

Evaluation of Machine Learning Models

Daniel Wehner, M.Sc.

SAP SE / DHBW Mannheim

Summer term 2025



Find all slides on [GitHub](#) (DaWe1992/Applied_ML_Fundamentals)

Lecture Overview

- I** Machine Learning Introduction
- II** Optimization Techniques
- III** Bayesian Decision Theory
- IV** Non-parametric Density Estimation
- V** Probabilistic Graphical Models
- VI** Linear Regression
- VII** Logistic Regression
- VIII** Deep Learning
- **IX** Evaluation
- X** Decision Trees
- XI** Support Vector Machines
- XII** Clustering
- XIII** Principal Component Analysis
- XIV** Reinforcement Learning
- XV** Advanced Regression

Agenda for this Unit

① Evaluation Methods and Data Splits

② Evaluation Metrics for Classifiers

③ Evaluation Metrics for Regressors

④ Model Selection and Model Complexity

⑤ Wrap-Up

Section:

Evaluation Methods and Data Splits

Introduction

Out-of-Sample Testing and Data Splits

Cross-Validation / LOO-Validation

Evaluation of trained Models

- ① **Validation through experts:** A domain expert checks the plausibility
 - Subjective, time-intensive, and costly
 - Often the only option
- ② **Validation on data:** Evaluate the performance on a **separate (!)** test set
 - Labeled data is scarce and could be better used for training
 - Fast and simple, no domain knowledge needed
- ③ **On-line validation:** Test the model in a field test
 - Bad models may be costly (*e. g. autonomous driving*)
 - Gives the best estimate for the overall utility



Out-of-Sample Testing

- The performance cannot be measured on the training data (**why?**)
- Usually, a portion of the available data is reserved for testing
 - $\frac{2}{3}$ for training
 - $\frac{1}{3}$ for testing (*evaluation*)
 - The model is trained on the training set and evaluated on the test set
- **Stratified splits**
 - **Random splits may be suboptimal** if the class distribution is imbalanced
 - A stratified split ensures that the proportion of each class in the original dataset is preserved in both the training and test sets



Three Splits: Train, Dev/Validation, Test

In practice it is also common to split the data into three portions:

- ① **Training set** (*used for training as before*)
- ② **Dev/Validation set**
 - Used for tuning the hyperparameters of the model
 - Using the test set for that would be cheating
- ③ **Test set**
 - The final model is tested on the test set
 - The test set is used to estimate the **generalization error** of the model

Issues with fixed Splits

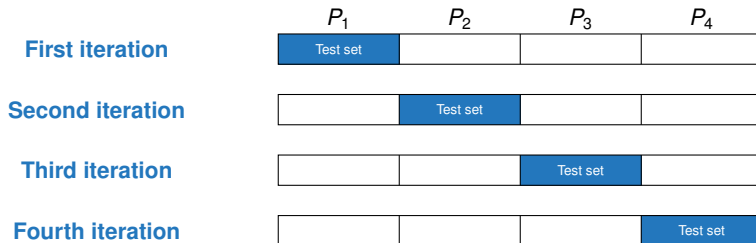
Problems:

- ① Training data is wasted for testing
- ② Labeling additional training examples may be expensive (*with respect to time and money*)

Solution: Use the cross-validation (X-Val) technique

Cross-Validation

- Also referred to as K -fold X-Val
- Split the dataset into K equally sized partitions $\mathbf{P} := \{P_1, P_2, \dots, P_K\}$
- For each partition P_k : Use $\mathbf{P} \setminus \{P_k\}$ for training and P_k for testing
- Average the resulting errors





Cross-Validation (Ctd.)

We get K trained models when using K -fold X-Val

Question: Which of these models is used in production?

Answer:

- None of these models is used
- X-Val is only used for error estimation
- The final model is trained on the whole dataset

Leave-One-Out Cross-Validation (LOO X-Val)

LOO X-Val is equal to N -fold X-Val

- We use $N - 1$ examples for training
- We use one example for testing

Advantage: LOO X-Val makes best use of the available data

Disadvantage: This technique is very expensive for large datasets

Section: Evaluation Metrics for Classifiers

Confusion Matrices
Drawback of Accuracy
Precision, Recall and F_1 -Score
ROC and AUC

Types of Errors

False positive (type I error):

A \ominus example is classified as \oplus , e. g. a non-spam e-mail is classified as spam

False negative (type II error):

An instance labeled as \oplus is classified as \ominus , e. g. a spam e-mail is not detected

Please note: The costs of false negatives and false positives can be different depending on the context!



