Machine Learning

A Brief Introduction

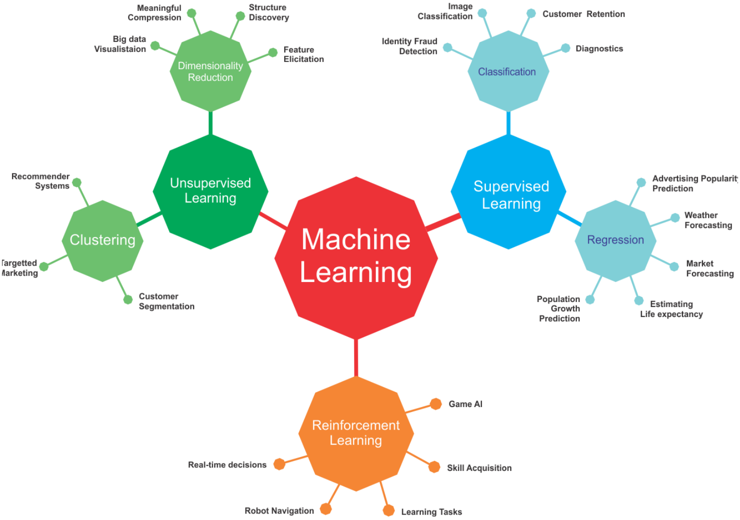
Marvin N. Wright

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

December 2022

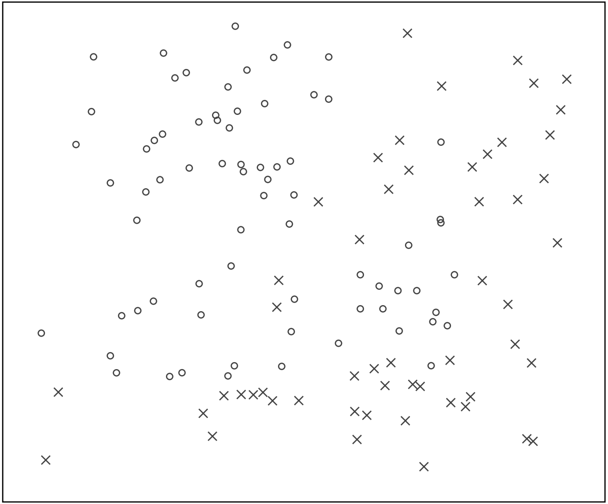
Outline

Machine Learning



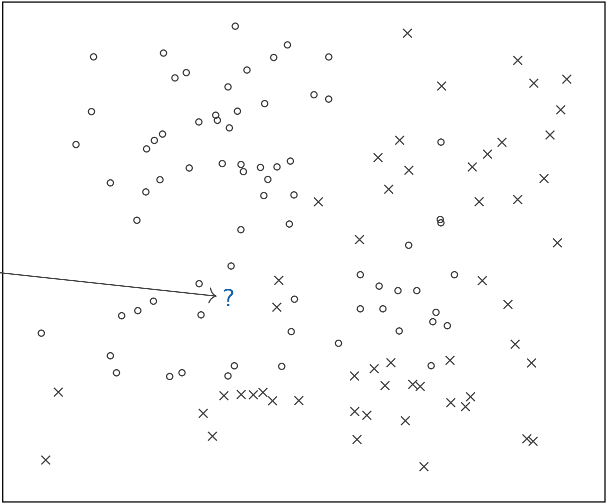
Karim 2018 ISBN 9781788997454

k-Nearest Neighbors



k-Nearest Neighbors

new
observation

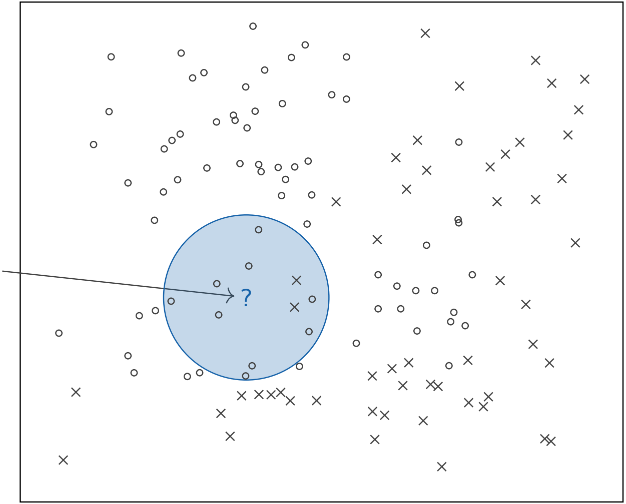


k-Nearest Neighbors

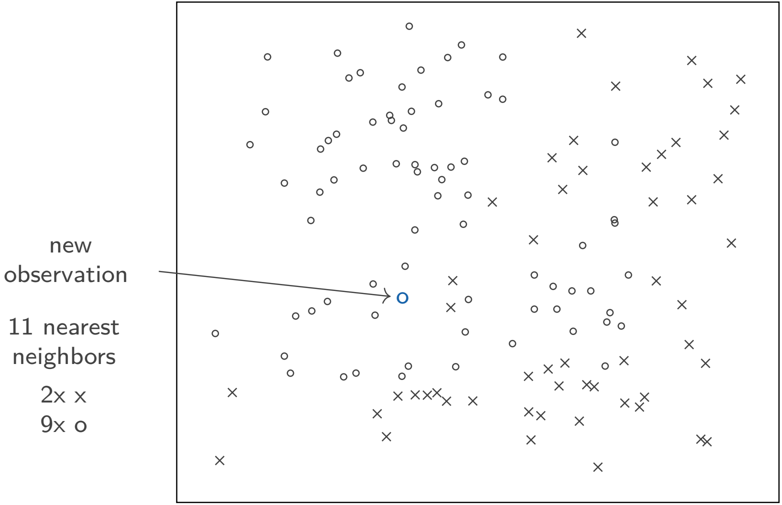
new observation

11 nearest neighbors

2x x
9x o



k-Nearest Neighbors



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

$n = 3$

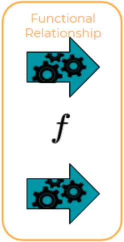
$p = 2$

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

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

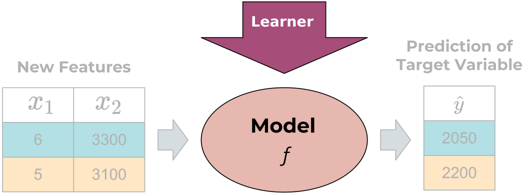
Functional relationship between **features** x and **target** y

