Model Selection

Maschinelles Lernen 1 - Grundverfahren WS20/21

Prof. Gerhard Neumann

KIT, Institut für Anthrophomatik und Robotik

Learning Outcomes

- Understand the overfitting problem and …
- … its relation to the complexity of the model class
- Bias variance tradeoff
- Why we need test-sets and cross-validation
- Understand different regularization methods

Agenda for today

Model Selection

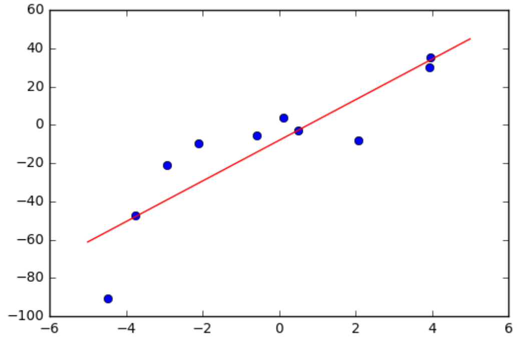
- Overfitting and model complexity
- Bias variance trade-off
- Hold-out set and cross validation

Regularization:

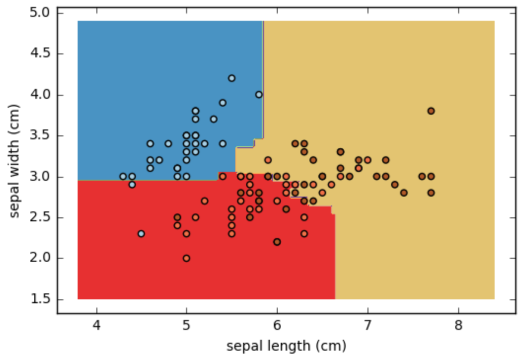
- Limit complexity
- Penalty terms
- Early stopping
- Noise and data augmentation

Zoo of algorithms for machine learning

- **Regression: Continuous output labels**
  - Linear regression, Polynomial Regression, kNN, Regression Trees, Gaussian Processes, Neural Nets



- **Classification: Discrete / Nominal output labels**
  - Logistic Regression, Decision Trees, Neural Nets, SVMs, kNN



Model Complexity

For most of these algorithms, we have to choose the model complexity

- **Linear Regression:** number of features, regularization coefficient
- **Decision Trees:** maximum depth, number of leaves
- **Neural Networks:** number of layers, number of neurons
- **Support Vector Machine:** which features, regularization
- **Gaussian Processes:** kernel bandwith

Choosing the right complexity is a model selection problem!

- And one of the most fundamental problems in ML

True risk vs. empirical risk

True risk: performance on a random test point (x,y)

- **Classification:** probability of misclassification $p(y \neq f(x))$
- **Regression:** expected squared error $\mathbb{E}_{\boldsymbol{x},y}\left[(f(\boldsymbol{x}) - y)^2\right]$

➢ Unknown!

Empirical risk: performance on the training set

- **Classification:** proportion of misclassified samples $\frac{1}{n}\sum_i \mathbb{I}(f(\boldsymbol{x}_i) \neq y_i)$
- **Regression:** average squared error $\frac{1}{n}\sum_i (f(\boldsymbol{x}_i) - y_i)^2$

➢ Can be evaluated

Overfitting

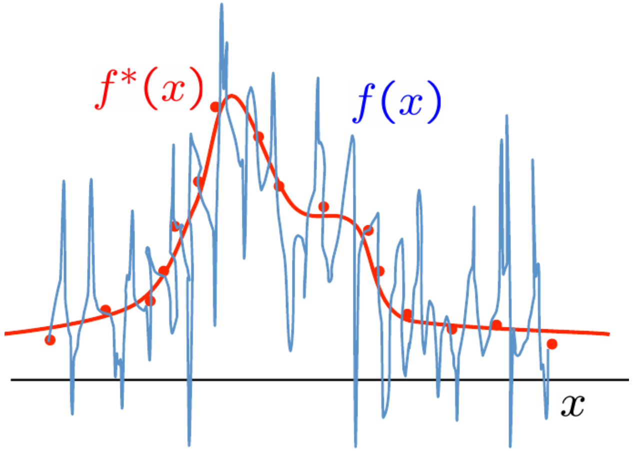
Is the following predictor good?

$$f(x) = \begin{cases} y_i, & \text{if } x = x_i \text{ for } i = 1 \dots n \\ \text{any other value, else} \end{cases}$$

Empirical risk? Zero!

True risk? Huge!

