Introduction to Machine Learning

DZHW Symposium "No Free Lunch: Machine Learning & Qualitative Dokumentenanalyse – Practical and Methodological Insights"

Christoph Kern[1]

c.kern@uni-mannheim.de

12.10.2021

[1]Thanks to Frauke Kreuter (LMU), Malte Schierholz (BA, IAB)

GitHub repo

https://github.com/chkern/dzhw-workshop

Outline

What is Machine Learning?

> *A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.*

– Tom Mitchell (1997)

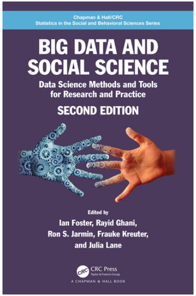
ML applications

- Make use of new, high-dimensional data sources (Mullainathan et al. 2017)
- Detect model misspecification (Hainmueller and Hazlett 2014, Kopf et al. 2010)
- Estimate propensity scores (McCaffrey et al. 2004)
- Missing data imputation (Shah et al. 2014)
- Estimate response propensities for weighting (Buskirk and Kolenikov 2015)
- Predict nonreponse in panel surveys (Kern et al. 2019)
- ...

More examples and resources

Foster, I., Ghani, R., Jarmin, R. S., Kreuter, F., and Lane, J. (Eds.). (2020). *Big Data and Social Science: Data Science Methods and Tools for Research and Practice*. 2nd Edition. Provided by the Coleridge Initiative: https://textbook.coleridgeinitiative.org/

Kern, C., Klausch, T., and Kreuter, F. (2019). Tree-based Machine Learning Methods for Survey Research. *Survey Research Methods 13*(1), 73–93. https://doi.org/10.18148/srm/2019.v1i1.7395



ML process

Machine Learning task

↓

Get and integrate data

↓

Explore and pre-process data

↓

Features

↓

Training and tuning

↓

Evaluation

↓

Deploy, maintain and update

ML basics

Unsupervised Learning

- Finding patterns in data using a set of input variables X

Supervised Learning

- Predicting an output variable Y based on a set of input variables X
  1. Learn the relationship between input and output using **training data** (with X and Y)

$$Y = f(X) + \varepsilon$$

  2. Predict the output based on the prediction model (of step 1) for **new test data** ($\sim$only X available)
- continuous Y: regression, categorical Y: classification

ML basics

Unsupervised Learning

- Finding patterns in data using a set of input variables X

Supervised Learning

- Predicting an output variable Y based on a set of input variables X
  1. Learn the relationship between input and output using **training data** (with X and Y)

$$Y = f(X) + \varepsilon$$

  2. Predict the output based on the prediction model (of step 1) for **new test data** ($\sim$only X available)
- continuous Y: regression, categorical Y: classification

ML basics

Unsupervised Learning

- Finding patterns in data using a set of input variables X

Supervised Learning

- Predicting an output variable Y based on a set of input variables X
    1. Learn the relationship between input and output using **training data** (with X and Y)

$$Y = f(X) + \varepsilon$$

    2. Predict the output based on the prediction model (of step 1) for **new test data** ($\sim$only X available)
- continuous Y: regression, categorical Y: classification

ML basics

Unsupervised Learning

- Finding patterns in data using a set of input variables X

Supervised Learning

- Predicting an output variable Y based on a set of input variables X
  1. Learn the relationship between input and output using **training data** (with X and Y)

$$Y = f(X) + \varepsilon$$

  2. Predict the output based on the prediction model (of step 1) for **new test data** ($\sim$only X available)
- continuous Y: regression, categorical Y: classification

ML basics

Supervised Learning: Find function $f(x)$ that makes optimal predictions in a **new data set**

Prerequisites:

- **Representation**: What is the *hypothesis space*, the family of functions to search over?
  - Describes possible relationships between X and Y
  - Examples: $f(x) = x'\beta$ is linear, or f is a tree.
- **Evaluation**: What is the criterion to choose between different functions?
  - Measures predictive performance
  - Examples: Mean Squared Error, Logistic Loss
- **Computation**: How is f actually calculated?
  - Speed and memory space may be limiting factors

ML basics

Supervised Learning: Find function $f(x)$ that makes optimal predictions in a **new data set**

Prerequisites:

- **Representation**: What is the *hypothesis space*, the family of functions to search over?
  - Describes possible relationships between X and Y
  - Examples: $f(x) = x'\beta$ is linear, or f is a tree.
- **Evaluation**: What is the criterion to choose between different functions?
  - Measures predictive performance
  - Examples: Mean Squared Error, Logistic Loss
- **Computation**: How is f actually calculated?
  - Speed and memory space may be limiting factors

ML basics

Supervised Learning: Find function $f(x)$ that makes optimal predictions in a **new data set**

Prerequisites:

- **Representation**: What is the *hypothesis space*, the family of functions to search over?
  - Describes possible relationships between X and Y
  - Examples: $f(x) = x'\beta$ is linear, or f is a tree.
- **Evaluation**: What is the criterion to choose between different functions?
  - Measures predictive performance
  - Examples: Mean Squared Error, Logistic Loss
- **Computation**: How is f actually calculated?
  - Speed and memory space may be limiting factors

ML basics

Supervised Learning: Find function $f(x)$ that makes optimal predictions in a **new data set**

Prerequisites:

- **Representation**: What is the *hypothesis space*, the family of functions to search over?
  - Describes possible relationships between X and Y
  - Examples: $f(x) = x'\beta$ is linear, or f is a tree.
- **Evaluation**: What is the criterion to choose between different functions?
  - Measures predictive performance
  - Examples: Mean Squared Error, Logistic Loss
- **Computation**: How is f actually calculated?
  - Speed and memory space may be limiting factors

ML basics

Table: Estimating $f(x)$

Regression methods	(tree-based) ML methods
parametric	non-parametric
linearity, additivity	flexible functional form
prior model specification	"built-in" feature selection
theory-driven	data-driven
$\rightarrow$ Inference	$\rightarrow$ Prediction

Training and test error

Training error

$$\overline{\text{err}} = \frac{1}{N}\sum_{i=1}^{N} L(y_i, \hat{f}(x_i))$$

- Prediction error based on **training data**
- with e.g. squared error loss L

Test error

$$\text{Err}_{\mathcal{T}} = \text{E}(L(Y, \hat{f}(X))|\mathcal{T})$$

- Prediction error using **test data** (given training data $\mathcal{T}$)

Training and test error

Training error

$$\overline{\text{err}} = \frac{1}{N}\sum_{i=1}^{N} L(y_i, \hat{f}(x_i))$$

- Prediction error based on **training data**
- with e.g. squared error loss L

Test error

$$\text{Err}_{\mathcal{T}} = \text{E}(L(Y, \hat{f}(X))|\mathcal{T})$$

- Prediction error using **test data** (given training data $\mathcal{T}$)

Training and test error

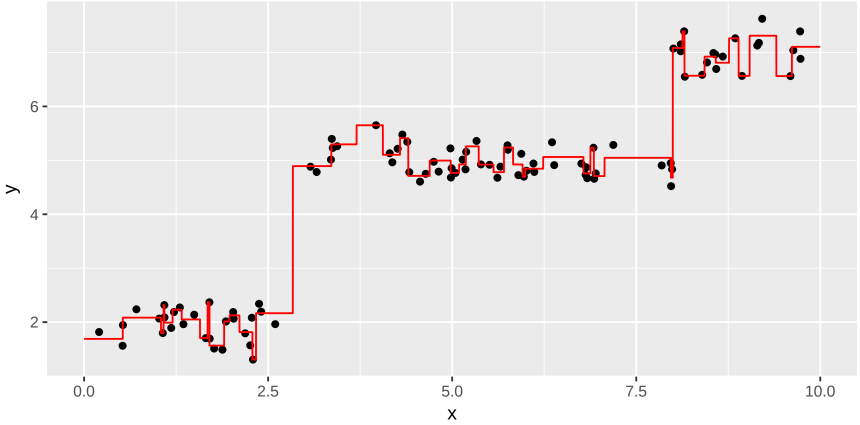
Expected test error decomposition

$$\mathrm{Err}(x_0) = \mathrm{Bias}^2(\hat{f}(x_0)) + \mathrm{Var}(\hat{f}(x_0)) + \mathrm{Var}(\varepsilon)$$

- Minimizing the (expected) test error requires
  - Low bias ($[\mathrm{E}\hat{f}(x_0) - f(x_0)]^2$) **and**
  - Low variance ($\mathrm{E}[\hat{f}(x_0) - \mathrm{E}\hat{f}(x_0)]^2$)

Bias-Variance Trade-Off

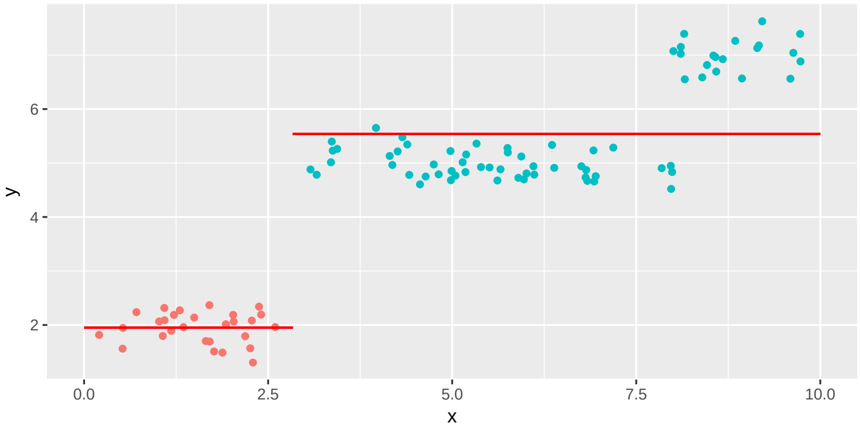
Figure: High Variance in Trees



- High Variance = Different data would lead to a different function
- Overfitting = Poor generalization to new data

Bias-Variance Trade-Off

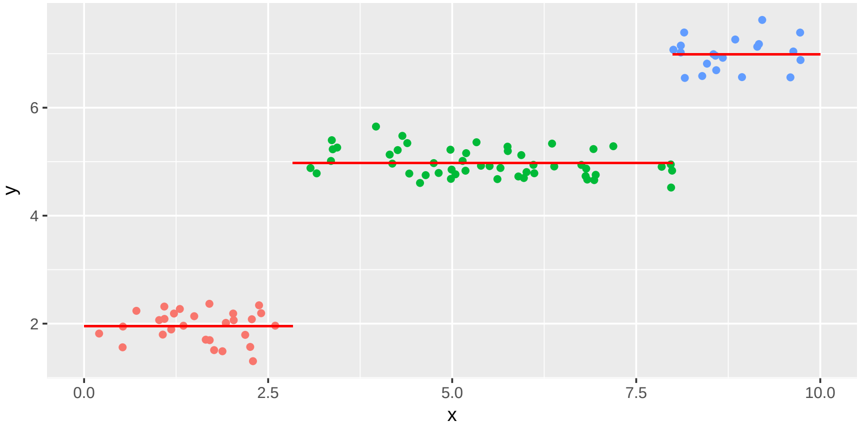
Figure: High Bias in Trees



- High Bias = Blue points are poorly predicted
- Underfitting = Function should adapt better to the data

