Precision Medicine: Lecture 03
Machine Learning

Michael R. Kosorok,
Nikki L. B. Freeman and Owen E. Leete

Department of Biostatistics
Gillings School of Global Public Health
University of North Carolina at Chapel Hill

Fall, 2021

Machine Learning

- Supervised Learning
  - Classification
  - Regression
- Unsupervised Learning
- Reinforcement Learning
  - Continuously update decision rules as new information becomes available
- In precision medicine we generally focus on supervised learning (this lecture), and reinforcement learning (lectures 7 & 8)
- Many methods were originally developed with an emphasis on prediction rather than inference, but we are working on inference (many open questions)

Outline

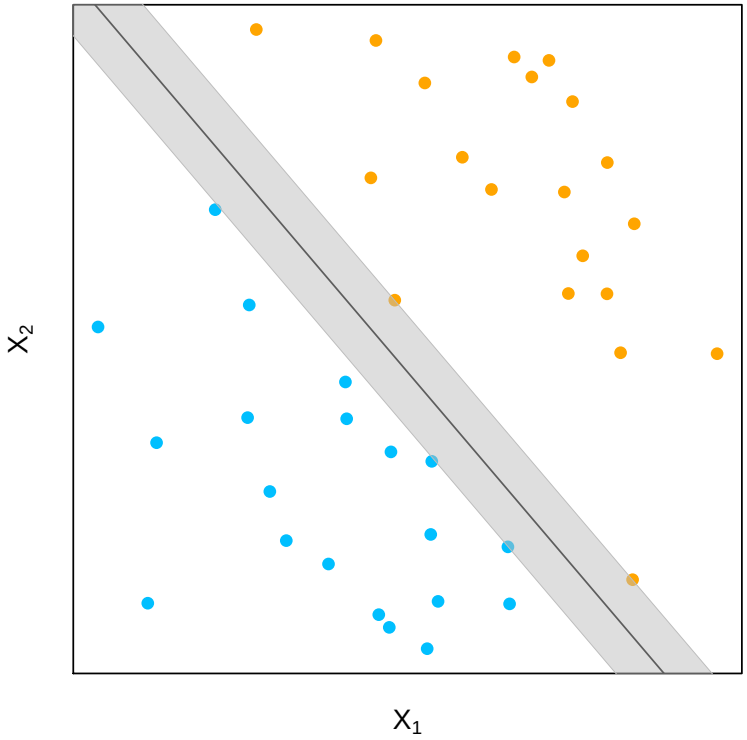
Support Vector Machines (SVM)

- Supervised learning methods originally developed for linear binary classification
- Several extensions have been developed for multi-class classification, non-linear classification, linear and non-linear regression, clustering, etc.
- All variants are based on the construction of one or more hyperplanes in high- or infinite-dimensional space
- Often used in the high-dimensional ($p \gg n$) setting

Separating Hyperplane Classifiers

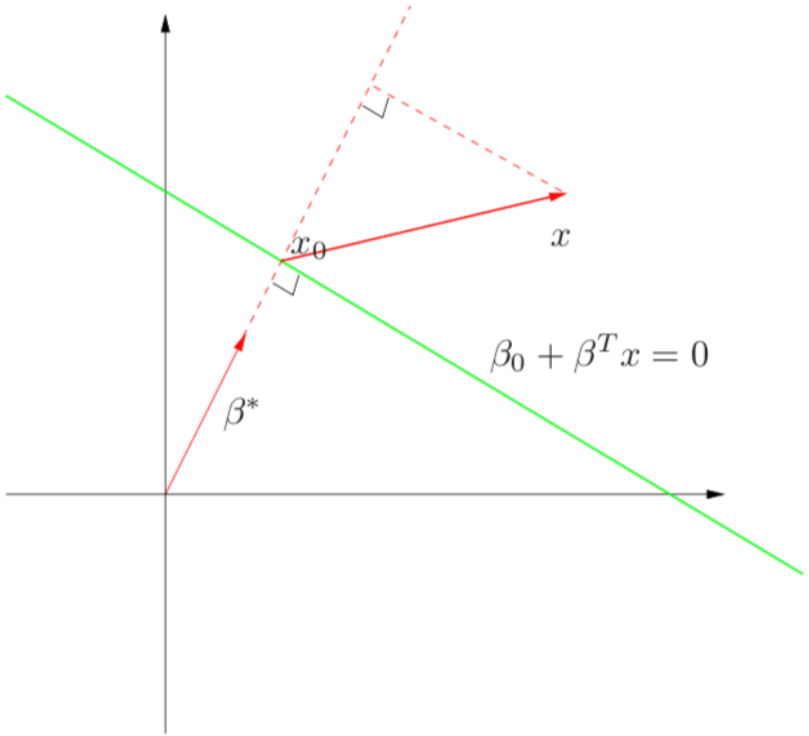
For binary classification the training data consists of n pairs of $(x_1, y_1), \ldots, (x_n, y_n)$ with $x_i \in \mathbb{R}^p$, $y_i \in \{-1, +1\}$

- Separating hyperplane classifiers construct linear decision boundaries that separate the data
- In the example there are infinitely many possible separating hyperplanes
- The SVM decision boundary is the the hyperplane that provides maximal separation



Linear Algebra of a Hyperplane

- The green line is the affine set L defined by the equation $f(x) = \beta_0 + \beta^T x = 0$
- $\beta^* = \beta / \|\beta\|$ is the vector normal to the surface of L
- For any point x_0 in L, $\beta^T x_0 = -\beta_0$
- The signed distance of any point x to L is given by $\frac{1}{\|\beta\|}(\beta^T x - \beta_0)$



Source: ESL Ch. 4

Optimal Separating Hyperplane

The optimal separating hyperplane separates the two classes and maximizes the distance to the closest point from either class

- Provides a unique solution for the separating hyperplane
- Reduces generalization error
- Found as the solution to the optimization problem