Confusion Matrices (two Classes)

- How often is class \mathcal{C}_i confused with class \mathcal{C}_j ?
- Calculate **accuracy**:

Legend:

- tp true positive
- fn false negative
- fp false positive
- tn true negative

		Classified	
		\oplus	\ominus
Is	\oplus	#tp	#fn
	\ominus	#fp	#tn

$$\text{accuracy} = \frac{\text{\#tp} + \text{\#tn}}{\text{\#tp} + \text{\#tn} + \text{\#fp} + \text{\#fn}}$$

$$\text{error} = 1 - \text{accuracy}$$

Confusion Matrices (multiple Classes)

		Classified			
		A	B	C	Σ
Is	A	$N_{A,A}$	$N_{B,A}$	$N_{C,A}$	N_A
	B	$N_{A,B}$	$N_{B,B}$	$N_{C,B}$	N_B
	C	$N_{A,C}$	$N_{B,C}$	$N_{C,C}$	N_C
	Σ	$\overline{N_A}$	$\overline{N_B}$	$\overline{N_C}$	N

$$accuracy = \frac{N_{A,A} + N_{B,B} + N_{C,C}}{N}$$



Drawback of Accuracy

- Real-world datasets are usually **imbalanced**, i. e. some classes appear more frequently than others
- **Example:**
 - A dataset \mathcal{D} contains two classes, \mathcal{C}_1 and \mathcal{C}_2
 - \mathcal{C}_1 appears 99 % of the time, \mathcal{C}_2 only 1 % of the time
 - It is easy to reach 99 % accuracy by always predicting the majority class
 - **Is this useful?** *Probably not...*

Conclusion: We need some more sophisticated evaluation metrics!



Precision and Recall

Precision: Ratio of #tp to all instances predicted as \oplus

$$P := \frac{\text{\#tp}}{\text{\#tp} + \text{\#fp}} \quad (1)$$

Recall: Ratio of #tp to all instances actually labeled as \oplus (Sensitivity)

$$R := \frac{\text{\#tp}}{\text{\#tp} + \text{\#fn}} \quad (2)$$

Precision-Recall Trade-Off

There is a trade-off between precision and recall!

It is very easy to get 100 % precision:

- Simply classify one instance as \oplus where you are absolutely sure
- But recall is bad... (*many \oplus instances are not detected*)

It is also quite easy to achieve 100 % recall:

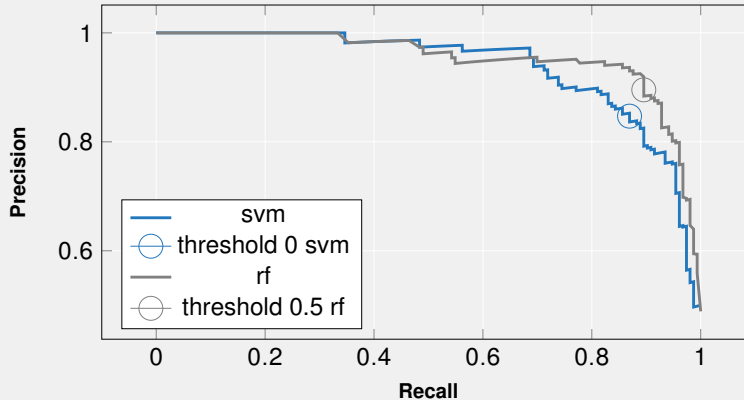
- Classify all instances as \oplus
- But precision is bad... (*many \ominus instances are classified as \oplus*)

Precision-Recall Curves / P-R Curves

- Precision-recall curves are used to visualization of the precision-recall trade-off
- Influence precision and recall by varying thresholds
- **Example:**
 - Consider a ranker, e. g. a logistic regression classifier
 - It outputs probabilities for each class
 - The threshold when to predict the positive class \oplus can be changed
 - This has an influence on precision and recall

A P-R-curve plots precision and recall for all possible thresholds

Example: Precision-Recall Curves / P-R-Curves





Precision or Recall?

Question: When to use precision, when to use recall?

