OEAW AI SUMMER SCHOOL

Introduction to Supervised Machine Learning

Johannes Brandstetter, Michael Widrich
Institute for Machine Learning

JOHANNES KEPLER
UNIVERSITY LINZ

Institute for
Machine Learning

Supervised Machine Learning

How to solve these tasks?

- Prediction of trajectory of a space shuttle
- Translation of one language into another
- Prediction of protein function

Explicit models

Traditional approach: Explicit model

- Use explicit knowledge to design model deductively

Explicit models

Traditional approach: Explicit model

- Use explicit knowledge to design model deductively
- Pros:
  - Knowledge about behavior of model and environment/problem
  - Knowledge about restrictions of model and reasons for design choices

Explicit models

Traditional approach: Explicit model

- Use explicit knowledge to design model deductively
- Pros:
  - Knowledge about behavior of model and environment/problem
  - Knowledge about restrictions of model and reasons for design choices
- Cons:
  - Sometimes problem is too complex to model
  - Consequences of simplifications of problem/model hard to asses
  - Insufficient knowledge about problem/environment

Machine Learning

Machine Learning: Inductive Learning

- Use previously observed data to create model inductively

Machine Learning

Machine Learning: Inductive Learning

- Use previously observed data to create model inductively
- Pros:
  - Problem can be solved without (exhaustive) knowledge about problem
  - Predictions/Insights are created directly from data
  - Can handle complex problems and profits from big data

Machine Learning

Machine Learning: Inductive Learning

- Use previously observed data to create model inductively
- Pros:
  - Problem can be solved without (exhaustive) knowledge about problem
  - Predictions/Insights are created directly from data
  - Can handle complex problems and profits from big data
- Cons:
  - Data is required (sometimes a lot of data!)
  - Complex models (deep learning) can end up being a black box
  - Naive application might lead to biases

Supervised Machine Learning

Supervised Machine Learning:

- Learning from input values and corresponding target values
  - E.g. image + object type, DNA sequence + phenotype, ...

Supervised Machine Learning

Supervised Machine Learning:

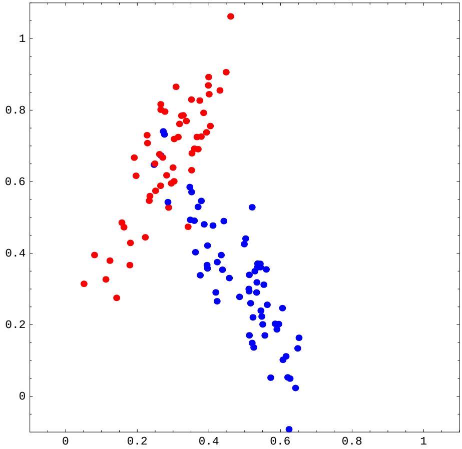
- Learning from input values and corresponding target values
  - E.g. image + object type, DNA sequence + phenotype, ...
- Typical usage: predictive modeling
  - Train model on dataset with input+target values
  - Use trained model to predict target values for other (new) inputs

Supervised Machine Learning

Supervised Machine Learning:

- Learning from input values and corresponding target values
  - E.g. image + object type, DNA sequence + phenotype, ...
- Typical usage: predictive modeling
  - Train model on dataset with input+target values
  - Use trained model to predict target values for other (new) inputs
- Classification: target value is class label
- Regression: target value is numerical value

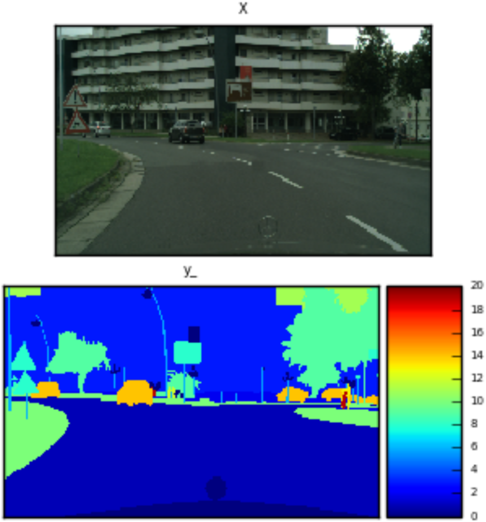
Example data for Supervised ML (1)



Example data for Supervised ML (2)

0.99516	0.890813	0.933726	0.793397	0.826405	0.236946	-1
0.853206	0.611647	0.317486	0.633609	0.411492	0.985231	+1
0.387494	0.459847	0.815049	0.394526	0.678227	0.031886	-1
0.733515	0.640438	1.19068	0.639685	0.0793674	0.160503	+1
0.274817	0.261054	1.20056	0.689895	0.401913	0.277955	-1
0.329943	0.241299	0.848705	0.721673	0.973852	0.795238	-1
0.334784	0.350487	0.315131	0.928277	0.816343	0.558292	-1
0.481578	0.738839	0.0925513	0.294667	0.612725	0.573062	-1
0.0940846	0.278992	0.451819	0.900141	0.220497	0.541176	+1
0.360569	0.638554	1.0307	0.260456	0.00658296	0.380672	+1
0.0857518	0.3775	0.386551	0.570562	0.15437	0.102717	+1
0.755808	0.1362	0.544536	0.848888	0.874862	0.307479	-1
0.421025	0.785714	0.449038	0.920612	0.420418	0.749187	-1
0.939446	0.0468747	0.15846	0.625944	0.198894	0.176125	+1
0.845362	0.767883	0.824993	0.725803	0.808218	0.63495	-1
0.484793	0.129329	0.0783719	0.465347	0.291457	0.254278	+1
0.399041	0.751829	0.763511	0.894785	0.47902	0.15156	-1
0.643232	0.615629	0.430261	0.0458972	0.446513	0.844081	+1
...	...	...	...	...	...	...

