Wiederholung der Grundlagen

1. **Regression und Klassifikation**
2. Aufbau und Training von tiefen Neuronalen Netzen
3. Faltungsnetze
4. Fehlermaße der Klassifikation

Überwachtes Lernen

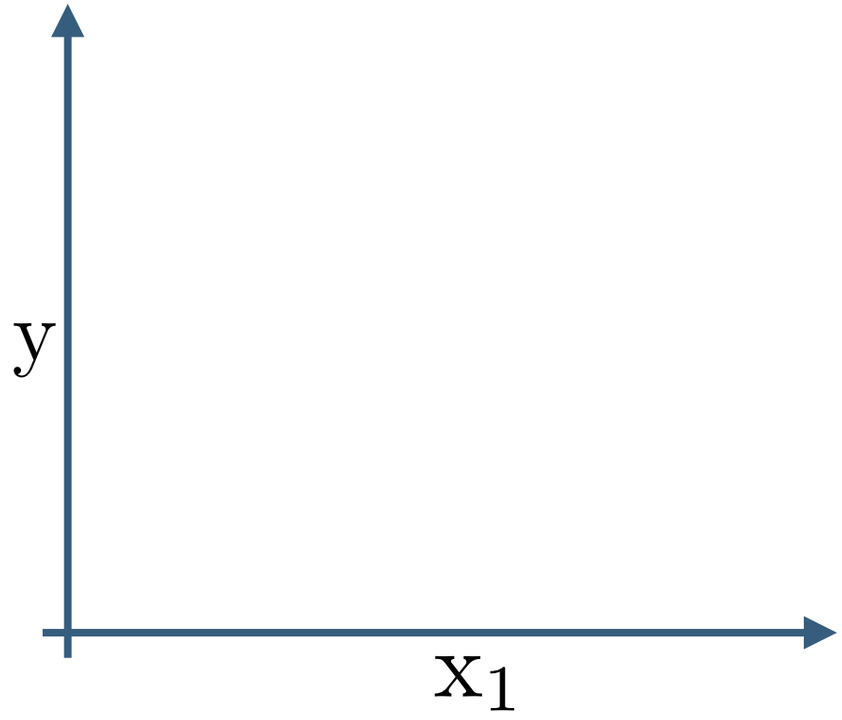
Klasse ist bekannt

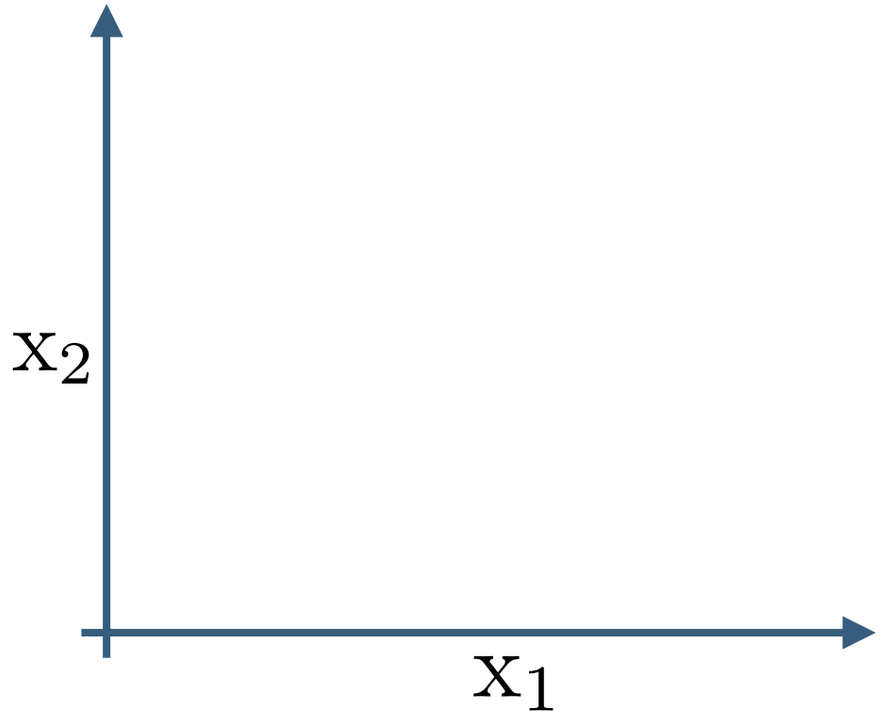
Input Variablen: $\mathbf{x}=\begin{pmatrix} \mathrm{x}_1 \\ \mathrm{x}_2 \\ \dots \\ \mathrm{x}_P \end{pmatrix}$

Output Variable: y

n^{th} Input Vektor: $\boldsymbol{x}^{(n)} = \begin{pmatrix} x_1^{(n)} \\ x_2^{(n)} \\ \dots \\ x_P^{(n)} \end{pmatrix}$

$n \in \{1, \dots, N\}$



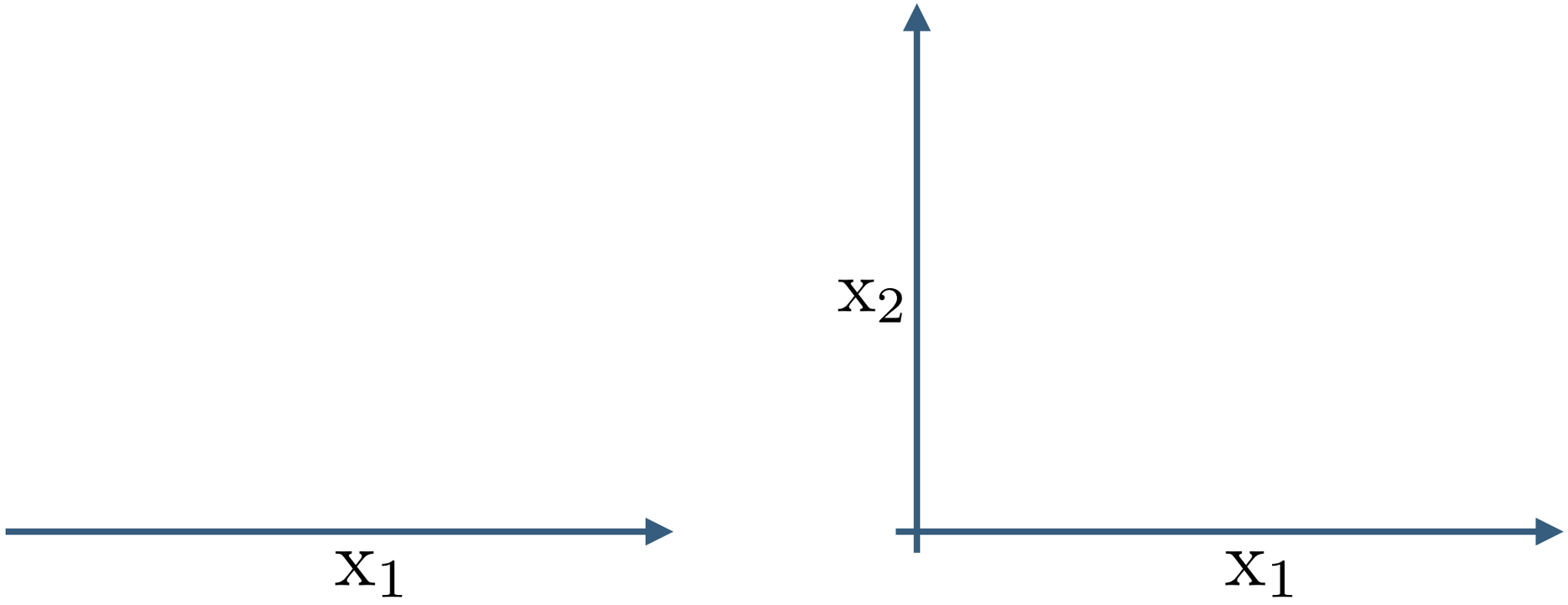


Unüberwachtes Lernen

Klasse ist unbekannt

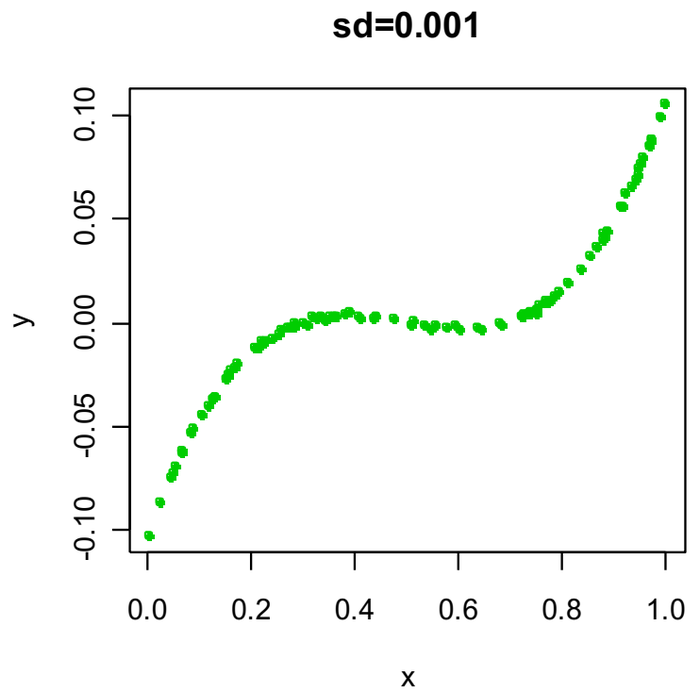
$$\text{Input Variablen: } \mathbf{x}=\begin{pmatrix} \mathrm{x}_1 \\ \mathrm{x}_2 \\ \dots \\ \mathrm{x}_P \end{pmatrix} \qquad n^{\text{th}} \text{ Input Vektor: } \boldsymbol{x}^{(n)} = \begin{pmatrix} x_1^{(n)} \\ x_2^{(n)} \\ \dots \\ x_P^{(n)} \end{pmatrix}$$

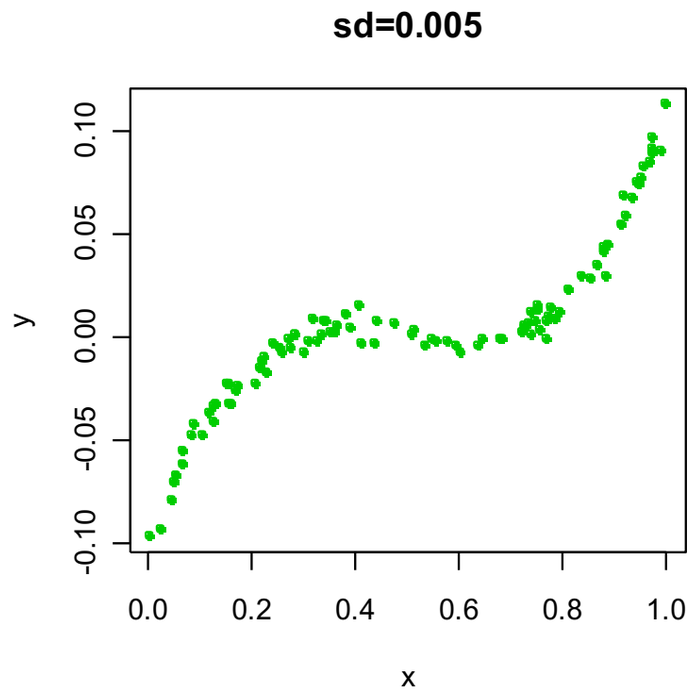
$$n \in \{1, \dots, N\}$$

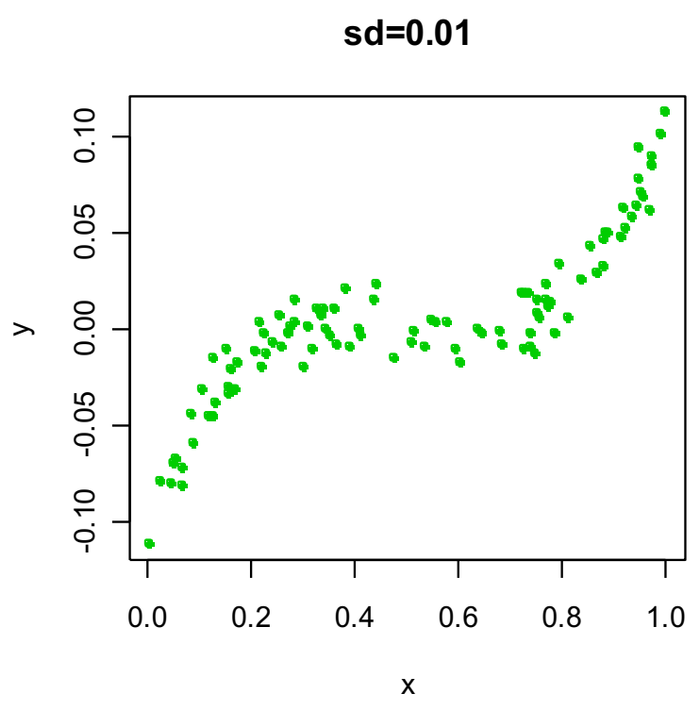


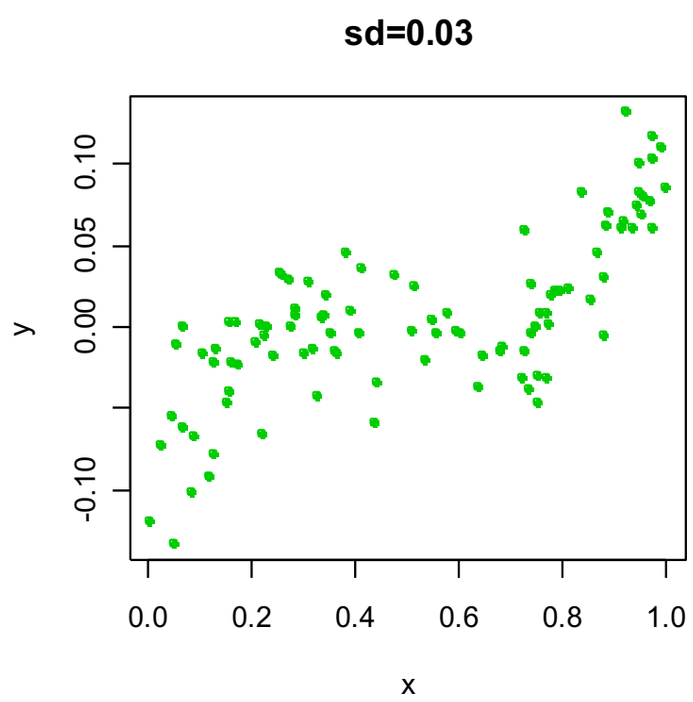
Regression

$$y = f(x) + \epsilon$$









Lernen und Fehlerfunktion

Schätzung der Funktion *f*

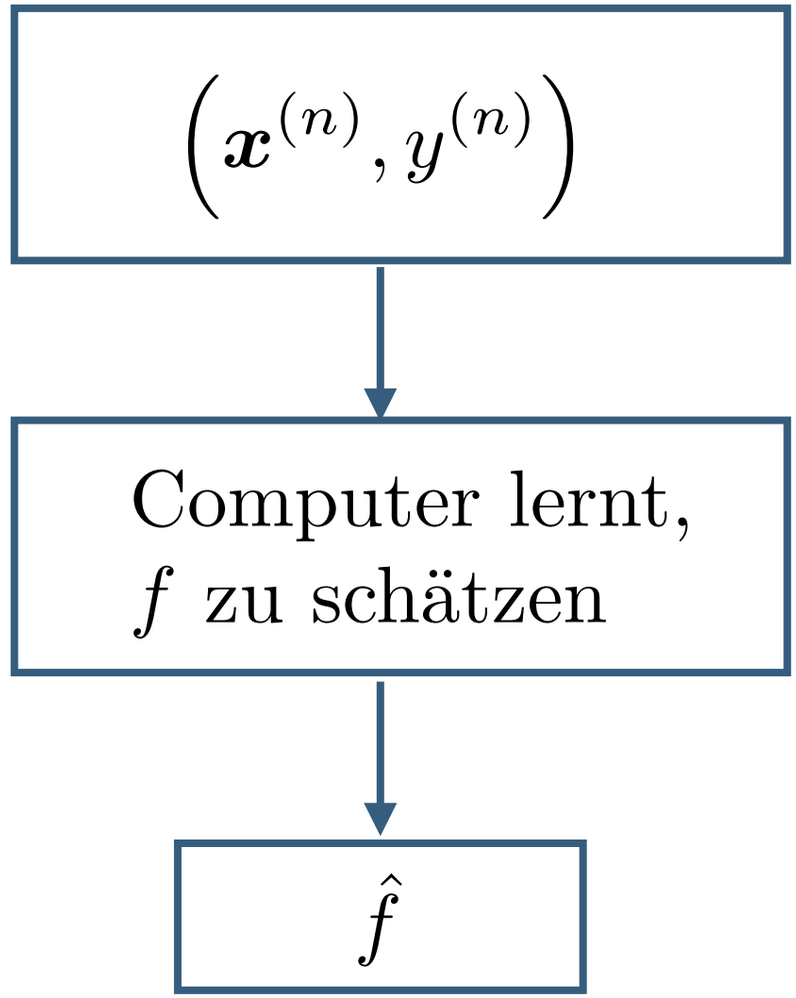
aus den Trainingsdaten

Fehlerfunktion

Schätzung vs. tatsächliche Funktion

Mittlerer quadratischer Fehler
(Mean squared error)

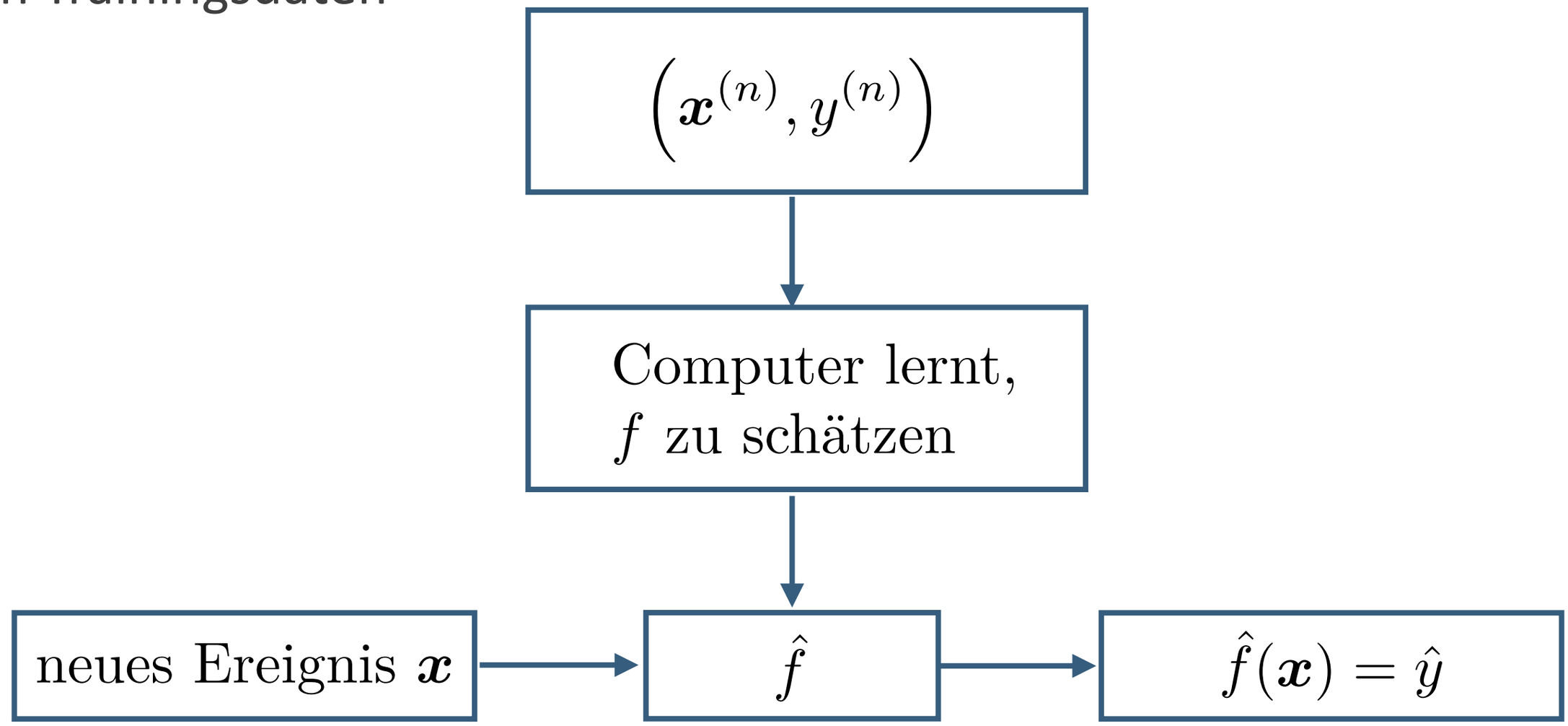
$$D_{\mathrm{MSE}} = \frac{1}{N} \sum_{n=1}^{N} \left(y^{(n)} - \hat{f}\left(\boldsymbol{x}^{(n)} \right) \right)^2$$



