Machine Learning with Tabular Clinical Data

Christopher Meaney

PhD Candidate
Dalla Lana School of Public Health
Division of Biostatistics
University of Toronto
&
Biostatistician
Department of Family and Community Medicine
University of Toronto

January 18, 2021

Machine Learning and Clinical Predictive Modelling

Machine Learning and Clinical Predictive Models

- Focus supervised machine learning (i.e. regression/classification problems).
- Primary goal is predictive inference. Discrimination/Calibration. Model generalizability.
- Explanability, evaluation of feature/outcome relationships important secondary objectives.
- Relationships between complexity, bias/variance, regularization, and generalization error.
- Generalization error. Internal validation (resampling, held-out test set). External validation.

Families of Supervised Machine Learning Models

- Linear/logistic regression. Regularization. Transformations and basis expansions.
- Trees. Tree Ensembles. Random Forests. Gradient Boosted Trees.
- Neural Networks. Deep Learning.
- Max-margin classifiers. Hinge Loss. Kernel Methods. Support Vector Machines.

Challenges Applying Supervised ML Models to Tabular Clinical Datasets

- Big N. Maybe large P (or P $>$ N).
- Features/variables (p=1...P) of mixed type (boolean, integer, categorical, numeric, etc.).
- Missing data common.
- Complex hierarchical, temporal, spatial dependencies in datasets.
- Selection effects. Informative observation marks/times.
- Measurement error (both in outcomes and/or features).

Hastie, Tibshirani, Friedman. (2009). Elements of Statistical Learning.

Model Complexity

- Model family/class endowed with innate complexity.
- Model families allow complexity to be increased/decreased, via tuneable hyper-parameters.
- Model performance dependent on suitable hyper-parameter tuning (complexity).

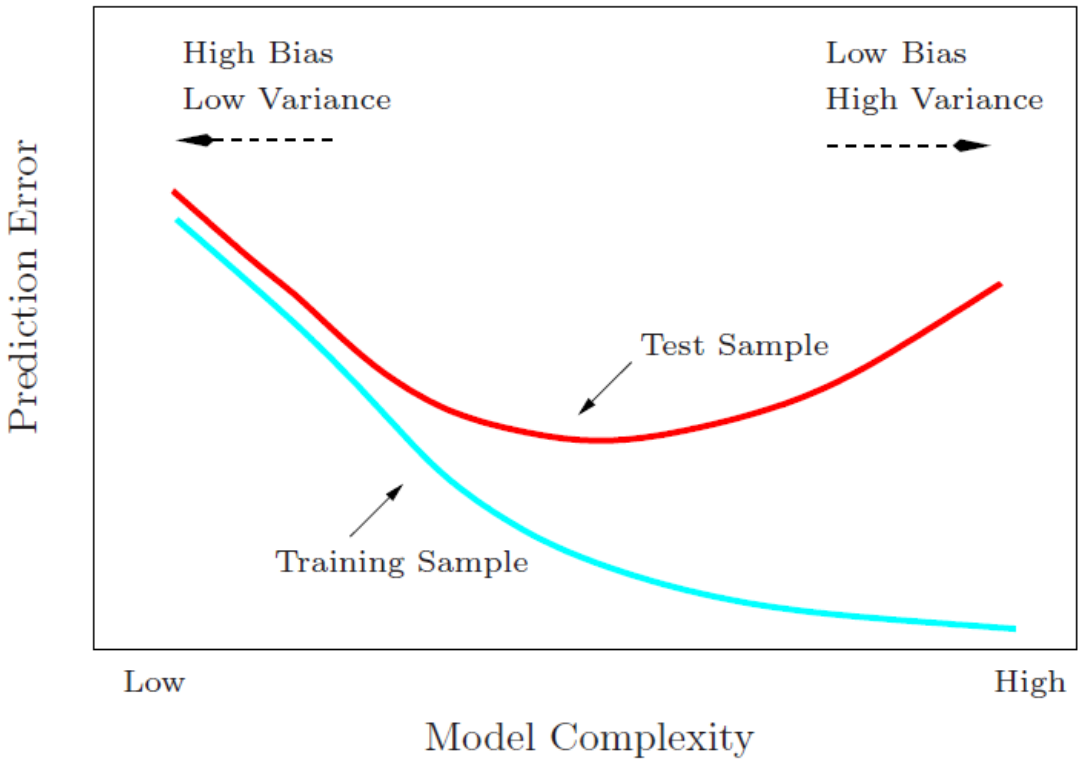


FIGURE 2.11. *Test and training error as a function of model complexity.*

Hastie, Tibshirani, Friedman. (2009). Elements of Statistical Learning. Chapter 2.
Steyerberg. (2019). Sys. Rev. No Performance Benefit ML Methods vs. Logistic Reg. JCE.

Model Evaluation

- Evaluation metric. Discrimination metric. Calibration index.
- Evaluation metric may be same as model implied loss/objective function (not always).
- Data adaptive models, hyper-parameter tuning, risk over-fitting.
- Model selection, hyper-parameter tuning on CV dataset.
- Generalization error estimated using held out test dataset.

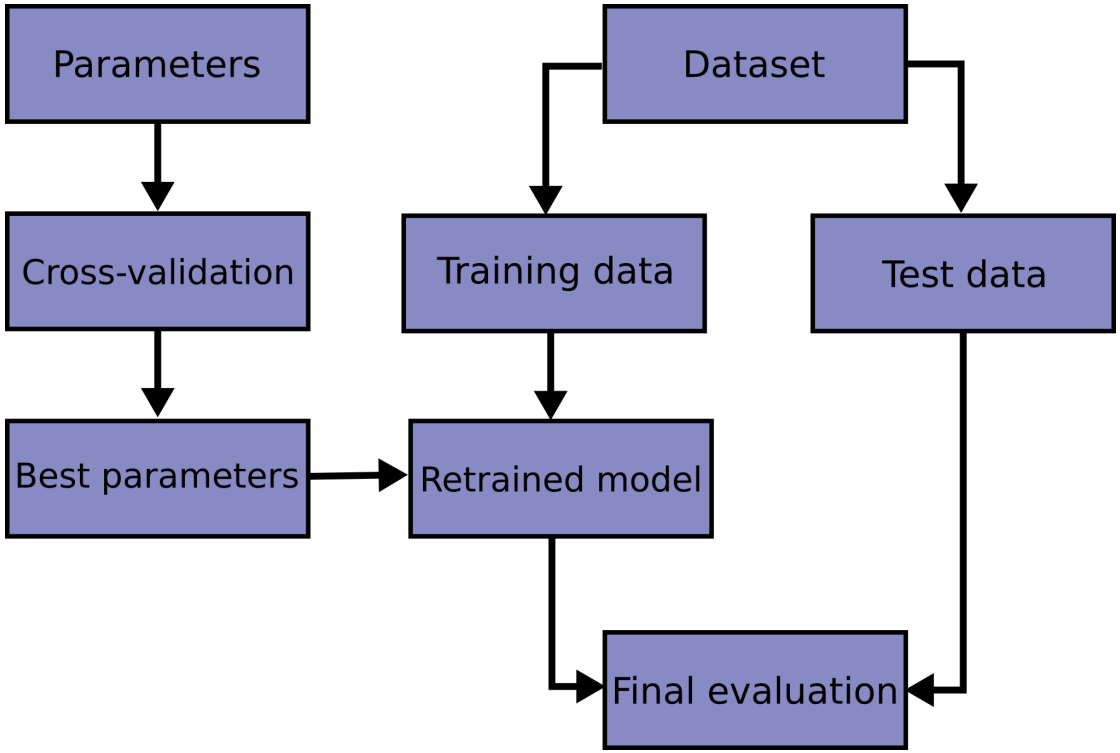


Figure Courtesy Python SKLearn Documentation: Cross Validation.