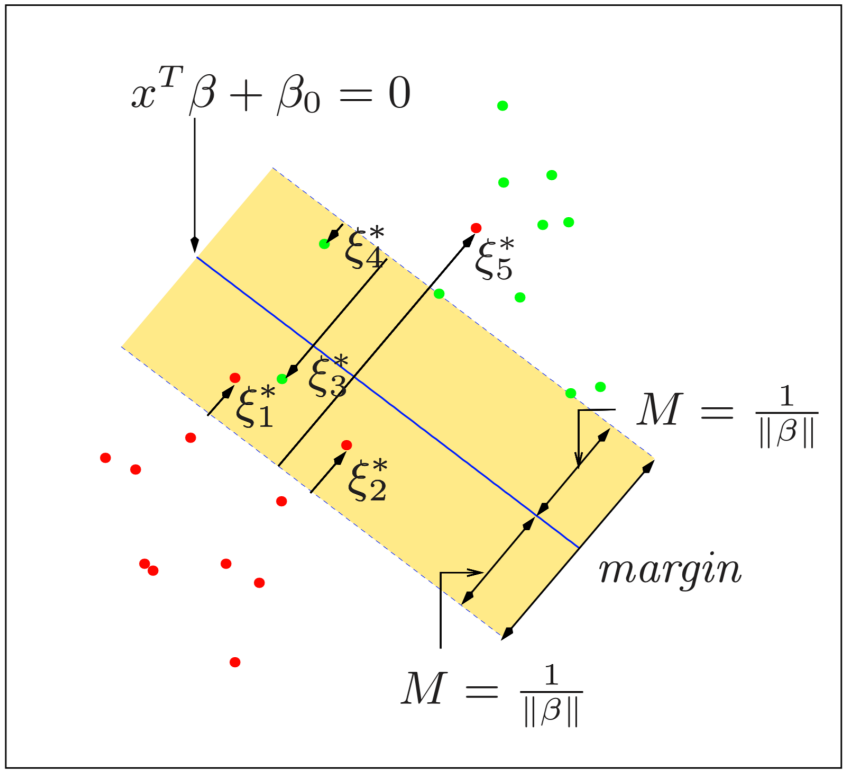
$$\max_{\beta,\beta_0,\|\beta\|=1} M$$
$$\text{subject to } y_i(x_i^T\beta + \beta_0) \geq M,\ i = 1, \ldots, n$$

Or equivalently

$$\min_{\beta,\beta_0,} \|\beta\|^2$$
$$\text{subject to } y_i(x_i^T\beta + \beta_0) \geq 1,\ i = 1, \ldots, n$$

Non-separable Case

- What if there is no hyperplane that separates the data?
- We can introduce slack variables, ξ_i, that measure how far observations are from the correct side of the margin
- The support vectors are observations that lie on the margin or have non-zero slack variables



Source: ESL Ch. 12

Non-separable Case, cont.

- For the non-separable case, the optimization problem becomes

$$\min_{\beta,\beta_0,} \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{n}\xi_i$$
$$\text{subject to } y_i(x_i^T\beta + \beta_0) \geq 1 - \xi_i,\ \xi_i \geq 0\ \forall i$$

  where C is a tuning parameter
- When $C = \infty$ this is equivalent to the separable case
  - The non-separable SVM can be applied to separable data with a finite value for C
  - Higher misclassification, but better generalization error
  - More on this later

Computing the support vector classifier

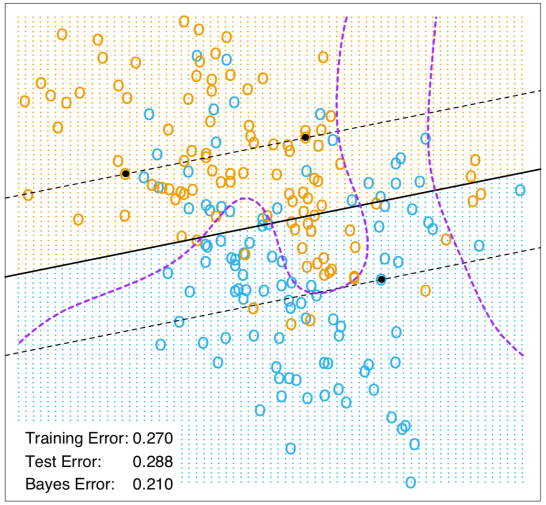
- We can formulate the SVM objective as a quadratic optimization problem with linear inequality constraints using the Lagrange dual representation:

$$\hat{f}_{\text{SVM}} = \max_{\alpha} \left[\sum_{i=1}^{n} \alpha_i - \sum_{i=1}^{n} \sum_{i'=1}^{n} \alpha_i \alpha_{i'} y_i y_{i'} x_i^T x_{i'} \right]$$

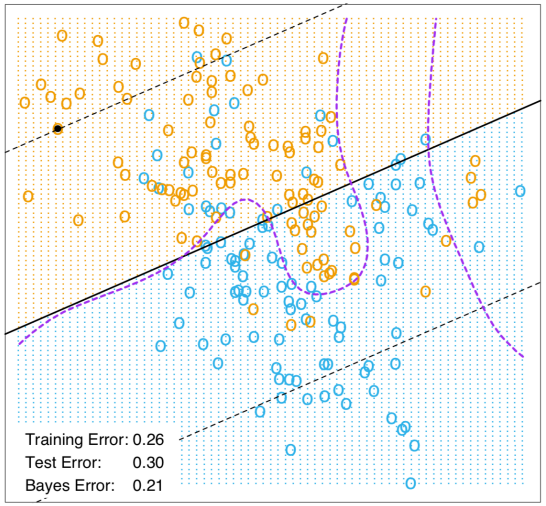
- Maximizing this subject to $0 \leq \alpha_i \leq C$, $\sum_{i=1}^{n} \alpha_i y_i = 0$ and the Karush-Kuhn-Tucker conditions uniquely determines the solution to the optimization problem
- This representation is the basis for several extensions of SVM

Mixture Example (effect of cost 'C')

- Smaller C results in a more complex decision boundary (i.e. $\|\beta\|^2$ can be large)
  - Better training error
  - Poor generalization error
- Larger C results in a simpler decision boundary
  - Worse training error
  - Better generalization error



