Machine Learning Institute Certificate

Ensemble Learning

MLI
Module 1, Lecture 3
4th May 2021

[Adriano Koshiyama]

Overview

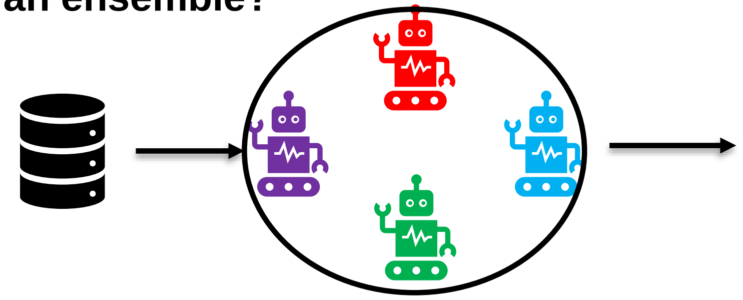
- **Ensemble Models**
  - Main Idea, 3 Main Types, Theoretical Framework, Empirical Evidence
- **Weak learner: Decision Tree**
  - Main Idea, Partitioning the Input/Output Space, Typical Components of a Decision Tree
  - Growing a Decision Tree, Impurity Functions, Stopping Criteria
  - Some Final Remarks
- **Random Forest**
  - Main Idea, Variance Reduction, Bagging and Subspace Projections
- **Gradient Boosting Trees**
  - Main Idea, Algorithm, Bias Reduction
- **A drill-down into some practical topics**
  - Random Forest vs Gradient Boosting Trees
  - Out-of-bag error
  - Feature Importance
  - Partial Dependence Plots
- **Stacking Approach**
  - Main idea, Some useful methods, Theoretical justification

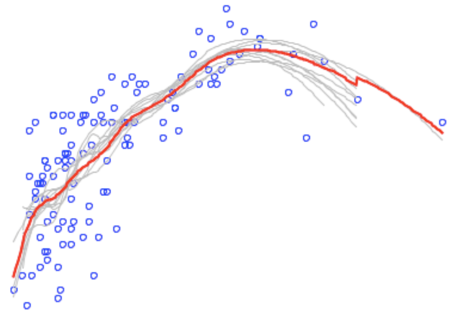
Ensemble Models

- Main Idea
- 3 Main Types
- Theoretical Framework
- Empirical Evidence

Ensemble Models – Main Idea

- **What is an ensemble?**

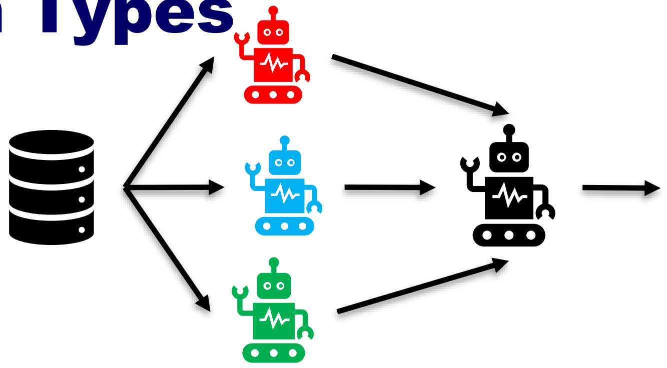




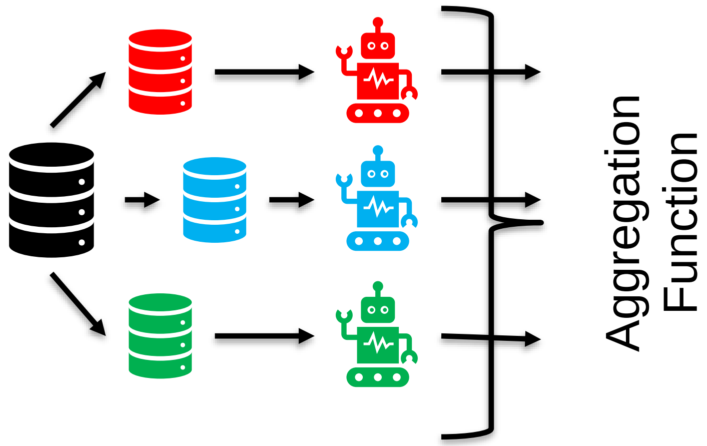
- A meta-model M that provides a consensus prediction $\hat{y}_i = M(\hat{f}_1, \dots, \hat{f}_J)$
- Usually every member of the ensemble/committee
  - can provide different answers for the same input
  - is individually weak (high-variance/bias)
- **Why ensemble anyway?**
  - Weak-learners tend to be easy to train, maintain, fine-tune and understand
  - Avoiding model selection, and sometimes hyper-parameter optimization
  - Creating nonlinearities and strengthening their performance by building an army and “averaging”

Ensemble Models – 3 Main Types

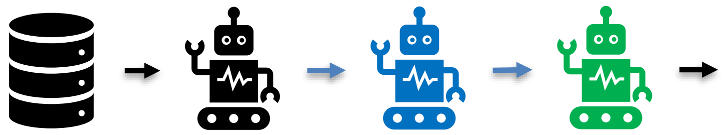
- **Stacking**
- Use a final model to combine the predictions (usually out-of-sample) offered by several models (committee)



- **Bagging**
- Split data in many batches using a resampling technique (bootstrap)
- For every batch train a weak-model
- Aggregate them, using mean/median



- **Boosting**
- Train models in a chain-style format, feeding the subsequent model with residuals and other inputs from the previous model



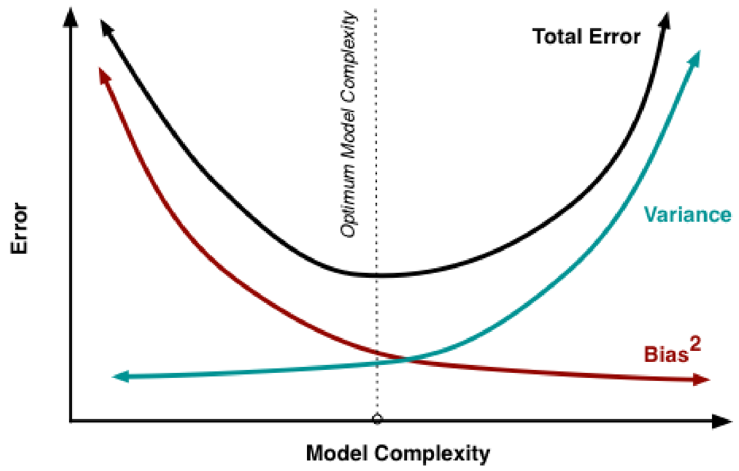
Ensemble Models – Theoretical Framework

- Each **Ensemble approach try to surf** from different directions on the Bias-Variance trade-off curve

$$\mathcal{E}(\hat{f}|X_*) = \mathbf{MSE}(\hat{f}|X_*) = \boldsymbol{Var}[\varepsilon_*] + \boldsymbol{Bias}(\hat{f}|X_*)^2 + \boldsymbol{Var}(\hat{f}|X_*)$$

- **Variance reduction**: Bagging
- **Sequential bias reduction**: Boosting
- **MSE reduction**: Stacking

- **Often used implementations**:
  - Random Forest
  - Gradient Boosting Trees
- **But others are available**
  - XGBoost
  - LightGBM
  - Conditional Forest, etc.



Ensemble Models – Empirical Evidence

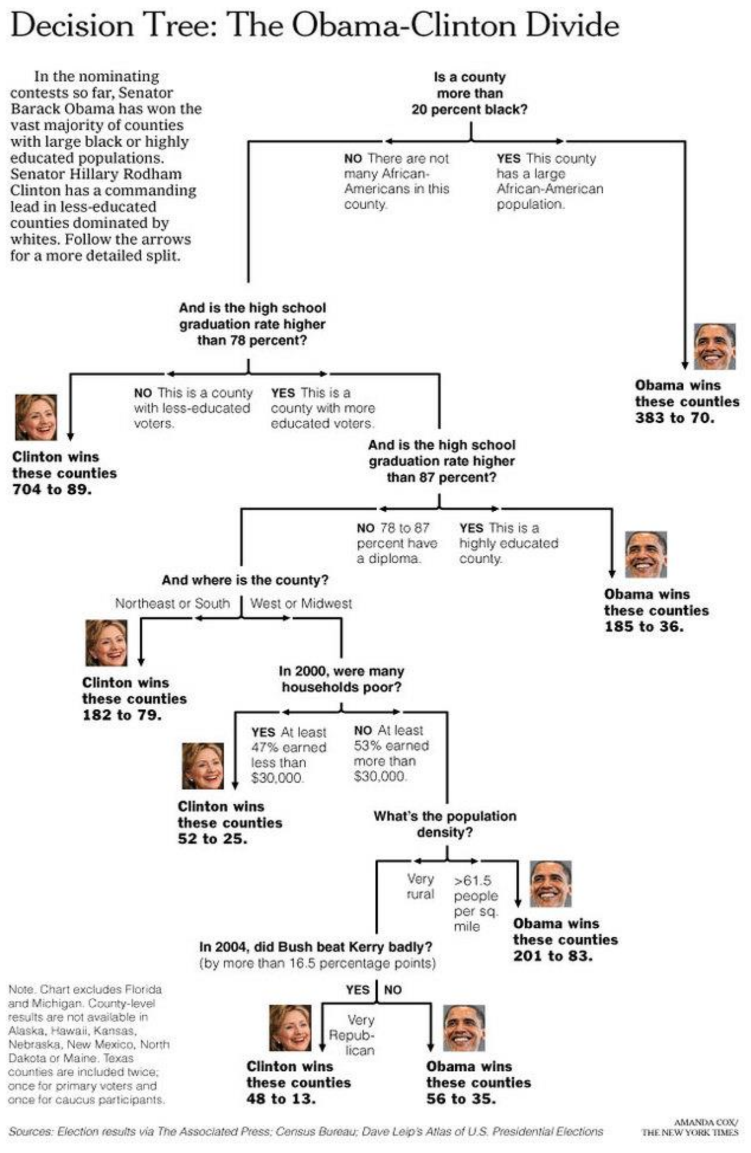
- **Empirical evidence pro-ensemble**
  - **M Competitions [Makridakis & Hibon, 2000; Makridakis et al., 2018]**
  - Large international forecasting competition (~200 participants)
    - M3 = 3,003 time series, M4 = 100,000 time series
    - Mostly monthly and “economic” data
    - Leaderboard was dominated by “**Combination**” methods
  - **“Do we need a hundred classifiers?” paper [Delgado et al., 2014]**
    - 179 classifiers applied to 121 datasets
      - In fact different implementations (software, variations, etc.) of 17 families/meta-classifiers
    - **Random Forest** was the best family in average
    - **Caveats**: most UCI obtained datasets; small sample size

