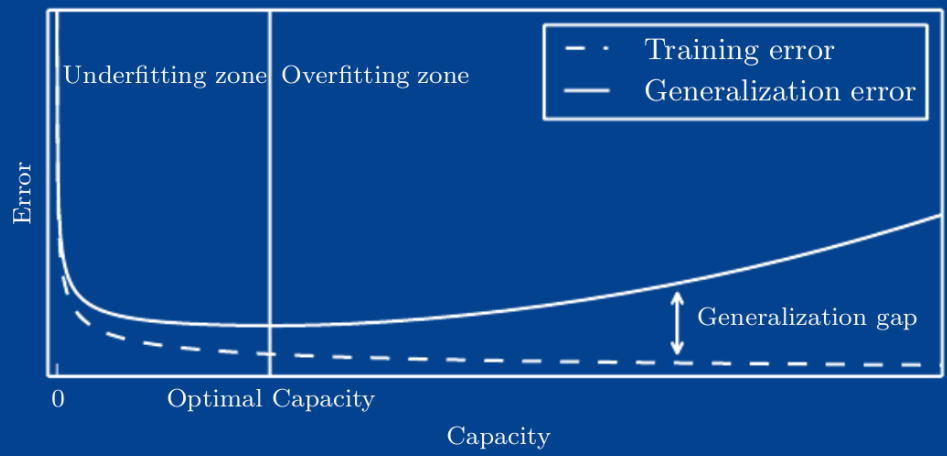
AARHUS
UNIVERSITY
SCHOOL OF ENGINEERING

# LESSON 7: Koncepts II

## capacity, under- and overfit, generalization

CARSTEN EIE FRIGAARD

AUTUMN 2019



# Supergruppe diskussion

Diskussion ml. Grp A og B:

- ca. 15 min: Grp A genfortæller første halvdel af § 2
  - ”Look at the Big Picture”,
  - ”Get the Data”,

    eksklusiv ”Create the Workspace” og ”Download the Data”,
  - ”Discover and Visualize the Data to Gain Insights”,
  - ”Prepare the Data for Machine Learning Algorithms”,
  - ”Select and Train a Model”,
- ca 15 min: herefter byttes og Grp B genfortæller..
  - ”Fine-Tune Your Model”,
  - ”Launch, Monitor, and Maintain Your System”,
  - ”Try It Out!”.

Inddrag og relater til jeres eget slut-projekt,
Inddrag The Map for Supervised learning
Session I: 8:30..9:15
Session II: 9:15..10:00

# The Map

*model selection*

# Pipelines

## Brief intro to Scikit-learn pipelines..

### Python code from `capacity_under_overfitting.ipynb`

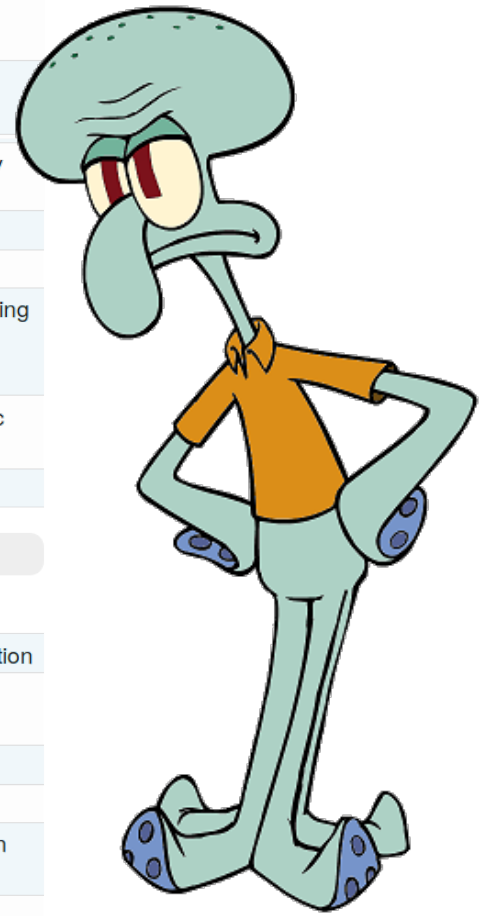
```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import cross_val_score

..

polynomial_features = PolynomialFeatures(degree=degrees[i], ..
linear_regression = LinearRegression()

pipeline = Pipeline([
        ("polynomial_features", polynomial_features),
        ("linear_regression", linear_regression)
    ])
pipeline.fit(X[:, np.newaxis], y)

scores = cross_val_score(pipeline, X[:, np.newaxis], y, scoring=
    "neg_mean_squared_error", cv=10)
..
score_mean = -scores.mean()
```

# RESUMÉ: L02/performance_metrics.ipynb

## Classification metrics

See the Classification metrics section of the user guide for further details.