$C = 10000$



$C = 0.01$

Source: ESL Ch. 12

Support Vector Machines and Kernels

- We can enlarge the feature space using basis expansions (e.g. higher order terms x_1^2, interaction terms $x_1 x_2$, etc.)
- Linear decision boundaries in higher dimensional space correspond to non-linear decision boundaries in the original space
- Consider fitting the SVM with basis functions $h(x_i) = (h_1(x_i), h_2(x_i), \ldots, h_{p'}(x_i))$ producing the non-linear function $\hat{f}(x_i) = h(x)^T \beta + \beta_0$ where the classifier is $\hat{G}(x) = \text{sign}(\hat{f}(x_i))$

Computing the SVM for Classification

For the enlarged feature space, the Lagrange dual becomes :

$$L_D = \sum_{i=1}^{n} \alpha_i - \sum_{i=1}^{n} \sum_{i'=1}^{n} \alpha_i \alpha_{i'} y_i y_{i'} \langle h(x_i), h(x_{i'}) \rangle$$

and the solution function $f(x)$ can be written

$$f(x) = h(x)^T \beta + \beta_0 = \sum_{i=1}^{n} \alpha_i y_i \langle h(x), h(x_i) \rangle + \beta_0$$

Each of these depend on the covariates x only through the inner products

- The transformation $h(x)$ need not be explicitly specified
- All we need to know is the kernel function $\mathcal{K}(x_i, x_{i'}) = \langle h(x_i), h(x_{i'}) \rangle$

Kernel examples

We generally assume that the enlarged feature space is a reproducing kernel Hilbert space (RKHS) with kernel function $\mathcal{K}(x_i, x_{i'})$ which computes the inner product in the feature space

The kernels $\mathcal{K}$ should be symmetric positive semi-definite functions

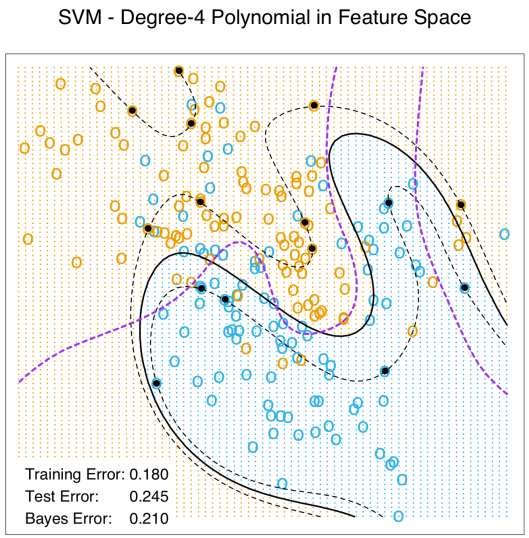
Common kernels used are:

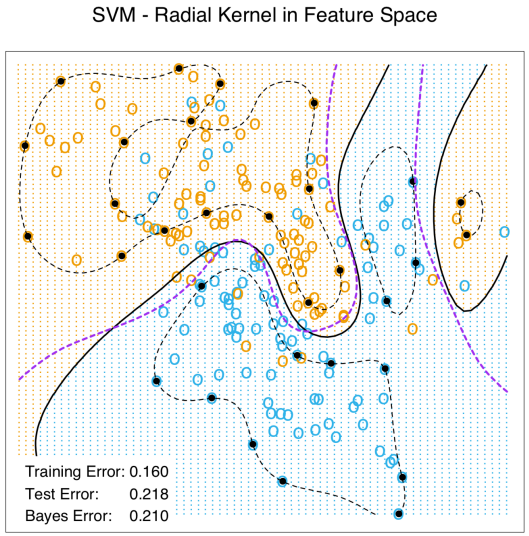
d^{th}-degree polynomial	$K(x, x') = (1 + \langle x, x' \rangle)^d$
Gaussian radial basis	$K(x, x') = \exp\{-\gamma \lVert x - x' \rVert^2\}$
Neural network	$K(x, x') = \tanh(\kappa_1 \langle x, x' \rangle + \kappa_2)$

Replacing x with a kernel is common in machine learning and is often referred to as the "kernel trick"

Non-linear SVM

- Using kernels allows for highly non-linear decision functions
- The Gaussian radial basis kernel corresponds to an infinite dimensional feature space (i.e. All data with unique points x are linearly separable)
  - We still use the non-separable formulation to reduce overfitting





Source: ESL Ch. 12

The SVM as a Penalization Method

Recall the definition of the SVM objective:

$$\min_{\beta,\beta_0,} \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{n}\xi_i$$
$$\text{subject to } y_i(h(x_i)^T\beta + \beta_0) \geq 1 - \xi_i,\ \xi_i \geq 0\ \forall i$$

For $f(x) = h(x_i)^T\beta + \beta_0$, the optimization problem

$$\min_{\beta,\beta_0} \sum_{i=1}^{n}[1 - y_i f(x_i)]_+ + \frac{\lambda}{2}\|\beta\|^2$$

is equivalent to the SVM objective, where $[a]_+ = \max(0, a)$ is the hinge loss. This resembles the form of a penalized method

Support Vector Machines for Regression (SVR)

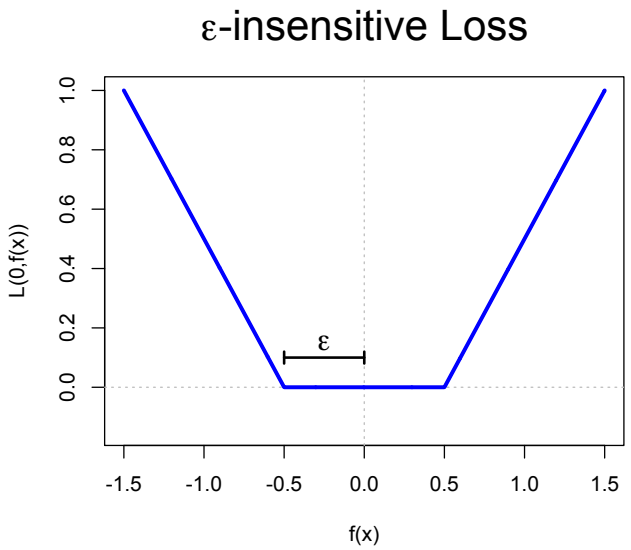
- SVM methods have been extended to regression

$$\min_{\beta,\beta_0} \sum_{i=1}^{n} \|y_i - f(x_i)\|_\epsilon + \frac{\lambda}{2}\|\beta\|^2$$

  where $\|c\|_\epsilon = \max(0, |c - \epsilon|)$
- Typical SVR setup tries to retain some of the desirable properties of support vector classification
- By using different loss functions and penalization terms, SVR can resemble many (penalized) regression methods (e.g. OLS, quantile regression, ridge regression, LASSO, etc.)

