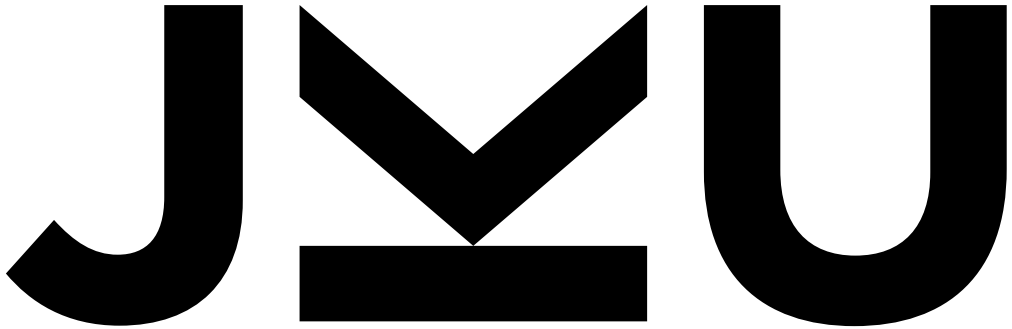
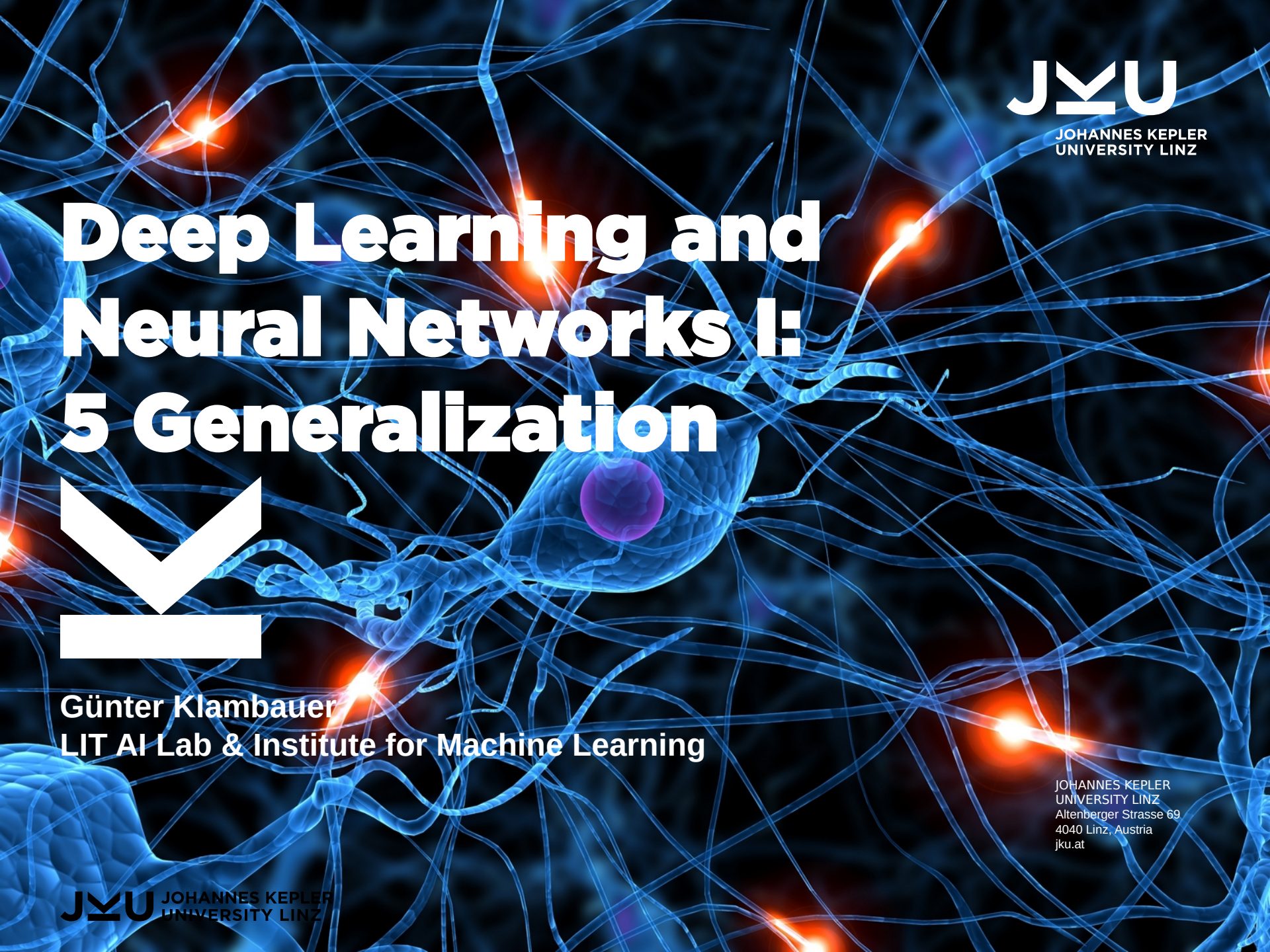


JOHANNES KEPLER
UNIVERSITY LINZ



JOHANNES KEPLER UNIVERSITY LINZ

Deep Learning and Neural Networks I: 5 Generalization

Günter Klambauer
LIT AI Lab & Institute for Machine Learning

JOHANNES KEPLER
UNIVERSITY LINZ
Altenberger Strasse 69
4040 Linz, Austria
jku.at

JOHANNES KEPLER UNIVERSITY LINZ

Recap: Backprop

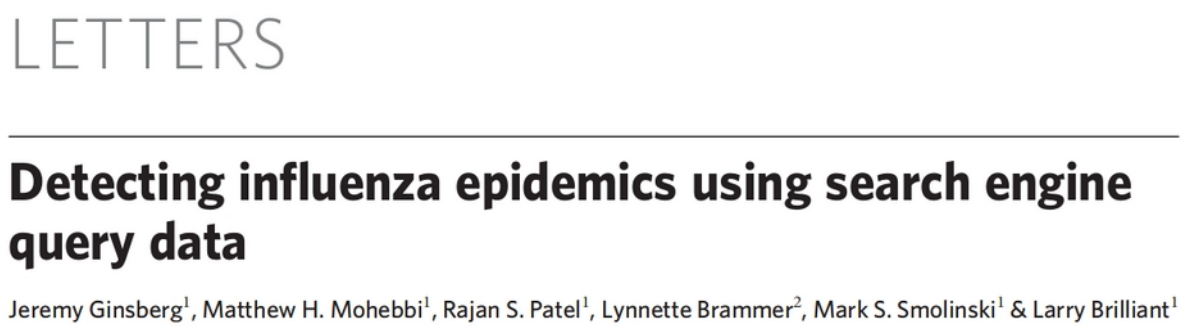
- Matrix dimensions in backprop

$$\frac{\partial}{w_{ij}^{[1]}} L(\boldsymbol{y}, g(\boldsymbol{x}; \boldsymbol{W}^{[1]}, \ldots, \boldsymbol{W}^{[L]})) = \underbrace{\frac{\partial L(\boldsymbol{y}, \hat{\boldsymbol{y}})}{\partial \hat{\boldsymbol{y}}} \frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{s}^{[L]}}}_{=:A} \cdot \ldots \cdot \underbrace{\frac{\partial \boldsymbol{s}^{[l]}}{\partial \boldsymbol{s}^{[l-1]}}}_{=:B} \cdot \ldots \cdot \underbrace{\frac{\partial \boldsymbol{s}^{[1]}}{\partial w_{ij}^{[1]}}}_{=:C}$$

$[1 \times 1]$ $\qquad$ $[1 \times K][K \times K] \ldots [n_h^{[l]} \times n_h^{[l-1]}] \ldots [D \times 1]$

Motivational example: The failure of Google Flu Trends

- In 2009, Google published a paper that states that flu trends can be predicted by search queries



LETTERS

Detecting influenza epidemics using search engine query data

Jeremy Ginsberg[1], Matthew H. Mohebbi[1], Rajan S. Patel[1], Lynnette Brammer[2], Mark S. Smolinski[1] & Larry Brilliant[1]

- The service repeatedly failed to correctly predict the trend (e.g. completely missing a non-seasonal flu)
- In 2015, the service was quietly abandoned
- What had happened? I attribute this to a thing called *overfitting...*

Introductory example

X	y
0.03	0.35
0.14	0.57
0.19	0.87
0.28	1.21
0.43	0.48
0.41	0.70
0.63	-0.44
0.69	-1.05
0.79	-1.29
0.85	-1.11
0.99	0.32

- Our supervised data set is:
  - Samples are rows in the data matrix $\boldsymbol{X}$:

$$\boldsymbol{X} = (\boldsymbol{x}^1, \ldots, \boldsymbol{x}^N)$$

  - Features are columns of $\boldsymbol{X}$:
    - Single features,e.g.: $x_{21} = 0.14$
    - Feature vector, e.g.: $\boldsymbol{x}_{.1}$
    - In this case, we only have a single feature
  - Scalar label for each data point:

$$\boldsymbol{y} = (y^1, \ldots, y^N)^T$$

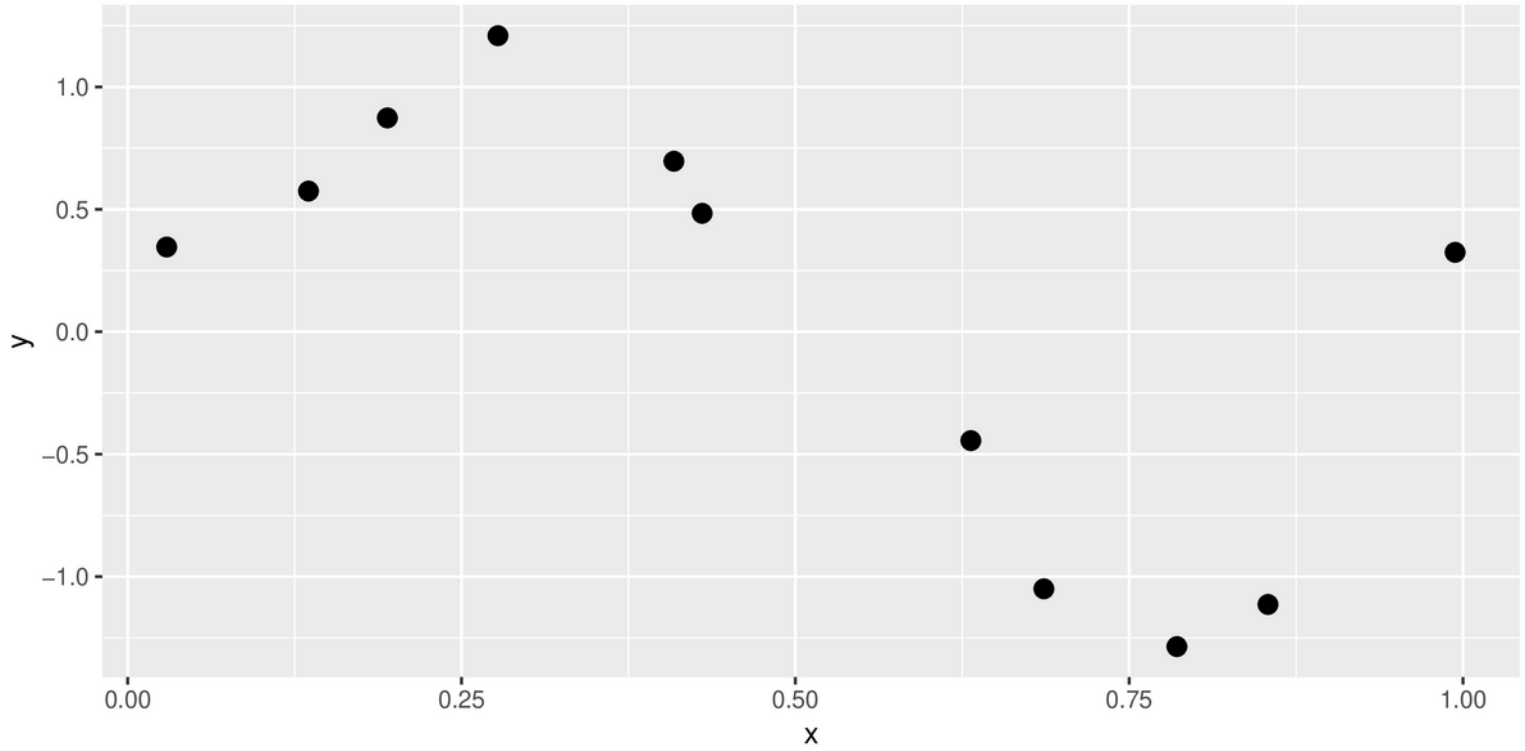
Introductory example

- A *machine learning model* is typically a function

$$\hat{y} = g(\boldsymbol{x}, \boldsymbol{w})$$

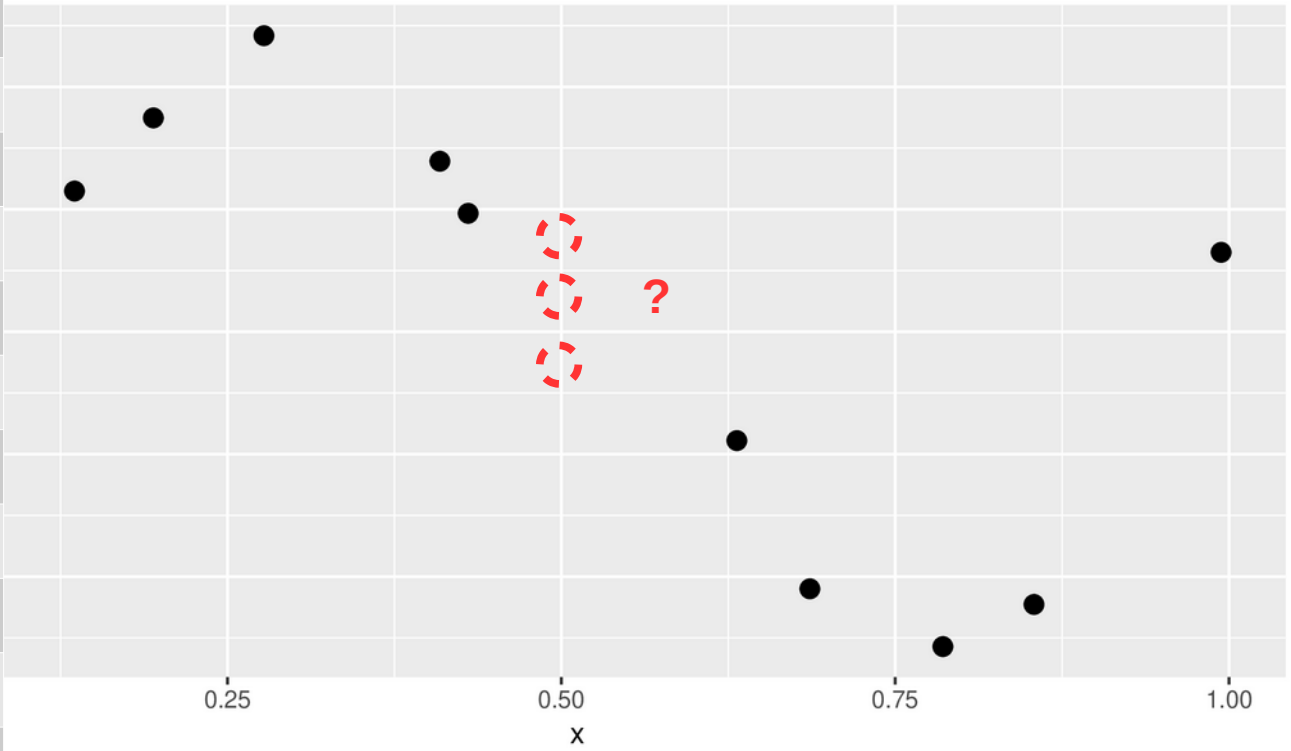
  - that takes an example $\boldsymbol{x}$
  - That has a concrete set of parameters $\boldsymbol{w}$
  - Outputs something that we are interested in: $\hat{y}$
- Important: Difference between *model class* and *model*
  - *Model class* is the type of function, e.g. $\boldsymbol{w}^T\boldsymbol{x}$, whereas
  - The *model* is a function with concrete parameter values
- Important questions in machine learning
  - What is the appropriate model class?
  - How do we select the model? How do we find $\boldsymbol{w}$?
  - **How good is a model? How can we assess this?**

Introductory example



Introductory example

X	y
0.03	0.35
0.14	0.57
0.19	0.87
0.28	1.21
0.43	0.48
0.41	0.70
0.63	-0.44
0.69	-1.05
0.79	-1.29
0.85	-1.11
0.99	0.32
0.5	?