- Will predict poorly on unseen data
- Large generalization error

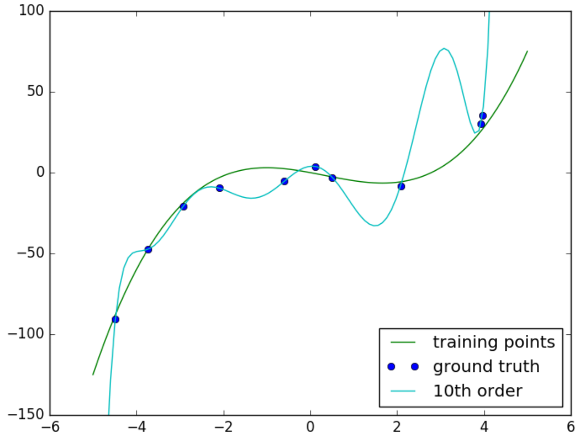


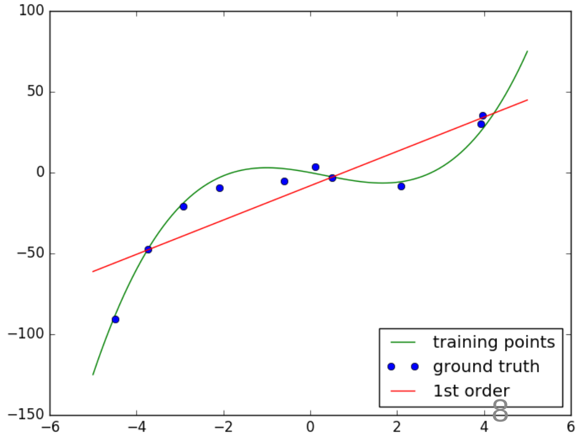
Model Selection

Choose complexity of the model class

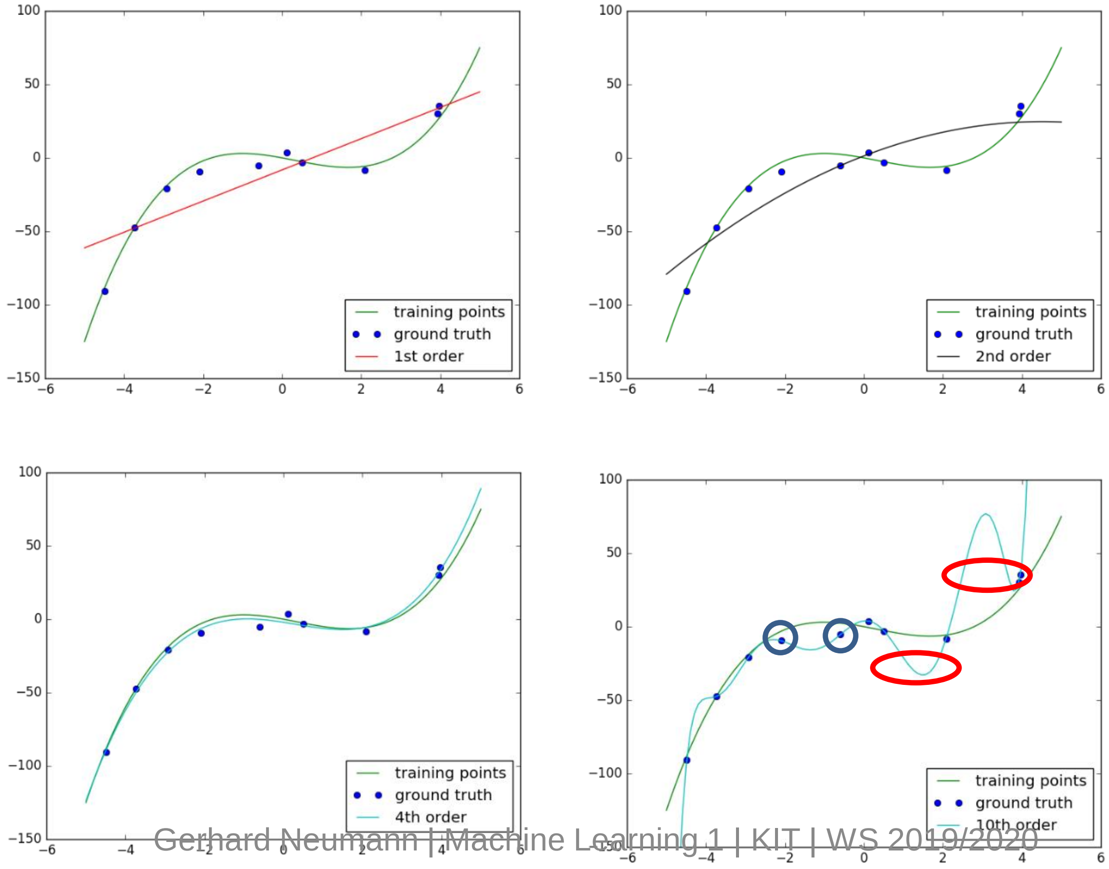
- **Too complex: Overfitting**
  - Fit noise in the data
  - Unspecified behavior between data points
  - Not enough data
- **Too simple: Underfitting**
  - We can not represent the underlying function

Lets look at an example…





Polynomial regression up to kth order



Last model overfits the data:

- Fits the noise
- Unspecified behavior in between datapoints

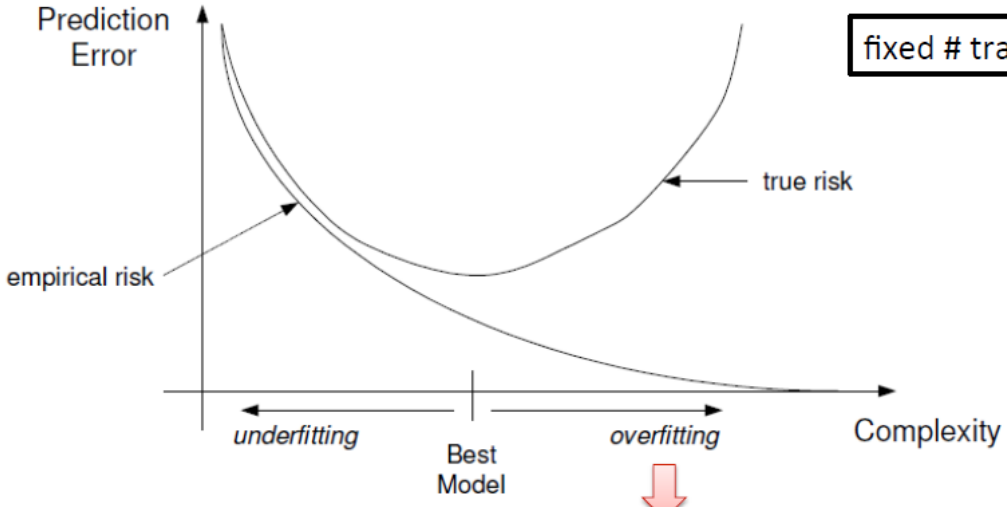
Effect of Model Complexity

Overfitting:

Small empirical risk, but true risk is high

Underfitting:

High empirical risk and true risk

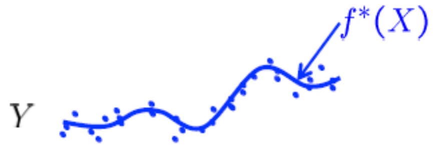


fixed # training data

Empirical risk is no longer a good indicator of true risk

Bias-Variance Decomposition

Regression example: $y = f(\boldsymbol{x}) + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma^2)$



Expected Loss = Variance + Bias² + Noise

Bias:

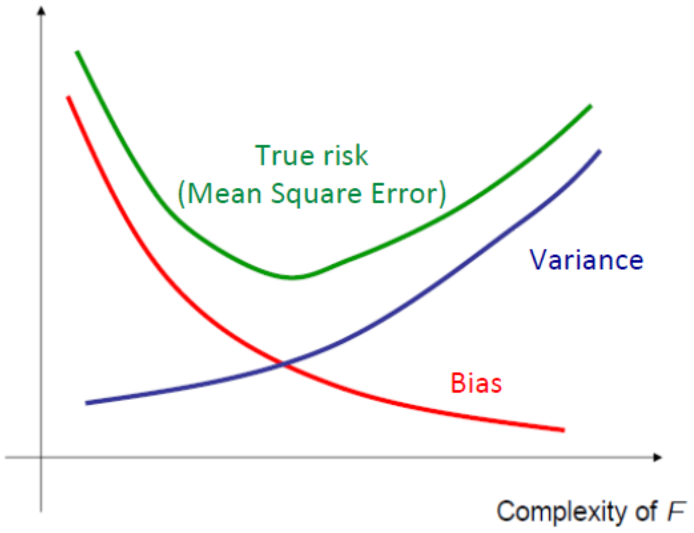
- Due to restriction of your model class
- Also called "structure error"

Variance:

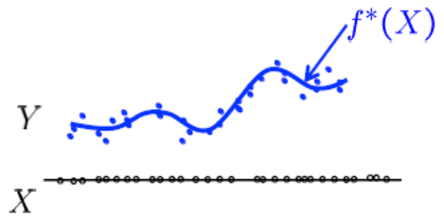
- Due to randomness of the data set

Noise:

- Nothing we can do about it…



Understanding the true risk



Regression example: $y = f(\boldsymbol{x}) + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma^2)$

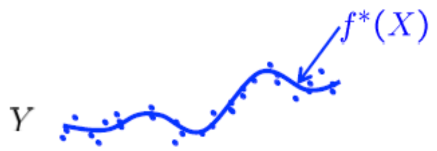
Expected Loss:

$$R(\hat{f}_{D_n}) = \mathbb{E}_{D_n}\left[\underbrace{\mathbb{E}_{x,y}\left[(\hat{f}_{D_n}(\boldsymbol{x}) - y)^2\right]}_{}\right]$$

Expected error for training with a specific dataset
Expectation is done w.r.t to all possible inputs and corresponding outputs