ϵ-insensitive loss functions

- Errors of size ϵ or less are ignored
- Larger values of ϵ lead to more sparse solutions
- Reminiscent of maximum margin in classification SVM
- The linear ϵ-insensitive loss function has been compared to the loss functions used in robust regression



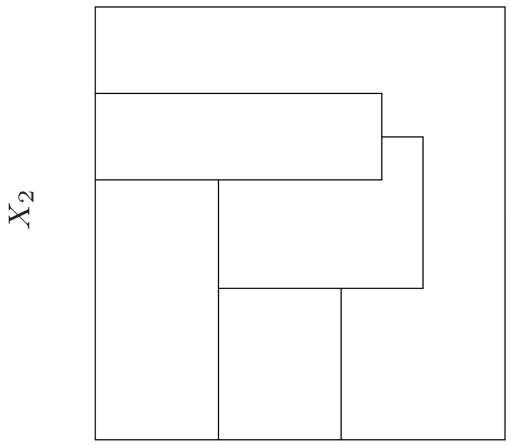
Outline

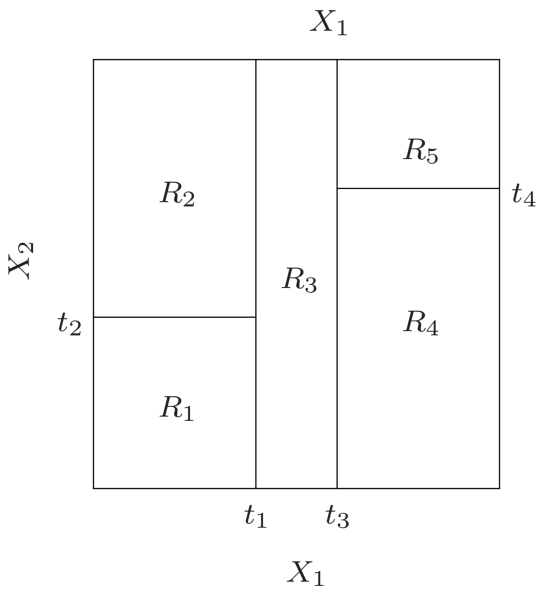
Introduction to Tree Based Methods

- Binary decision trees
  - Decision tree methods recursively partition the feature space into a set of rectangles and then fit a simple model in each one
  - Decision trees are easily interpretable
    - Popular among medical scientists
  - There are tree based methods for classification and regression
- Bagging
  - Bagging methods fit a (simple) model on several bootstrap samples of the data and average the results over the bootstrap replicates
  - Works best for high-variance low-bias estimators

Partitions

- Not all partitions are possible with decision trees
  - The partition in the top figure cannot be obtained from recursive binary splitting
  - The Partition in the bottom figure can
- With sufficient sample size, Trees can capture complex relationships

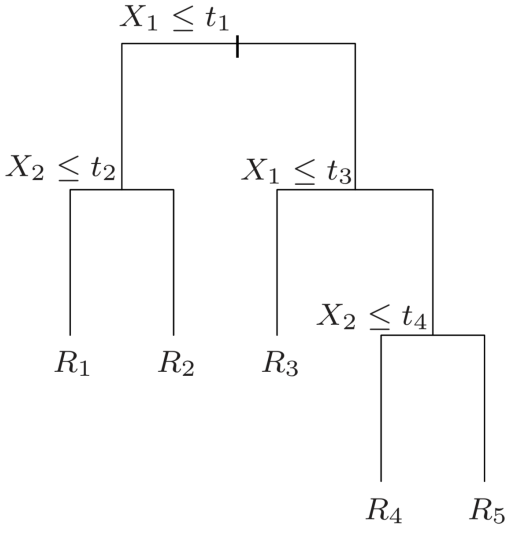


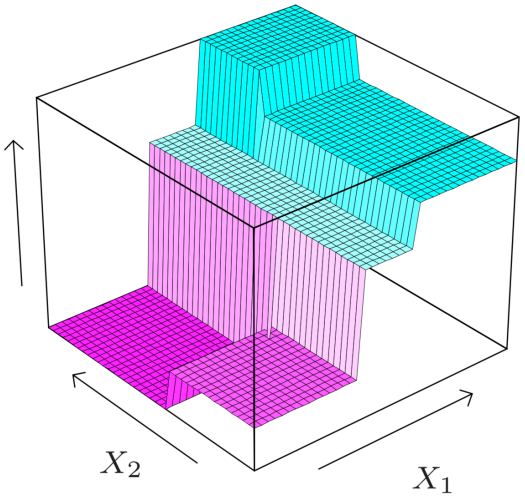


Source: ESL Ch. 9

Tree Example

- Top figure shows the tree diagram corresponding to the partition in the previous slide
  - Observations that meet the criteria go to the left, otherwise they go to the right
- Bottom figure shows a potential prediction surface for the tree





Source: ESL Ch. 9

Classification and Regression Trees

- The training data consists of n pairs of $(x_1, y_1), \ldots, (x_n, y_n)$ with $x_i \in \mathbb{R}^p$
  - For classification $y_i \in \{1, \ldots, k\}$
  - For regression $y_i \in \mathbb{R}$
- Trees are grown with a greedy algorithm
  - At each terminal node, the daughter nodes are chosen to minimize the error according to a prespecified splitting criteria
- Single trees are usually overfit leading to a high variance low bias estimator
- Fully grown trees are "pruned" to balance the bias variance trade-off

Definition of Random Forests

The original random forest algorithm, proposed by Leo Breiman, was the application of bagging to binary decision trees

- Trees can capture complex interaction structures in the data
- Decision trees, grown sufficiently deep, have high variance and low bias
- Each tree is identically distributed
  - The expectation of the average of B trees is the same as the expectation of the individual trees
- These qualities make trees ideal for bagging

Random Forest algorithm

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 combination among the m variables.
      iii. Split the node into two daughter nodes.
2. Output the ensemble of trees $\{T_b\}_1^B$.