| | |
|---|---|
| **metrics.accuracy_score** (y_true, y_pred[, …]) | Accuracy classification score. |
| **metrics.auc** (x, y[, reorder]) | Compute Area Under the Curve (AUC) using the trapezoidal rule |
| **metrics.average_precision_score** (y_true, y_score) | Compute average precision (AP) from prediction scores |
| **metrics.cohen_kappa_score** (y1, y2[, labels, …]) | Cohen's kappa: a statistic that measures inter-annotator agreement. |
| **metrics.confusion_matrix** (y_true, y_pred[, …]) | Compute confusion matrix to evaluate the accuracy of a classification |
| **metrics.f1_score** (y_true, y_pred[, labels, …]) | Compute the F1 score, also known as balanced F-score or F-measure |
| **metrics.log_loss** (y_true, y_pred[, eps, …]) | Log loss, aka logistic loss or cross-entropy loss. |
| **metrics.precision_score** (y_true, y_pred[, …]) | Compute the precision |
| **metrics.recall_score** (y_true, y_pred[, …]) | Compute the recall |
| **metrics.roc_auc_score** (y_true, y_score[, …]) | Compute Area Under the Receiver Operating Characteristic Curve (ROC AUC) from prediction scores. |
| **metrics.roc_curve** (y_true, y_score[, …]) | Compute Receiver operating characteristic (ROC) |
| **metrics.zero_one_loss** (y_true, y_pred[, …]) | Zero-one classification loss. |



## Regression metrics

See the Regression metrics section of the user guide for further details.

| | |
|---|---|
| **metrics.explained_variance_score** (y_true, y_pred) | Explained variance regression score function |
| **metrics.max_error** (y_true, y_pred) | max_error metric calculates the maximum residual error. |
| **metrics.mean_absolute_error** (y_true, y_pred) | Mean absolute error regression loss |
| **metrics.mean_squared_error** (y_true, y_pred[, …]) | Mean squared error regression loss |
| **metrics.mean_squared_log_error** (y_true, y_pred) | Mean squared logarithmic error regression loss |
| **metrics.median_absolute_error** (y_true, y_pred) | Median absolute error regression loss |
| **metrics.r2_score** (y_true, y_pred[, …]) | R^2 (coefficient of determination) regression |

# Model capacity

Exercise: `capacity_under_overfitting.ipynb`

Dummy and Paradox classifier:

*capacity fixed* $\sim 0$, cannot generalize at all!

Linear regression for a polynomial model:

*capacity* $\sim$ degree of the polynomial, $x^n$

Neural Network model:

*capacity* $\propto$ number of neurons/layers

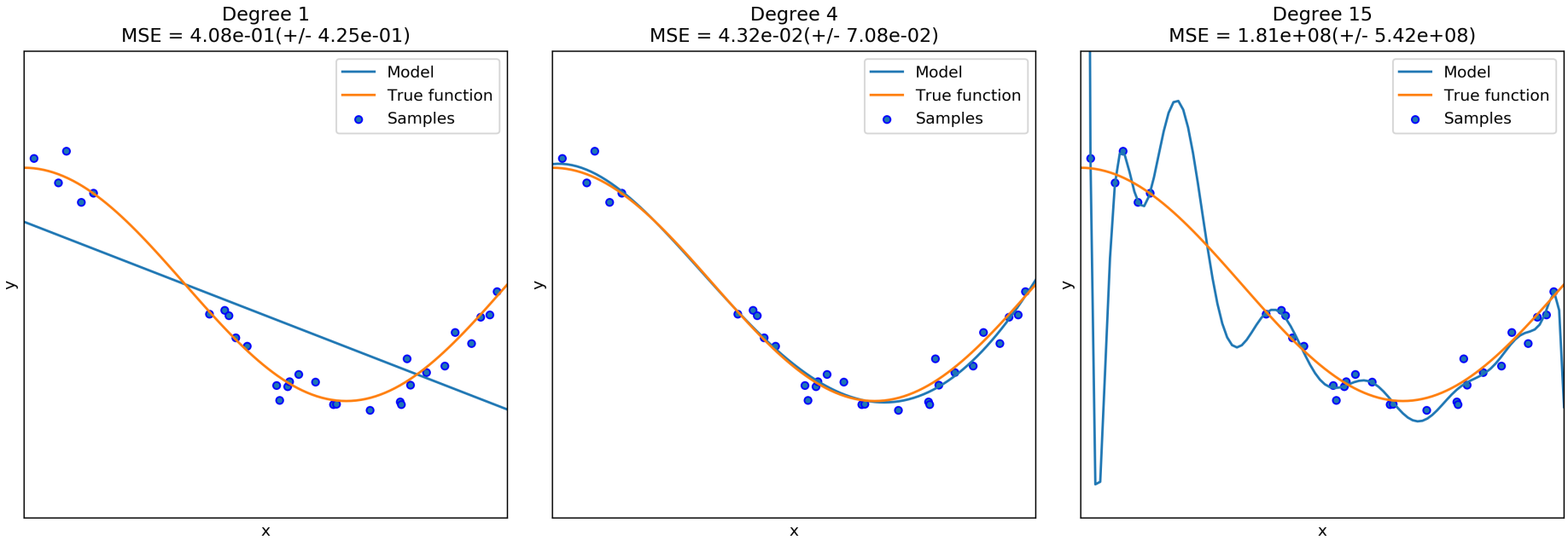
$\Rightarrow$ **Capacity** can be hard to express as a quantity for some models, but you need to choose..