Example data for Supervised ML (3)



Terminology

Model: parameterized function/method with specific parameter values (e.g. a trained neural network)

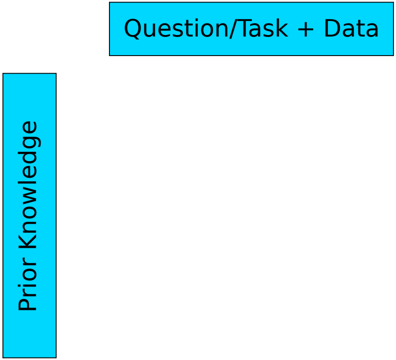
Model class: the class of models in which we search for the model (e.g. neural networks, SVMs, ...)

Parameters: representations of concrete models inside the given model class (e.g. network weights)

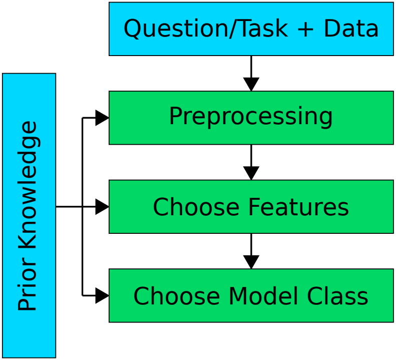
Hyperparameters: parameters controlling model complexity or the training procedure (e.g. network learning rate)

Model selection/training: process of finding a model from the model class

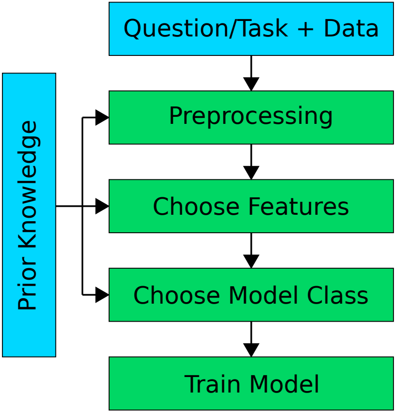
Basic data analysis workflow



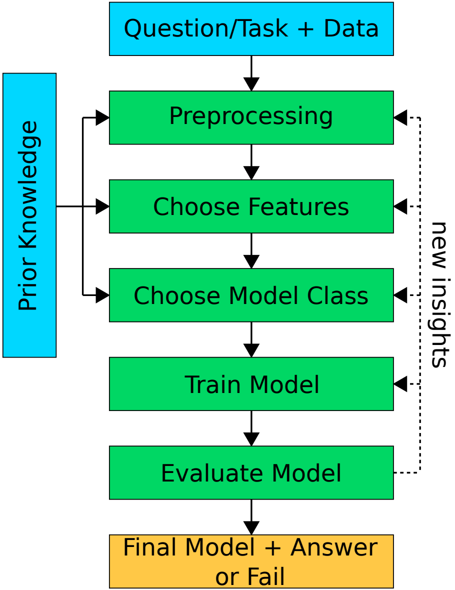
Basic data analysis workflow



Basic data analysis workflow



Basic data analysis workflow



Introductory example: Fish recognition

- Example borrowed from

  R. O. Duda, P. E. Hart, and D. G. Stork. Pattern Classification*. 2nd edition. John Wiley & Sons, 2001. ISBN 0-471-05669-3.*

- Automated system to sort fish in a fish-packing company: salmons must be distinguished from sea bass optically
- **Given:** a set of pictures with fish labels
- **Goal:** distinguish between salmons and sea bass

→ Classification task with two labels (salmon vs. sea bass)

Our data (two sample images)

Salmon:

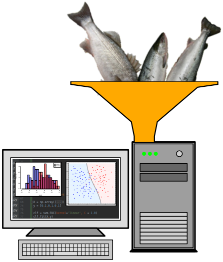
Sea bass:

Our data (two sample images)

Salmon:

Sea bass:

How can we distinguish these two kinds of fish?



Our data (two sample images)

Salmon:

Sea bass:

How can we distinguish these two kinds of fish?

First step: Let's take a look at our data!

Preprocessing & Feature Selection

Feature selection:

- What data do we have?
- Removal of redundant features
- Removal of features the model class cannot utilize
- (Deep Learning: Feature selection mainly by neural network)

Preprocessing & Feature Selection

Feature selection:

- What data do we have?
- Removal of redundant features
- Removal of features the model class cannot utilize
- (Deep Learning: Feature selection mainly by neural network)

Preprocessing:

- Contrast and brightness correction
- Segmentation
- Alignment
- Normalization
- ...

Back to our data

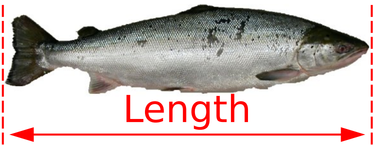
Salmon:

Sea bass:

…assume we use length and brightness as features

Back to our data

Salmon:



Sea bass:

...assume we use length and brightness as features

$\rightarrow$ How do we express/represent these features?