Weak learner: Decision Tree

- Main Idea
- Partitioning the Input/Output Space
- Typical Components of a Decision Tree
- Growing a Decision Tree
- Impurity Functions
- Stopping Criteria
- Some Final Remarks

Decision Trees

- **Main Idea**
  - Split the feature space into a set of rectangles, and inside each subspace fit a simple predictive model
- **Some interesting characteristics**
  - Recursive algorithm
  - Greedy construction
  - Automatic feature selection
  - A good way to start data exploration and analysis
- **Main flavours available**
  - Classification Tree
  - Regression Tree


Decision Tree: The Obama-Clinton Divide

In the nominating contests so far, Senator Barack Obama has won the vast majority of counties with large black or highly educated populations. Senator Hillary Rodham Clinton has a commanding lead in less-educated counties dominated by whites. Follow the arrows for a more detailed split.

Note: Chart excludes Florida and Michigan. County-level results are not available in Alaska, Hawaii, Kansas, Nebraska, New Mexico, North Dakota or Maine. Texas counties are included twice: once for primary voters and once for caucus participants.

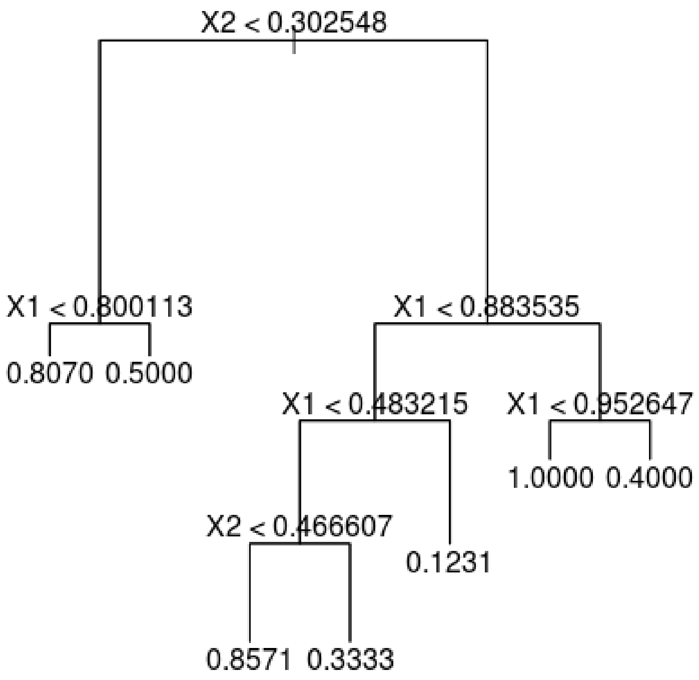
Sources: Election results via The Associated Press; Census Bureau; Dave Leip's Atlas of U.S. Presidential Elections

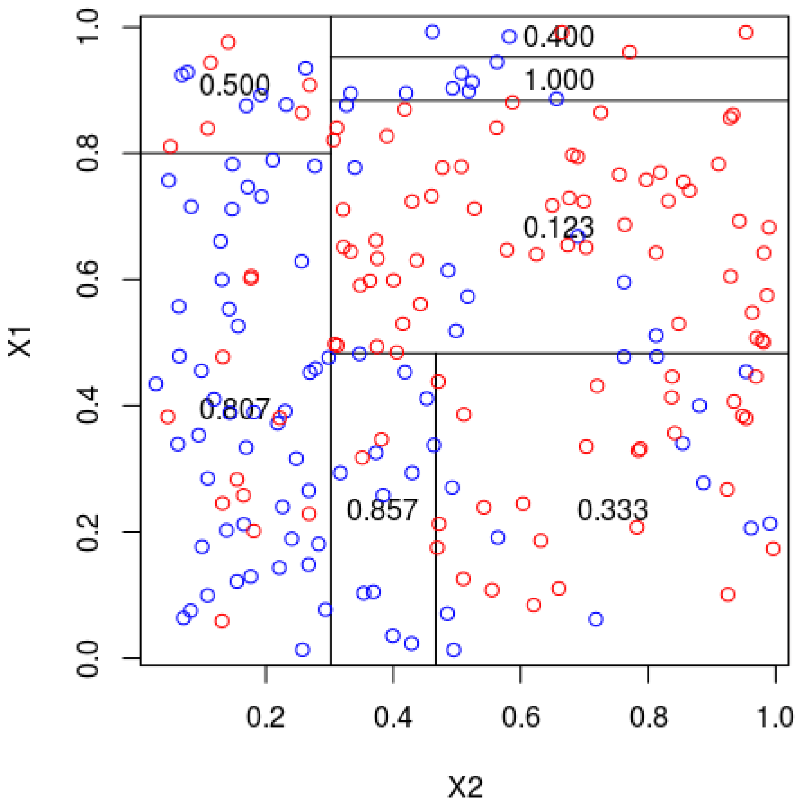
AMANDA COX/THE NEW YORK TIMES

Is a county more than 20 percent black?
NO There are not many African-Americans in this county.
YES This county has a large African-American population.
Obama wins these counties 383 to 70.
And is the high school graduation rate higher than 78 percent?
NO This is a county with less-educated voters.
YES This is a county with more educated voters.
Clinton wins these counties 704 to 89.
And is the high school graduation rate higher than 87 percent?
NO 78 to 87 percent have a diploma.
YES This is a highly educated county.
Obama wins these counties 185 to 36.
And where is the county?
Northeast or South
West or Midwest
Clinton wins these counties 182 to 79.
In 2000, were many households poor?
YES At least 47% earned less than $30,000.
NO At least 53% earned more than $30,000.
Clinton wins these counties 52 to 25.
What's the population density?
Very rural
>61.5 people per sq. mile
Obama wins these counties 201 to 83.
In 2004, did Bush beat Kerry badly? (by more than 16.5 percentage points)
YES
NO
Very Repub-lican
Clinton wins these counties 48 to 13.
Obama wins these counties 56 to 35.

Partitioning the Input/Output Space

- **Input and output space example, Regression Tree**
- Two input variables, X1 and X2, and a single output variable $Y \in [0,1]$





Typical Components of a Decision Tree

- There are **different algorithms** **[Breiman et al., 1984; Quinlan, 1993; Witten et al., 2011]** available to elaborate a **Decision Tree**
  - Chi-squared Automatic Interaction Detection (CHAID)
  - Iterative Dichotomiser 3 (ID3)
  - Hoeffding Trees
  - Classification and Regression Trees (CART)
  - C4.5 and C.50 (http://www.rulequest.com/Personal/)
- In summary, what **distinguish each one** of them are the process of:
  - Feature Selection
  - Partitioning Criteria
  - Stopping Criteria
- We **focus our attention on CART**, since
  - it is widely used and studied
  - have several computational implementations (sklearn for instance)

Growing a Decision Tree

- **Initial Information**
  - Dataset $D = \{(\boldsymbol{x_i}, y_i)\}_{i=1}^{n}$
  - **Stopping criteria**: maximum depth or number of nodes, and the minimum number of samples in each p-th node
  - **Impurity function** H that will measure the quality of a given split
- **CART Main Loop**

I. For each candidate split $\theta_{j,m} = (j,\ \tau_m)$, consisting of a feature j and a threshold τ_m, divide the data points n_m in node *m* available in two exclusive sets (regions):

$$R_{left}(\theta_{j,m}) = (x_i, y_i) \mid x_j \leq \tau_m$$

$$R_{right}(\theta_{j,m}) = (x_i, y_i) \mid x_j > \tau_m$$

II. Compute the impurity of this split at node *m* as

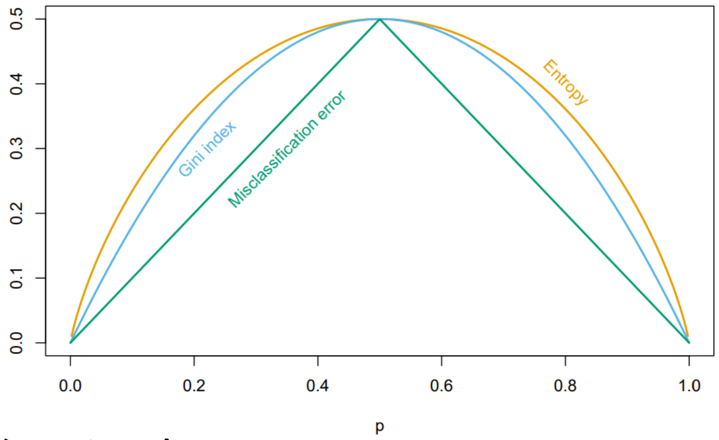
$$Q\left(R_{left}(\theta_{j,m}), R_{right}(\theta_{j,m})\right) = \frac{|R_{left}(\theta_{j,m})|}{n_m} H\left(R_{left}(\theta_{j,m})\right) + \frac{|R_{right}(\theta_{j,m})|}{n_m} H\left(R_{right}(\theta_{j,m})\right)$$