Introductory example

- We fit a function (model) to the given data:

$$\hat{y} = g(\boldsymbol{x}, \boldsymbol{w})$$

  For each data point $\boldsymbol{x}$, this function g should provide the value $\hat{y}$ that is close to the label y.

- We fit the parameters $\boldsymbol{w}$ such that the squared distance between the function value and the label is minimized:

$$R_{\text{emp}}(\boldsymbol{w}) = \frac{1}{2} \sum_{n=1}^{N} (g(\boldsymbol{x}^n, \boldsymbol{w}) - y^n)^2$$

  This function is called *empirical error function (empirical risk)*.
- The factor 0.5 is included for later convenience

Introductory example

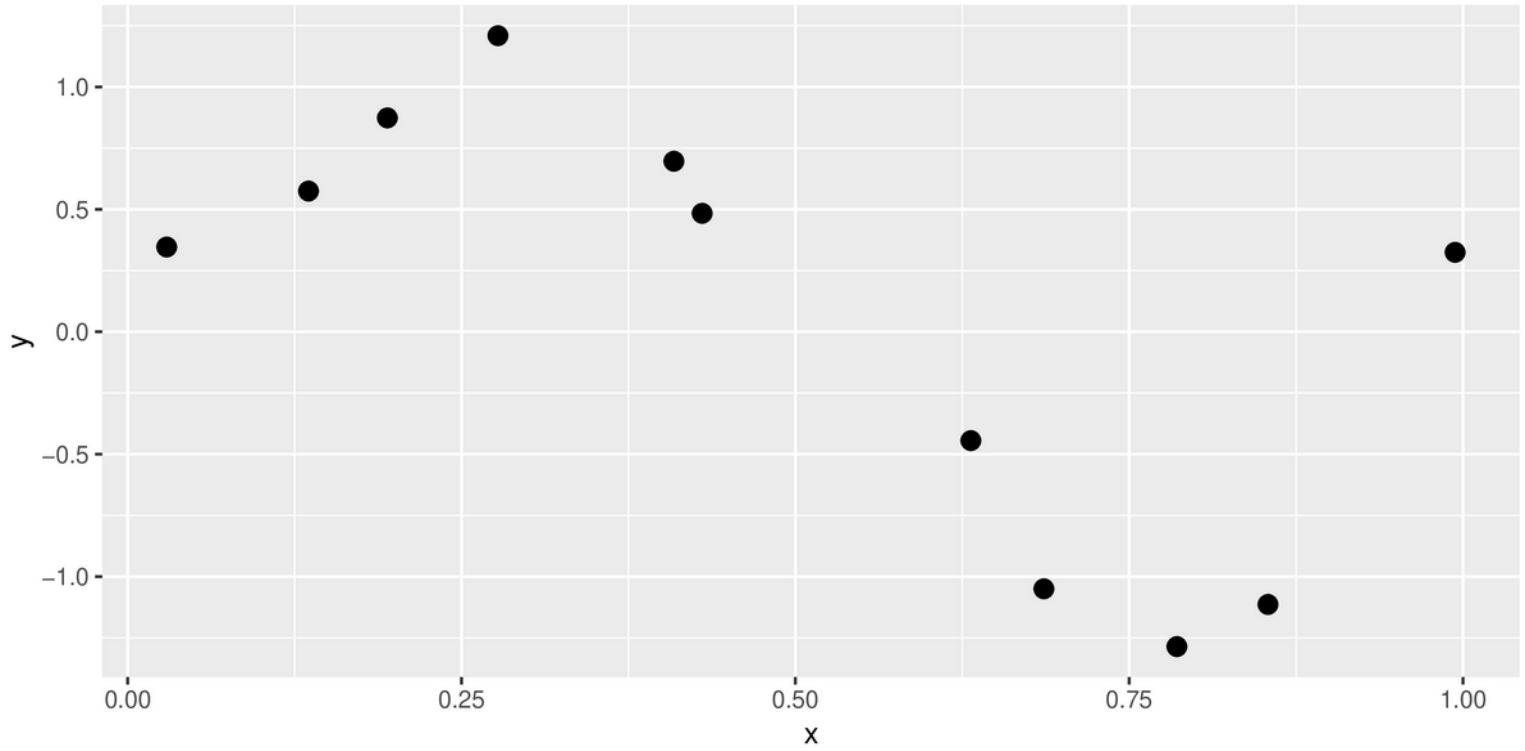
$$R_{\text{emp}}(\boldsymbol{w}) = \frac{1}{2}\sum_{n=1}^{N}(g(\boldsymbol{x}^n, \boldsymbol{w}) - y^n)^2$$

- This empirical error is larger or equal zero.
- It is zero if and only if the curve passes exactly through each data point.
- We can solve the curve fitting problem by choosing the value of $\boldsymbol{w}$ for which $R_{\text{emp}}(\boldsymbol{w})$ is as small as possible
- Let us – for now – assume that we can minimize $R_{\text{emp}}(\boldsymbol{w})$ with respect to $\boldsymbol{w}$

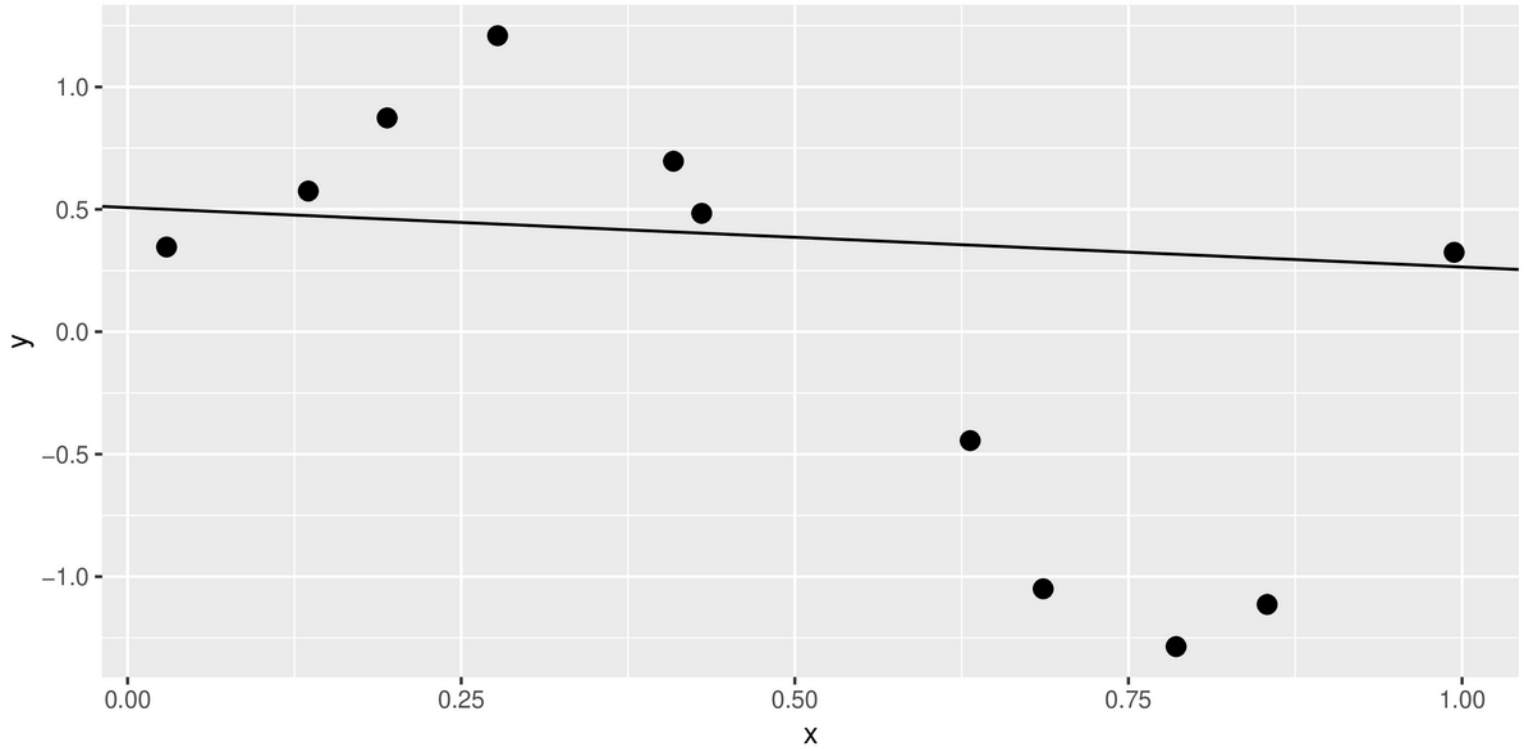
$$\boldsymbol{w}^* = \text{argmin}_{\boldsymbol{w}}\, R_{\text{emp}}(\boldsymbol{w})$$

- We know how to find $\boldsymbol{w}^*$, for example for linear models.