Input representation

- We can represent our objects by vectors of feature values (=feature vectors) of length d

$$\mathbf{x} = (x^1, \ldots, x^d)^T$$

  - E.g.: fish is represented as feature vector with two values length and brightness (i.e. $d = 2$)

Input representation

- We can represent our objects by vectors of feature values (=feature vectors) of length d

$$\mathbf{x} = (x^1, \ldots, x^d)^T$$

  - E.g.: fish is represented as feature vector with two values length and brightness (i.e. $d = 2$)
- An object described by feature vector is also referred to as sample

Input representation

- We can represent our objects by vectors of feature values (=feature vectors) of length d
$$\mathbf{x} = (x^1, \ldots, x^d)^T$$
  - E.g.: fish is represented as feature vector with two values length and brightness (i.e. $d = 2$)
- An object described by feature vector is also referred to as sample
- We assume our feature vectors to be from a set/space X
$$\mathbf{x} = (x^1, \ldots, x^d)^T \in X$$
- If X is finite set of labels, we speak of *categorical variables/features*
- If $X = \mathbb{R}$, real interval, etc., we speak of *numerical variables/features*

Input and output representation

- Assume our dataset consists of l objects with feature vectors $\mathbf{x}_1, \ldots, \mathbf{x}_l$

Input and output representation

- Assume our dataset consists of l objects with feature vectors $\mathbf{x}_1, \ldots, \mathbf{x}_l$
- Assume we are given a target value $y_i \in \mathbb{R}$ for each sample $\mathbf{x}_i$

Input and output representation

- Assume our dataset consists of l objects with feature vectors $\mathbf{x}_1, \dots, \mathbf{x}_l$
- Assume we are given a target value $y_i \in \mathbb{R}$ for each sample $\mathbf{x}_i$
- Then all target values constitute the target/label vector:
$$\mathbf{y} = (y_1, \dots, y_l)^T$$

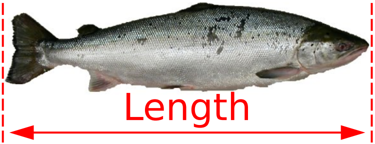
Input and output representation

- Assume our dataset consists of l objects with feature vectors $\mathbf{x}_1, \ldots, \mathbf{x}_l$
- Assume we are given a target value $y_i \in \mathbb{R}$ for each sample $\mathbf{x}_i$
- Then all target values constitute the target/label vector:
$$\mathbf{y} = (y_1, \ldots, y_l)^T$$
- Often we write our dataset, including input features and targets, as data matrix $\mathbf{Z}$:

$$\mathbf{Z} = \begin{pmatrix} \mathbf{X} \\ \mathbf{y} \end{pmatrix} = \begin{pmatrix} x_1^1 & \ldots & x_l^1 \\ \vdots & \ddots & \vdots \\ x_1^d & \ldots & x_l^d \\ y_1 & \ldots & y_l \end{pmatrix}$$

Back to our data

Salmon:

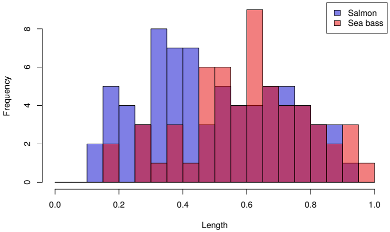


Sea bass:

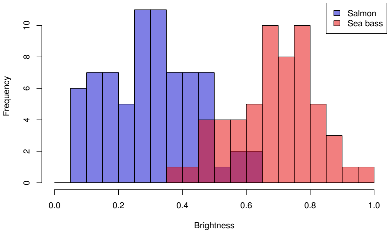
- We now know how to represent our data (=fish features and labels) and will take a look at it via histograms

Back to our data

Length:

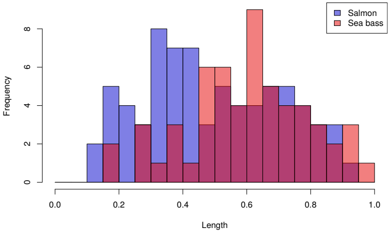


Brightness:

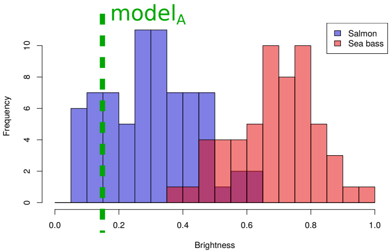


Back to our data

Length:



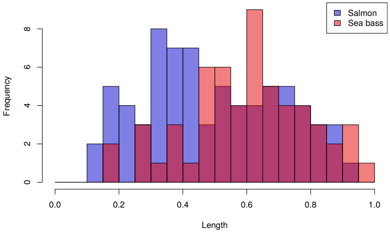
Brightness:



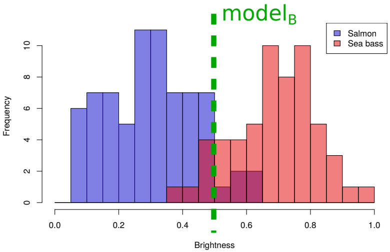
- Assume we want to use a simple threshold as model class to classify our data
  - = model with 1 parameter (threshold value)
  - we have to decide on a single feature (e.g. brightness)
  - we have to choose the model parameter(s)

Back to our data

Length:



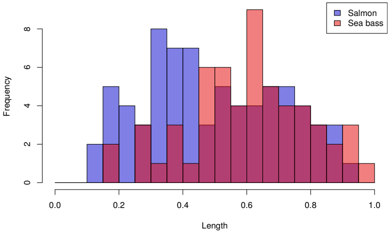
Brightness:



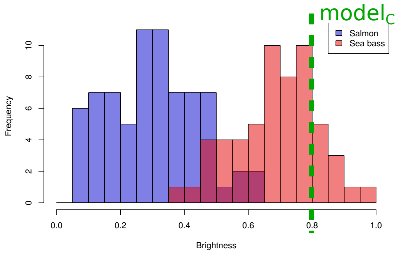
- Assume we want to use a simple threshold as model class to classify our data
  - = model with 1 parameter (threshold value)
  - we have to decide on a single feature (e.g. brightness)
  - we have to choose the model parameter(s)

Back to our data

Length:



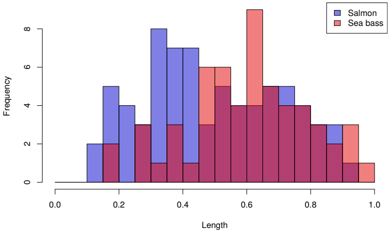
Brightness:



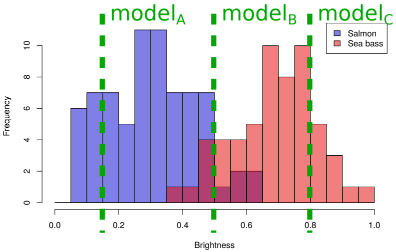
- Assume we want to use a simple threshold as model class to classify our data
  - = model with 1 parameter (threshold value)
  - we have to decide on a single feature (e.g. brightness)
  - we have to choose the model parameter(s)

Back to our data

Length:



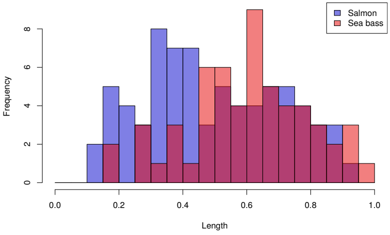
Brightness:



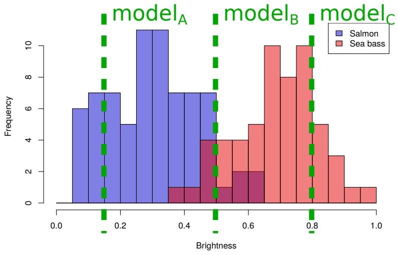
- How do we get the "best" model?

Back to our data

Length:



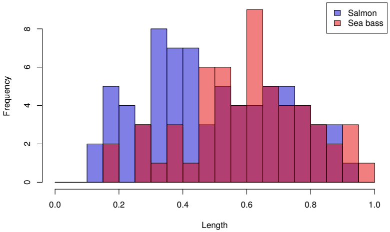
Brightness:



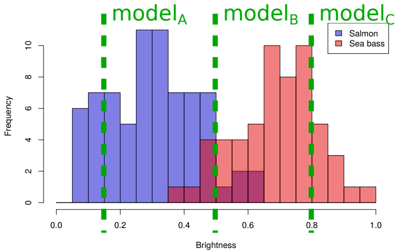
- How do we get the "best" model?
  1. How does our model perform on our data?

Back to our data

Length:



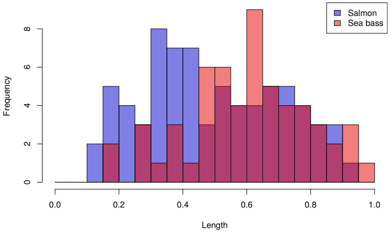
Brightness:



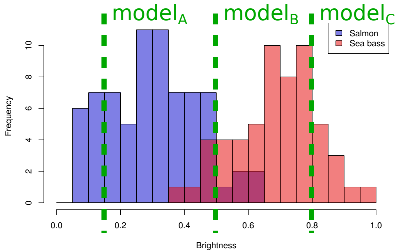
- How do we get the "best" model?
  1. How does our model perform on our data?
  2. How will it perform on (unseen) future data? (=how will it generalize?)

Back to our data

Length:



Brightness:



- How do we get the "best" model?
  1. How does our model perform on our data? – Loss function
  2. How will it perform on (unseen) future data? (=how will it generalize?)

Scoring our models: Loss function

- Assume we have a model g, parameterized by $\mathbf{w}$
- $g(\mathbf{x}; \mathbf{w})$ maps an input vector $\mathbf{x}$ to an output value $\hat{y}$
- We want $\hat{y}$ to be as close as possible to the true target value y

Scoring our models: Loss function

- Assume we have a model g, parameterized by $\mathbf{w}$
- $g(\mathbf{x}; \mathbf{w})$ maps an input vector $\mathbf{x}$ to an output value $\hat{y}$
- We want $\hat{y}$ to be as close as possible to the true target value y
- We can use a loss function

$$L(y, g(\mathbf{x}; \mathbf{w}))$$

  to measure how close our prediction is to the true target for a given sample with $\mathbf{z} = (\mathbf{x}^T, y)^T$
- The smaller the loss/cost, the better our prediction