Answer: This depends on the cost of fp and fn

- If fp are expensive \Rightarrow **use precision (why?)**
- If fn are expensive \Rightarrow **use recall (why?)**

Often we combine both metrics, precision and recall, to define a new metric



Combining Precision and Recall: F_1 -Score

We define the F_1 -score:

$$F_1 := \frac{2 \cdot P \cdot R}{P + R} \quad (3)$$

$$F_\beta := (1 + \beta^2) \cdot \frac{P \cdot R}{(\beta^2 \cdot P) + R} \quad (\beta \in \mathbb{R}^+) \quad (4)$$

- The F_1 -score is the harmonic mean of precision and recall
- Large values of β emphasize recall



Calculation for multiple Classes (Example Precision)

- Precision must be calculated for each class separately
- For K classes we get K results: **How to combine these results?**
 - **Macro average:** Calculate P for each class and average the results

$$P_{\text{macro}} := \frac{1}{K} \sum_{k=1}^K P_k \quad (5)$$

- **Micro average:** Sum the number of tp and fp for all classes and calculate P

$$P_{\text{micro}} := \sum_{k=1}^K \#tp_k / \sum_{k=1}^K (\#tp_k + \#fp_k) \quad (6)$$

Calculation for multiple Classes (Example Precision, Ctd.)

		Classified				
		A	B	C	D	Σ
Is	A	40	12	4	8	64
	B	7	51	2	0	60
	C	2	17	27	11	57
	D	39	4	15	8	66
	Σ	88	84	48	27	247

Calculation for multiple Classes (Example Precision, Ctd.)

$$P_A = \frac{40}{40 + 48} = 0.45$$

$$P_B = \frac{51}{51 + 33} = 0.61$$

$$P_C = \frac{27}{27 + 21} = 0.56$$

$$P_D = \frac{8}{8 + 19} = 0.30$$

$$P_{macro} = \frac{0.45 + 0.61 + 0.56 + 0.30}{4} = 0.48$$

$$P_{micro} = \frac{40 + 51 + 27 + 8}{(40 + 51 + 27 + 8) + (48 + 33 + 21 + 19)} = 0.51$$

ROC-Curves

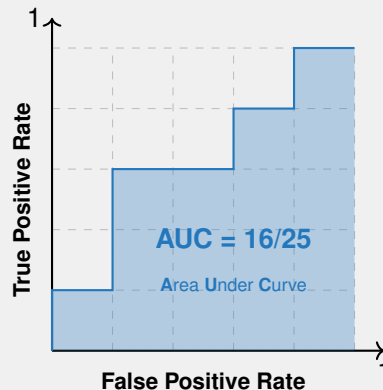
- ROC is short for **R**eciever **O**perating **C**haracteristic
- Borrowed from signal theory (*hence the name...*)
- Uses *true positive rate* (recall) and *false positive rate* $:= \frac{\#fp}{\#fp + \#tn}$

General procedure:

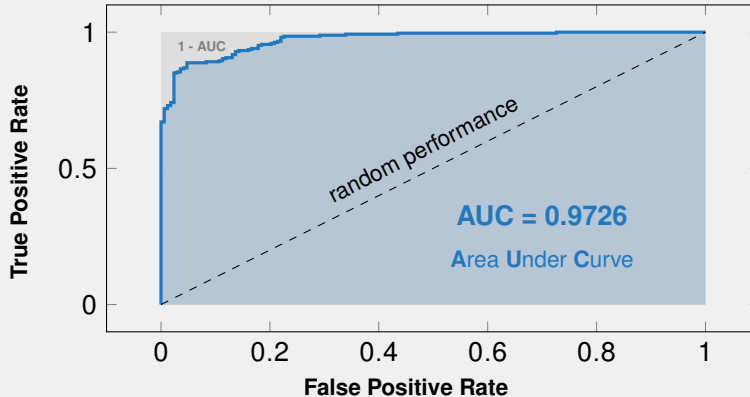
- Rank test instances by decreasing certainty of the positive class \oplus
- Start at the origin $(0, 0)$
- If the next instance in the ranking is \oplus : move $1/|\oplus|$ up
- If the next instance in the ranking is \ominus : move $1/|\ominus|$ right

Sample ROC-Curve (Example I)

Rank	Prob.	True class
1	0.95	\oplus
2	0.85	\ominus
3	0.78	\oplus
4	0.75	\oplus
5	0.62	\ominus
6	0.41	\ominus
7	0.37	\oplus
8	0.22	\ominus
9	0.15	\oplus
10	0.05	\ominus