Details of Random Forests

- For classification, the predicted class is a simple majority based on a vote from each tree
- For regression, the random forest prediction is an average of the individual tree predictions

Tuning parameters

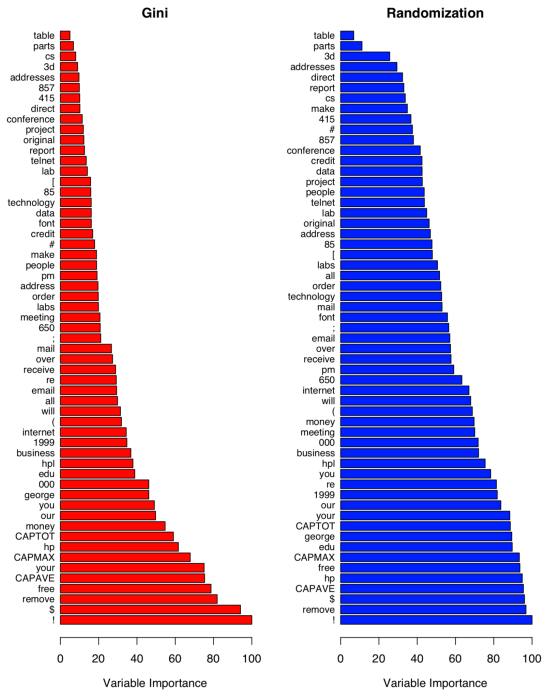
- Candidate variables at each split m
  - Default for classification $m = \sqrt{p}$
  - Default for Regression $m = \frac{p}{3}$
- Minimum node size n_{min}
- Number of trees B

Out of Bag Samples

- On average, about 37% of observations will not appear in a given bootstrap sample
- These out of bag (OOB) samples can be used to estimate the out of sample error rate
- For each observation (x_i, y_i) construct its OOB predictor by averaging only the trees for which (x_i, y_i) was not in the bootstrap sample
- The OOB error rate can be used to determine the optimal number of trees B

Variable Importance

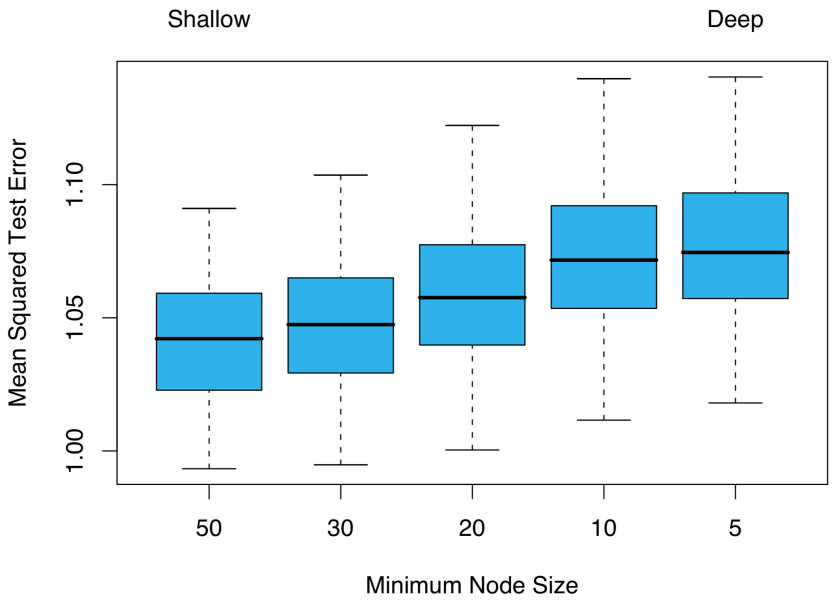
- One variable importance measure (red) is based on how often a variable is used and how much it improves the splitting criteria
- The OOB variable importance measure (blue) is based on the predictive ability of a variable compared to a random reshuffling of that variable



Source: ESL Ch. 15

Random Forests and Overfitting

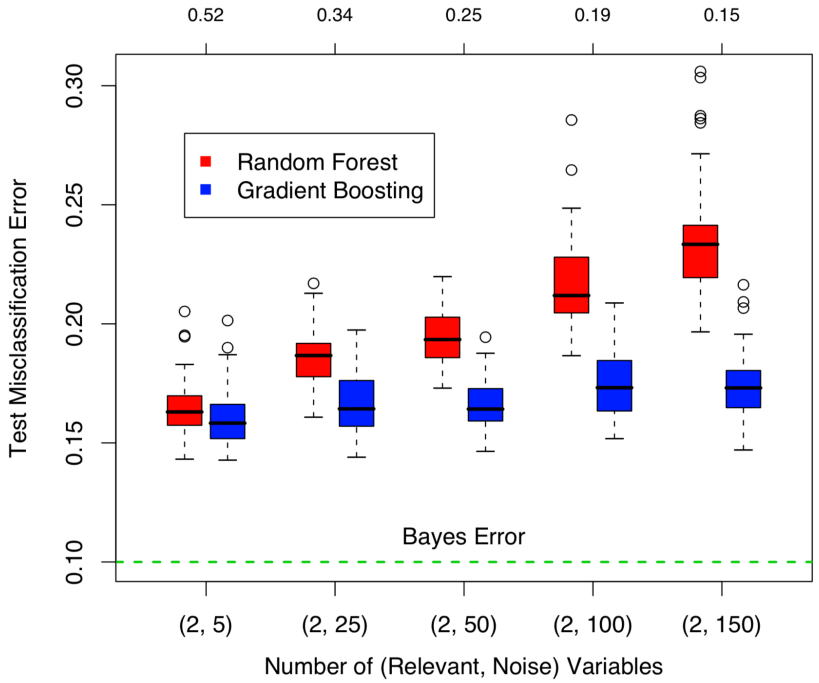
- Some claim that random forests cannot overfit the data
  - Increasing the number of trees B does not result in decreased out-of-sample performance
  - However, fully grown trees can result in a model that is too rich leading to unnecessary variance



