Machine Learning

A Brief Introduction

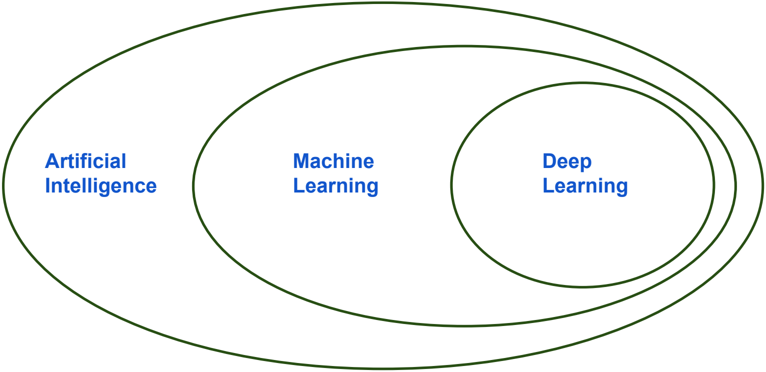
Marvin N. Wright

Leibniz Institute for Prevention Research & Epidemiology – BIPS
University of Bremen
University of Copenhagen

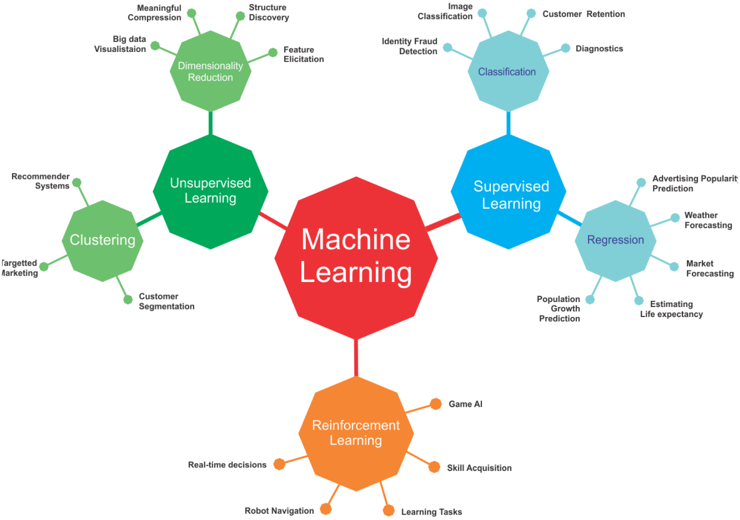
March 2023

Outline

Machine Learning

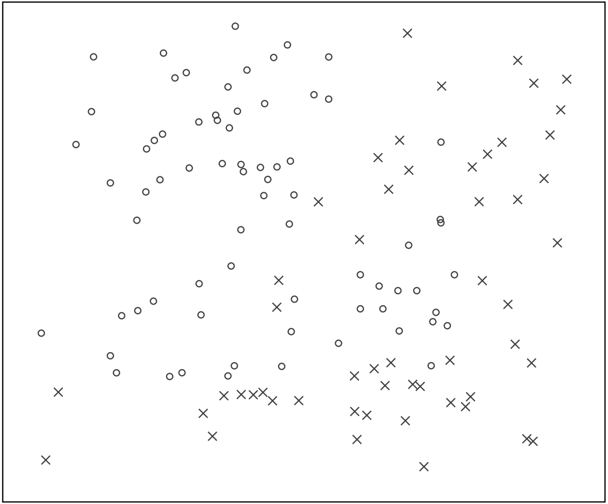


Machine Learning



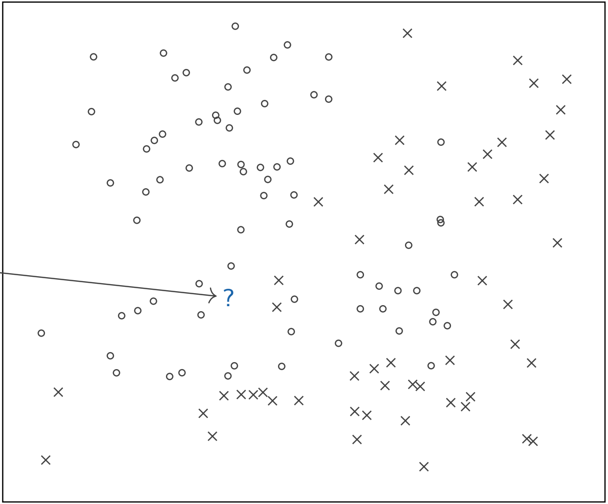
Karim 2018 ISBN 9781788997454

k-Nearest Neighbors



k-Nearest Neighbors

new
observation



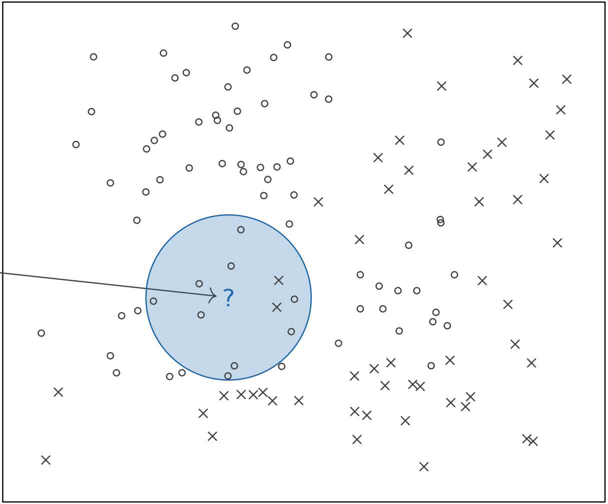
k-Nearest Neighbors

new observation

11 nearest neighbors

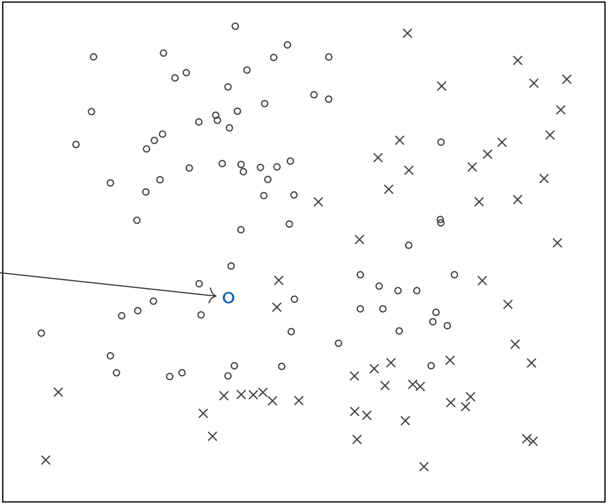
2x x

9x o



k-Nearest Neighbors

new observation



11 nearest neighbors

2x x

9x o

k-Nearest Neighbors

What is kNN formally?