Sample ROC-Curve (Example II)



ROC-Curve Interpretation

- The AUC can be interpreted as **the probability of a positive example always being listed before a negative example**
- A high AUC value implies a good class separation:
 - **AUC = 1.0:** All \oplus listed before all \ominus (**desiderata**)
 - **AUC = 0.5:** Random ordering (**worst case**)
 - **AUC = 0.0:** All \ominus listed before all \oplus (**not the worst case \Rightarrow invert classification**)

Please note: ROC-curves are only defined for binary classification problems
(*however, generalizations for multi-class problems exist*)

Section: Evaluation Metrics for Regressors

Coefficient of Determination, RMSE and MAE
An Example

Coefficient of Determination

- **Coefficient of determination R^2 :**

$$R^2 := \frac{\sum_{n=1}^N (h_{\theta}(\mathbf{x}^n) - \bar{y})^2}{\sum_{n=1}^N (y_n - \bar{y})^2}, \quad R^2 \in [0, 1] \quad (7)$$

- The numerator represents the variance explained by the model
- The denominator represents the total variance in the dataset

RMSE and MAE

- **Root mean square error (RMSE):**

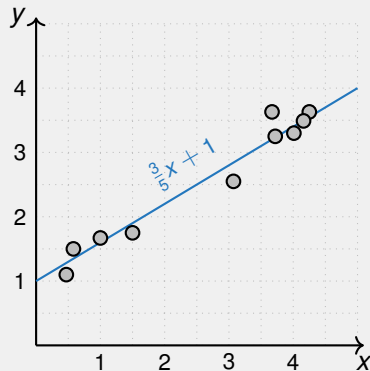
$$\text{RMSE} := \sqrt{\frac{1}{N} \sum_{n=1}^N (h_{\theta}(\mathbf{x}^n) - y_n)^2} \quad (8)$$

- **Mean absolute error (MAE):**

$$\text{MAE} := \frac{1}{N} \sum_{n=1}^N |h_{\theta}(\mathbf{x}^n) - y_n| \quad (9)$$

Example: Evaluation of Regressors

\mathbf{x}^n	y_n	$h_{\theta}(\mathbf{x}^n)$
0.47	1.10	1.28
0.58	1.50	1.35
1.00	1.67	1.60
1.50	1.75	1.90
3.07	2.55	2.84
3.67	3.63	3.20
3.72	3.25	3.23
4.01	3.30	3.41
4.16	3.49	3.50
4.25	3.63	3.55
$\bar{y} = 2.59$		



Example: Evaluation of Regressors (Ctd.)

- Coefficient of determination:

$$\begin{aligned} R^2 &= \frac{(1.28 - 2.59)^2 + \dots + (3.55 - 2.59)^2}{(1.10 - 2.59)^2 + \dots + (3.63 - 2.59)^2} \\ &= \frac{7.97}{8.89} = \mathbf{0.90} \end{aligned}$$

Example: Evaluation of Regressors (Ctd.)

- Root mean square error:

$$\text{RMSE} = \sqrt{\frac{1}{10} [(1.28 - 1.10)^2 + \dots + (3.55 - 3.63)^2]} = \mathbf{0.19}$$

- Mean absolute error:

$$\text{MAE} = \frac{1}{10} (|1.28 - 1.10| + \dots + |3.55 - 3.63|) = \mathbf{0.15}$$

Section:

Model Selection and Model Complexity

Hyperparameter Tuning: Grid Search and Random Search
Bias and Variance / Bias-Variance Decomposition

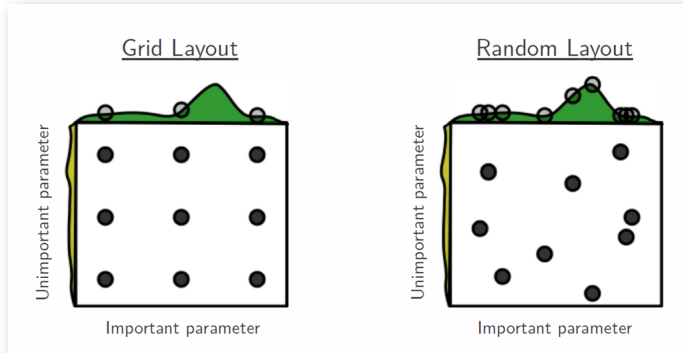
Hyperparameter Tuning with Grid Search

- **Grid search** is applied to find **optimal hyperparameter settings**
- Hyperparameter tuning should be done on the `dev` set
- We have to specify the search space / ranges of hyperparameter values
- Grid search will try **all combinations** to find the best model

