Causal Analysis

Impact Evaluation and Causal Machine Learning with Applications in R

Chapter 5: Causal Machine Learning (1)

Table of Contents

Causal Machine Learning (CML) for Covariate Selection

- Traditional methods assume that appropriate set of covariates X is known and preselected by the researcher:
  - Requires contextual knowledge about which covariates to include.
  - Risk of ad hoc selection leading to incorrect p-values and confidence intervals.
- Causal machine learning (CML):
  - Controls for covariates in a data-driven way.
  - Provides valid inference (e.g., p-values, confidence intervals) under specific assumptions.
- Particularly useful in high-dimensional data with many potential covariates (wide data) where manual covariate selection is complicated or infeasible.

Causal Machine Learning (CML) for Covariate Selection

- Data-driven covariate selection does not remove the need for fundamental identification assumptions in causal analysis.
- CML can be applied if there is a subset of covariates that suffices to effectively tackle confounding but is a priori unknown.
- Assumption: A limited subset of covariates permits controlling for the most important confounders.
- If this assumption holds, CML can be:
  - Approximately unbiased in sufficiently large samples (bias is negligibly close to zero).
  - $\sqrt{n}$-consistent, even if confounding is not perfectly controlled for.
- Beyond estimating average effects, CML can detect effect heterogeneities across subpopulations defined by covariates X.

Table of Contents

Double Machine Learning (DML)

- Double machine learning (DML) is a CML approach for estimating the average treatment effect (ATE) or other causal effects.
- DML relies on Neyman (1959)-orthogonal functions for treatment effect estimation.
- Neyman-orthogonality implies that treatment effect estimation is relatively robust to approximation errors in the estimation of:
  - Treatment propensity score $p(X)$.
  - Conditional mean outcomes $\mu_1(X)$ and $\mu_0(X)$.
- Doubly robust (DR) estimators satisfy this robustness property.
- Advantage:
  - By incorporating both $\mu_1(X)$, $\mu_0(X)$ and $p(X)$, approximation errors enter multiplicatively into the estimation problem.
  - Therefore, small errors in either become negligible when multiplied.

Predictive ML vs. Causal ML

- CML is conceptually different from predictive machine learning.
- Aim of predictive ML: Accurately predict an outcome based on minimizing the prediction error (e.g., mean squared error).
- Causal effects of any of the predictors cannot be learned from this forecasting approach:
  - Correlation between treatment D and covariates X generally leads to a biased causal effect estimate of D on outcome Y.
  - Even without such a correlation, bias can arise if the causal effect of D on Y is small relative to the importance of X for predicting Y.

Role of Machine Learning in DML

- In DML, machine learning is not directly applied to treatment effect estimation.
- Instead, it predicts plug-in parameters, $p(X)$, $\mu_1(X)$, and $\mu_0(X)$.
- Machine learning is used to separately predict:
  - D as a function of X.
  - Y among the treated as a function of X.
  - Y among the nontreated as a function of X.
- This is motivated by the fact that covariates X are used only to tackle confounding, not for estimating causal effects.
- The causal effect of D is estimated using the sample analog of equation (4.45).

Lasso Regression (1)

- Both OLS and lasso regression minimize the sum of squared residuals.
- **Key difference:** Lasso regression (Tibshirani, 1996) includes a penalty term λ on the sum of the absolute values of the slope coefficients.
  - For $\lambda = 0$, lasso regression is equivalent to OLS.
- **Purpose of penalization:**
  - Constrain the influence of regressors when predicting the outcome.
  - Optimally balance bias and variance by shrinking the absolute coefficients of less important regressors toward zero.
- Lasso regression may even shrink some coefficients exactly to zero, effectively dropping irrelevant/less relevant regressors from the model.

Lasso Regression (2)

For predicting $\mu_1(X)$, lasso regression solves the following penalized minimization problem to obtain the coefficients:

$$(\hat{\alpha}, \hat{\beta}_1, \ldots) = \arg \min_{\alpha^*, \beta_1^*, \beta_2^*, \ldots} \sum_{i:D_i=1} (Y_i - \alpha^* - \beta_1^* X_{i1} - \beta_2^* X_{i2} - \ldots)^2 + \lambda \sum_{j=1}^{p} |\beta_j^*| \quad (5.1)$$

- p: Number of regressors (including higher-order and interaction terms of X).
- λ: Nonnegative penalization term on the sum of absolute slope coefficients.
- $|\cdot|$: Absolute value.