Lernen und Fehlerfunktion

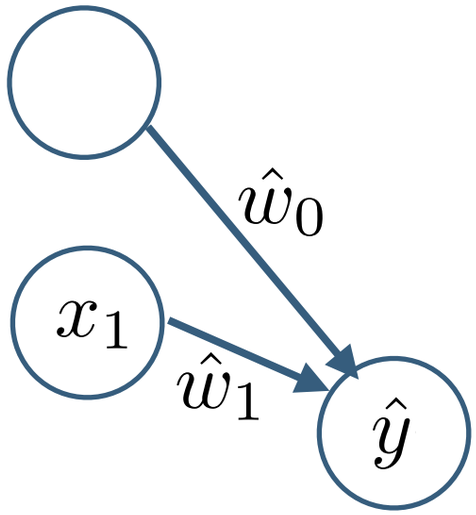
Schätzung der Funktion *f*

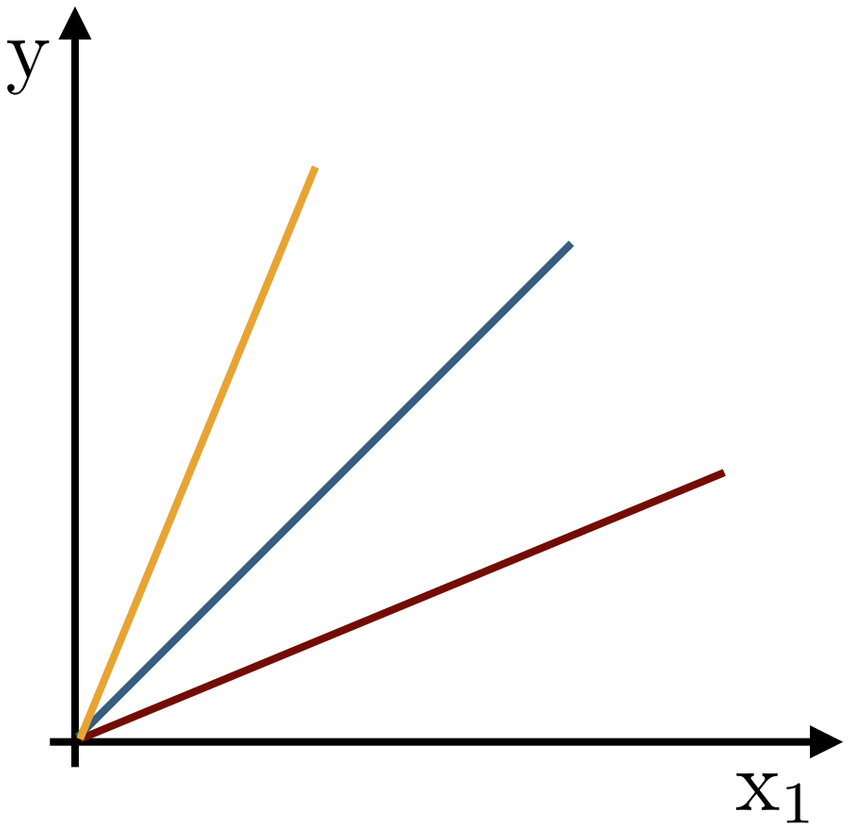
aus den Trainingsdaten

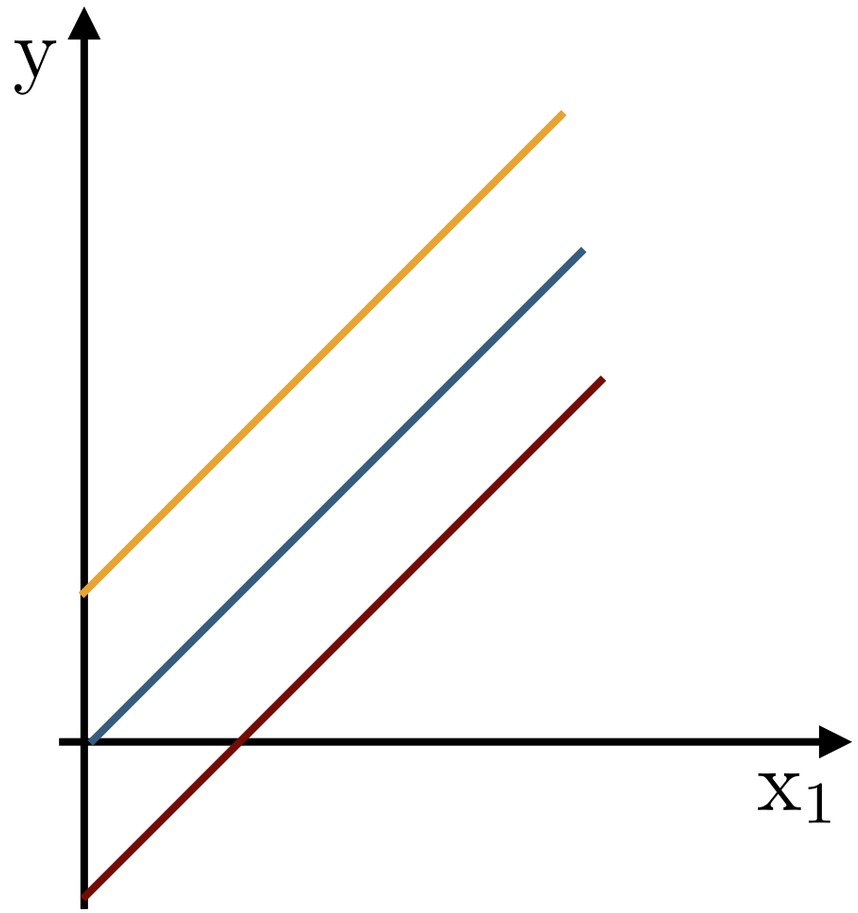


Lineare Regression

als Graph

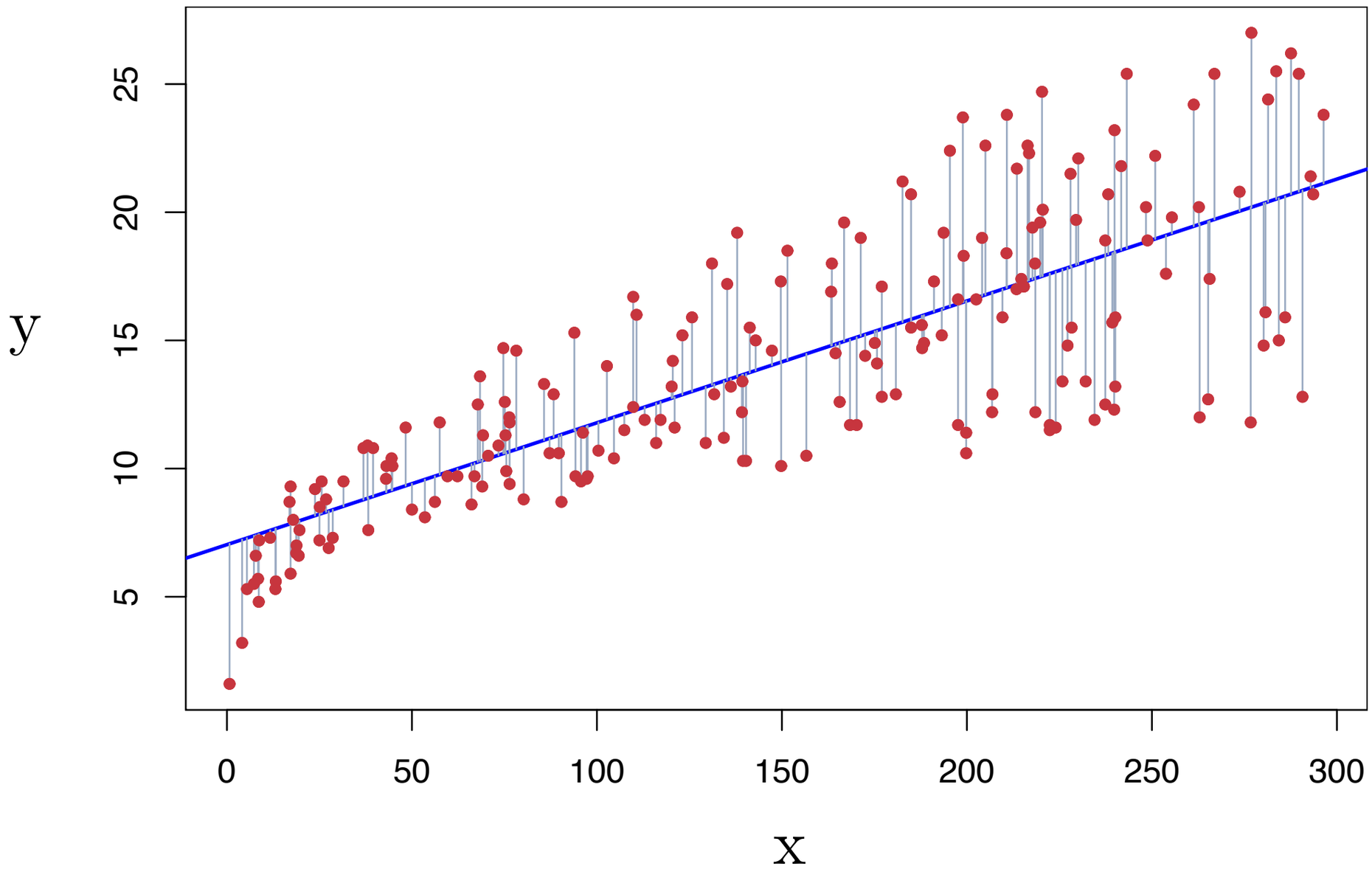






$$\hat{y} = \hat{w}_0 + \hat{w}_1 x_1$$

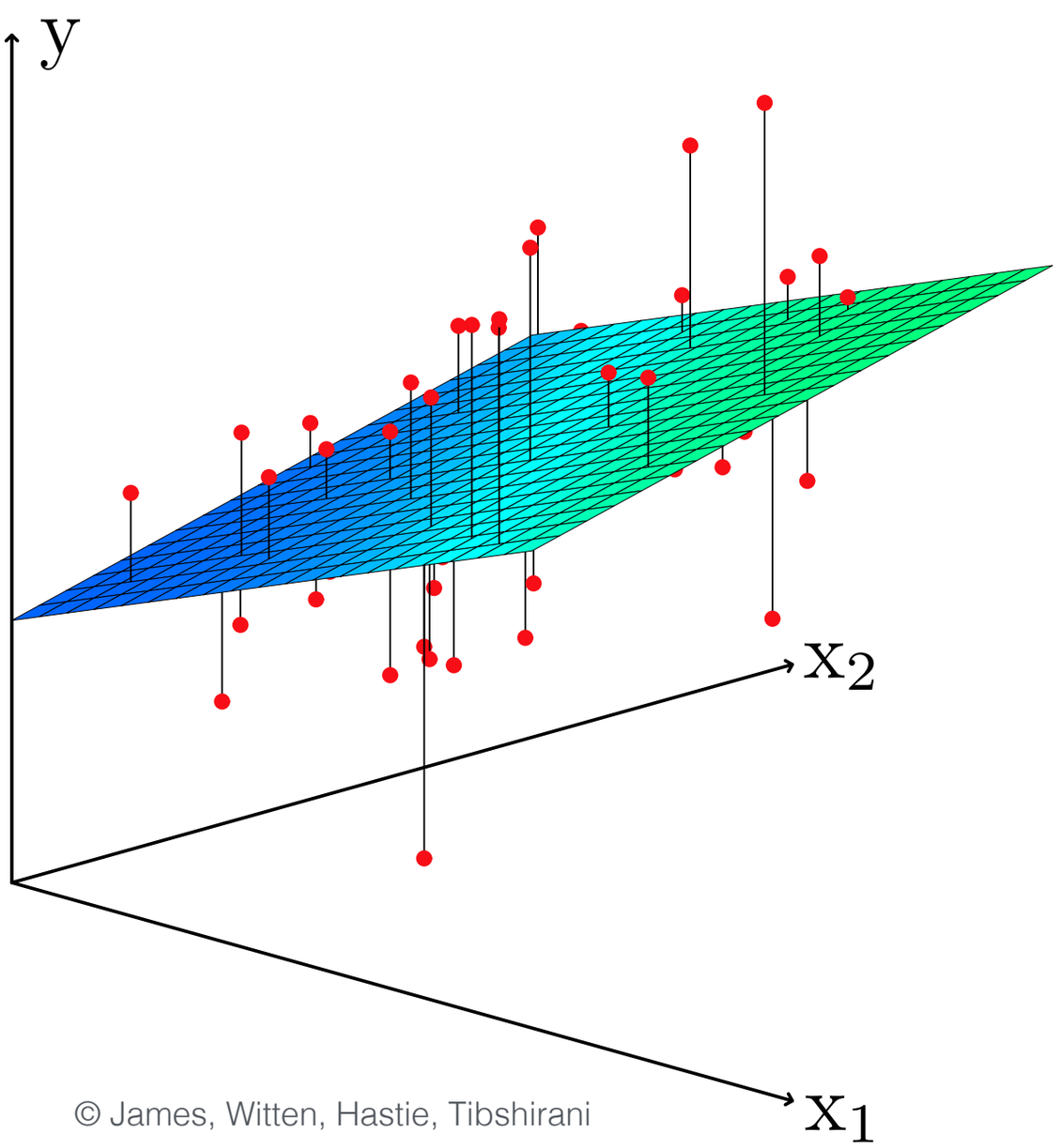
Lineare Regression



© James, Witten, Hastie, Tibshirani

Multiple Lineare Regression

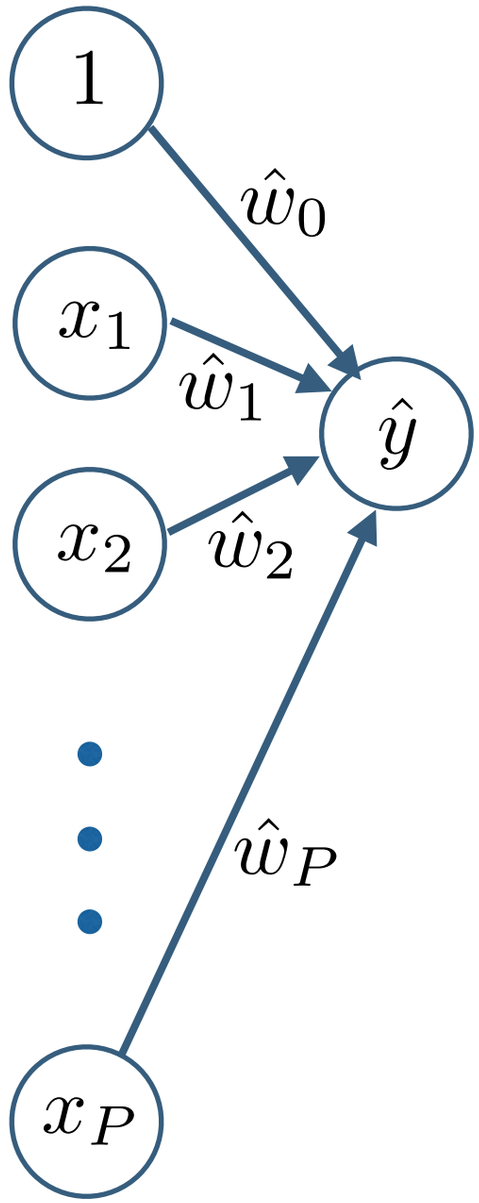
$$\hat{y} = \hat{w}_0 + \hat{w}_1 x_1 + \hat{w}_2 x_2 + \ldots + \hat{w}_P x_P$$



© James, Witten, Hastie, Tibshirani

Multiple Lineare Regression

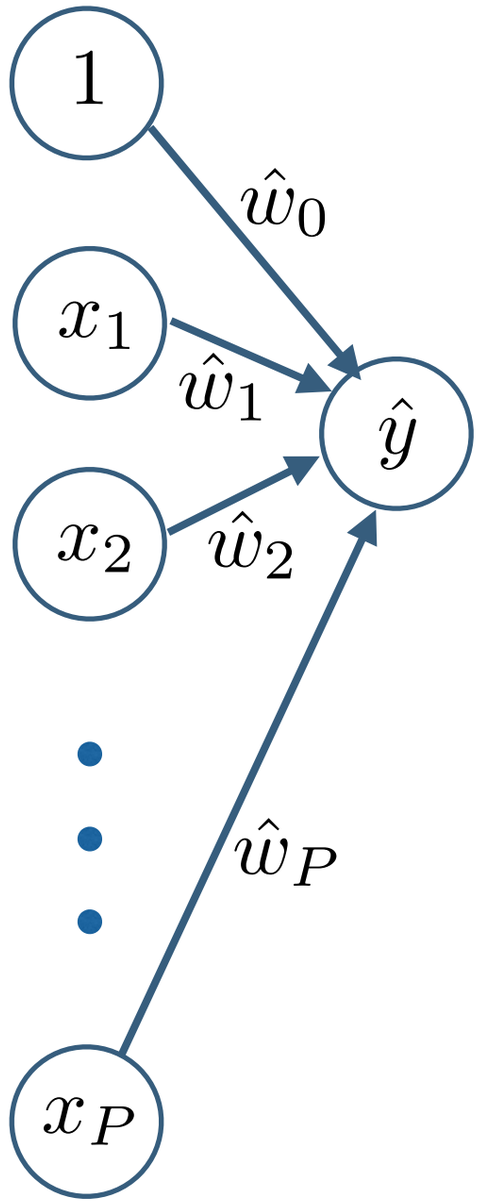
als Graph



$$\hat{y} = \hat{w}_0 + \hat{w}_1 x_1 + \hat{w}_2 x_2 + \ldots + \hat{w}_P x_P = \hat{\boldsymbol{w}}^T \cdot \boldsymbol{x}$$

Multiple Lineare Regression

als Graph



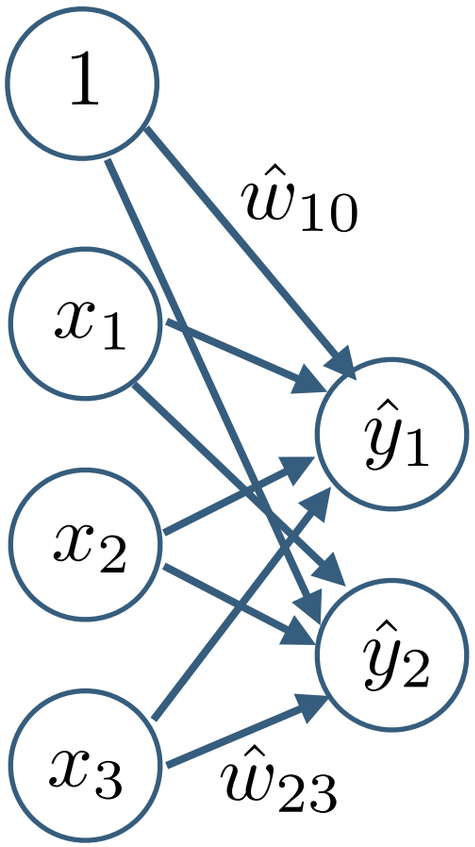
$$D_{\mathrm{MSE}} = \frac{1}{N} \sum_{n=1}^{N} \left(y^{(n)} - \hat{y}^{(n)} \right)^2$$

$$\text{mit } \hat{y}^{(n)} = \hat{\boldsymbol{w}}^T \boldsymbol{x}^{(n)}$$

$$\hat{y} = \hat{w}_0 + \hat{w}_1 x_1 + \hat{w}_2 x_2 + \ldots + \hat{w}_P x_P = \hat{\boldsymbol{w}}^T \cdot \boldsymbol{x}$$