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

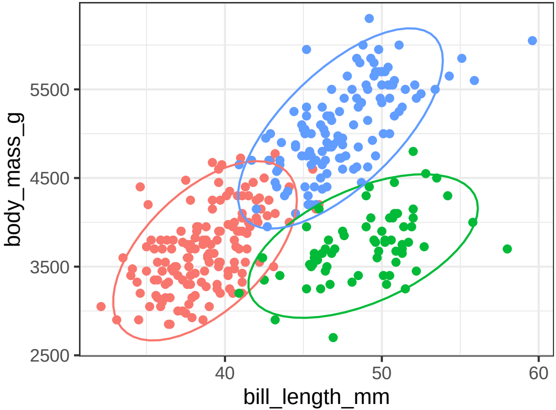
Models differ in size and complexity

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

Unsupervised Learning

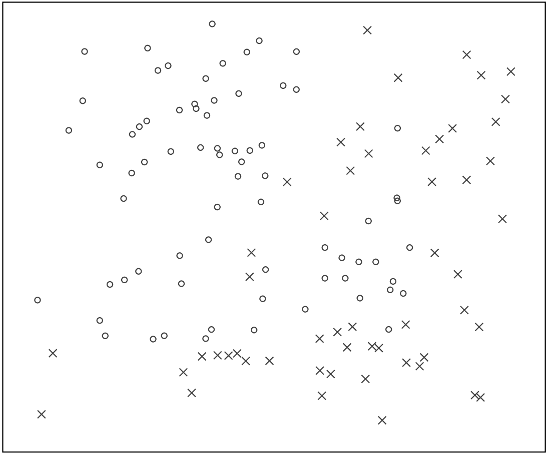
No **target** y available

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

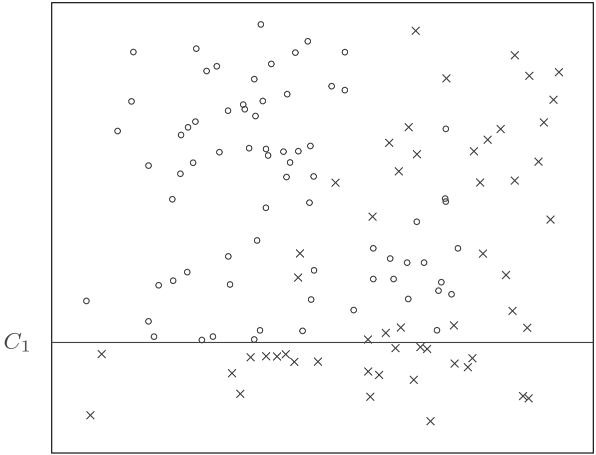


Outline

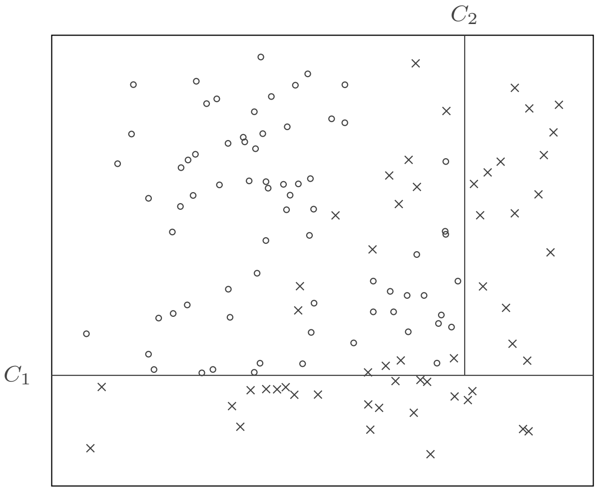
Decision Trees



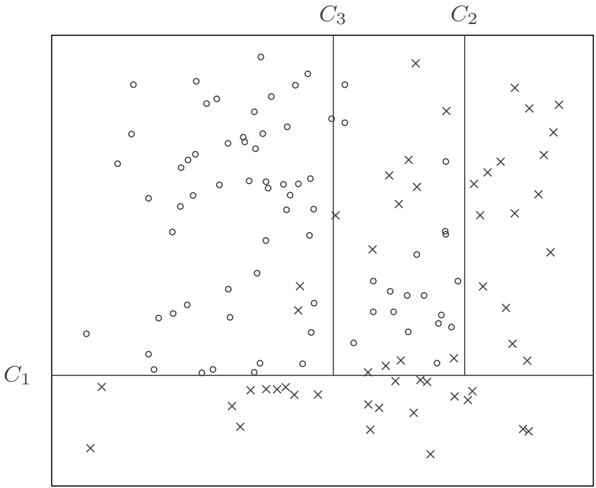
Decision Trees



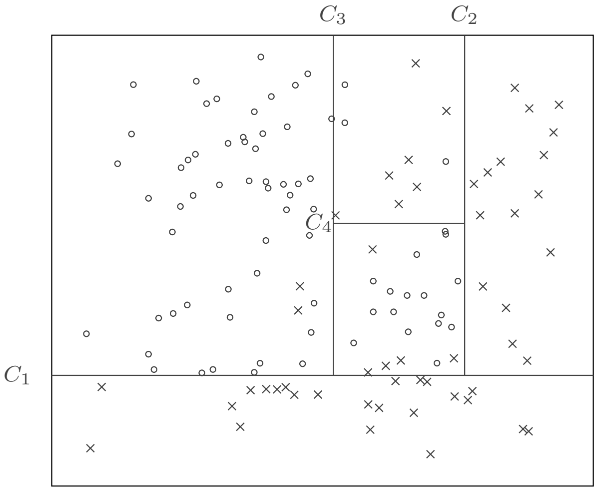
Decision Trees



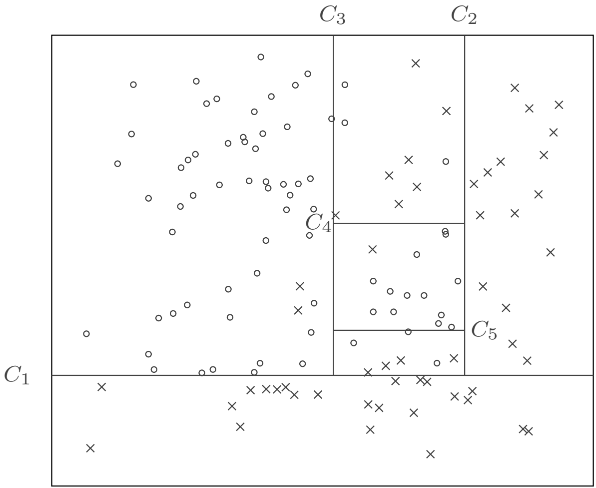
Decision Trees



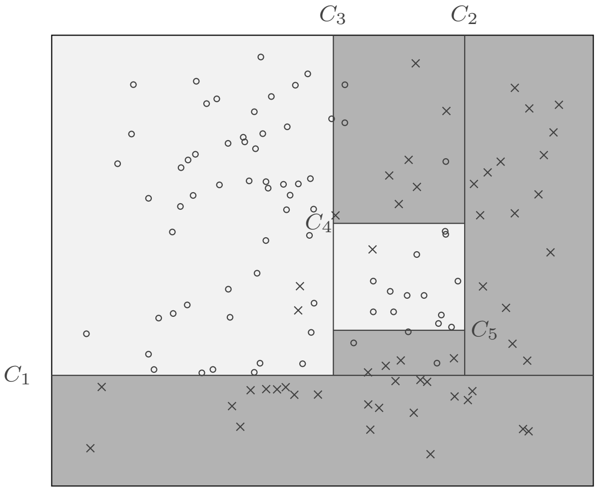
Decision Trees



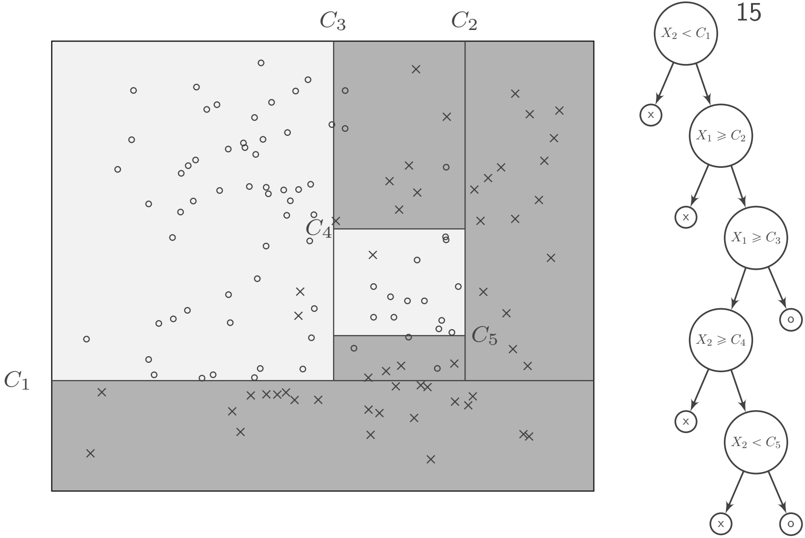
Decision Trees



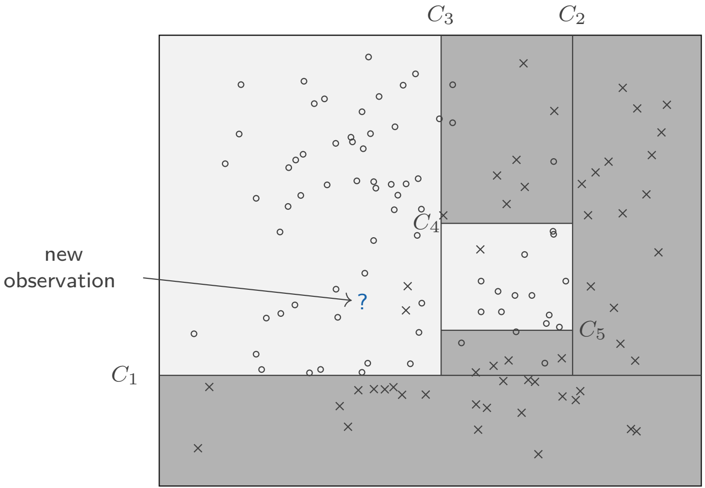
Decision Trees

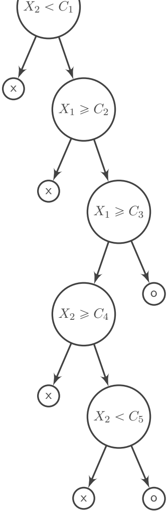


Decision Trees

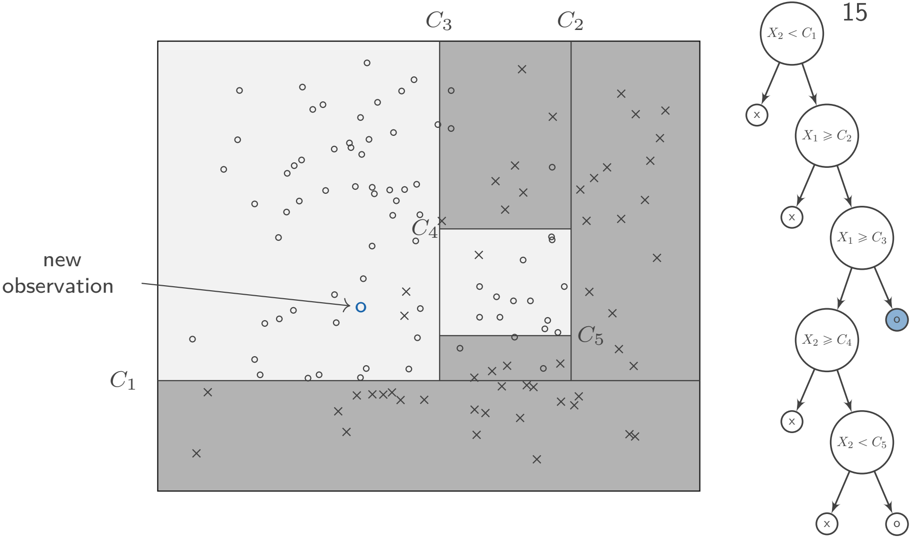


Decision Trees





Decision Trees