Source: ESL Ch. 15

Random Forests and Overfitting

- The performance of a random forest suffers when it is unlikely that a relevant variable will be chosen at a given split
- For comparison, gradient boosting is able to disregard noise variables



Source: ESL Ch. 15

Analysis of Random Forests

- The limiting form ($B \to \infty$) of the random forest regression is

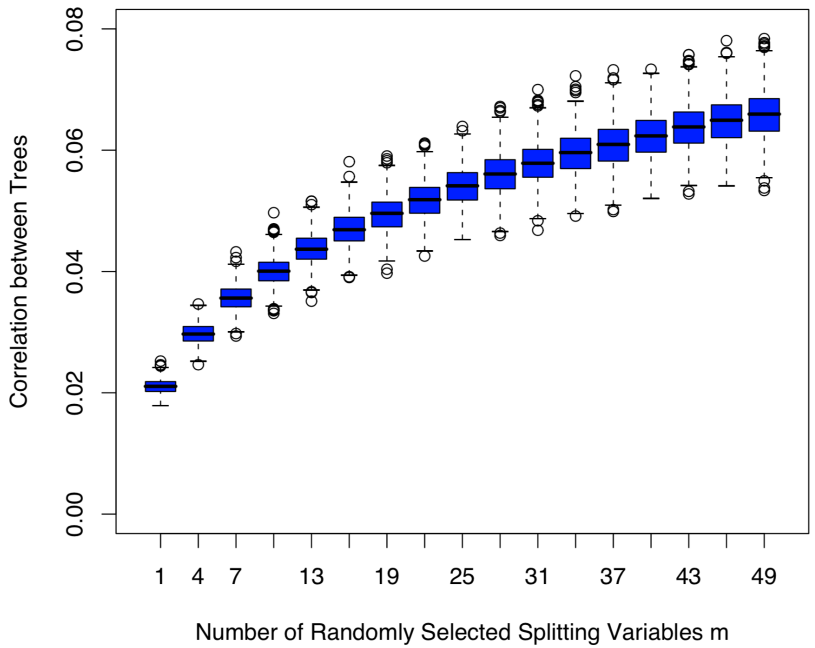
$$\hat{f}_{\text{rf}}(x) = E_{\Theta|\mathbf{Z}} T(x; \Theta, \mathbf{Z})$$

- Consider estimation at a single point x, the limiting variance of the estimate is

$$\lim_{B \to \infty} \operatorname{Var} \hat{f}_{\text{rf}}(x) = \rho(x)\sigma^2(x)$$

where

$$\rho(x) = \operatorname{corr}[T(x; \Theta_1(\mathbf{Z})), T(x; \Theta_2(\mathbf{Z}))]$$
$$\sigma^2(x) = \operatorname{Var} T(x; \Theta, \mathbf{Z})$$



Source: ESL Ch. 15

Bias of Random Forests

The bias of a random forest is the same as the bias of any individual tree in the forest $T(x; \Theta(\mathbf{Z}))$

$$\begin{aligned}\text{Bias}(x) &= \mu(x) - E_{\mathbf{Z}} \hat{f}_{\text{rf}}(x) \\ &= \mu(x) - E_{\mathbf{Z}} E_{\Theta|\mathbf{Z}} T(x; \Theta(\mathbf{Z}))\end{aligned}$$

The bias of a random forest tree is typically greater than the bias of a fully grown, un-pruned tree grown to $\mathbf{Z}$

- Bias increases due to randomization and the reduced number of unique observations in the bootstrap sample

Randomness in Random Forests

- RFs incorporate randomness to create unique trees
  - Bootstrap samples
  - Random selection of variables considered at each split
- Randomization reduces correlation between trees, which gives better performance
- Example: Extremely Randomized Trees (ERT) [Geurts et al 2005]
  - Uses full data for each tree (Not bootstrap samples)
  - Random selection of variables considered at each split
  - A single randomly drawn split-point is considered for each variable
  - Selects variable/split-point that optimizes splitting criteria
  - Can reduce bias and variance compared to traditional RF

Outline

AdaBoost Algorithm (HTF Ch. 10)

1. Initialize observation weights $w_i = 1/n$, $i = 1, \ldots, n$
2. For $m = 1, \ldots, M$:
   2.1 Fit a classifier $G_m(x)$ to training data weighted proportional to w_i, e.g., $G_m = \text{argmin}_{G_m} \sum_{i=1}^{n} w_i L(y_i, G_m(x_i)) + B(G_m)$
   2.2 Compute
   $$err_m = \frac{\sum_{i=1}^{n} w_i 1_{\{y_i \neq G_m(x_i)\}}}{\sum_{i=1}^{n} w_i}$$
   2.3 Compute $\alpha = \log((1 - err_m)/err_m)$
   2.4 Set $w_i \leftarrow w_i \cdot \exp\left[\alpha_m 1_{\{y_i \neq G_m(x_i)\}}\right]$, $i = 1, \ldots, n$
3. Use final classifier $G(x) = \text{sign}\left\{\sum_{m=1}^{M} \alpha_m G_m(x)\right\}$

Outline

Random Survival Forests

- Survival trees are an extension of binary trees to right censored data
- Each terminal node is split into two daughter nodes using a predetermined survival criterion (e.g. Log-rank test statistic)
- Trees are grown under the constraint that a terminal node should have no less than $d_0 > 0$ unique failures
- The Cumulative hazard function (CHF) is calculated in each terminal node (i.e. Nelson-Aalen Estimator)
- The node level CHFs are averaged to obtain the ensemble CHF.

