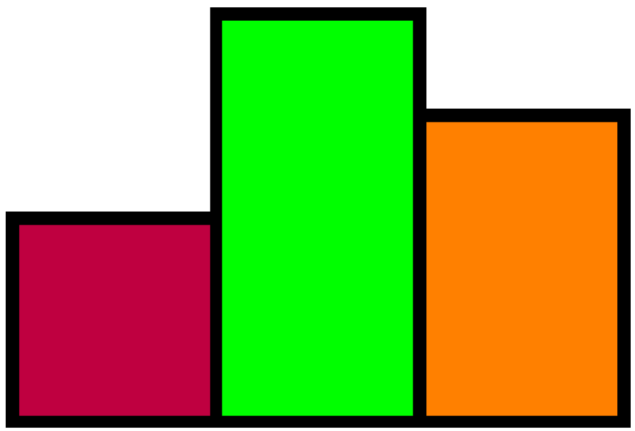
# Introduction to Machine Learning

# Evaluation: Introduction and Remarks

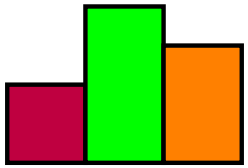


**Learning goals**

- Understand the goal of performance estimation
- Understand the difference between outer and inner loss
- Know the definition of generalization error

# PERFORMANCE EVALUATION

How well does my model perform...



... on data from the same data-generating process?

In practice:

... on current data (training data)?
... on new data (test data)?
... based on a certain measure/metric?
...

# PERFORMANCE EVALUATION

ML performance evaluation provides clear and simple protocols for reliable model validation.

- Often simpler than classical statistical model diagnosis
- Relies only on few assumptions
- Still hard enough and offers **lots** of options to cheat / make mistakes

# PERFORMANCE MEASURES

We measure performance using a statistical estimator for the **generalization error** (GE).

GE $=$ expected loss of a fixed model

$\hat{\text{GE}}$ $=$ average loss

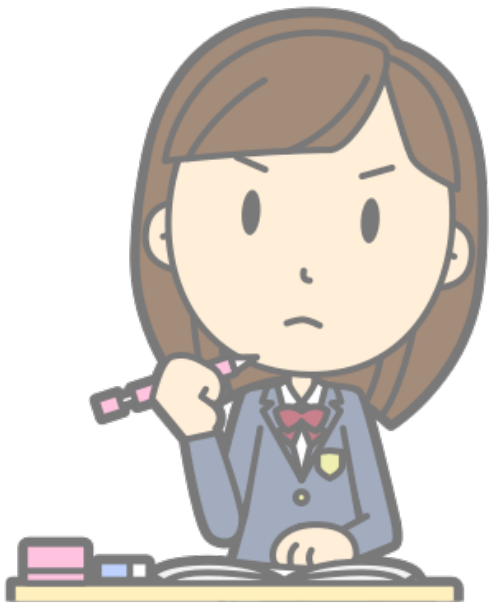
Example: Mean squared error (L2 loss)

$$\hat{\text{GE}} = MSE = \frac{1}{n}\sum_{i=1}^{n}(y^{(i)} - \hat{y}^{(i)})^2$$

# MEASURES: INNER VS. OUTER LOSS

Inner loss = loss used in learning

Outer loss = loss used in evaluation
= evaluation measure



# MEASURES: INNER VS. OUTER LOSS

Optimally: inner loss = outer loss

Not always possible:
some losses are hard to optimize / no loss is specified directly

Example:

Logistic Regression $\rightarrow$ minimize binomial loss
kNN $\rightarrow$ no explicit loss minimization

- When evaluating the models we might be interested in (cost-weighted) classification error
- Or some of the more advanced measures from ROC analysis like AUC

# Einführung in das Statistische Lernen

# Evaluation: Measures for Regression

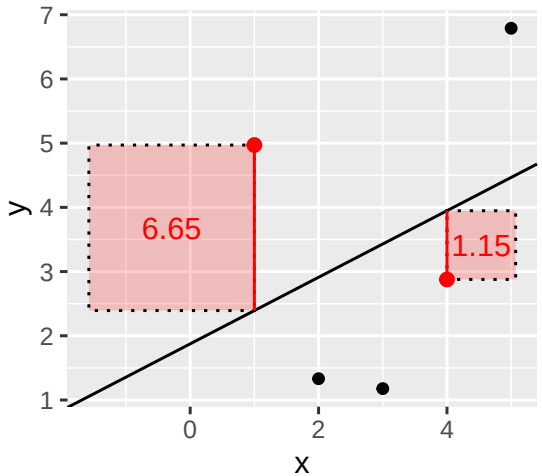
**Learning goals**

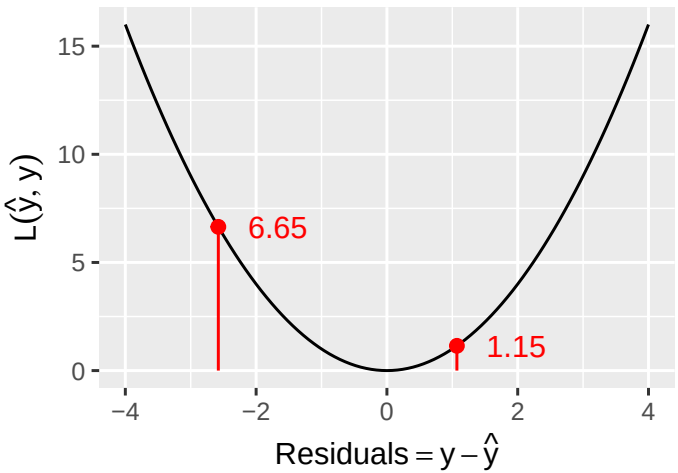
- Know the definitions of mean squared error (MSE) and mean absolute error (MAE)
- Understand the connections of MSE and MAE to L2 and L1 loss
- Know the definitions of $R^2$ and generalized $R^2$

# MEAN SQUARED ERROR

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (y^{(i)} - \hat{y}^{(i)})^2 \in [0; \infty) \qquad \rightarrow \text{L2 loss.}$$

Single observations with a large prediction error heavily influence the **MSE**, as they enter quadratically.



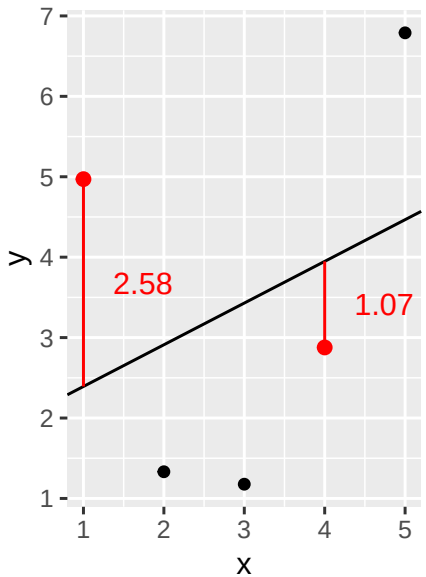



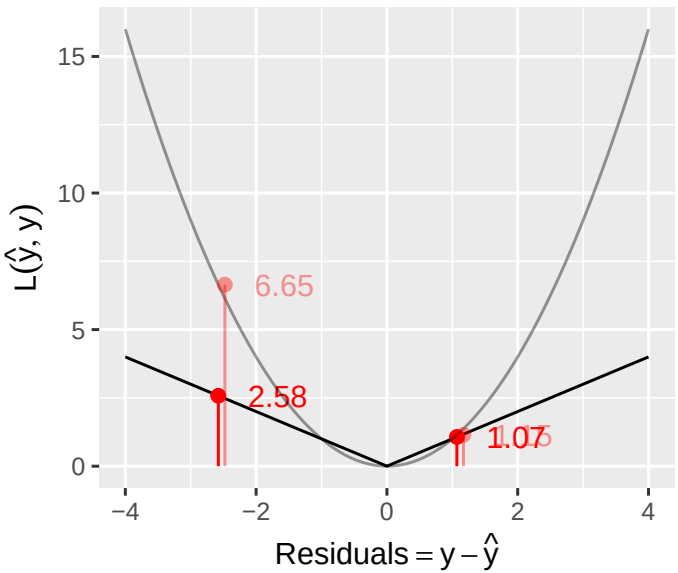

Similar measures: sum of squared errors (SSE), root mean squared error (RMSE, brings measurement back to the original scale of the outcome).