Disadvantages of grid search:

- 1 It is **computationally very expensive** \Rightarrow `Scikit-learn` provides parameters to parallelize the search: `n_jobs=-1` to use all cores
- 2 It may not find the optimal setting \Rightarrow **Use random search!**

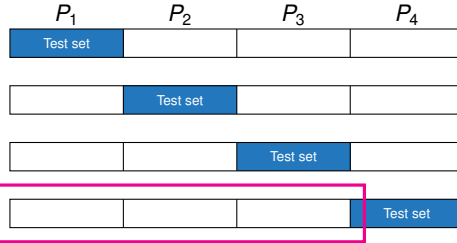
Grid Search vs. random Search



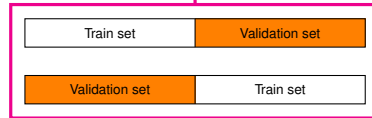
cf. BERGSTRÄ/BENGIO.2012, <https://www.jmlr.org/papers/volume13/bergstra12a/bergstra12a.pdf>, page 284

Nested Cross-Validation

Outer Loop
(Average test scores)



Inner Loop
(Hyperparameter tuning)



Bias-Variance Decomposition for MSE

- Suppose that we have a training set $\mathcal{D} := \{(x_1, y_1), \dots, (x_N, y_N)\}$
- There is an **unknown** function $f(x)$ such that

$$y = f(x) + \varepsilon \quad (10)$$

- The noise term ε has zero mean and variance σ_ε^2 , i. e. $\mathbb{E}\{\varepsilon\} = 0$ and $\mathbb{V}\{\varepsilon\} = \sigma_\varepsilon^2$
- Based on the dataset \mathcal{D} we want to find a function $\hat{f}(x; \mathcal{D})$ which approximates $f(x)$ as well as possible
- For this we try to minimize the mean squared error $\mathbb{E}_{\mathcal{D}, \varepsilon} \left\{ (y - \hat{f}(x; \mathcal{D}))^2 \right\}$

Bias-Variance Decomposition for MSE (Ctd.)

- We can decompose the **expected error** of $\hat{f}(x, \mathcal{D})$ into bias and variance
- We use $\mathbb{B}_{\mathcal{D}}$ to denote the **bias** (*expected deviation of the model from the true function*), and $\mathbb{V}_{\mathcal{D}}$ to denote the **variance** of the model \hat{f}
- The variance of the noise term σ_{ε}^2 is **irreducible**

Bias-Variance decomposition of the mean squared error:

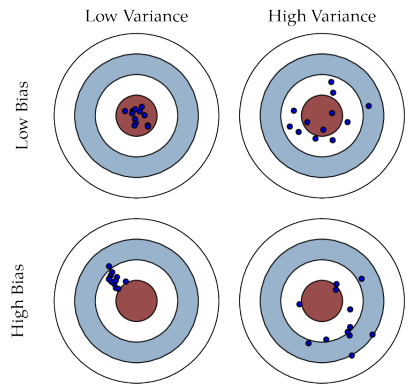
$$\mathbb{E}_{\mathcal{D}, \varepsilon} \left\{ (y - \hat{f}(x; \mathcal{D}))^2 \right\} = \mathbb{B}_{\mathcal{D}}^2 \{ \hat{f}(x; \mathcal{D}) \} + \mathbb{V}_{\mathcal{D}} \{ \hat{f}(x; \mathcal{D}) \} + \sigma_{\varepsilon}^2 \quad (11)$$



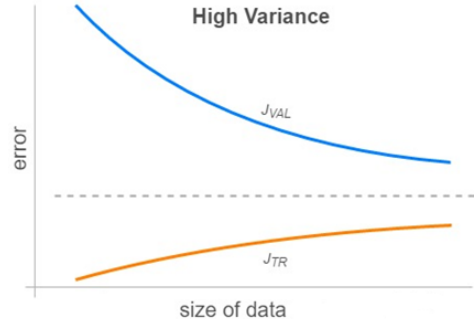
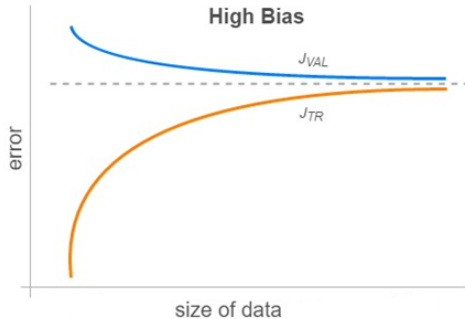
Bias and Variance

The **bias** results from erroneous assumptions in the learning algorithm. High bias can cause an algorithm to miss the relevant relations between features and target outputs (**underfitting**).