Choosing λ via K-Fold Cross-Validation

- λ may be chosen by a cross-validation procedure.
- Determines the optimal amount of shrinkage that minimizes the MSE for outcome prediction among candidate values for λ.
- **K-fold cross-validation:**
  - Observations are randomly divided into K nonoverlapping subsets.
  - $k \in \{1, \ldots, K\}$ is a specific subset of observations, and k_i is the subset in which observation i is situated.
  - Select λ that minimizes:

$$\sum_{i:D_i=1} [Y_i - \hat{\mu}_{1,-k_i}(X_i)]^2 \tag{5.2}$$

where $\hat{\mu}_{1,-k_i}(X_i)$ is the prediction of $\mu_1(X)$ based on coefficients estimated from observations not in i's subset k_i.

Sample Splitting

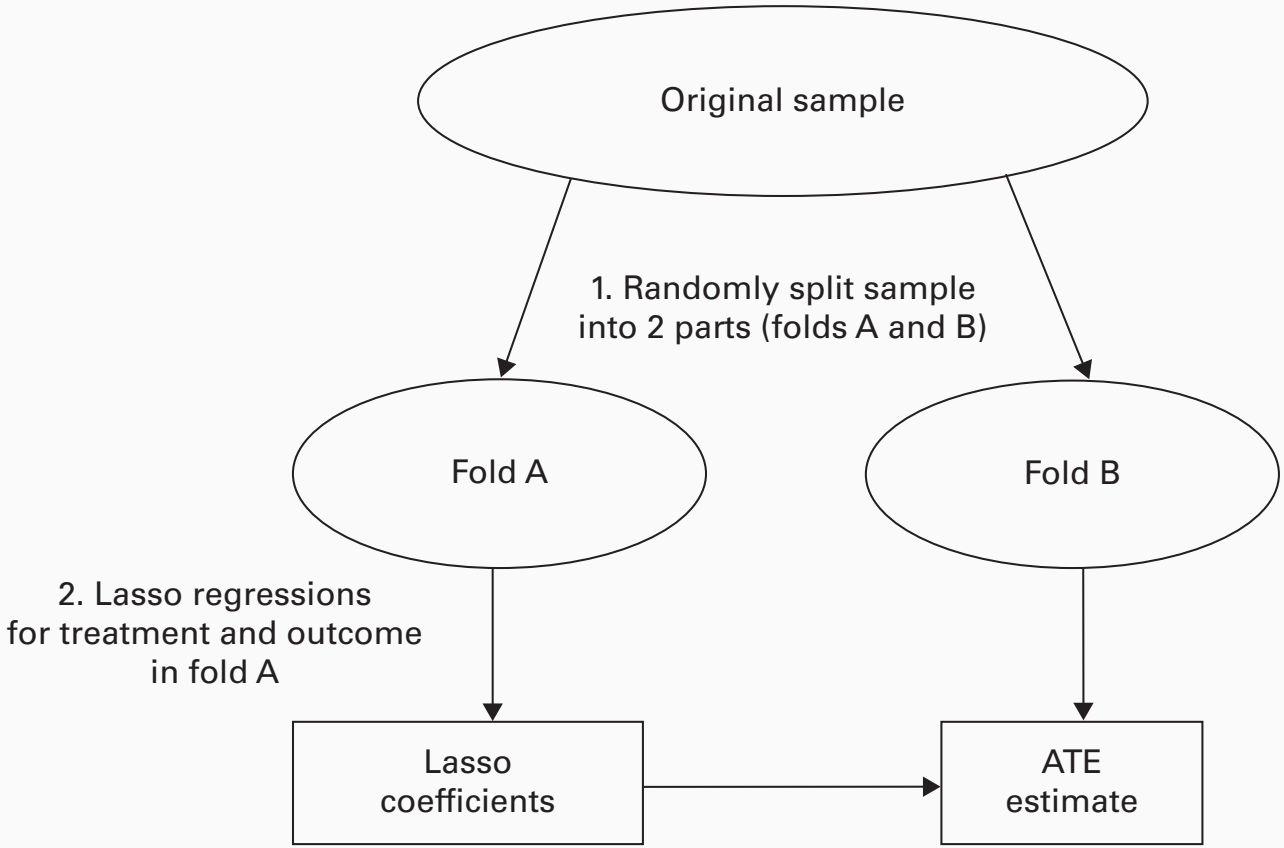
- Use independent samples for estimating the plug-in parameters $p(X)$, $\mu_1(X)$, and $\mu_0(X)$ and the treatment effects.
- **Process:**
  - Randomly divide the sample into two nonoverlapping folds.
  - Estimate $p(X), \mu_1(X), \mu_0(X)$ in the first fold.
  - Predict plug-ins and estimate treatment effects (e.g., ATE) in the second fold.
- Sample splitting avoids correlations between the two estimation steps.
- For this reason, it prevents overfitting bias related to fitting the models too much to the data points in a sample.