III. Select the $\theta^* = argmin_{\theta_{j,m}}\, Q\left(R_{left}(\theta_{j,m}), R_{right}(\theta_{j,m})\right)$

IV. Stop if some stopping criteria is reached, or n_p= 1; else return to (I)

- **We need to briefly discuss**: Impurity functions and Stopping Criteria

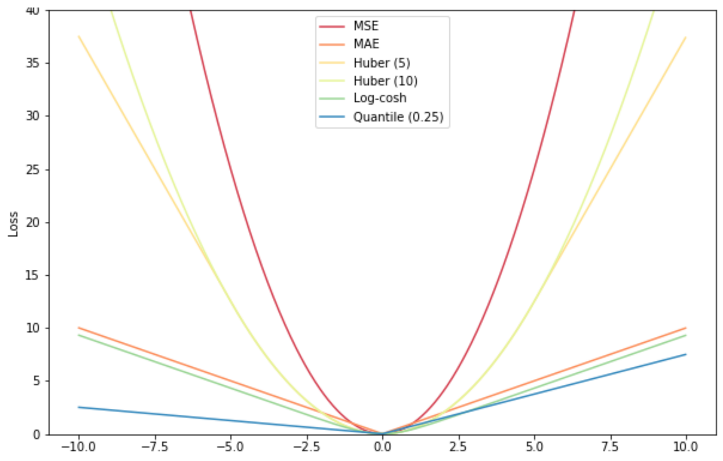
Impurity Functions



- **Many measures for Classification**
- Let $\hat{p}_{m,k} = \frac{1}{n_m} \sum_{x_i \in R_m} I(y_i = k)$ represent the proportion of class k observations in node *m*
  - *Misclassification Error*: $\frac{1}{n_m} \sum_{x_i \in R_m} I(y_i \neq k) = 1 - \hat{p}_{m,k}$
  - *Gini Index*: $\sum_k \hat{p}_{m,k} (1 - \hat{p}_{m,k})$
  - *Entropy*: $-\sum_k \hat{p}_{m,k} \log \hat{p}_{m,k}$
- **Usually MSE is used for Regression**

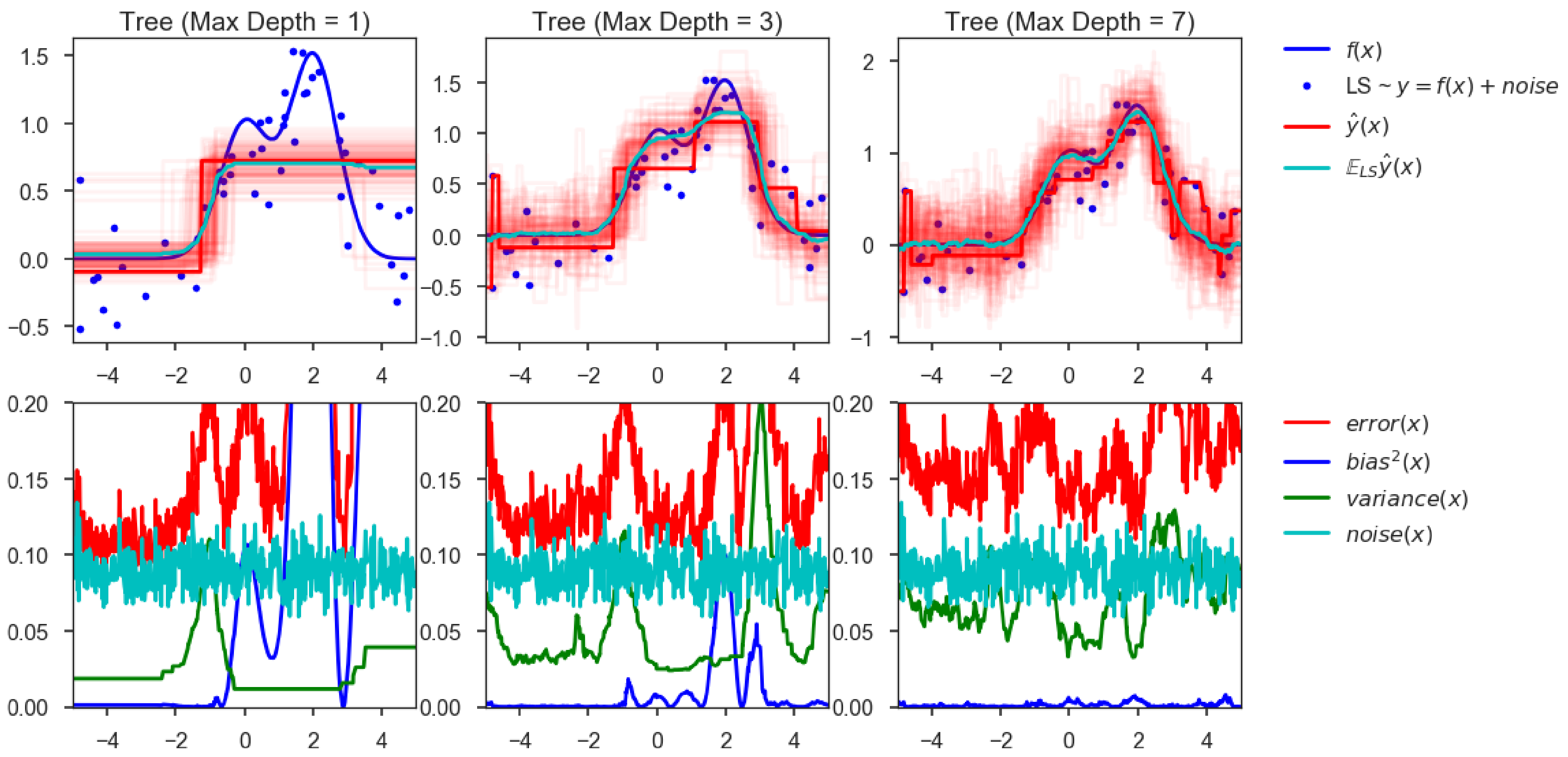
$$\min_{j,m} \left[\min_{c_1} \sum_{x_i \in R_{left}} (y_i - c_1)^2 + \min_{c_2} \sum_{x_i \in R_{right}} (y_i - c_2)^2 \right]$$

- Since c_1 and c_2 are constant, it is not hard to show that solution are the conditional average
- But other functions can be used
  - MAE, Huber, MAPE, etc.



Stopping Criteria – Bias-Variance Impact

- Tree (Max Depth = 1): 0.2585 (error) = **0.1413** (bias^2) + 0.0269 (var) + 0.0889 (noise)
- Tree (**Max Depth = 3**): **0.1482** (error) = 0.0089 (bias^2) + 0.0494 (var) + 0.0889 (noise)
- Tree (Max Depth = 7): 0.1781 (error) = 0.0009 (bias^2) + **0.0873** (var) + 0.0889 (noise)



Decision Tree – Some Final Remarks

Feature	Neural Nets	SVM	Trees
Natural handling of data of “mixed” type	fair	fair	good
Handling of missing values	poor	poor	good
Robustness to outliers in input space	poor	poor	good
Insensitive to monotone transformations of inputs	poor	poor	good
Computational scalability (large N)	fair	fair	good
Ability to deal with irrelevant inputs	poor	poor	good
Ability to extract linear combination of features	good	good	poor
Interpretability	poor	poor	fair
Predictive power	good	good	poor

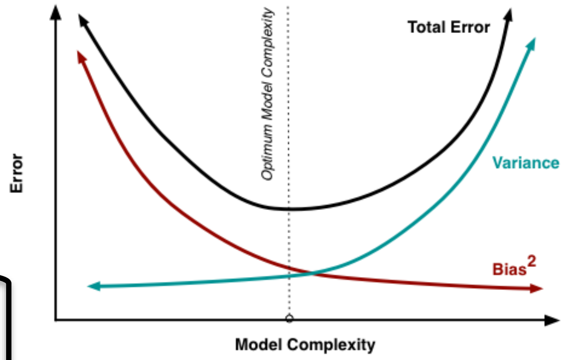
extracted and adapted from Elements of Statistical Learning