The **variance** is an error from sensitivity to small fluctuations in the training set. High variance may result from an algorithm modeling the random noise in the training data (**overfitting**).



Bias and Variance (Ctd.)





Recall: Expectation and Variance

- Let \mathcal{X} be a random variable. $\Omega(\mathcal{X})$ is the domain of \mathcal{X} (*i. e. the set of possible values \mathcal{X} can take*)
- Definition of **expectation** (*discrete case*):

$$\mathbb{E}\{\mathcal{X}\} := \sum_{k \in \Omega(\mathcal{X})} k \cdot p(\mathcal{X} = k) \quad (12)$$

- Definition of **variance** (*discrete case*):

$$\mathbb{V}\{\mathcal{X}\} := \sum_{k \in \Omega(\mathcal{X})} (k - \mathbb{E}\{\mathcal{X}\})^2 \cdot p(\mathcal{X} = k) \quad (13)$$



Recall: Expectation and Variance (Ctd.)

Let \mathcal{X} and \mathcal{Y} be random variables and $a, b \in \mathbb{R}$:

- **Linearity** of \mathbb{E} : **(very important!)**

$$\mathbb{E}\{a\mathcal{X} + b\mathcal{Y}\} = a\mathbb{E}\{\mathcal{X}\} + b\mathbb{E}\{\mathcal{Y}\} \quad (14)$$

- If \mathcal{X} and \mathcal{Y} are independent: $\mathbb{E}\{\mathcal{X}\mathcal{Y}\} = \mathbb{E}\{\mathcal{X}\}\mathbb{E}\{\mathcal{Y}\}$
- $\mathbb{V}\{\mathcal{X}\} = \mathbb{E}\{\mathcal{X} - \mathbb{E}\{\mathcal{X}\}\}^2 = \mathbb{E}\{\mathcal{X}^2\} - \mathbb{E}\{\mathcal{X}\}^2$
- \mathbb{V} is **not** linear: $\mathbb{V}\{a + b\mathcal{X}\} = b^2\mathbb{V}\{\mathcal{X}\}$
- However, if \mathcal{X} and \mathcal{Y} are uncorrelated: $\mathbb{V}\{\mathcal{X} + \mathcal{Y}\} = \mathbb{V}\{\mathcal{X}\} + \mathbb{V}\{\mathcal{Y}\}$



Derivation of the Bias-Variance Decomposition for MSE

- **Notation:** In the following we will drop the dependence on the dataset to simplify the notation, i. e. we write \hat{f} instead of $\hat{f}(x; \mathcal{D})$
- The MSE is given by

$$\begin{aligned}\text{MSE} &:= \mathbb{E}\{(y - \hat{f})^2\} = \mathbb{E}\{y^2 - 2y\hat{f} + \hat{f}^2\} \\ &= \underbrace{\mathbb{E}\{y^2\}}_{\text{①}} - 2 \underbrace{\mathbb{E}\{y\hat{f}\}}_{\text{②}} + \underbrace{\mathbb{E}\{\hat{f}^2\}}_{\text{③}}\end{aligned}\tag{15}$$



Derivation of the Bias-Variance Decomposition for MSE (Ctd.)

We rewrite term **1**:

$$\mathbb{E}\{y^2\} = \mathbb{E}\{(f + \varepsilon)^2\}$$

Definition of y

$$= \mathbb{E}\{f^2\} + 2\mathbb{E}\{f\varepsilon\} + \mathbb{E}\{\varepsilon^2\}$$

Linearity of \mathbb{E}

$$= f^2 + 2f\mathbb{E}\{\varepsilon\} + \mathbb{E}\{\varepsilon^2\}$$

f does not depend on \mathcal{D}

$$= f^2 + \sigma_\varepsilon^2$$

$$\mathbb{E}\{\varepsilon\} = 0 \quad (16)$$



Derivation of the Bias-Variance Decomposition for MSE (Ctd.)