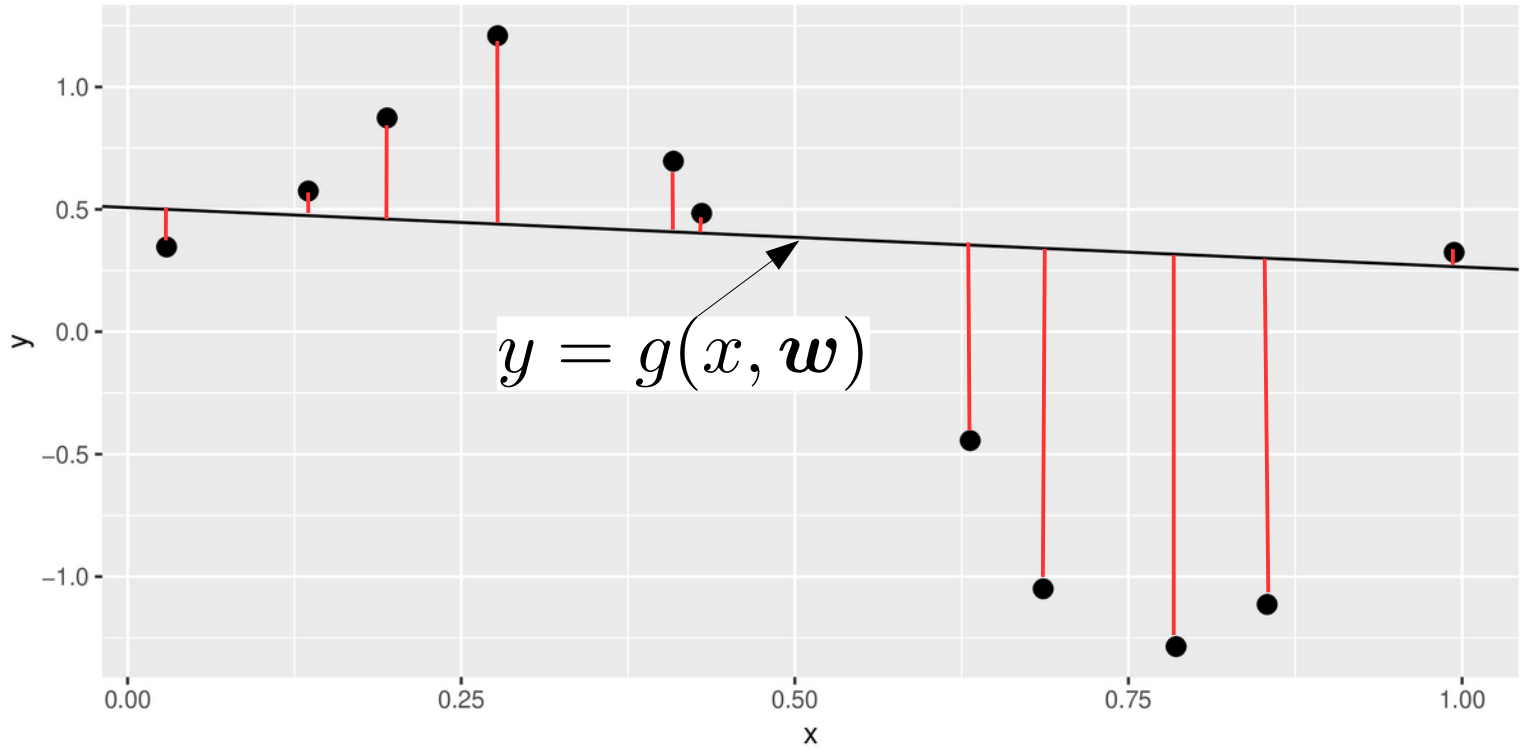
Introductory example



Introductory example



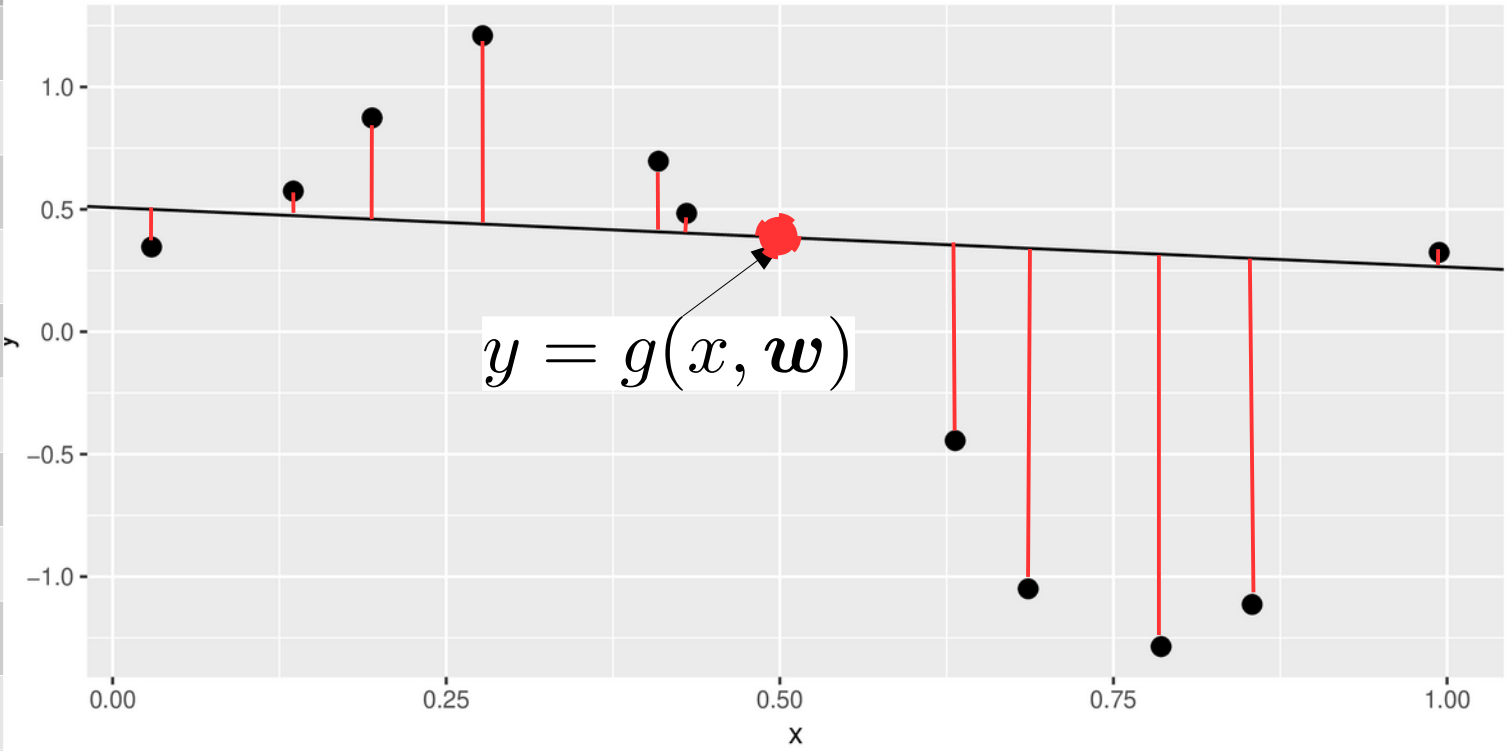
Introductory example



- We fit: $g(x, \boldsymbol{w}) = w_0 + w_1 x$ $\quad \boldsymbol{w} = (w_0, w_1)$
- We obtain: $g(x, \boldsymbol{w}) = 0.507 - 0.242x$

Introductory example

X	y
0.03	0.35
0.14	0.57
0.19	0.87
0.28	1.21
0.43	0.48
0.41	0.70
0.63	-0.44
0.69	-1.05
0.79	-1.29
0.85	-1.11
0.99	0.32
0.5	**0.39?**



- We fit: $g(x, \boldsymbol{w}) = w_0 + w_1 x$
- We obtain: $g(x, \boldsymbol{w}) = 0.507 - 0.242x$

Introductory example

- What if we would fit the following function?

$$g(x, \boldsymbol{w}) = w_0 + w_1 x + w_2 x^2$$

- Btw… we could also write

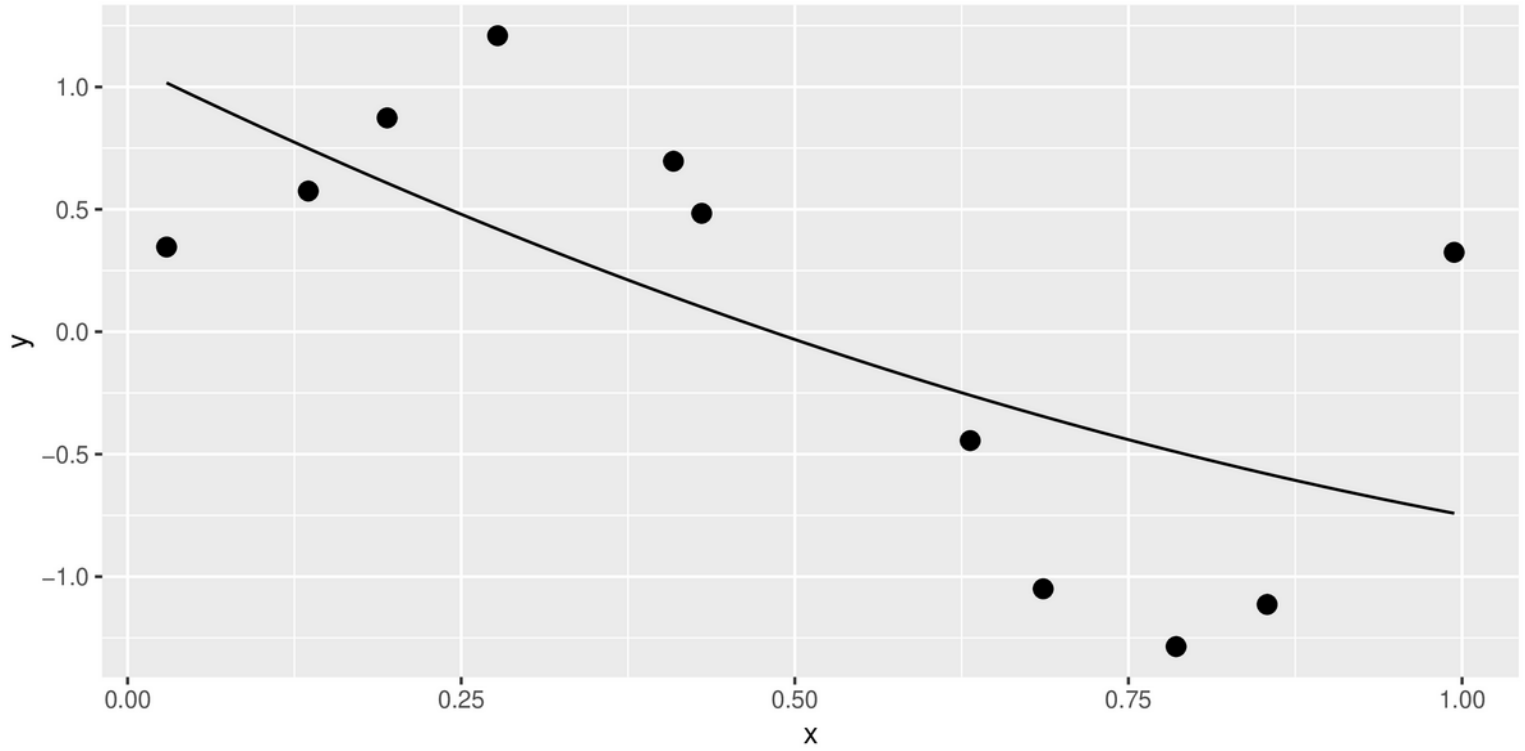
$$g(\boldsymbol{x}, \boldsymbol{w}) = \boldsymbol{w}^T \boldsymbol{x},$$

where

$$\boldsymbol{w} = (w_0, w_1, w_2, ..., w_k)$$

$$\boldsymbol{x} = (1, x, x^2, \ldots, x^k)$$

Introductory example



$$g(x, \boldsymbol{w}) = 1.09 - 2.66x + 0.82x^2$$

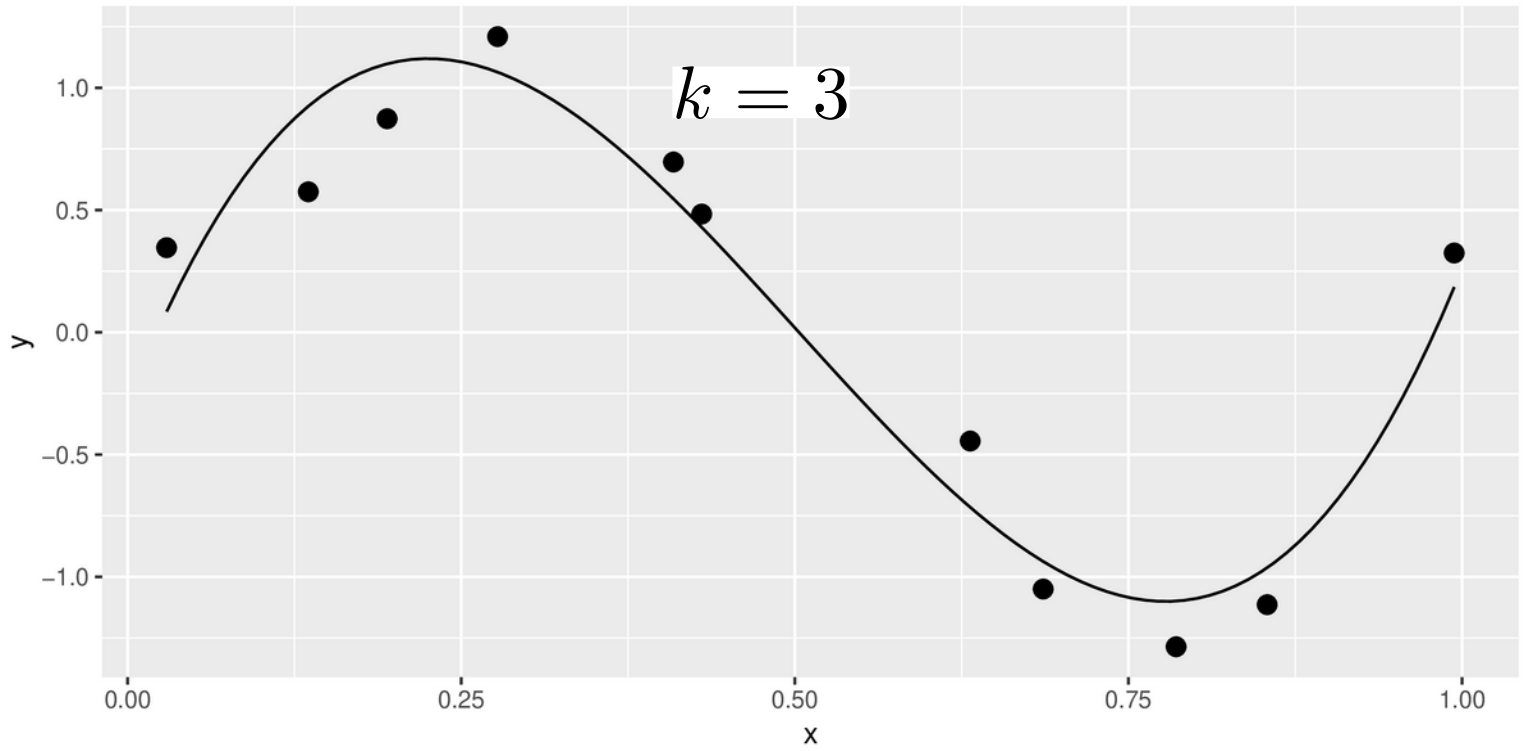
Introductory example

- Ok, let's try polynomials of higher order...