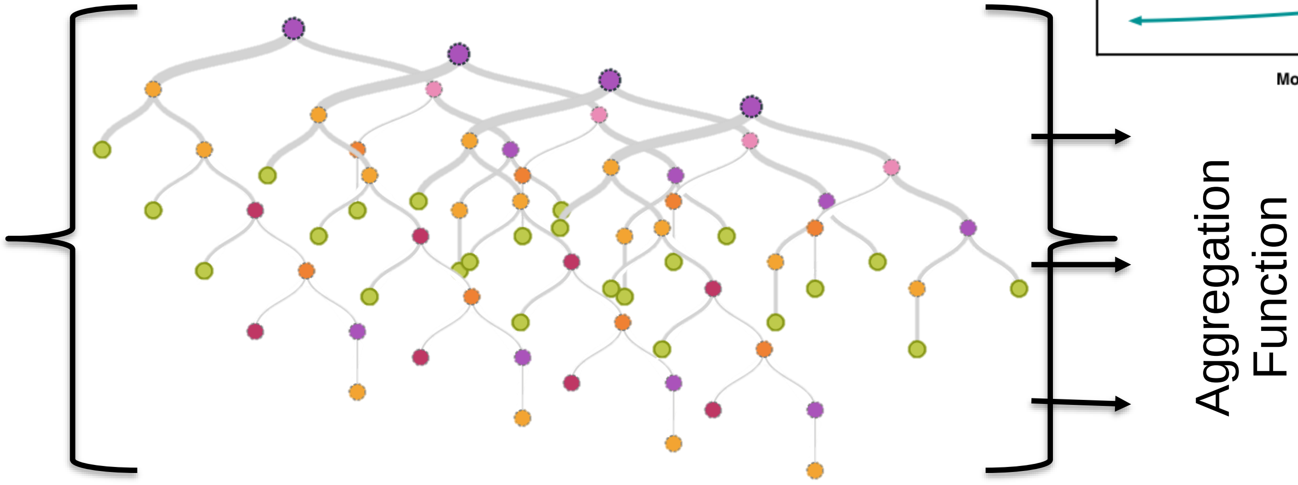
Random Forest

- Main Idea
- Variance Reduction
- Bagging and Subspace Projections

Random Forest

- **Main Idea**: Fit a set of deep decision trees, each of them with different parts of the dataset





- **Aggregate** each tree output by taking the mean or median
- **Main argument**: though each tree has individually *High Variance and Low Bias*, by a **diversification trick** we can **reduce the variance** in aggregate
- **Issue**: how to feed each tree with a diverse set of datapoints?

Random Forest – Variance Reduction

- **Main mathematical device**: very similar to Markowitz's Optimal Portfolio
- Suppose we are provided with a correlated sequence of $\hat{Y}_1, \hat{Y}_2, \dots, \hat{Y}_T$, then

$$\boldsymbol{V}\left[\frac{1}{T}\ \hat{Y}_t\right] = \frac{1}{T^2}\sum_{t=1}^{T}\boldsymbol{V}[\hat{Y}_t] + \frac{2}{T^2}\sum_{t=1}^{T}\boldsymbol{Cov}[\hat{Y}_t, \hat{Y}_k]$$

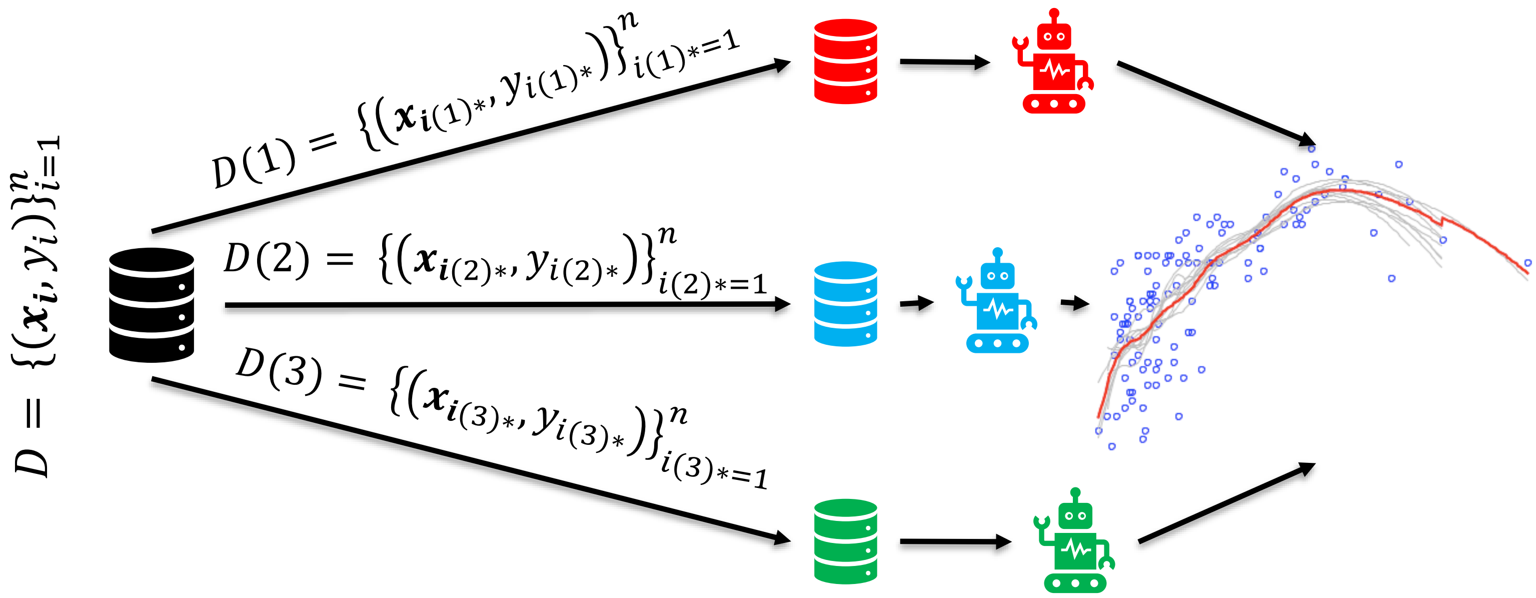
- If we assume, for analytical purpose, that $\mathbf{V}[\hat{Y}_t] = \sigma^2$ and $\mathbf{Cov}[\hat{Y}_t, \hat{Y}_{\boldsymbol{k}}] = \rho\sigma^2$

$$\boldsymbol{V}\left[\frac{1}{T}\ \hat{Y}_t\right] = \frac{\sigma^2}{T} + \frac{T-1}{T}\rho\sigma^2 = \sigma^2\left(\frac{1}{T} + \frac{T-1}{T}\rho\right) \leq \sigma^2$$

- Hence, by Bias-Variance argument, we are able to reduce the variance of the ensemble, by averaging many slightly correlated predictors
- **Remining issue**: reducing the intra-correlation component
    - Boostrap Aggregation (Bagging)
    - Random Subspace Projection (Feature Elimination)
- Overall, diversifying as possible the pool of predictors – this is also true for other off-the-shelf models: logistic regression, shallow neural networks, etc.

Bagging and Random Subspace Projections

- **Bootstrap aggregation (Bagging)**
- Take *B* bootstrap samples (sampling with replacement) from the dataset, fit a model with this new sample, and average their outputs