Multiple Lineare Regression

mit mehr als einem Output

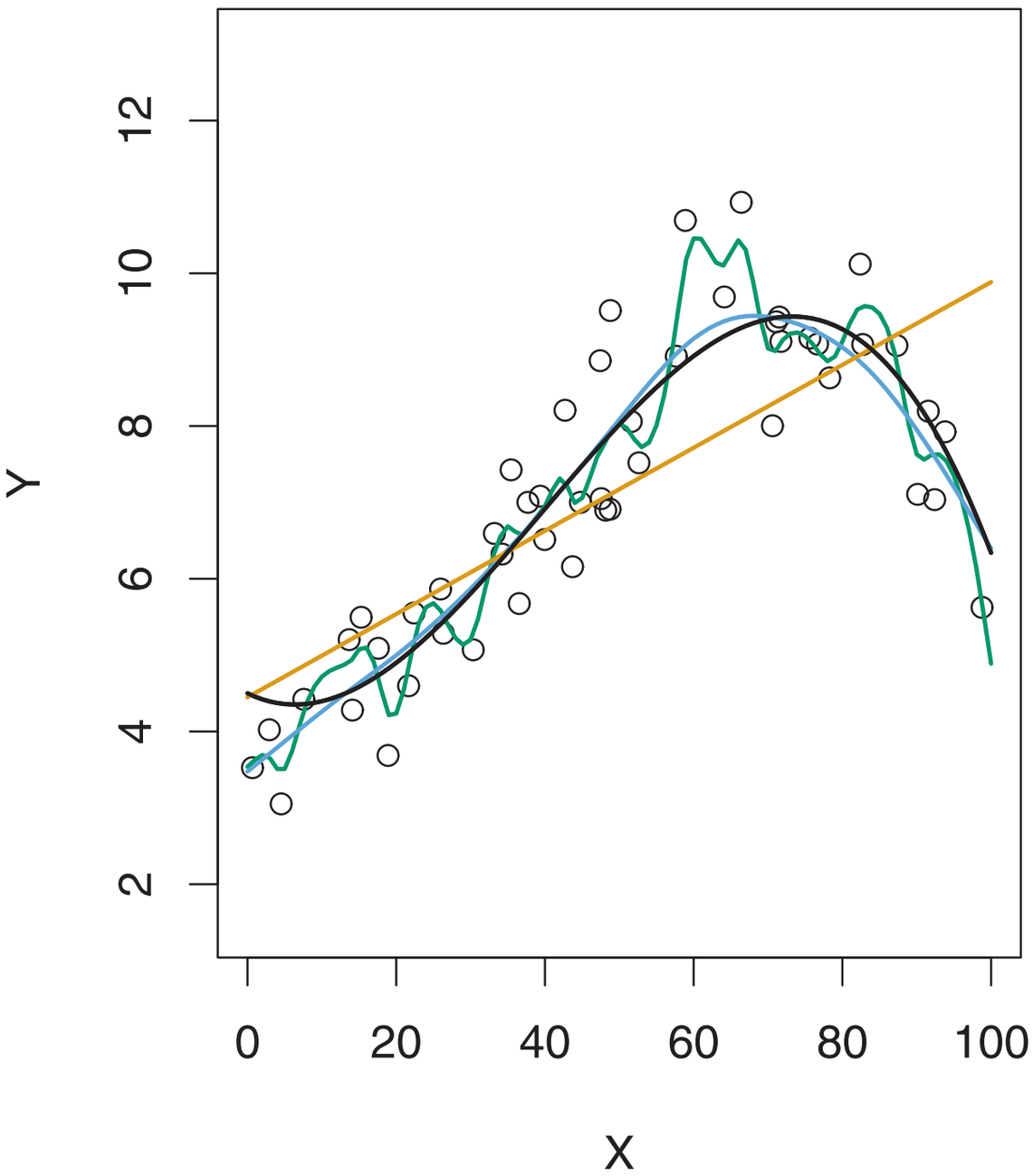


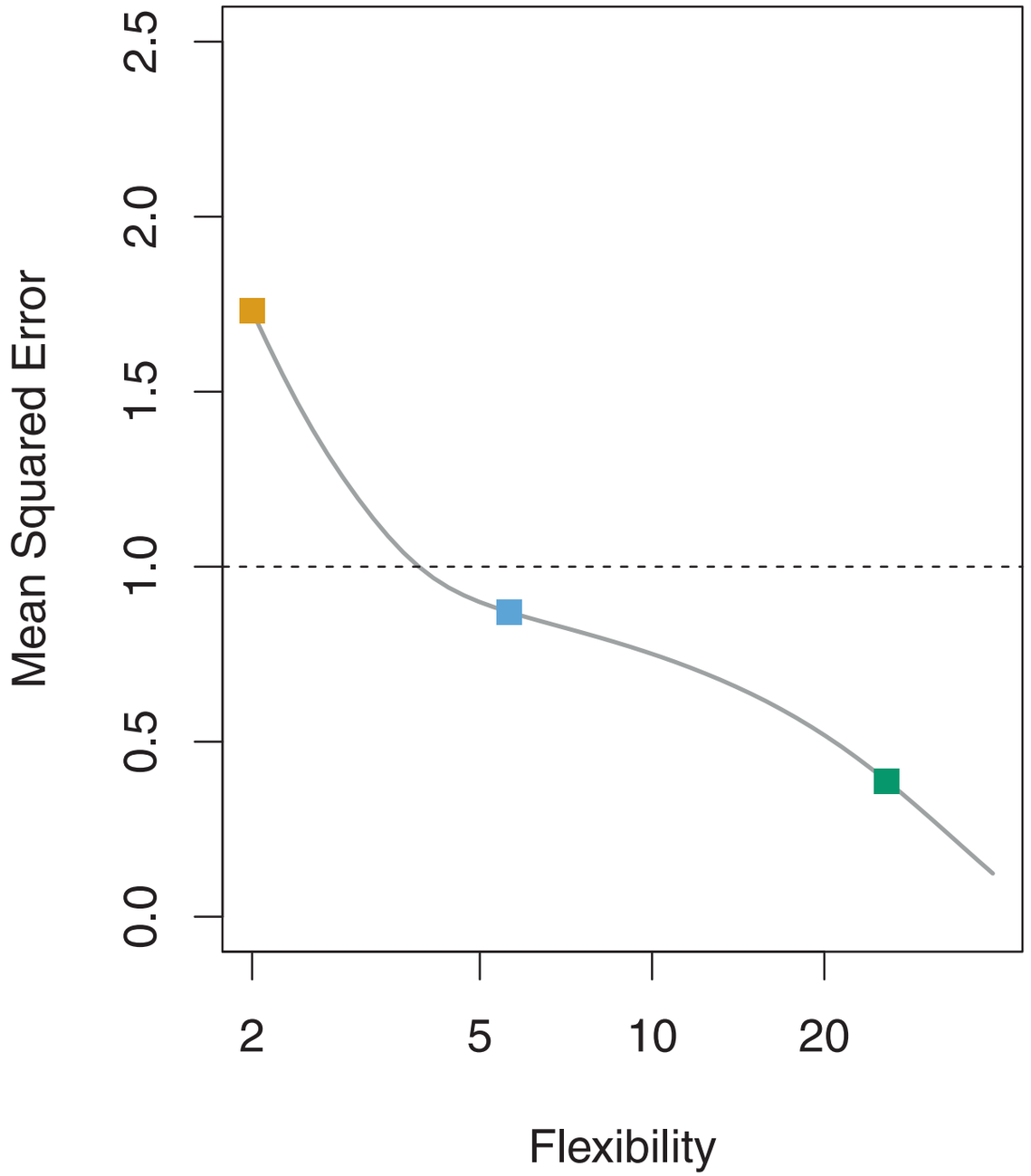
$$D_{\mathrm{MSE}} = \frac{1}{N}\sum_{n=1}^{N}\left(\boldsymbol{y}^{(n)} - \hat{\boldsymbol{y}}^{(n)}\right)^2$$

$$\text{mit } \hat{\boldsymbol{y}}^{(n)} = \hat{\boldsymbol{W}}^T\boldsymbol{x}^{(n)}$$

$$\begin{pmatrix} \hat{y}_1 \\ \hat{y}_2 \end{pmatrix} = \begin{pmatrix} \hat{w}_{10} & \hat{w}_{11} & \hat{w}_{12} & \hat{w}_{13} \\ \hat{w}_{20} & \hat{w}_{21} & \hat{w}_{22} & \hat{w}_{23} \end{pmatrix} \begin{pmatrix} 1 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix} = \widehat{\boldsymbol{W}}^T \cdot \boldsymbol{x}$$

Regression





Output

Regression

$$y \in \mathbb{R}$$

Klassifikation

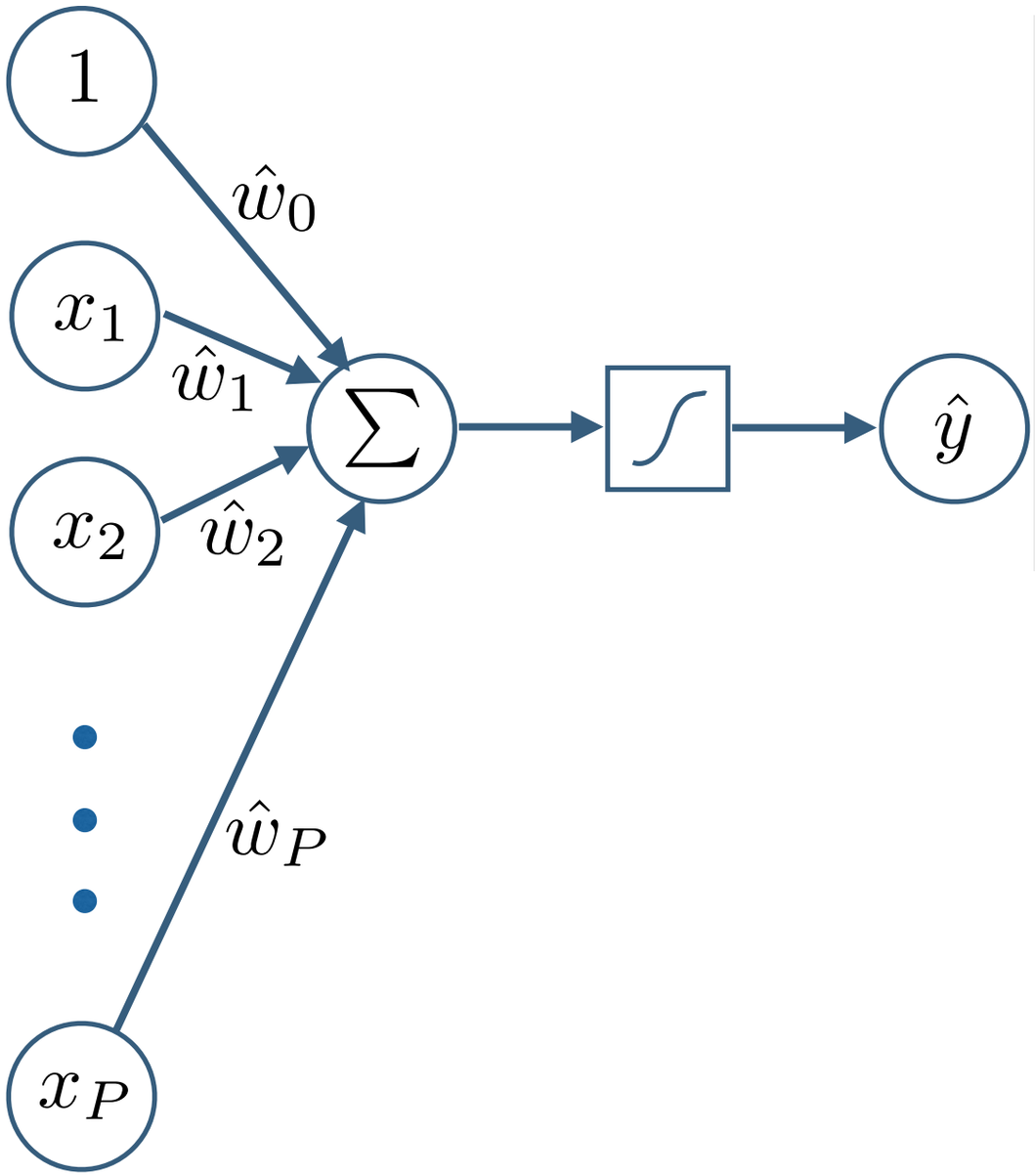
$$y \in \{0, 1\}$$

$$\boldsymbol{y} = (y_1, \ldots, y_K)^T \qquad \text{mit } y_k \in \{0, 1\}$$

Sei c die korrekte Klasse

$$\boldsymbol{y}_c = \boldsymbol{y} \text{ with } \begin{cases} y_k = 1 & \text{if } k = c \\ y_k = 0 & \text{if } k \neq c \end{cases}$$

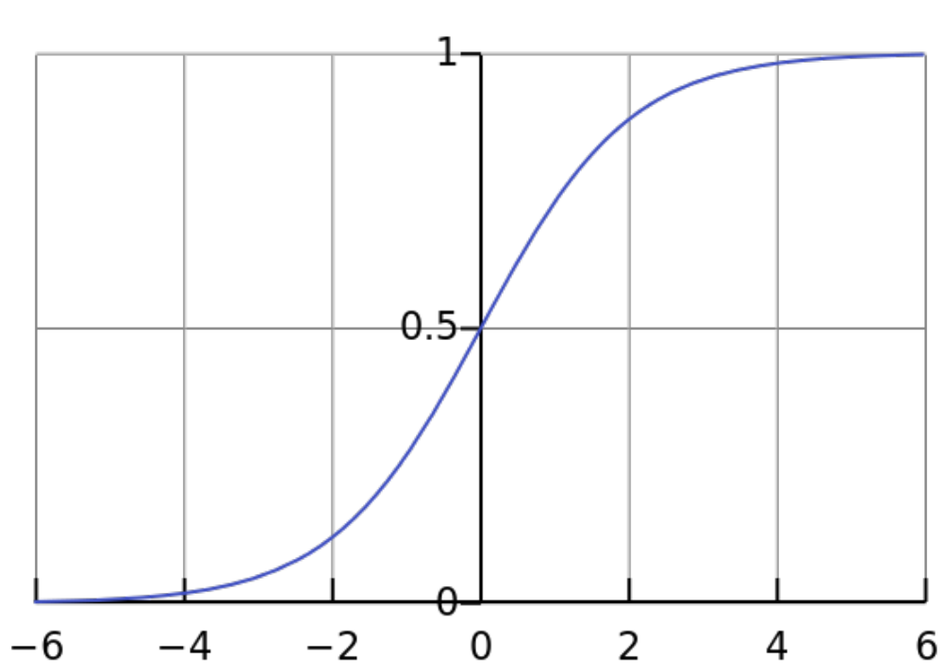
Logistische Regression



Binäre Klassifikation

$$\hat{y} = \begin{cases} 1 & \text{if } \sigma(a) > 0.5 \\ 0 & \text{else} \end{cases}$$

$$\sigma(a) = \frac{e^a}{1 + e^a} \text{ mit } a = \left(\hat{\boldsymbol{w}}^T \cdot \boldsymbol{x}\right)$$



© Wikipedia

Logistische Regression

Modell für a-posteriori Verteilung

$$p(\hat{y} = 1 | \boldsymbol{x}; \hat{\boldsymbol{w}}) = \sigma(a)$$

$$\sigma(a) = \frac{e^a}{1 + e^a} \text{ mit } a = \left(\hat{\boldsymbol{w}}^T \cdot \boldsymbol{x}\right)$$

Maximum Likelihood Schätzer

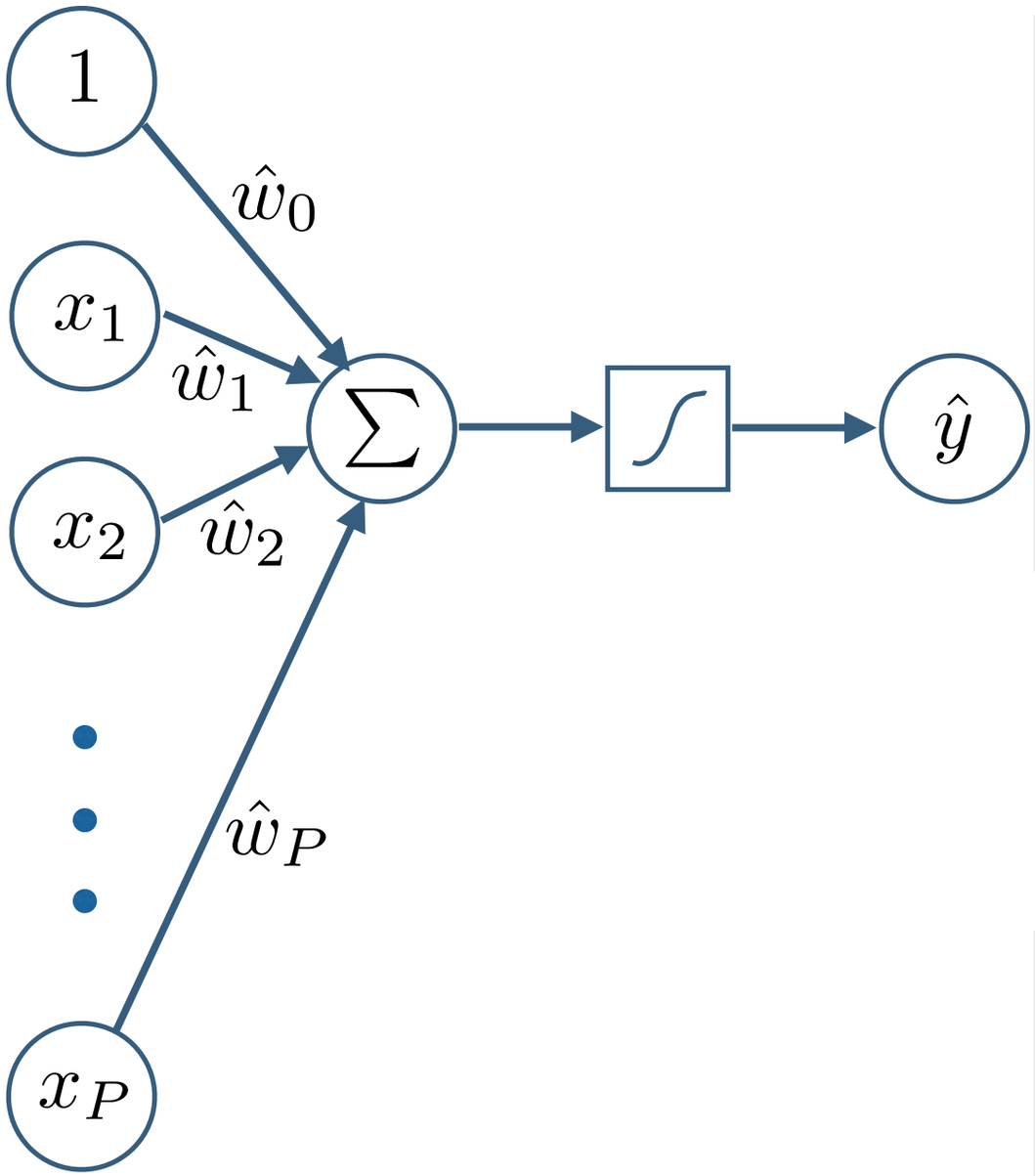
Likelihood Funktion:

$$p(\hat{y} | \boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(N)}; \hat{\boldsymbol{w}}) = \prod_n p\left(\hat{y} = 1 \middle| \boldsymbol{x}^{(n)}; \hat{\boldsymbol{w}}\right)^{y^{(n)}} \left[1 - p\left(\hat{y} = 1 \middle| \boldsymbol{x}^{(n)}; \hat{\boldsymbol{w}}\right)\right]^{1 - y^{(n)}}$$