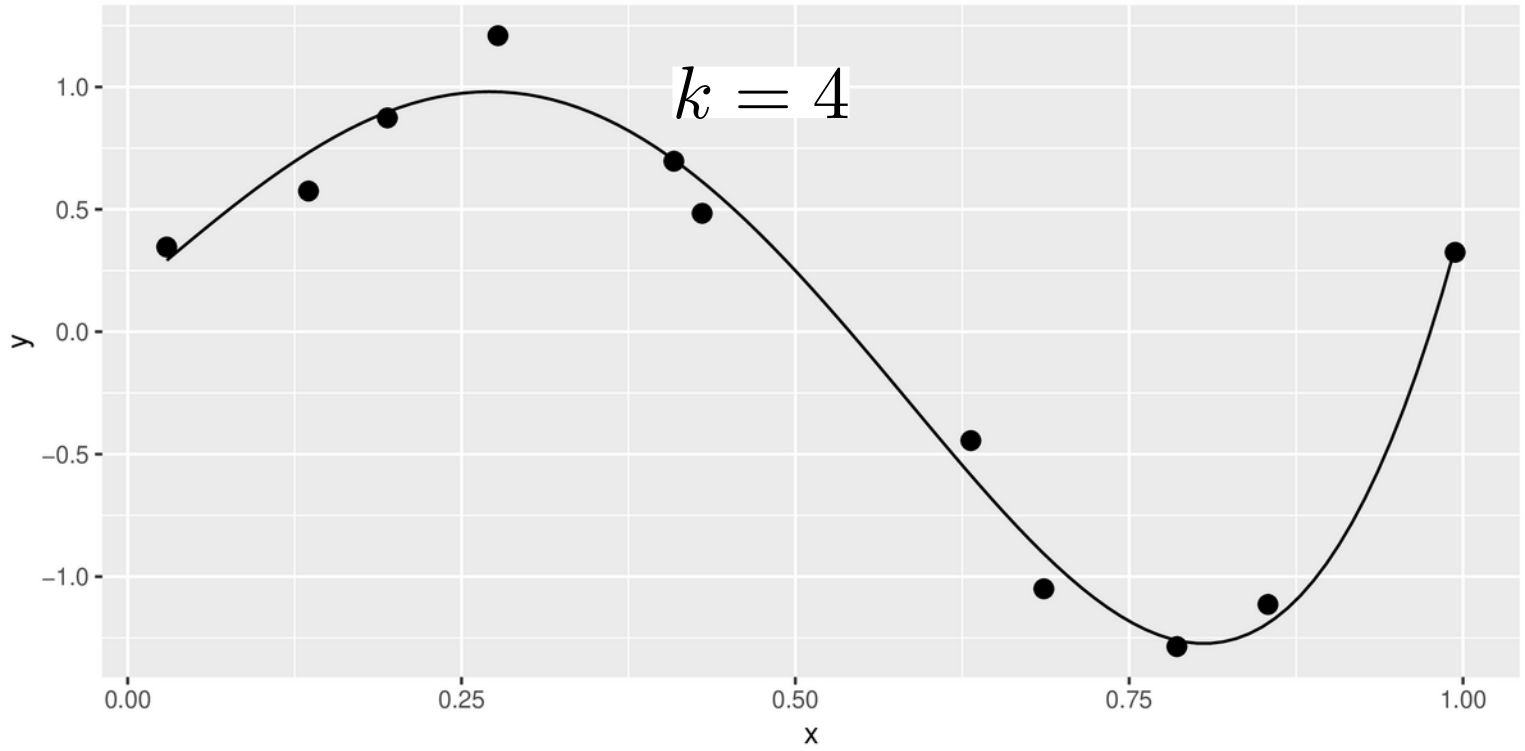
Power operation

$$g(x, \boldsymbol{w}) = \sum_{j=0}^{k} w_j x^j$$

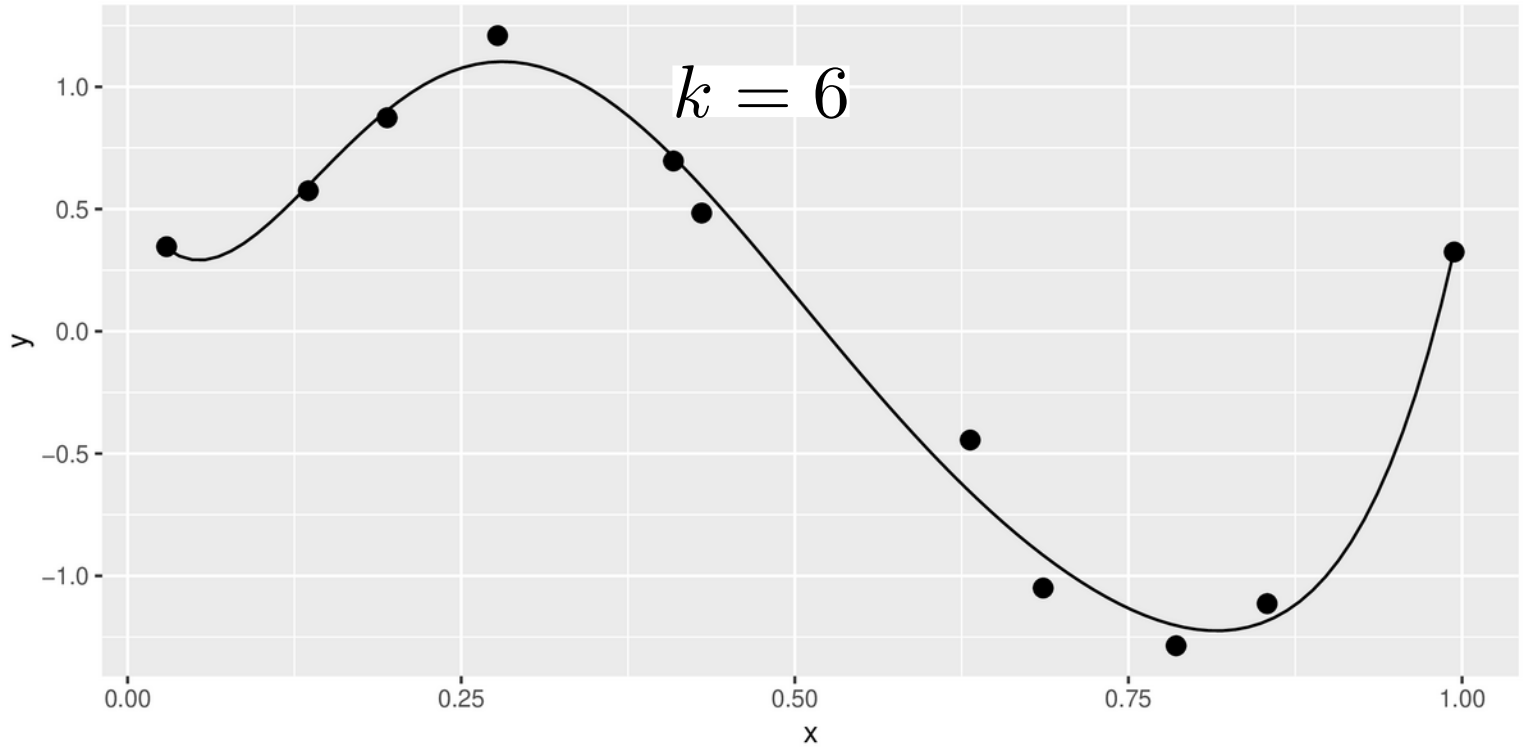
Introductory example



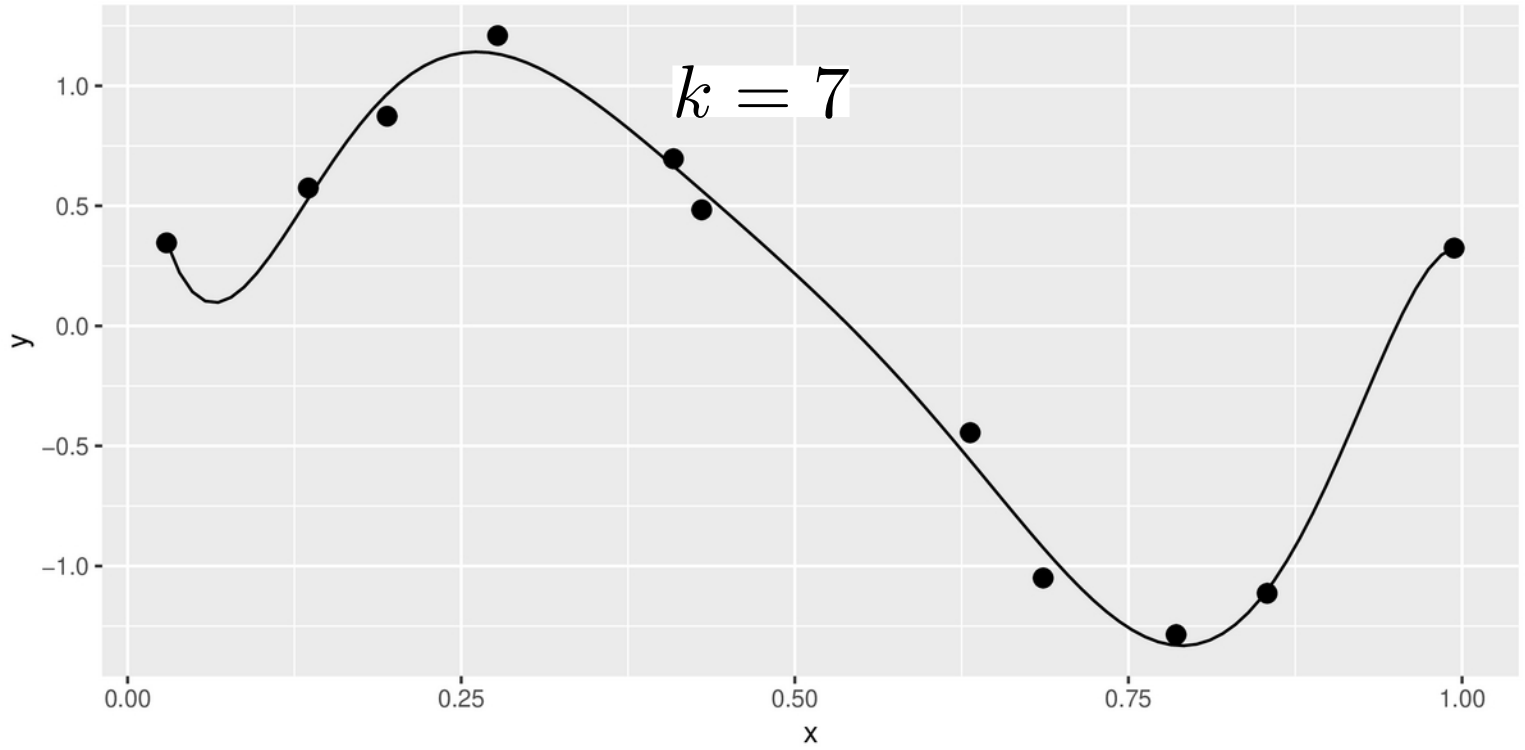
Introductory example



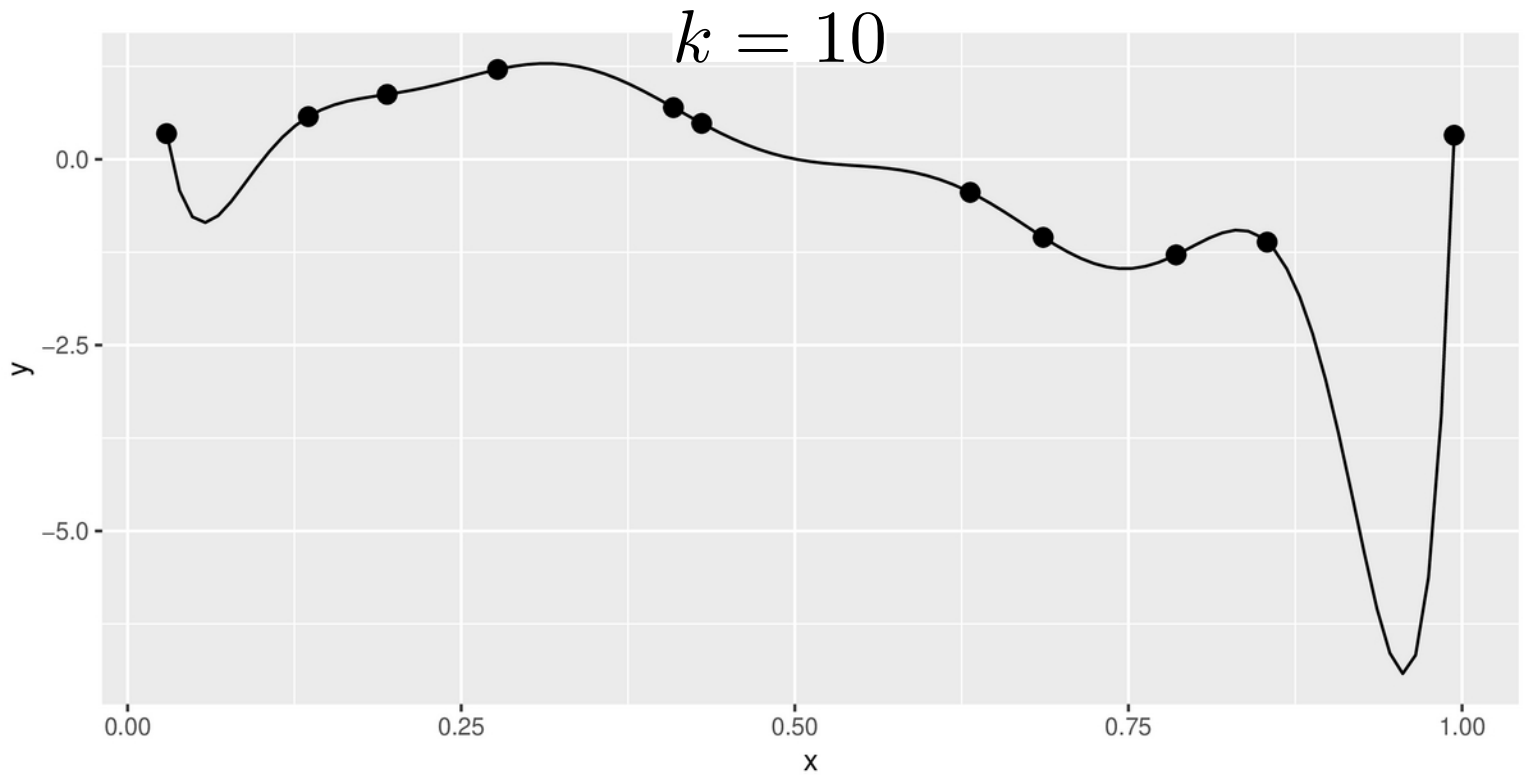
Introductory example



Introductory example



Introductory example



Introductory example

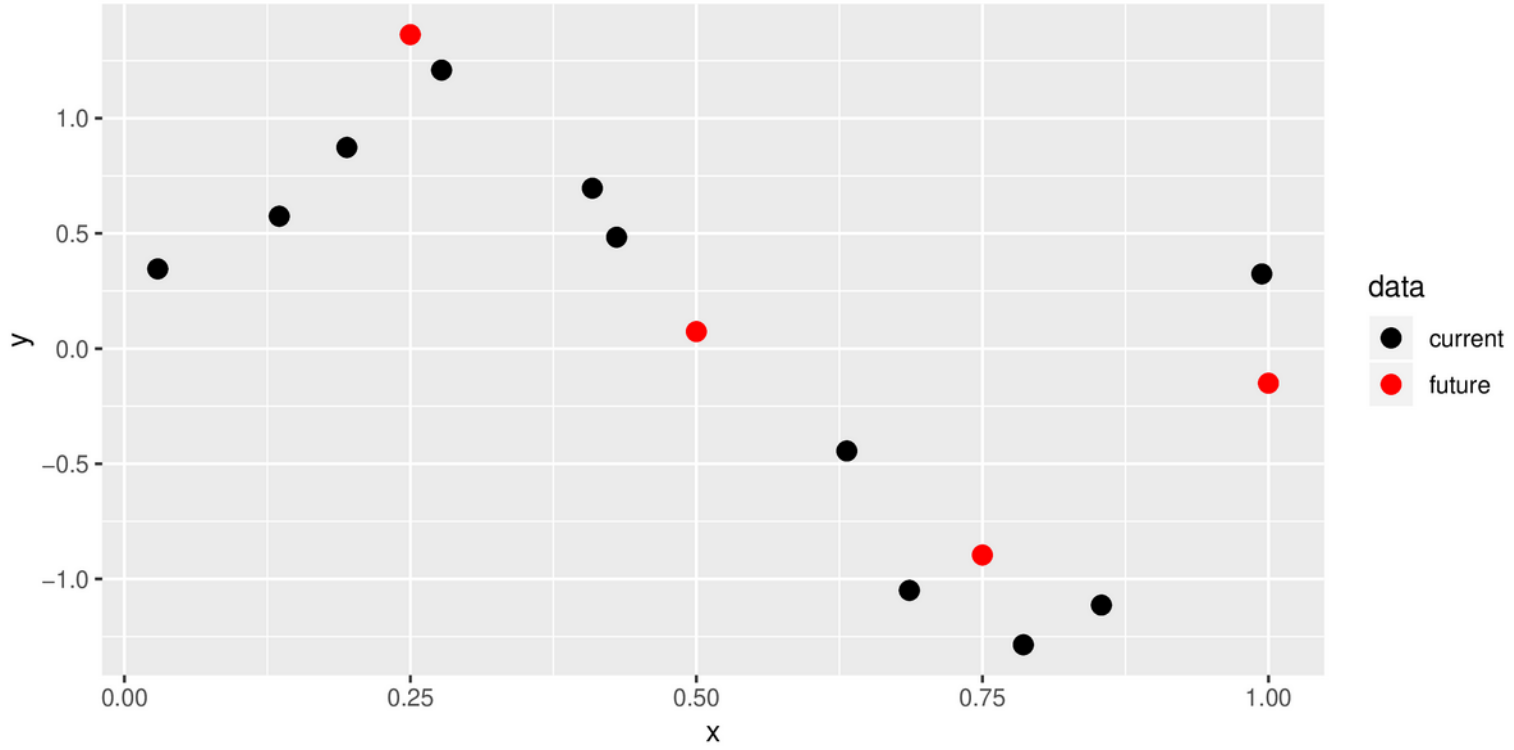
- Which of these functions was the best one?