Steyerberg. (2009). Clinical Prediction Models. Chapter 15 - Model Evaluation. Springer

Linear Models for Regression and Classification

Linear Regression

- Setup: $\mathcal{D} = \{y_i, x_i\}_{i=1}^{N}$. Assume $y_i \in \mathcal{R}$ and $x_i = (1, x_{i1}, x_{i2}, \cdots, x_{iP}) \in \mathcal{R}^{P+1}$.
- Estimate regression function - $\mathbb{E}(Y|X)$ - as linear function of parameters $(\beta_0, \beta_1, \cdots, \beta_P)$.

$$\mathbb{E}(Y|X) = \beta_0 + \sum_{j=1}^{P} \beta_j * x_j$$

Linear Regression: Modelling Assumptions

- Linearity and Additivity.
- Homoskedasticity.
- Independence.
- Weak Exogeneity.
- Full Rank Design Matrix.

Linear Regression: Least Squares Parameter Estimates

- Residual Sum of Squares Loss Function: $RSS(\beta) = (y - X\beta)^T(y - X\beta) = \|y - X\beta\|_2^2$
- First Derivative Loss Function: $\frac{\partial RSS}{\partial \beta} = -2X^T(y - X\beta)$
- Second Derivate Loss Function: $\frac{\partial^2 RSS}{\partial \beta \partial \beta^T} = -2X^T X$
- Optimal Parameter Values: $\hat{\beta} = (X^T X)^{-1} X^T y$
- Estimator $\hat{\beta}$ provides best linear unbiased estimates (Gauss Markov Theorem).

Geometry of Linear Regression: Estimation and Prediction

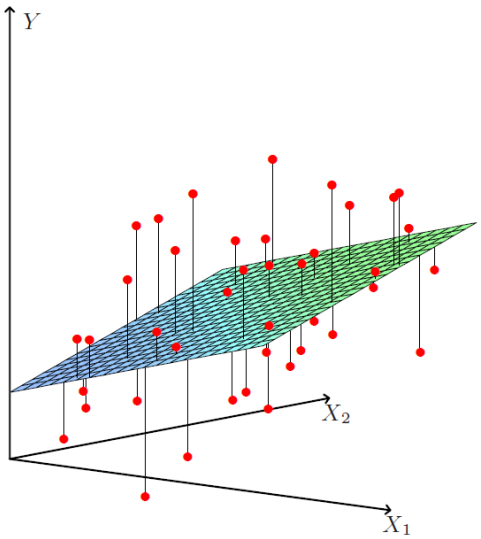


FIGURE 3.1. *Linear least squares fitting with $X \in \mathbb{R}^2$. We seek the linear function of X that minimizes the sum of squared residuals from Y.*

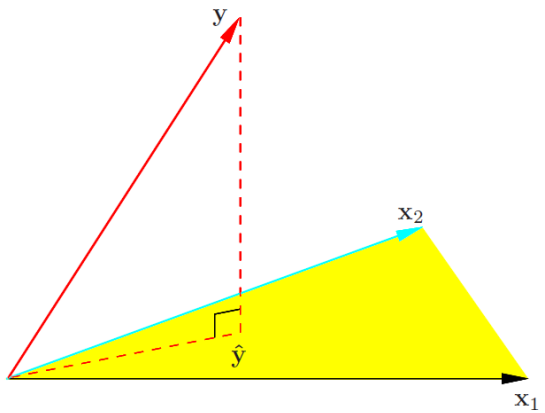


FIGURE 3.2. *The N-dimensional geometry of least squares regression with two predictors. The outcome vector $\mathbf{y}$ is orthogonally projected onto the hyperplane spanned by the input vectors $\mathbf{x}_1$ and $\mathbf{x}_2$. The projection $\hat{\mathbf{y}}$ represents the vector of the least squares predictions*

Hastie, Tibshirani, Friedman. (2009). Elements of Statistical Learning. Chapter 3.

Logistic Regression

- Setup: $\mathcal{D} = \{y_i, x_i\}_{i=1}^N$. Assume $y_i \in \{0, 1\}$ and $x_i = (1, x_{i1}, x_{i2}, \cdots, x_{iP}) \in \mathcal{R}^{P+1}$.
- Probability model for event: $\mathbb{P}(Y_i = 1|X_i) = \mathbb{E}(Y_i|X_i) = \pi_i$.
- Linear predictor: $g(\pi_i) = \eta_i = \beta_0 + \sum_{j=1}^P \beta_j * x_j$
- (Logit) Link Function: $log(\frac{\pi_i}{1-\pi_i}) = \beta_0 + \sum_{j=1}^P \beta_j * x_j$.
- (Expit) Inverse Link Function: $\pi_i = \frac{exp(\beta_0 + \sum_{j=1}^P \beta_j * x_j)}{1 + exp(\beta_0 + \sum_{j=1}^P \beta_j * x_j)}$

Logistic Regression: Estimation via SGD vs. Newton Raphson

- Binomial deviance: $\mathcal{L}(y; \beta) = \sum_{i=1}^N (y_i \log p(x_i; \beta) + (1 - y_i) \log(1 - p(x_i; \beta)))$
- Binomial deviance: $\mathcal{L}(y; \beta) = \sum_{i=1}^N (y_i \beta^T x_i - \log(1 + exp(\beta^T x_i)))$
- First derivative loss function: $\frac{\partial \mathcal{L}}{\partial \beta} = \sum_{i=1}^N x_i (y_i - p(x_i; \beta)) = X^T (y - p)$
- Second derivative loss function: $\frac{\partial^2 \mathcal{L}}{\partial \beta \partial \beta^T} = \sum_{i=1}^N x_i x_i^T p(x_i; \beta)(1 - p(x_i; \beta)) = -X^T W X$
- SGD Update: $\beta^{new} = \beta^{old} - \alpha * \frac{\partial \mathcal{L}}{\partial \beta} = \beta^{old} - \alpha * X^T (y - p)$
- NR Update: $\beta^{new} = \beta^{old} - (\frac{\partial^2 \mathcal{L}}{\partial \beta \partial \beta^T})^{-1} * \frac{\partial \mathcal{L}}{\partial \beta} = (X^T W X)^{-1} X^T W (X \beta^{old} + W^{-1}(y - p))$