Fehlerfunktion:

$$\begin{aligned} D_{\mathrm{CE}}(\hat{\boldsymbol{w}}) &= -\ln p(\hat{y} | \boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(N)}; \hat{\boldsymbol{w}}) \\ &= -\sum_{n=1}^{N} \ln\left(\sigma(a)^{y^{(n)}}\right) - \sum_{n=1}^{N} \ln\left(\left[1 - \sigma(a)\right]^{1 - y^{(n)}}\right) \end{aligned}$$

Logistische Regression



Binäre Klassifikation

$$\hat{y} = \begin{cases} 1 & \text{if } \sigma(a) > 0.5 \\ 0 & \text{else} \end{cases}$$

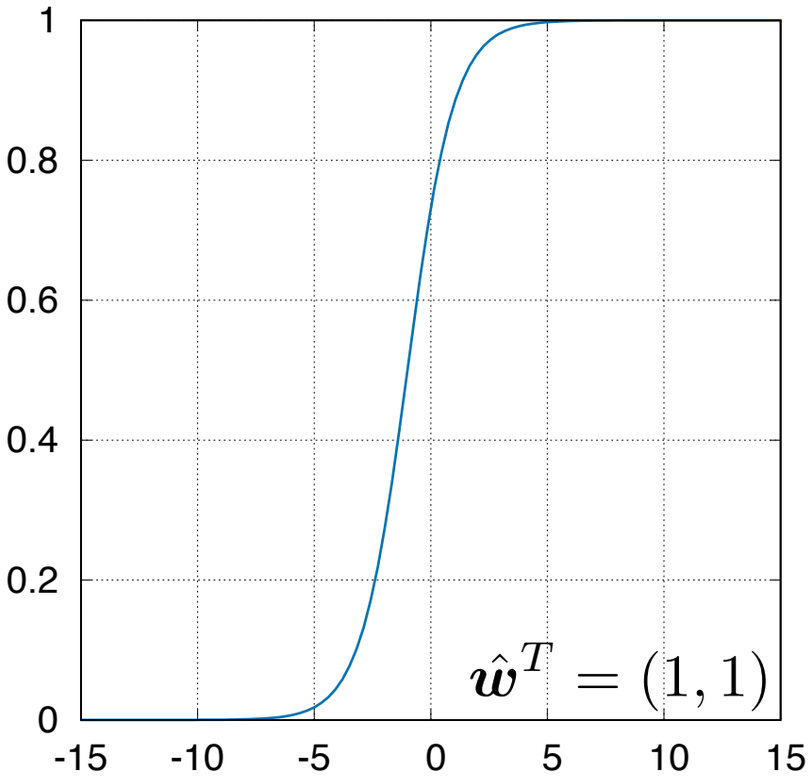
$$\sigma(a) = \frac{e^a}{1 + e^a} \text{ mit } a = \left(\hat{\boldsymbol{w}}^T \cdot \boldsymbol{x}\right)$$

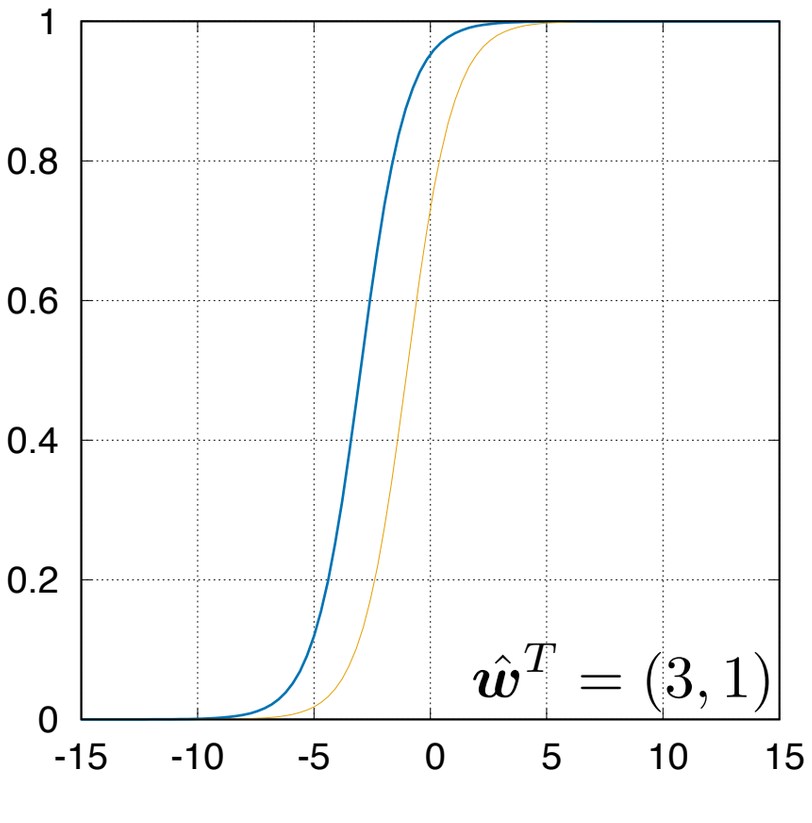
Optimierungsproblem

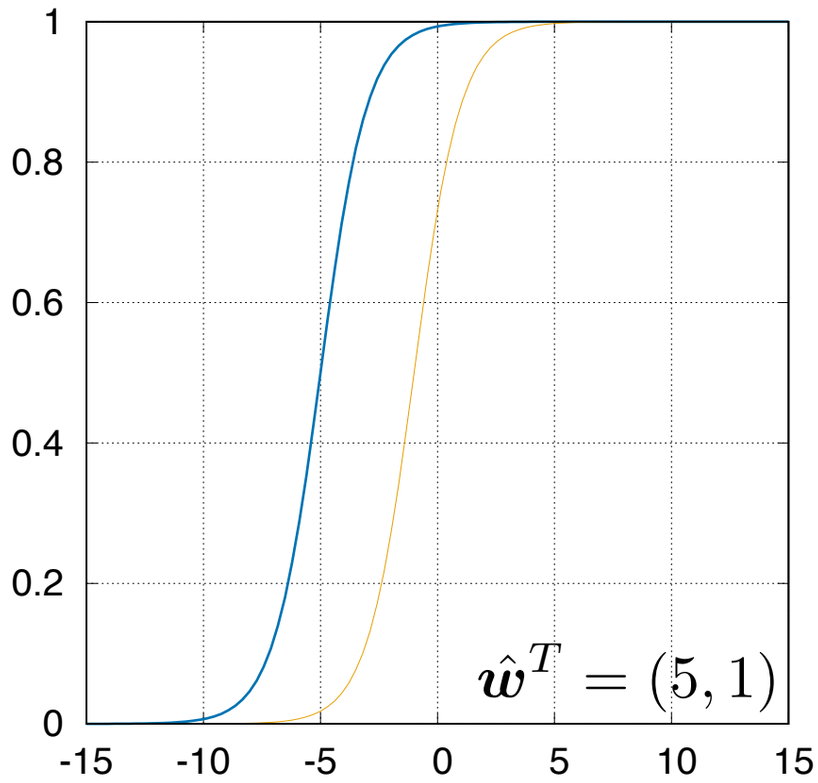
$$\underset{\mathbf{w}}{\operatorname{argmin}} = \left\{ D(\mathbf{w}) \,\middle|\, D(\hat{\boldsymbol{w}}) = -\ln p(\hat{y}|\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(N)}; \hat{\boldsymbol{w}}) \right\}$$

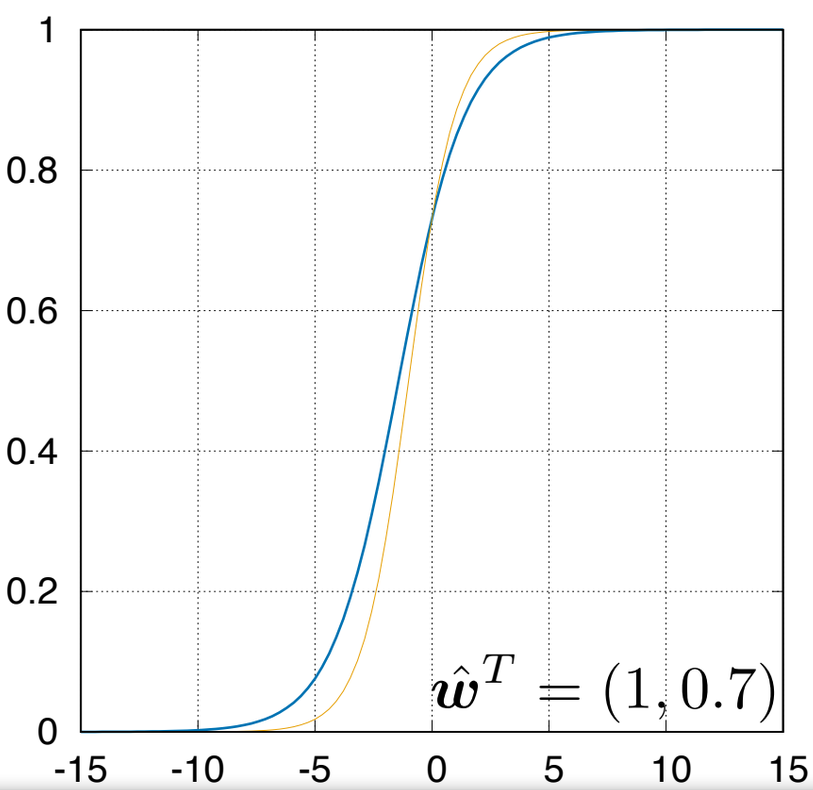
Logistische Regression

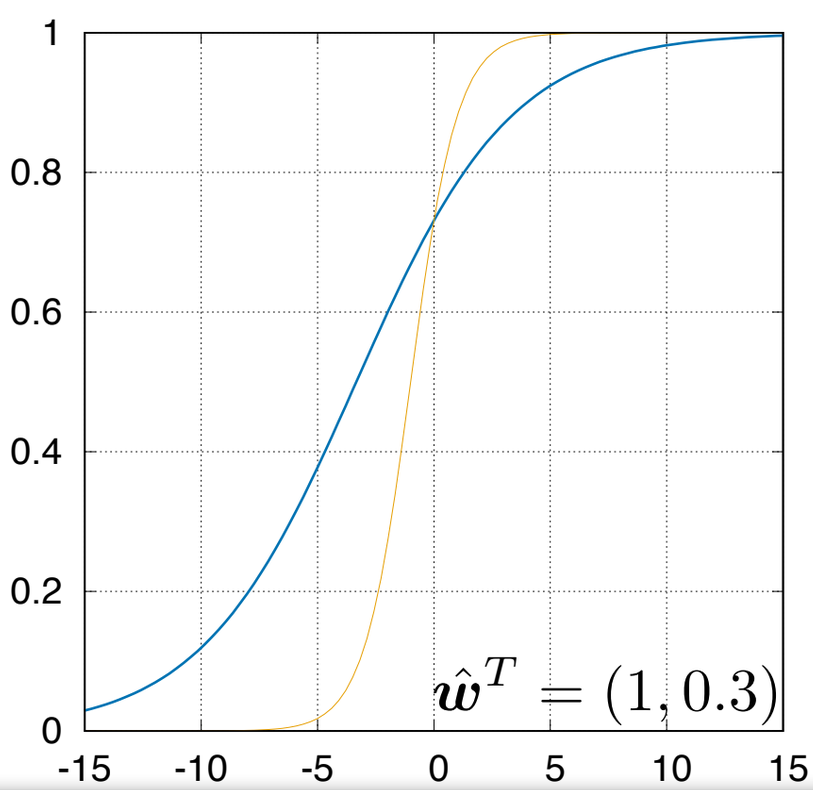
$$\sigma(a) = \frac{e^a}{1+e^a} \text{ mit } a = \left(\hat{\boldsymbol{w}}^T \cdot \boldsymbol{x}\right)$$







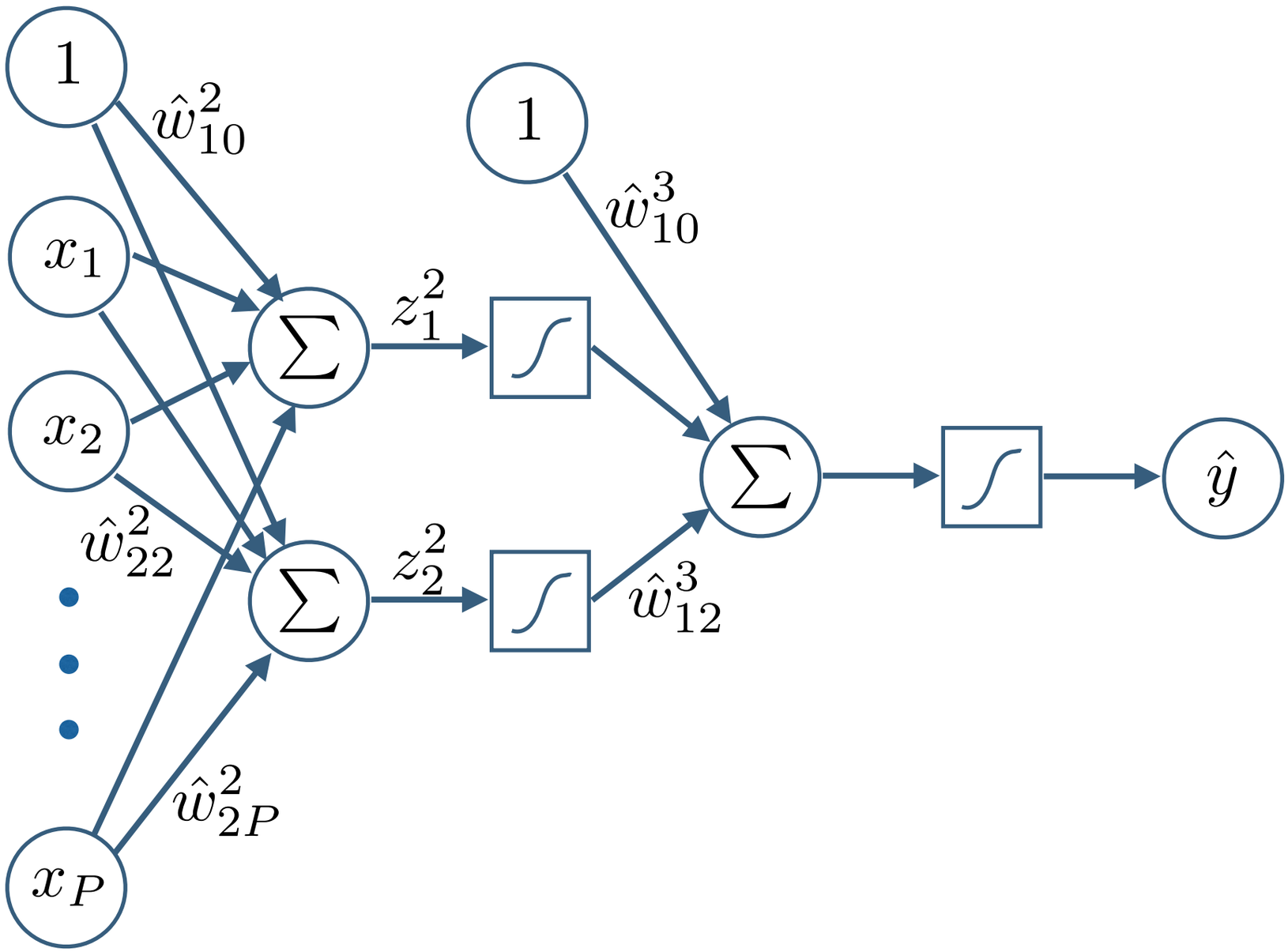




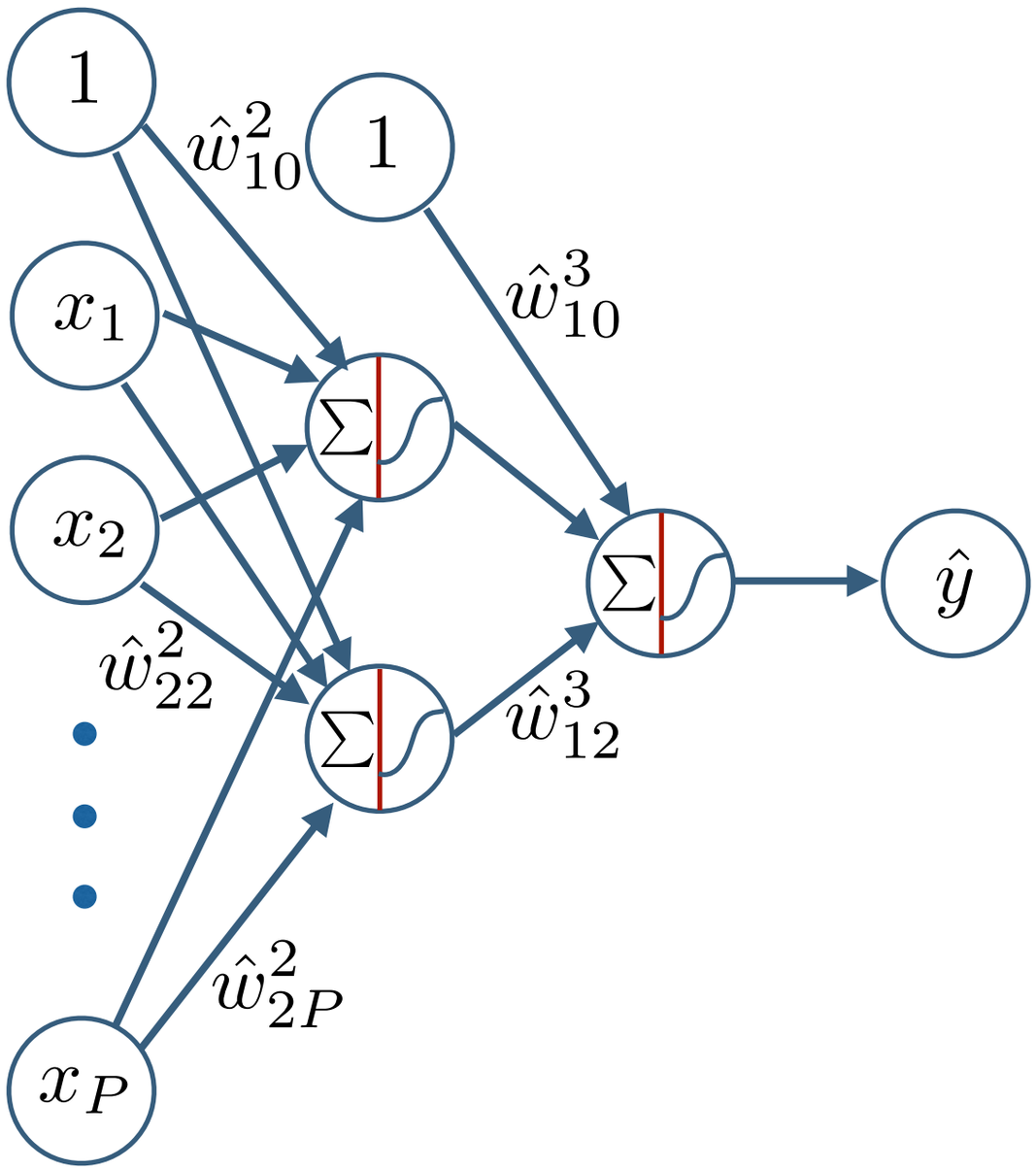
Wiederholung der Grundlagen

1. Regression und Klassifikation
2. **Aufbau und Training von tiefen Neuronalen Netzen**
3. Faltungsnetze
4. Fehlermaße der Klassifikation