RIST Motivation

- Tree-based survival regression can be robust under violation of the restrictive proportional hazards assumptions
- Censored data is hard to use — censored observations are typically only used to calibrate the risk sets of the log-rank statistics
- We can't obtain as much information from censored data as there is in noncensored data, but what is the best we can do?
- Can we avoid the requirement of a minimum number of observed failure events in each terminal node?

Data Setup

- Let $X = (X_1, \ldots, X_p)$ denote a set of covariates from a feature space $\mathcal{X}$
- The failure time T given $X = x$ is generated from the distribution function $F_x(\cdot) = 1 - S_x(\cdot)$
- The censoring time C given $X = x$ has conditional distribution function $G_x(\cdot)$
- The observed data are (Y, δ, X), where $Y = \min(T, C)$ and $\delta = I\{T \leq C\}$
- Assume T and C are independent given covariates X, $(T \perp C|X)$
- Also assume that there is a maximum length of follow-up time τ

Imputation motivation

- A censored observation will always fall into one of the following categories:
  - The true survival time T is larger than τ so that we would not observe it, even if the subject started at time 0 and was followed to the end of study
  - The true survival time T is less than τ so that we would observe the failure if the subject started at time 0 and there was no censoring prior to the end of study
- The category is masked whenever a subject is censored
- How to classify censored observations and how to impute values for them if they fall into either category

Algorithm for tree fitting

(1) Survival tree model fitting
(2) Conditional survival distribution
(3) One-step imputation for censored observations
(4) Refit imputed dataset and further calculation
(5) Final prediction

Survival Tree Model Fitting

- B independent trees are fit to the entire training dataset with the ERT model:
  - In each terminal node, select m covariates along with one random split point per non-constant covariate
  - The log-rank test statistic is used to determine the best split
  - Each terminal node is split again until no further splitting can be done without causing a terminal node to have fewer than n_{min} events (i.e., observations with $\delta = 1$)
- Each terminal node is treated as a group of homogeneous subjects for purposes of estimation and inference.

Imputation

- For the ith individual, estimate the survival function for each tree $\hat{S}_b^i(t)$
- Averaging over B trees, we have the forest-level survival function $\hat{S}_i = \frac{1}{B}\sum_{b=1}^{B}\hat{S}_b^i$
- Generate a new observation Y_i^* from the forest-level survival function, and treat it as the observed failure time
  - Note that the observed failure events in the dataset are not modified by this procedure
- Independently generate B imputed datasets, fit a single ERT to each of them, and pool the B trees to calculate new survival function estimates

Recursion

- For each of the B imputed datasets, fit a single ERT to each of them, and pool the B trees to calculate new survival function estimates
- The new survival function estimates can be used to ganerate another set of B imputed datasets
- This recursive approach can be repeated multiple times prior to the final step
- Denote the process involving q imputations as q-fold RIST, or simply RIST_q

Why RIST Works

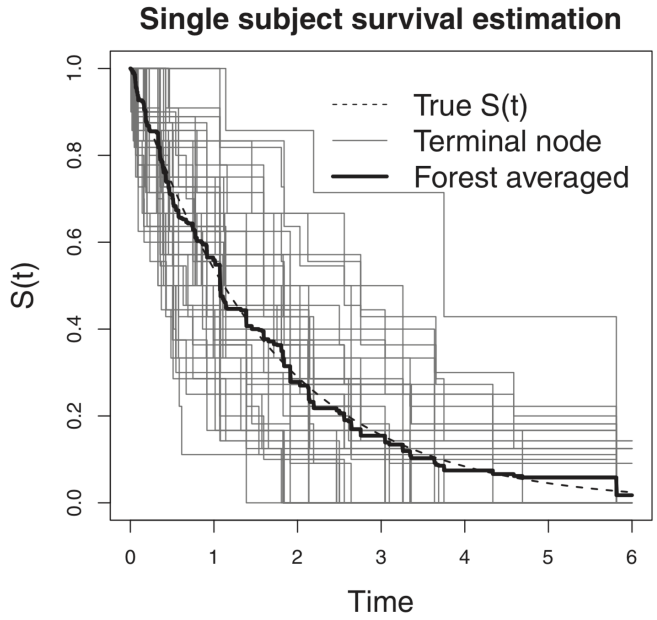
- Since the entire training set is used to build each single tree, ERT can build larger models (i.e., with more terminal nodes) compared with RF, which uses bootstrap samples
- After the first imputation cycle, additional observed events are created that allow each tree to grow even deeper
- The random generation of the imputed values provides sufficient diversity which will help eliminate overfitting

Why RIST Works, cont.

- The RIST algorithm can be viewed in the context of a Monte Carlo EM algorithm
  - The random generation of imputed values can be viewed as the Monte Carlo E-step without taking the average of all randomly generated sample points
  - The survival tree fitting procedure is explicitly an M-step to maximize the nonparametric model structure
- The imputation procedure preserves the information carried by censored observations, and introduces an extra level of diversity into the next level of model fitting
- Diversity is one of the driving forces behind the success of ensemble methods

Diversity and Forest Averaging

- An interesting phenomenon of diversity can be seen when averaging the terminal node survival function estimation over the forest
- Even though an individual terminal node estimation (using n_{min} observed events) could have a high variance or be largely biased, the overall forest estimation will still be very accurate



Source: Zhu & Kosorok (2012)

Outline

Identifying Heterogeneous Treatment Effects

- We want to use data to draw inferences about the causal effect of a treatment
- We now have enough data to meaningfully explore heterogeneity of treatment effects in several subgroups
- We can reduce the rate of false positives by specifying in advance which subgroups will be analyzed
- However, this can make it difficult to discover strong but unexpected treatment effect heterogeneity
- We need methods that yield valid asymptotic confidence intervals for the true underlying treatment effect