- $\hat{f}_{D_n}$ is the estimate of f we obtain using data D_n
- The expectation is done w.r.t all training-sets D_n of size n. What does that mean?
  - Assume there is a process that can generate data sets n data-points, e.g.:
    - Sample x_i uniformly in range [-1,1]
    - Sample y_i as $y_i = f(\boldsymbol{x}_i) + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma^2)$
    - Repeat n times
  - The expectation averages the error over all training-sets D_n

Understanding the true risk



Regression example: $y = f(\boldsymbol{x}) + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma^2)$

Expected Loss:

$$R(\hat{f}_{D_n}) = \mathbb{E}_{D_n}\left[\mathbb{E}_{x,y}\left[(\hat{f}_{D_n}(\boldsymbol{x}) - y)^2\right]\right]$$

$$= \underbrace{\mathbb{E}_{D_n}\left[\mathbb{E}_{x,y}\left[\left(\hat{f}_{D_n}(\boldsymbol{x}) - \hat{f}_*(\boldsymbol{x})\right)^2\right]\right]}_{\text{Variance}} + \underbrace{\mathbb{E}_{x,y}\left[\left(\hat{f}_*(\boldsymbol{x}) - f(\boldsymbol{x})\right)^2\right]}_{\text{Bias}^2} + \underbrace{\sigma^2}_{\text{noise}}$$

- $\hat{f}_*(\boldsymbol{x}) = \mathbb{E}_{D_n}[\hat{f}_{D_n}(\boldsymbol{x})]$ is the average estimate, averaged over all data sets of size n
- … which can approximately be seen as training with infinite data

Bias-Variance Decomposition

Observations:

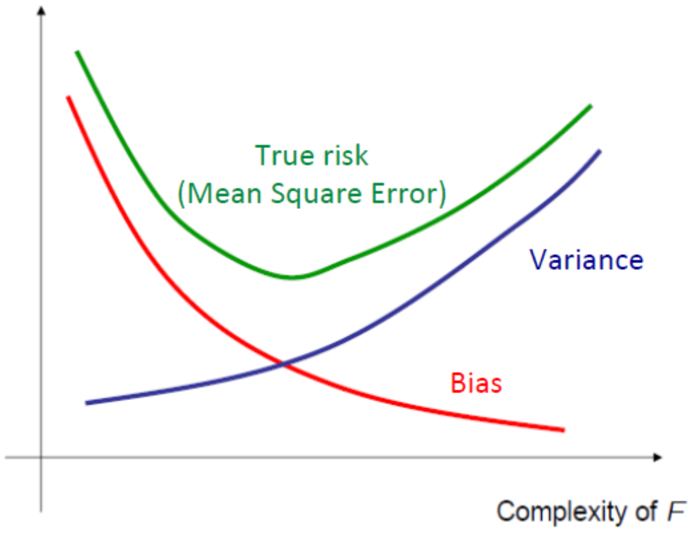
- We can not get better than the noise

Bias: $\mathbb{E}_{x,y}\left[\left(\hat{f}_*(\boldsymbol{x}) - f(\boldsymbol{x})\right)^2\right]$

- Difference of true function to the “best” estimate
- The best you can do with your model class
- Also called “approximation error”

Variance: $\mathbb{E}_{D_n}\left[\mathbb{E}_{x,y}\left[\left(\hat{f}_{D_n}(\boldsymbol{x}) - \hat{f}_*(\boldsymbol{x})\right)^2\right]\right]$

- Difference of the estimates to the “best” estimate
- Due to limited size of the data set
- Depends on number of data-points
- Also called “estimation error”



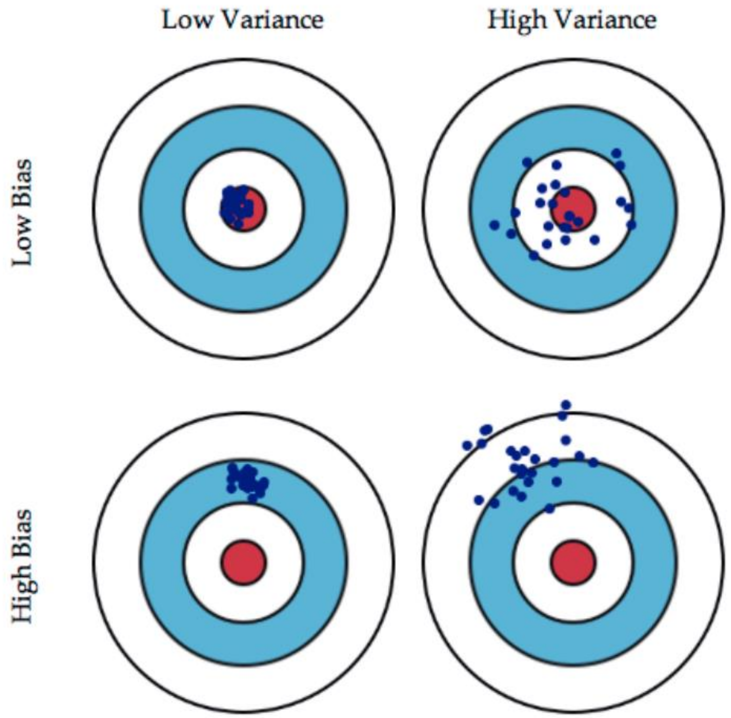
Derivation of the Bias-Variance Trade-off

A lot of math… look in the appendix if interested

Bias-Variance Trade-off

Each point represents a fitted model with a different dataset

- **Low variance, high bias:** Underfitting
- **High variance, low bias:** Overfitting
- **High variance, high bias:** something is terribly wrong
- **Low variance, low bias:** too good to be true



Agenda for today

Model Selection

- Overfitting and model complexity
- Bias variance trade-off
- Hold-out set and cross validation

Regularization:

- Limit complexity
- Penalty terms
- Early stopping
- Noise and data augmentation

Evaluation Methods

We have seen that the empirical risk on the training set is not a good indicator for the quality of your model. What can we do?

- Hold-out method
- Cross-validation

Hold-out method

- We would like to pick the model M with the smallest generalization error
- Can judge the generalization error by independent data set (not used for training)

Hold-out procedure: n datapoints available $D = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$

1. Split into 2 datasets:

   Training Data $D_T = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^m$

   Validation Data $D_V = \{(\boldsymbol{x}_i, y_i)\}_{i=m+1}^n$

2. Train on training data to obtain $\hat{f}_{D_T}(\boldsymbol{x})$ for each model class M
3. Evaluate resulting estimators on validation data, e.g: $\mathrm{MSE}(D_V, \hat{f}_{D_T}) = \frac{1}{n-m} \sum_{i=m+1}^{n} (\hat{f}_{D_T}(\boldsymbol{x}_i) - y_i)^2$
4. Pick model with best validation loss

Hold-out method

Drawbacks:

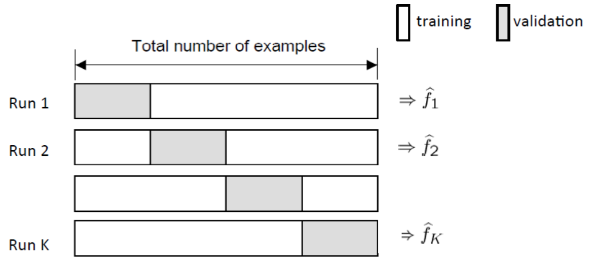
- Costly in terms of data
- “Unlucky” splits might give misleading results