Multi-layer Neuronales Netz

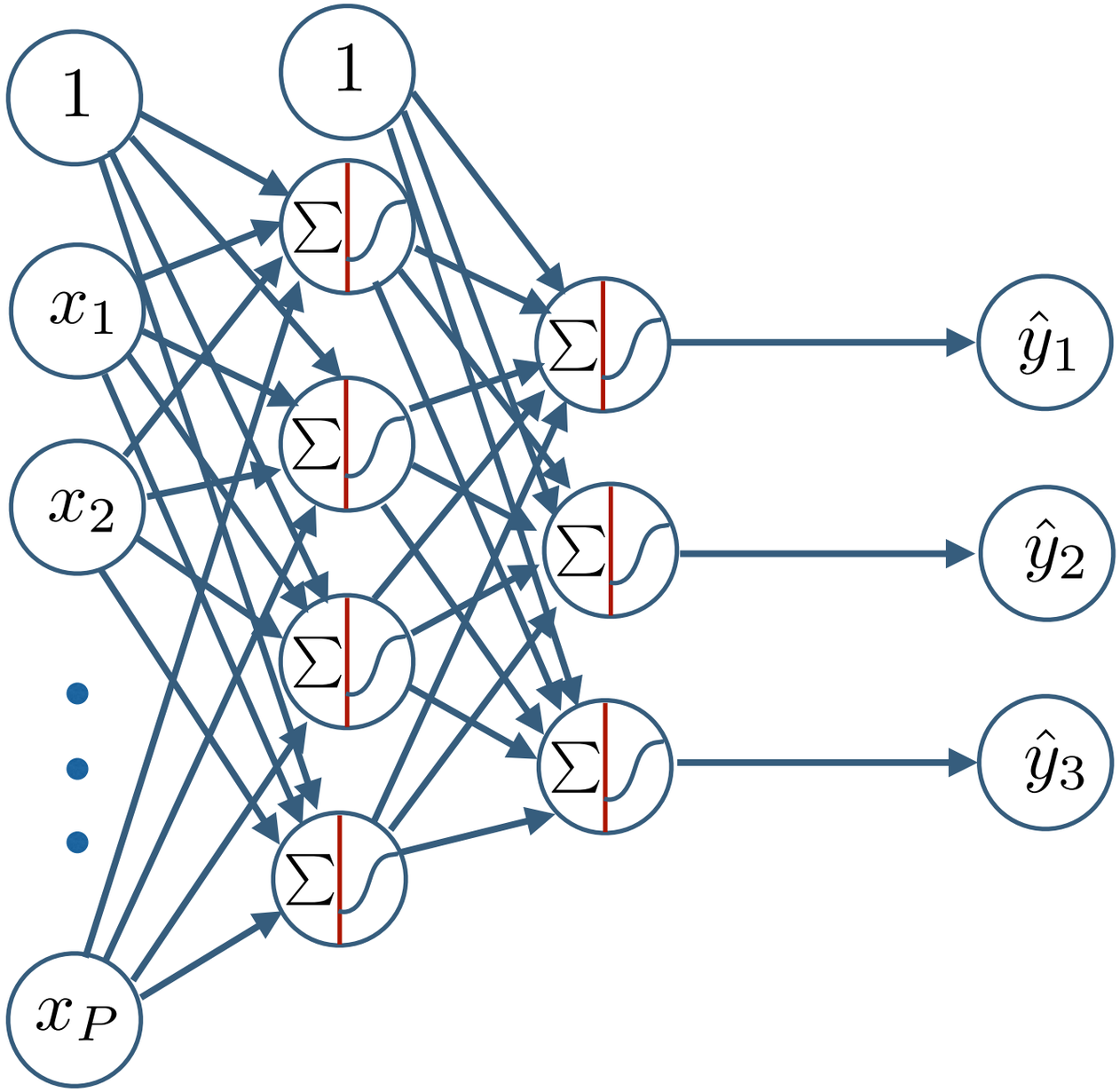


Multi-layer Neuronales Netz



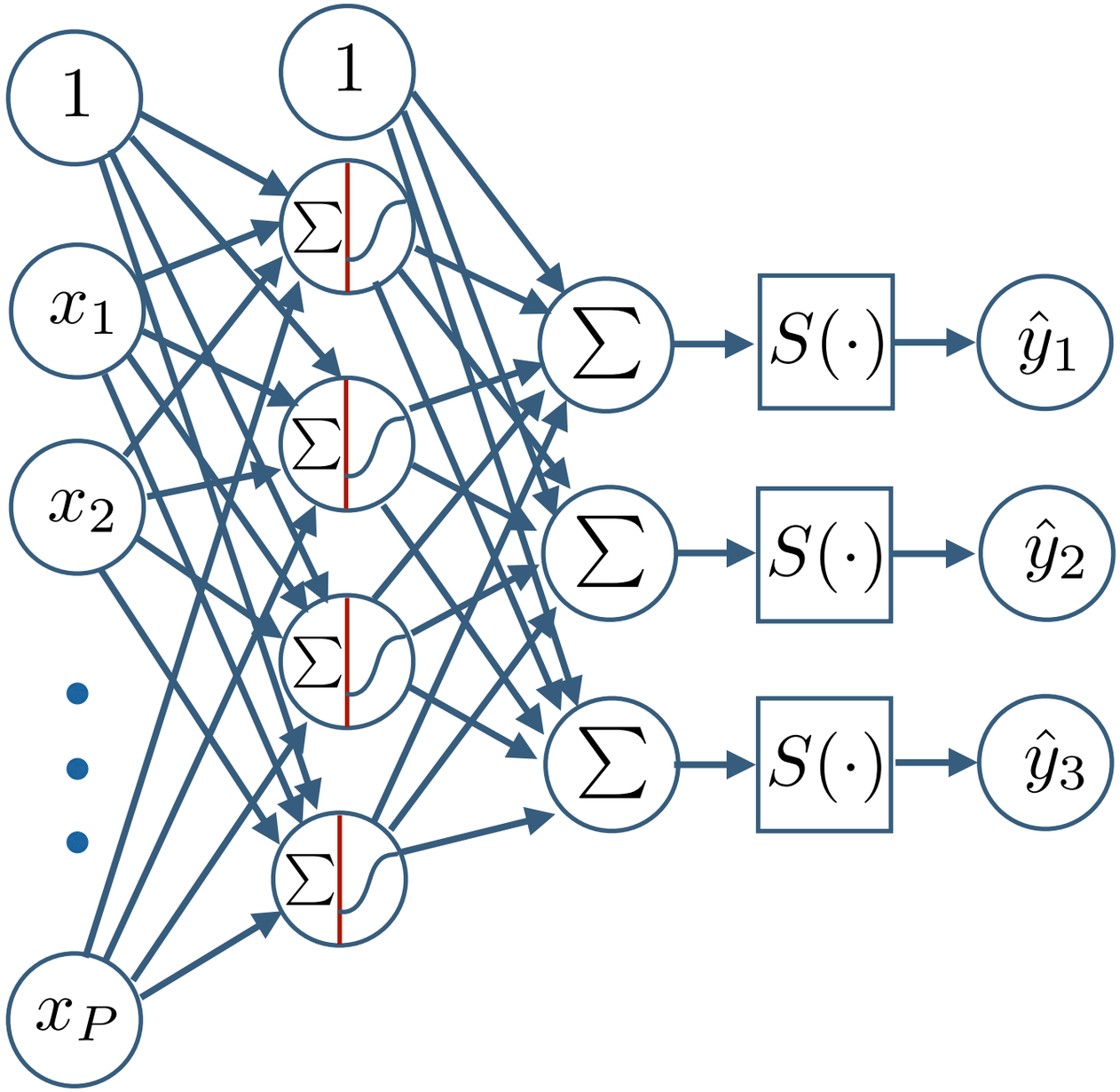
Beispiel für ein 2-Klassen Problem

Multi-layer Multi-Klassen Neuronales Netz



Beispiel für ein 3-Klassen Problem

Multi-layer Multi-Klassen Neuronales Netz



Beispiel für ein 3-Klassen Problem

$$\text{Softmax Funktion: } S(a_k) = \frac{e^{a_k}}{\sum_j e^{a_j}}$$

Softmax Funktion

Beispiel 1

© Martin L: https://de.wikipedia.org/wiki/Datei:Elefant_baby_(318774561).jpg

Label	a_k	e^{a_j}	$S(a_k)$
Katze	-3.44	0.03	0.0006
Hund	1.16	3.19	0.0596
Ente	-0.81	0.44	0.0083
Elefant	3.91	49.9	0.9315

$$\text{Softmax Funktion: } S(a_k) = \frac{e^{a_k}}{\sum_j e^{a_j}}$$

Multi-layer Multi-Klassen Neuronales Netz

Multi-Klassen Klassifikation

$$\hat{\boldsymbol{y}} = \boldsymbol{y}_c \text{ if } S(a_c) > S(a_k)\ \forall c, k \in \{1, 2, \ldots, K\}$$

$$S(a_k) = \frac{e^{a_k}}{\sum_j e^{a_j}} \quad \text{mit } a_k = \sum_{j=0}^{P_{l-1}} \hat{w}_{kj}^{l} \cdot z_j^{l-1}$$

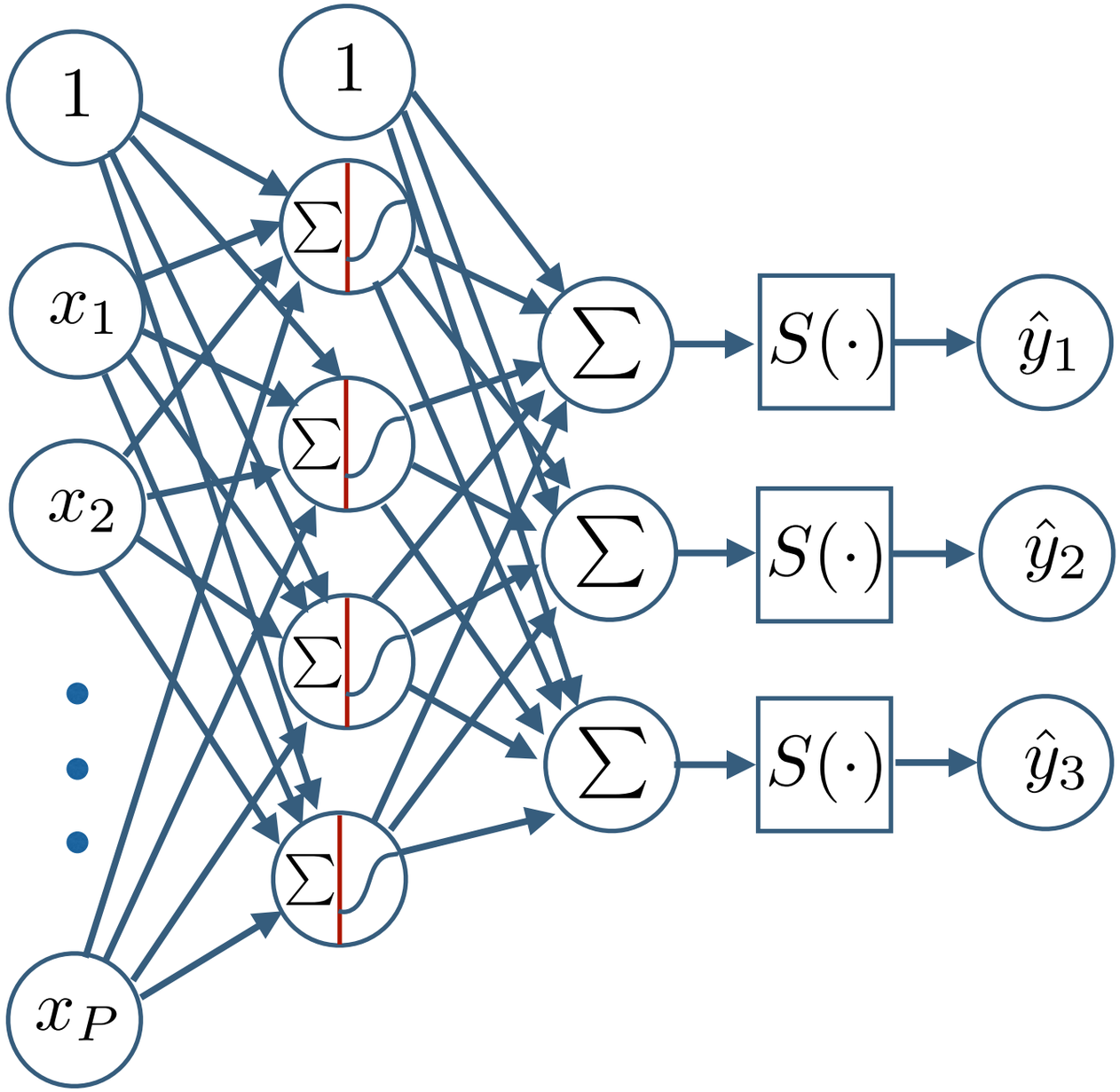
und l ist letzte Logit-Schicht

Binäre Klassifikation

$$\hat{y} = \begin{cases} 1 & \text{if } \sigma(a) > 0.5 \\ 0 & \text{else} \end{cases}$$

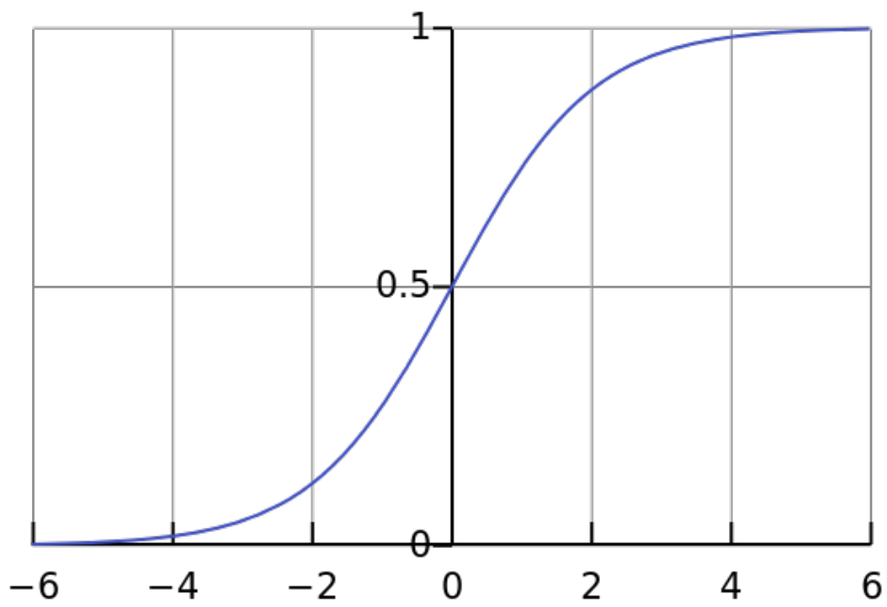
$$\sigma(a) = \frac{e^{a}}{1 + e^{a}} \text{ mit } a = \left(\hat{\boldsymbol{w}}^T \cdot \boldsymbol{x}\right)$$

Multi-layer Multi-Klassen Neuronales Netz



Nicht-Linearität

$$\sigma(a) = \frac{e^a}{1+e^a} \text{ mit } a = \left(\hat{\boldsymbol{w}}^T \cdot \mathbf{x}\right)$$



Multi-layer Multi-Klassen Neuronales Netz

Multi-Klassen Klassifikation

$$\hat{\boldsymbol{y}} = \boldsymbol{y}_c \text{ if } S(a_c) > S(a_k)\ \forall c, k \in \{1, 2, \ldots, K\}$$

$$S(a_k) = \frac{e^{a_k}}{\sum_j e^{a_j}} \quad \text{mit } a_k = \sum_{j=0}^{P_{l-1}} \hat{w}_{kj}^{l} \cdot z_j^{l-1}$$

und l ist letzte Logit-Schicht

Optimierungsproblem

$$\underset{\mathbf{w}}{\operatorname{argmin}} = \left\{ D(\mathbf{w}) \,\middle|\, D(\hat{\boldsymbol{w}}) = -\ln p(\hat{y}|\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(N)}; \hat{\boldsymbol{w}}) \right\}$$

Optimierungsproblem

$$\underset{\mathbf{w}}{\operatorname{argmin}} = \left\{ D_{CE}(\mathbf{W}) \,\middle|\, D_{CE}(\widehat{\boldsymbol{W}}) = -\sum_{n=1}^{N} \sum_{k=1}^{K} y_k^{(n)} \ln p(\hat{y}_k|\boldsymbol{x}^{(n)}; \widehat{\boldsymbol{W}}) \right\}$$

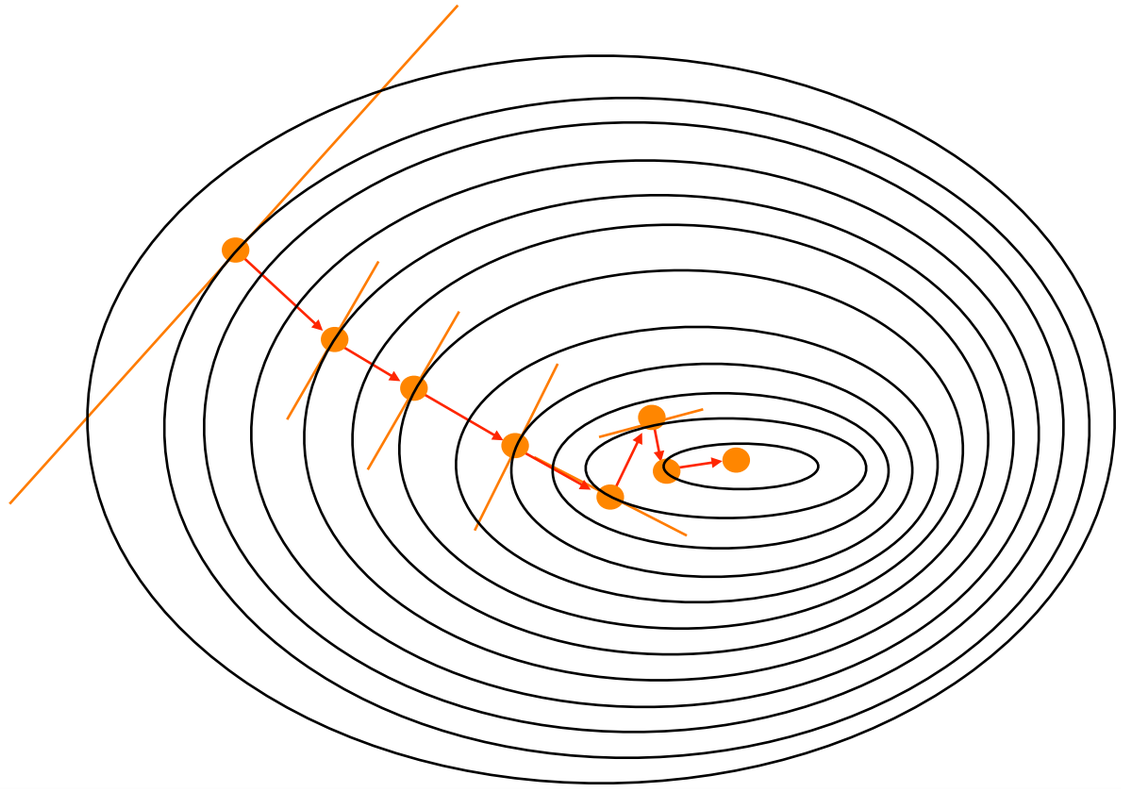
Gradientenabstiegsverfahren

Fehlerfunktion *D*

Lernprinzip

„zufällige“ Initialisierung

iteratives Update der Gewichte $\hat{\boldsymbol{w}}_{(t)} = \hat{\boldsymbol{w}}_{(t-1)} - \eta \nabla D\left(\hat{\boldsymbol{w}}_{(t-1)}\right)$