Geometry of Logistic Regression: Classification Boundary

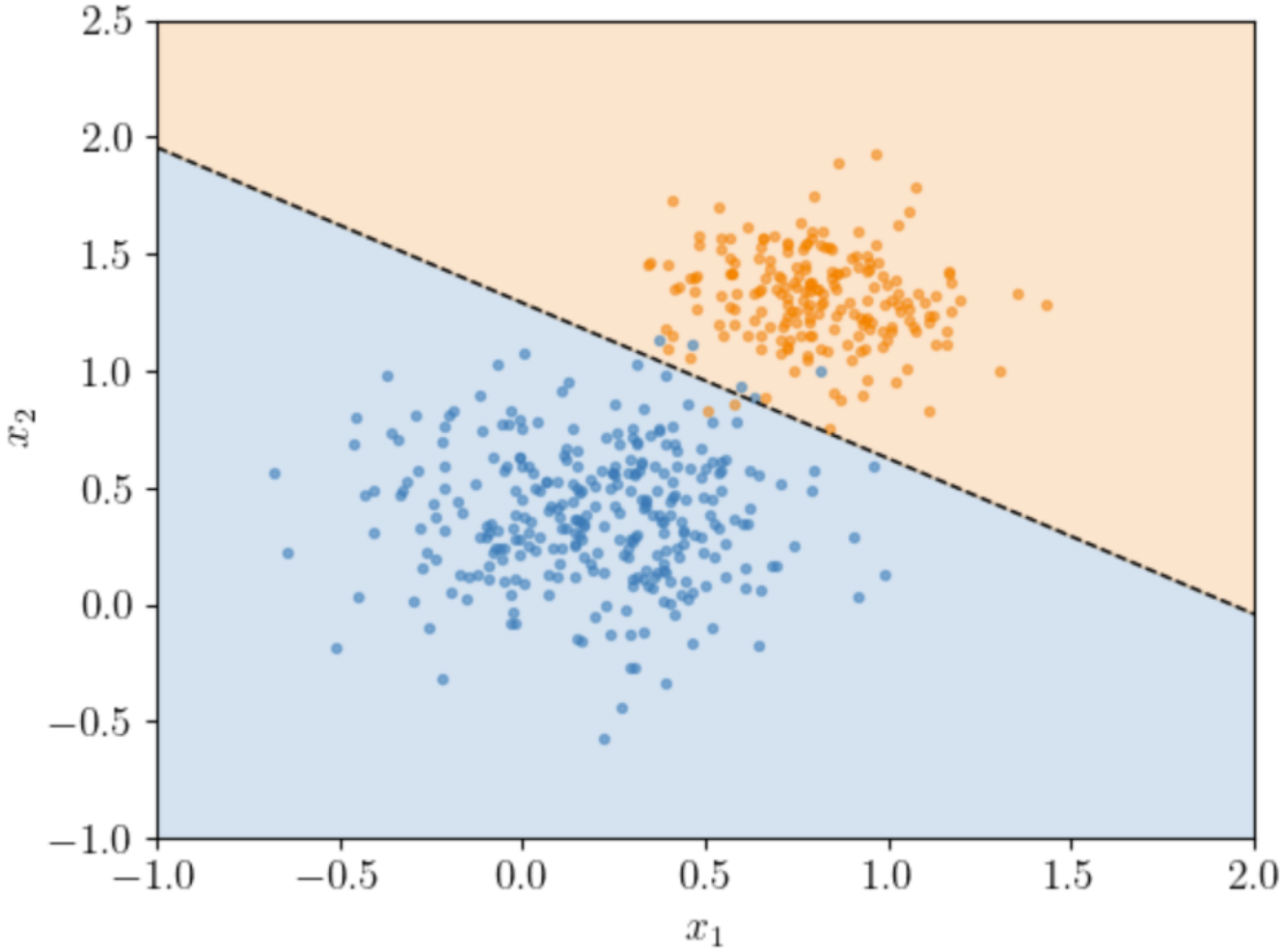


Figure Courtesy SciPython Documentation: Logistic Regression.

Increasing Complexity with Feature Transformations

- Identity transformation (corresponds to baseline complexity for design matrix).
- Polynomial basis $(x, x^2, x^3, x^4, \ldots)$. Interaction basis $(x_1 * x_2, x_1 * x_3, x_2 * x_3, \ldots.)$.
- Nonlinear feature transformations (e.g $log(x)$, $\sqrt{x}$, $sin(x)$, $cos(x)$, etc.).
- Piece-wise linear transformations $\left((x \in \mathcal{R}_j). \text{ and } (\bigcup_{j=1}^{J} \mathcal{R}_j = \mathcal{S}_x) \text{ and } (\mathcal{R}_j \text{ disjoint}) \right)$.

Increasing Complexity with Spline Bases

- Spline: a piece-wise polynomial of degree M with M-1 continuous derivatives.
- Knots placement: K knots, divide covariate space into K+1 regions.
- Fit M degree polynomial (M+1 parameters) in each region.
- Enforce constraints (continuity, M-1 derivative constraints).
- Degrees of Freedom: (K+1 regions)*(M+1 parameters) - (K knots)*(M constraints)

Increasing Complexity with Spline Bases

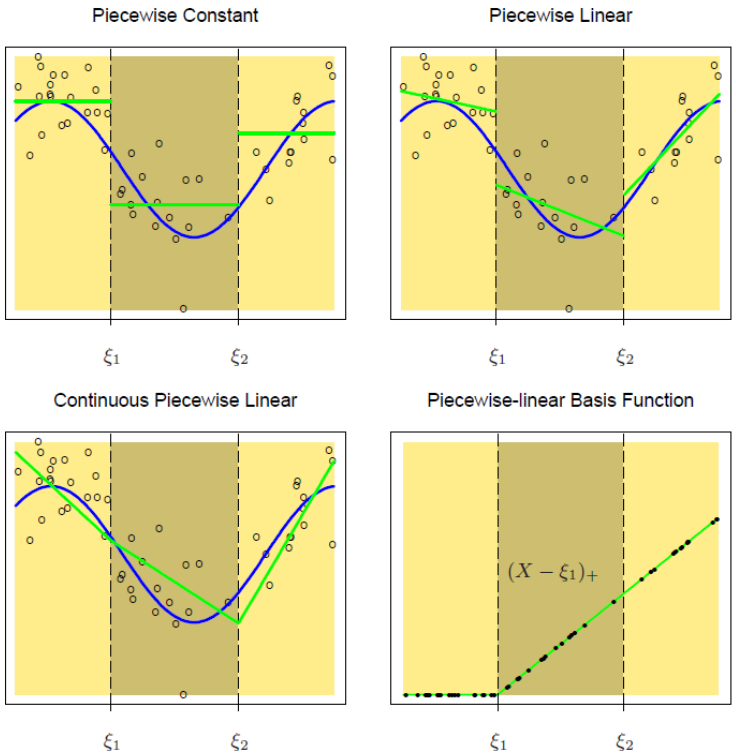


FIGURE 5.1. *The top left panel shows a piecewise constant function fit to some artificial data. The broken vertical lines indicate the positions of the two* knots *ξ_1 and ξ_2. The blue curve represents the true function, from which the data were generated with Gaussian noise. The remaining two panels show piecewise linear functions fit to the same data—the top right unrestricted, and the lower left restricted to be continuous at the knots. The lower right panel shows a piecewise–linear basis function, $h_3(X) = (X - \xi_1)_+$, continuous at ξ_1. The black points indicate the sample evaluations $h_3(x_i)$, $i = 1, \ldots, N$.*