Bias-Variance Trade-Off

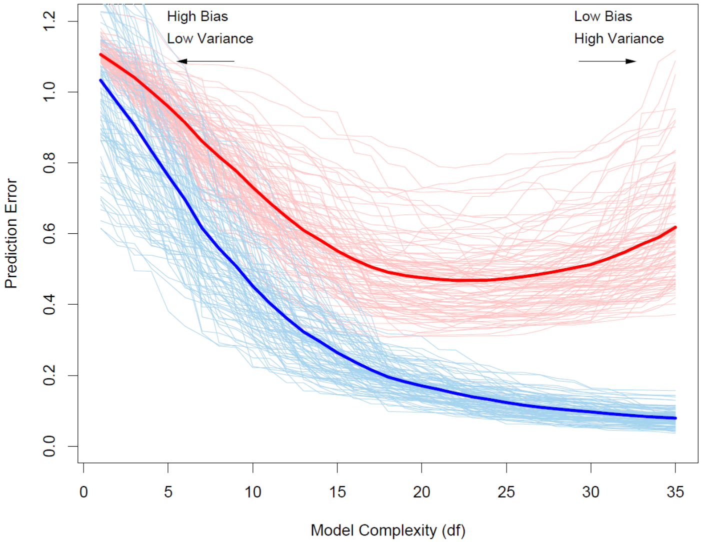
Figure: Optimal Solution



- Goal: Find optimal compromise between bias and variance

Bias-Variance Trade-Off

Figure: Training error and test error by model complexity (Hastie et al. 2009)



Bias-Variance Trade-Off

Solution: Solve

$$\arg\min_{f \in \mathcal{F}_K} \frac{1}{N} \sum_{i=1}^{N} L(f(x_i), y_i)$$

but f must come from a **restricted** hypothesis space (limited capacity)

- Tree with at most K leaves
- Regression with $\sum |\beta_j| < K$
- General form: Penalty$(f) < K$

This is **regularization** – in general form:

$$\arg\min_{f \in \mathcal{F}} \frac{1}{N} \sum_{i=1}^{N} L(f(x_i), y_i) + \lambda \cdot \text{Penalty}(f)$$

Train-test splits, Cross-Validation

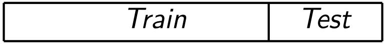
Validation set approach

Training set & test set

- Estimate prediction error on new data
  1. Fit model using one part of training data
  2. Compute test error for the excluded section

→ Model assessment

Figure: 80/20 train-test split

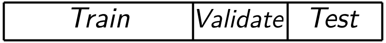


Training set, validation set & test set

- Compare models and estimate prediction error
  1. Fit models with training set
  2. Choose best model using validation set
  3. Evaluate final model using test set

→ Model selection & assessment

Figure: 50/25/25 Train-validation-test split



Validation set approach

Training set & test set

- Estimate prediction error on new data
  1. Fit model using one part of training data
  2. Compute test error for the excluded section

$\rightarrow$ Model assessment

Figure: 80/20 train-test split

Train	*Test*

Training set, validation set & test set

- Compare models and estimate prediction error
  1. Fit models with training set
  2. Choose best model using validation set
  3. Evaluate final model using test set

$\rightarrow$ Model selection & assessment

Figure: 50/25/25 Train-validation-test split

Train	*Validate*	*Test*

Leave test data untouched until the end of analysis!

Cross-Validation

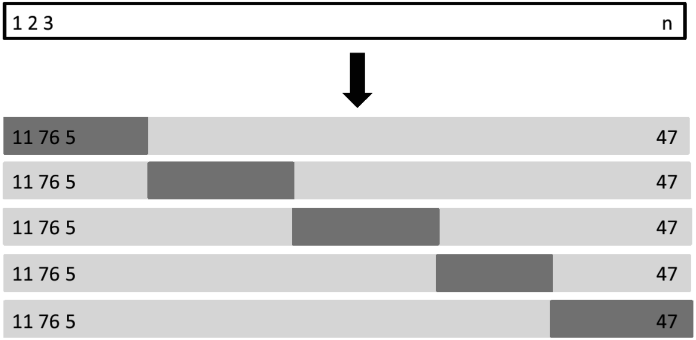
- LOOCV (Leave-One-Out Cross-Validation)
  1. Fit model on training data while excluding one case
  2. Compute test error for the excluded case
  3. Repeat step 1 & 2 n times

- k-Fold Cross-Validation
  1. Fit model on training data while excluding one group
  2. Compute test error for the excluded group
  3. Repeat step 1 & 2 k times (e.g. $k = 5$, $k = 10$)

$$CV(\hat{f}) = \frac{1}{n} \sum_{i=1}^{n} L(y_i, \hat{f}^{-\kappa(i)}(x_i))$$

Cross-Validation

Figure: 5-Fold Cross-Validation with training set and validation set (James et al. 2013)