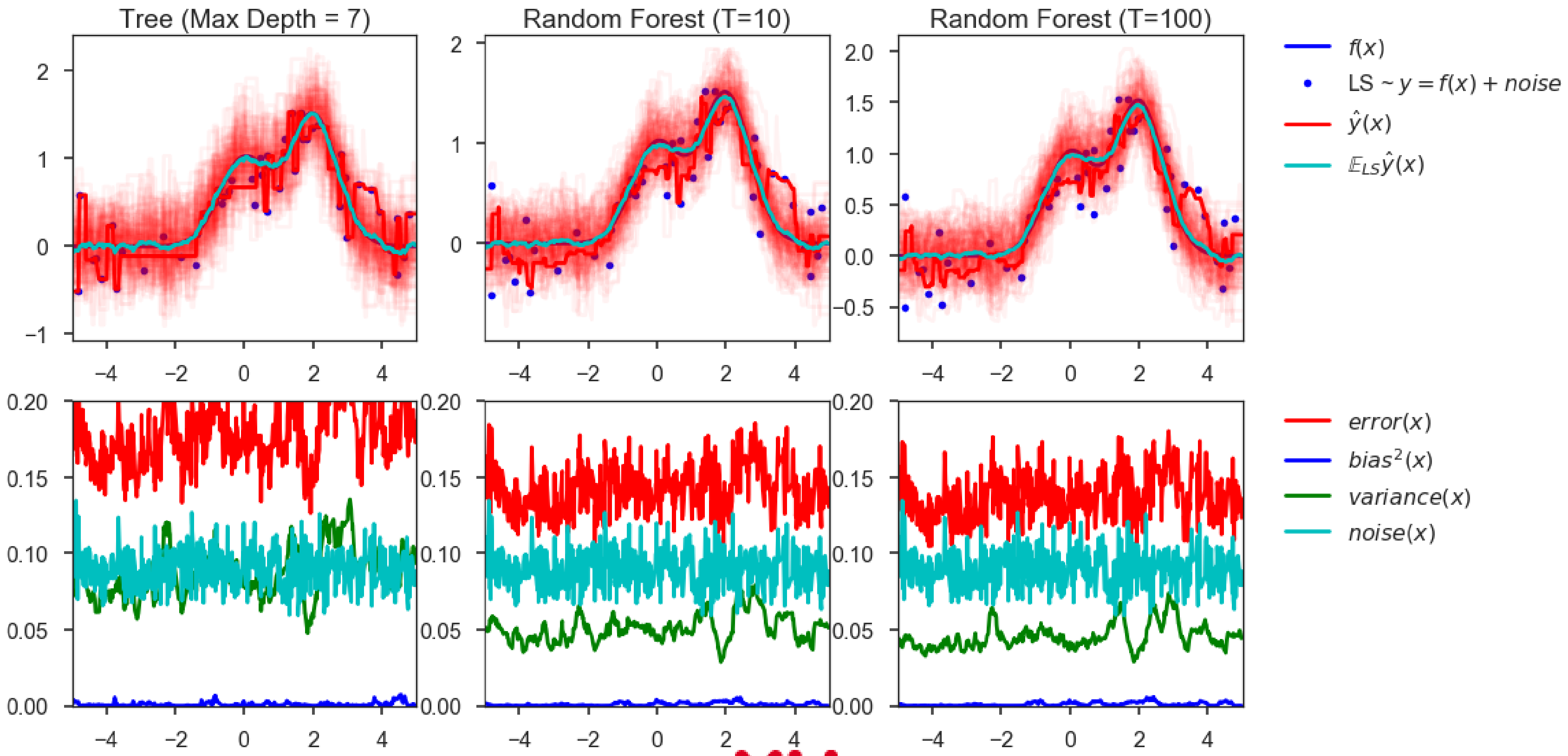


$D = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$
$D(1) = \{(\boldsymbol{x}_{i(1)*}, y_{i(1)*})\}_{i(1)*=1}^n$
$D(2) = \{(\boldsymbol{x}_{i(2)*}, y_{i(2)*})\}_{i(2)*=1}^n$
$D(3) = \{(\boldsymbol{x}_{i(3)*}, y_{i(3)*})\}_{i(3)*=1}^n$

- **Random subspace projections**: randomly delete some of the features during the process of tree growing

Random Forest

- Tree (Max Depth = 7): 0.1781 (error) = **0.0009** (bias^2) + 0.0873 (var) + 0.0889 (noise)
- Random Forest (T=10): 0.1420 (error) = 0.0010 (bias^2) + 0.0511 (var) + 0.0889 (noise)
- Random Forest (**T=100**): **0.1366** (error) = 0.0010 (bias^2) + **0.0457** (var) + 0.0889 (noise)

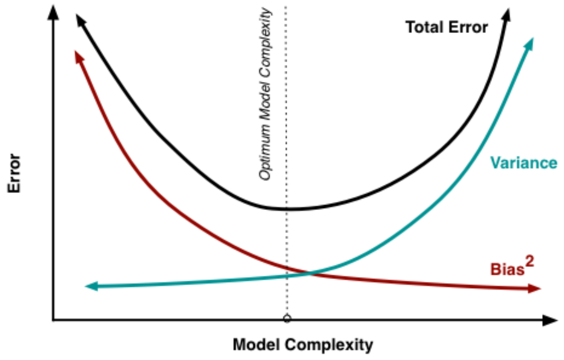


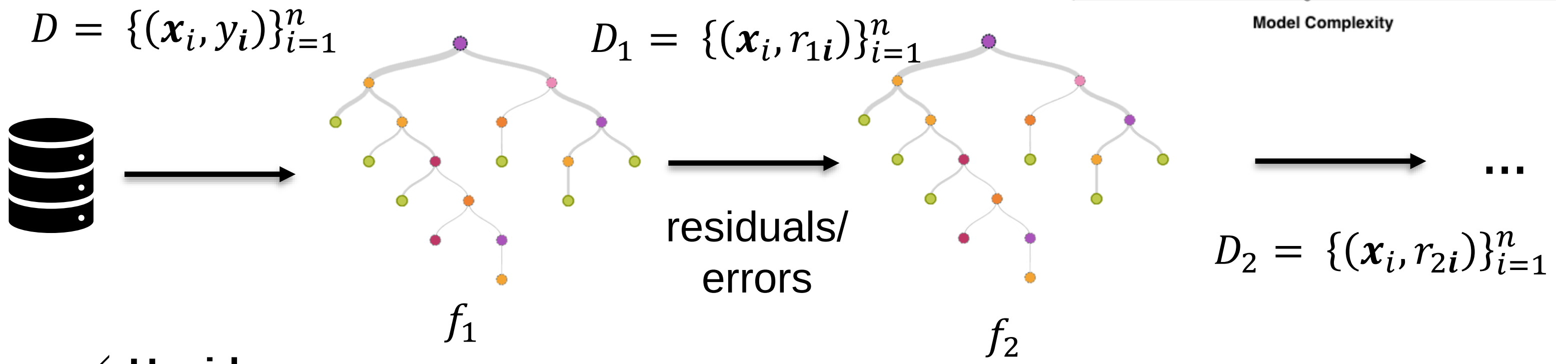
Gradient Boosting Trees

- Main Idea
- Algorithm
- Bias Reduction

Gradient Boosting Trees

- **Main Idea**: Fit a set of shallow decision trees, in a chain-style format, feeding the subsequent model with residuals and other inputs




$D = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n}$
$D_1 = \{(\boldsymbol{x}_i, r_{1i})\}_{i=1}^{n}$
residuals/ errors
...
$D_2 = \{(\boldsymbol{x}_i, r_{2i})\}_{i=1}^{n}$
f_1
f_2

- **Upsides**
  - Robustness to outliers in output space (via robust loss functions)
  - Off-the-shelf and generalized to any loss function
- **Downsides**
  - Scalability, due to the sequential nature of boosting it can hardly be parallelized.

Gradient Boosting Trees - Algorithm

- **Main Algorithm:** Base learners (trees): $f_1, \dots, f_T$ and $D = \{(\boldsymbol{x_i}, y_i)\}_{i=1}^{n}$
- Initialize $F_0 = constant$; For t = 1, …., T, do:
  - I. Fit f_t using the residual as the output $(\boldsymbol{x}_1, r_1), \dots, (\boldsymbol{x}_n, r_n)$
    where $r_i = -\frac{dL(y_i, \hat{y}_i)}{d\hat{y}_i}\Big|_{\hat{y}_i = F_{t-1}(\boldsymbol{x}_i)}$; for **MSE**: $r_i = (y_i - F_{t-1}(\boldsymbol{x}))$
  - *II.* $\gamma_t \coloneqq argmin_\gamma \sum_i L\big(y_i,\ \hat{y}_i = F_{t-1}(\boldsymbol{x}_i) + \gamma f_t(\boldsymbol{x}_i)\big)$ (update coefficient)
  - *III.* $F_t = F_{t-1} + \gamma_t \cdot \epsilon \cdot f_t$ (gradient update with learning rate ϵ)

- **Gradient Boosting Trees hyperparameters**
  - **Max depth size** (usually small to prevent variance inflation)
  - **Learning rate** (very small -- help to reduce variance, offset bias reduction)
  - **Number of trees** (has a negative relationship with learning rate)
- **Further good additions**:
  - Bagging and Random subspace projection
  - Find optimal values at the terminal node level

Gradient Boosting Trees – Why it works?

- **Main Idea**: Sequential Bias Reduction
- We can express the ***MSE*** of a Gradient Boosting Tree model, by

$$\boldsymbol{MSE}\left[\sum_{t=1}^{T}\hat{Y}_t\right]=\boldsymbol{E}\left[\left(Y-\sum_{t=1}^{T}\hat{Y}_t\right)^2\right]=\boldsymbol{V}[\varepsilon]+\boldsymbol{Bias}\left[\sum_{t=1}^{T}\hat{Y}_t\right]^2+\boldsymbol{V}\left[\sum_{t=1}^{T}\hat{Y}_t\right]$$

- $\hat{Y}_t$ is the output of a tree fitted at step *t* with residuals $r_{t-1}=Y-\hat{Y}_{t-1}$
- Considering the case $T=2$, the above expression simplifies to

$$\boldsymbol{E}\left[\left(Y-\hat{Y}_1-\hat{Y}_2\right)^2\right]=\boldsymbol{E}\left[\left(r_1-\hat{Y}_2\right)^2\right]=\boldsymbol{E}[r_1^2]+\boldsymbol{E}\left[\hat{Y}_2^2\right]-2\boldsymbol{E}[r_1\hat{Y}_2]$$