Cross-Fitting

- In sample splitting, only part of the data is used to estimate the causal effect, thus increasing the variance.
- This issue can be tackled by cross-fitting, which consists of swapping the roles of the folds in a second estimation round:
  - The second fold is used for obtaining the lasso coefficients for the plug-in models.
  - The first fold is used for treatment effect estimation.
- Compute the final effect estimate (e.g., ATE) by taking the average of the estimates in either fold.

Graphical Illustration

Figure 5.1: Sample splitting when estimating the ATE.



Properties of DML with Cross-Fitting

$\sqrt{n}$-consistency

- DML with cross-fitting can be $\sqrt{n}$-consistent and asymptotically normal (Chernozhukov et al., 2018).
- $\sqrt{n}$-consistency requires that plug-in estimates of $p(X)$, $\mu_1(X)$ and $\mu_0(X)$ converge to true values at a faster rate than $n^{-1/4}$.
- This rate is slower than $\sqrt{n}$ but attainable by many machine learning or deep learning algorithms under certain conditions.
- Asymptotic variance is unaffected by machine learning and cross-fitting (not higher than if covariates X were known a priori).
- Standard errors can be computed using conventional asymptotic approximations in large samples.
- TMLE (chapter 4.6) can also attain $\sqrt{n}$-consistency with the machine learning and cross-fitting, see Zheng and van der Laan (2011).

Lasso Regression and Cross-Fitting

Approximate sparsity

Number of important predictors required to decently approximate the plug-in parameters (up to minor approximation error) is small relative sample size n.

- Lasso regression attains the $n^{-1/4}$-rate requirement under approximate sparsity (Belloni, Chernozhukov, and Hansen, 2014).
- With cross-fitting, the number of important covariates or interaction/higher-order terms must be small relative to n.
- Without cross-fitting, the number of important covariates or interaction/higher-order terms must be small relative to $\sqrt{n}$ (stronger condition).

Partialling Out

Partialling out approach:

- Remove influence of covariates X on outcome Y and treatment D (Robinson, 1988) prior to assessing the treatment effect.
- **Process:**
  - Split data into two folds.
  - In the first fold, use lasso regression or postlasso OLS to obtain coefficient estimates for the models $E[Y|X]$ and $E[D|X]$.
  - In the second fold, predict outcome residuals $Y - E[Y|X]$ and treatment residuals $D - E[D|X]$.
  - In the second fold, regress outcome residuals on treatment residuals using OLS to estimate the ATE.
  - Swap roles of the data sets and average the ATE over both folds.
- $\sqrt{n}$-consistent if postlasso-based estimators of $E[Y|X]$ and $E[D|X]$ converge at least with rate $n^{-1/4}$ to the respective true models.

Table of Contents

Introduction to Further Machine Learning Algorithms

Other machine learning methods:

- Share the idea of optimally trading off the bias and variance in predicting the plug-in parameters $p(X), \mu_1(X), \mu_0(X)$.
- Applicable in DML or partialling out approaches if they satisfy regularity conditions like $n^{-1/4}$-convergence.

Ridge Regression and Elastic Nets

Ridge regression:

- Ridge regression (Tihonov, 1963) penalizes sum of squared coefficients on regressors (L^2 norm).
- Comparison of lasso and ridge regression:
  - Ridge regression cannot shrink coefficients exactly to zero, while lasso regression can (as it is based on an L^1 norm).
  - Thus, only lasso regression is able to perform variable selection.
  - Depending on the data, either ridge or lasso regression might do better for estimating $p(X), \mu_1(X), \mu_0(X)$.