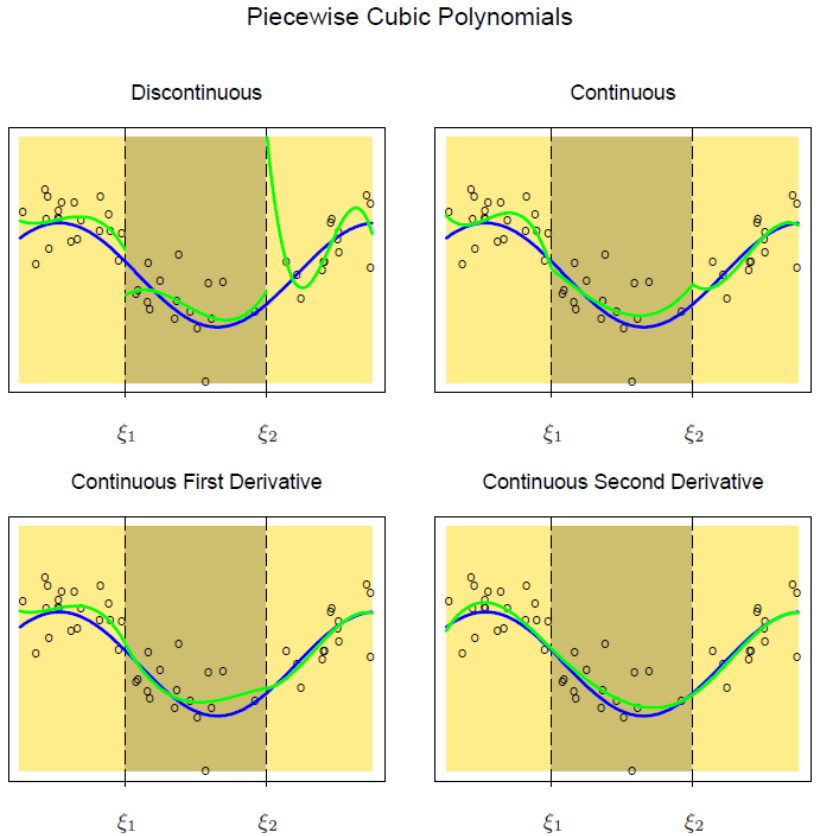


FIGURE 5.2. *A series of piecewise-cubic polynomials, with increasing orders of continuity.*

Hastie, Tibshirani, Friedman. (2009). Elements of Statistical Learning. Chapter 5.

Controlling Complexity with Regularized Statistical Learning

- Often, linear/logistic regression too complex, and can overfit training data.
- Can govern complexity with norm penalties added to model implied loss function.
- Norm penalties add bias and reduce variance, ideally improving overall model accuracy.
- Norm penalties yield parameter estimates with desired properties (sparsity, shrinkage, etc.).

Regularization Penalty	Regularized Linear Regression Objective Function
Ridge Penalty (ℓ_2 Norm)	$\min_{\beta \in \mathbb{R}^p} \left\{ \|y - X\beta\|_2^2 + \lambda \|\beta\|_2^2 \right\}$
Lasso Penalty (ℓ_1 Norm)	$\min_{\beta \in \mathbb{R}^p} \left\{ \|y - X\beta\|_2^2 + \lambda \|\beta\|_1 \right\}$
Elastic Net Penalty	$\min_{\beta \in \mathbb{R}^p} \left\{ \|y - X\beta\|_2^2 + \lambda_1 \|\beta\|_1 + \lambda_2 \|\beta\|_2^2 \right\}$
Group Lasso Penalty	$\min_{\beta \in \mathbb{R}^p} \left\{ \left\| y - \sum_{j=1}^J X_j \beta_j \right\|_2^2 + \lambda \sum_{j=1}^J \|\beta_j\|_{K_j} \right\}$, $\|z\|_{K_j} = (z^t K_j z)^{1/2}$

Constraint Regions of Regularizing Norms

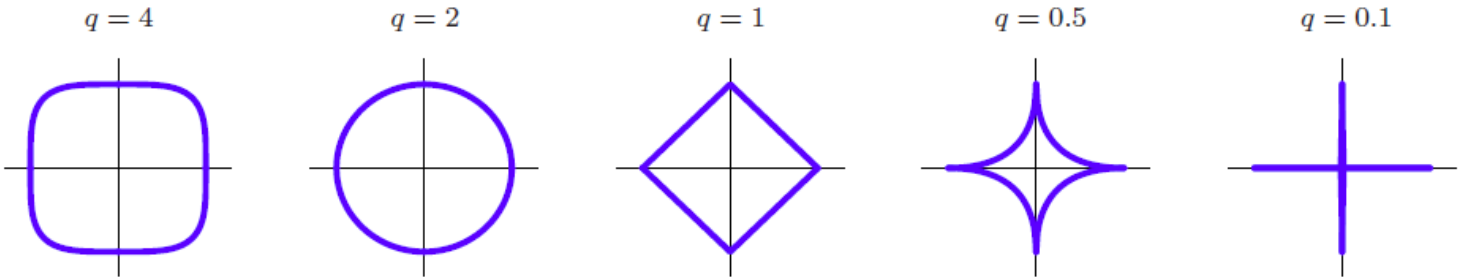


Figure 2.6 *Constraint regions* $\sum_{j=1}^p |\beta_j|^q \leq 1$ *for different values of* q. *For* $q < 1$, *the constraint region is nonconvex.*

Hastie, Tinshirani, Wainwright. (2015). Statistical Learning with Sparsity. Chapter 2.