Gradientenabstiegsverfahren

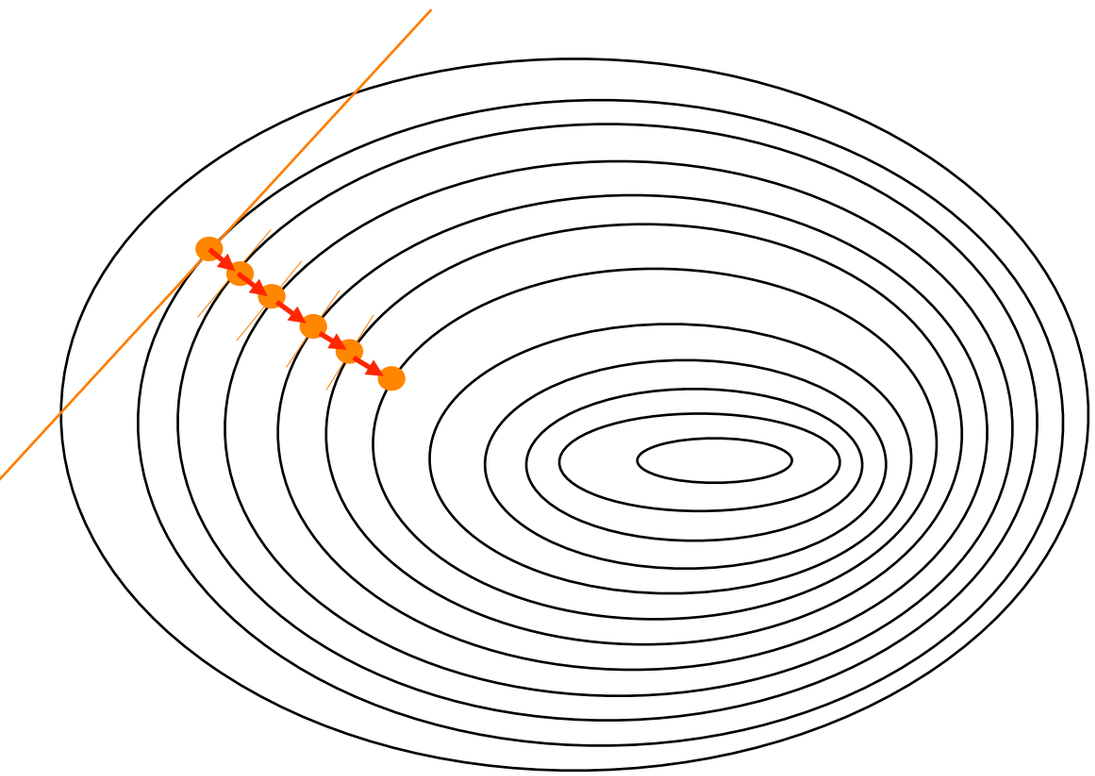
Fehlerfunktion *D*

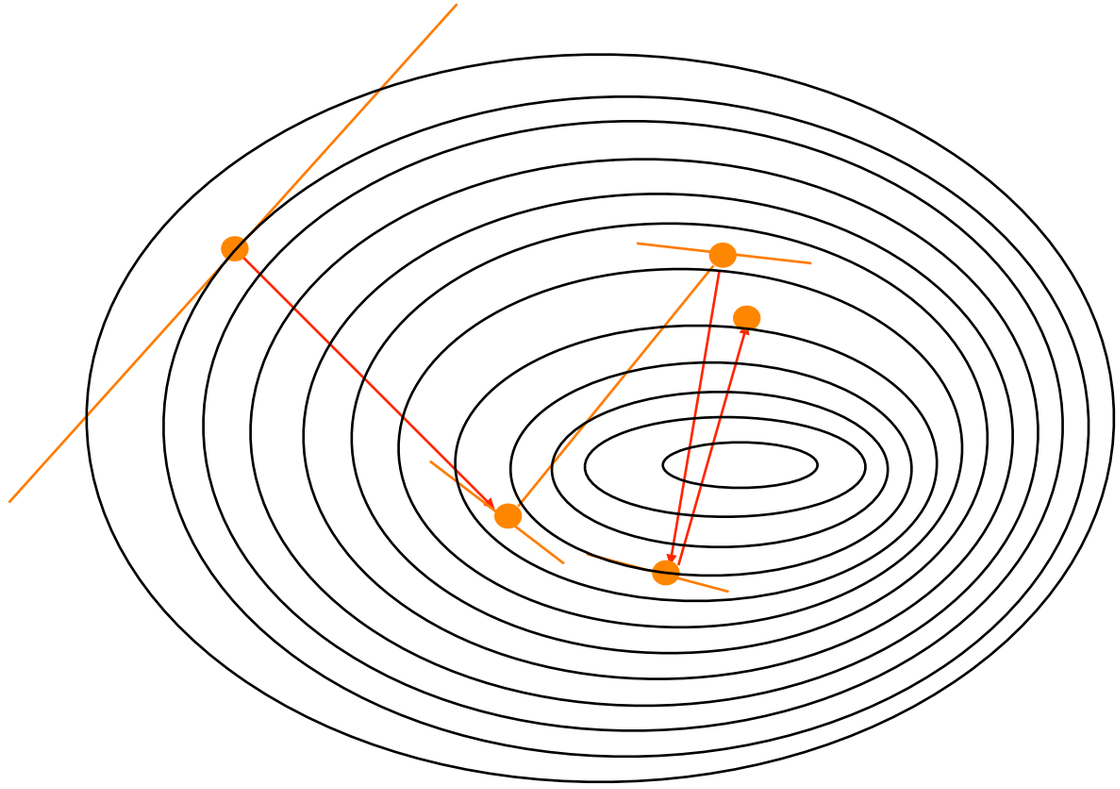
Lernprinzip

„zufällige“ Initialisierung

iteratives Update der Gewichte

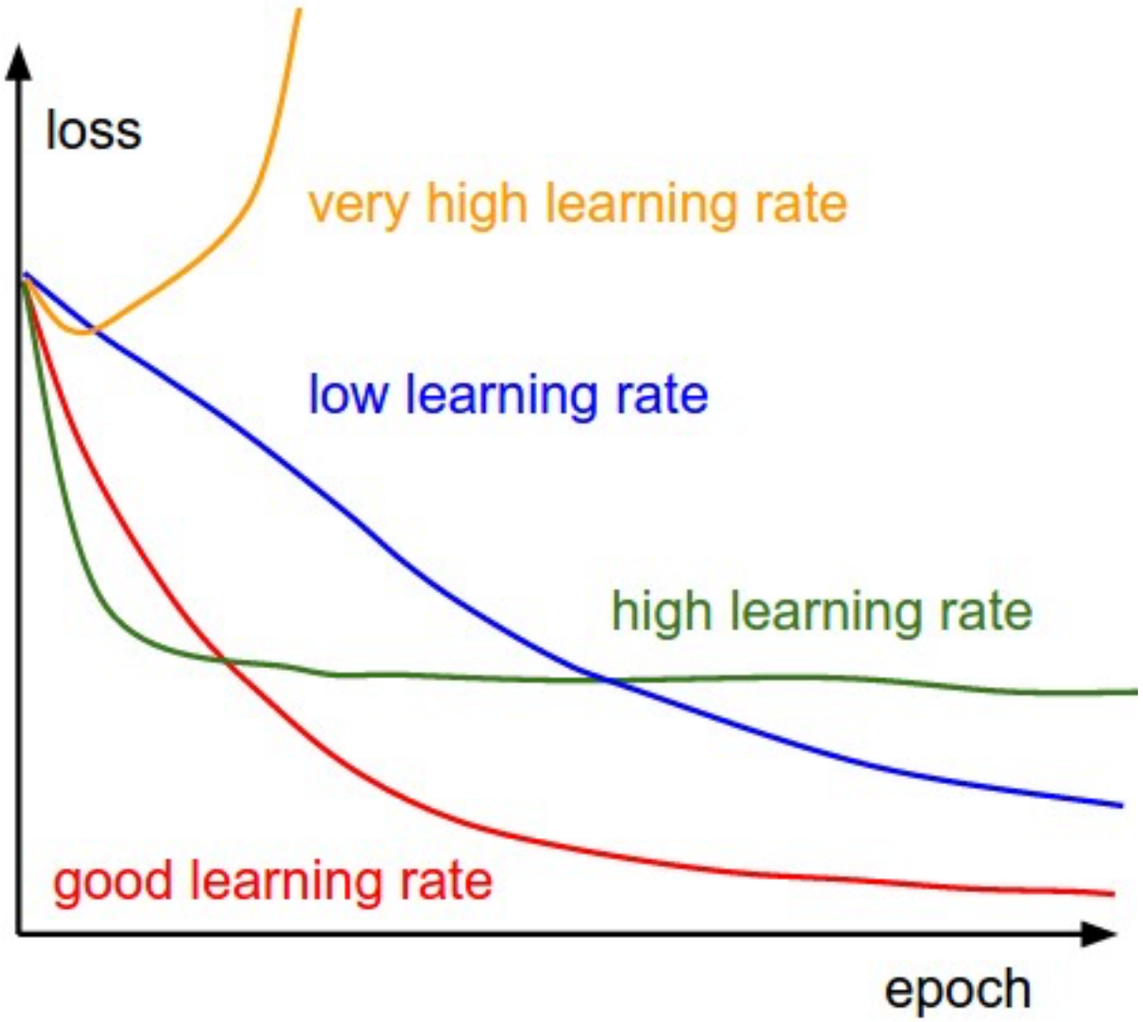
$$\hat{\boldsymbol{w}}_{(t)} = \hat{\boldsymbol{w}}_{(t-1)} - \eta \nabla D\left(\hat{\boldsymbol{w}}_{(t-1)}\right)$$





Gradientenabstiegsverfahren

$$\hat{\boldsymbol{w}}_{(t)} = \hat{\boldsymbol{w}}_{(t-1)} - \eta \nabla D\left(\hat{\boldsymbol{w}}_{(t-1)}\right)$$



Gradientenabstieg

$$\hat{w}^l_{kj,(t)} = \hat{w}^l_{kj,(t-1)} - \eta \frac{\partial D(\widehat{\boldsymbol{W}}_{(t-1)})}{\partial \mathrm{w}^l_{kj}}$$

mit Tensor $\widehat{\boldsymbol{W}}$ aller Gewichte $\hat{w}^l_{k,j}$

Batch Gradientenabstieg

alle Trainingsdaten verwenden, um den Gradienten zu berechnen

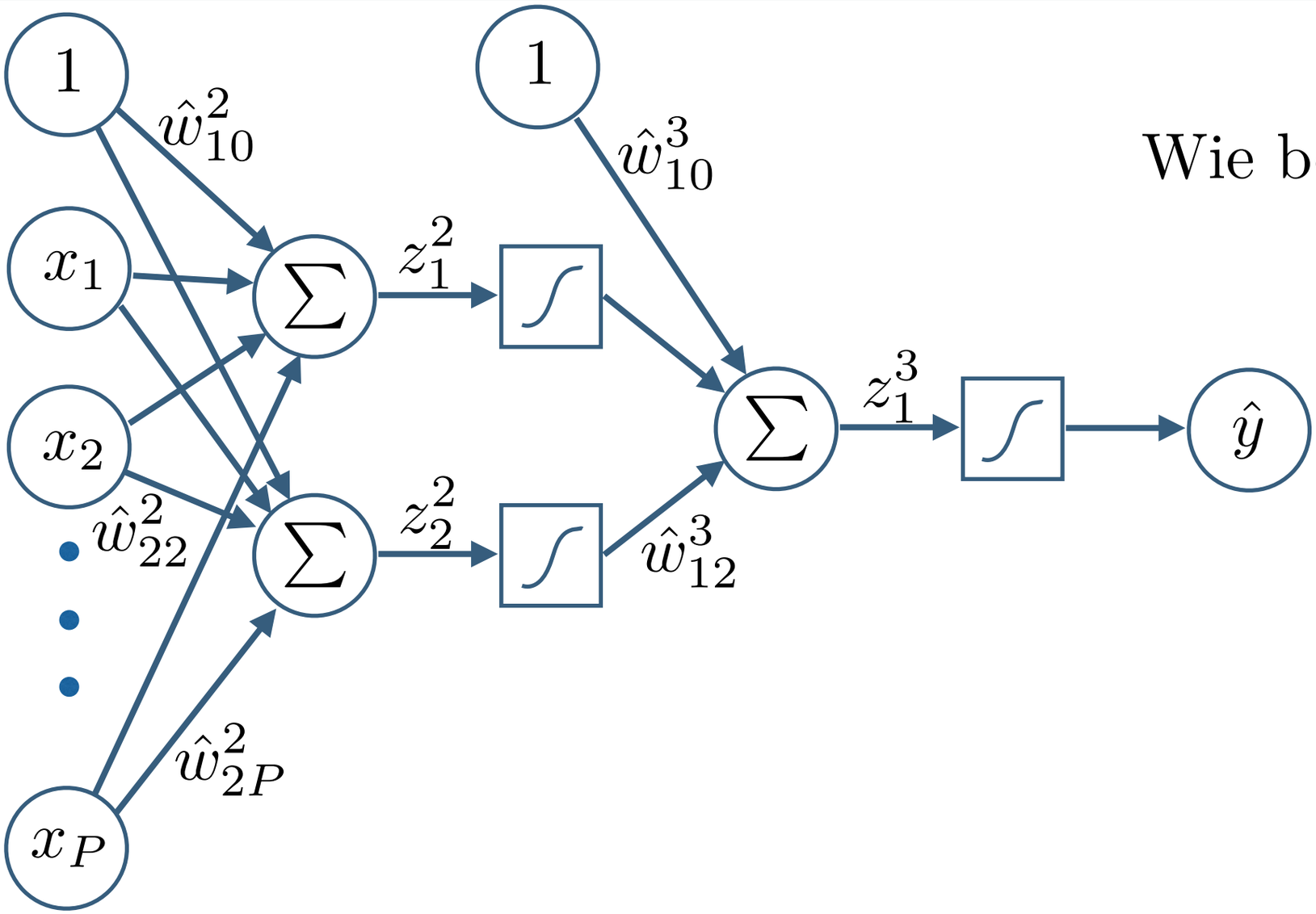
Stochastischer Gradientenabstieg

ein Trainingsdatum verwenden, um den Gradienten zu berechnen

Minibatch Stochastischer Gradientenabstieg

verwende einen mini batch von $B \ll N$ Trainingsdaten

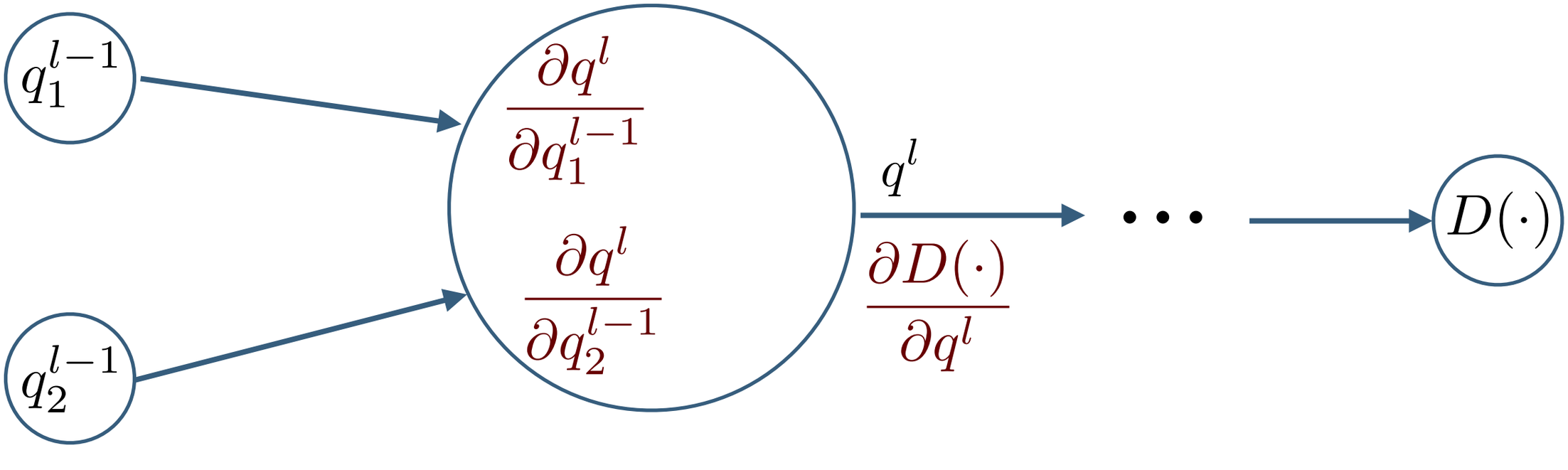
Backpropagation



Wie berechnet man $\frac{\partial D(\widehat{\boldsymbol{W}})}{\partial w^2_{22}}$?