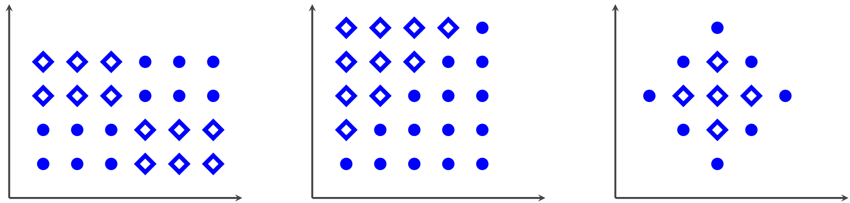


Advantages of decision trees

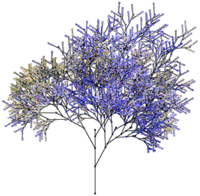
- Procedure intuitive
- Small trees simple to interpret
- Intrinsic variable selection
- Simple handling of outliers
- Fast training
- Usually better prediction performance than kNN

Disadvantages of decision trees

- Trees unstable
- Pruning can be computationally intensive
- Usually worse prediction performance than random forests (covered later) and boosted trees
- Problematic data sets

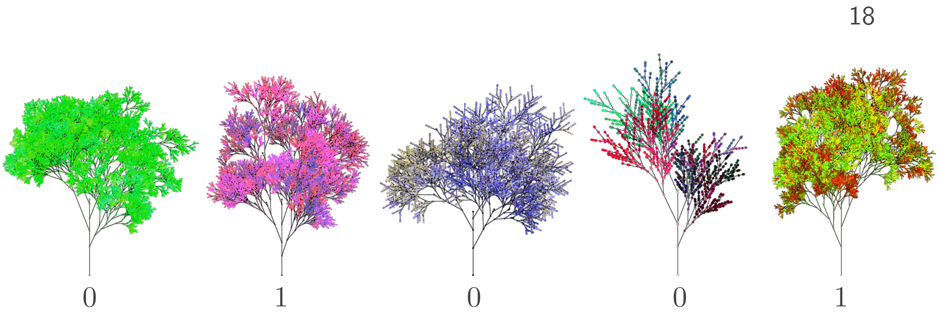


Random Forests



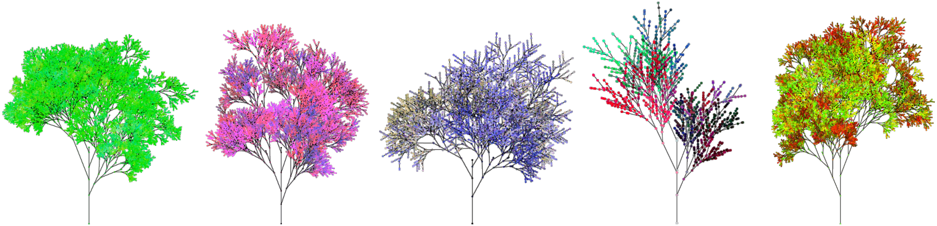
0

Random Forests



Breiman 2001 Mach Learn 45:5-32 • Malley et al. 2012 Methods Inf Med 51:74-81 • luc.devroye.org

Random Forests

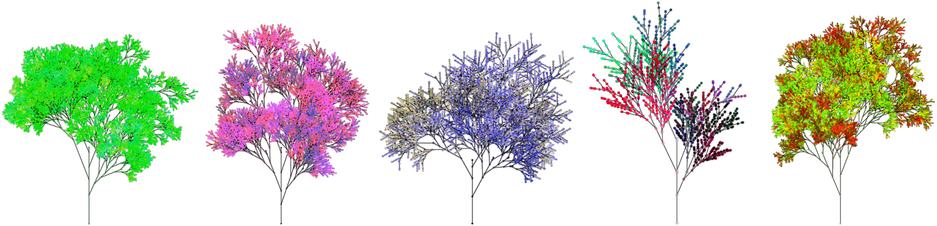


0 1 0 0 1

0

Classification: **majority vote** over all trees

Random Forests



0 1 0 0 1

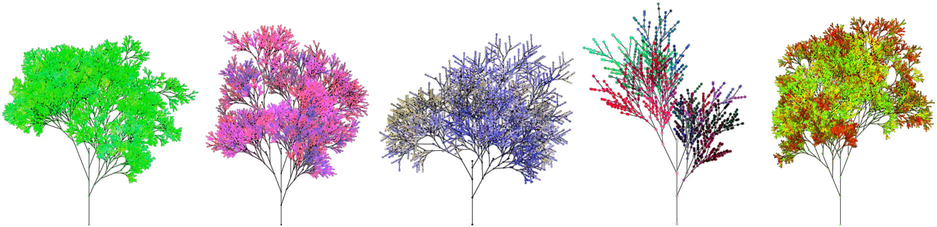
$$\underbrace{\qquad\qquad}_{0.4 < 0.5 \Rightarrow 0}$$