# MEAN ABSOLUTE ERROR

$$MAE = \frac{1}{n} \sum_{i=1}^{n} |y^{(i)} - \hat{y}^{(i)}| \in [0; \infty) \qquad \rightarrow \text{L1 loss.}$$

Less influenced by large errors and maybe more intuitive than the MSE.




Similar measures: median absolute error (for even more robustness).

# $R^2$

Well-known measure from statistics.

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (y^{(i)} - \hat{y}^{(i)})^2}{\sum_{i=1}^{n} (y^{(i)} - \bar{y})^2} = 1 - \frac{SSE_{LinMod}}{SSE_{Intercept}}$$

- Usually introduced as *fraction of variance explained* by the model
- Simpler: compares SSE of constant model (baseline) with complex model (LM)
- $R^2 = 1$: all residuals are 0, we predict perfectly,
  $R^2 = 0$: we predict as badly as the constant model
- If measured on the training data, $R^2 \in [0; 1]$ (LM must be at least as good as the constant)
- On other data $R^2$ can even be negative as there is no guarantee that the LM generalizes better than a constant (overfitting)

# GENERALIZED $R^2$ FOR ML

A simple generalization of $R^2$ for ML seems to be:

$$1 - \frac{Loss_{ComplexModel}}{Loss_{SimplerModel}}$$

- Works for arbitrary measures (not only SSE), for arbitrary models, on any data set of interest
- E.g. model vs constant, LM vs non-linear model, tree vs forest, model without some features vs model with them included
- Fairly unknown; our terminology (generalized $R^2$) is non-standard

# Einführung in das Statistische Lernen

# Evaluation: Simple Measures for Classification

| | | **True Class** $y$ | |
|---|---|---|---|
| | | $+$ | $-$ |
| **Pred.** | $+$ | True Positive (TP) | False Positive (FP) |
| $\hat{y}$ | $-$ | False Negative (FN) | True Negative (TN) |

## Learning goals

- Know the definitions of misclassification error rate (MCE) and accuracy (ACC)
- Understand the entries of a confusion matrix
- Understand the idea of costs
- Know defintions of Brier score and log loss

# LABELS VS PROBABILITIES

In classification we predict:

1. Class labels $\rightarrow \hat{h}(\mathbf{x}) = \hat{y}$
2. Class probabilities $\rightarrow \hat{\pi}_k(\mathbf{x})$

$\rightarrow$ We evaluate based on those

# LABELS: MCE

The misclassification error rate (MCE) counts the number of incorrect predictions and presents them as a rate:

$$MCE = \frac{1}{n} \sum_{i=1}^{n} [y^{(i)} \neq \hat{y}^{(i)}] \in [0; 1]$$

Accuracy is defined in a similar fashion for correct classifications:

$$ACC = \frac{1}{n} \sum_{i=1}^{n} [y^{(i)} = \hat{y}^{(i)}] \in [0; 1]$$

- If the data set is small this can be brittle
- The MCE says nothing about how good/skewed predicted probabilities are
- Errors on all classes are weighted equally (often inappropriate)

# LABELS: CONFUSION MATRIX

True classes in columns.
Predicted classes in rows.

| | setosa | versicolor | virginica | -err.- | -n- |
|---|---|---|---|---|---|
| setosa | 50 | 0 | 0 | 0 | 50 |
| versicolor | 0 | 46 | 4 | 4 | 50 |
| virginica | 0 | 4 | 46 | 4 | 50 |
| -err.- | 0 | 4 | 4 | 8 | NA |
| -n- | 50 | 50 | 50 | NA | 150 |

We can see class sizes (predicted and true) and where errors occur.

# LABELS: CONFUSION MATRIX

## In binary classification

| | | **True Class** $y$ | |
|---|---|---|---|
| | | $+$ | $-$ |
| **Pred.** | $+$ | True Positive (TP) | False Positive (FP) |
| $\hat{y}$ | $-$ | False Negative (FN) | True Negative (TN) |

# LABELS: COSTS

We can also assign different costs to different errors via a cost matrix.

$$Costs = \frac{1}{n} \sum_{i=1}^{n} C[y^{(i)}, \hat{y}^{(i)}]$$

Example:
Predict if person has a ticket (yes / no).
Should train conductor check ticket of a person?

**Costs:**

| | |
|---|---|
| Ticket checking: | 3 EUR |
| Fee for fare-dodging: | 40 EUR |

http://www.oslobilder.no/OMU/OB.%C3%9864/2902

# LABELS: COSTS

Predict if person has a ticket (yes / no).

```
Cost matrix C
     predicted
true   no yes
  no  -37   0
  yes   3   0
```

```
Confusion matrix
     predicted
true  no yes
  no   7   0
  yes 93   0
```

```
Confusion matrix * C
     predicted
true    no yes
  no  -259   0
  yes  279   0
```

**Costs:**

| | |
|---|---|
| Ticket checking: | 3 EUR |
| Fee for fare-dodging: | 40 EUR |

Our model says that we should not trust anyone and check the tickets of all passengers.

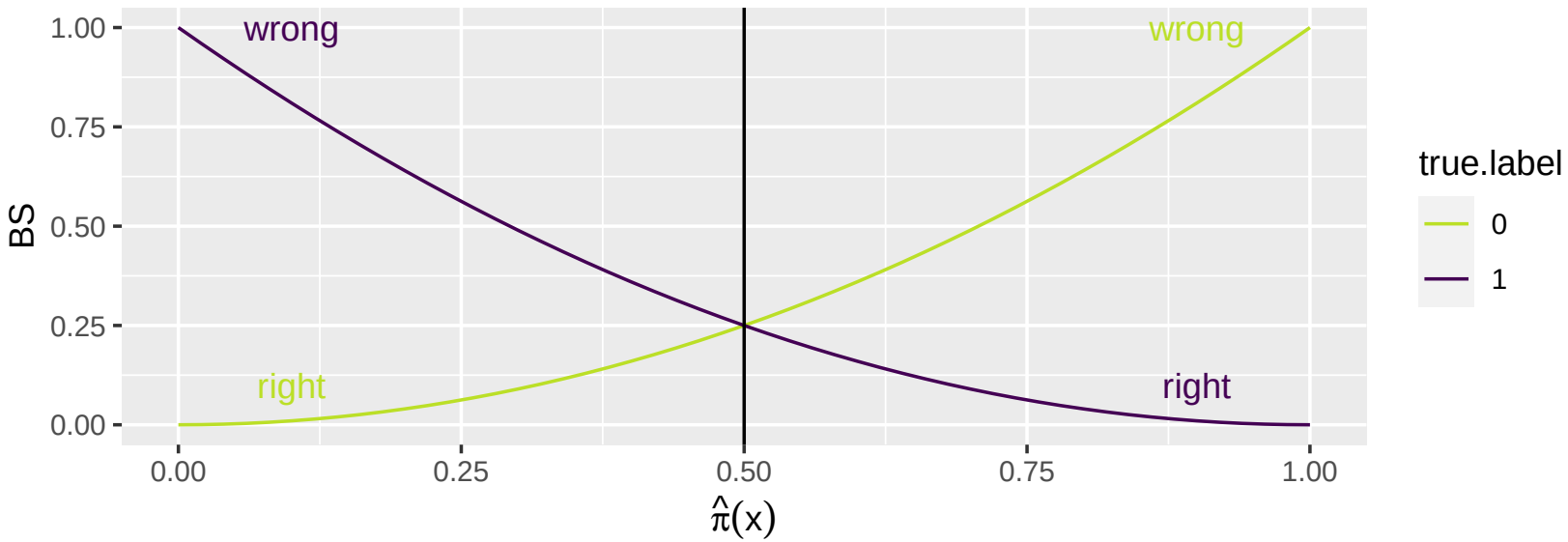
$$
\begin{aligned}
\mathit{Costs} &= \frac{1}{n} \sum_{i=1}^{n} C[y^{(i)}, \hat{y}^{(i)}] \\
&= \frac{1}{100} \left(-37 \cdot 7 + 0 \cdot 0 + 3 \cdot 93 + 0 \cdot 0\right) \\
&= \frac{20}{100} = 0.2
\end{aligned}
$$