Examples of loss functions

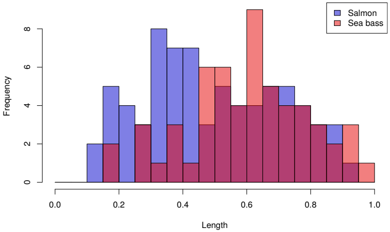
Zero-one loss: $L_{\mathbf{zo}}(y, g(\mathbf{x};\mathbf{w})) = \begin{cases} 0 & y = g(\mathbf{x};\mathbf{w}) \\ 1 & y \neq g(\mathbf{x};\mathbf{w}) \end{cases}$

Quadratic loss: $L_{\mathbf{q}}(y, g(\mathbf{x};\mathbf{w})) = (y - g(\mathbf{x};\mathbf{w}))^2$

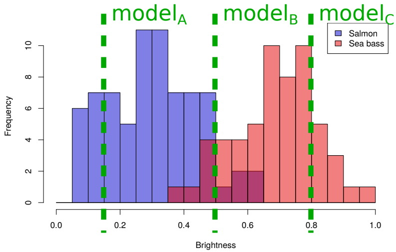
- Many other loss functions available with different justifications
- Not every loss function is suitable for every task
- Choice of loss function depends on data, task, and model class

Back to our data

Length:



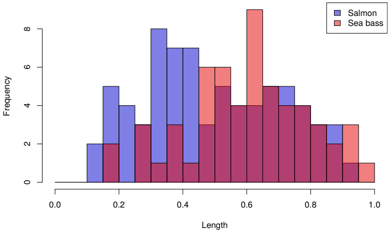
Brightness:



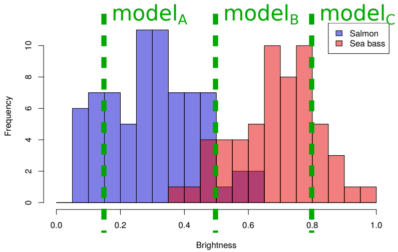
- How do we get the "best" model?
  1. ~~How does our model perform on our data?~~ ✓
  2. How will it perform on (unseen) future data? (=how will it generalize?)

Back to our data

Length:



Brightness:



- How do we get the "best" model?
  1. ~~How does our model perform on our data?~~ ✓
  2. How will it perform on (unseen) future data? (=how will it generalize?) – Generalization error/risk

Generalization error/risk

- The generalization error or risk is the expected loss on future data for a given model $g(.;\mathbf{w})$:

$$R(g(.;\mathbf{w})) = \int_X \int_{\mathbb{R}} L(y, g(\mathbf{x};\mathbf{w})) \cdot p(\mathbf{x}, y) dy d\mathbf{x}$$

- $R(g(\mathbf{x};\mathbf{w}))$ denotes the expected loss for input $\mathbf{x}$

Generalization error/risk

- The generalization error or risk is the expected loss on future data for a given model $g(.;\mathbf{w})$:

$$R(g(.;\mathbf{w})) = \int_X \int_{\mathbb{R}} L(y, g(\mathbf{x};\mathbf{w})) \cdot p(\mathbf{x}, y) dy d\mathbf{x}$$

- $R(g(\mathbf{x};\mathbf{w}))$ denotes the expected loss for input $\mathbf{x}$
- In practice, we hardly have any knowledge about $p(\mathbf{x}, y)$

Generalization error/risk

- The generalization error or risk is the expected loss on future data for a given model $g(.;\mathbf{w})$:

$$R(g(.;\mathbf{w})) = \int\limits_X \int\limits_{\mathbb{R}} L(y, g(\mathbf{x};\mathbf{w})) \cdot p(\mathbf{x}, y) dy d\mathbf{x}$$

- $R(g(\mathbf{x};\mathbf{w}))$ denotes the expected loss for input $\mathbf{x}$
- In practice, we hardly have any knowledge about $p(\mathbf{x}, y)$

$\rightarrow$ We have to estimate the generalization error

Empirical Risk Minimization (ERM)

- We do not know the true $p(\mathbf{x}, y)$ but we have access to a subset of n data samples (=our dataset)

Empirical Risk Minimization (ERM)

- We do not know the true $p(\mathbf{x}, y)$ but we have access to a subset of n data samples (=our dataset)
- We can minimize the empirical risk R_{emp} on our dataset (=Empirical Risk Minimization):

$$R_{\text{emp}}(g(.;\mathbf{w}), \mathbf{Z}_n) = \frac{1}{n} \cdot \sum_{i=1}^{n} L(y^i, g(\mathbf{x}^i; \mathbf{w}))$$

Empirical Risk Minimization (ERM)

- We do not know the true $p(\mathbf{x}, y)$ but we have access to a subset of n data samples (=our dataset)
- We can minimize the empirical risk R_{emp} on our dataset (=Empirical Risk Minimization):

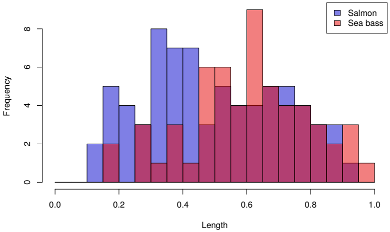
$$R_{\text{emp}}(g(.;\mathbf{w}), \mathbf{Z}_n) = \frac{1}{n} \cdot \sum_{i=1}^{n} L(y^i, g(\mathbf{x}^i; \mathbf{w}))$$

- Strong law of large numbers:

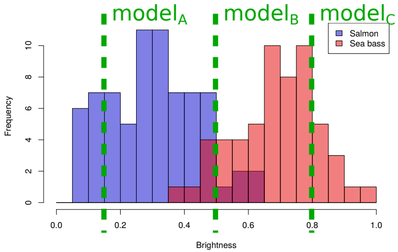
$$R_{\text{emp}}(g(.;\mathbf{w})) \rightarrow R(g(.;\mathbf{w})) \text{ for } n \rightarrow \infty$$

Back to our data

Length:



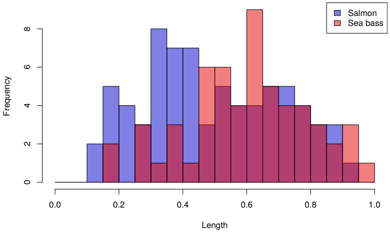
Brightness:



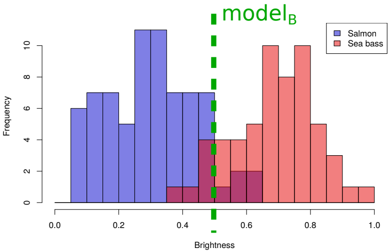
- How do we get the "best" model?
    1. ~~How does our model perform on our data?~~ ✓
    2. ~~How will it perform on (unseen) future data? (=how will it generalize?)~~ ✓

Back to our data

Length:



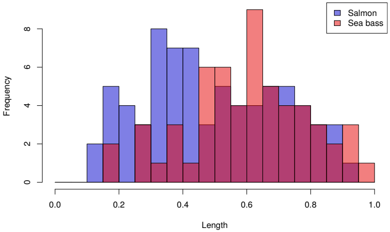
Brightness:



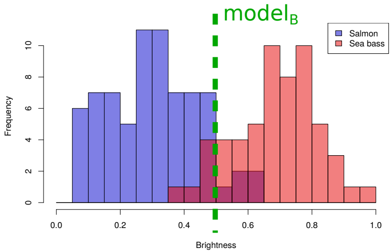
- We can now optimize our model by minimizing the risk on our (training) dataset!

Back to our data

Length:



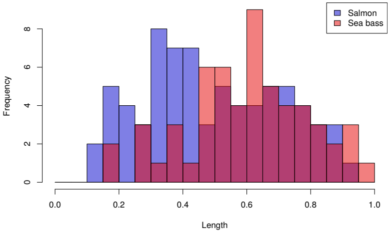
Brightness:



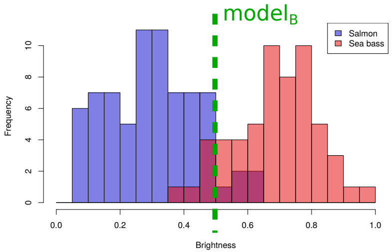
- We can now optimize our model by minimizing the our (training) dataset!
- . . . but the individual features do not separate the classes well :-(

Back to our data

Length:



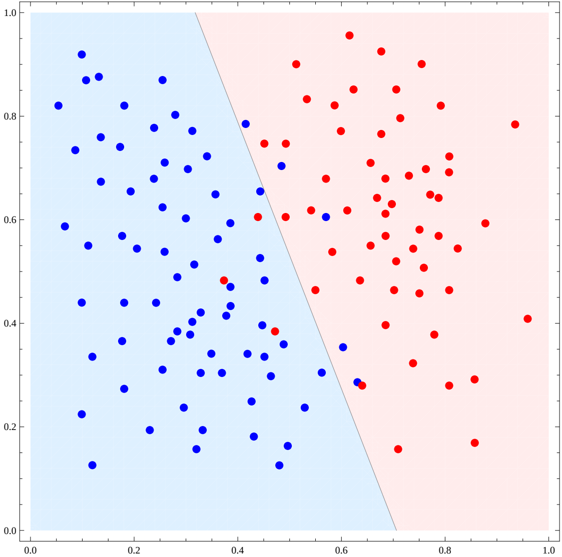
Brightness:



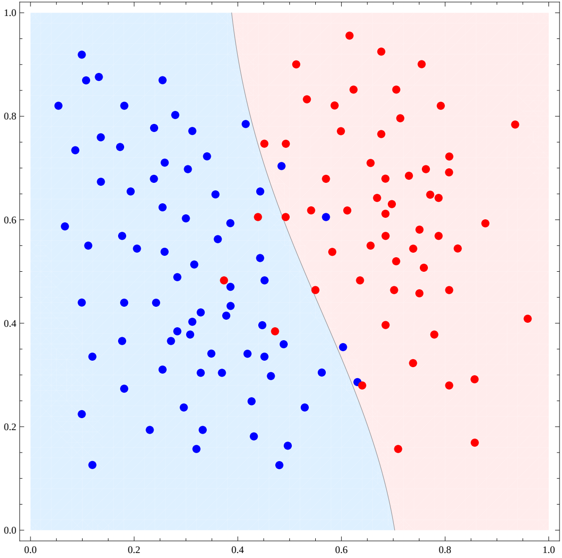
- We can now optimize our model by minimizing the our (training) dataset!
- . . . but the individual features do not separate the classes well :-(

$\rightarrow$ Let’s combine our features and use a different model class!