We rewrite term ②:

$$\mathbb{E}\{y\hat{f}\} = \mathbb{E}\{(f + \varepsilon)\hat{f}\}$$

Definition of y

$$= f\mathbb{E}\{\hat{f}\} + \mathbb{E}\{\varepsilon\hat{f}\}$$

Linearity of \mathbb{E}

$$= f\mathbb{E}\{\hat{f}\} + \mathbb{E}\{\varepsilon\}\mathbb{E}\{\hat{f}\}$$

\hat{f} and ε are independent

$$= f\mathbb{E}\{\hat{f}\}$$

$$\mathbb{E}\{\varepsilon\} = 0 \quad (17)$$



Derivation of the Bias-Variance Decomposition for MSE (Ctd.)

Term ③ is straight-forward:

$$\begin{aligned}\mathbb{E}\{\widehat{f}^2\} &= \mathbb{E}\{\widehat{f}^2\} - \mathbb{E}^2\{\widehat{f}\} + \mathbb{E}^2\{\widehat{f}\} \\ &= \mathbb{V}\{\widehat{f}\} + \mathbb{E}^2\{\widehat{f}\}\end{aligned}\tag{18}$$



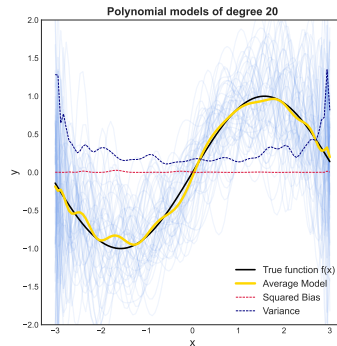
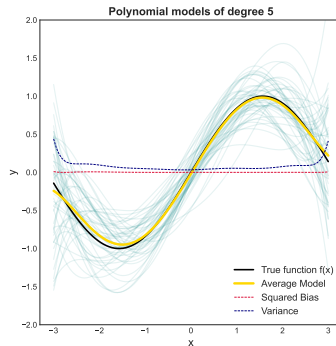
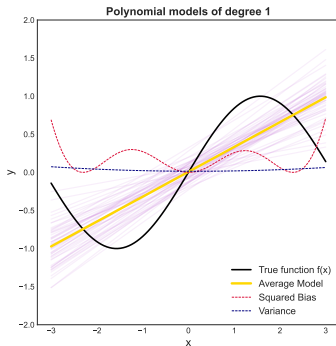
Derivation of the Bias-Variance Decomposition for MSE (Ctd.)

- Let us now plug these results into our MSE definition:

$$\begin{aligned}\text{MSE} &= \mathbb{E}\{y^2\} - 2\mathbb{E}\{y\hat{f}\} + \mathbb{E}\{\hat{f}^2\} \\ &= f^2 + \sigma_\varepsilon^2 - 2f\mathbb{E}\{\hat{f}\} + \mathbb{V}\{\hat{f}\} + \mathbb{E}^2\{\hat{f}\} \\ &= (f - \mathbb{E}\{\hat{f}\})^2 + \mathbb{V}\{\hat{f}\} + \sigma_\varepsilon^2 \\ &= \mathbb{E}^2\{(f - \hat{f})\} + \mathbb{V}\{\hat{f}\} + \sigma_\varepsilon^2 \\ &= \mathbb{B}^2\{\hat{f}\} + \mathbb{V}\{\hat{f}\} + \sigma_\varepsilon^2\end{aligned}\tag{19}$$

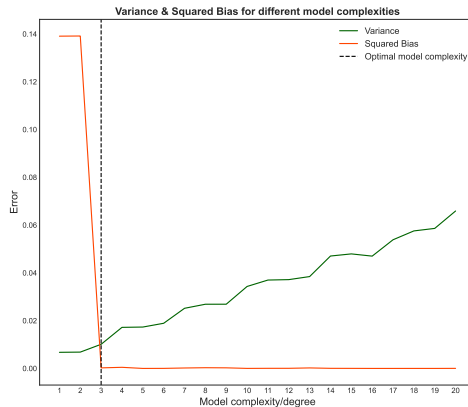
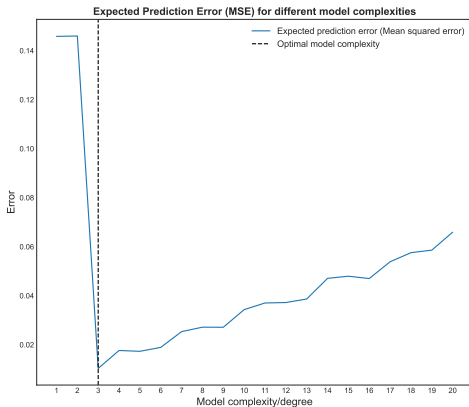
Visualization: Bias-Variance Decomposition