Classification: **majority vote** over all trees

Identical to average over all trees, cut point 0.5

Random Forests

18



8.7% 52.6% 21.3% 3.1% 69.6%

$$\underbrace{\qquad\qquad\qquad\qquad\qquad\qquad}_{31.1\%}$$

Probability estimation: Average over all trees

Two components of randomization

- Data manipulation in rows: bootstrapping / subsampling
- Data manipulation in columns: feature subsampling

Bootstrap aggregating (bagging)

- Ensemble = committee of experts
- Single weak learner = single committee member
- Ensemble decision = committee decision

Fundamental idea of bagging (bootstrap aggregating)

Any machine can be used as *base learner*, e.g. kNN or tree

Breiman 2001 Mach Learn 45:5-32

Bootstrapping

- Sampling **with** replacement
- Original sample size n, resampled sample size n
- On average $lim_{n\to\infty} \left(1 - \frac{1}{n}\right)^n \approx 0.632 \approx 2/3$ resampled

Subsampling

- Sampling **without** replacement
- Original sample size n, resampled sample size $< n$
- Standard: resampling of $0.632n$

Random Forests

Feature subsampling

At a node consider only subset of features

- Trees vary
- "Experts" differ in their opinion
- Reduce correlation between trees

Number of features considered at split

$\texttt{mtry} = \sqrt{d}$, $\ln d$ or $d/3 \rightarrow$ Tuning possible (later)

Breiman 2001 Mach Learn 45:5-32

Random forest algorithm

For each tree

1. Draw bootstrap sample with replacement
2. Grow tree
   a) Use random subset of variables (`mtry`) at each node
   b) Stop if minimum node size reached
3. Determine proportion of '1' in each terminal node

New subject

1. Drop down subject in each single tree
2. Store proportion from all trees
3. Average proportion of '1's over all trees

Kruppa et al. 2013 Expert Sys Appl 40:5125-31

Random Forests

Advantages of random forests

- As with trees: Procedure intuitive, intrinsic variable selection, simple handling of outliers, fast training
- Work well with high dimensional data
- Work well without (or with only a little) tuning
- Usually better prediction performance than a single tree

Disadvantages of random forests

- Not simple to interpret
- Sometimes worse prediction performance than well tuned boosted trees
- Bad prediction performance on image, text and speech data