# PROBABILITIES: BRIER SCORE

Measures squared distances of probabilities from the true class labels:

$$BS1 = \frac{1}{n} \sum_{i=1}^{n} \left( \hat{\pi}(\mathbf{x}^{(i)}) - y^{(i)} \right)^2$$

- Fancy name for MSE on probabilities
- Usual definition for binary case, $y^{(i)}$ must be coded as 0 and 1.



# PROBABILITIES: BRIER SCORE

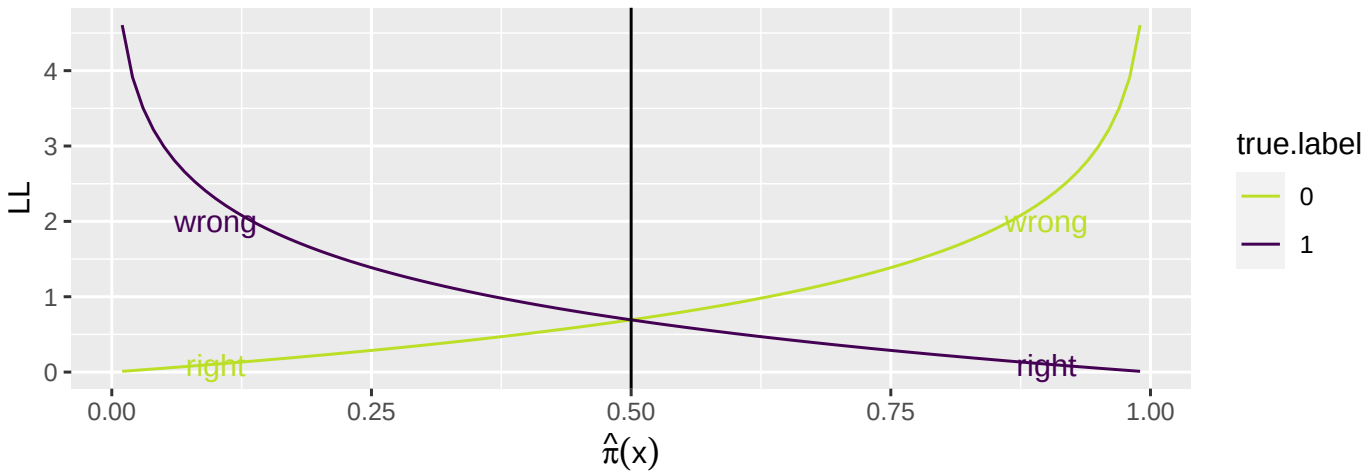
$$BS2 = \frac{1}{n} \sum_{i=1}^{n} \sum_{k=1}^{g} \left( \hat{\pi}_k(\mathbf{x}^{(i)}) - o_k^{(i)} \right)^2$$

- Original by Brier, works also for multiple classes
- $o_k^{(i)} = [y^{(i)} = k]$ is a 0-1-one-hot coding for labels
- For the binary case, BS2 is twice as large as BS1, because in BS2 we sum the squared difference for each observation regarding class 0 **and** class 1, not only the true class.

# PROBABILITIES: LOG-LOSS

Logistic regression loss function, a.k.a. Bernoulli or binomial loss, $y^{(i)}$ coded as 0 and 1.

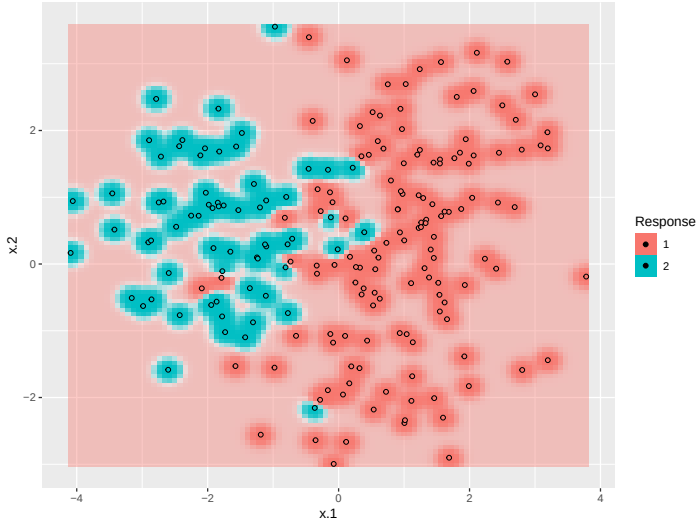
$$LL = \frac{1}{n} \sum_{i=1}^{n} \left( -y^{(i)} \log(\hat{\pi}(\mathbf{x}^{(i)})) - (1 - y^{(i)}) \log(1 - \hat{\pi}(\mathbf{x}^{(i)})) \right)$$




- Optimal value is 0, “confidently wrong” is penalized heavily
- Multiclass version: $LL = -\frac{1}{n} \sum_{i=1}^{n} \sum_{k=1}^{g} o_k^{(i)} \log(\hat{\pi}_k(\mathbf{x}^{(i)}))$

# Introduction to Machine Learning

# Evaluation: Overfitting

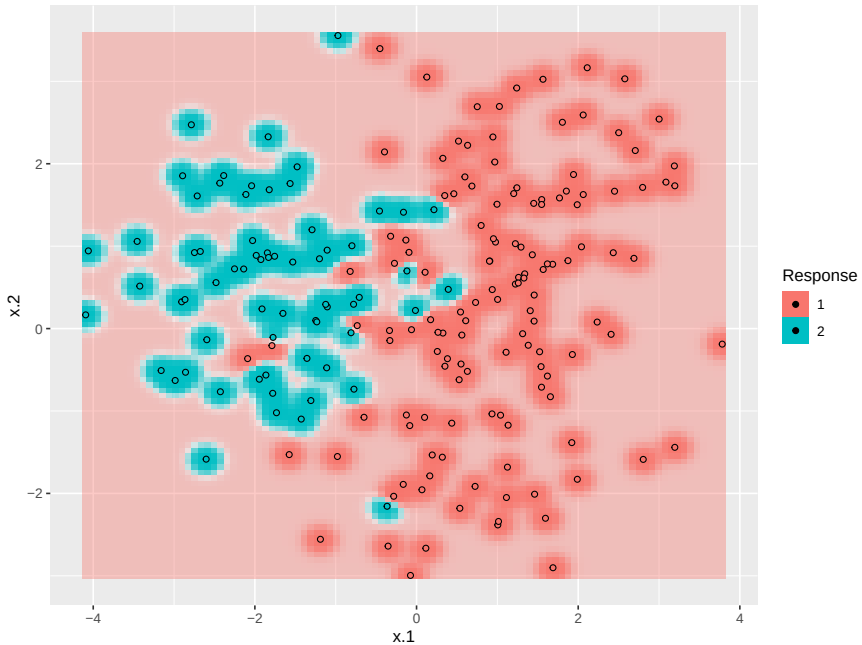



## Learning goals

- Understand what overfitting is and why it is a problem
- Understand how to avoid overfitting