Cross-Validation

More on data splitting

- Simple random splits
  - General approach for "unstructured" data
  - Typically 75% or 80% go into training set
- Stratified splits
  - For classification problems with class imbalance
  - Sampling within each class of Y to preserve class distribution
- Splitting by groups
  - For (temporal) structured data
  - Use specific groups (temporal holdouts) for validation

Performance evaluation

Performance measures for classification

Probabilities, thresholds and prediction for classification

$$y_i = \begin{cases} 1 & if \quad p_i > c \\ 0 & if \quad p_i \leq c \end{cases}$$

Table: Confusion matrix

		Prediction 0	Prediction 1	
Reference	0	True Negatives (TN)	False Positives (FP)	N'
	1	False Negatives (FN)	True Positives (TP)	P'
		N	P	

Performance measures for classification

Confusion matrix metrics

- Global performance
  - Accuracy: $\frac{TP+TN}{TP+FP+TN+FN}$
  - Misclassification rate: $\frac{FP+FN}{TP+FP+TN+FN}$
  - No Information rate
- Row / column performance
  - Sensitivity (Recall): $\frac{TP}{TP+FN}$
  - Specificity: $\frac{TN}{TN+FP}$
  - Positive predictive value (Precision): $\frac{TP}{TP+FP}$
  - Negative predictive value: $\frac{TN}{TN+FN}$
  - False positive rate: $\frac{FP}{FP+TN}$
  - False negative rate: $\frac{FN}{FN+TP}$

Table: Confusion matrix

		Prediction 0	Prediction 1	
Reference	0	TN	FP	N'
	1	FN	TP	P'
		N	P	

Performance measures for classification

Combined measures

- Balanced Accuracy

$$(\mathit{Sensitivity} + \mathit{Specificity})/2$$

- $F1$

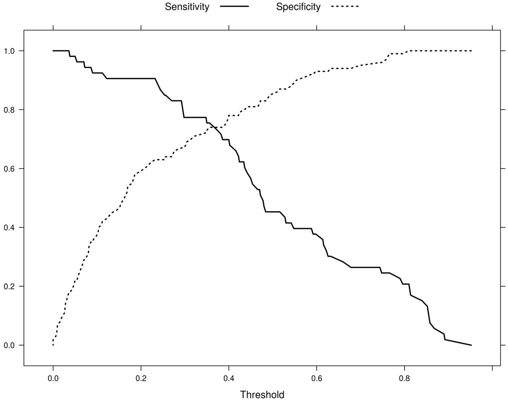
$$2 \times \frac{\mathit{Precision} \times \mathit{Recall}}{\mathit{Precision} + \mathit{Recall}}$$