Geometry of Regularizers: Lasso, Group Lasso, and Elastic Net Norms

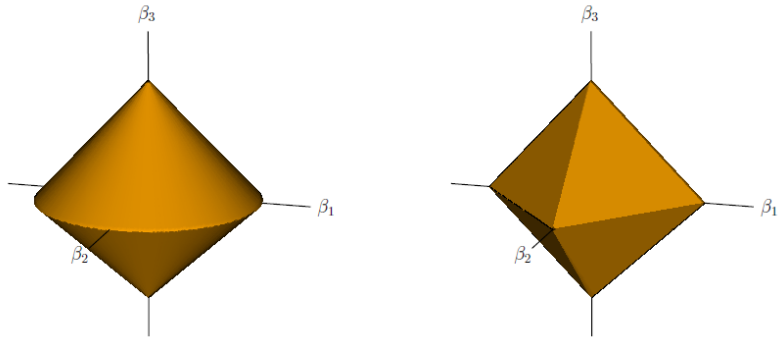


Figure 4.3 *The group lasso ball (left panel) in $\mathbb{R}^3$, compared to the ℓ_1 ball (right panel). In this case, there are two groups with coefficients $\theta_1 = (\beta_1, \beta_2) \in \mathbb{R}^2$ and $\theta_2 = \beta_3 \in \mathbb{R}^1$.*

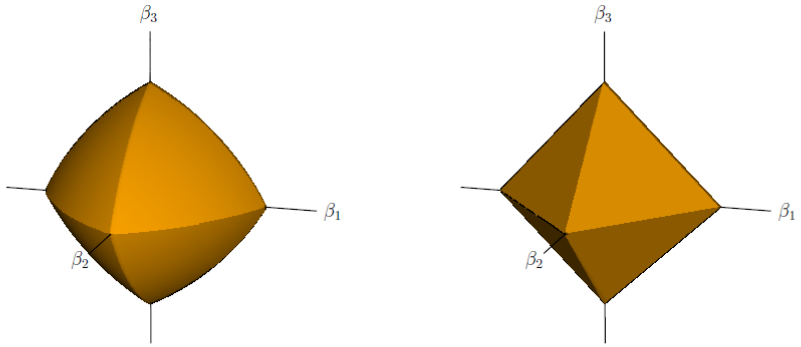


Figure 4.2 *The elastic-net ball with $\alpha = 0.7$ (left panel) in $\mathbb{R}^3$, compared to the ℓ_1 ball (right panel). The curved contours encourage strongly correlated variables to share coefficients (see Exercise 4.2 for details).*

Hastie, Tinshirani, Wainwright. (2015). Statistical Learning with Sparsity. Chapter 4.

Tree Ensembles for Regression and Classification

Classification and Regression Trees

- Two steps: (1) learn tree structure, (2) learn parameters of terminal prediction nodes.
- Tree structure: partition feature space into disjoint axis-aligned regions $(R_1, R_2, \ldots R_M)$.
- Partition corresponds to a tree, with internal (split) nodes, and terminal (predication) nodes.
- Learning optimal model is NP hard; use greedy forward stage-wise heuristic algorithm.
- Node impurity measures guide greedy search. Grow tree if information gain on split is large.
- Prediction node outputs $(c_1, c_1, \ldots c_M)$ are simple statistics (mean, proportion, etc.).

$$f(x) = \sum_{m=1}^{M} c_m \mathbb{I}(x_i \in R_m)$$

Hastie, Tibshirani, Friedman. (2009). Elements of Statistical Learning. Chapter 9.
Strobl, Malley, Tutz. (2009). Intro to Recursive Partitioning.

Partition of Feature Space using Tree Methods

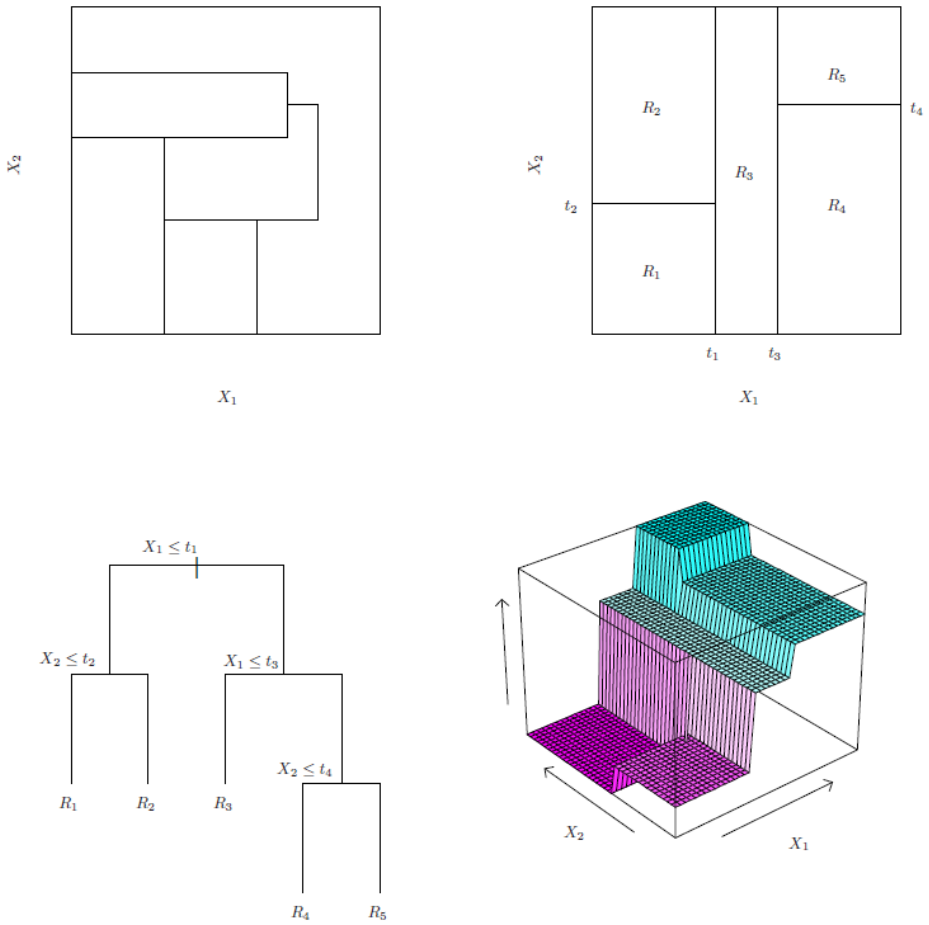


FIGURE 8.3. Top Left: *A partition of two-dimensional feature space that could not result from recursive binary splitting.* Top Right: *The output of recursive binary splitting on a two-dimensional example.* Bottom Left: *A tree corresponding to the partition in the top right panel.* Bottom Right: *A perspective plot of the prediction surface corresponding to that tree.*

James, Tibs, Witten, Hastie. (2015). Intro to Elements of Statistical Learning. Chapter 8.

Metrics Guiding Structural Search of Trees: Node Impurity Measures

- Regression (Squared Error): $(y_i - f(x))^2$
- Classification (Misclassification error): $1 - \max(\hat{p}, 1 - \hat{p})$
- Classification (Gini Index): $2 * \hat{p}(1 - \hat{p})$
- Classification (Cross-Entropy or Binomial Deviance): $-\hat{p}\log(\hat{p}) - (1 - \hat{p})\log(1 - \hat{p})$.