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

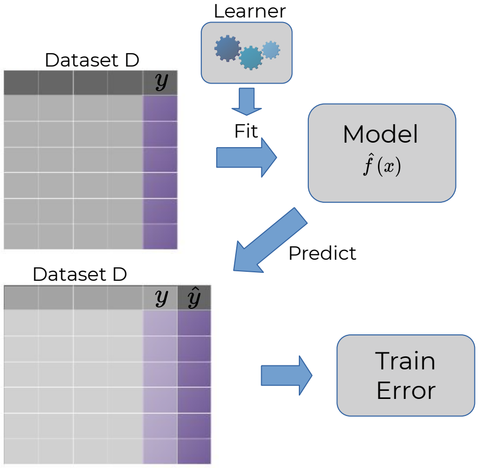
Survival outcome

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

Model Evaluation

Training error

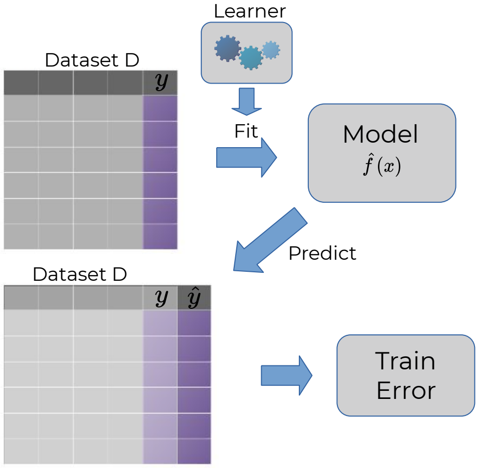
Evaluate performance on training data



Model Evaluation

Training error

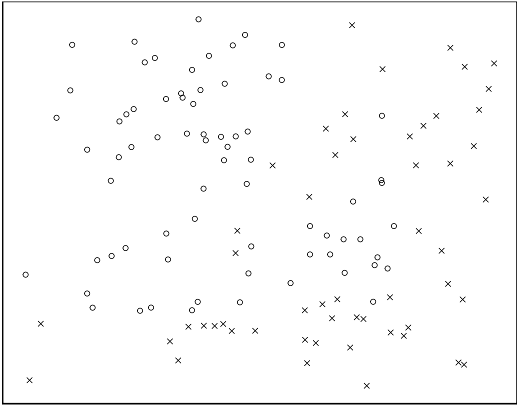
Evaluate performance on training data



Problem: Overfitting

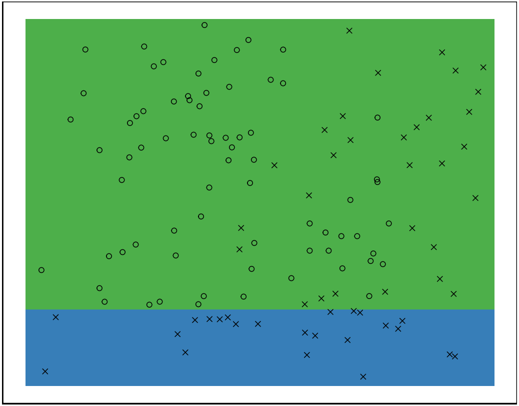
Model Evaluation

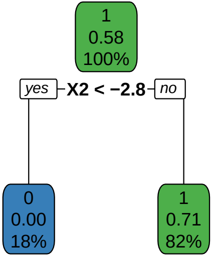
Overfitting



Model Evaluation

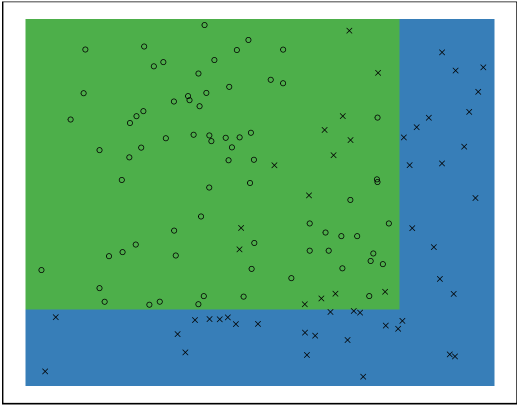
Overfitting

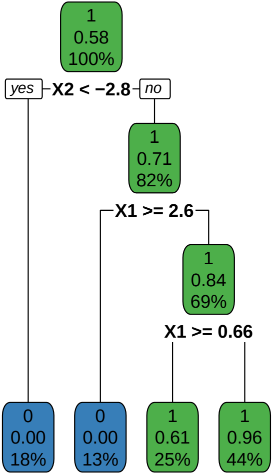




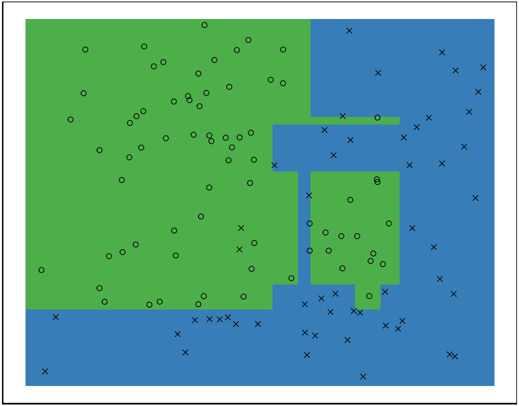
Model Evaluation

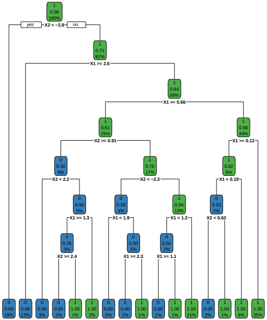
Overfitting





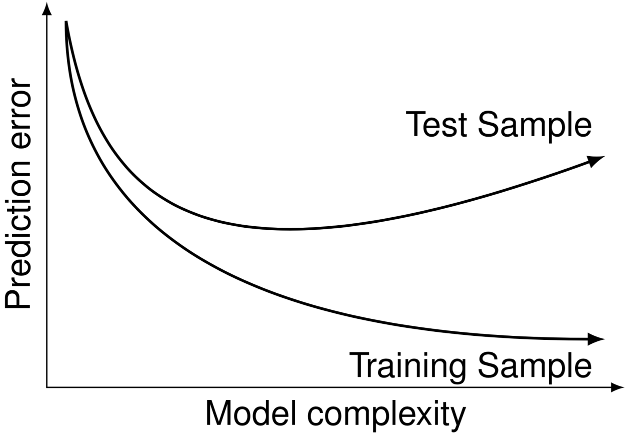
Overfitting





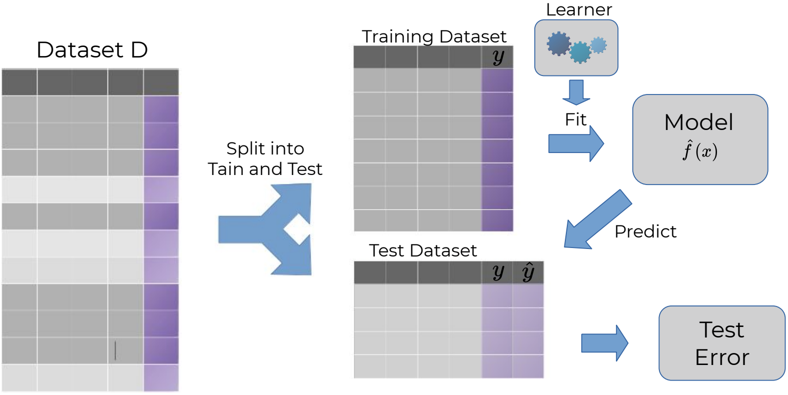
Model Evaluation

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

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.

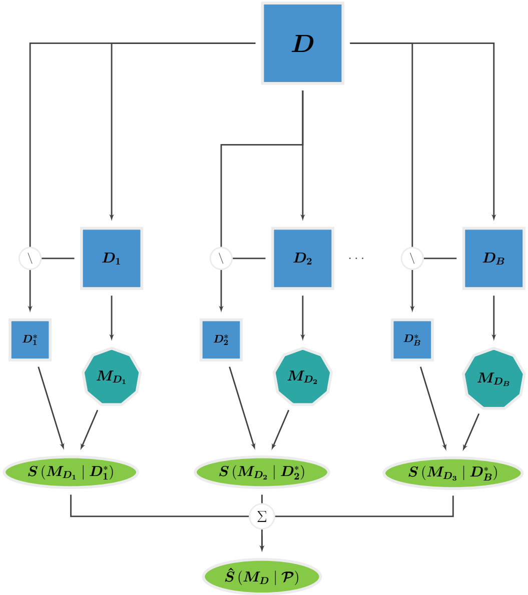
Hyperparameter tuning

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

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

$\rightarrow$ Nested resampling

Resampling



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

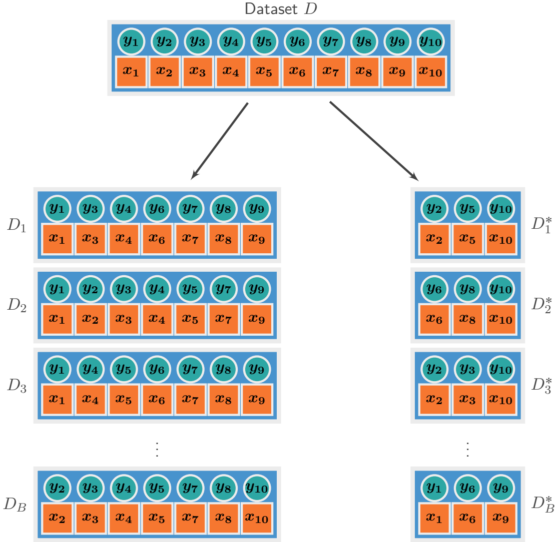
Resampling

- 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

Simon 2007 ISBN 9780387475080 • Bischl et al. 2012 Evol Comput 20:249–75

Resampling

Subsampling



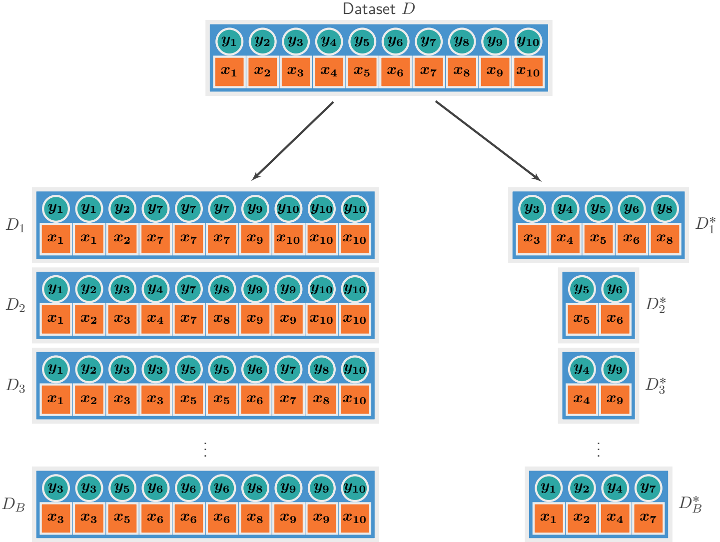
Subsampling

- 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

Resampling

Bootstrapping

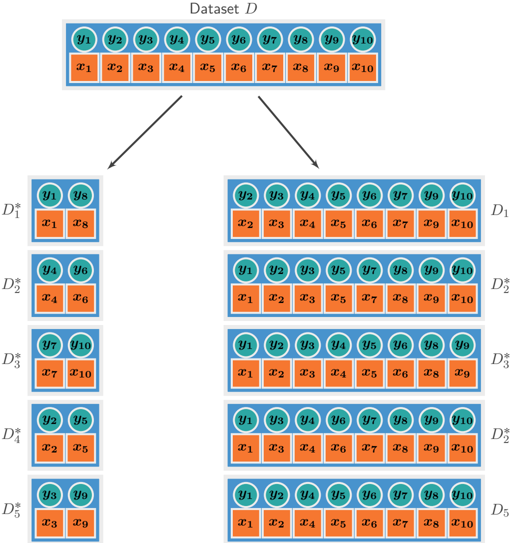


Bootstrapping

- 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)



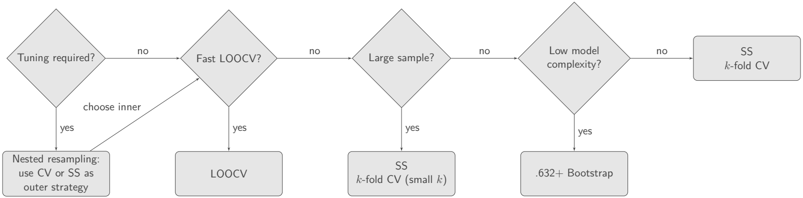
Cross validation (CV)

- Split D in B test datasets D_b^*
- Use $D_b = D \setminus D_b^*$ 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 → 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

Resampling

How to choose the resampling method?