- Cohen's κ
  - Compares observed and “random” accuracy

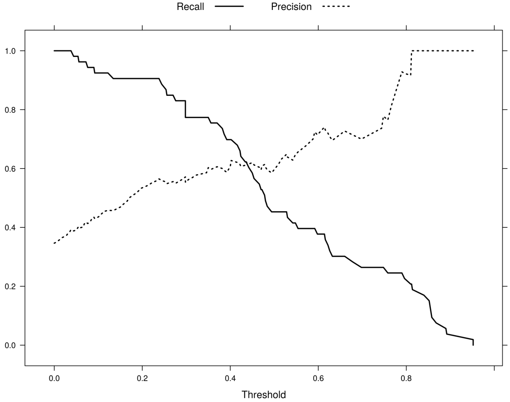
Performance measures for classification

Figure: Varying the classification threshold I

(a) Sensitivity and specificity

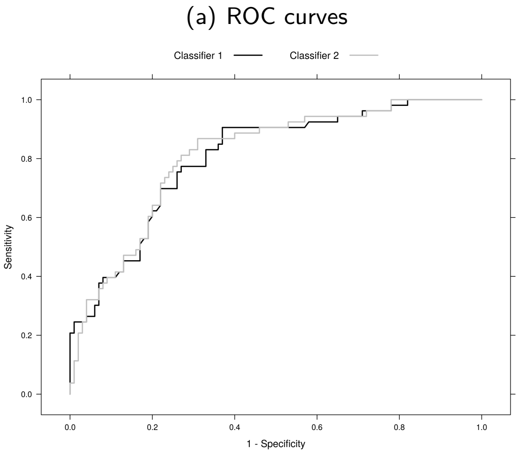


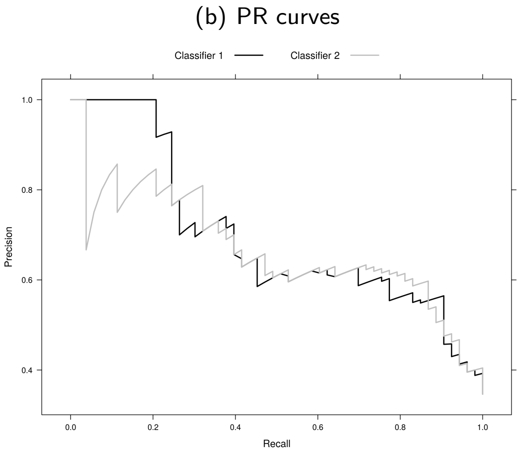
(b) Precision and recall



Performance measures for classification

Figure: Varying the classification threshold II





→ AUC-ROC: Area under the receiver operating characteristic curve

→ AUC-PR: Area under the precision–recall curve

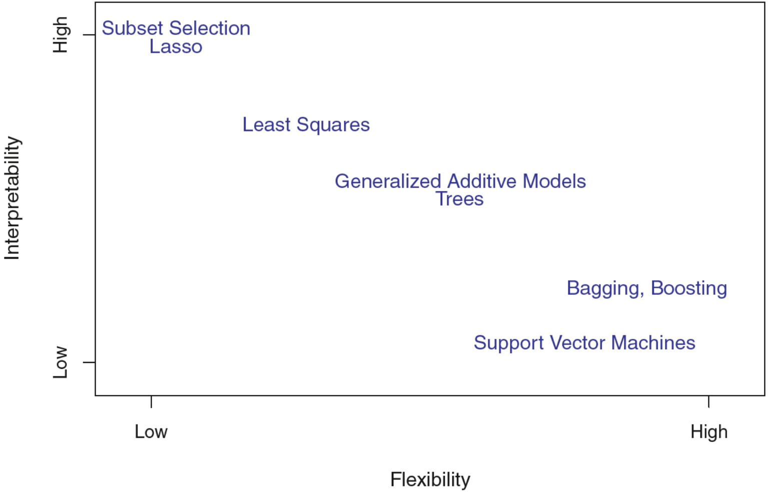
Summary

- Expected test error can be decomposed into bias and variance components
- Bias-Variance Trade-off represents decisive concept in ML
- Aim at model (setup) that generalizes well to new data (vs. over- and underfitting)
- Various types of Cross-Validation can be used for model selection and assessment
- Large number of performance metrics for classification available
- Important to compare against reference level (e.g., no information rate)

Machine Learning Methods

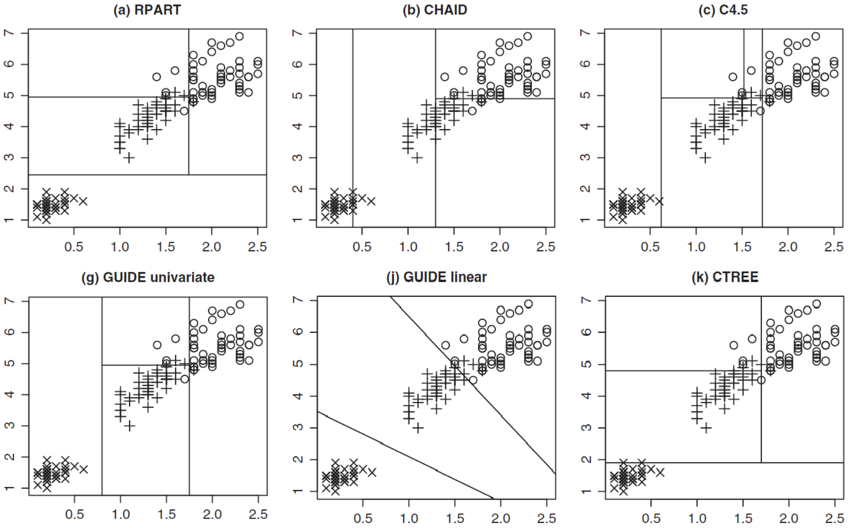
Machine Learning Methods

Figure: Flexibility-Interpretability Trade-Off (James et al. 2013)



Decision Trees

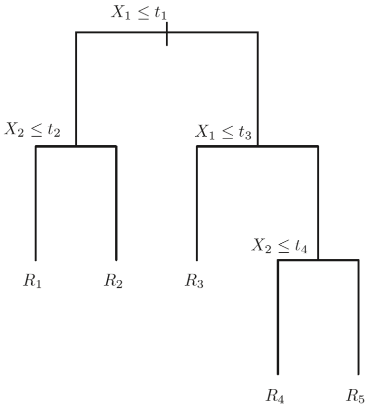
Figure: Decision Tree Algorithms (Loh 2014)



Classification and Regression Trees (CART)

- Approach for partitioning the predictor space into smaller subregions via "recursive binary splitting"
- Results in a "top-down" tree structure with...
  - Internal nodes within the tree
  - Terminal nodes as endpoints
- Can be applied to regression and classification problems
- Important building block for ensemble methods