- $N_k(\boldsymbol{x})$ neighborhood of $\boldsymbol{x}$ defined by k closest points $\boldsymbol{x}_i$ in training data
- $\hat{y} = \frac{1}{k} \sum_{\boldsymbol{x}_i \in N_k(\boldsymbol{x})} y_i$
- Closeness implies metric
- Standard metrics: Euclidian, Mahalanobis distance
- Generalization: Weighting schemes, e.g. $w = \frac{1}{d(x, x_i)}$
- kNN assumes: Regression function $\mathbb{E}(y \mid \boldsymbol{x})$ well approximated by locally constant function

Wettschereck et al. 1997 Artif Intell Rev 10:1-37

Outline

Example: House Prices

Predict the price for a house in a certain area

Features x				Target y
square footage of the house	**number of bedrooms**	**swimming pool (yes/no)**	**...**	**house price in US$**
1,180	3	0	...	221,900
2,570	3	1	...	538,000
770	2	0	...	180,000
1,960	4	1	...	604,000

Example: Length of hospital stay

Predict days a patient has to stay in hospital

Features x					**Target y**
diagnosis category	**admission type**	**gender**	**age**	**...**	**Length-of-stay in the hospital in days**
heart disease	elective	male	75	...	4.6
injury	emergency	male	22	...	2.6
psychosis	newborn	female	0	...	8
pneumonia	urgent	female	67	...	5.5

Example: Life Insurance

Predict risk category for a life insurance customer

Features x				Target y
job type	**age**	**smoker**	**...**	**risk group**
carpenter	34	1	...	3
stuntman	25	0	...	5
student	23	0	...	1
white-collar worker	39	0	...	2

Supervised Learning

Learn a functional relationship between **features** x and **target** y

Features x		**Target** y
People in Office (Feature 1) x_1	**Salary (Feature 2)** x_2	**Worked Minutes Week (Target Variable)**
4	4300 €	2220
12	2700 €	1800
5	3100 €	1920

$n = 3$

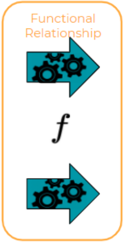
$p = 2$

$x_1^{(2)}$ $x_2^{(1)}$ $y^{(3)}$

Supervised Learning

Use labeled data to learn a model f
Use model f to predict target y of new data

	x_1	x_2	y
Already seen Data	4	4300 €	2200
	12	2700 €	1800
	15	3100 €	1920
New Data	6	3300 €	???
	5	3100 €	???



Supervised Learning

Model

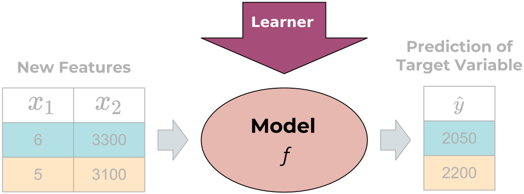
Functional relationship between **features** x and **target** y

Learner (or inducer)

Algorithm for finding model

Train Set

y	x_1	x_2
2200	4	4300
1800	12	2700
1920	15	3100



Supervised Learning

Example

- Learner: Artificial neural network (as a concept)
- Model: Actual network with learned weights

Models differ in size and complexity

- Linear model: Coefficients β
- Neural network: Weights for all units in all layers
- Decision trees: Many binary splits
- k-nearest neighbors: Complete training data

Supervised Learning

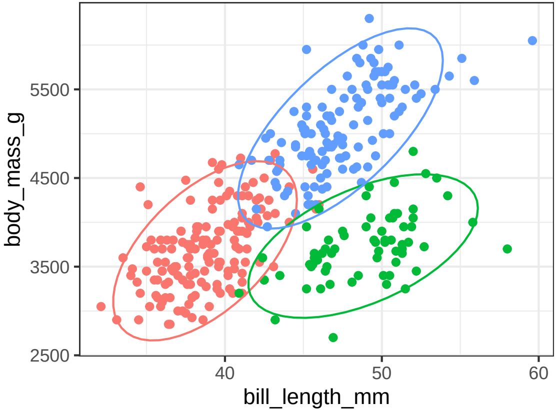
Summary

- Learn relationship between **features** x and **target** y
- Model: Learned relationship $f(x)$
- Learner: Algorithm for finding a model
- Predict $\hat{y} = f(x)$
- Later: Evaluate by comparing $\hat{y}$ with y
- Tomorrow: Understand / interpret / explain model f or predictions $\hat{y} = f(x)$

Unsupervised Learning

No **target** y available

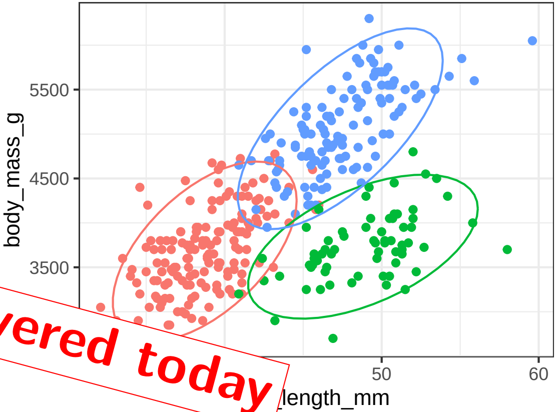
Search for patterns in the data x, e.g. clustering:



Unsupervised Learning

No **target** y available

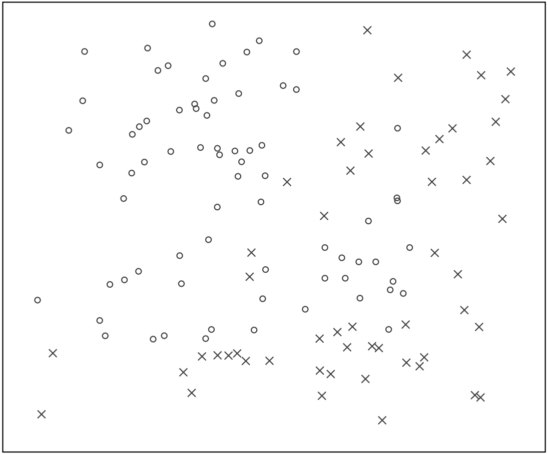
Search for patterns in the data x, e.g. clustering:



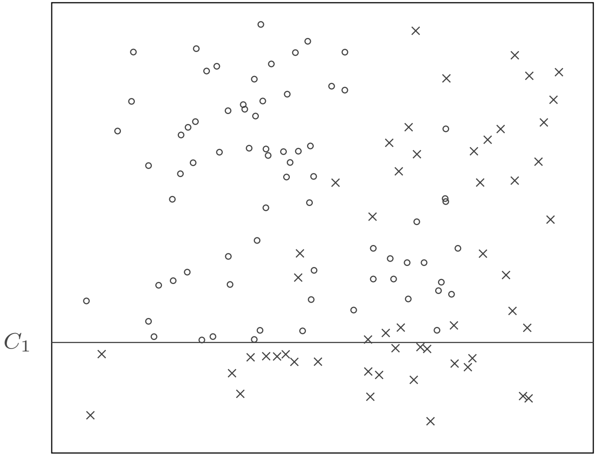
Not covered today

Outline

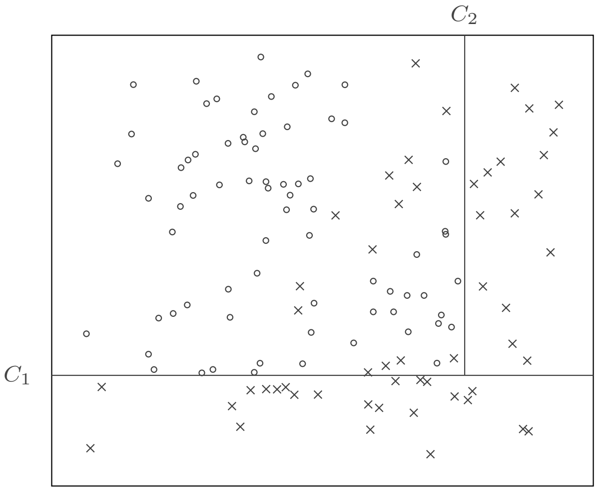
Decision Trees



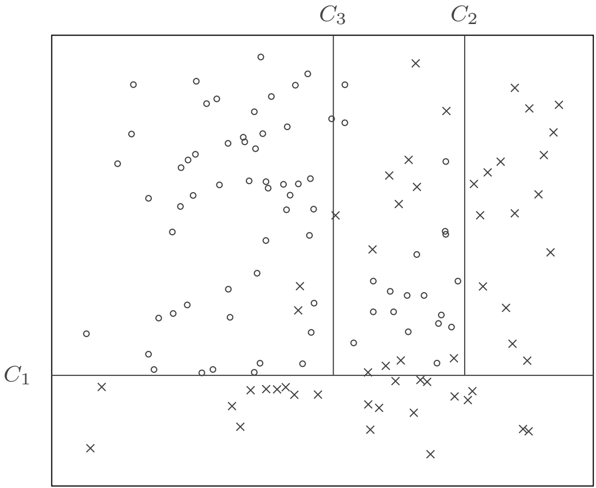
Decision Trees



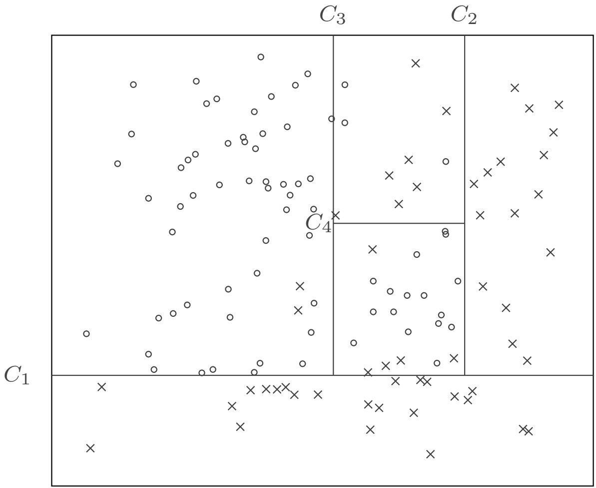
Decision Trees



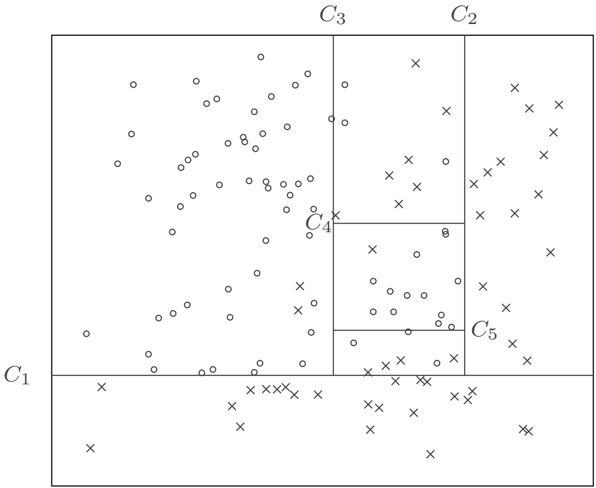
Decision Trees



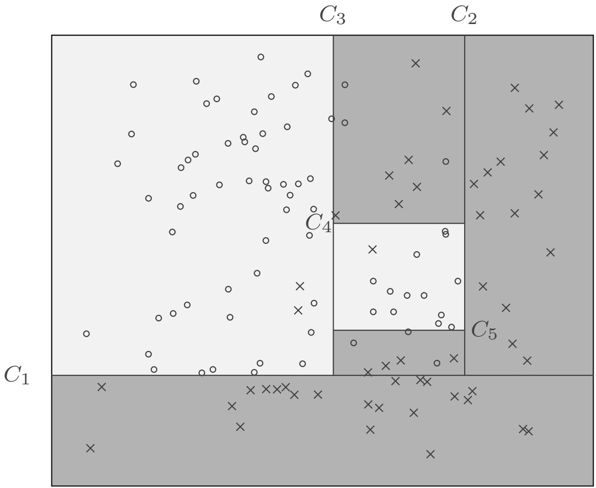
Decision Trees



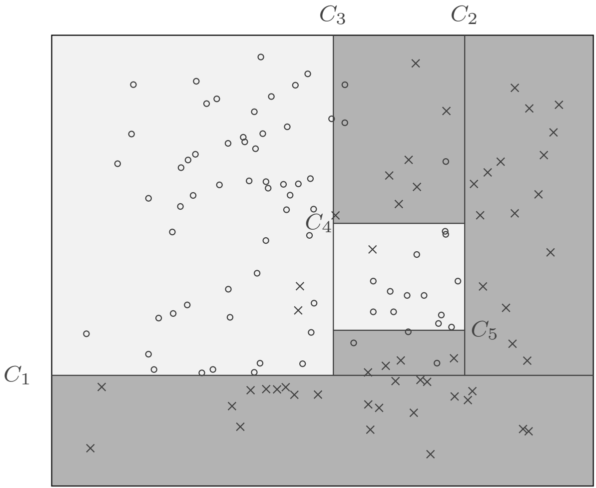
Decision Trees

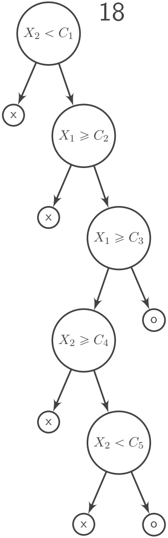


Decision Trees

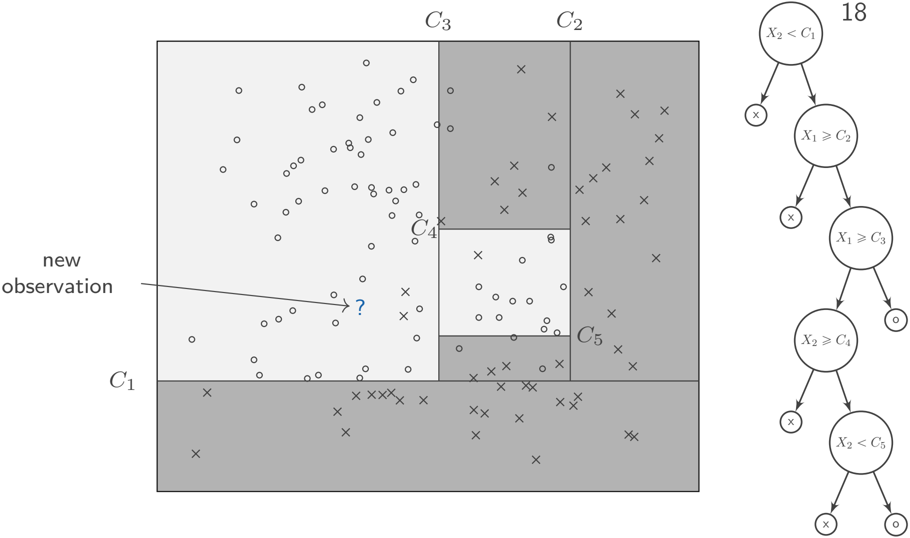


Decision Trees

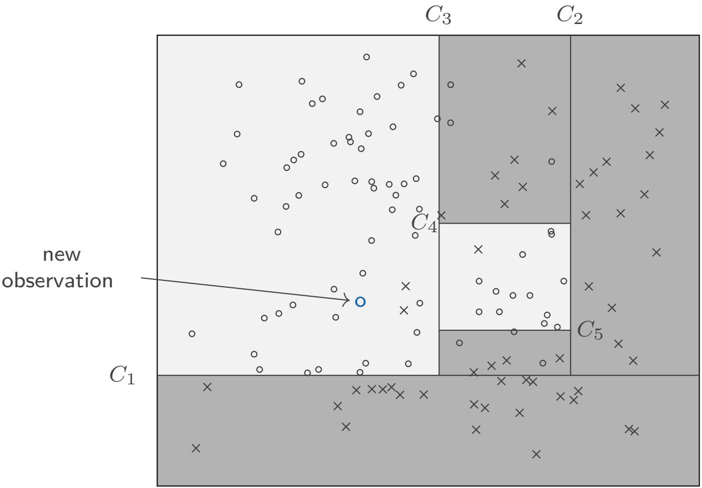


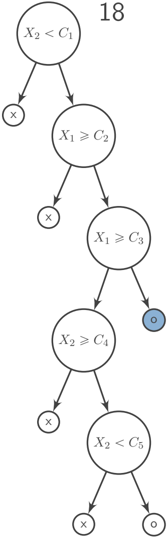


Decision Trees



Decision Trees





Decision Trees

Algorithm

1. Grow tree
2. Stop tree growing process
3. Prune back branches
4. Select optimal tree

Decision Trees

Algorithm

1. Grow tree : Recursively split nodes
   a) For each independent variable x_j, consider each possible binary split (partition), compute child node impurity
   b) Select variable x_j and split point yielding largest decrease in impurity
   c) Split in exactly two child nodes at optimal split point
2. Stop tree growing process
3. Prune back branches
4. Select optimal tree

Breiman et al. 1984 ISBN 9780412048418 • König et al. 2008 Int J Data Min Bioinform 2:289

Algorithm

1. Grow tree
2. Stop tree growing process
   a) In a node with only identical outcome (pure node)
   b) In a node with only identical variable values (no split possible)
   c) If external limit on tree complexity, tree depth or node size reached
3. Prune back branches
4. Select optimal tree

Breiman et al. 1984 ISBN 9780412048418 • König et al. 2008 Int J Data Min Bioinform 2:289

Decision Trees

Algorithm

1. Grow tree
2. Stop tree growing process
3. Prune back branches
   a) Tradeoff between complexity and accuracy
   b) Estimate accuracy in test data set
   c) Time consuming
4. Select optimal tree

Outline

Model Evaluation

How goood is a prediction model?

Compare true target y with predicted target $\hat{y}$

Examples

- How many patients correctly diagnosed?
- How many emails correctly detected as ham or spam?
- How close is the predicted price of a house to the true value?
- How close is the length of hospitalization to the true value?