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

Outline

Penalized Regression

Generalized linear model

$$
\begin{aligned}
g(\mathbb{E}(Y)) &= \beta_0 + \beta_1 \cdot X_1 + \ldots + \beta_p \cdot X_p \\
&= X\beta
\end{aligned}
$$

g: Link function

Penalized Regression

Generalized linear model

$$\begin{aligned} g(\mathbb{E}(Y)) &= \beta_0 + \beta_1 \cdot X_1 + \ldots + \beta_p \cdot X_p \\ &= X\beta \end{aligned}$$

g: Link function

Linear model

$$\mathbb{E}(Y) = X\beta$$

Penalized Regression

Ordinary least squares

Minimize squared differences

$$
\begin{aligned}
L_{\mathsf{OLS}} &= \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \\
&= \|y - X\beta\|_2^2 \\
&= (y - X\beta)'(y - X\beta)
\end{aligned}
$$

Penalized Regression

Ordinary least squares

Minimize squared differences

$$
\begin{aligned}
L_{\mathsf{OLS}} &= \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \\
&= \|y - X\beta\|_2^2 \\
&= (y - X\beta)'(y - X\beta)
\end{aligned}
$$

Solution:

$$
\beta_{\mathsf{OLS}} = \left(X'X\right)^{-1} X'y
$$

Penalized Regression

Ridge regression

Penalize large parameter estimates (L2 regularization)

$$
\begin{aligned}
L_{\text{Ridge}} &= \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{m} \beta_j^2 \\
&= \|y - X\beta\|_2^2 + \lambda\|\beta\|_2^2
\end{aligned}
$$

Penalized Regression

Ridge regression

Penalize large parameter estimates (L2 regularization)

$$
\begin{aligned}
L_{\text{Ridge}} &= \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{m} \beta_j^2 \\
&= \|y - X\beta\|_2^2 + \lambda \|\beta\|_2^2
\end{aligned}
$$

Solution:

$$
\beta_{\text{Ridge}} = \left(X'X + \lambda I\right)^{-1} X'y
$$

Penalized Regression

Ridge regression

Penalize large parameter estimates (L2 regularization)

$$\begin{aligned} L_{\text{Ridge}} &= \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{m} \beta_j^2 \\ &= \|y - X\beta\|_2^2 + \lambda\|\beta\|_2^2 \end{aligned}$$

Solution:

$$\beta_{\text{Ridge}} = \left(X'X + \lambda I\right)^{-1} X'y$$

Shrink parameter estimates towards zero

Penalized Regression

How to find best λ?

Minimize L_{Ridge} in cross validation

$\rightarrow$ Hyperparameter tuning

LASSO: Least absolute shrinkage and selection operator

Penalize large parameter estimates (L1 regularization)

$$
\begin{aligned}
L_{\text{LASSO}} &= \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{m} |\beta_j| \\
&= \|y - X\beta\|_2^2 + \lambda \|\beta\|_1
\end{aligned}
$$

LASSO: Least absolute shrinkage and selection operator

Penalize large parameter estimates (L1 regularization)

$$
\begin{aligned}
L_{\mathsf{LASSO}} &= \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{m} |\beta_j| \\
&= \|y - X\beta\|_2^2 + \lambda \|\beta\|_1
\end{aligned}
$$

No closed-form solution

Penalized Regression

LASSO: Least absolute shrinkage and selection operator

Penalize large parameter estimates (L1 regularization)

$$\begin{aligned} L_{\mathsf{LASSO}} &= \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{m} |\beta_j| \\ &= \|y - X\beta\|_2^2 + \lambda \|\beta\|_1 \end{aligned}$$

No closed-form solution

Shrink parameter estimates to (exactly) zero

Penalized Regression

Elastic net: Combination of Ridge and LASSO

L1 and L2 regularization

$$
\begin{aligned}
L_{\mathsf{Elnet}} &= \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda_1 \sum_{j=1}^{m}|\beta_j| + \lambda_2 \sum_{j=1}^{m}\beta_j^2 \\
&= \|y - X\beta\|_2^2 + \lambda_1\|\beta\|_1 + \lambda_2\|\beta\|_2^2
\end{aligned}
$$

Advantages of penalized regression

- Reduces overfitting
- Avoid multicolinarity issues of (non-penalized) regression models
  $\rightarrow$ Work well with high-dimensional data
- Same general concept of (non-penalized) regression models
  $\rightarrow$ Interpretable model
- Better prediction performance than non-penalized regression (less variance)
- Implicit variable selection (LASSO)

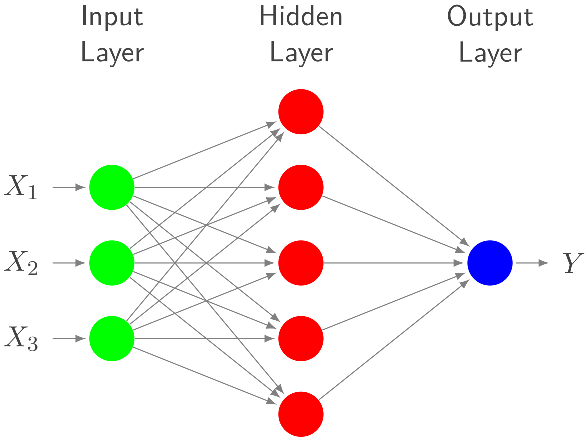
Penalized Regression

Disadvantages of penalized regression

- Biased parameter estimates
- Cannot use statistical inference methods used in non-penalized regression
- Interactions and non-linear effects have to be explicitly specified
- Often worse prediction performance than (other) machine learning algorithms

Outline

Artificial Neural Networks



Artificial Neural Networks

For each hidden unit:

Activation function

Unit input

$$z = \sigma(\mathbf{w} \cdot \mathbf{x} + b)$$

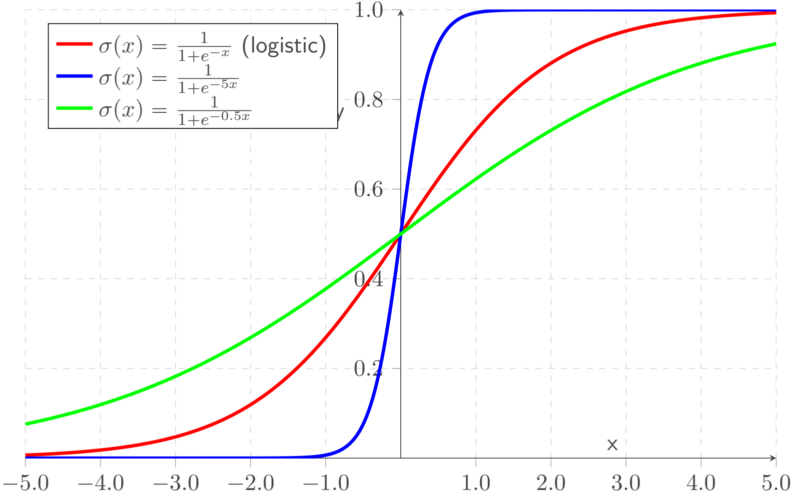
Unit output

Bias

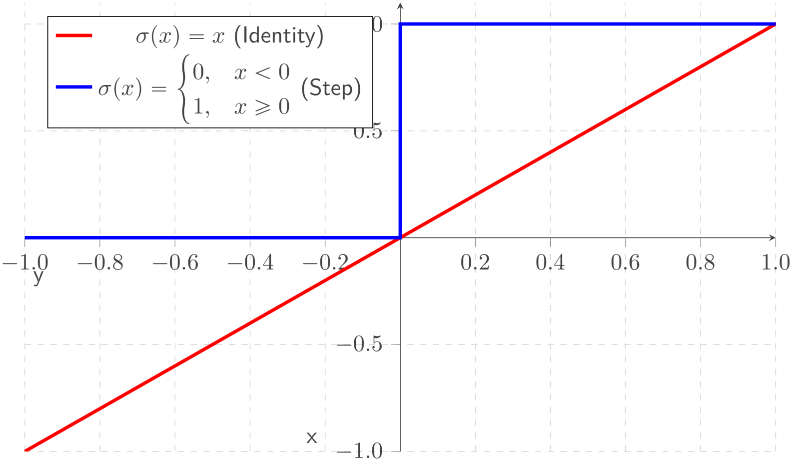
Weights

Artificial Neural Networks

Activation function: Sigmoid



Special cases



Identity: Linear model

Step: Perceptron

Artificial Neural Networks

How to fit a neural network?