Determination of Tree Structure by Information Gain

$$IG = \text{Entropy}(A) - \left(\frac{N_L}{N} \text{Entropy}(A_L) + \frac{N_R}{N} \text{Entropy}(A_R) \right)$$

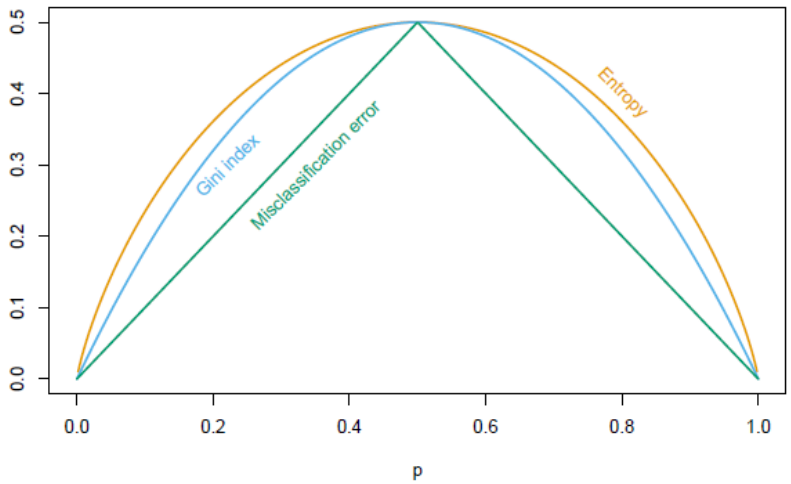


FIGURE 9.3. *Node impurity measures for two-class classification, as a function of the proportion p in class 2. Cross-entropy has been scaled to pass through* $(0.5, 0.5)$.

Hastie, Tibshirani, Friedman. (2009). Elements of Statistical Learning. Chapter 9.

Recursive Binary Partitioning and Cost Complexity Pruning

- $\alpha = 0$. Deep trees, no pruning.
- $\alpha = \infty$ Near all internal nodes penalized, resulting in tree stump model.
- Cross-validation for determination optimal tree complexity measure (α).

$$C_\alpha(T) = \sum_{m=1}^{T} N_m Q_m + \alpha |T|$$

Algorithm 8.1 *Building a Regression Tree*

1. Use recursive binary splitting to grow a large tree on the training data, stopping only when each terminal node has fewer than some minimum number of observations.

2. Apply cost complexity pruning to the large tree in order to obtain a sequence of best subtrees, as a function of α.

3. Use K-fold cross-validation to choose α. For each $k = 1, \ldots, K$:

   (a) Repeat Steps 1 and 2 on the $\frac{K-1}{K}$th fraction of the training data, excluding the kth fold.

   (b) Evaluate the mean squared prediction error on the data in the left-out kth fold, as a function of α.

   Average the results, and pick α to minimize the average error.

4. Return the subtree from Step 2 that corresponds to the chosen value of α.

James, Tibs, Witten, Hastie. (2015). Intro to Elements of Statistical Learning. Chapter 8.

Decision Boundaries for Tree vs. Linear Classifiers

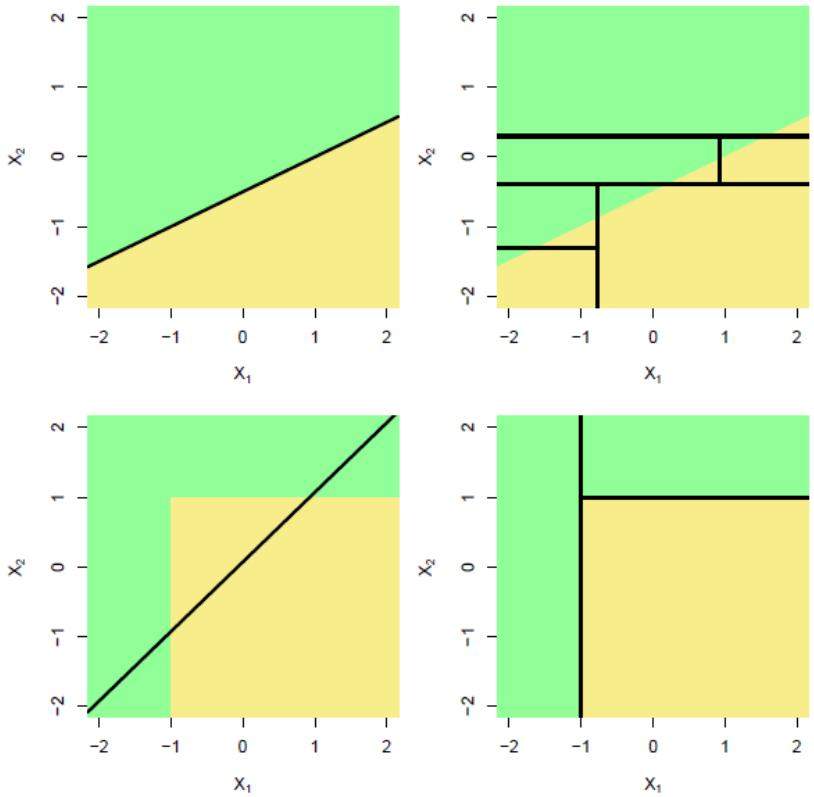


FIGURE 8.7. Top Row: *A two-dimensional classification example in which the true decision boundary is linear, and is indicated by the shaded regions. A classical approach that assumes a linear boundary (left) will outperform a decision tree that performs splits parallel to the axes (right).* Bottom Row: *Here the true decision boundary is non-linear. Here a linear model is unable to capture the true decision boundary (left), whereas a decision tree is successful (right).*

James, Tibs, Witten, Hastie. (2015). Intro to Elements of Statistical Learning. Chapter 8.

Tree Ensembles, Bagging and Random Forests

- Draw $b = 1...B$ random samples (with replacement) of size N from original dataset.
- Fit $b = 1...B$ regression/classification trees ($T(\Theta_b; x)$) to each bootstrap sample.
- Final prediction/classification is average or majority vote over bagged samples.
- Random sample columns reduces de-correlates ensemble predictions: $\rho\sigma^2 + \frac{1-\rho}{B}\sigma^2$.
- Bagging reduces variance/instability of individual trees. Ensembles highly predictive.

$$f_{RF}^{B}(x) = \frac{1}{B}\sum_{b=1}^{B} T(\Theta_b; x)$$

Random Forest Out of Bag (OOB) Error Estimates

- Probability in OOB sample: $(1 - \frac{1}{N})^N \approx exp(-1) = 0.368$.
- Compute output for $b = 1...B$ OOB samples. OOB prediction as average or majority vote.
- OOB metrics of generalization performance closely approximate CV estimates.