Cross-validation methods fix these issues

Cross validation

K-fold cross validation

1. Create k-fold partition of the dataset
2. Estimate k hold-out predictors using 1 partition as validation and k-1 partitions as training set

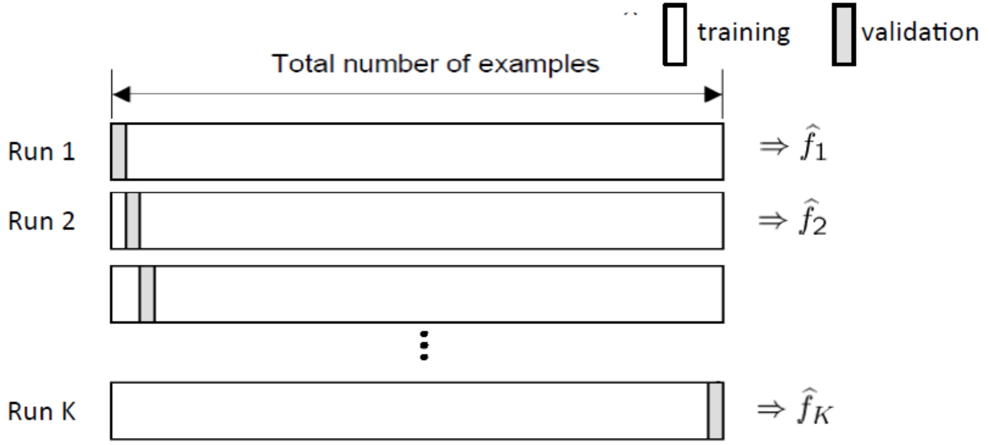


k predictors for each model class:

Cross validation

Leave-One-Out (LOO) cross validation

1. Special case with k = n
2. Consequently, estimate n hold-out predictors using 1 sample as validation and n-1 samples as training set

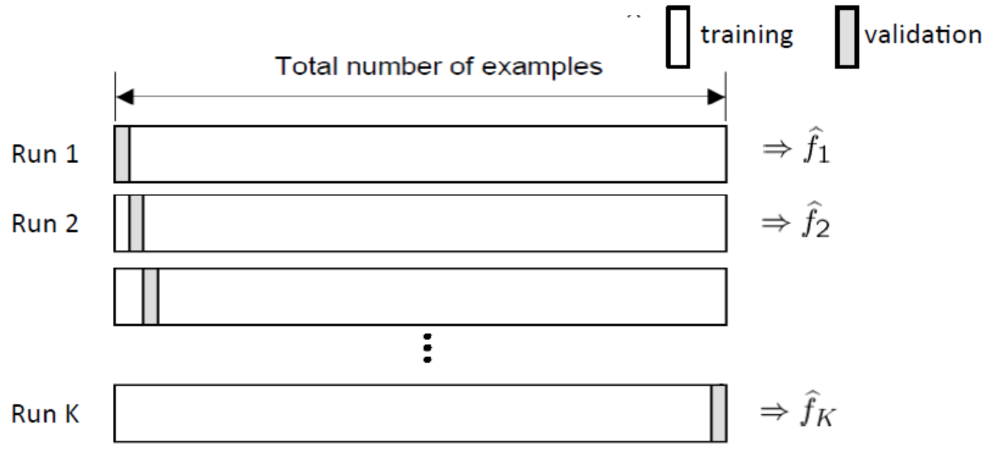


k predictors for each model class:

Cross validation

Random sub-sampling

1. Randomly sample a fraction of alpha * n (0 < alpha < 1) data points for validation
2. Train on remaining points and validate, repeat K times



k predictors for each model class:

Regularization techniques

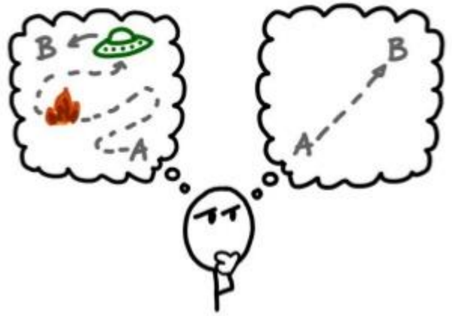
How to avoid overfitting?

- Limit the complexity of the model (# neurons, # of leaves, etc…)
- Regularization penalty
- Early stopping
- Noise and Data-Augmentation
- …

Occam's Razor

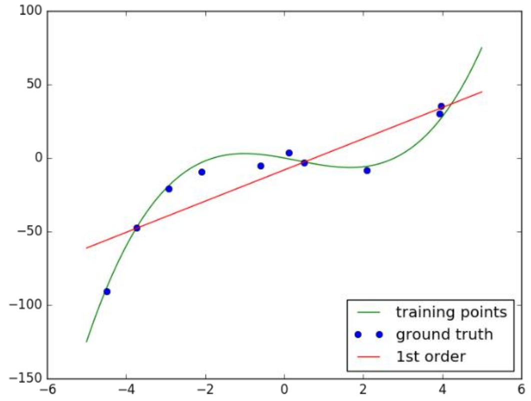
- Named after William of Occam – AD 1300s
- Prefer simpler explanations over more complex ones
  - "Numquam ponenda est pluralitas sine necessitate"
  - (Never posit plurality without necessity.)
- Historically, a widely prevalent idea across different schools of philosophy
- Directly applicable for model selection in ML

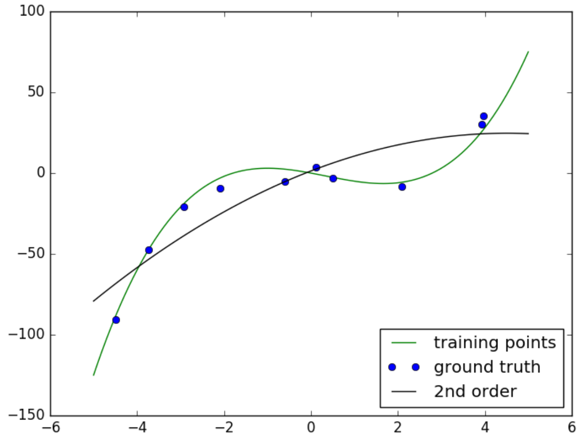
Occam's Razor

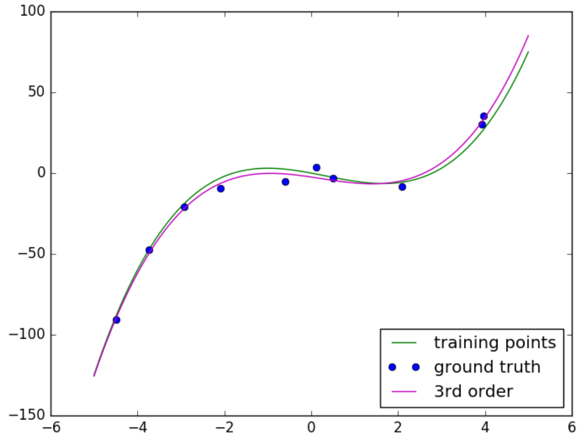


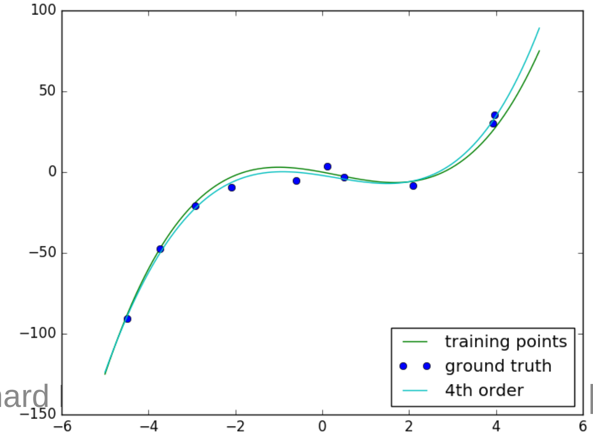
"When faced with two equally good hypotheses, always choose the simpler."