Elastic nets:

- Weighted average of lasso and ridge penalization (e.g., 60% lasso, 40% ridge penalization).
- Might outperform any single method.
- Optimal weights can be determined via cross-validation.

Decision Trees: Introduction

Decision trees (Morgan and Sonquist, 1963): recursively split the covariate space (i.e., the set of possible values of X) into nonoverlapping subsets.

- **Tree structure:**
  - Nodes represent covariate values at which the sample is split.
  - Leaves are terminal subsets where no further splitting occurs.
- **Splitting process:**
  - Finds the split that entails the highest reduction in the summed sums of squared residuals across subsets.
  - Greedy approach: Split is optimal at the current stage without assessing performance several splits ahead.
  - Applied recursively: Subsets are split into further subsets.

Decision Trees: Splitting Process Example (1)

Example

- Suppose migrant status is the most predictive element in X for treatment assignment.
  - Most migrants receive treatment (e.g., a language course), while most natives do not.
  - The sum of the migrant status-specific sum of squared residuals is lower than the sum of squared residuals in the total sample:

$$\sum_{i:\mathbf{migrant}=1} (D_i - \bar{D}_{\mathbf{migrant}=1})^2 + \sum_{i:\mathbf{migrant}=0} (D_i - \bar{D}_{\mathbf{migrant}=0})^2 < \sum_{i=1}^{n} (D_i - \bar{D})^2 \qquad (5.3)$$

- Migrant subset might be further split by age (e.g., < 50 vs. ≥ 50).
  - Further splits are chosen to maximize additional reduction in the summed sum of squared residuals across all subsets.

Decision Trees: Splitting Process Example (2)

Example (cont.)

- Tree structures can be represented by regression equations:

$$D_i = \hat{\alpha} + \hat{\beta}_1 I\{\text{migrant} = 1, \text{age} < 50\} + \hat{\beta}_2 I\{\text{migrant} = 1, \text{age} \geq 50\} + \hat{V}_i. \quad (5.4)$$

  - $\hat{\alpha}$: Average treatment in the reference category (e.g., natives).
  - $\hat{\beta}_1$, $\hat{\beta}_2$: Differences in the treatment averages between the respective other subset and the reference category.
  - $\hat{V}_i$: Estimated residual of the treatment equation.
- Based on the coefficient estimates, the estimated propensity score can be computed for each subset, e.g.:

$$\hat{p}(\text{migrant} = 1, \text{age} < 50) = \hat{\alpha} + \hat{\beta}_1$$

Decision Trees: Stopping Rule

- Splitting continues until a predefined stopping rule is reached (e.g., maximum subsets or minimum observations per subset).
- There is a variance-bias trade-off concerning the number of splits in decision trees.
  - More splits imply that the observations within a subset are more homogeneous in terms of X, which reduces the bias.
  - More splits also imply fewer observations per subset to estimate conditional mean outcomes, which increases the variance.
- Use cross-validation to determine the optimal number of splits that minimizes MSE by optimally trading off bias and variance.

Decision Trees: Advantages and Disadvantages

- Advantages:
  - Nonparametric method: Splitting does not impose functional form assumptions about how D and X are associated.
  - Greater model flexibility compared to a parametric approach like lasso regression.
- Disadvantages:
  - Estimated propensity score $\hat{p}(x)$ changes discontinuously across subsets:
    - $\hat{p}(x)$ is an unweighted average of the outcomes of all observations with X values in the same subset as value x:

$$\hat{p}(x) = \frac{\sum_{i=1}^{n} D_i \cdot I\{X_i \in L_x\}}{\sum_{i=1}^{n} I\{X_i \in L_x\}} \tag{5.5}$$

    - L_x: subset (or leaf) in which value x is situated.
    - Discontinuity results from the nonsmooth indicator functions in (5.5).
  - High variance: Small data changes can entail substantially different splitting rules.

Bagged Trees (1)

Bagging (bootstrap aggregating; Breiman, 1996) mitigates the issue that a single decision tree with many leaves likely suffers from a high variance.