Hastie, Tibshirani, Friedman. (2009). Elements of Statistical Learning. Chapter 15.
Louppe. (2014). Understanding Random Forests. PhD Thesis.

Random Forest Algorithm

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 $\{T_b\}_1^B$.

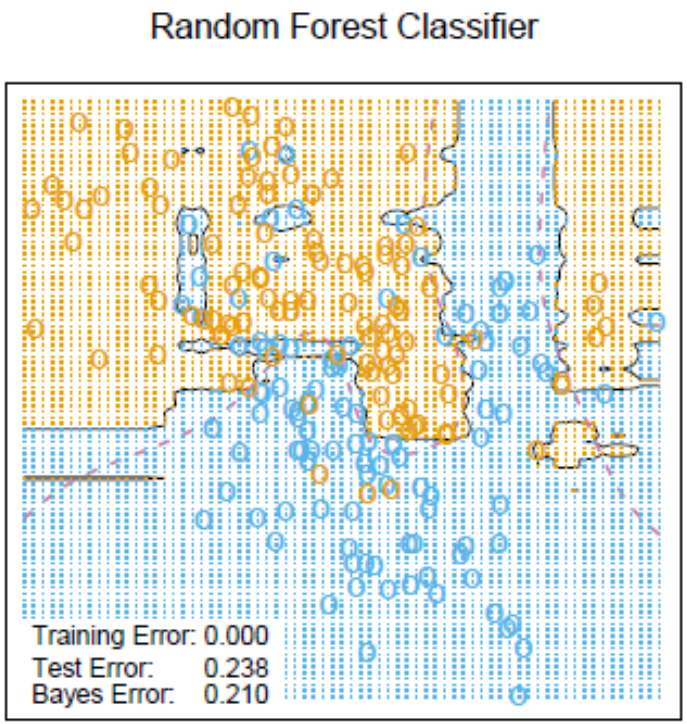
To make a prediction at a new point x:

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 $\hat{C}_{\text{rf}}^B(x) = \textit{majority vote}\ \{\hat{C}_b(x)\}_1^B$.

Hastie, Tibshirani, Friedman. (2009). Elements of Statistical Learning. Chapter 15.

Random Forest: Geometry of Decision Boundaries



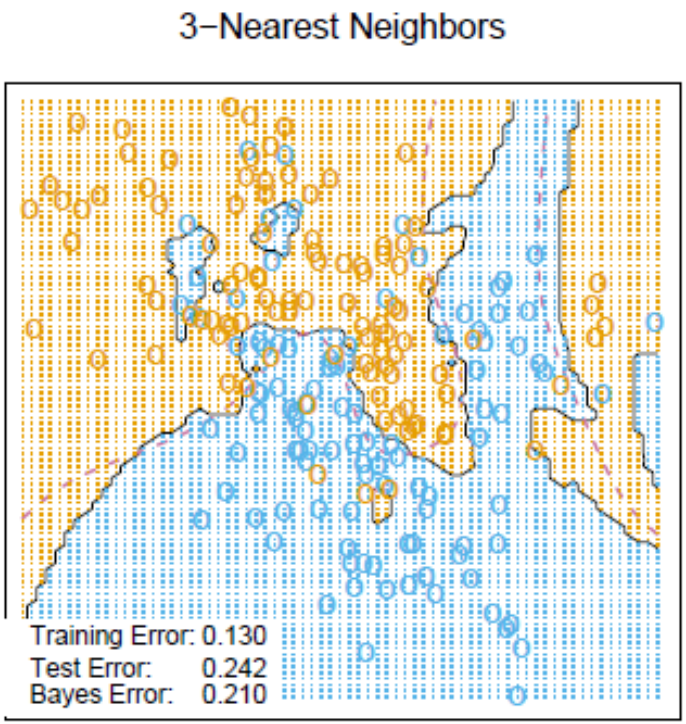


FIGURE 15.11. *Random forests versus 3-NN on the mixture data. The axis-oriented nature of the individual trees in a random forest lead to decision regions with an axis-oriented flavor.*

Hastie, Tibshirani, Friedman. (2009). Elements of Statistical Learning. Chapter 15.

Tree Ensembles, Forward Stagewise Fitting and Boosting

- Fit $m = 1 \ldots M$ trees to dataset in greedy forward-stagewise manner.
- Each tree in ensemble intended to improve upon previous trees/functions errors.
- Different boosting algorithms: AdaBoost vs. Gradient/Newton Boosting.

AdaBoost

- Initialize observations weights to $w_i = 1/N$.
- For $m = 1 \ldots M$
  - Fit classifier $G_m(x)$ to weighted training data.
  - Compute errors: $\text{err}_m = \frac{\sum_{i=1}^{N} w_i \mathbb{I}(y_i = G_m(x_i))}{\sum_{i=1}^{N}} w_i$
  - Compute tree weights: $\alpha_m = \log(\frac{1-\text{err}_m}{\text{err}_m})$
  - Re-estimate observation weights: $w_i^{new} = w_i^{old} \exp \alpha_m \mathbb{I}(y_i \neq G_m(x_i))$
- Output final model: $G(x) = \text{sign}(\sum_{m=1}^{M} \alpha_m G_m(x))$

Gradient/Newton Boosting

- Initial predictions are constant: mean/proportion.
- For $m = 1 \ldots M$
  - Compute pseudo-residuals.
  - Fit regression/classification tree to pseudo-residuals.
  - Determine output values for terminal nodes of tree.
  - Update model: $f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J_m} c_{jm} \mathbb{I}(x \in R_{jm})$.
- Output final model: $\hat{f}(x) = f_M(x)$

Hastie, Tibshirani, Friedman. (2009). Elements of Statistical Learning. Chapter 10.

GBM: Gradient Boosting Algorithm

Algorithm 4: Gradient tree boosting

Input : Data set $\mathcal{D}$.
A loss function L.
The number of iterations M.
The learning rate η.
The number of terminal nodes T

1 Initialize $\hat{f}^{(0)}(x) = \hat{f}_0(x) = \hat{\theta}_0 = \arg\min_{\theta} \sum_{i=1}^{n} L(y_i, \theta)$;

2 **for** $m = 1,2,..,M$ **do**

3 $\quad \hat{g}_m(x_i) = \left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f(x)=\hat{f}^{(m-1)}(x)}$;

4 $\quad$ Determine the structure $\{\hat{R}_{jm}\}_{j=1}^{T}$ by selecting splits which maximize $Gain = \frac{1}{2}\left[\frac{G_L^2}{n_L} + \frac{G_R^2}{n_R} - \frac{G_{jm}^2}{n_{jm}}\right]$;

5 $\quad$ Determine the leaf weights $\{\hat{w}_{jm}\}_{j=1}^{T}$ for the learnt structure by $\hat{w}_{jm} = \arg\min_{w_j} \sum_{i \in I_{jm}} L(y_i, \hat{f}^{(m-1)}(x_i) + w_j)$;

6 $\quad \hat{f}_m(x) = \eta \sum_{j=1}^{T} \hat{w}_{jm} \mathrm{I}(x_i \in \hat{R}_{jm})$;

7 $\quad \hat{f}^{(m)}(x) = \hat{f}^{(m-1)}(x) + \hat{f}_m(x)$;

8 **end**

Output: $\hat{f}(x) \equiv \hat{f}^{(M)}(x) = \sum_{m=0}^{M} \hat{f}_m(x)$

Nielsen. (2016). Why XGBoost Wins All ML Competitions. MSc Thesis.