Model Evaluation

Dichotomous (binary) outcome

- Proportion of correct classifications (PC); also accuracy: $\widehat{PC} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}_{y_i = \hat{y}_i}$
- Sensitivity, specificity, ROC, AUC: $\hat{\mathbb{P}}(y = 1 \mid x)$
- Brier score (BS), i.e., MSE of probability estimates; also probability score (PS): $\widehat{BS} = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{\mathbb{P}}\left(y_i = 1 \mid x_i\right)\right)^2$

Multicategory outcome

- Proportion of correct classifications (PC)
- Averaged class-wise PC
- ROC, AUC: several extensions

Model Evaluation

Continuous outcome

- MSE: $\widehat{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$
- MAE: $\widehat{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$
- RMSE: $\widehat{RMSE} = \sqrt{\widehat{MSE}}$
- Explained variance: $\hat{R}^2 = \frac{1-\widehat{MSE}}{\widehat{\mathbb{V}ar}(y)}$

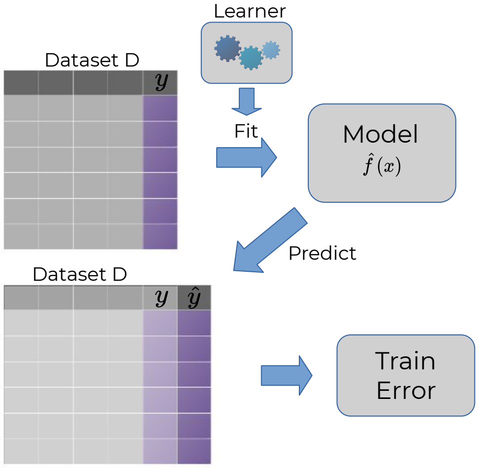
Survival outcome

- Time-dependent Brier Score
- Integrated Brier score
- C-Index

Model Evaluation

Training error

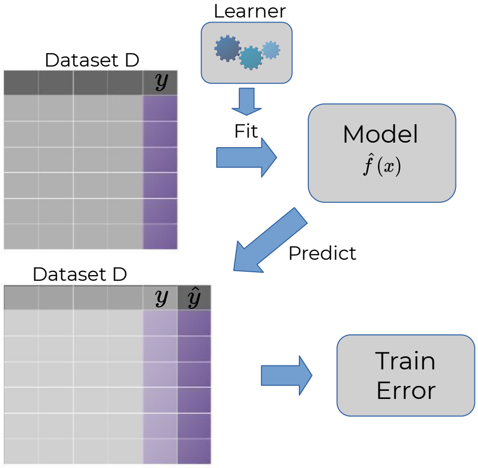
Evaluate performance on training data



Model Evaluation

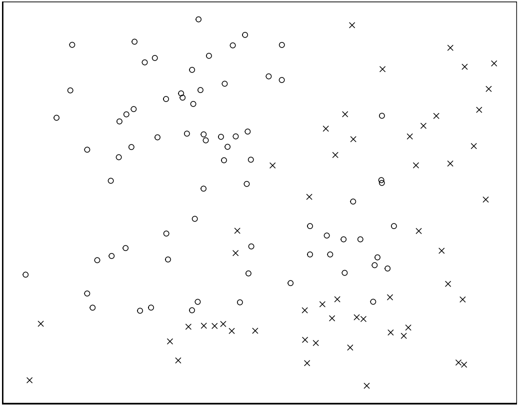
Training error

Evaluate performance on training data

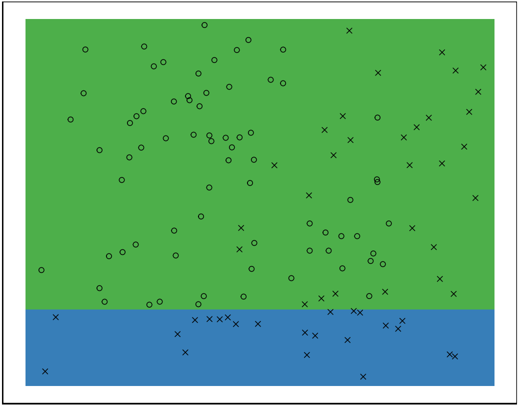


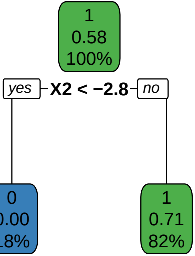
Problem: Overfitting

Overfitting

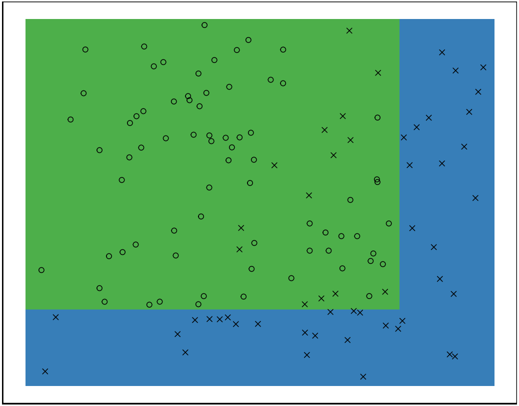


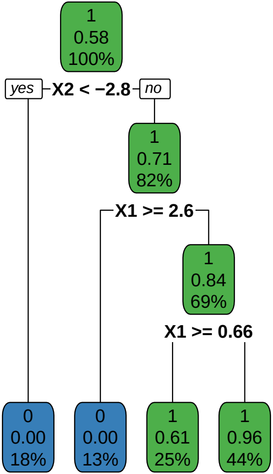
Overfitting



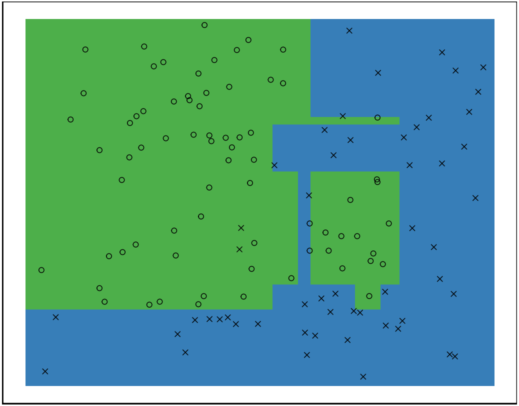


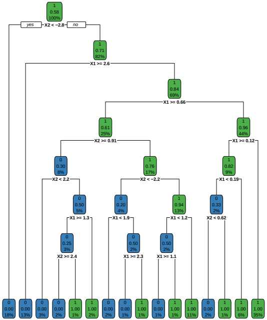
Overfitting