- Repeatedly draw bootstrap samples (with replacement) from the original data.
- Estimate trees in each bootstrap sample.
- Predict treatment/outcome by averaging predictions from all trees:

$$\hat{p}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{p}^b(x) = \frac{1}{B}\sum_{b=1}^{B}\frac{\sum_{i=1}^{n} D_i^b \cdot I\{X_i^b \in L_x^b\}}{\sum_{i=1}^{n} I\{X_i^b \in L_x^b\}} \tag{5.6}$$

  - B: Number of bootstrap samples
  - b: Indexes the parameters (e.g., treatments, covariates, leaves) in a specific bootstrap sample b.

Bagged Trees (2)

- Bagging has smaller variance than basing propensity score estimation on a single tree.
- This procedure also implies $\hat{p}(x)$ is a smooth function of X:

where

$$\hat{p}(x) = \sum_{i=1}^{n} w_i(x)^{\text{bagged}} \cdot D_i, \tag{5.7}$$

$$w_i(x)^{\text{bagged}} = \frac{1}{B} \sum_{b=1}^{B} \frac{I\{X_i \in L_x^b\}}{\sum_{j=1}^{n} I\{X_j^b \in L_x^b\}}$$

- Weights $w_i(x)^{\text{bagged}}$ are smooth in X due to averaging over the indicator functions of individual trees.
- They depend on predictive power of regressors, reducing sensitivity to weak predictors.

Random Forests

Random forests (Ho, 1995; Breiman, 2001) are a further variation of tree-based methods.

- Similar to bagged trees:
  - Rely on repeatedly drawing samples from the original data for estimating many trees.
  - Aggregate (or average) predictions across trees.
  - Random forest–based predictions can be represented by smooth weighting functions.
- Key difference: At each split, only a random subset of covariates is chosen as potential variables for splitting.
- Goal: Reduce correlation of tree structures across samples to further reduce variance in the plug-in parameter estimation.

Boosting and BARTs

Boosting (Freund and Schapire, 1997):

- Improves weak machine learners through sequential application.
- Process:
  - Apply a weak learner (e.g., a simple decision tree with few splits).
  - Compute residuals (e.g., difference between treatment and average treatment in a leaf).
  - Apply the learner to those residuals again.
  - Repeat these steps many times.
- Permits flexible approximation of the association between the covariates and the variable to be predicted.

Bayesian additive regression trees (BARTs) (Chipman, George, and McCulloch, 2010):

- Includes regularization prior in boosting process: penalizes too many splits in the tree structure to prevent excessive variance.