$\Longrightarrow$ how to choose the **optimal** *capacity*?

# Under- and overfitting

Exercise: `capacity_under_overfitting.ipynb`

Polynomial linear reg. fit for underlying model: `cos(x)`




- underfitting: capacity of model too low,
- overfitting: capacity to high.

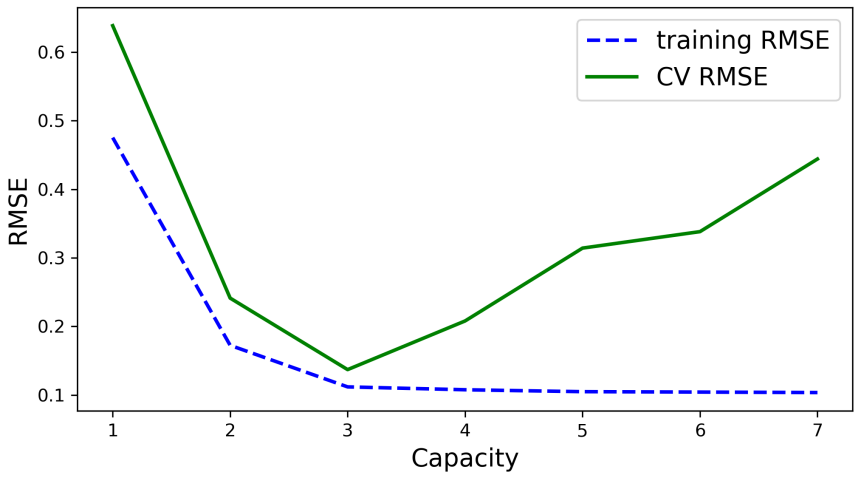
$\Longrightarrow$ how to choose the **optimal** capacity?

# Generalization Error

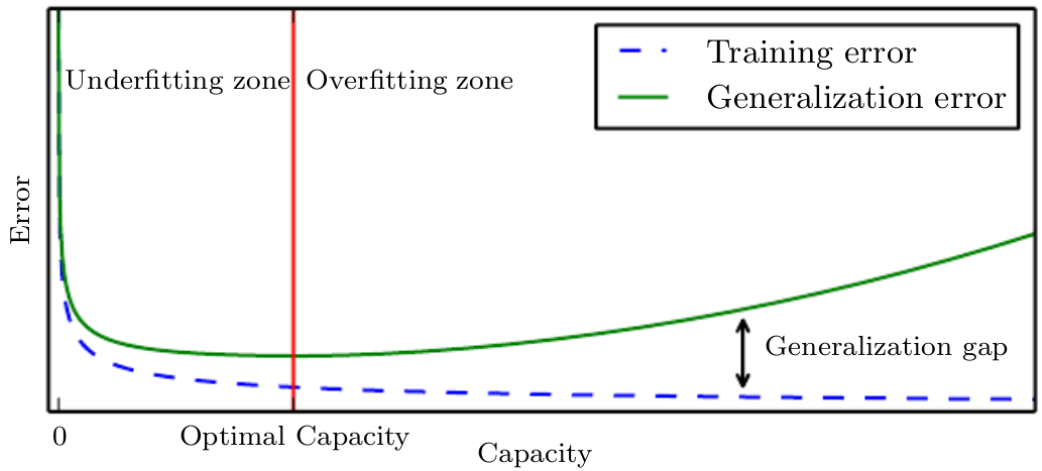
Exercise: `generalization_error.ipynb`

RMSE-capacity plot for lin. reg. with polynomial features

(capacity $\sim$ degree of poly)




(Figure 5.3 from [DL])




Inspecting the plots from the exercise (`.ipynb`) and [DL], extracting the concepts:

- training/generalization error,
- generalization gab,
- underfit/overfit zone,
- optimal capacity (best-model, early stop),
- (and the two axes: x/capacity, y/error.)

# Generalization Error

Exercise: `generalization_error.ipynb`

NOTE: three methods/plots:

i) via **learning curves** as in [HOML],
ii) via an **error-capacity** plot as in [GITHOLM] and[DL],
ii) via an **error-epoch** plot as in [GITHOML].