Backpropagation

Ein Knoten in einem Neuronalen Netz



© Li, Karpathy, Johnson

Regularisierung

Ziel

Overfitting vermeiden

Methoden

Adaptiere die Kapazität eines Neuronalen Netzes gemäß der Anwendung

(1) Gewichte einschränken

(a) vorzeitiger Abbruch

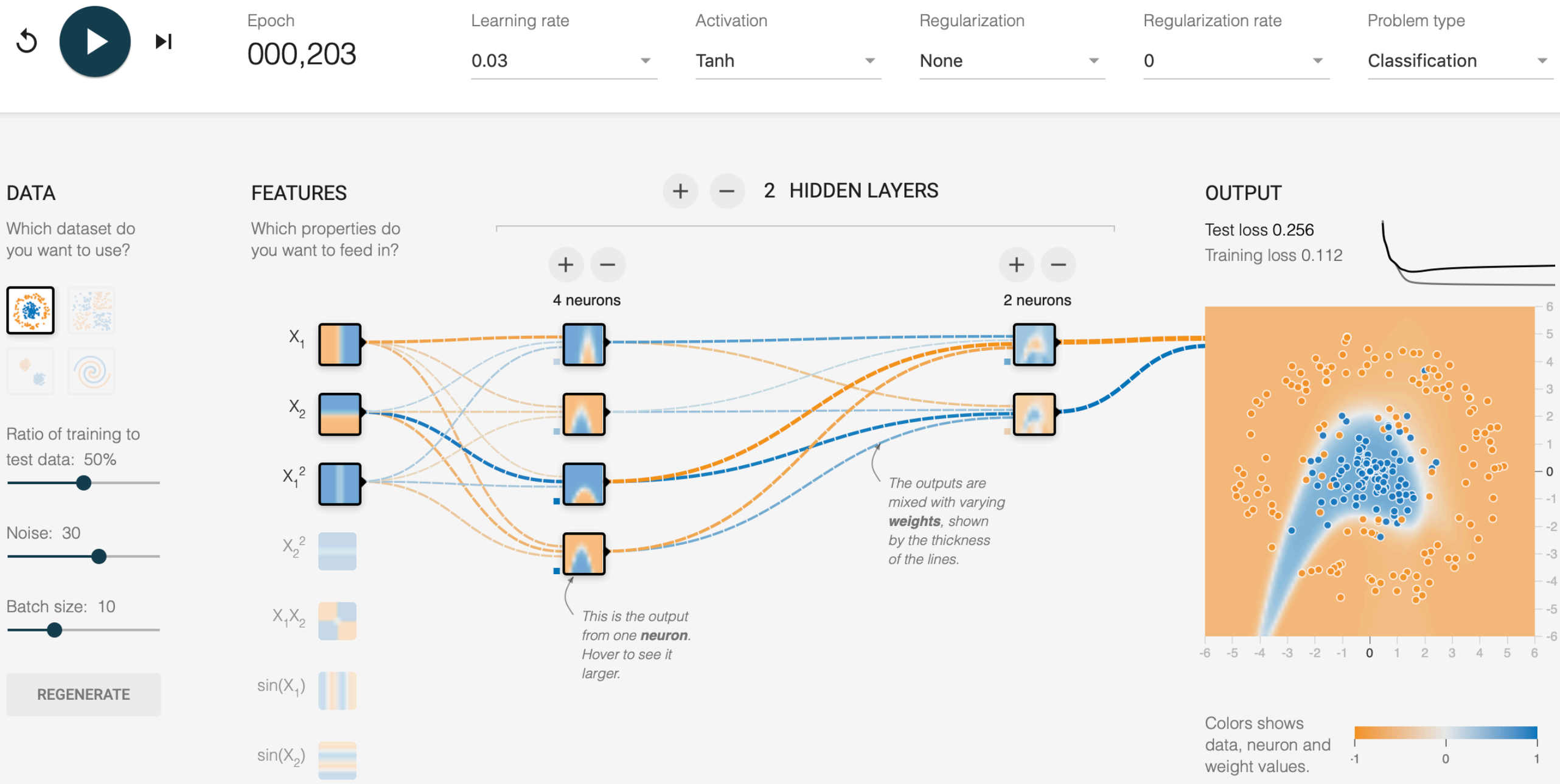
(b) Dämpfung der Gewichte

(2) Reduziere die Zahl der Neuronen

(a) Dropout

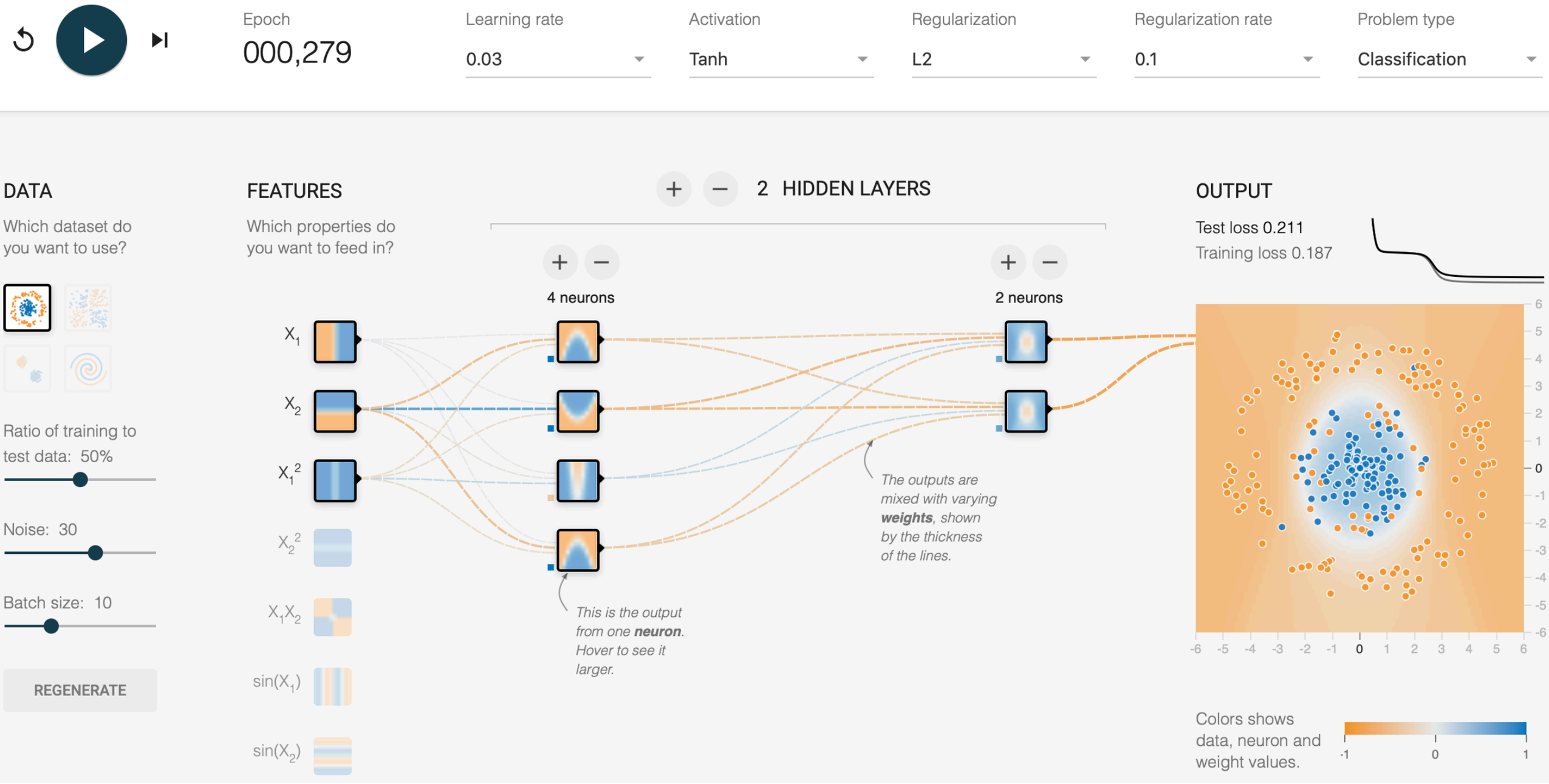
Google Tensorflow Playground

https://playground.tensorflow.org/



Google Tensorflow Playground

https://playground.tensorflow.org/



Wiederholung der Grundlagen

1. Regression und Klassifikation
2. Aufbau und Training von tiefen Neuronalen Netzen
3. **Faltungsnetze**
4. Fehlermaße der Klassifikation

Faltungsnetzwerke

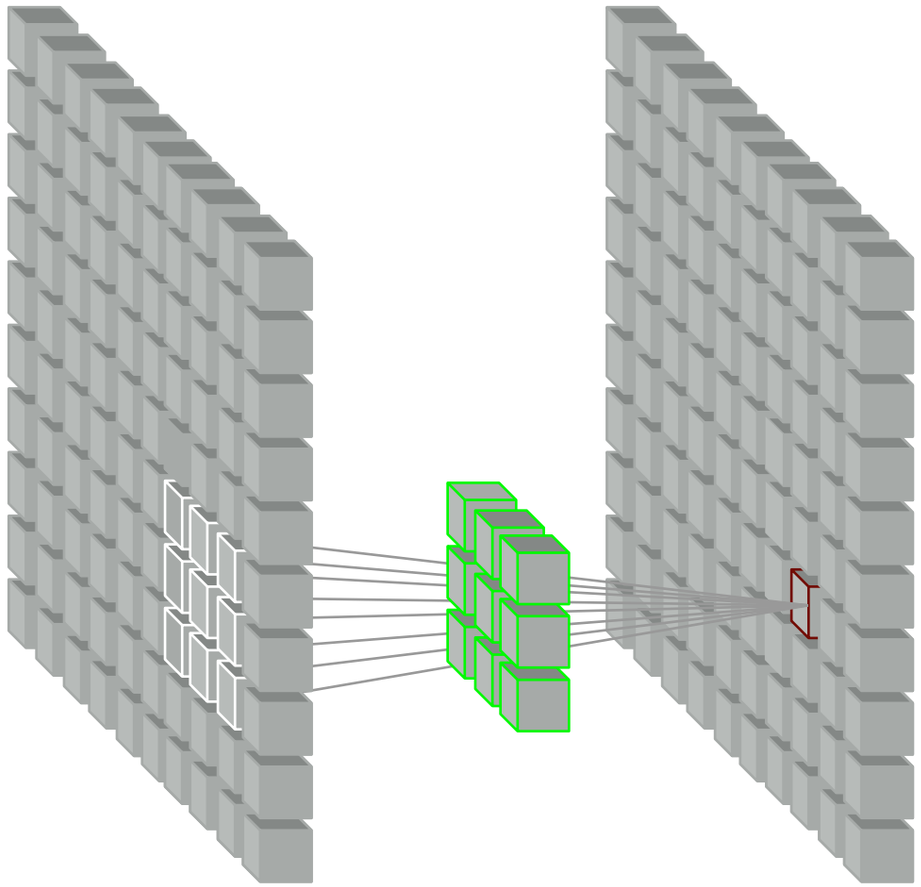
Frage:

Was zeigt dieses Bild?

58	67	65	66	82	58	72	76	67	65	66
60	59	67	72	66	60	61	79	59	67	72
63	72	58	67	65	66	82	61	72	58	67
67	65	60	59	67	72	66	67	65	60	59
69	66	76	58	67	65	66	82	58	72	76
74	72	79	60	59	67	72	66	60	61	79
69	67	61	63	72	58	67	65	66	82	61
83	59	67	67	65	60	59	67	72	66	67
67	65	60	59	67	72	66	67	65	60	59

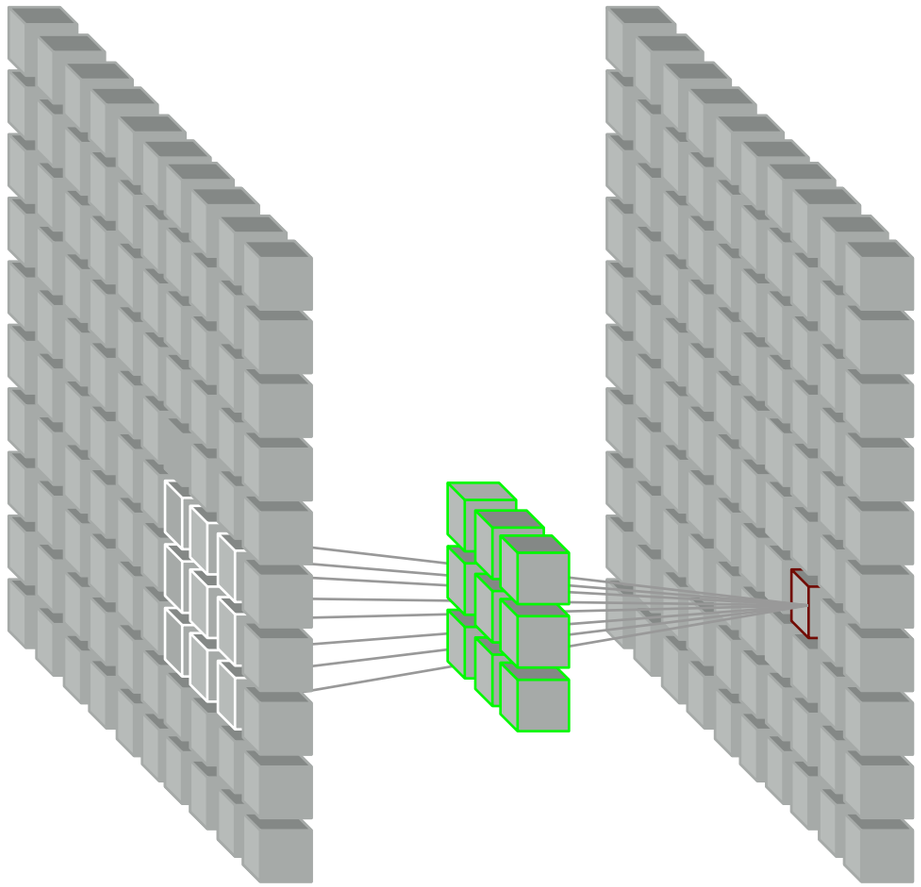
© Li, Karpathy, Johnson

Faltung

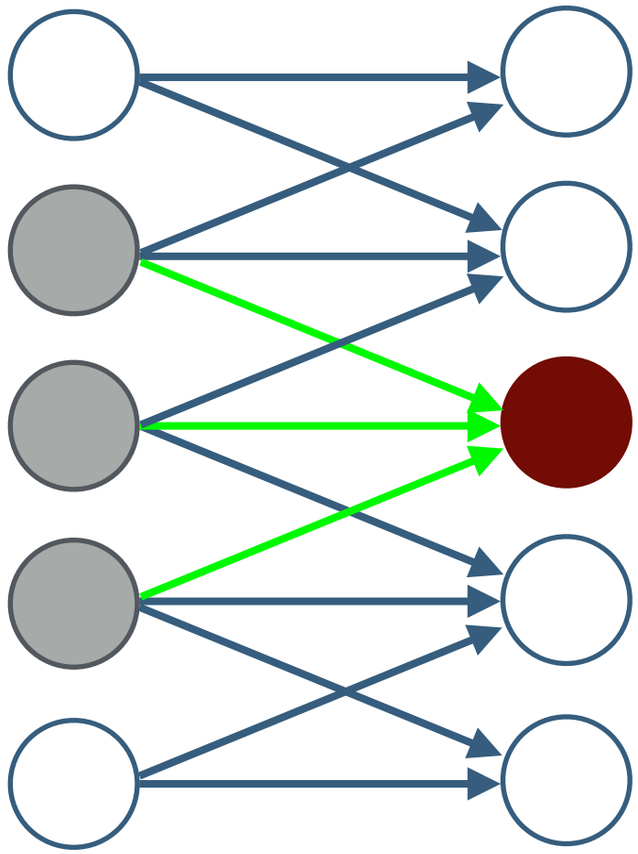


$$g(p,q) = \sum_{i=-\lfloor \frac{I}{2} \rfloor}^{\frac{I}{2}} \sum_{j=-\lfloor \frac{J}{2} \rfloor}^{\frac{J}{2}} f(p+i, q+j) h(i,j)$$

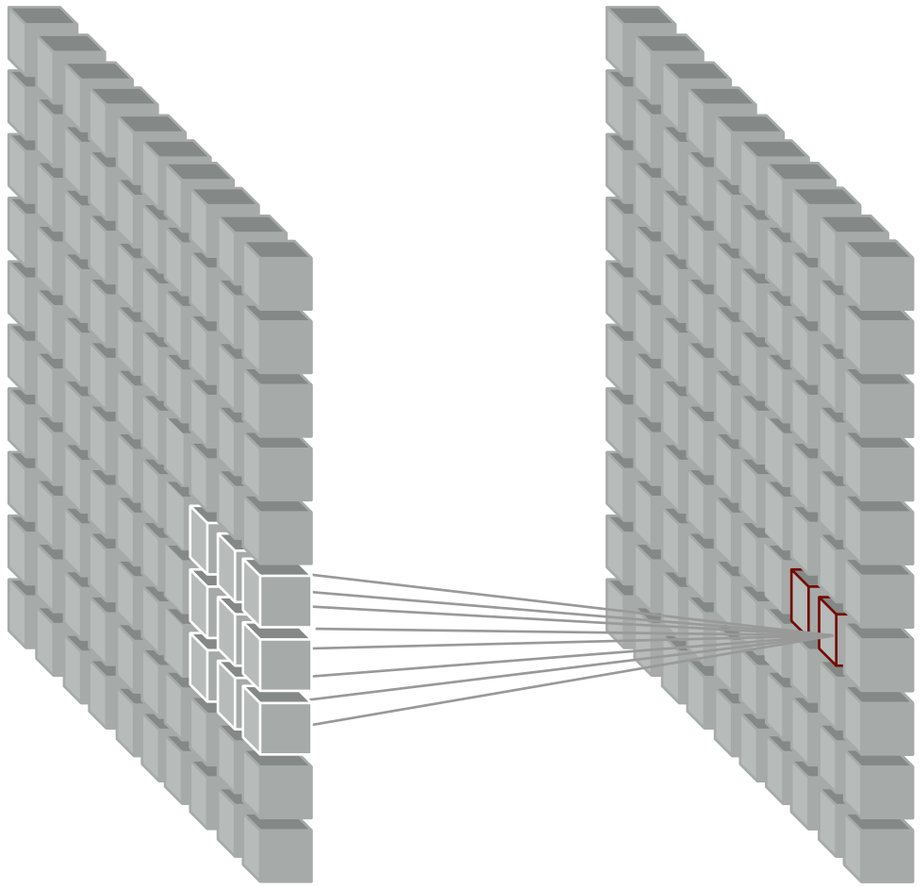
Faltung

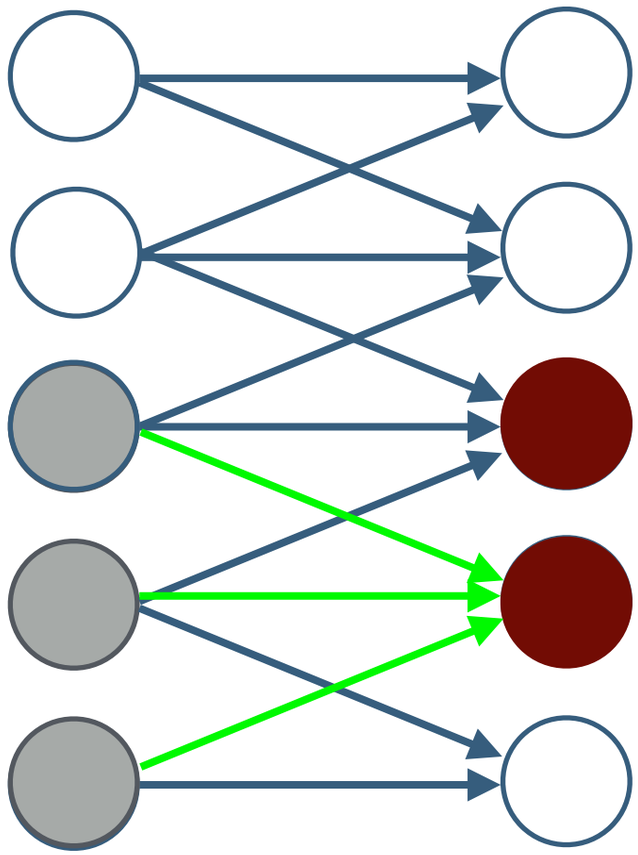


$$g(p,q) = \sum_{i=-\lfloor \frac{I}{2} \rfloor}^{\frac{I}{2}} \sum_{j=-\lfloor \frac{J}{2} \rfloor}^{\frac{J}{2}} f(p+i, q+j)h(i,j)$$