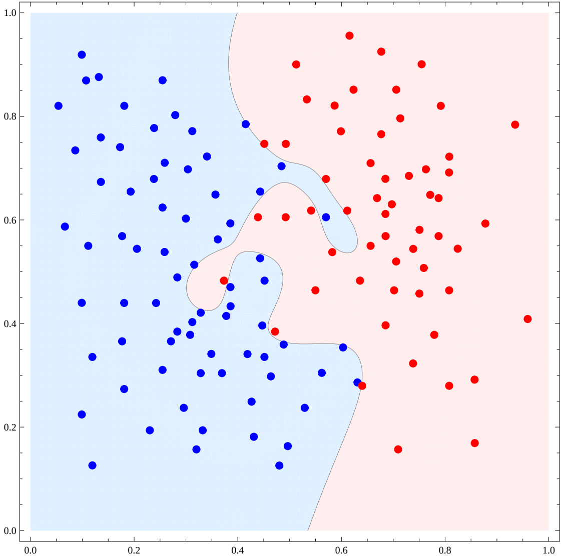
Combined features: Linear separation



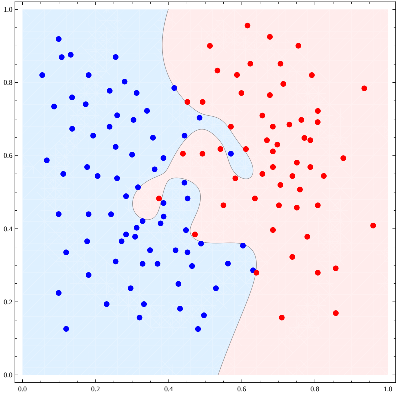
Combined features: Mildly non-linear separation



Combined features: Highly non-linear separation

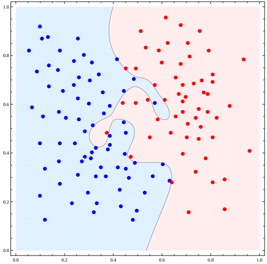


Combined features: Highly non-linear separation



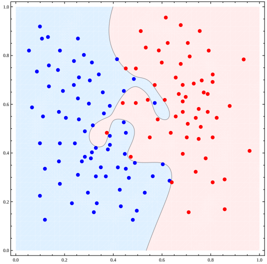
Seems perfect on our data, right? Are we done now?

The problem of overfitting



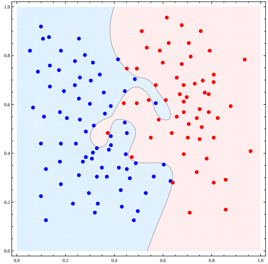
- With ERM we can optimize our model by minimizing the risk on our (training) dataset

The problem of overfitting



- With ERM we can optimize our model by minimizing the risk on our (training) dataset
- Problem: We might fit our parameters to noise specific to our training dataset (=overfitting)

The problem of overfitting



- With ERM we can optimize our model by minimizing the risk on our (training) dataset
- Problem: We might fit our parameters to noise specific to our training dataset (=overfitting)

$\rightarrow$ we need to get a better estimate for the (true) risk

Risk estimation: Test set method

- Assume our data samples are independently and identically distributed (i.i.d.)*
- We can split our dataset of n samples into 2 subsets:

  Training set: the subset with l samples we perform ERM on (i.e. optimize parameters on)

  Test set: a subset with m samples we use to estimate the risk

[*) i.i.d.: each sample has the same probability distribution as the others and all are mutually independent.]

Risk estimation: Test set method

- Assume our data samples are independently and identically distributed (i.i.d.)*
- We can split our dataset of n samples into 2 subsets:

  Training set: the subset with l samples we perform ERM on (i.e. optimize parameters on)

  Test set: a subset with m samples we use to estimate the risk
- Our estimate R_E on test set will show if we overfit to noise in training set! :)

[*) i.i.d.: each sample has the same probability distribution as the others and all are mutually independent.]

Test set method: Practical hints

- No overlap between training and test set samples (i.i.d.!)
- Random sampling of training and test set samples (i.i.d.!)
- Test set samples are not to be used for preprocessing, feature selection, model selection, etc.

Test set method: Practical hints

- No overlap between training and test set samples (i.i.d.!)
- Random sampling of training and test set samples (i.i.d.!)
- Test set samples are not to be used for preprocessing, feature selection, model selection, etc.
- We might want to use 3 separate subsets:

  Training set: subset we train a model on (optimize model parameters)

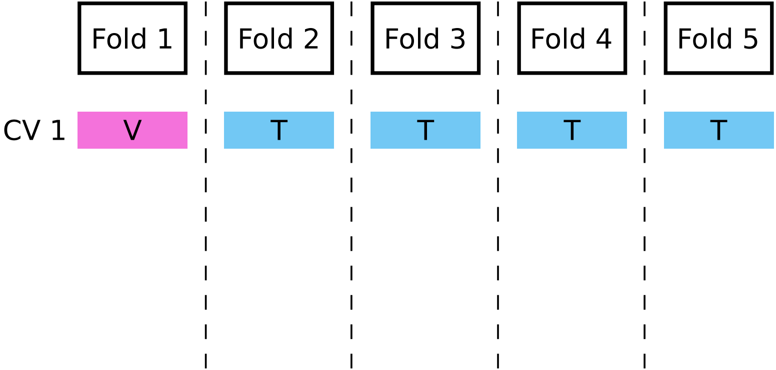
  Validation set: subset we select best model from training on (=model selection)

  Test set: subset we use to estimate risk

Cross Validation (1)