Polynomial models of different degrees fit on random data

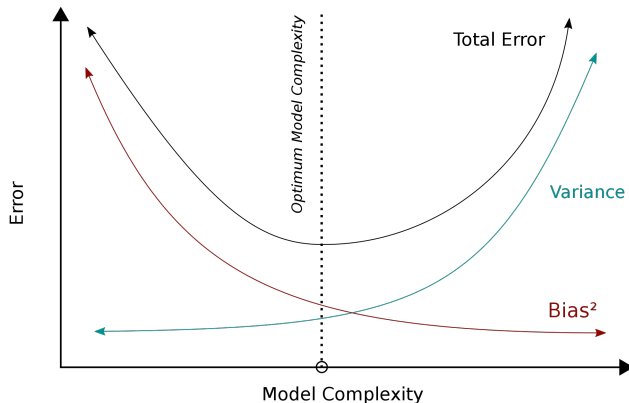




Visualization: Bias-Variance Decomposition (Ctd.)

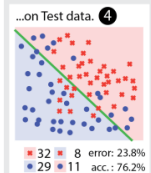


Bias, Variance, and Model Complexity



Use early stopping!

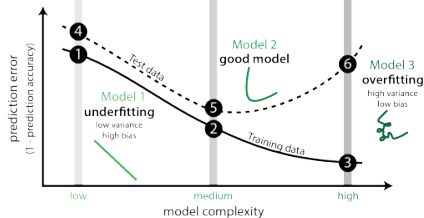
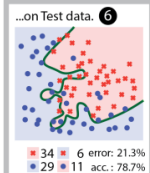
Model 1...



Model 2...



Model 3...



Section: Wrap-Up

Summary
Recommended Literature
Self-Test Questions
Lecture Outlook

Summary

- **Out-of-sample testing:** Split data into `train`, `dev` and `test` sets
- Cross-validation makes **maximum use of the data**
- Confusion matrices reveal **which classes are frequently confused**
- Precision, recall, and F_1 -score are **more robust w. r. t. imbalanced datasets**
- ROC curves are used for the evaluation of rankers
- Hyperparameters are optimized using **grid search** or **random search**
- Keep the **bias-variance trade-off** in mind! We can decompose the error into bias and variance

Recommended Literature

1 Evaluation metrics

- [MURPHY.2012], chapter 5.7.2,
pages 182 – 186

2 Bias-Variance decomposition

- [BISHOP.2006], chapter 3.2,
pages 147 – 152

(For free PDF versions, see list in GitHub readme!)



Self-Test Questions

- 1 Why should you split the data into `train`, `dev` and `test` sets?
- 2 You perform 10-fold cross validation. How many models do you have to learn?
Which one do you use in production?
- 3 What is the problem with accuracy?
- 4 Why do we apply the harmonic mean to compute the F_1 -score?
- 5 Your model produces an AUC value of 0. What does this mean?
- 6 Random search is usually preferred to optimize hyperparameters. Why?
- 7 Your model does not perform well due to its high bias. Your boss suggests adding more training examples. How would you respond?

What's next...?

- | | | | |
|-------------|-----------------------------------|-------------|------------------------------|
| I | Machine Learning Introduction | IX | Evaluation |
| II | Optimization Techniques | • X | Decision Trees |
| III | Bayesian Decision Theory | XI | Support Vector Machines |
| IV | Non-parametric Density Estimation | XII | Clustering |
| V | Probabilistic Graphical Models | XIII | Principal Component Analysis |
| VI | Linear Regression | XIV | Reinforcement Learning |
| VII | Logistic Regression | XV | Advanced Regression |
| VIII | Deep Learning | | |

Thank you very much for the attention!

***** Artificial Intelligence and Machine Learning *****

Topic: Evaluation of Machine Learning Models

Term: Summer term 2025

Contact:

Daniel Wehner, M.Sc.

SAP SE / DHBW Mannheim

daniel.wehner@sap.com

Do you have any questions?