k	$R(\boldsymbol{w})$
1	1.266
2	0.383
3	0.041
4	0.013
5	0.009
6	0.005
8	0.005
9	0.001
10	0.000

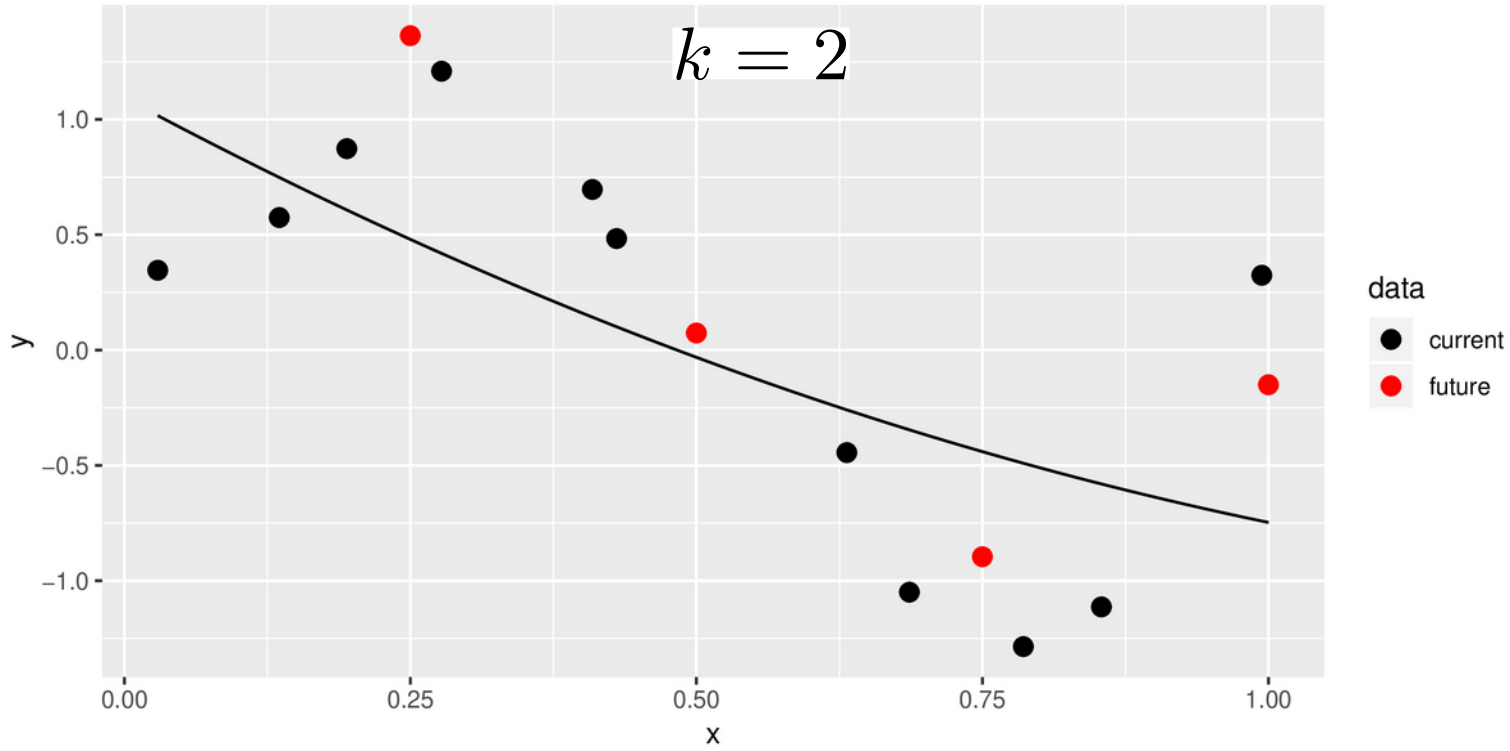
$\boldsymbol{w}$ \ k	1	2	3	4	6	7	8	9
	0.51	1.09	-0.29	0.16	0.57	1.14	0.58	-2.55
	-0.24	-2.66	13.82	4.50	-11.71	-39.45	-10.28	165.30
		0.82	-39.56	2.66	151.26	495.42	74.25	-2889.32
			26.29	-38.33	-585.86	-2431.31	317.47	24171.73
				31.47	983.67	6015.98	-3545.80	-110968.15
					-782.52	-8064.19	10837.85	300400.90
					245.08	5561.06	-15725.69	-492113.49
						-1538.32	11176.11	478948.01
							-3124.27	-254578.08
								56866.84

Introductory example

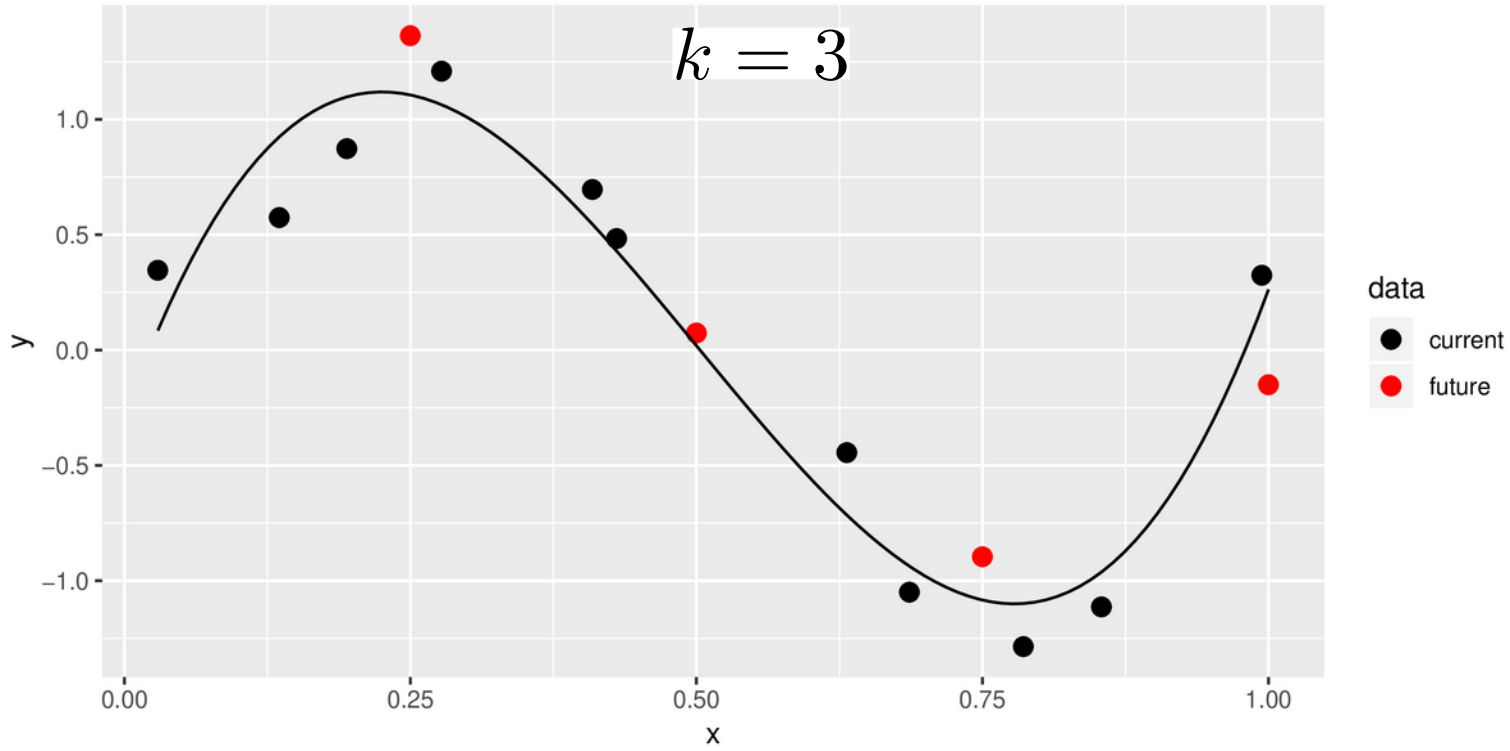
- What if we had new data points (future data)? Or: What if we had left out some data points?