Figure: A small tree



Growing a regression tree

Define pairs of regions for all $X_1, X_2, ..., X_p$ predictors and cutpoints c

$$\tau_L(j,c) = \{X | X_j < c\} \text{ and } \tau_R(j,c) = \{X | X_j \geq c\}$$

Find split s which maximizes the reduction in RSS

$$\Delta RSS(s,\tau) = RSS(\tau) - RSS(\tau_L) - RSS(\tau_R)$$

$$RSS(\tau) = \sum_{i \in \tau} (y_i - \hat{y})^2$$

with $\hat{y}$ being the mean of y in node τ

Growing a regression tree

Define pairs of regions for all $X_1, X_2, ..., X_p$ predictors and cutpoints c

$$\tau_L(j,c) = \{X|X_j < c\} \text{ and } \tau_R(j,c) = \{X|X_j \geq c\}$$

Find split s which maximizes the reduction in RSS

$$\Delta RSS(s,\tau) = RSS(\tau) - RSS(\tau_L) - RSS(\tau_R)$$

$$RSS(\tau) = \sum_{i \in \tau} (y_i - \hat{y})^2$$

with $\hat{y}$ being the mean of y in node τ

Growing a classification tree

Define pairs of regions for all $X_1, X_2, ..., X_p$ predictors and cutpoints c

$$\tau_L(j,c) = \{X | X_j < c\} \text{ and } \tau_R(j,c) = \{X | X_j \geq c\}$$

Find split s which maximizes the reduction in node impurity

$$\Delta I(s,\tau) = I(\tau) - p(\tau_L)I(\tau_L) - p(\tau_R)I(\tau_R)$$

Impurity measures

$$I_{Gini}(\tau) = \sum_{k=1}^{K} \hat{p}_k(1 - \hat{p}_k)$$

$$I_{entropy}(\tau) = -\sum_{k=1}^{K} \hat{p}_k \log \hat{p}_k$$

with $\hat{p}_k$ being the proportion of observations from class k in node τ

Growing a classification tree

Define pairs of regions for all $X_1, X_2, ..., X_p$ predictors and cutpoints c

$$\tau_L(j,c) = \{X | X_j < c\} \text{ and } \tau_R(j,c) = \{X | X_j \geq c\}$$

Find split s which maximizes the reduction in node impurity

$$\Delta I(s,\tau) = I(\tau) - p(\tau_L)I(\tau_L) - p(\tau_R)I(\tau_R)$$

Impurity measures

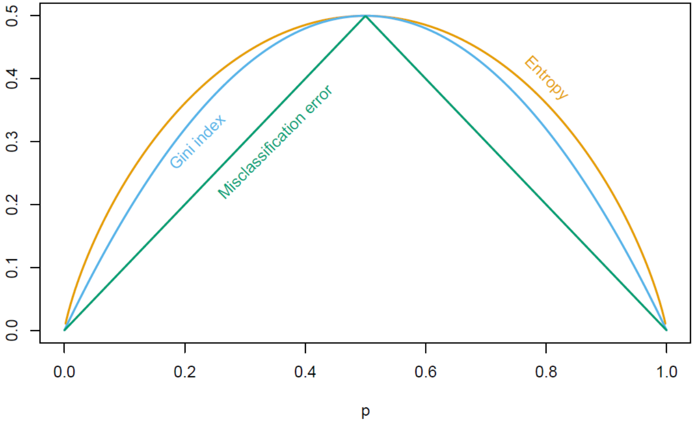
$$I_{Gini}(\tau) = \sum_{k=1}^{K} \hat{p}_k(1 - \hat{p}_k)$$

$$I_{entropy}(\tau) = -\sum_{k=1}^{K} \hat{p}_k \log \hat{p}_k$$

with $\hat{p}_k$ being the proportion of observations from class k in node τ

Tree growing

Figure: Misclassification error, Gini index & entropy (scaled, Hastie et al. 2009)



Tree growing

Algorithm 1: Tree growing process

```
Define stopping criteria;
Assign training data to root node;
if stopping criterion is reached then
    end splitting;
else
    find the optimal split point;
    split node into two subnodes at this split point;
    for each node of the current tree do
        continue tree growing process;
    end
end
```

Tree structure

A given tree

$$\mathcal{T} = \sum_{m=1}^{M} \gamma_m \cdot 1_{(i \in \tau_m)}$$

consists of a set of $m = 1, 2, \ldots, M$ nodes which can be used for prediction by...

- Regression
  - ...using the mean of y for training observations in τ_m
- Classification
  - ...going with the majority class in τ_m

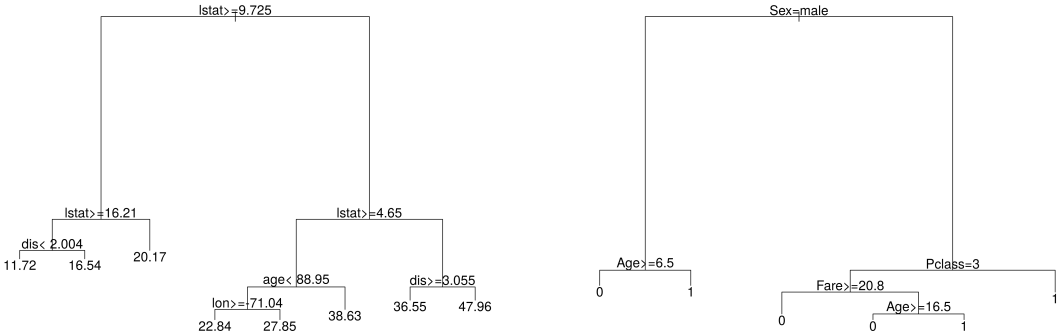
$\rightarrow$ Prediction surface: Block-wise relationship between features and outcome

Tree structure

Figure: CART examples

(a) Regression tree

(b) Classification tree



Tree pruning

Stopping rules

- Minimum number of cases in terminal nodes
- Decrease in impurity exceeds some threshold

$\rightarrow$ However, worthless splits can be followed by good splits

Cost complexity pruning

Find optimal subtree(s) $\mathcal{T}_\alpha$ by balancing tree quality $SSE(\mathcal{T}) = \sum(y_i - \hat{y}_i(\mathcal{T}))^2$ and tree size $|\mathcal{T}|$

$$C_\alpha(\mathcal{T}) = SSE(\mathcal{T}) + \alpha|\mathcal{T}|$$

- α controls the penalty on the number of terminal nodes
- α can be chosen through CV

Tree pruning

Stopping rules

- Minimum number of cases in terminal nodes
- Decrease in impurity exceeds some threshold