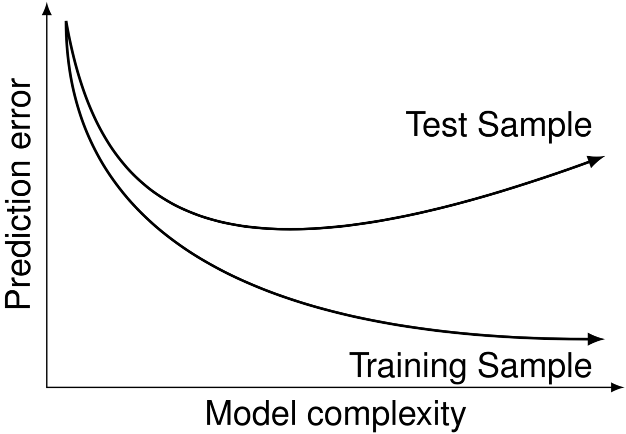
Overfitting





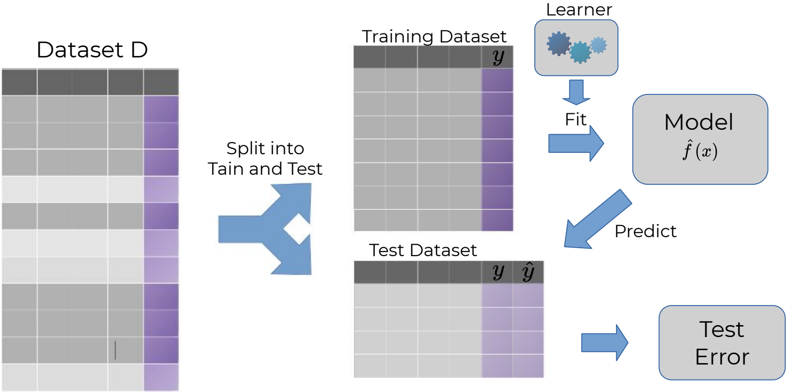
Model Evaluation

Overfitting



Model Evaluation

Test error



Model Evaluation

Training and test error

- Training error heavily biased
- Test error (almost) unbiased but variance unknown

Resampling

- Repeated training/test splits (subsampling)
- Cross validation
- Repeated cross validation
- Bootstrap

Model Evaluation

Hyperparameters

Most (all?) learners have hyperparameters, e.g.:

- k-nearest neighbors: Number of neighbors k, distance weighting, etc.
- Decision trees: Tree depth, splitting criterion, etc.
- Neural networks: Number and size of layers, activation function, regularization, etc.

Hyperparameter tuning

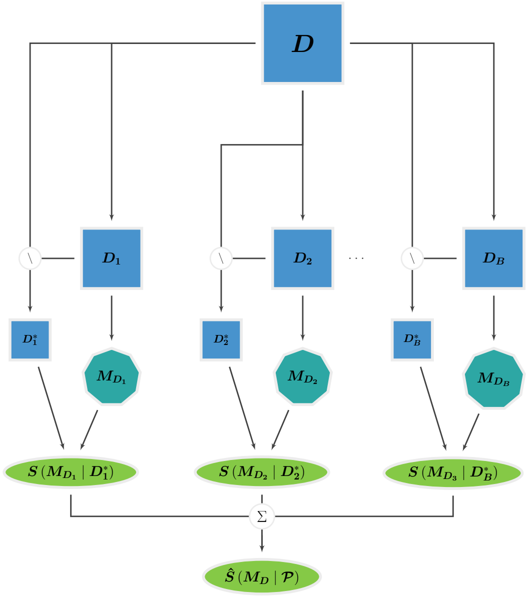
- Optimize (tune) the hyperparameters
- Do not tune and evaluate on same data

$\rightarrow$ 3-fold split into training, validation, test

$\rightarrow$ Nested resampling

Outline

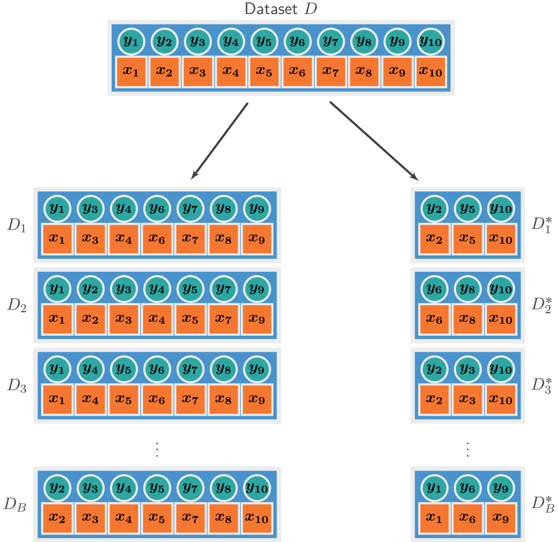
Resampling


D
$\setminus$
D_1
D_2
$\cdots$
D_B
D_1^*
D_2^*
D_B^*
M_{D_1}
M_{D_2}
M_{D_B}
$S\left(M_{D_1} \mid D_1^*\right)$
$S\left(M_{D_2} \mid D_2^*\right)$
$S\left(M_{D_3} \mid D_B^*\right)$
Σ
$\hat{S}\left(M_D \mid \mathcal{P}\right)$

Resampling

- Estimate performance on independent data
- Used for
  - Performance estimation
  - Hyperparameter tuning
  - Model selection
- Resampling based performance estimation
  1. Split dataset in several (smaller) datasets D_b
  2. On each dataset D_b:
     - 2.1 Train learner
     - 2.2 Estimate performance on $D_b^* = D \backslash D_b$
  3. Aggregate performance estimates

