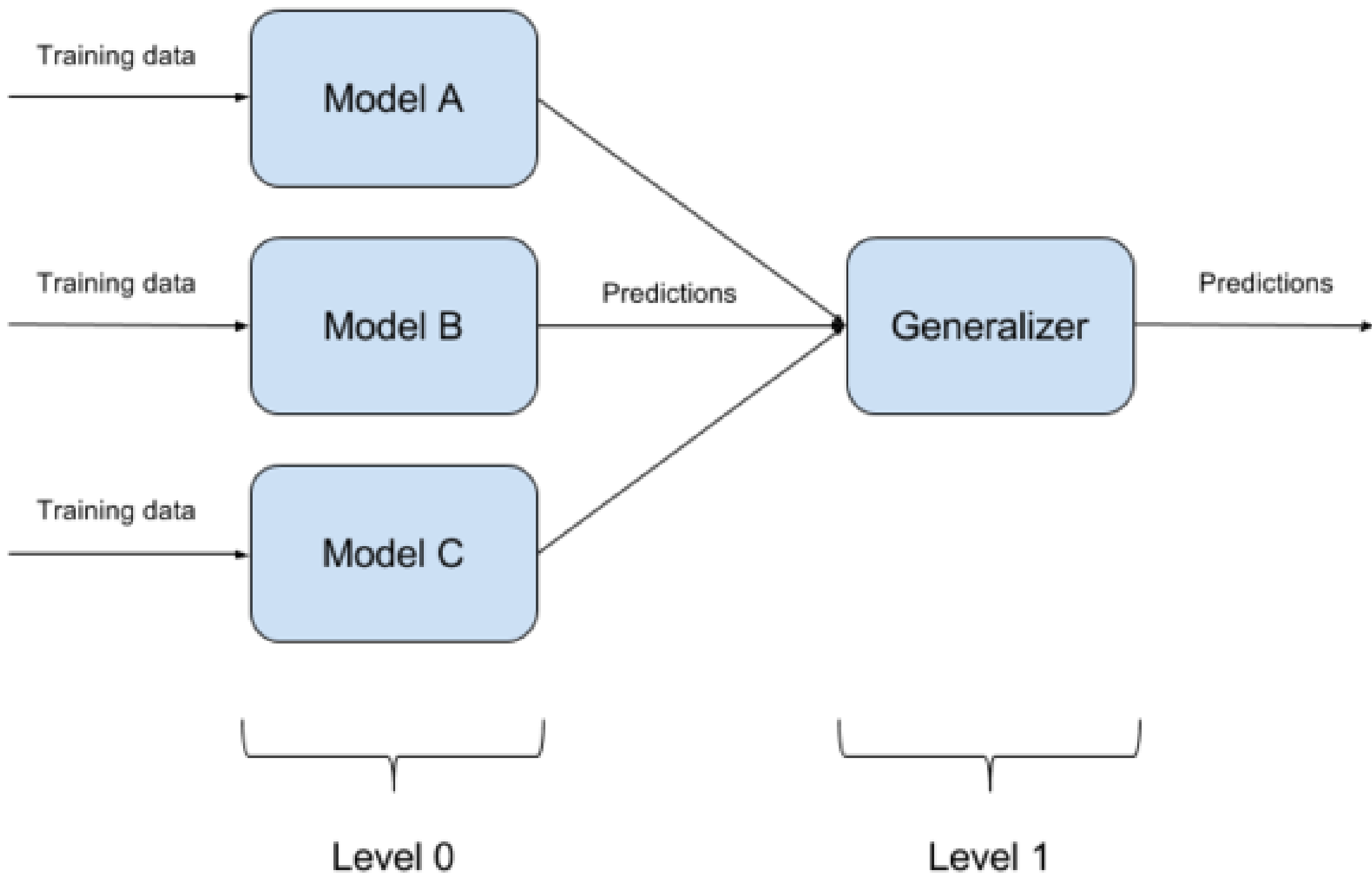


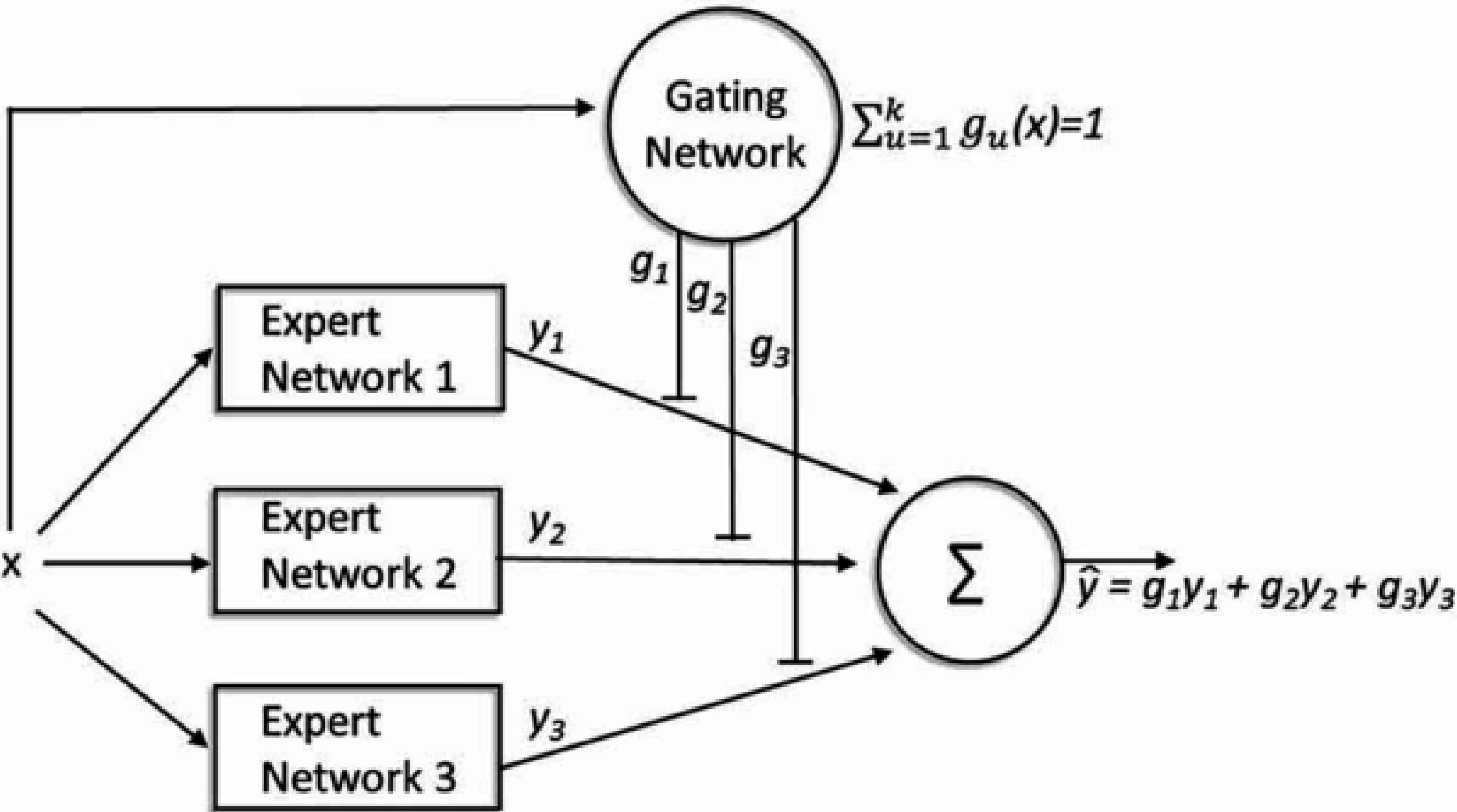
Degree = 5



Degree = 15







X_2

t_2

R_2

R_1

t_1

R_3

t_3

R_5

R_4

t_4

X_1