$\rightarrow$ However, worthless splits can be followed by good splits

Cost complexity pruning

Find optimal subtree(s) $\mathcal{T}_\alpha$ by balancing tree quality $SSE(\mathcal{T}) = \sum(y_i - \hat{y}_i(\mathcal{T}))^2$ and tree size $|\mathcal{T}|$

$$C_\alpha(\mathcal{T}) = SSE(\mathcal{T}) + \alpha|\mathcal{T}|$$

- α controls the penalty on the number of terminal nodes
- α can be chosen through CV

Bagging and Random Forests

Ensembles

Some limitations of (single) trees

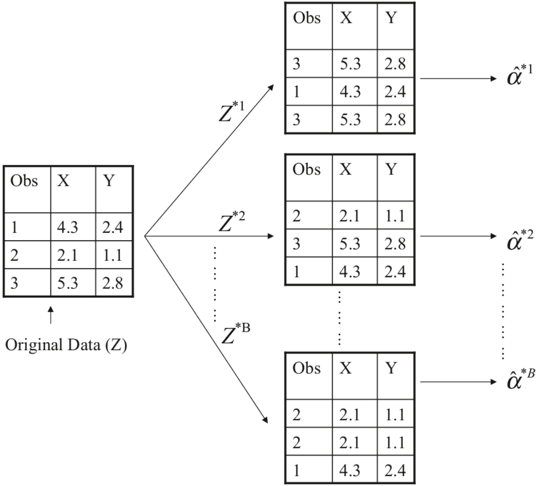
- Difficulties in modeling additive structures
- Lack of smoothness of prediction surface
- High variance / **instability** due to hierarchical splitting process

$\rightarrow$ **Ensemble methods**

- Address instability via combining multiple prediction models
- Combine diverse models into a more robust ensemble

Bootstrap

Figure: Bootstrap process (James et al. 2013)



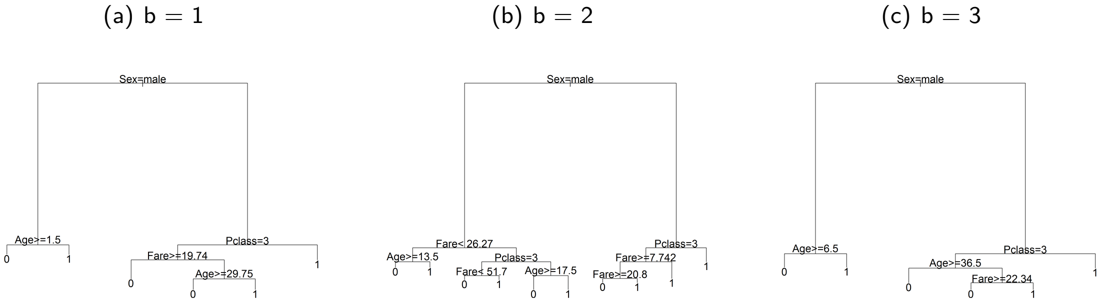
Bagging: Bootstrap Aggregating

Algorithm 2: Bagging Trees

```
Set number of trees B;
Define stopping criteria;
for b = 1 to B do
    draw a bootstrap sample from the training data;
    assign sampled data to root node;
    if stopping criterion is reached then
        end splitting;
    else
        find the optimal split point among the predictor space;
        split node into two subnodes at this split point;
        for each node of the current tree do
            continue tree growing process;
        end
    end
end
```

Bagging Trees

Figure: Bagging Trees



Random Forests

From Bagging to Random Forests

Variance of an average of B i.i.d. random variables

$$\frac{1}{B}\sigma^2$$

$\rightarrow$ Bagging: Averaging over B trees decreases variance

Variance of an average of B i.d. random variables with $\rho > 0$

$$\rho\sigma^2 + \frac{1-\rho}{B}\sigma^2$$

$\rightarrow$ **Random Forests**: Averaging over B trees with m out of p predictors per split decreases variance and decorrelates trees

Random Forests

From Bagging to Random Forests

Variance of an average of B i.i.d. random variables

$$\frac{1}{B}\sigma^2$$

$\rightarrow$ Bagging: Averaging over B trees decreases variance

Variance of an average of B i.d. random variables with $\rho > 0$

$$\rho\sigma^2 + \frac{1-\rho}{B}\sigma^2$$

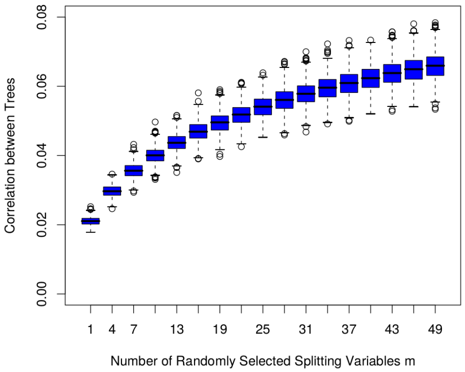
$\rightarrow$ **Random Forests**: Averaging over B trees with m out of p predictors per split decreases variance and decorrelates trees

Random Forests

The Random Forest trick (Breiman 2001)

- Sample m out of p predictors per split
- Randomization with respect to rows *and* columns
- Weaker predictors have more of a chance
- Results in diverse and *decorrelated* trees

Figure: Correlations between pairs of trees (Hastie et al. 2009)



Algorithm 3: Grow a Random Forest

```
Set number of trees B;
Set predictor subset size m;
Define stopping criteria;
for b = 1 to B do
    draw a bootstrap sample from the training data;
    assign sampled data to root node;
    if stopping criterion is reached then
        end splitting;
    else
        draw a random sample m from the p predictors;
        find the optimal split point among m;
        split node into two subnodes at this split point;
        for each node of the current tree do
            continue tree growing process;
        end
    end
end
```

Growing a Forest

A Random Forest

$$\{T_b\}_1^B$$

consists of a set of $b = 1, 2, \ldots, B$ trees which can be used for prediction by...