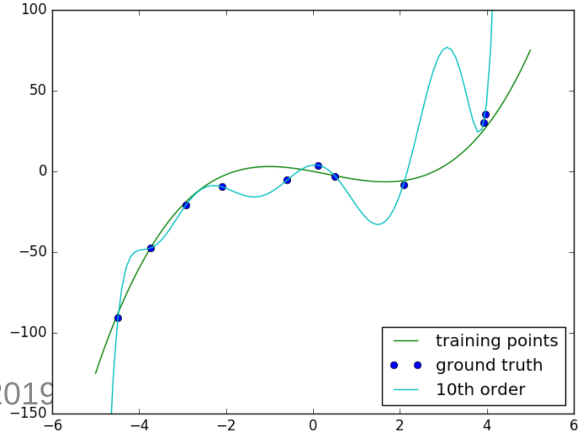
Limit complexity – Example with polynoms





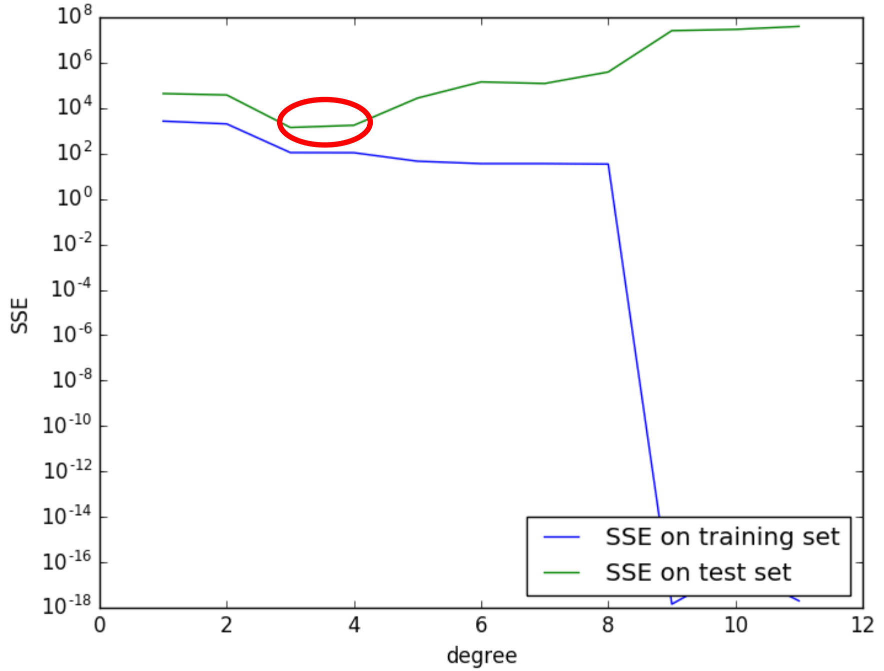






Model Selection for polynomial regression

- **Overfitting:**
  - Training error goes down
  - Validation error goes up
- **Underfitting:**
  - Training + Validation error are high
- **Optimum:** 3rd or 4th degree



Regularization penalty

Can be used for most optimization-based algorithms

- Linear Regression + Classification, Neural Networks, GPs, …

We, typically optimize a (sample-based) **loss plus a regularization penalty**

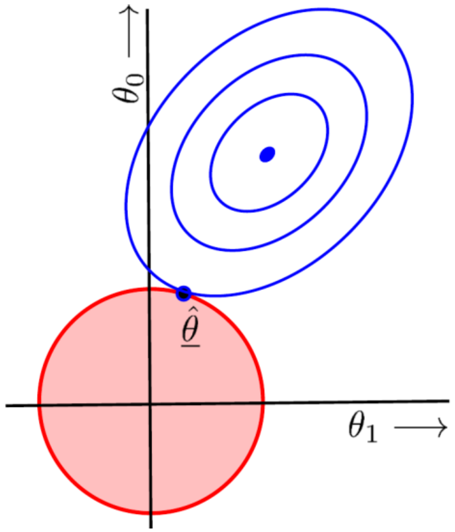
$$\underset{\text{parameters } \boldsymbol{\theta}}{\arg\min} \sum_{i=1}^{N} l(\boldsymbol{x}_i, \boldsymbol{\theta}) + \lambda \, \text{penalty}(\boldsymbol{\theta})$$

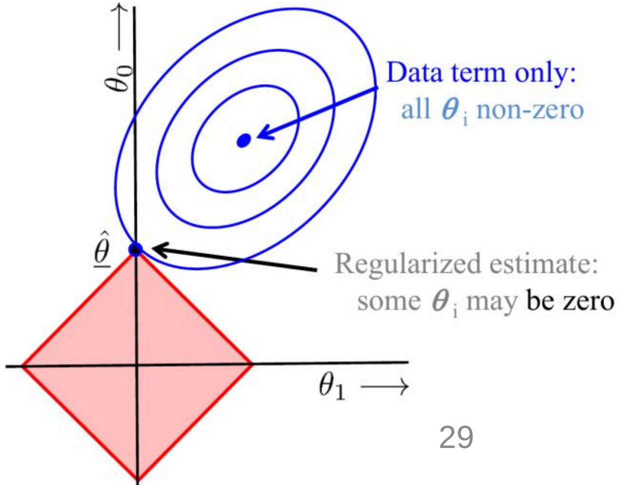
- Penalty keeps parameters small
- Small parameters -> **smoother function** estimate
- Implicitly **limits the complexity** of the learned model (larger lambda -> smaller complexity)

Regularization penalty

Which penalty functions can we use?