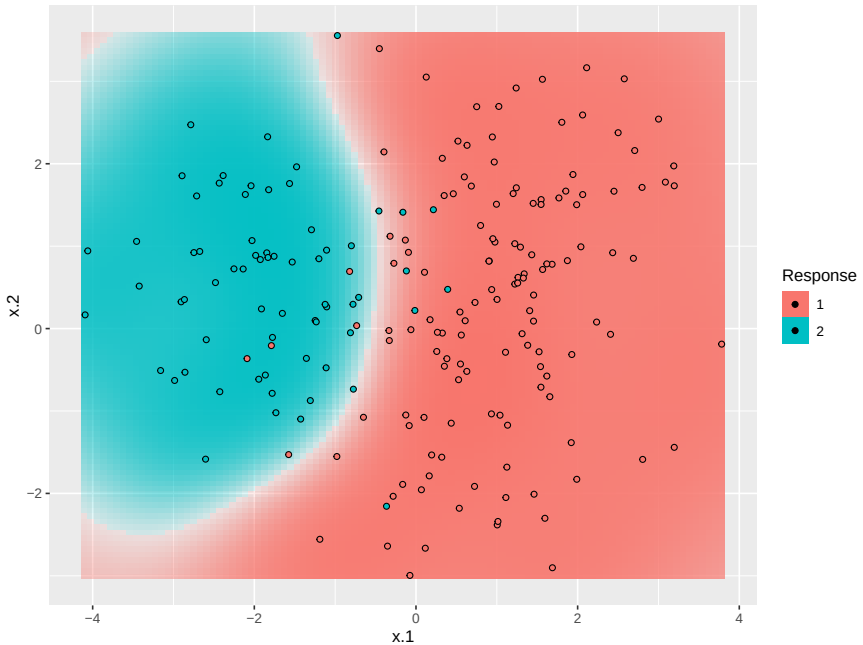
# OVERFITTING

Overfitting learner




Better training set performance (seen examples)

Non-overfitting learner

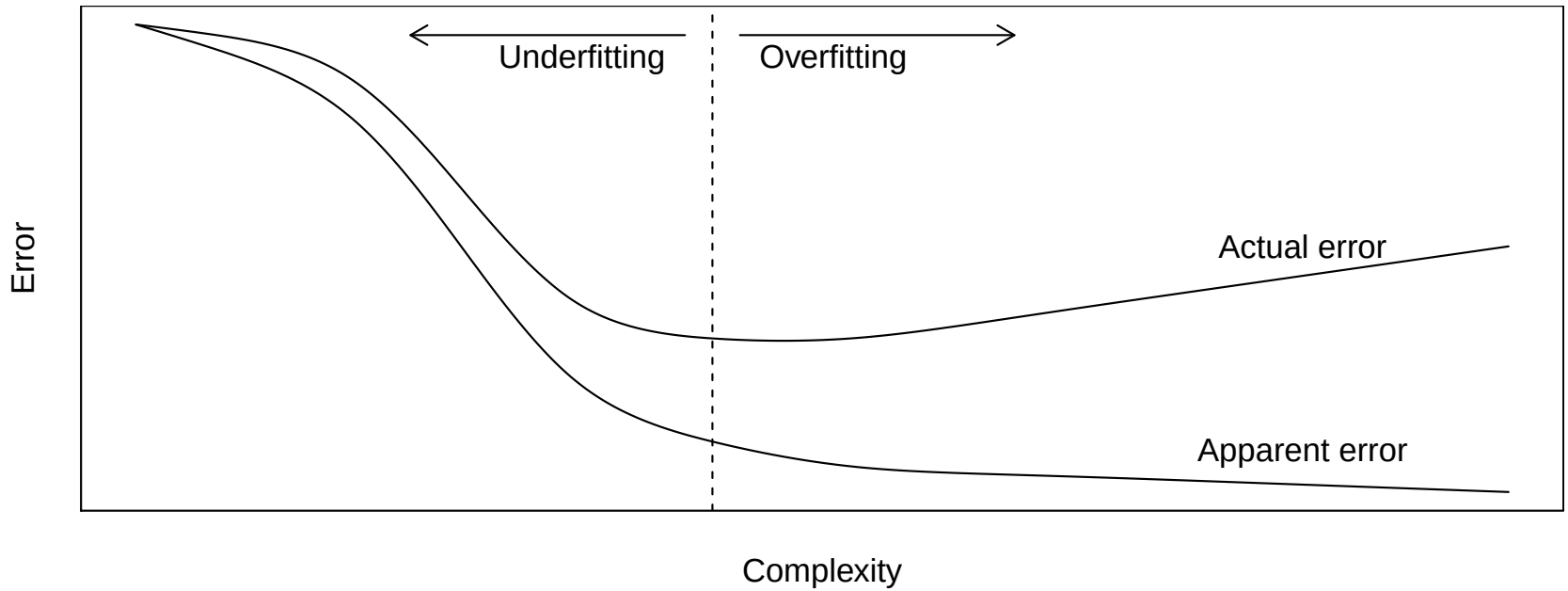



Better test set performance (unseen examples)

# OVERFITTING

- Happens when algorithm models patterns beyond the data-generating process, e.g., noise or artefacts in the training data
- Reason: too many hypotheses and not enough data to tell them apart
- Less in bigger data sets
- If hypothesis space is not constrained, there may never be enough data
- Many learners have a parameter that allows constraining (*regularization*)
- Check for overfitting by validating on a new unseen test data set

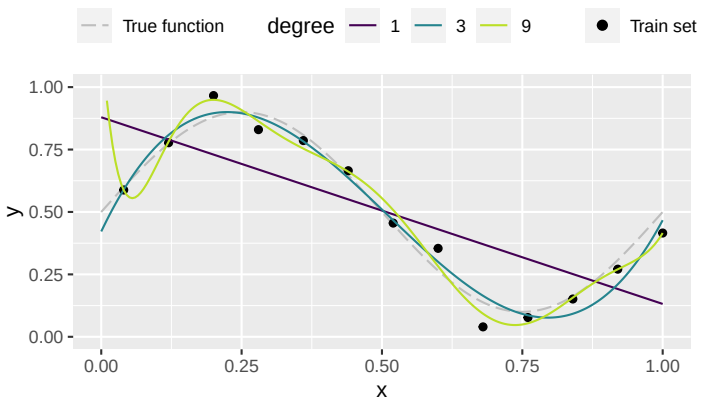
# TRADE-OFF BETWEEN GENERALIZATION ERROR AND COMPLEXITY




$\Rightarrow$ Optimization regarding model complexity is desirable:
Find the right amount of complexity for the given amount of data where generalization error becomes minimal.