Data and Setup

- The training data consist of n iid replicates of
  - Feature Vector: $X_i \in [0,1]^p$
  - Response: $Y_i \in \mathbb{R}$
  - Treatment indicator: $A_i \in \{0,1\}$
- We use the potential outcomes framework
- Our goal is to estimate the treatment effect at x

$$\tau(x) = \mathbb{E}\left[Y_i^{(1)} - Y_i^{(0)} \middle| X_i = x\right]$$

- To estimate this quantity, we assume unconfoundedness

$$\{Y_i^{(0)}, Y_i^{(1)}\} \perp A_i | X_i$$

Treatment Effect Estimation

For the bth tree, let L be the leaf that contains the point x and define the tree level estimated treatment effect as

$$\hat{\tau}_b(x) = \frac{1}{|\{i : A_i = 1, X_i \in L\}|} \sum_{\{i:A_i=1, X_i \in L\}} Y_i - \frac{1}{|\{i : A_i = 0, X_i \in L\}|} \sum_{\{i:A_i=0, X_i \in L\}} Y_i$$

The causal forest estimate is found by averaging over the tree level estimates

$$\hat{\tau}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{\tau}_b(X)$$

Asymptotic Inference

- We want a consistent estimator with a well-understood asymptotic sampling distribution
- With asymptotic normality we can test hypotheses and make confidence intervals

$$(\hat{\tau}(x) - \tau(x))/\sqrt{\mathrm{Var}[\hat{\tau}(x)]} \Rightarrow \mathcal{N}(0,1)$$

- To prove asymptotic normality, we need:
  - To establish conditions under which $\hat{\tau}(x) \to \tau(x)$
  - Find a consistent estimator of $\mathrm{Var}[\hat{\tau}(x)]$

Honest trees

- Showing that $\hat{\tau}(x) \to \tau(x)$ requires some conditions on the forest-growing scheme:
  - The trees used to build the forest must be grown on subsamples of the training data where the number of subsamples $s \approx n^{\beta}$ for $\beta < 1$
  - Honesty: The splitting rule must not "inappropriately" incorporate information about the outcomes Y_i
- A tree is honest if, for each training example i, it only uses the response Y_i to estimate the within-leaf treatment effect τ or to decide where to place the splits, but not both
- We briefly describe two algorithms that meet the honesty criteria

Double Sample Trees

- Input: n training examples of the form (X_i, Y_i) for regression trees or (X_i, Y_i, A_i) for causal trees. A minimum leaf size $n_{\min}$.
  1. Draw a random subsample of size s from $\{1, \ldots, n\}$ without replacement, and then divide it into two disjoint sets of size $|\mathcal{I}| = \lfloor s/2 \rfloor$ and $|\mathcal{J}| = \lceil s/2 \rceil$
  2. Grow a tree via recursive partitioning. The splits are chosen using any data from the $\mathcal{J}$ sample and (X, A) from the $\mathcal{I}$ sample, but without using Y from the $\mathcal{I}$-sample
  3. Estimate leafwise responses using only the $\mathcal{I}$-sample observations.

Double Sample Trees, cont.

- Sample splitting procedures are sometimes criticized as inefficient because they "waste" half of the training data
- Forest subampling mechanism enables us to achieve honesty without wasting any data in this sense, because we rerandomize the $\mathcal{I}/\mathcal{J}$-data splits over each subsample
  - No data point can be used for split selection and leaf estimation in a single tree
  - Each data point will participate in both $\mathcal{I}$ and $\mathcal{J}$ samples of some trees, and so will be used for both specifying the structure and treatment effect estimates of the forest
- Double-sample trees were proposed to eliminate bias, but they can reduce mean-squared error as well

Propensity Trees

- Input: n training examples (X_i, Y_i, A_i), where X_i are features, Y_i is the response, and A_i is the treatment assignment. A minimum leaf size $n_{\min}$.
  1. Draw a random subsample $\mathcal{I} \in \{1, \ldots, n\}$ of size $|\mathcal{I}| = s$ (no replacement).
  2. Train a classification tree using sample $\mathcal{I}$ where the outcome is the treatment assignment, that is, on the (X_i, A_i) pairs with $i \in \mathcal{I}$. Each leaf of the tree must have k or more observations of each treatment class.
  3. Estimate $\tau(x)$ on the leaf containing x using data from $\mathcal{J}$.
- Propensity trees can be particularly useful in observational studies, where we want to minimize bias due to variation in the propensity, e(x)

Variance of Causal Forests

To define the variance estimates, let $\tau_b^*(x)$ be the treatment effect estimate given by the bth tree, and let $N_{ib}^* \in \{0, 1\}$ indicate whether or not the ith training example was used for the bth tree.

$$\widehat{V}_{IJ}(x) = \frac{n-1}{n} \left(\frac{n}{n-s} \right)^2 \sum_{i=1}^{n} \text{Cov}_*[\tau_b^*(x), N_{ib}^*]^2$$

- The term $n(n-1)/(n-s)^2$ is a finite-sample correction for forests grown by subsampling without replacement
- This variance estimate is consistent, in the sense that $\widehat{V}_{IJ}(x)/Var[\hat{\tau}(x)] \to_p 1$

Performance of Causal Forests

- Causal forests maintain good MSE even as the dimension of the covariate space p gets large
  - The MSE can improve as p increases
  - The variance of a random forest depends on the variance of trees times the correlation between trees
  - When p is larger, the individual trees have more flexibility, reducing their correlation and decreasing the variance of the full ensemble
- The performance of the confidence intervals starts to decay when p gets large
  - As p increases, random forests tend to be dominated by bias instead of variance, so the confidence intervals are not centered

Conclusion

- The asymptotic properties of causal forests apply to other random forest methods so long as they meet the subsampling and honesty assumptions
- There are still many open questions in causal forest research
  - The bias is not well controlled for large p, splitting rules that focus on the coordinates with the greatest signal could reduce the bias
  - It is unclear from a theoretical perspective whether double sampling or propensity trees are better
  - There is no principled, automatic way of selecting the subsample size s
  - We would like to extend pointwise confidence intervals to simultaneous confidence intervals

References

1. Hastie T, Tibshirani R, and Friedman J (2011). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2nd Ed., Springer: New York.
2. Wager S and Athey S (2018). Estimation and inference of heterogeneous treatment effects using random forests. *Journal of the American Statistical Association* 113:1228–1242.
3. Zhu R and Kosorok MR (2012). Recursively imputed survival trees. *Journal of the American Statistical Association* 107:331–340.