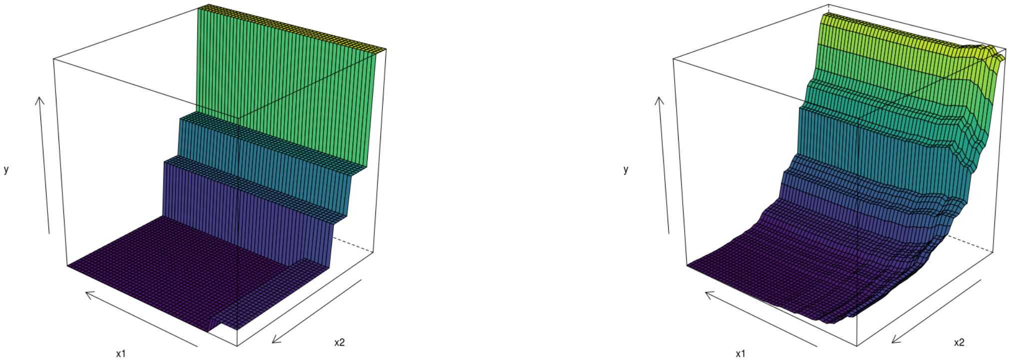
- Regression
  - Averaging predictions over all trees
  - $\hat{f}_{rf}^B(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x)$
- Classification
  - Using most commonly occurring class among all trees
  - $\hat{C}_{rf}^B(x) = \text{majority vote}\{\hat{C}_b(x)\}_1^B$
- Probability estimation
  - Using the proportion of class votes of all trees
  - Averaging predicted probabilities over all trees

RF vs. CART

Figure: Prediction surface (example)

(a) CART

(b) Random Forest



Tuning RF

Tuning Random Forests

- Predictor subset size m out of p (`mtry`)
  - Most important tuning parameter in RF
  - Starting value; $m = \sqrt{p}$ (classification), $m = p/3$ (regression)
  - Can be chosen using OOB errors based on different m
- Number of trees
  - sufficiently high (e.g. 500)
- Node size (number of observations in terminal nodes)
  - sufficiently low (e.g. 5)

Summary

- Decision Trees: Divide-and-conquer strategy that splits the data into subgroups
- No need to specify the functional form in advance (unlike regression)
- Non-linearities and interactions are handled automatically
- Limitations of (single) trees: Instability, competition among correlated predictors, biased variable selection
- Bagging, RF stabilize predictions from high-variance methods (e.g., CART)

Resources

Resources

Resources for R – ML packages

- Overview
  - https://cran.r-project.org/web/views/MachineLearning.html
- caret
  - http://topepo.github.io/caret/index.html
- mlr3
  - https://mlr3.mlr-org.com/

Resources

Resources for R – Tree-based methods

- Standard package to build CARTs: `rpart`
- Unified infrastructure for tree representation: `partykit`
- Standard package to grow RFs: `randomForest`
- Fast implementation of RFs: `ranger`

References

Breiman, L. (2001). Random forests. *Machine Learning 45*(1), 5–32.

Hastie, T., Tibshirani, R., Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. New York, NY: Springer.

James, G., Witten, D., Hastie, T., Tibshirani, R. (2013). *An Introduction to Statistical Learning*. New York, NY: Springer.

Loh, W.-Y. (2014). Fifty Years of Classification and Regression Trees. *International Statistical Review 82*(3), 329–348.

Mitchell, T. M. (1997). *Machine Learning*. Maidenhead: McGraw-Hill.

References

Buskirk, T. D., Kolenikov, S. (2015). Finding respondents in the forest: A comparison of logistic regression and random forest models for response propensity weighting and stratification. Survey Methods: Insights from the Field. `https://surveyinsights.org/?p=5108`

Hainmueller, J. and Hazlett, C. (2014). Kernel Regularized Least Squares: Reducing Misspecification Bias with a flexible and interpretable machine learning approach. *Political Analysis 22*. 143–168.

Kern, C., Klausch, T., and Kreuter, F. (2019). Tree-based Machine Learning Methods for Survey Research. *Survey Research Methods 13*(1), 73–93.

Kopf, J., Augustin, T., and Strobl, C. (2010). *The Potential of Model-based Recursive Partitioning in the Social Sciences – Revisiting Ockham's Razor*. Technical Report Number 88, University of Munich.

McCaffrey, D. F., Ridgeway, G., and Morral, A. R. (2004). Propensity score estimation with boosted regression for evaluating causal effects in observational studies. *Psychological Methods 9*(4). 403–425.

Mullainathan, S., Spiess, J. (2017). Machine Learning: An Applied Econometric Approach. *Journal of Economic Perspectives 31*(2). 87–106.

Shah, A.D., Bartlett, J.W., Carpenter, J., Nicholas, O., Hemingway, H. (2014). Comparison of Random Forest and Parametric Imputation Models for Imputing Missing Data Using MICE: A CALIBER Study. *American Journal of Epidemiology 179*(6). 764–774.