- With some tedious algebra, it is possible to rewrite it as

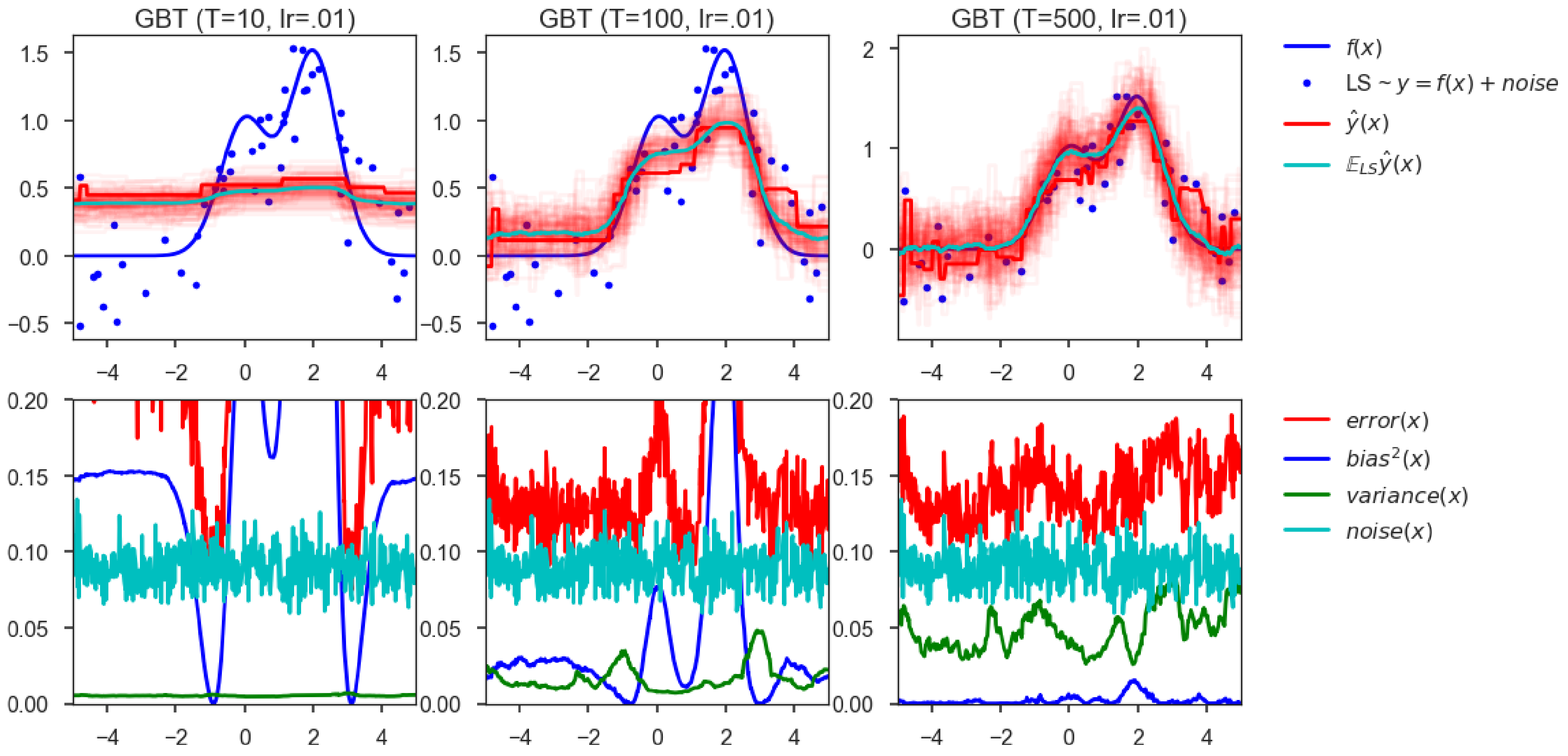
$$\boldsymbol{E}\left[\left(Y-\hat{Y}_1-\hat{Y}_2\right)^2\right]=\boldsymbol{V}[\varepsilon]+\boldsymbol{V}\left[\hat{Y}_1+\hat{Y}_2\right]+\left(\boldsymbol{Bias}\left(\hat{Y}_1\right)-\boldsymbol{E}\left[\hat{Y}_2\right]\right)^2-2\boldsymbol{Cov}\left(Y,\hat{Y}_2\right)$$

- Hence, apart from a variance inflation $\boldsymbol{V}\left[\hat{Y}_1+\hat{Y}_2\right]$, Bias is reduced sequentially at every iteration of the algorithm

Machine Learning Institute Certificate

Gradient Boosting Trees

- GBT (T=10, lr=.01): 0.3186 (error) = 0.2247 (bias^2) **+ 0.0052 (var)** + 0.0889 (noise)
- GBT (T=100, lr=.01): 0.1508 (error) = 0.0448 (bias^2) + 0.0165 (var) + 0.0889 (noise)
- GBT (T=500, lr=.01): **0.1407** (error) = **0.0020 (bias^2)** + 0.0489 (var) + 0.0889 (noise)



A drill-down in some practical aspects

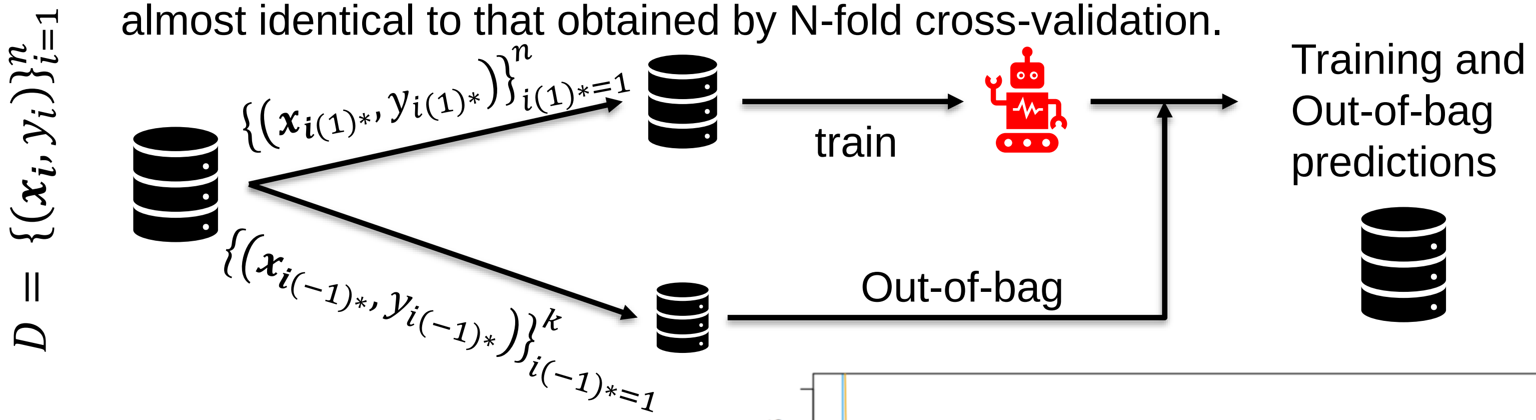
- Random Forest vs Gradient Boosting Trees
- Out-of-bag error
- Feature Importance
- Partial Dependence Plots

Random Forest vs Gradient Boosting Trees

Characteristic	Random Forest	Grad Boost Trees
Performance	Performs well across a wide range of tasks; with a large ensemble size, it is robust at tree level to parameter misspecification	Has more upside chance to outperform Random Forest, since it has a more flexible structure; it requires more time and ability to fine-tune
Scalability	Can be parallelized; deep trees require more time to grow	Serial construction; there are some "light" implementations available
Main Hyperparameters	Number of Trees	Number of Trees, Tree Depth, Shrinkage Factor
Bias-Variance	Variance Reduction	Bias Reduction

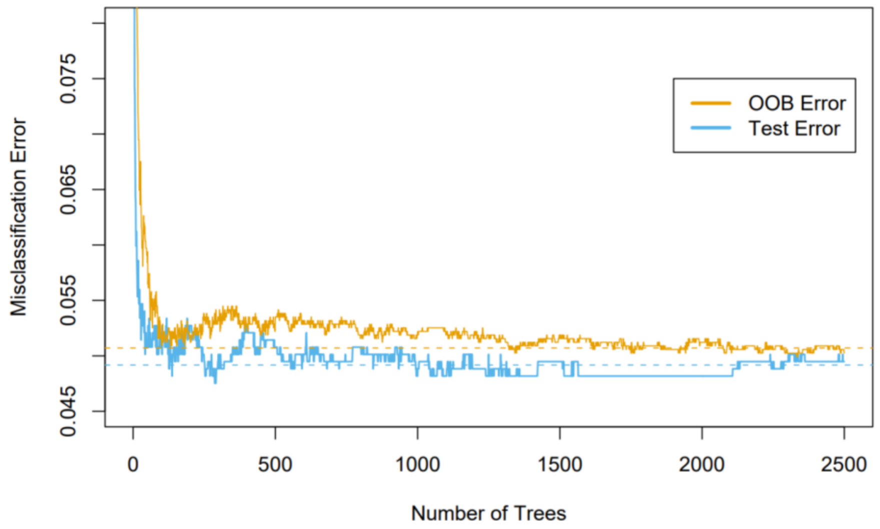
Out-of-bag Error

- For each **pair** (x_i, y_i), **query and average only** those trees that **did not used this** pair for training. An **out-of-bag (oob) error** estimate is almost identical to that obtained by N-fold cross-validation.