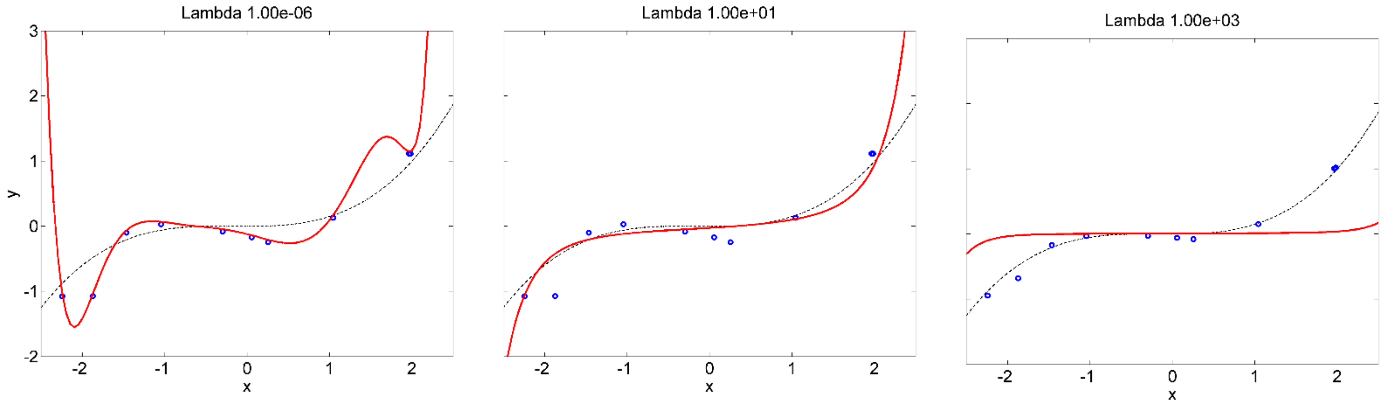
- ***L₂* penalty:** $$\text{penalty}(\boldsymbol{\theta}) = ||\boldsymbol{\theta}||_2 = \sum_d \theta_d^2$$
  - Easy to optimize (strongly convex)
  - Closed form solutions exists
  - Redundant parameters will be close to 0, but never 0
- ***L₁* penalty:** $$\text{penalty}(\boldsymbol{\theta}) = ||\boldsymbol{\theta}||_1 = \sum_d |\theta_d|$$
  - Induces **sparse** solutions
  - Called "Lasso" regularization
  - Much harder to optimize (not in this lecture)


$\theta_0 \longrightarrow$
$\hat{\underline{\theta}}$
$\theta_1 \longrightarrow$


$\theta_0 \longrightarrow$
Data term only: all $\boldsymbol{\theta}_i$ non-zero
$\hat{\underline{\theta}}$
Regularized estimate: some $\boldsymbol{\theta}_i$ may be zero
$\theta_1 \longrightarrow$

Example: ridge regression

Ridge regression with polynomial of degree n=15

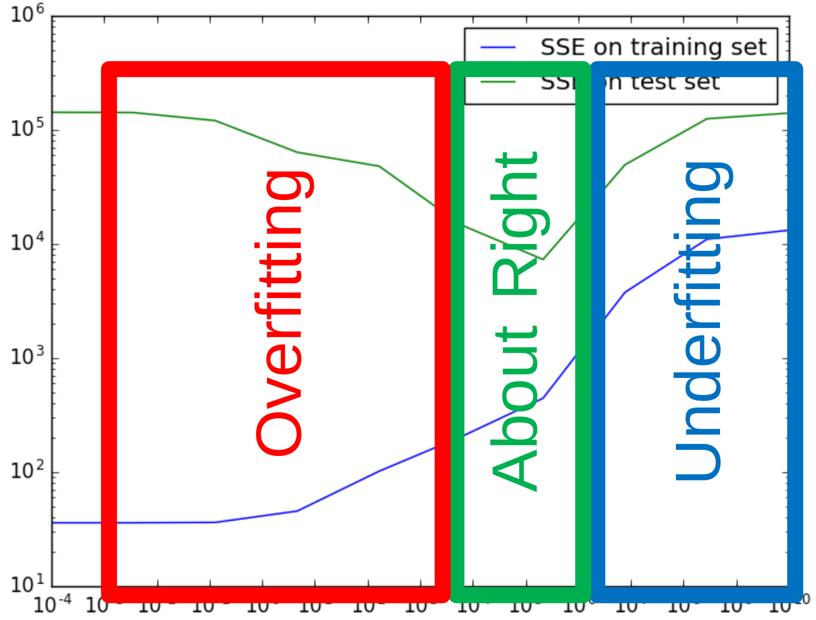


Influence of the regularization constant

Example: Ridge regression

Influence of the lambda parameter

- High Lambda: Underfitting
  - High training and validation error
- About Right:
  - Validation error is minimal
- Small Lambda: Overfitting
  - Small training error, but high validation error



Early stopping

Idea: don’t train to too small training error

- Used with incremental learning rules (e.g. gradient descent)
- Prevent overfitting: do not push the model too much; use validation error to decide when to stop
- Implicitely limits complexity

Early stopping

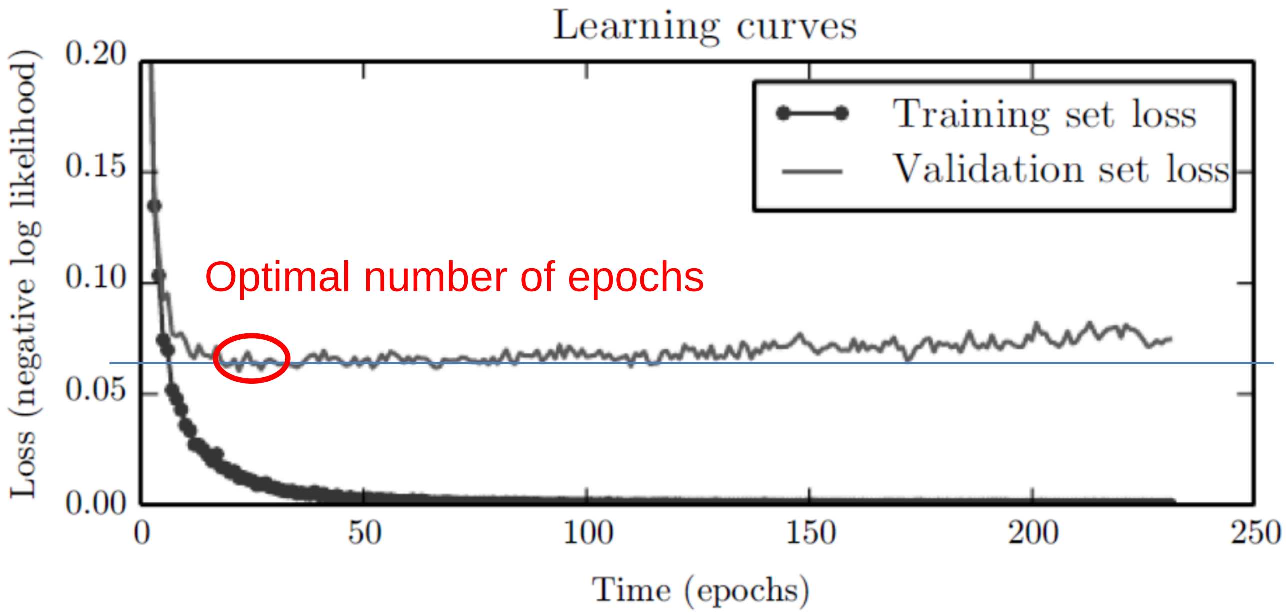


Figure from *Deep Learning*, Goodfellow, Bengio and Courville

- Validation **error goes up due to overfitting**

Early Stopping

- When training, also output validation error
- Every time validation error improved, store a copy of the weights
- When validation error not improved for some time, stop
- Return the copy of the weights stored

Early stopping as regularizer

- Early stopping has similar effects than L_2 regularization

Advantage

- Efficient: along with training; only store an extra copy of weights
- Simple: no change to the model/algo
- No hyper-parameter (such as lambda)