XGBoost: Newton Boosting Algorithm

Algorithm 3: Newton tree boosting

Input : Data set $\mathcal{D}$.
A loss function L.
The number of iterations M.
The learning rate η.
The number of terminal nodes T

Initialize $\hat{f}^{(0)}(x) = \hat{f}_0(x) = \hat{\theta}_0 = \arg\min_{\theta} \sum_{i=1}^{n} L(y_i, \theta)$;

for $m = 1,2,..,M$ **do**

$\hat{g}_m(x_i) = \left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f(x)=\hat{f}^{(m-1)}(x)}$;

$\hat{h}_m(x_i) = \left[\frac{\partial^2 L(y_i, f(x_i))}{\partial f(x_i)^2}\right]_{f(x)=\hat{f}^{(m-1)}(x)}$;

Determine the structure $\{\hat{R}_{jm}\}_{j=1}^{T}$ by selecting splits which maximize $Gain = \frac{1}{2}\left[\frac{G_L^2}{H_L} + \frac{G_R^2}{H_R} - \frac{G_{jm}^2}{H_{jm}}\right]$;

Determine the leaf weights $\{\hat{w}_{jm}\}_{j=1}^{T}$ for the learnt structure by $\hat{w}_{jm} = -\frac{G_{jm}}{H_{jm}}$;

$\hat{f}_m(x) = \eta \sum_{j=1}^{T} \hat{w}_{jm} \mathrm{I}(x \in \hat{R}_{jm})$;

$\hat{f}^{(m)}(x) = \hat{f}^{(m-1)}(x) + \hat{f}_m(x)$;

end

Output: $\hat{f}(x) \equiv \hat{f}^{(M)}(x) = \sum_{m=0}^{M} \hat{f}_m(x)$

Nielsen. (2016). Why XGBoost Wins All ML Competitions. MSc Thesis.

Regularization XGBoost

- Boosting Parameters
  - Number of Trees
  - Learning Rate
- Tree Parameters
  - Number of terminal nodes (T_{max}).
  - Minimum coverage (H_{min}) per terminal node.
  - Terminal Node Penalization: $IG = \left(\frac{1}{2} (\frac{G_L}{H_L} + \frac{G_R}{H_R} - \frac{G}{H}) \right) - \gamma$
  - L2 Penalization Outputs:
    - $IG = \frac{1}{2} \left(\frac{G_L}{H_L + \lambda} + \frac{G_R}{H_R + \lambda} - \frac{G}{H + \lambda} \right)$
    - $w_{jm} = -\frac{G_{jm}}{H_{jm} + \lambda}$
  - L1 Penalization Outputs:
    - $IG = \frac{1}{2} \left(\frac{P_\alpha(G_L)}{H_L} + \frac{P_\alpha(G_R)}{H_R} - \frac{P_\alpha(G)}{H} \right)$
    - $w_{jm} = -\frac{P_\alpha(G_{jm})}{H_{jm} + \lambda}$
    - $P_\alpha(G) = sign(G) max(0, G - \alpha)$
- Randomization Parameters
  - Sample rows of data matrix.
  - Sample columns of data matrix.

Nielsen. (2016). Why XGBoost Wins All ML Competitions. MSc Thesis.

GBM: Loss Functions, Gradients and Terminal Node Estimators

4.1 Gaussian

Deviance	$\frac{1}{\sum w_i}\sum w_i(y_i - f(\mathbf{x}_i))^2$
Initial value	$f(\mathbf{x}) = \frac{\sum w_i(y_i - o_i)}{\sum w_i}$
Gradient	$z_i = y_i - f(\mathbf{x}_i)$
Terminal node estimates	$\frac{\sum w_i(y_i - f(\mathbf{x}_i))}{\sum w_i}$

4.2 AdaBoost

Deviance	$\frac{1}{\sum w_i}\sum w_i \exp(-(2y_i - 1)f(\mathbf{x}_i))$
Initial value	$\frac{1}{2}\log\frac{\sum y_i w_i e^{-o_i}}{\sum(1-y_i)w_i e^{o_i}}$
Gradient	$z_i = -(2y_i - 1)\exp(-(2y_i - 1)f(\mathbf{x}_i))$
Terminal node estimates	$\frac{\sum(2y_i - 1)w_i \exp(-(2y_i - 1)f(\mathbf{x}_i))}{\sum w_i \exp(-(2y_i - 1)f(\mathbf{x}_i))}$

4.3 Bernoulli

Deviance	$-2\frac{1}{\sum w_i}\sum w_i(y_i f(\mathbf{x}_i) - \log(1 + \exp(f(\mathbf{x}_i))))$
Initial value	$\log\frac{\sum w_i y_i}{\sum w_i(1-y_i)}$
Gradient	$z_i = y_i - \frac{1}{1+\exp(-f(\mathbf{x}_i))}$
Terminal node estimates	$\frac{\sum w_i(y_i - p_i)}{\sum w_i p_i(1-p_i)}$ where $p_i = \frac{1}{1+\exp(-f(\mathbf{x}_i))}$

Ridgeway. (2020). gbm: Gradient Boosting Machines in R.

Comparing ML Models for Tabular Clinical Datasets

Comparing ML Models for Tabular Clinical Datasets

Some characteristics of different learning methods.
Key: 🟢= good, 🟡=fair, and 🔴=poor.

Characteristic	Neural Nets	SVM	CART	GAM	KNN, kernels	MART
Natural handling of data of "mixed" type	🔴	🔴	🟢	🟢	🔴	🟢
Handling of missing values	🔴	🔴	🟢	🔴	🟢	🟢
Robustness to outliers in input space	🔴	🔴	🟢	🟡	🟢	🟢
Insensitive to monotone transformations of inputs	🔴	🔴	🟢	🟢	🔴	🟢
Computational scalability (large N)	🔴	🔴	🟢	🟢	🔴	🟢
Ability to deal with irrelevant inputs	🔴	🔴	🟢	🟡	🔴	🟢
Ability to extract linear combinations of features	🟢	🟢	🔴	🔴	🟡	🟡
Interpretability	🔴	🔴	🟡	🟢	🔴	🟢
Predictive power	🟢	🟢	🔴	🔴	🟢	🟢

Hastie. (2003). Boosting: Stanford Course Notes.

Conclusions