# Introduction to Machine Learning

# Evaluation: Training Error

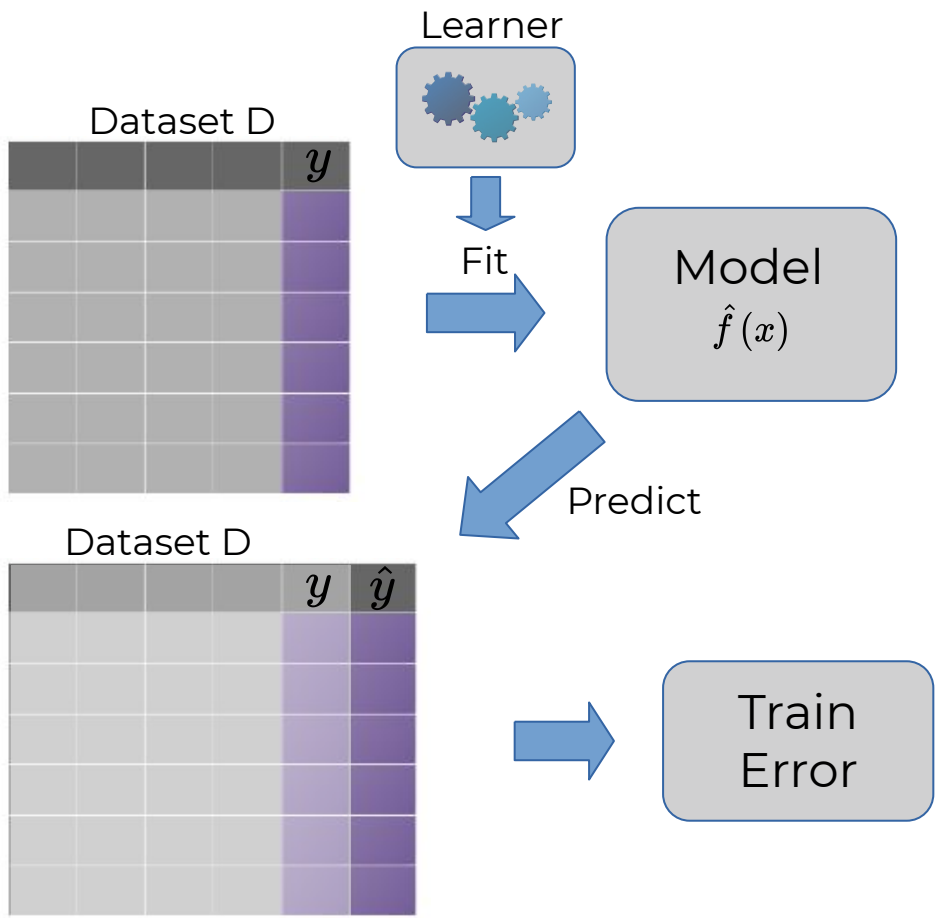



## Learning goals

- Understand the definition of training error
- Understand why training error is no reliable estimator of future performance

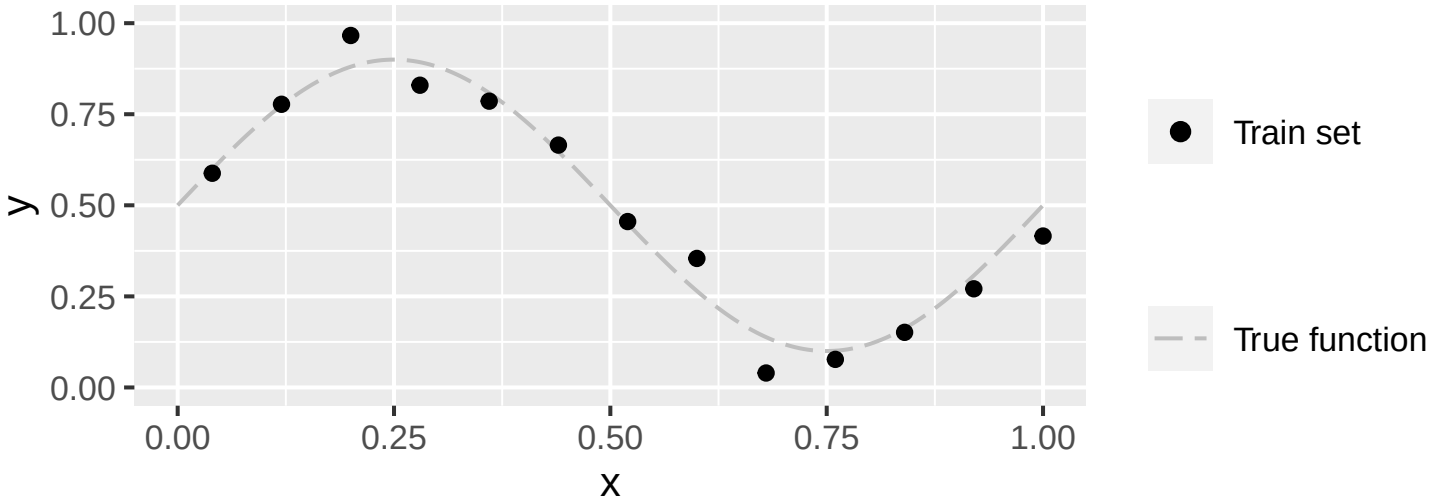
# TRAINING ERROR

(also: apparent error / resubstitution error)



# EXAMPLE: POLYNOMIAL REGRESSION

Sample data from sinusoidal function $0.5 + 0.4 \cdot \sin(2\pi x) + \epsilon$ with measurement error $\epsilon$.

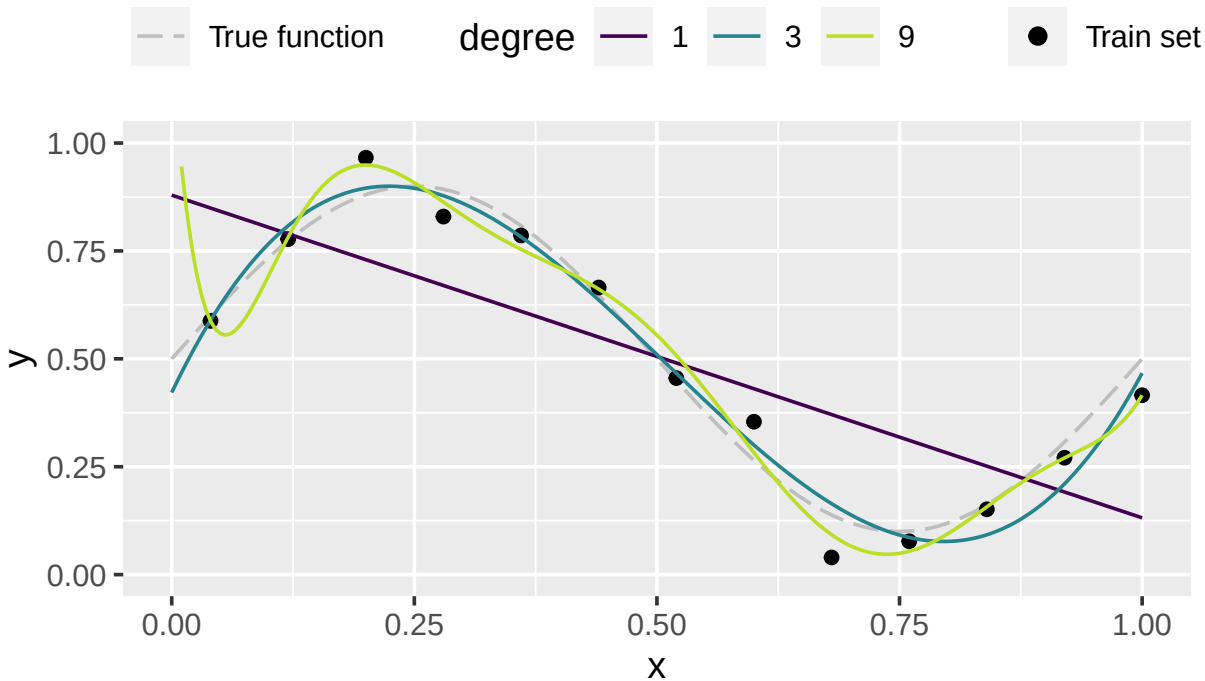



Assume data-generating process unknown.
Try to approximate with a $d^{th}$-degree polynomial:

$$f(\mathbf{x} \mid \boldsymbol{\theta}) = \theta_0 + \theta_1 x + \cdots + \theta_d x^d = \sum_{j=0}^{d} \theta_j x^j.$$

# EXAMPLE: POLYNOMIAL REGRESSION

Models of different *complexity*, i.e., of different orders of the polynomial are fitted. How should we choose $d$?