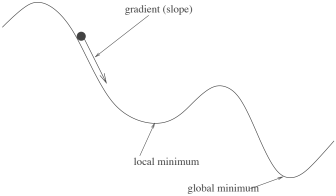
Loss function: Error as function of network weights, e.g,

$$L(\mathcal{W}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Aim: Find weights that minimize error

Gradient descent

Adjust weights in direction with steepest descent

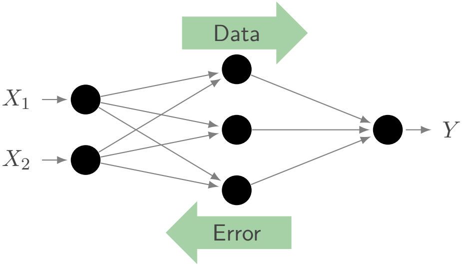


http://bengio.abracadoudou.com/lectures/old/tex_ann.pdf

How to fit a neural network?

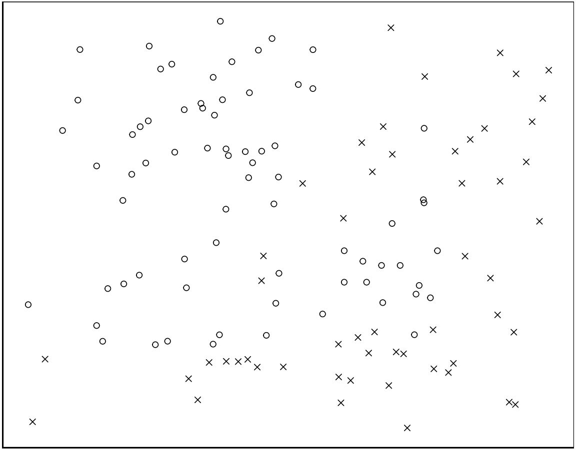
Backpropagation

1. Initialize weights randomly
2. For k iterations repeat
   a) Compute error function
   b) Adjust weights in output layer
   c) Propagate error backwards through network and adjust weights



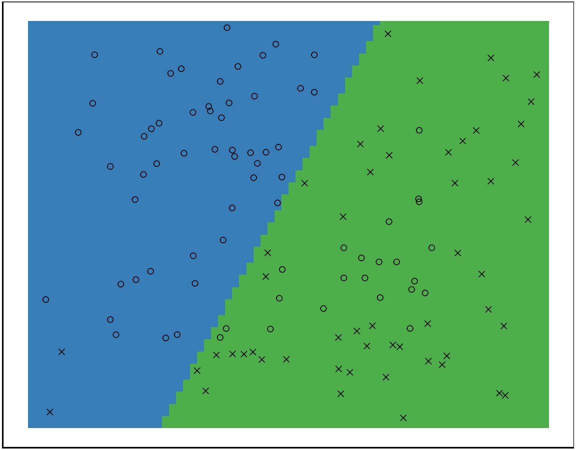
Artificial Neural Networks

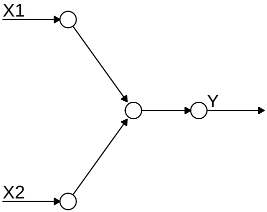
Do neural networks overfit?



Artificial Neural Networks

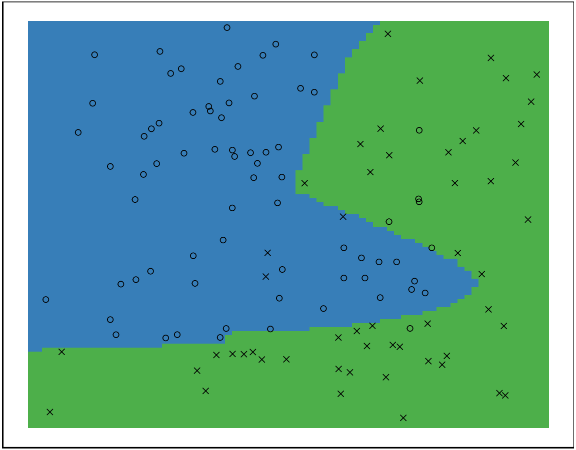
Do neural networks overfit?

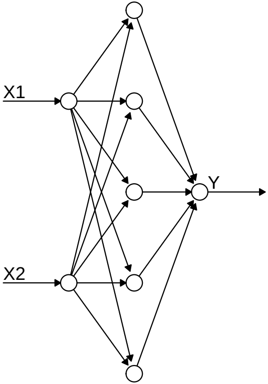




Artificial Neural Networks

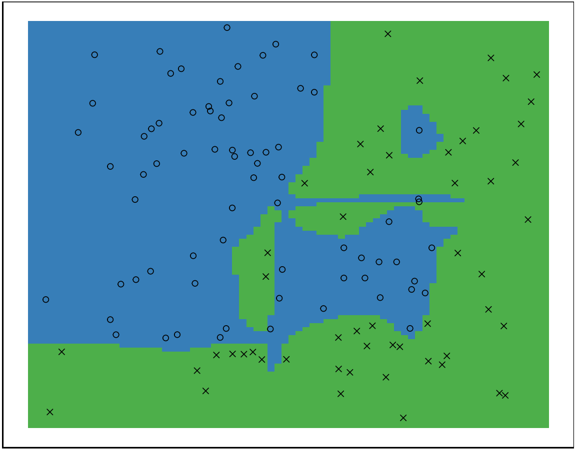
Do neural networks overfit?

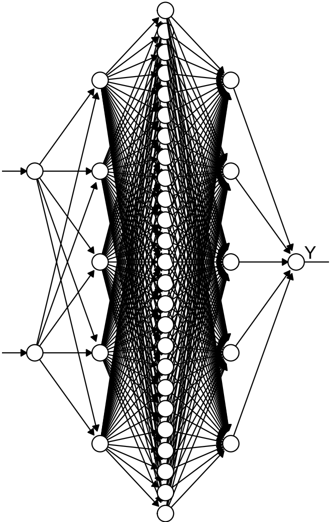




Artificial Neural Networks

Do neural networks overfit?





Artificial Neural Networks

L2 regularization

$$L(\mathcal{W}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda \underbrace{\sum_{\mathbf{w}\in\mathcal{W}} \mathbf{w}^2}_{\text{Penalty term}}$$

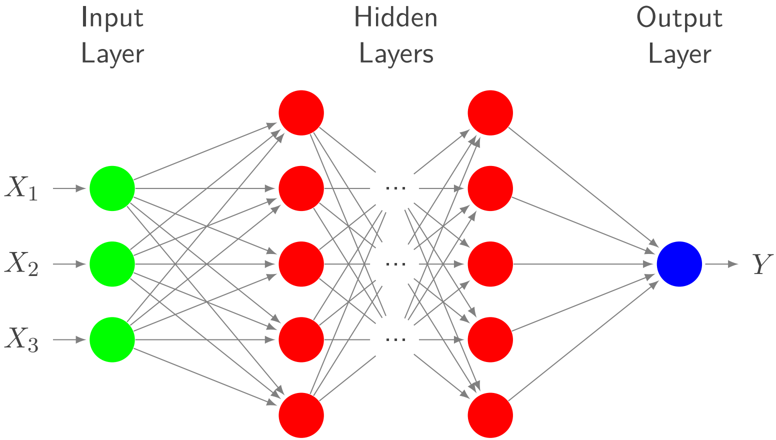
Alternative: Dropout

Temporarily remove units while fitting

Other alternative: Early stopping

Artificial Neural Networks

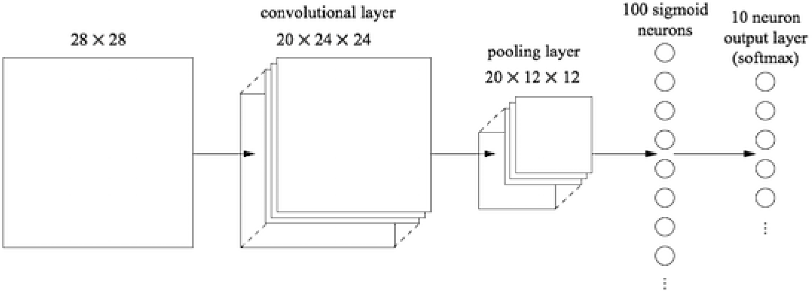
What is (not) deep learning?