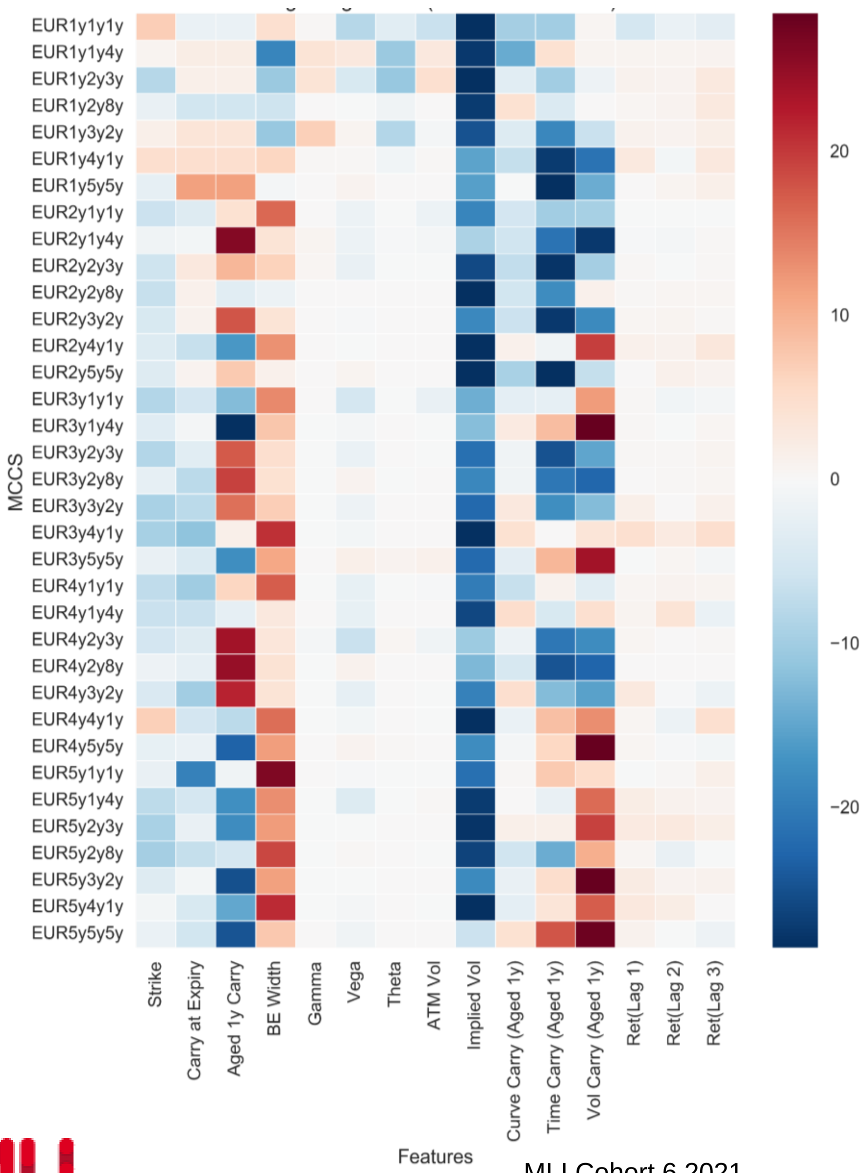
$D = \{(x_i, y_i)\}_{i=1}^{n}$
$\{(x_{i(1)*}, y_{i(1)*})\}_{i(1)*=1}^{n}$
train
$\{(x_{i(-1)*}, y_{i(-1)*})\}_{i(-1)*=1}^{k}$
Out-of-bag
Training and Out-of-bag predictions

- Hence unlike many other nonlinear estimators, random forests can be fit in one sequence, with cross-validation being performed along the way. Once the oob error stabilizes, the training can be terminated.



Feature Importance

- Often, the input predictor variables are seldom equally relevant. Often, only a few of them have substantial relevance
- **Feature Importance** metrics is a way in which "black-box" models become transparent
  - Weight of each feature
  - Positive/negative relationship
- **Some ways to calculate**
  - Weighted node impurity reduction
  - Out-of-bag error degradation via Row permutation
- **Caveat**: will underestimate "importance" in the presence of correlated variables



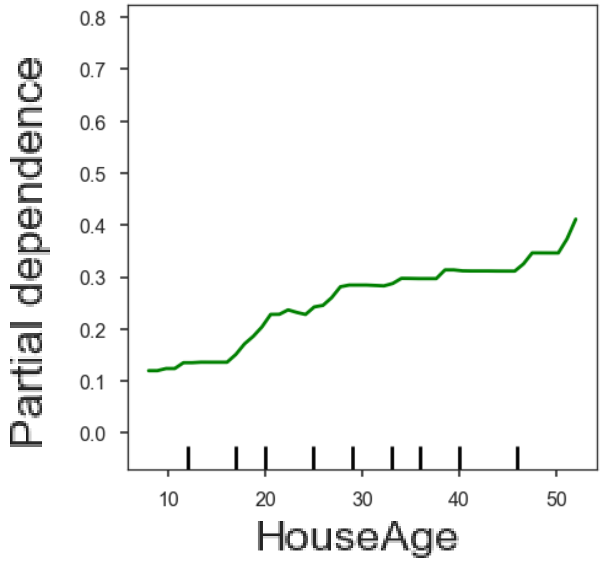
Partial Dependence Plots

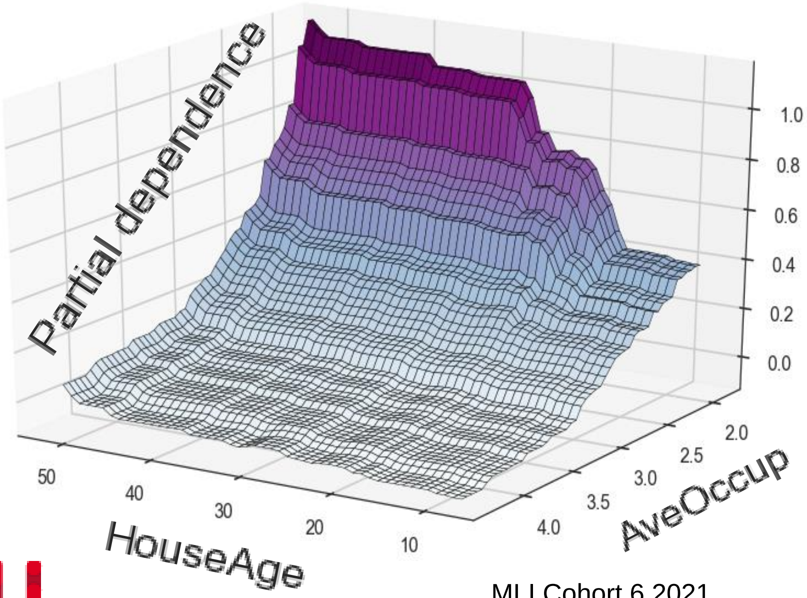
- Partial dependence functions can be used to interpret the results of any “black box” learning method.
- Given a feature set *S* (feature A, say) and its complement *C*, partial dependence can be estimated by:

$$\hat{y}_S(\boldsymbol{x}_S) = \frac{1}{n}\sum_{i=1}^{n} F(\boldsymbol{x}_S, x_{iC})$$

for every value $\boldsymbol{x}_S$ in the range

- For a given model, it involves a heavy amount of computation
- However, for decision trees it is rapidly computed from the tree itself without reference to the data



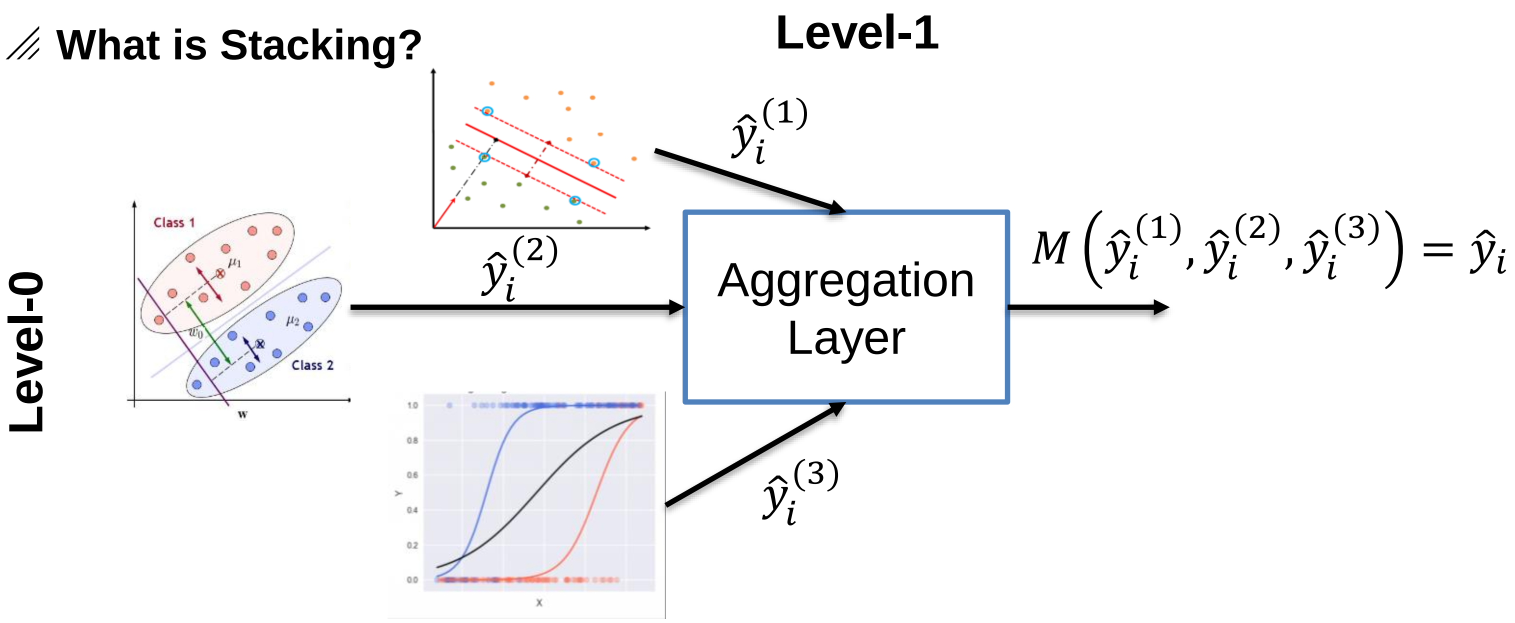


Stacking Approach

- Main idea
- Some useful methods
- Theoretical justification

Stacking Approach

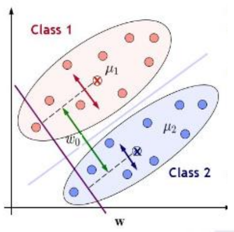
- **What is Stacking?**

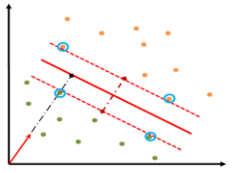


- **Why and when to stack anyway?**
  - A good way to avoid model selection
  - Need to add legacy models and experts predictions
  - “Don’t put all eggs in one basket” principle