- d=1: MSE = 0.036: Clear underfitting
- d=3: MSE = 0.003: Pretty OK?
- d=9: MSE = 0.001: Clear overfitting

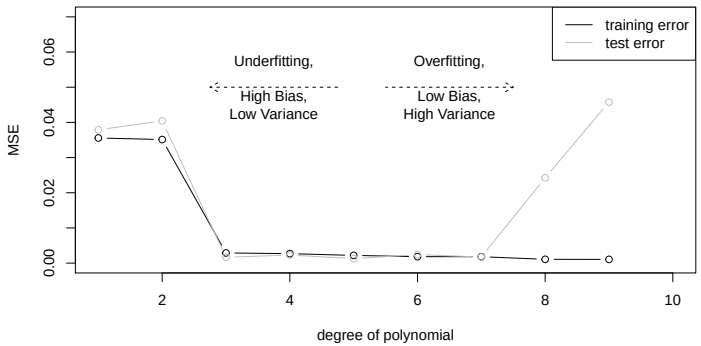
Simply using the training error seems to be a bad idea.

# TRAINING ERROR PROBLEMS

- Unreliable and overly optimistic estimator of future performance.
  E.g., training error of 1-NN is always zero as each observation is its own NN during test time.
- Goodness-of-fit measures like (classic) $R^2$, likelihood, AIC, BIC, deviance are all based on the training error.
- For models of restricted capacity, and given enough data, the training error may provide reliable information.
  E.g., LM with $p = 5$ features, $10^6$ training points.
  But: impossible to determine when training error becomes unreliable.