Disadvantage

- need validation data

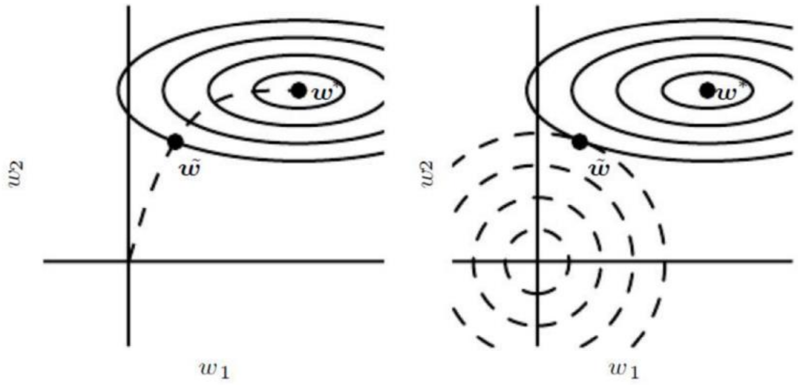
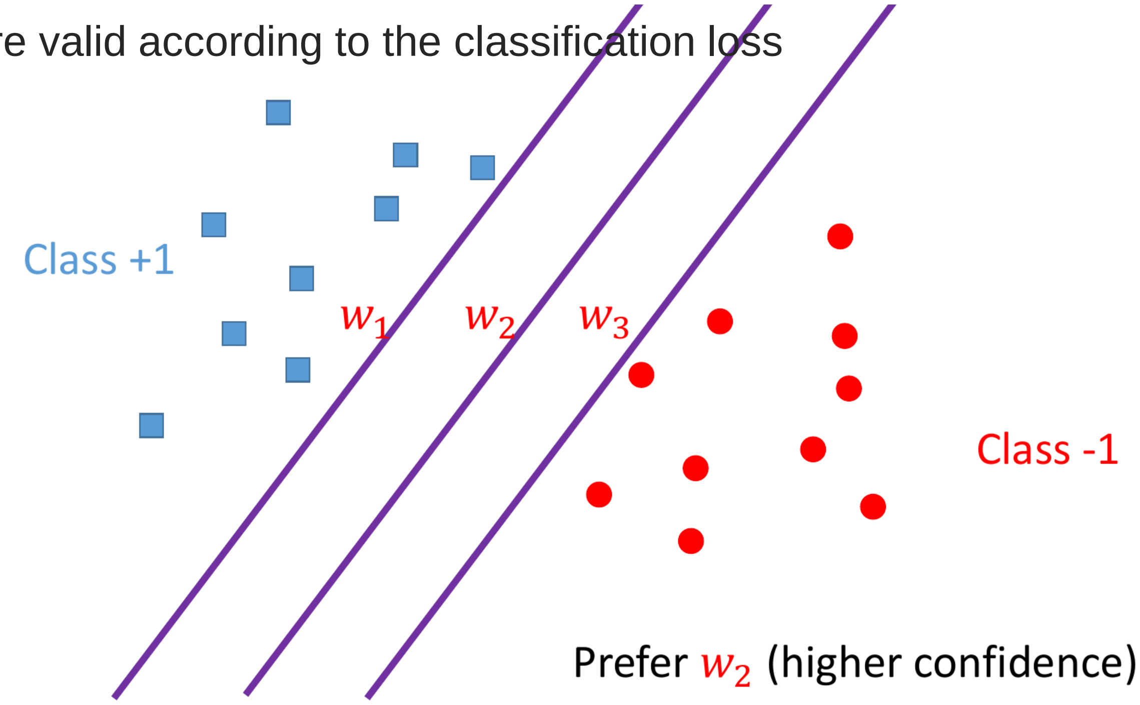


Figure from *Deep Learning*, Goodfellow, Bengio and Courville

Robustness to noise

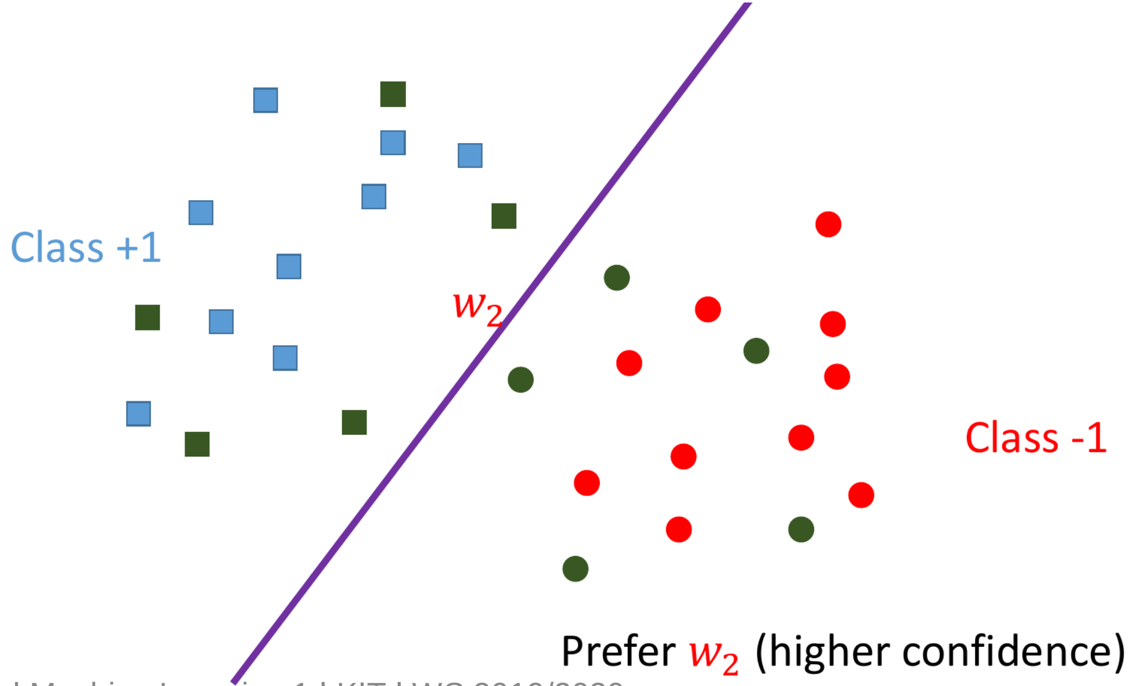
All solutions are valid according to the classification loss



Prefer w_2 (higher confidence)

Adding noise to the inputs

- Rules out unlikely / not robust solutions



Equivalence to L2 regularization

For a linear regression model, the noise model is given by

$$f(\boldsymbol{x}+\boldsymbol{\epsilon}) = \boldsymbol{w}^T(\boldsymbol{x}+\boldsymbol{\epsilon}), \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \lambda \boldsymbol{I})$$