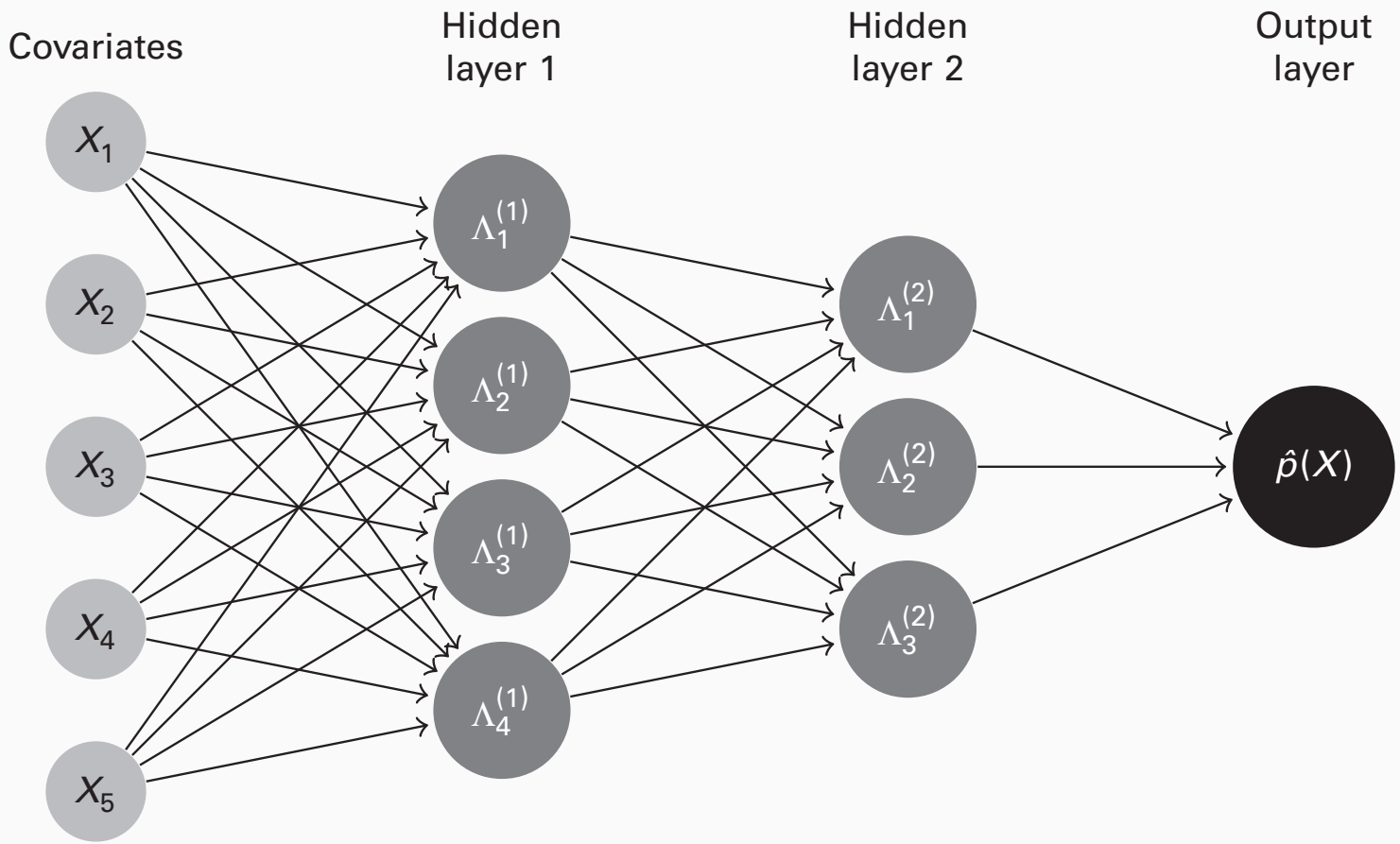
Neural Networks

Neural networks (McCulloch and Pitts, 1943; Ripley, 1996):

- Fit a system of nonlinear regression functions to flexibly model the influence of regressors/covariates on a variable to be predicted (e.g. outcome).
- Regressors serve as inputs for nonlinear intermediate functions (e.g., logistic or rectifier functions) called hidden nodes.
- The predicted values obtained from hidden nodes are inputs for the output layer (the model of the predicted variable).
- Hidden nodes can be arranged in a single layer (shallow network) or across multiple layers (deep network), where earlier layers feed into later ones that ultimately generate the outcome/treatment prediction.
- Several layers of hidden nodes allow modeling interactions.
- Trade-off: More hidden nodes and layers reduce bias, but increase variance.

Graphical Illustration

Figure 5.2: A neural network for treatment prediction.



Extensions of Basic Neural Networks

Deep learning: Refined and extended versions of basic neural networks.

- **Convolutional neural networks** (CNNs; LeCun et al., 1998):
  - Learn autonomously to create relevant predictors from unstructured data (e.g., images).
  - Use filters to represent particular spatial patterns (e.g., edges) by numeric features.
  - Features are typically aggregated by a pooling step (e.g., by taking average or maximum values over adjacent features).
  - Filtering and pooling steps can be repeatedly applied to further transform and aggregate the features.
  - Refined features are used as regressors in a standard neural network.
- **Recurrent neural networks (RNNs)**:
  - Allow feedback processes between hidden nodes in distinct hidden layers when optimizing prediction.

Support Vector Machines

Support vector machines (Boser, Guyon, and Vapnik, 1992):

- Nonlinearly transform covariates to fit a linear hyperplane in the transformed covariate space.
- For a binary treatment D, the hyperplane accurately separates the treatment values into two subsets:
  - One with mostly treated units ($D = 1$).
  - The other with mostly nontreated units ($D = 0$).
- The hyperplane is fitted to maximize the distance to the closest observation from either subset.
- This maximizes confidence in the classification into predominantly treated and nontreated subsets.

Ensemble Methods

Ensemble methods (van der Laan, Polley, and Hubbard, 2007; Zhou, 2012):

- Combine several machine learning algorithms to make predictions.
- Individual predictions of machine learning algorithms are combined using a simple or weighted average.
- Optimal weights may be determined by cross-validation to maximize predictive accuracy.
- Ensemble methods may outperform individual algorithms in terms of prediction.