# Introduction to Machine Learning

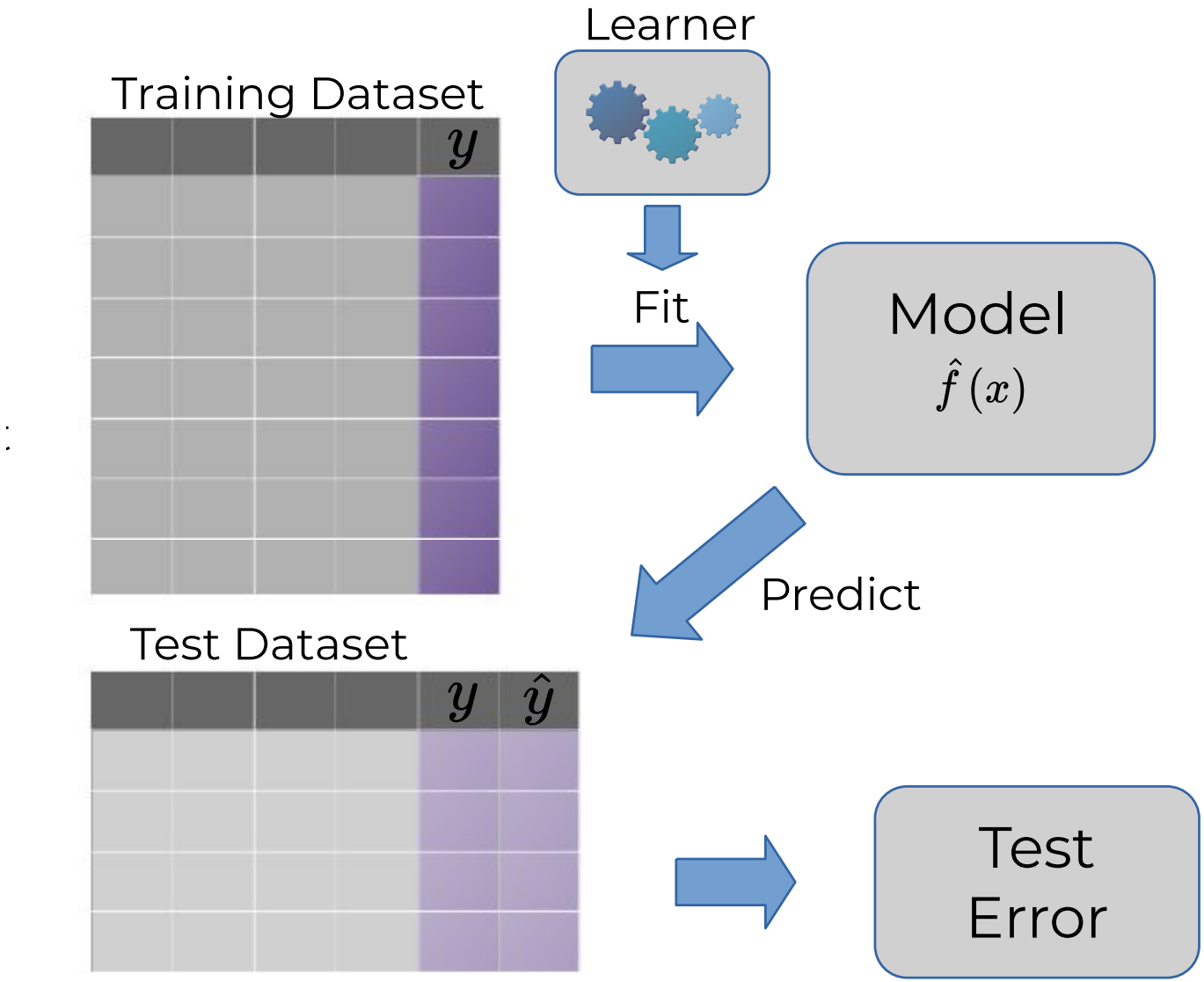
# Evaluation: Test Error




## Learning goals

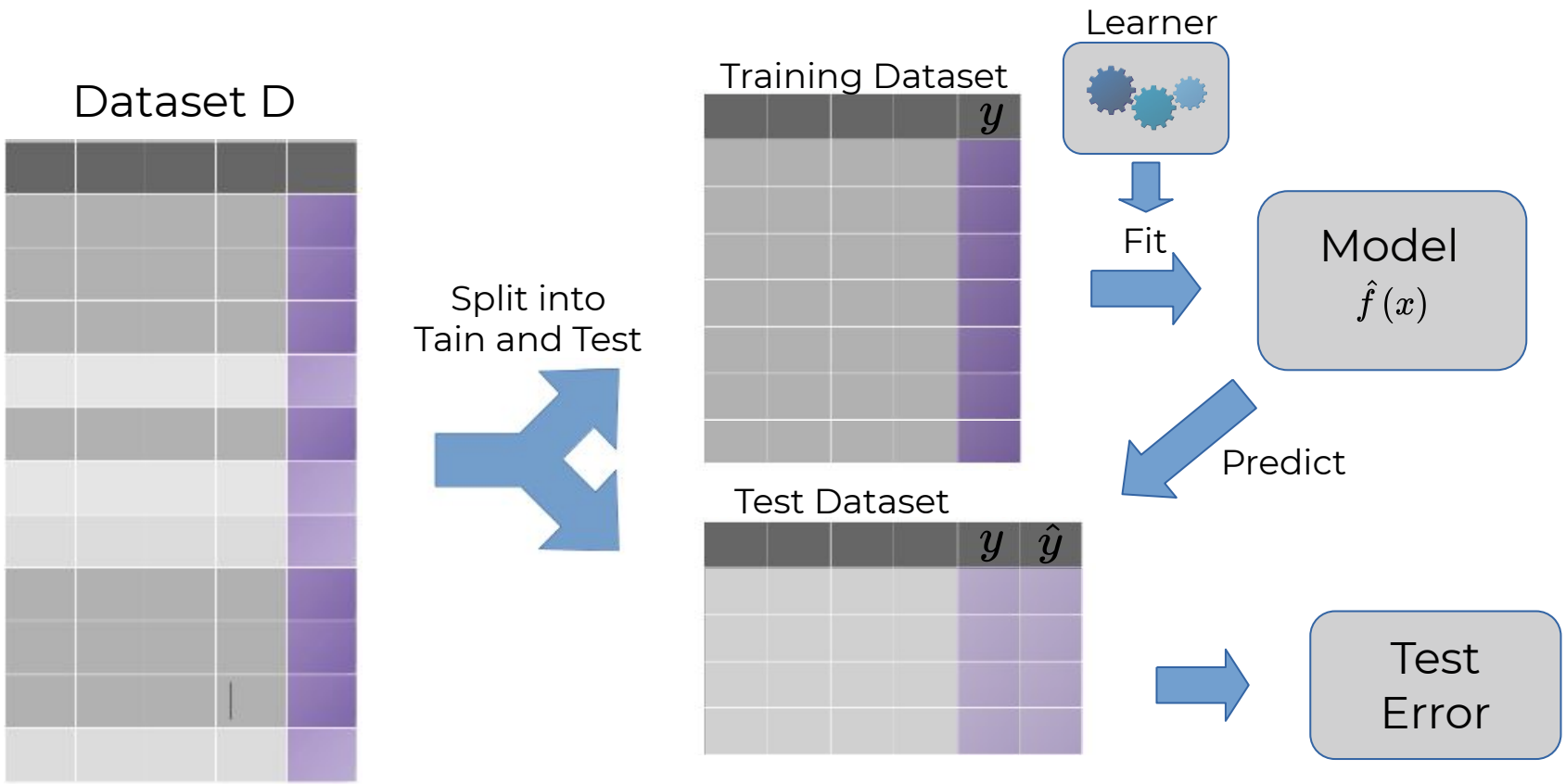
- Understand the definition of test error
- Understand how overfitting can be seen in the test error

# TEST ERROR


Learner
Training Dataset
$y$
Fit
Model
$\hat{f}(x)$
Predict
Test Dataset
$y$
$\hat{y}$
Test
Error

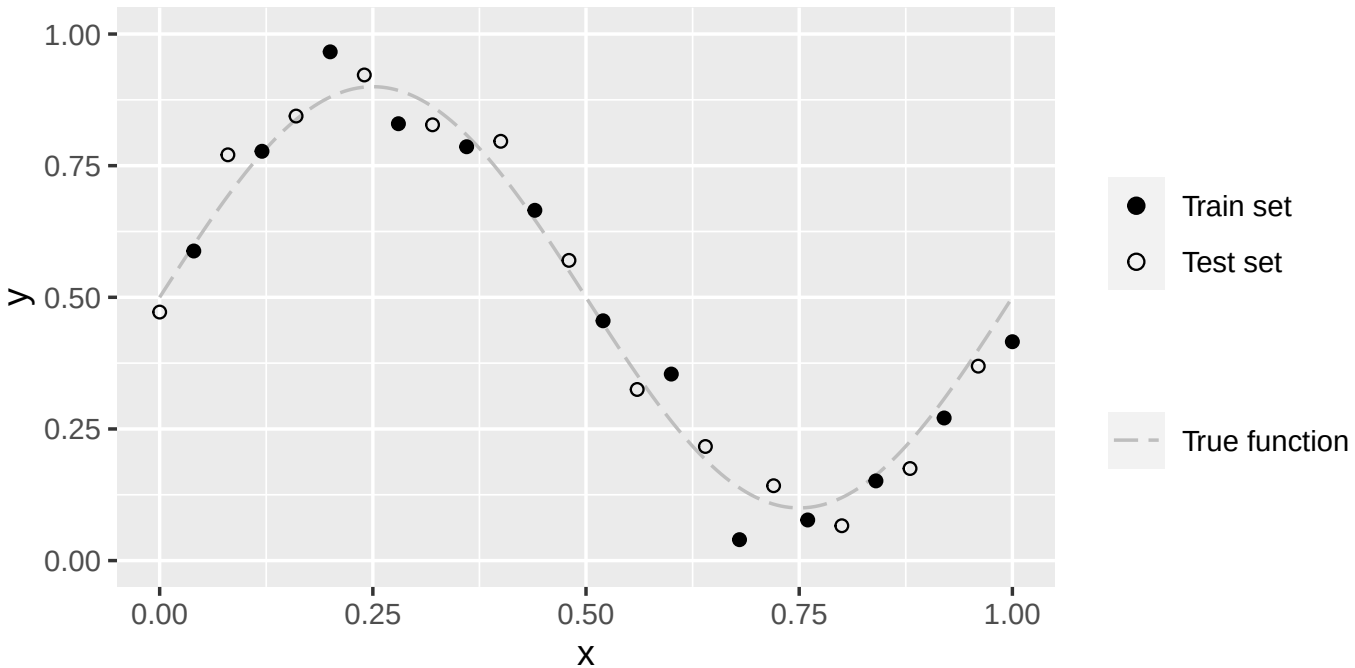
# TEST ERROR AND HOLD-OUT SPLITTING

- Split data into 2 parts, e.g., 2/3 for training, 1/3 for testing
- Evaluate on data not used for model building