This leads to the following loss:

$$\begin{aligned}\mathrm{MSE}(\boldsymbol{w}) &= \mathbb{E}_{\boldsymbol{x},y,\boldsymbol{\epsilon}}\left[(\boldsymbol{w}^T\boldsymbol{x} - y + \boldsymbol{w}^T\boldsymbol{\epsilon})^2\right] \\ &= \underbrace{\mathbb{E}_{\boldsymbol{x},y}\left[(\boldsymbol{w}^T\boldsymbol{x} - y)^2\right]}_{n\mathrm{SSE}(\boldsymbol{w})} + 2\underbrace{\mathbb{E}_{\boldsymbol{x},y,\boldsymbol{\epsilon}}\left[(\boldsymbol{w}^T\boldsymbol{x} - y)\boldsymbol{w}^T\boldsymbol{\epsilon}\right]}_{=0,\text{zero mean, i.i.d. noise}} + \underbrace{\mathbb{E}_{\boldsymbol{x},y,\boldsymbol{w}}\left[(\boldsymbol{w}^T\boldsymbol{\epsilon})^2\right]}_{\lambda\boldsymbol{w}^T\boldsymbol{w}} \\ &= n\mathrm{SSE}(\boldsymbol{w}) + \lambda||\boldsymbol{w}||_2^2\end{aligned}$$

I.e., for linear regression, **input noise** is the same as **L2 regularization**

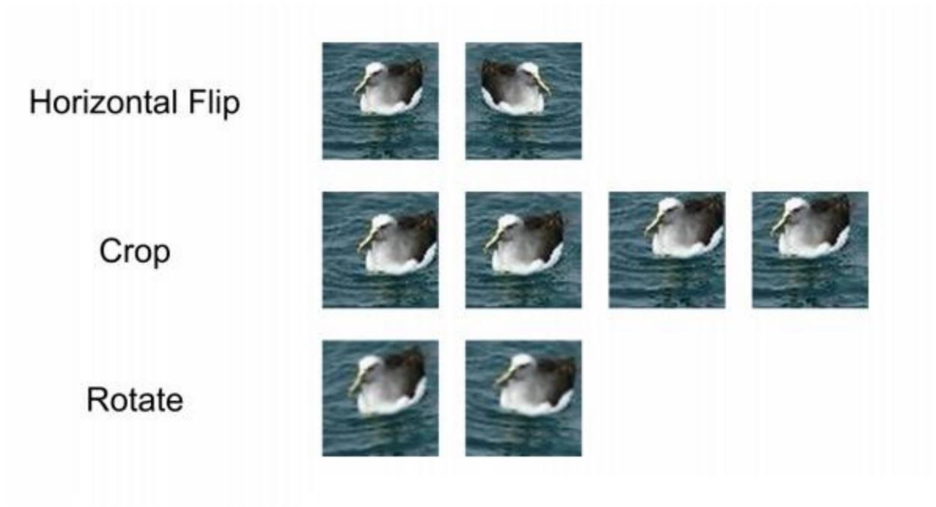
- For other models, the effect is similar, but not exactly the same

Data Augmentation

Create additional artificial samples

Yet, be careful about the transformation applied

- Example: classify ‘b’ and ‘d’
- Example: classify ‘6’ and ‘9’



Takeaway messages

What have we learned today?

- **Never** use training set to evaluate your model!
- Understand the causes of overfitting
  - Learn noise in the data
  - Unspecified behaviour between data points
- Bias-Variance tradeoff and its relation to the model complexity
- How to evaluate models
- Different regularization strategies

Self-test questions

What you should understand by now:

- Why is it a bad idea to evaluate your algorithm on the training set
- What is the difference between true and empirical risk
- The true risk can be decomposed in which parts?
- How is the bias and the variance of a learning algorithm defined and how do the contribute to the true risk?
- What is the advantage/disadvantage of k-fold CV vs. the Hold-out method?
- Why does it make sense to penalize the norm of the weight vector?
- Which norms can we use and what are the different effects?
- What is the effect of early stopping?

Appendix: Derivation of the Bias-Variance Trade-off 1/2

$$
\begin{aligned}
R(\hat{f}_{D_n}) &= \mathbb{E}_{x,y,D_n}\left[(\hat{f}_{D_n}(\boldsymbol{x}) - y)^2\right] = \mathbb{E}_{x,y,D_n}\left[\left((\hat{f}_{D_n}(\boldsymbol{x}) - \mathbb{E}_{D_n}[\hat{f}_{D_n}(\boldsymbol{x})]) + (\mathbb{E}_{D_n}[\hat{f}_{D_n}(\boldsymbol{x})] - y)\right)^2\right] \\
&= \mathbb{E}_{x,y,D_n}\Big[(\hat{f}_{D_n}(\boldsymbol{x}) - \mathbb{E}_{D_n}[\hat{f}_{D_n}(\boldsymbol{x})])^2 + (\mathbb{E}_{D_n}[\hat{f}_{D_n}(\boldsymbol{x})] - y)^2 \\
&\qquad + 2(\hat{f}_{D_n}(\boldsymbol{x}) - \mathbb{E}_{D_n}[\hat{f}_{D_n}(\boldsymbol{x})])(\mathbb{E}_{D_n}[\hat{f}_{D_n}(\boldsymbol{x})] - y)\Big] \\
&= \mathbb{E}_{x,y,D_n}\left[(\hat{f}_{D_n}(\boldsymbol{x}) - \mathbb{E}_{D_n}[\hat{f}_{D_n}(\boldsymbol{x})])^2\right] + \mathbb{E}_{x,y,D_n}\left[(\mathbb{E}_{D_n}[\hat{f}_{D_n}(\boldsymbol{x})] - y)^2\right] \\
&\qquad + 2\mathbb{E}_{x,y}\left[\underbrace{(\mathbb{E}_{D_n}[\hat{f}_{D_n}(\boldsymbol{x})] - \mathbb{E}_{D_n}[\hat{f}_{D_n}(\boldsymbol{x})])}_{=0}(\mathbb{E}_{D_n}[\hat{f}_{D_n}(\boldsymbol{x})] - y)\right]
\end{aligned}
$$

Appendix: Derivation of the Bias-Variance Trade-off 2/2

$$R(\hat{f}_{D_n}) = \underbrace{\mathbb{E}_{x,y,D_n}\left[\left(\hat{f}_{D_n}(\boldsymbol{x}) - \mathbb{E}_{D_n}[\hat{f}_{D_n}(\boldsymbol{x})]\right)^2\right]}_{\text{variance}} + \mathbb{E}_{x,y,D_n}\left[\left(\mathbb{E}_{D_n}[\hat{f}_{D_n}(\boldsymbol{x})] - y\right)^2\right]$$

2nd term:

$$\mathbb{E}_{x,y,D_n}\left[\left(\mathbb{E}_{D_n}[\hat{f}_{D_n}(\boldsymbol{x})] - y\right)^2\right] = \mathbb{E}_{x,y}\left[\left(\mathbb{E}_{D_n}[\hat{f}_{D_n}(\boldsymbol{x})] - f(\boldsymbol{x}) - \epsilon\right)^2\right]$$

$$= \underbrace{\mathbb{E}_{x,y}\left[\left(\mathbb{E}_{D_n}[\hat{f}_{D_n}(\boldsymbol{x})] - f(\boldsymbol{x})\right)^2\right]}_{\text{bias}^2} + \underbrace{\mathbb{E}_{x,y}\left[\epsilon^2\right]}_{\text{noise}} - 2\underbrace{\mathbb{E}_{x,y}\left[\epsilon(\hat{f}_{D_n}(\boldsymbol{x})] - f(\boldsymbol{x}))\right]}_{\text{0 due to zero mean i.i.d noise}}$$