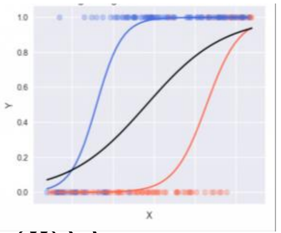
Aggregation Layer – Some useful methods

- **Stacked Regressions ([Wolpert, 1992] and [Breiman, 1996])**
- Split a dataset $D = \{(\boldsymbol{x_i}, y_i)\}_{i=1}^{n}$ in
  - Level-0 Dataset: $D_0 = \{(\boldsymbol{x_i}, y_i)\}_{i=1}^{n_0}$
  - Level-1 Dataset: $D_1 = \{(\boldsymbol{x_p}, y_p)\}_{p=1}^{n_1}$
- **Level-0 Model and Data**
  - For all models $f^{(1)}, \ldots, f^{(K)}$, fit and validate using D_0
  - From D_1, get $\hat{y}_p^{(1)} = f^{(1)}(\boldsymbol{x}_p), \ldots, \hat{y}_p^{(K)} = f^{(K)}(\boldsymbol{x}_p)$
- **Level-1 Model and Data**
  - Traditional: $\min_{\boldsymbol{w}} \boldsymbol{MSE}(\boldsymbol{w}) = \sum_{p}^{n_1} \left(y_p - \left(w_1 \hat{y}_p^{(1)} + \cdots + w_K \hat{y}_p^{(K)} \right) \right)$
  - Works better: $\min_{\boldsymbol{w} \geq \boldsymbol{0}} \boldsymbol{AMSE}(\boldsymbol{w}) = \boldsymbol{MSE}(\boldsymbol{w}) + \lambda(w_1^2 + \cdots + w_K^2)$
- **Why different levels?**
  - To avoid reusing the same data (overfitting)
  - For performance analysis purpose





Level-0



Level-1

Aggregation Layer

Aggregation Layer – Some useful methods

- **Some popular weighting methods in Forecasting**
  - Bates and Granger (1969)

$$w_k = \left(\frac{\boldsymbol{MSE}\left(f^{(k)}; D_1\right)}{\boldsymbol{MSE}\left(f^{(1)}; D_1\right) + \cdots + \boldsymbol{MSE}\left(f^{(K)}; D_1\right)}\right)^{-1}$$

  - Granger and Newbold (1974)

$$\boldsymbol{w} = \left(\mathbf{1}^{\boldsymbol{T}}\boldsymbol{\Sigma}^{-1}\mathbf{1}\right)^{-1}\boldsymbol{\Sigma}^{-1}\mathbf{1}, \text{ with } \boldsymbol{\Sigma}_{\mathrm{k,m}} = \mathrm{Cov}[(\hat{y}_{ik} - y_{\mathrm{i}})(\hat{y}_{im} - y_{\mathrm{i}})],$$

$$\mathbf{1} = [1,1,\ldots,1]^T \text{ and subject to } \mathbf{1}^{\boldsymbol{T}}\boldsymbol{w} = 1$$

  - Granger and Ramanathan (1984)

$$\boldsymbol{w} = \left(\boldsymbol{F}^{\boldsymbol{T}}\boldsymbol{F}\right)^{-1}\boldsymbol{F}^{\boldsymbol{T}}\boldsymbol{y} \text{ , with } \boldsymbol{F} = [\hat{\boldsymbol{y}}_{\mathbf{1}}, \ldots, \hat{\boldsymbol{y}}_K]^T, \boldsymbol{y} = \left[y_1, \ldots, y_{n_1}\right]^T$$

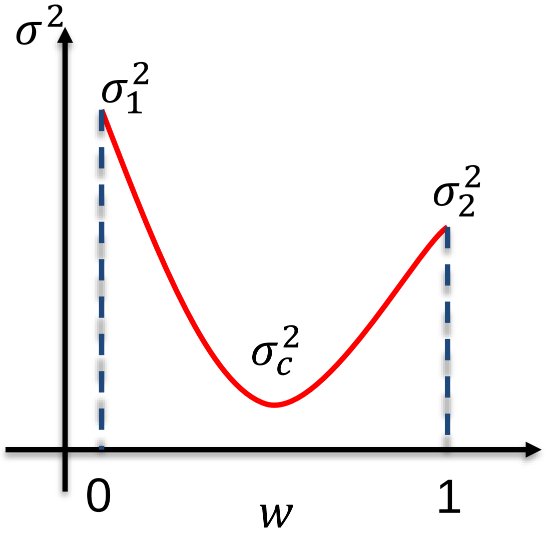
  - Bayesian Averaging (1997)

$$w_k = \frac{\exp\left(-\frac{1}{2}\Delta BIC_k\right)}{\exp\left(-\frac{1}{2}\Delta BIC_1\right) + \cdots + \exp\left(-\frac{1}{2}\Delta BIC_K\right)}, \Delta BIC_1 = BIC_1 - \min_{\mathrm{k}}(BIC_k)$$

Level-1

Aggregation Layer

Theoretical Justification



- Start from setting

$$\hat{e}_p^{(c)} = y_p - \left(w_1 \hat{y}_p^{(1)} + \cdots + w_K \hat{y}_p^{(K)}\right) = y_p - \hat{y}_p^{(c)}$$

- and we can rewrite the error term, as

$$\hat{e}_p^{(c)} = \left(y_p - w_1 \hat{y}_p^{(1)}\right) + \cdots = \left(w_1 \hat{e}_p^{(1)}\right) + \cdots + \left(w_K \hat{e}_p^{(K)}\right)$$

- If we just **consider the case of $\boldsymbol{K} = \mathbf{2}$ and $0 \le w \le 1$ then**

$$\hat{e}_p^{(c)} = w\hat{e}_p^{(1)} + (1-w)\hat{e}_p^{(2)}$$

- making it is possible to show that (under some assumptions)

$$\boldsymbol{Var}\left[\hat{e}^{(c)}\right] = \sigma_c^2(w) = w^2\sigma_1^2 + (1-w)^{\mathbf{2}}\sigma_2^2 + 2w(1-w)\rho\sigma_1\sigma_2$$

- which attains its minimum when

$$w^* = \frac{\sigma_2^2 - \rho\sigma_1\sigma_2}{\sigma_1^2 + \sigma_1^2 - 2\rho\sigma_1\sigma_2}$$

- It is easy to show that $\sigma_c^2(w^*) \le \min(\sigma_1^2, \sigma_2^2)$

Main Reading

- **Main References**
  - Friedman, J., Hastie, T., & Tibshirani, R. (2009). *The elements of statistical learning.* vol. 2. Springer. In particular chapters:
    - Ch 9 Additive Models, Trees, and Related Methods; Ch 9.2 Tree-Based Methods
    - Ch 8.7 Bagging; Ch 8.8 Model Averaging and Stacking
    - Ch 10 Boosting and Additive Trees

Further Reading

- **Empirical Evidence of Ensemble/Model Averaging Performance**
  - Fernández-Delgado, M., Cernadas, E., Barro, S., & Amorim, D. (2014). Do we need hundreds of classifiers to solve real world classification problems?. The Journal of Machine Learning Research, 15(1), 3133-3181.
  - Makridakis, S., & Hibon, M. (2000). The M3-Competition: results, conclusions and implications. International journal of forecasting, 16(4), 451-476.
  - Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2018). The M4 Competition: Results, findings, conclusion and way forward. International Journal of Forecasting.
- **Decision Trees**
  - L. Breiman, J. Friedman, R. Olshen, and C. Stone. Classification and Regression Trees. Wadsworth, Belmont, CA, 1984.
  - J.R. Quinlan. C4. 5: programs for machine learning. Morgan Kaufmann, 1993.
  - Ian H. Witten; Eibe Frank; Mark A. Hall (2011). Data Mining: Practical machine learning tools and techniques, 3rd Edition. Morgan Kaufmann, San Francisco. p. 191.

Further Reading

- **Random Forest**
  - Breiman, L. (1996). Bagging predictors. Machine learning, 24(2), 123-140.
  - Breiman, L. (2001). Random forests. Machine learning, 45(1), 5-32.
- **Gradient Boosting**
  - Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. Annals of statistics, 1189-1232.
  - Schapire, R. E., & Freund, Y. (2012). Boosting: Foundations and algorithms. MIT press.
- **Stacking and Model Combination**
  - Bates, J. M., & Granger, C. W. (1969). The combination of forecasts. Journal of the Operational Research Society, 20(4), 451-468.
  - Wolpert, D. H. (1992). Stacked generalization. Neural networks, 5(2), 241-259.
  - Breiman, L. (1996). Stacked regressions. Machine learning, 24(1), 49-64.
  - Timmermann, A. (2006). Forecast combinations. *Handbook of economic forecasting*, *1*, 135-196.
  - Hsiao, C., & Wan, S. K. (2014). Is there an optimal forecast combination?. Journal of Econometrics, 178, 294-309.

email: faculty@mlinstitute.com