Subsampling

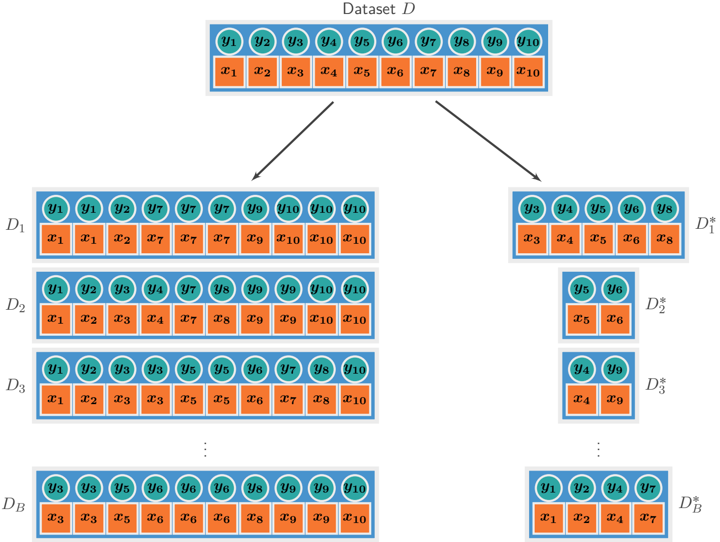


Subsampling

- Sample B training datasets D_b from D without replacement, usually $n_b = \frac{2}{3}n$
- Use $D_b^* = D \backslash D_b$ as test datasets
- D_b and D_b^* disjunct
- D_1 and D_2 not disjunct
- D_1^* and D_2^* not disjunct
- Performance estimator biased
- No optimal B, usually $100 < B < 1000$
- Special case with $B = 1$: Single train/test split (holdout)

Molinaro et al. 2005 Bioinformatics 21:3301–07 • Bischl et al. 2012 Evol Comput 20:249–75

Bootstrapping

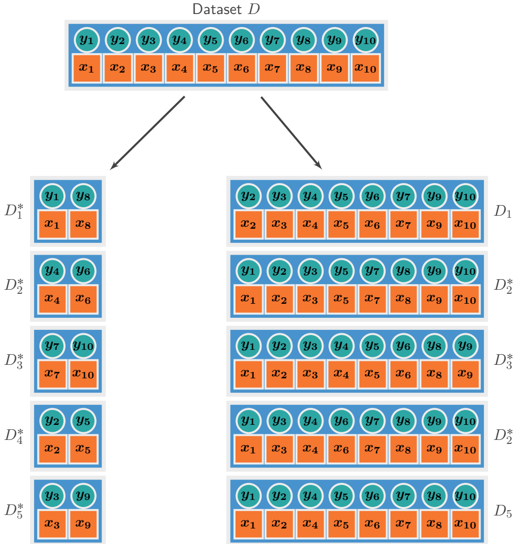


Bootstrapping

- Sample B training datasets D_b from D with replacement, usually $n_b = n$
- Use $D_b^* = D \backslash D_b$ as test datasets
- D_b and D_b^* disjunct
- D_1 and D_2 not disjunct
- D_1^* and D_2^* not disjunct
- Performance estimator biased
- Adaptive weighting to reduce bias (.632+ bootstrap)
- Small variance (large B)
- No optimal B, usually $100 < B < 1000$

Molinaro et al. 2005 Bioinformatics 21:3301–07 • Efron & Tibshirani 1997 JASA 92:548–60 • Bischl et al. 2012 Evol Comput 20:249–75

Cross validation (CV)



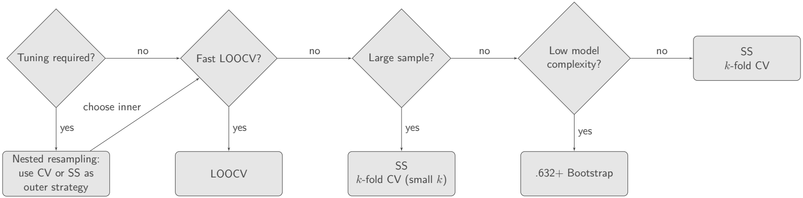
Resampling

Cross validation (CV)

- Split D in B test datasets D_b^*
- Use $D_b = D \backslash D_b^*$ as training datasets
- D_b and D_b^* disjunct
- D_1 and D_2 not disjunct
- D_1^* and D_2^* disjunct
- Special case with $B = n$: Leave-one-out CV (LOOCV)
  - Small bias, high variance
  - Long runtime
- No optimal B, usually $B = 5, 10$
  - Slightly more bias than LOOCV, but lower variance
  - Lowest B of all resampling methods $\rightarrow$ fast computation

Stone 1974 J Roy Stat Soc B Met 36:111–47 • Molinaro et al. 2005 Bioinformatics 21:3301–07
Bischl et al. 2012 Evol Comput 20:249–75

How to choose the resampling method?



Bischl et al. 2012 Evol Comput 20:249–75

Outline

Hyperparameter Tuning

Hyperparameters

Learners have hyperparameters, e.g.:

- Number of nearest neighbors k
- Depth of a tree
- Number of features to consider in each split of a random forest (mtry)
- Number of boosting iterations
- Kernel of SVM
- Architecture of neural network

Most learners have several hyperparameters

Have to be jointly optimized

Hyperparameter Tuning

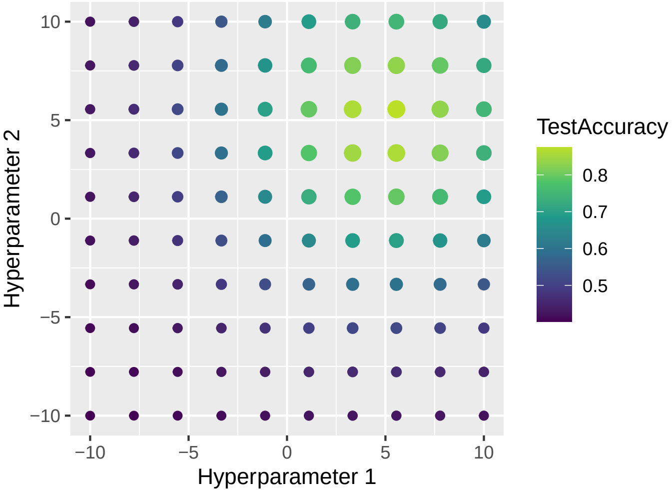
Search entire parameter space

- All possible combinations
- Grid search
- Randomly select combinations
- Model-based optimization

Use resampling

- Evaluate each parameter combination on all resampling iterations/folds
- Choose parameter maximizing aggregated performance measure