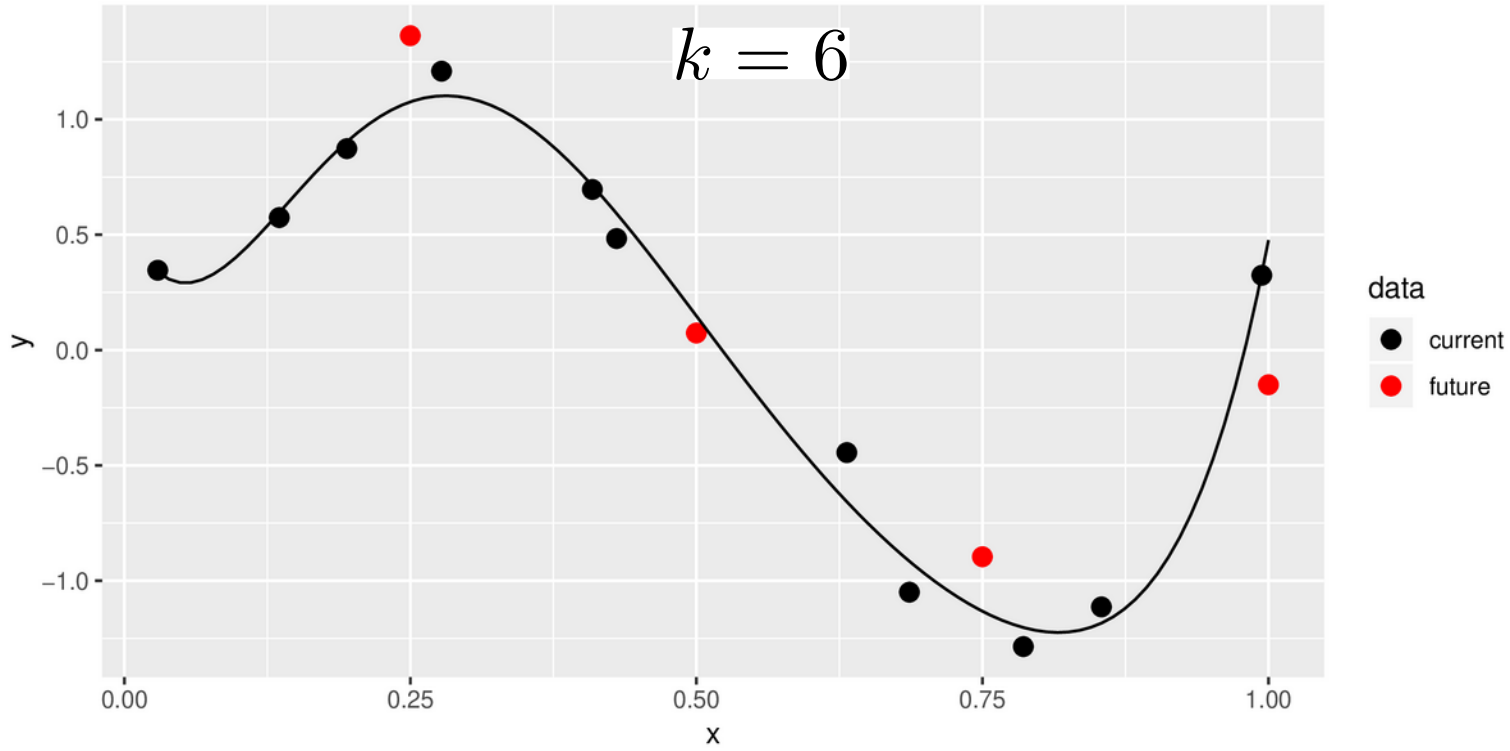
Introductory example



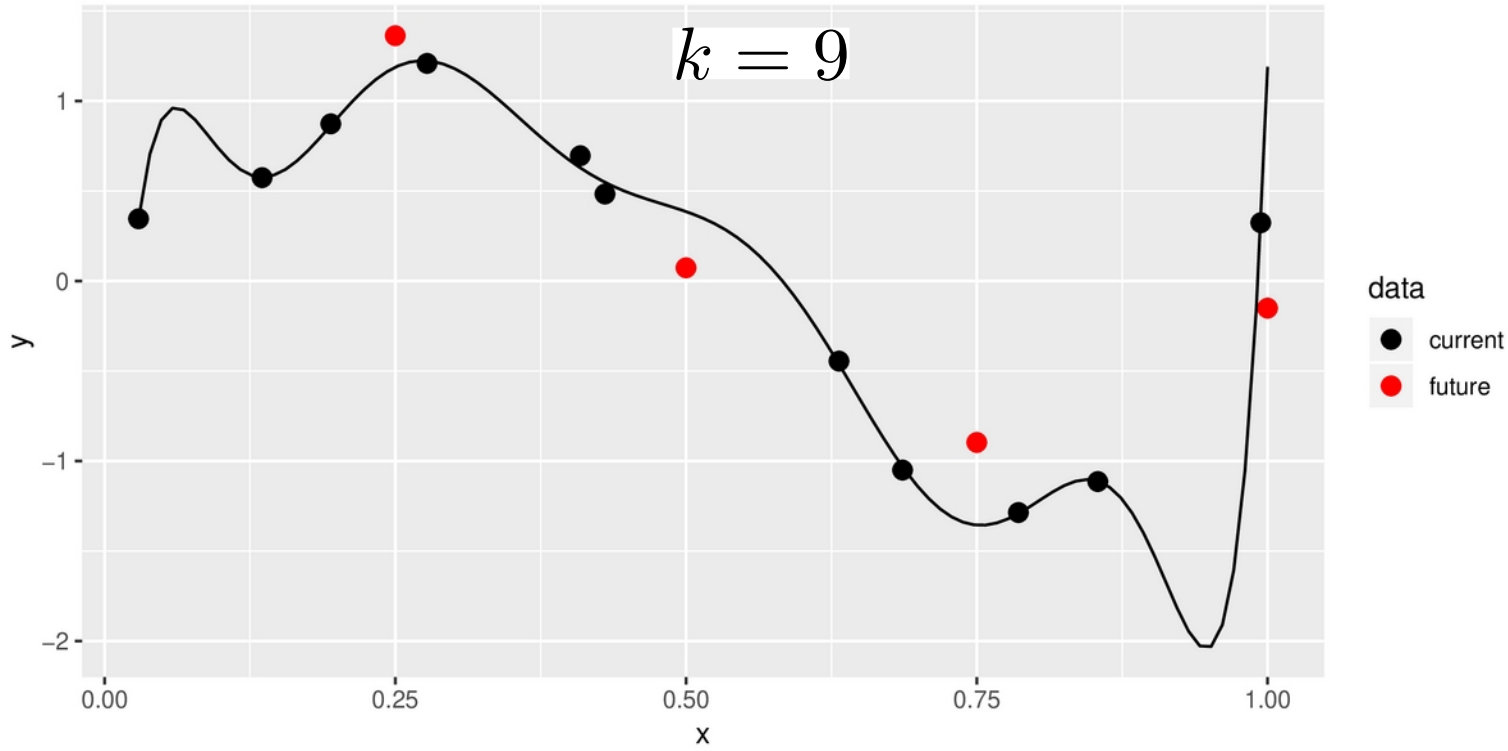
Introductory example



Introductory example



Introductory example



Introductory example

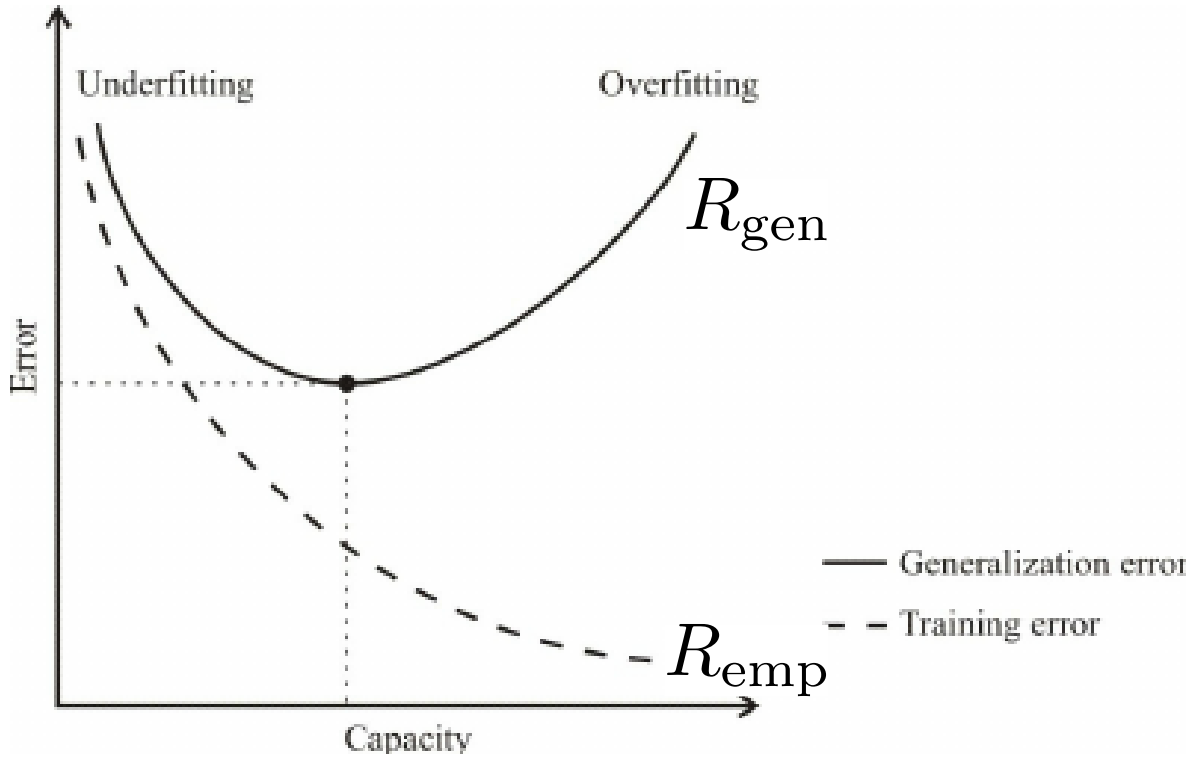
- Do we know more now that we have these “future data points”?

k	*Empirical error*	*Estimate of “future error”*
1	1.27	1.17
2	0.38	0.34
3	0.04	0.07
4	0.01	0.16
5	0.01	0.13
6	0.00	0.11
8	0.00	0.08
9	0.00	0.53
10	0.00	3.57

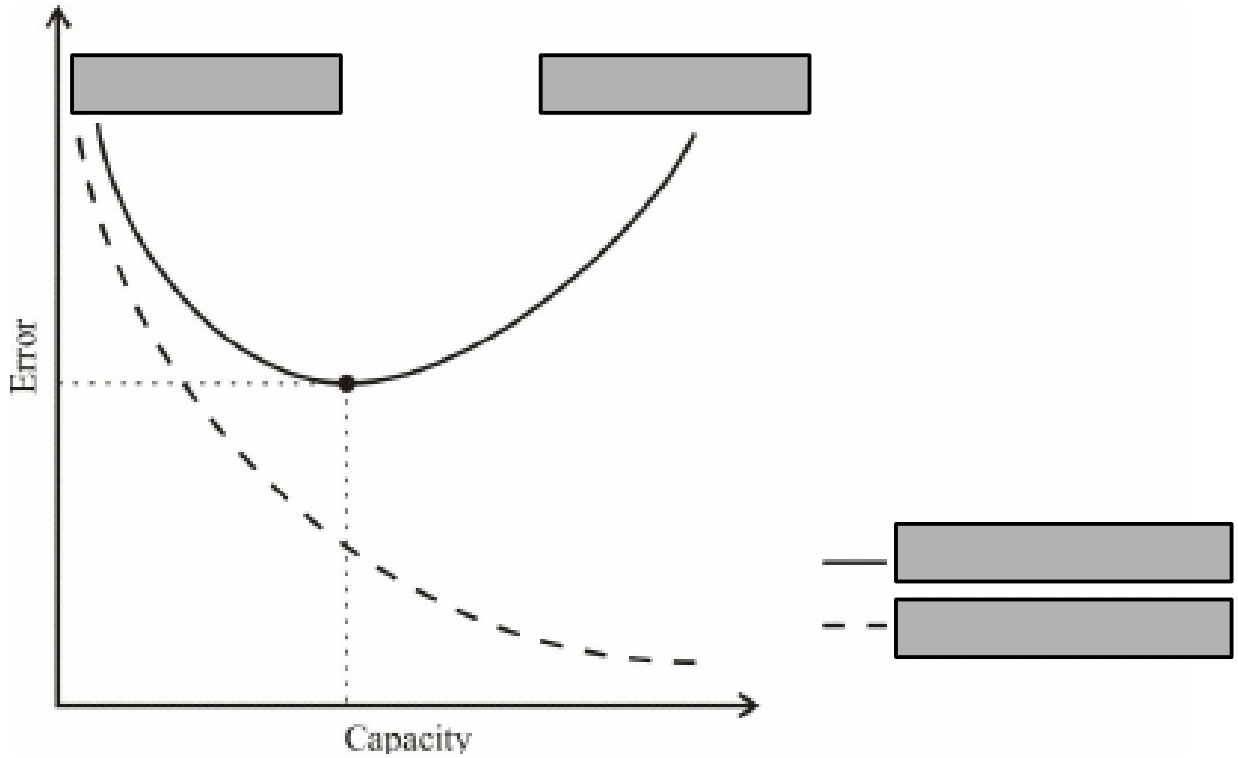
Introductory example

- What was actually the problem with our strategy to increase the order of the polynomial?
  - Our model g got increasingly “complex”
  - Our model g could fit the given data (training data) better and better
  - Our empirical error got better, but could not be used as an indicator how good the model would be on future data (test data)
    - The model parameters were adjusted to the given data

Capacity, overfitting and underfitting



Capacity, overfitting and underfitting



Loss, generalization error, risk

- Our supervised data set is:

$$\boldsymbol{x} \in \mathbb{R}^d, \; y \in \mathbb{R}$$

  - Samples are rows in the data matrix $\boldsymbol{X}$:

$$\boldsymbol{X} = (\boldsymbol{x}^1, \ldots, \boldsymbol{x}^N)$$

  - Scalar label for each data point:

$$\boldsymbol{y} = (y^1, \ldots, y^N)^T$$

  - Pairs of data points with labels:

$$\boldsymbol{z} = (\boldsymbol{x}, y)$$
$$\boldsymbol{Z} = (\boldsymbol{z}^1, \ldots, \boldsymbol{z}^N)$$

- Probability distribution:

$$p(\boldsymbol{z})$$

Loss, generalization error, risk

- Loss function is a function of model and target (=label):

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

  - Examples: Quadratic loss function

$$\mathcal{L}(y, g(\boldsymbol{x}; \boldsymbol{w})) = (y - g(\boldsymbol{x}; \boldsymbol{w}))^2$$

  - Examples: Zero-One loss function

$$\mathcal{L}(y, g(\boldsymbol{x}; \boldsymbol{w})) = \begin{cases} 0 & \text{for} \quad y = g(\boldsymbol{x}; \boldsymbol{w}) \\ 1 & \text{for} \quad y \neq g(\boldsymbol{x}; \boldsymbol{w}) \end{cases}$$

  - Examples: Cross-entropy loss function

$$\mathcal{L}(y, g(\boldsymbol{x}; \boldsymbol{w})) = -\left(y \log(g(\boldsymbol{x}; \boldsymbol{w})) + (1 - y) \log(1 - g(\boldsymbol{x}; \boldsymbol{w}))\right)$$

Loss, generalization error, risk

- The *generalization error* or *risk* is the expected loss on future data:

$$R_{\text{gen}}(g(.;\boldsymbol{w})) \;=\; \mathrm{E}_{\boldsymbol{z}}\left(\mathcal{L}(y, g(\boldsymbol{x};\boldsymbol{w}))\right)$$

$$R_{\text{gen}}(g(.;\boldsymbol{w})) \;=\; \int_{Z} \mathcal{L}(y, g(\boldsymbol{x};\boldsymbol{w}))\; p\left(\boldsymbol{z}\right)\; d\boldsymbol{z}$$

- Because we do not know $p(\boldsymbol{z})$ the risk cannot be computed; especially we do not know $p(y \mid \boldsymbol{x})$. Therefore, in practical situations, we have to approximate the risk.

Bayes optimal classifier (1/4)

For the zero-one loss, we obtain

$$R(g(.;\boldsymbol{w})) = \int_X \int_{\mathbb{R}} p(\boldsymbol{x}, y \neq g(\boldsymbol{x};\boldsymbol{w}))dyd\boldsymbol{x},$$

i.e. the *misclassification probability*. With the notations

$$X_{-1} = \{\boldsymbol{x} \in X \mid g(\boldsymbol{x};\boldsymbol{w}) < 0\}, \qquad X_{+1} = \{\boldsymbol{x} \in X \mid g(\boldsymbol{x};\boldsymbol{w}) > 0\},$$

we can conclude further:

$$R(g(.;\boldsymbol{w})) = \int_{X_{-1}} p(\boldsymbol{x}, y = +1)d\boldsymbol{x} + \int_{X_{+1}} p(\boldsymbol{x}, y = -1)d\boldsymbol{x}$$

Bayes optimal classifier (2/4)

So, we get:

$$R(g(.;\mathbf{w})) = \int_{X_{-1}} p(y=+1 \mid \mathbf{x}) \cdot p(\mathbf{x}) d\mathbf{x} + \int_{X_{+1}} p(y=-1 \mid \mathbf{x}) \cdot p(\mathbf{x}) d\mathbf{x}$$

$$= \int_{X} \left\{ \begin{array}{ll} p(y=-1 \mid \mathbf{x}) & \text{if } g(\mathbf{x};\mathbf{w}) = +1 \\ p(y=+1 \mid \mathbf{x}) & \text{if } g(\mathbf{x};\mathbf{w}) = -1 \end{array} \right\} \cdot p(\mathbf{x}) d\mathbf{x}$$

Hence, we can infer an optimal classification function, the so-called *Bayes-optimal classifier*:

$$g(\mathbf{x}) = \begin{cases} +1 & \text{if } p(y=+1 \mid \mathbf{x}) > p(y=-1 \mid \mathbf{x}) \\ -1 & \text{if } p(y=-1 \mid \mathbf{x}) > p(y=+1 \mid \mathbf{x}) \end{cases}$$

$$= \text{sign}(p(y=+1 \mid \mathbf{x}) - p(y=-1 \mid \mathbf{x})) \quad (1)$$

Bayes optimal classifier (3/4)

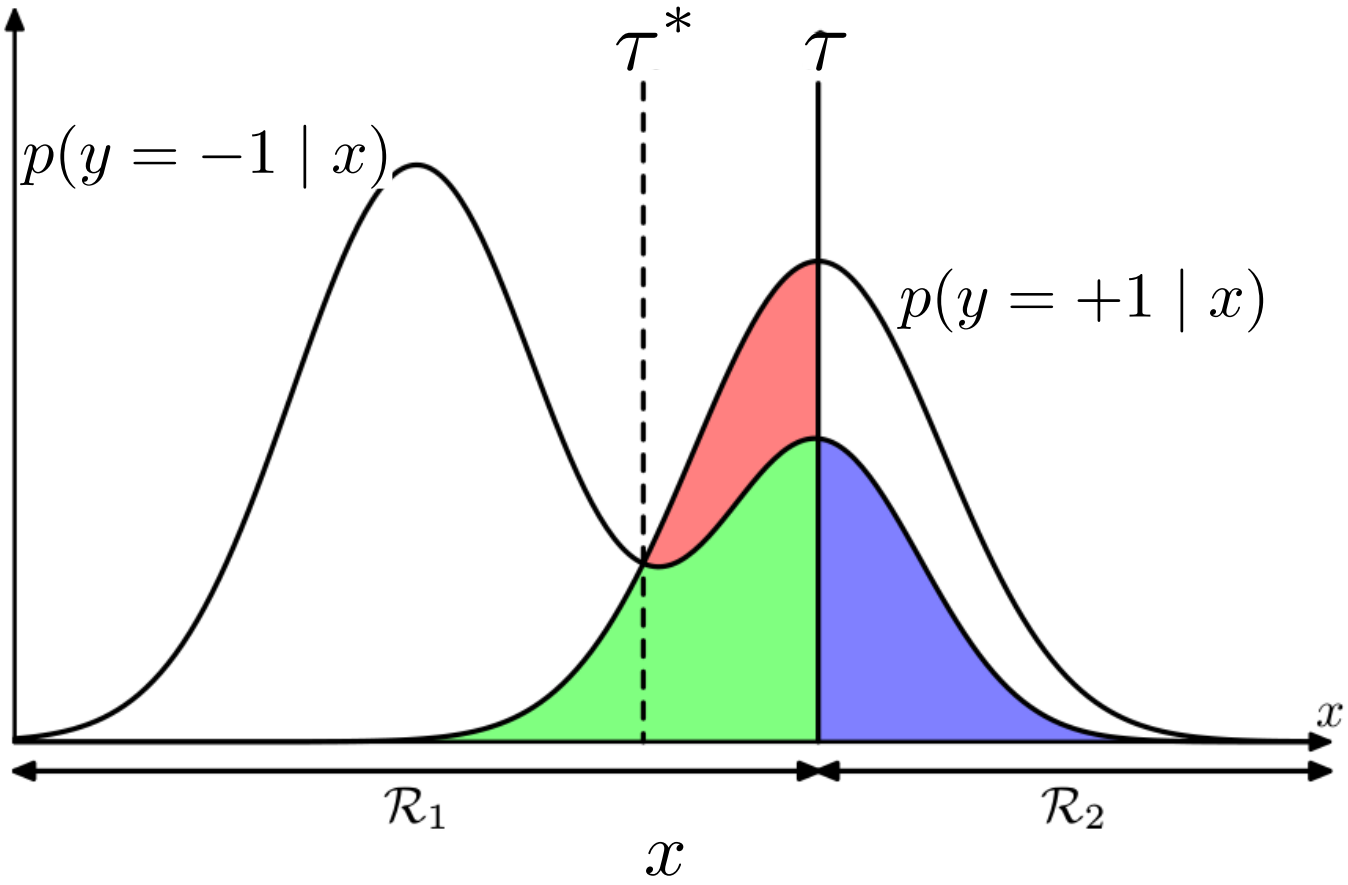
The resulting minimal risk is

$$R_{\min} = \int_X \min(p(\mathbf{x}, y = -1), p(\mathbf{x}, y = +1))d\mathbf{x}$$
$$= \int_X \min(p(y = -1 \mid \mathbf{x}), p(y = +1 \mid \mathbf{x})) \cdot p(\mathbf{x})d\mathbf{x}$$