What is deep learning?

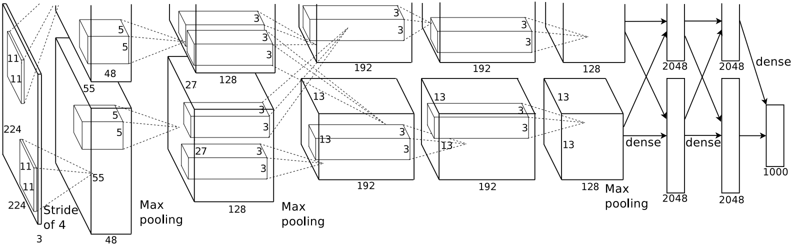
Idea: Build hierarchy of concepts (representations)

Convolutional neural networks



http://neuralnetworksanddeeplearning.com/chap6.html

Example



Krizhevsky et al. 2012 NIPS 25:1097

Advantages of neural networks

- Can fit any complicated function almost perfectly
- Learns representations
- Online learning possible
- Prediction very fast (matrix multiplication)
- Can be parallelized (GPU computing)

Artificial Neural Networks

Disadvantages of neural networks

- Prone to overfitting
- Difficult to design good networks
- Interpretation very difficult (black box)
- Learning can be slow
- Statistical properties not well studied

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)
- Architecture of neural network

Most learners have several hyperparameters

Have to be jointly optimized

Hyperparameter Tuning

Search entire parameter space

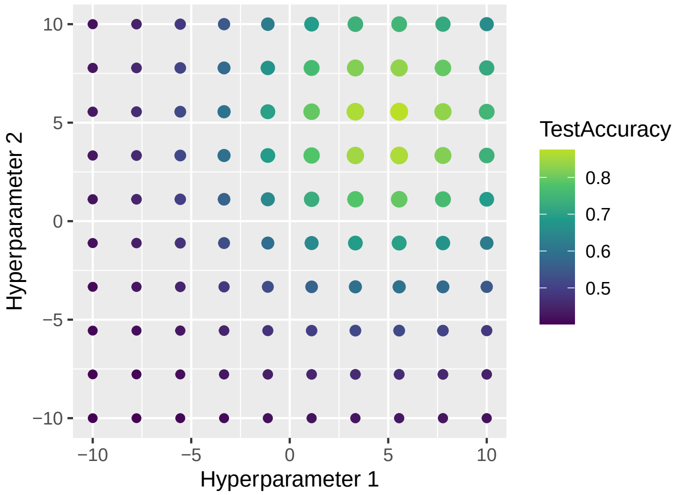
- All possible combinations
- Grid search
- Randomly select combinations
- Model-based optimization

Use resampling

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

Hyperparameter Tuning

Grid search



Hyperparameter Tuning

Grid search

Advantages

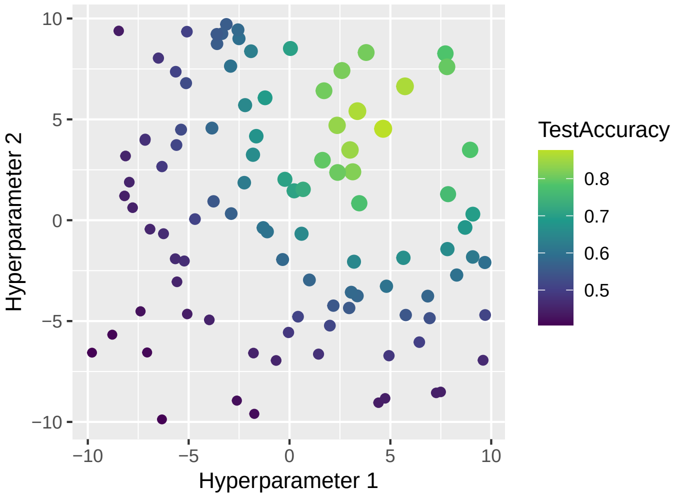
- Easy to implement
- All parameter types possible
- Easily parallelized

Disadvantages

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

Hyperparameter Tuning

Random search



Hyperparameter Tuning

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

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

Benchmarking

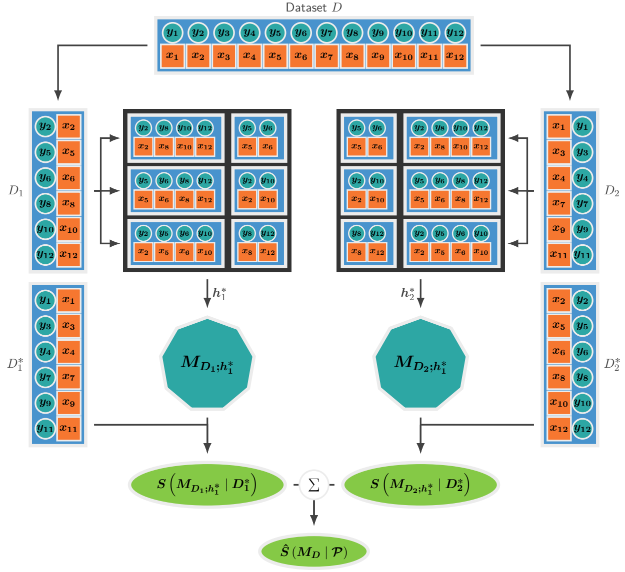
How can performance be compared?

Be fair!

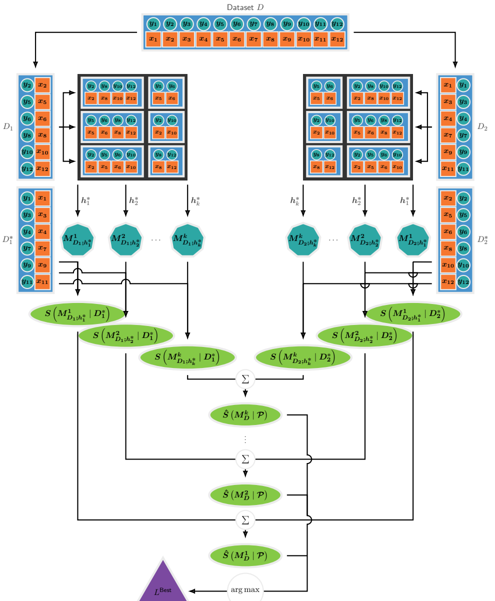
- Compare all learners and models on same data
- Tune parameters of all learners
- Don't overfit
- Don't publish over-optimistic results

Never learn, tune or evaluate on same data!

Nested Resampling



Model Selection



Benchmarking

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

Outline

Discussion

Is there a single best learner?

No!

General 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)
- SVM good alternative for binary classification with numerical features (when properly tuned)
- Image, text and speech data $\rightarrow$ Deep Learning

Discussion

Important aspects when applying machine learning

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