Grid search



TestAccuracy
0.8
0.7
0.6
0.5
Hyperparameter 2
Hyperparameter 1
10
5
0
−5
−10
−10
−5
0
5
10

Hyperparameter Tuning

Grid search

Advantages

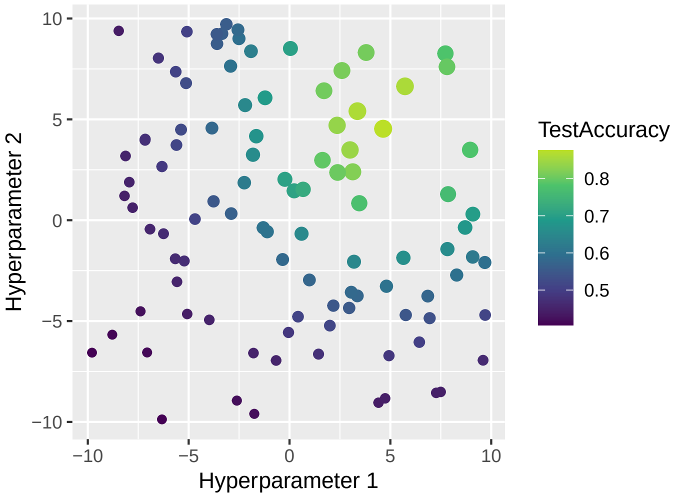
- Easy to implement
- All parameter types possible
- Easily parallelized

Disadvantages

- Computationally intensive
- Inefficient: Searches large irrelevant areas
- Arbitrary: Which values / discretization?

Hyperparameter Tuning

Random search



Random search

Advantages

- Same as grid search: Easy to implement, all parameter types possible, trivial parallelization
- Easy to adjust to computational budget
- No discretization
- Superior performance compared to grid search

Disadvantages

- Computationally intensive
- Inefficient: Searches large irrelevant areas

Hyperparameter Tuning

Model-based optimization

Surrogate model

Learn relationship between hyperparameters and prediction performance

Algorithm

1. Pick initial configuration (e.g. random)
2. Learn surrogate model
3. Predict new configuration with surrogate model
4. Repeat steps 2 and 3

Hyperparameter Tuning

Model-based optimization

Advantages

- All parameter types possible
- Efficient: Focus on promising areas
- Superior performance compared to grid and random search

Disadvantages

- Computationally intensive
- Non-trivial parallelization
- Harder to implement

Turner et al. 2020 NeurIPS PMLR 133:3

Outline

Benchmarking

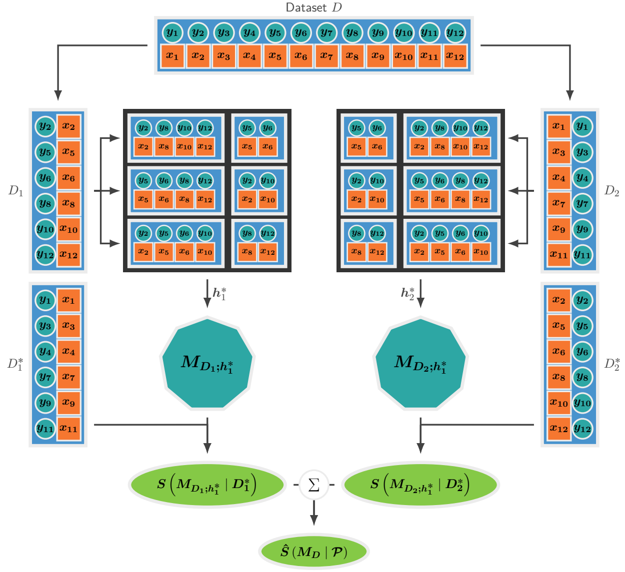
How can performance be compared?

Be fair!

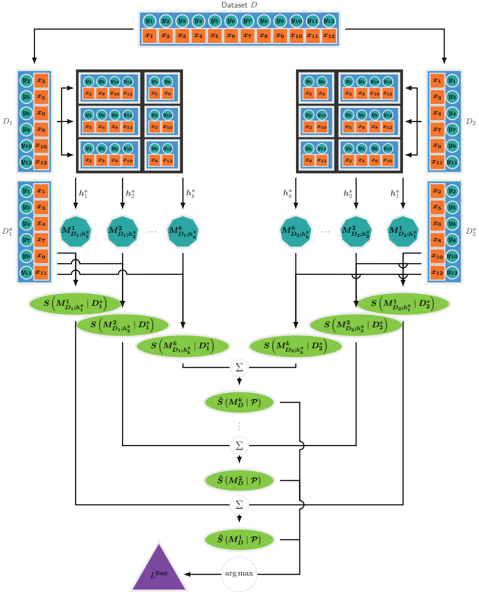
- Compare all learners and models on same data
- Tune parameters of all learners
- Don't overfit
- Don't publish over-optimistic results

Never learn, tune or evaluate on same data!

Nested Resampling



Model Selection



Nested Resampling

How to build a final model?

1. Select best learner with nested resampling
2. Find optimal hyperparameters of best learner with resampling
3. Train best learner with optimal hyperparameters on full data

Bischl et al. 2012 Evol Comput 20:249–75

Outline

Discussion

Is there a single best learner?

No!

Learner recommendations

- Typically RF $\approx$ Boosting $>$ Tree $>$ kNN
- RF robust, easy to tune and fast
- Boosting often slightly better than RF on tabular data (when properly tuned)
- Support vectot machine (SVM) good alternative for binary classification with numerical features (when properly tuned)
- Image, text and speech data $\rightarrow$ Deep Learning
- Consider ensembles, e.g. stacking $\rightarrow$ SuperLearner

Discussion

Tuning recommendations

- Never use default parameter settings
- Tune hyperparameters
- Tune hyperparameters jointly
- Parameter tuning simple and straightforward for
  - kNN
  - Decision trees
  - Boosting
  - Random forests
- Parameter tuning complex and not straightforward for
  - SVM: hyperparameters depend on kernel
  - ANN: tuning of architecture
- Use adequate resampling strategy
- Gold standard: nested cross validation

Acknowledgements

Some figures from course Introduction to Machine Learning (I2ML)

Bernd Bischl, Fabian Scheipl, Daniel Schalk, Heidi Seibold et al.
https://github.com/compstat-lmu/lecture_i2ml