# TEST ERROR

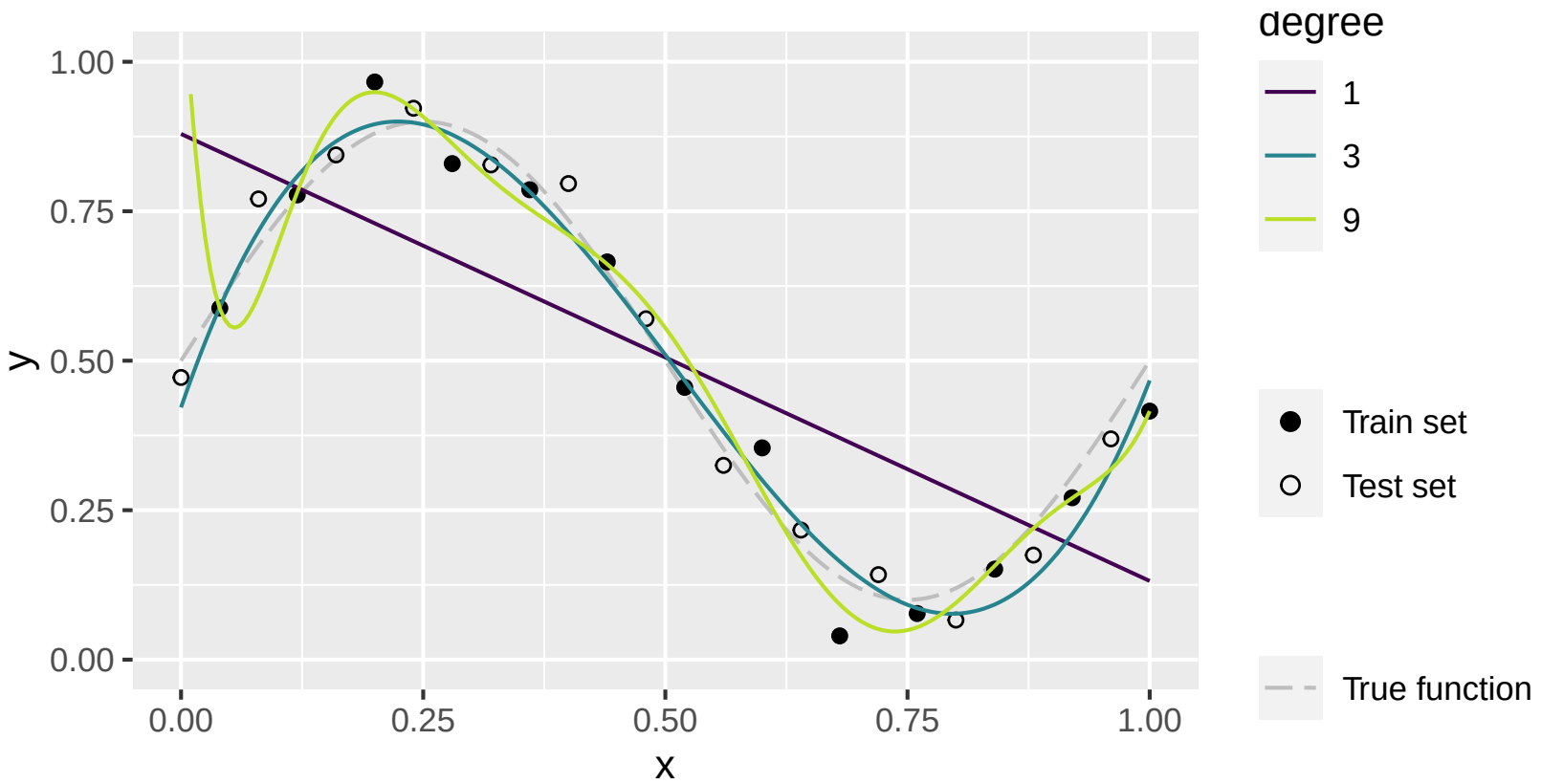
Let's consider the following example:
Sample data from sinusoidal function $0.5 + 0.4 \cdot \sin(2\pi x) + \epsilon$




Try to approximate with a $d^{th}$-degree polynomial:

$$f(\mathbf{x} \mid \boldsymbol{\theta}) = \theta_0 + \theta_1 x + \cdots + \theta_d x^d = \sum_{j=0}^{d} \theta_j x^j.$$

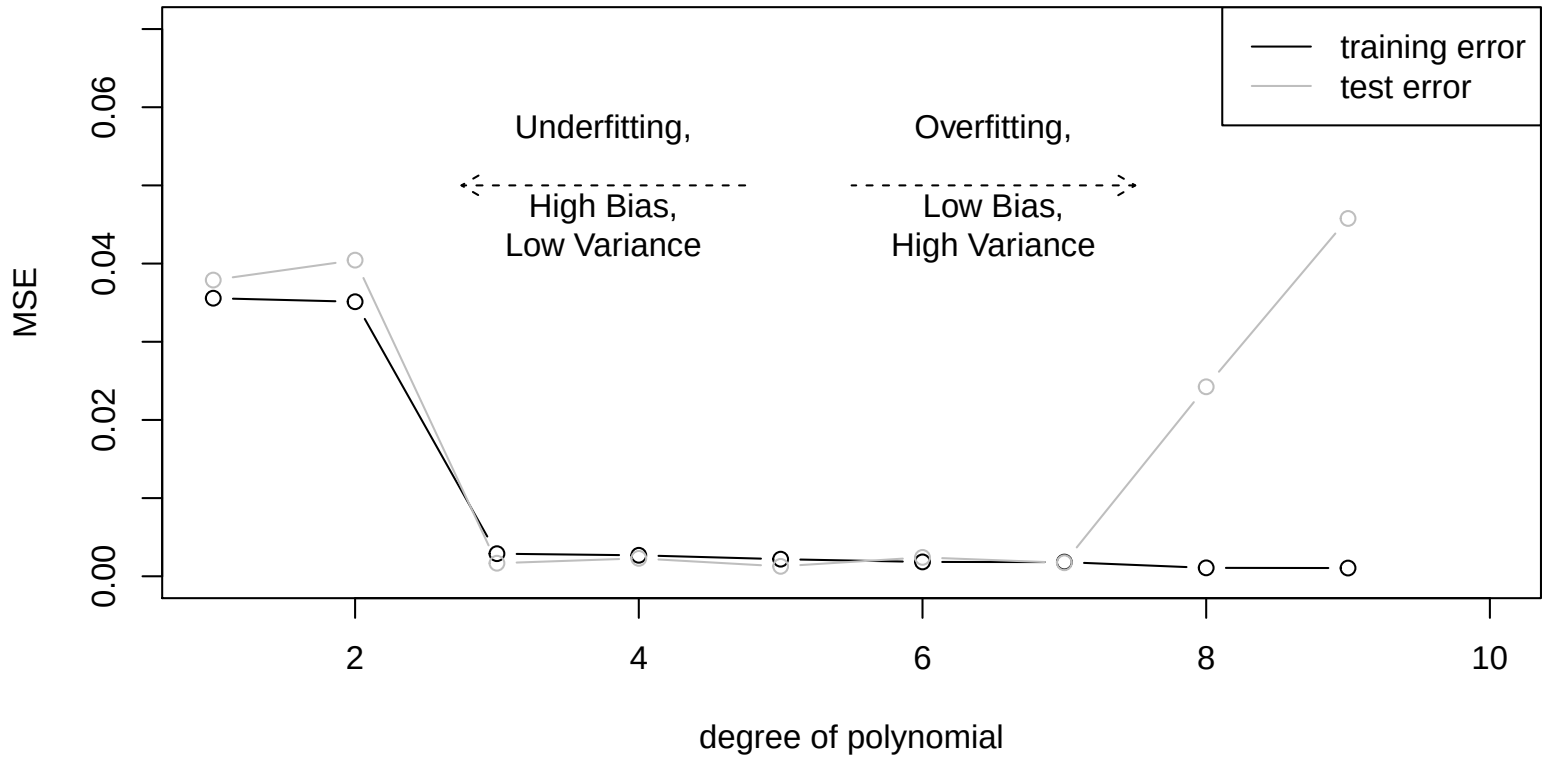
# TEST ERROR




- d=1: MSE = 0.038: Clear underfitting
- d=3: MSE = 0.002: Pretty OK
- d=9: MSE = 0.046: Clear overfitting

# TEST ERROR

Plot evaluation measure for all polynomial degrees:




Increase model complexity (tendentially)

- decrease in training error
- U-shape in test error
  (first underfit, then overfit, sweet-spot in the middle)

# TEST ERROR PROBLEMS

- Test data has to be i.i.d. compared to training data.
- Bias-variance of hold-out:
  - The smaller the training set, the worse the model $\rightarrow$ biased estimate.
  - The smaller the test set, the higher the variance of the estimate.
- If the size of our initial, complete data set $\mathcal{D}$ is limited, single train-test splits can be problematic.

# TEST ERROR PROBLEMS

A major point of confusion:

- In ML we are in a weird situation. We are usually given one data set. At the end of our model selection and evaluation process we will likely fit one model on exactly that complete data set. As training error evaluation does not work, we have nothing left to evaluate exactly that model.
- Hold-out splitting (and resampling) are tools to estimate the future performance. All of the models produced during that phase of evaluation are intermediate results.