Obviously, for non-overlapping classes, i.e. $\min(p(y = -1 \mid \mathbf{x}), p(y = +1 \mid \mathbf{x})) = 0$, the minimal risk is zero and the optimal classification function is

$$g(\mathbf{x}) = \begin{cases} +1 & \text{if } p(y = +1 \mid \mathbf{x}) > 0, \\ -1 & \text{if } p(y = -1 \mid \mathbf{x}) > 0. \end{cases}$$

Bayes optimal classifier (4/4)



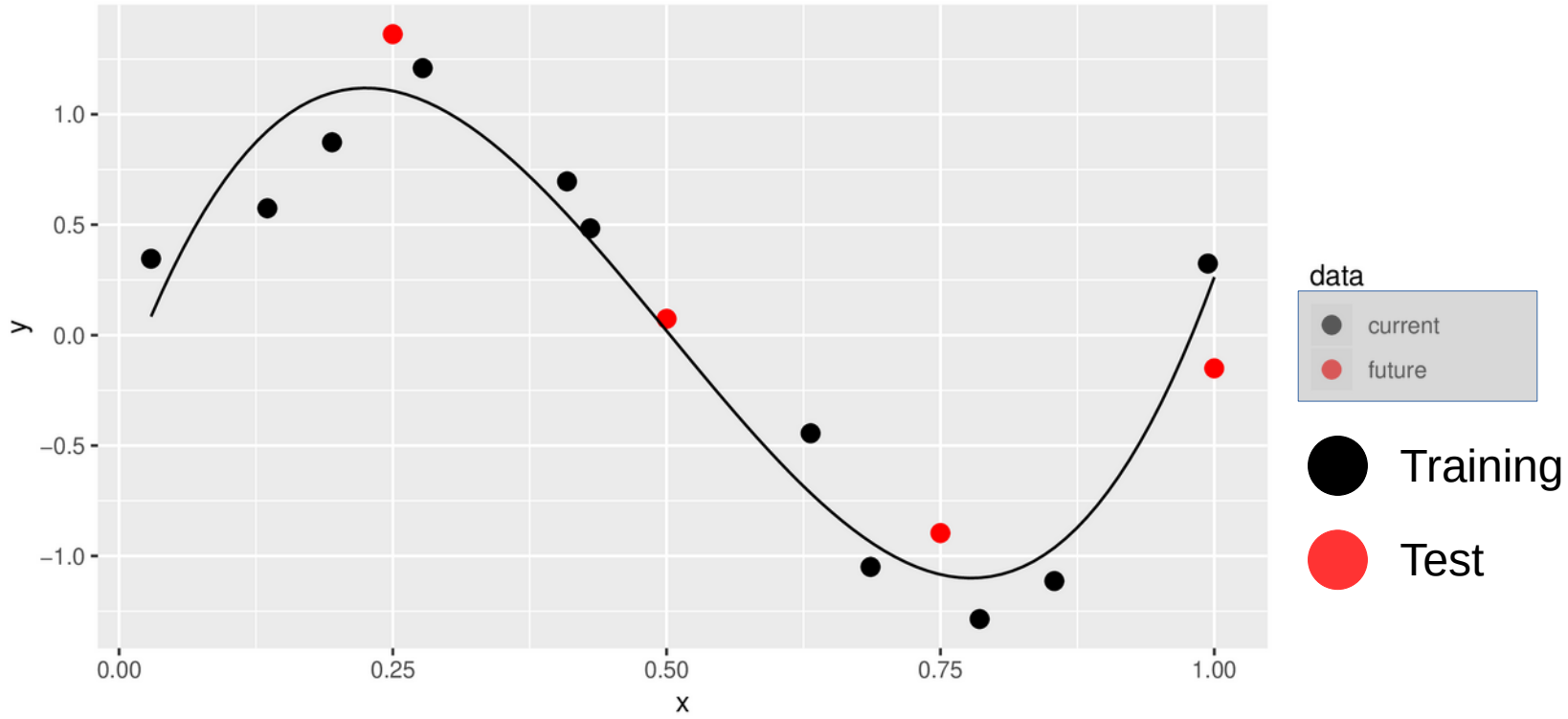
Empirical estimation of Generalization Error: Test Set

- We assume that data points are “iid” (independently identically distributed) and therefore:

$$R_{\text{gen}}(g(.;\boldsymbol{w})) \;=\; \mathrm{E}_{\boldsymbol{z}}\left(\mathcal{L}(y, g(\boldsymbol{x};\boldsymbol{w}))\right) \approx \;\frac{1}{M}\sum_{m=N+1}^{N+M}\mathcal{L}\left(y^m, g(\boldsymbol{x}^m;\boldsymbol{w})\right)$$

  where the set of elements $\{\boldsymbol{z}_{N+1}, \ldots, \boldsymbol{z}_{N+M}\}$ is called *test set*.

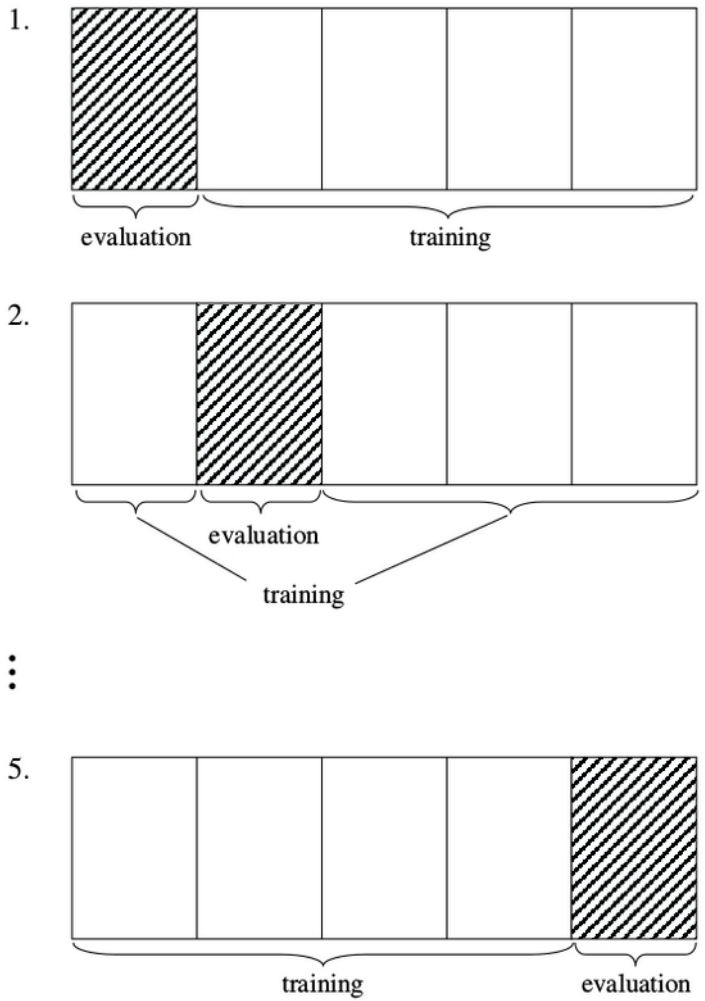
Empirical estimation of Generalization Error: Test Set



Empirical estimation of Generalization Error: Cross-validation

- Few data points available we want to use them all for learning and not for estimating the performance via a test set
- But we want to estimate the performance for our final model
  - We can divide the available data multiple times into training data and test data and average over the result.
  - Th problem here is that the test data is overlapping and we estimate with dependent test data points.
  - Solution: dividing the training data into several folds

Empirical estimation of Generalization Error: Cross-validation



Empirical estimation of Generalization Error: L-fold cross-validation

- We write $\boldsymbol{Z}_N := \boldsymbol{Z}$ as a variable for training sets with elements

$$R_{L-\mathrm{cv}}(\boldsymbol{Z}_N) = \frac{1}{L}\sum_{l=1}^{L}\frac{L}{N}\sum_{\boldsymbol{z}\in\boldsymbol{Z}^l_{N/L}}\mathcal{L}\left(y, g\left(\boldsymbol{x}; \boldsymbol{w}_l\left(\boldsymbol{Z}_N \setminus \boldsymbol{Z}^l_{N/L}\right)\right)\right)$$

  where $\boldsymbol{w}_l$ are the model parameters selected when removing the l-th fold.

- The risk for the l-th fold is:

$$R_{\mathrm{L-cv},l}(\boldsymbol{Z}_l) = \frac{L}{N}\sum_{\boldsymbol{z}\in\boldsymbol{Z}^l_{N/L}}\mathcal{L}\left(y, g\left(\boldsymbol{x}; \boldsymbol{w}_l\left(\boldsymbol{Z}_N \setminus \boldsymbol{Z}^l_{N/L}\right)\right)\right)$$

1 / Number of data points in one fold

Data points in test set

Weights selected L-1 folds

Data with one fold removed

Empirical estimation of Generalization Error: L-fold cross-validation

- Some explanations before:
  - If the data is iid, then:

$$\mathrm{E}_{\boldsymbol{z}}\left[f(\boldsymbol{z})\right] = \frac{1}{k}\sum_{i=1}^{k}\mathrm{E}_{\boldsymbol{z}}\left[f(\boldsymbol{z}^i)\right] = \mathrm{E}_{\boldsymbol{Z}_k}\left[\frac{1}{k}\sum_{i=1}^{k} f(\boldsymbol{z}^i)\right]$$

    where z^i are "copies" of the random variable z
  - Remember what the variables mean:
    Z_N : set with N elements; full training set

    $Z_{N/L}^l$: set with N/L elements; l-th hold-out set

    $Z_{N(1-1/L)}$: set with $N - N/L$ elements; e.g. 4/5 of data

    $Z_N \setminus Z_{N/L}^l$: full set without the l-th hold-out set

Empirical estimation of Generalization Error: L-fold cross-validation

- Statement “CV estimate for risk is almost unbiased”

$$\mathrm{E}_{\boldsymbol{Z}_{N(1-1/L)}}\left[R_{\mathrm{gen}}\left(g\left(.;\boldsymbol{w}\left(\boldsymbol{Z}_{N(1-1/L)}\right)\right)\right)\right] = \mathrm{E}_{\boldsymbol{Z}_N}\left[R_{L-\mathrm{cv}}\left(\boldsymbol{Z}_N\right)\right]$$

- Left hand side:

$$\begin{aligned}
&\mathrm{E}_{\boldsymbol{Z}_{N(1-1/L)}}\left[R_{\mathrm{gen}}\left(g\left(.;\boldsymbol{w}\left(\boldsymbol{Z}_{N(1-1/L)}\right)\right)\right)\right] \\
&\quad = \mathrm{E}_{\boldsymbol{Z}_{N(1-1/L)}\cup\boldsymbol{z}}\left[\mathcal{L}\left(y, g\left(\boldsymbol{x};\boldsymbol{w}\left(\boldsymbol{Z}_{N(1-1/L)}\right)\right)\right)\right] \\
&\quad = \mathrm{E}_{\boldsymbol{Z}_{N(1-1/L)}}\mathrm{E}_{\boldsymbol{Z}_{N/L}}\left[\frac{L}{N}\sum_{\boldsymbol{z}\in\boldsymbol{Z}_{N/L}}\mathcal{L}\left(y, g\left(\boldsymbol{x};\boldsymbol{w}\left(\boldsymbol{Z}_{N(1-1/L)}\right)\right)\right)\right].
\end{aligned}$$

- We used
  - that the data are iid
  - definition of generalization error

Empirical estimation of Generalization Error: L-fold cross-validation

- Statement “CV estimate for risk is almost unbiased”

$$\mathrm{E}_{\boldsymbol{Z}_{N(1-1/L)}}\left[R_{\mathrm{gen}}\left(g\left(.;\boldsymbol{w}\left(\boldsymbol{Z}_{N(1-1/L)}\right)\right)\right)\right]=\mathrm{E}_{\boldsymbol{Z}_N}\left[R_{L-\mathrm{cv}}\left(\boldsymbol{Z}_N\right)\right]$$

- Right hand side:

$$\begin{aligned}\mathrm{E}_{\boldsymbol{Z}_N}\left[R_{L-\mathrm{cv}}\left(\boldsymbol{Z}_N\right)\right] &= \mathrm{E}_{\boldsymbol{Z}_N}\left[\frac{1}{L}\sum_{l=1}^{L}\frac{L}{N}\sum_{\boldsymbol{z}\in\boldsymbol{Z}^l_{N/L}}\mathcal{L}\left(y,g\left(\boldsymbol{x};\boldsymbol{w}_l\left(\boldsymbol{Z}_N\setminus\boldsymbol{Z}^l_{N/L}\right)\right)\right)\right]\\ &= \frac{1}{L}\sum_{l=1}^{L}\mathrm{E}_{\boldsymbol{Z}_N}\left[\frac{L}{N}\sum_{\boldsymbol{z}\in\boldsymbol{Z}^l_{N/L}}\mathcal{L}\left(y,g\left(\boldsymbol{x};\boldsymbol{w}_l\left(\boldsymbol{Z}_N\setminus\boldsymbol{Z}^l_{N/L}\right)\right)\right)\right]\\ &= \mathrm{E}_{\boldsymbol{Z}_{N(1-1/L)}}\mathrm{E}_{\boldsymbol{Z}_{N/L}}\left[\frac{L}{N}\sum_{\boldsymbol{z}\in\boldsymbol{Z}_{N/L}}\mathcal{L}\left(y,g\left(\boldsymbol{x};\boldsymbol{w}\left(\boldsymbol{Z}_{N(1-1/L)}\right)\right)\right)\right]\end{aligned}$$