- For small datasets, the requirement that training and test set must not overlap is painful
- Solution: Cross Validation (CV)
  - Split dataset into n disjoint folds
  - Use $n-1$ folds as training set, left-out fold as test set
  - Train n times, every time leaving out a different fold as test set
  - Average over n estimated risks on test sets to get better estimate of generalization capability

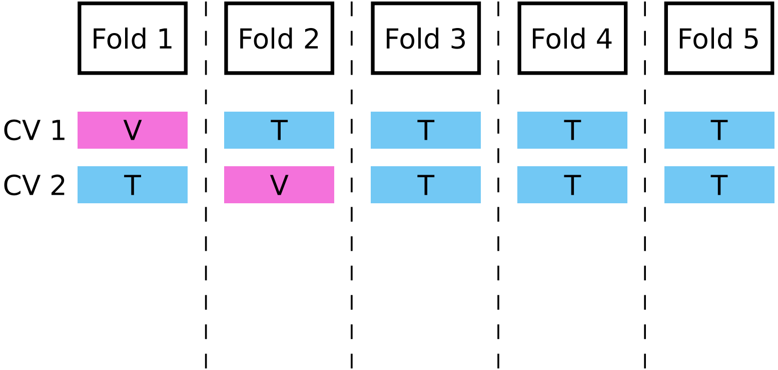
Cross Validation (1)



5-fold Cross Validation

T: Training set; V: Test set

Cross Validation (1)



5-fold Cross Validation

T: Training set; V: Test set

Cross Validation (1)

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
CV 1	V	T	T	T	T
CV 2	T	V	T	T	T
CV 3	T	T	V	T	T
CV 4	T	T	T	V	T
CV 5	T	T	T	T	V

5-fold Cross Validation

T: Training set; V: Test set

Cross Validation (2)

- Nested Cross Validation
  - We can apply another (inner) CV procedure within each training-set of the original (outer) CV

  $\rightarrow$ allows for evaluation of model selection procedure

Cross Validation (2)

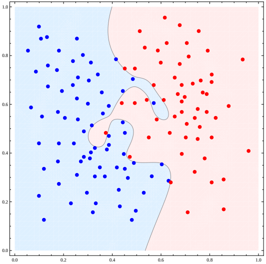
- Nested Cross Validation
  - We can apply another (inner) CV procedure within each training-set of the original (outer) CV

  $\rightarrow$ allows for evaluation of model selection procedure
- Getting a risk estimate on selected model:
  1. Apply cross validation on training set (withhold test set)
  2. Use test set to estimate risk for the model selected via CV

Cross Validation (2)

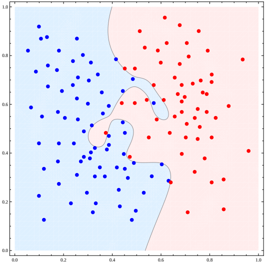
- Nested Cross Validation
  - We can apply another (inner) CV procedure within each training-set of the original (outer) CV
  - → allows for evaluation of model selection procedure
- Getting a risk estimate on selected model:
  1. Apply cross validation on training set (withhold test set)
  2. Use test set to estimate risk for the model selected via CV
- In practice, the found model is often trained further or re-trained on complete dataset for best performance

Back to our data



- Now we know we can use ERM to optimize a model on our training dataset (optionally via CV)
- A held-out test set will allow us to get an estimate about the performance on future data (optionally via CV)

Back to our data



- Now we know we can use ERM to optimize a model on our training dataset (optionally via CV)
- A held-out test set will allow us to get an estimate about the performance on future data (optionally via CV)

Done! :)

Common pitfalls

Underfitting: model is too coarse to fit training or test data (too low model complexity)

Overfitting: model fits well to training data but not to future/test data (too high model complexity)

Unbalanced dataset: datasets biased toward a single class need to be evaluated properly (balanced accuracy, ROC AUC, loss weighting, ...)

Hints

- Separate the test set as soon as possible (no feature selection on test set data!)
- Inspect your dataset (clusters/peculiarities due to data creation, artefacts, ...)
- Which CV/training/evaluation method to use depends on what you want to show/achieve (method comparison, winning a challenge, ...)

Hints

- Separate the test set as soon as possible (no feature selection on test set data!)
- Inspect your dataset (clusters/peculiarities due to data creation, artefacts, ...)
- Which CV/training/evaluation method to use depends on what you want to show/achieve (method comparison, winning a challenge, ...)
  - Example:
    - Your data was recorded by 5 different labs
    - You want the algorithm to generalize to new labs

    → If CV folds do not share the same labs we get an estimate for generalization to new labs (=cluster cross validation)

Summary

1. Acquire labeled dataset (input features + target values)
2. Divide dataset into training and test set
3. Select preprocessing pipeline, features, and model class based on training set
4. Optimize the model parameters on the training set
5. Optionally use validation set or CV to determine best model
6. Go back to step 3 if evaluation on validation/training set gave new insights
7. Use test set to calculate estimate for generalization error/risk

Summary

1. Acquire labeled dataset (input features + target values)
2. Divide dataset into training and test set
3. Select preprocessing pipeline, features, and model class based on training set
4. Optimize the model parameters on the training set
5. Optionally use validation set or CV to determine best model
6. Go back to step 3 if evaluation on validation/training set gave new insights
7. Use test set to calculate estimate for generalization error/risk

Done! :)