Remark: Regularization to reduce overfitting (chapter 8)

- Adding a regularization term for the error term to discourage high coefficients:

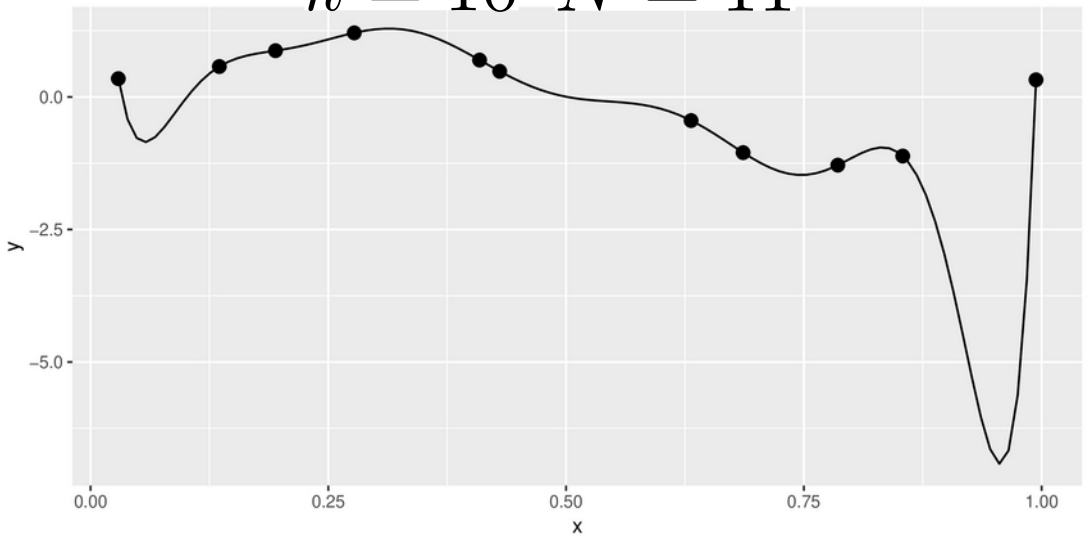
$$R_{\text{reg}}(\boldsymbol{w}) = \frac{1}{2}\sum_{n=1}^{N}(g(\boldsymbol{x}^n, \boldsymbol{w}) - y^n)^2 + \frac{\lambda}{2}\|\boldsymbol{w}\|^2,$$

  where $\|\boldsymbol{w}\|^2 = \boldsymbol{w}^T\boldsymbol{w} = w_0^2 + w_1^2 + \ldots + w_k^2$.

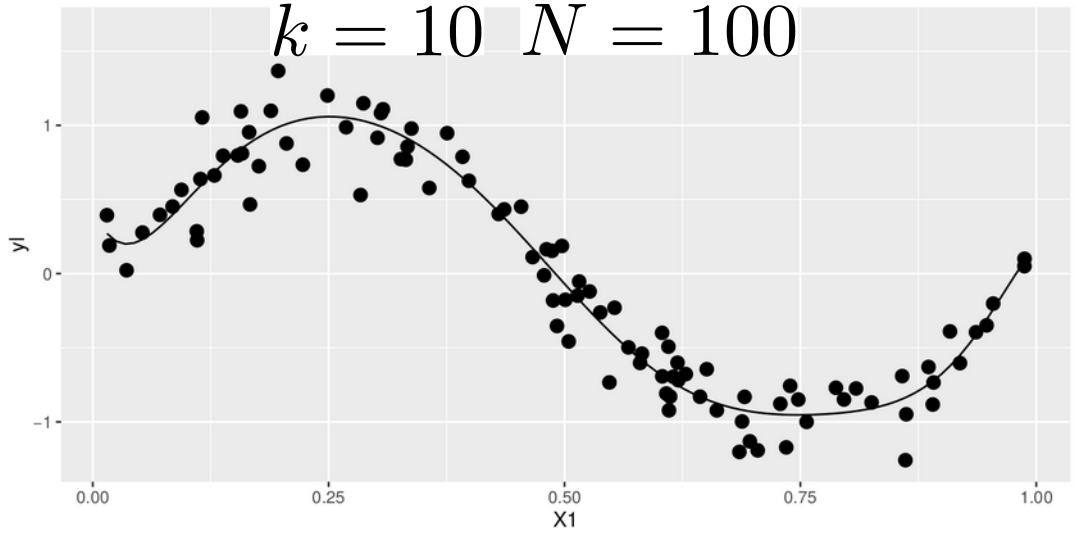
  The parameter λ governs the relative importance of the penalty term versus the error term. λ is a so-called *hyperparameter*.

- **Task:** Test the effect of the regularization term on our model!
- A large number of data points also allows for a more complex model

$k = 10 \;\; N = 11$



$k = 10 \;\; N = 100$



Remark: polynomial regression and its relation to linear regression

- Polynomial regression: one-dimensional input x

$$g(\boldsymbol{x}, \boldsymbol{w}) = \boldsymbol{w}^T \boldsymbol{x},$$

$$\boldsymbol{w} = (w_0, w_1, w_2, ..., w_k)$$

$$\boldsymbol{x} = (1, x, x^2, \ldots, x^k)$$

- Linear regression: multi-dimensional input $\boldsymbol{x} = (x_1, \ldots, x_D)$

$$g(\boldsymbol{x}, \boldsymbol{w}) = \boldsymbol{w}^T \boldsymbol{x},$$

$$\boldsymbol{w} = (w_0, w_1, w_2, ..., w_D)$$

$$\boldsymbol{x} = (1, x_1, x_2, \ldots, x_D)$$

Remark: which functions can we approximate with ML methods? universal function approximator theorem

- If we have a “complicated function” – can we still approximate it?
- Yes, the “universal function approximator theorem” states that a neural network with a single hidden layer can approximate continuous functions on compact subsets of Rn (under mild assumptions)
- However: It remains unclear whether the parameters of the neural network can be found (learned)
- See later in the lectures on neural networks

Task: solving the optimization problem

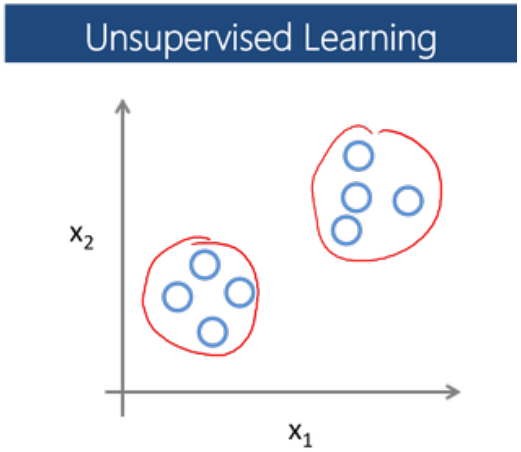
For a given data set $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ find the parameter vector $\boldsymbol{w}$ that solves the following optimization problem:

$$\min_{\boldsymbol{w}} R_{\text{emp}}(\boldsymbol{w}) = \min_{\boldsymbol{w}} \frac{1}{2} \sum_{n=1}^{N} (\boldsymbol{w}^T \boldsymbol{x}^n - y^n)^2,$$

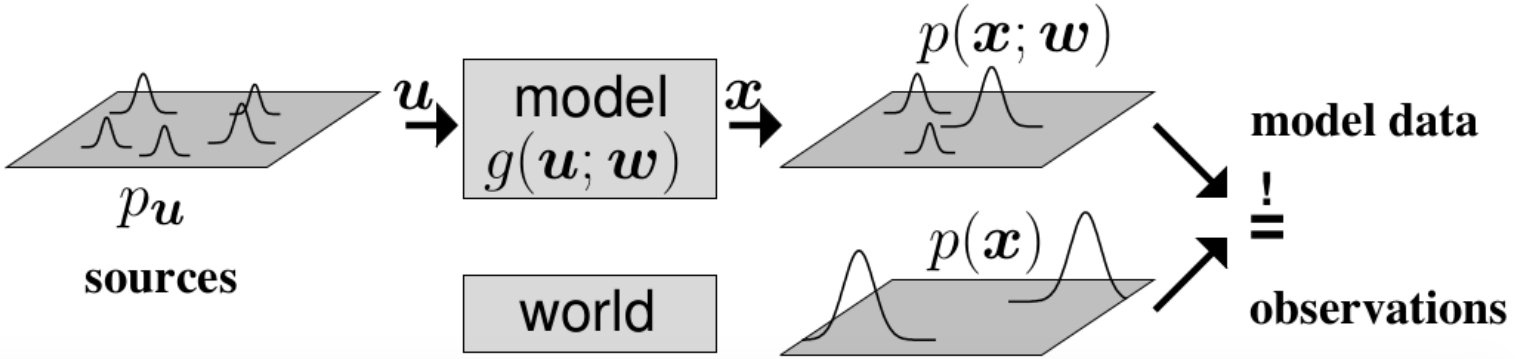
where $\boldsymbol{w} = (w_0, w_1, w_2, ..., w_k)$ and $\boldsymbol{x} = (1, x, x^2, \ldots, x^k)$ and $k + 1 < n$.

We already know how to solve this problem...

Loss for unsupervised tasks

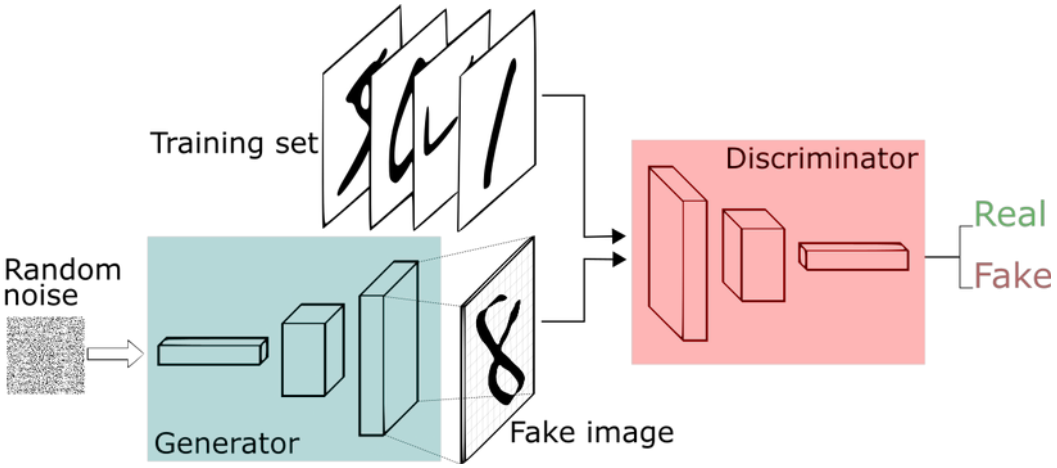


- Likelihood of the data
- Projection methods:
  - low dimensionality and information loss
  - independence of the components
- Generative models: approximate the distribution of the real data



Learned loss functions (GANs)

- Generative adversarial networks learn a loss function between real and generated data
- Mean squared error is not a good loss for a model that should generate realistic images (why?)



Ok, a new problem now

- We want to optimize architectures:
  - Train arch1 with lr=0.1 on training data; loss on “test set”: 0.82
  - Train arch1 with lr=0.01 on training data; loss on “test set”: 0.75
  - Train arch2 with lr=0.1 on training data; loss on “test set”: 0.72
  - Train arch2 with lr=0.01 on training data; loss on “test set”: 0.89
- Ok, we select arch2 with lr=0.1
  - Is 0.72 a “good” estimate for the generalization error?
  - Why/why not?

Validation set: optimizing hyperparameters (architecture)

- To avoid a *hyperparameter selection bias,* we have to optimize the hyperparameters/architecture on a validation set
- The validation set is part of the training set
- Evaluation on test set only after hyperparameters is selected and model is trained
- How can we do this in a cross-validation setting?

Nested cross-validation: estimating performance for

```
Define set of hyperparameter combinations C
Divide data in L folds

# outer loop
for fold l in L folds:
    set fold l as test set
    for parameter combination c in C:

        # inner loop
        for fold m in remaining L-1 folds:
            set fold m as validation set
            train model on remaining L-2 folds
            evaluate model performance on fold m
        Calculate average performance over L-1 folds for hyperparam c

    train model on L-1 folds using hyperparams with best avg.
    performance over all steps of the inner loop

    evaluate model performance on fold l

Calculate average performance over L folds
```

Summary

- *Training set*: for optimizing parameters(weights) of the model
- *Validation set*: for optimizing the architecture/hyperparams of the model
- *Test set*: for estimating the generalization error

- *Cross-validation*: estimate performance of a model (without hyperparam selection)
- *Nested cross-validation*: estimate performance of a model (with hyperparam selection)

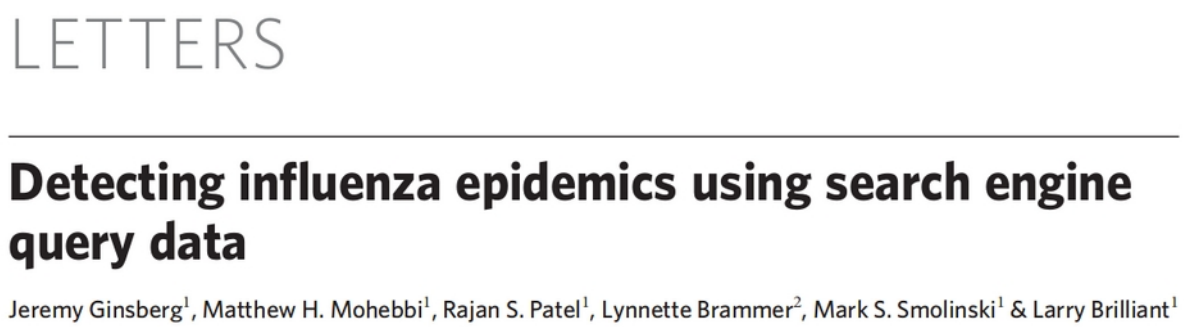
Recap: concepts and terminology

$$g(\boldsymbol{x}; \boldsymbol{w})$$

- Machine learning model
- Model class vs model
- Model parameters: *Learning is finding parameters*
- How can it be assessed how good a model is?
  - *Test set* or *cross validation* to assess generalization performance
- What are hyperparameters of a model?

Coming back to the start: The failure of Google flu trends

- In 2009, Google published a paper that states that flu trends can be predicted by search queries



LETTERS

Detecting influenza epidemics using search engine query data

Jeremy Ginsberg[1], Matthew H. Mohebbi[1], Rajan S. Patel[1], Lynnette Brammer[2], Mark S. Smolinski[1] & Larry Brilliant[1]

- The service repeatedly failed to correctly predict the trend (e.g. completely missing a non-seasonal flu)
- In 2015, the service was quietly abandoned
- They used top correlated search terms...
  - e.g. “basketball” was highly correlated