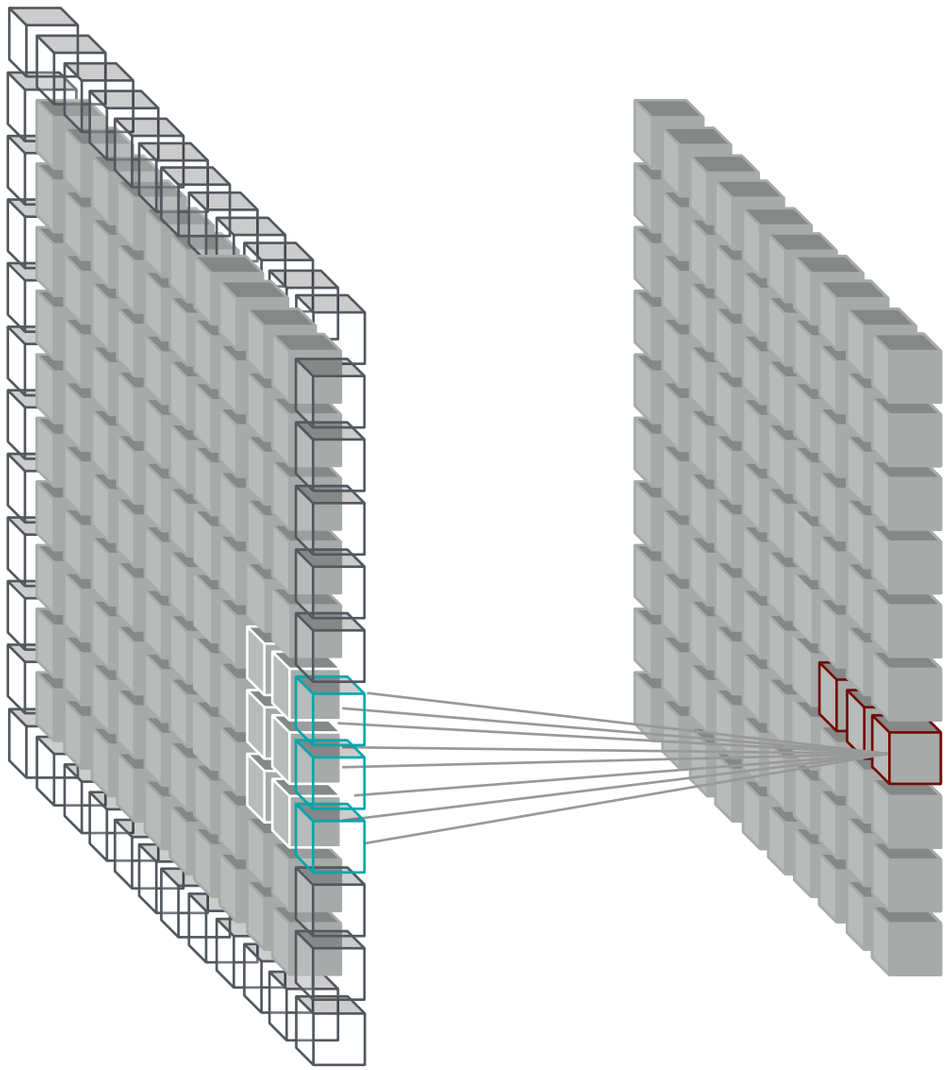


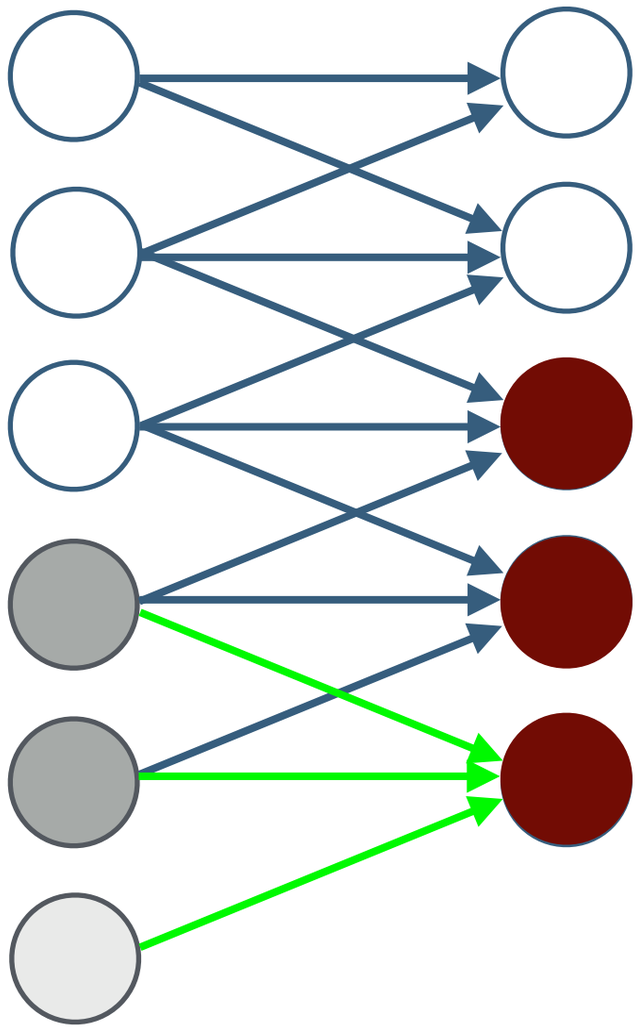
Faltung



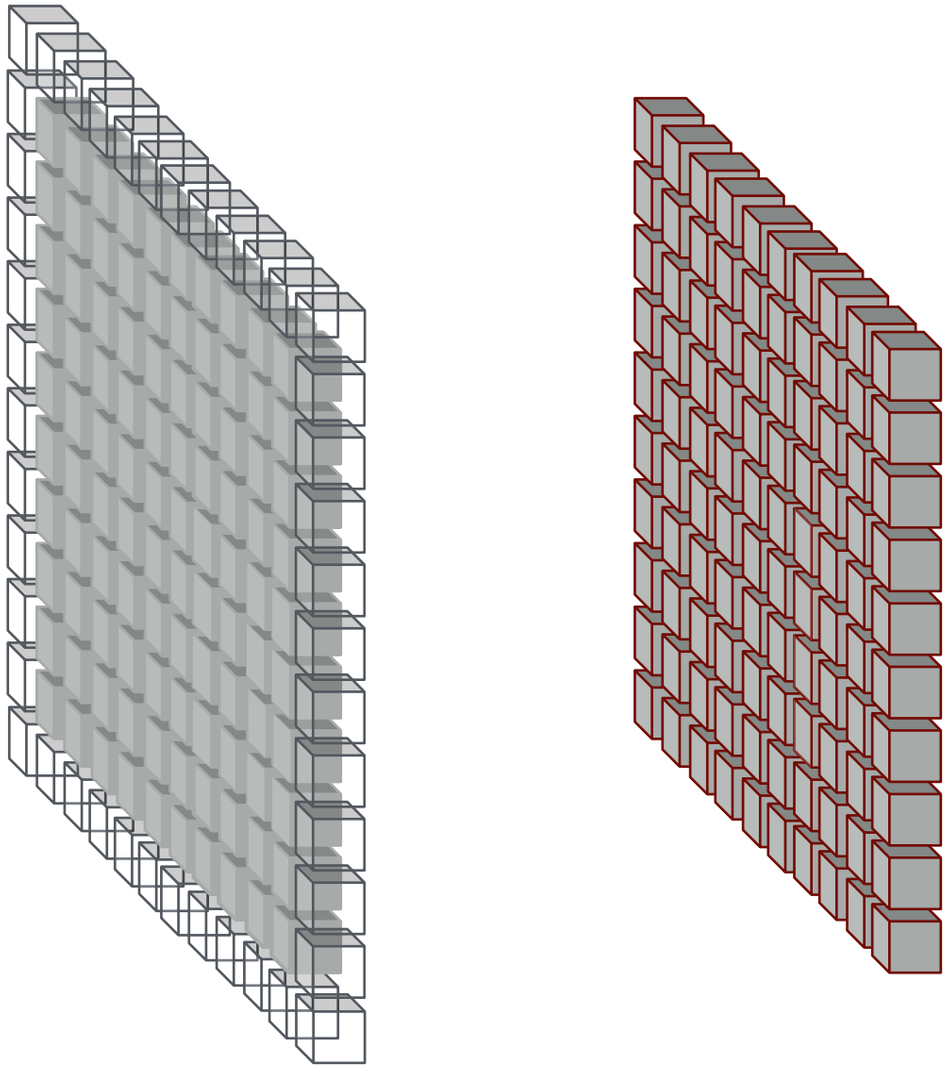


Faltung



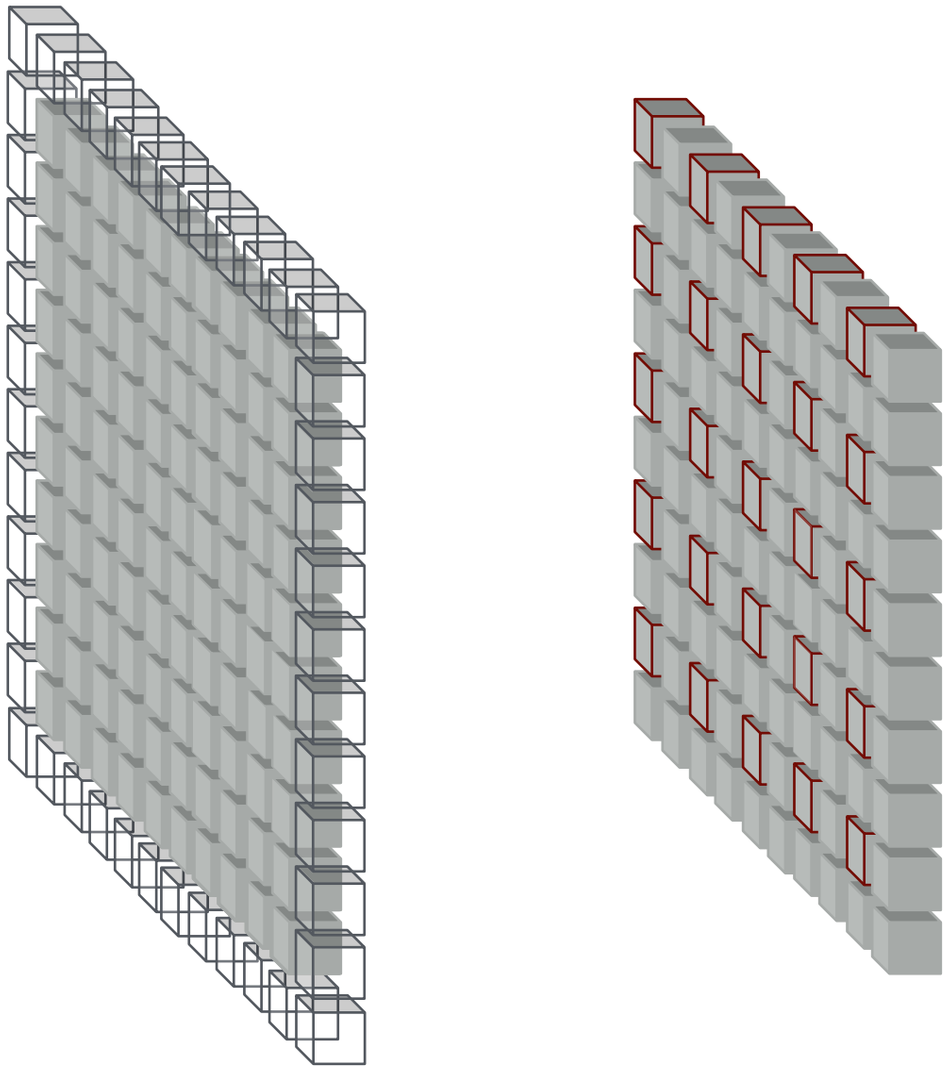


Faltung



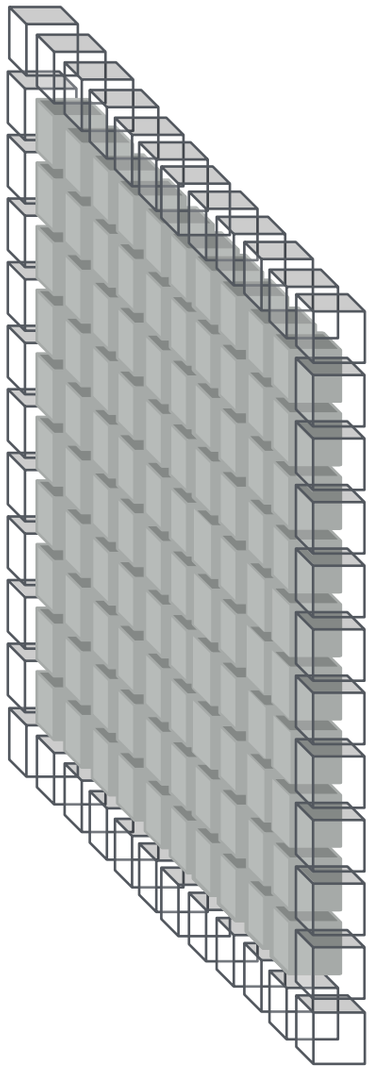
Faltung

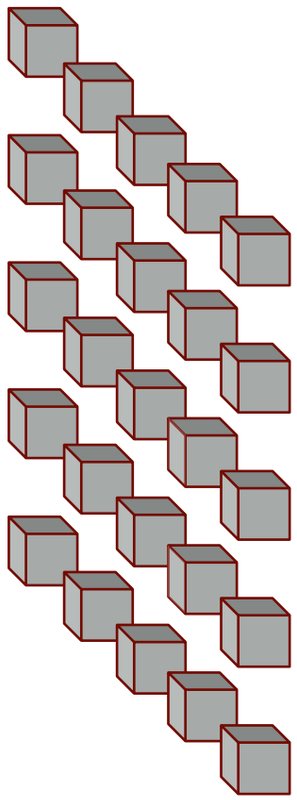
Schrittweite



Faltung

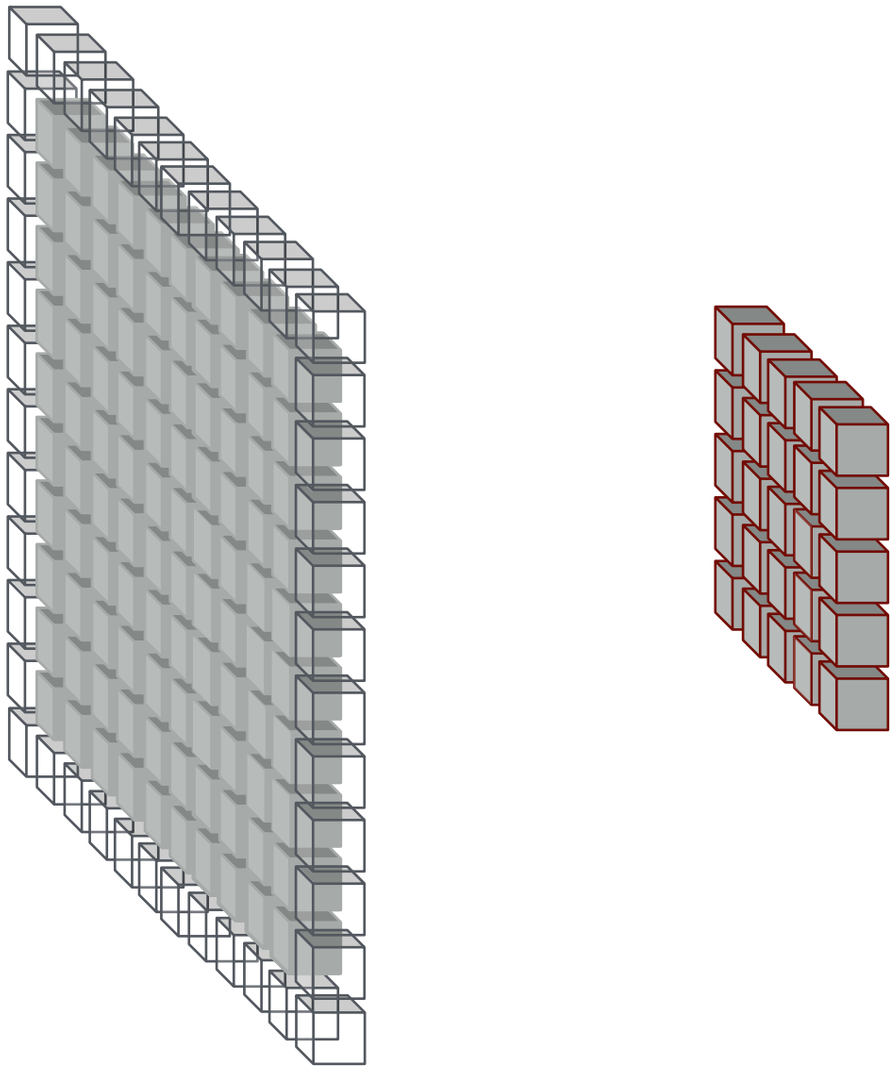
Schrittweite



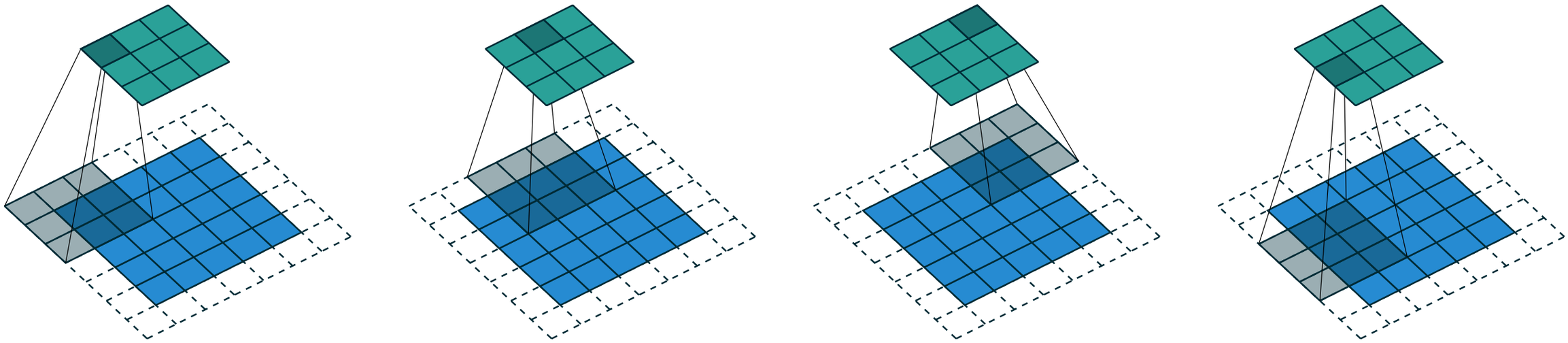


Faltung

Schrittweite



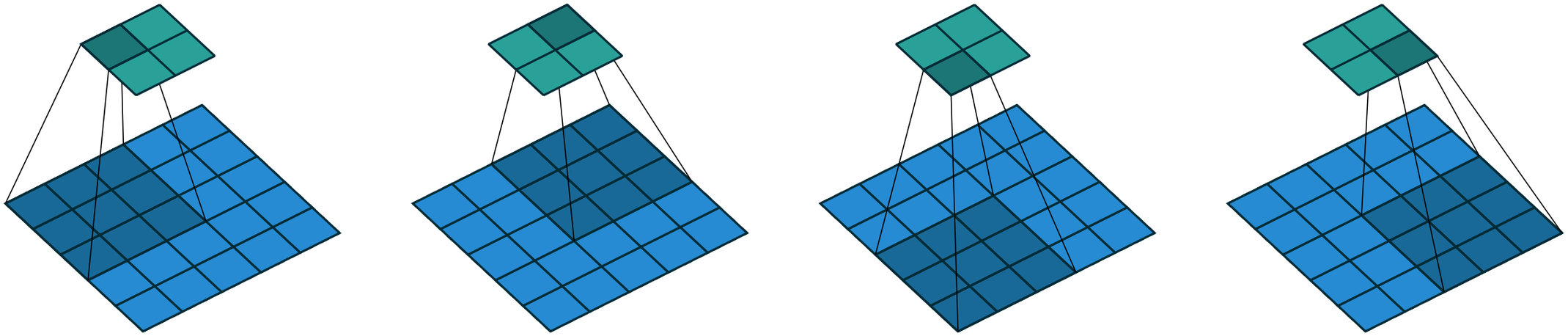
Faltung



Größe des Outputs in (p, q)-Richtung (mit zero padding):

$$\left(\left\lfloor \frac{P + 2 \cdot \lfloor \frac{I}{2} \rfloor - I}{s} \right\rfloor + 1, \left\lfloor \frac{Q + 2 \cdot \lfloor \frac{J}{2} \rfloor - J}{s} \right\rfloor + 1 \right)$$

Faltung

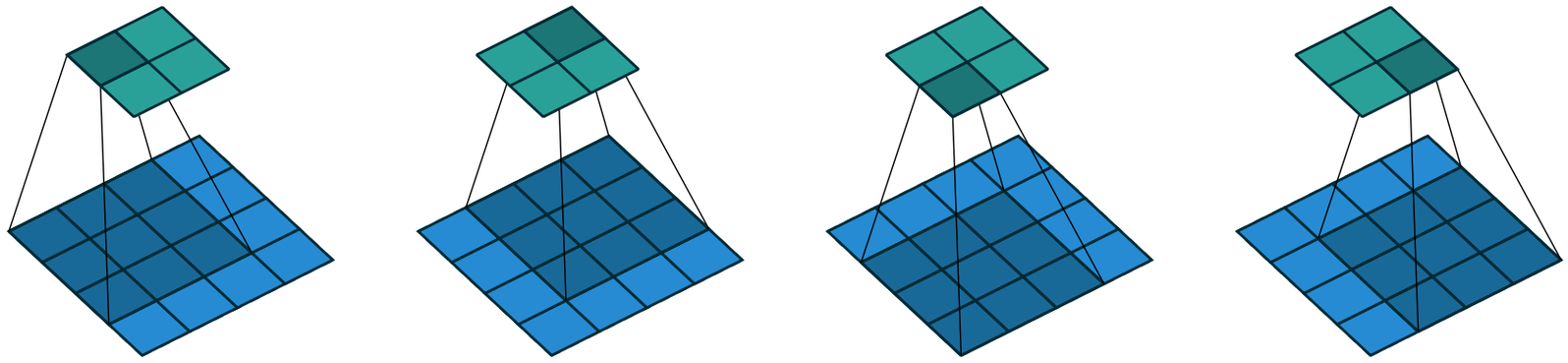


Größe des Outputs in (p, q)-Richtung (ohne zero padding):

$$\left(\left\lfloor \frac{P - I}{s} \right\rfloor + 1, \left\lfloor \frac{Q - J}{s} \right\rfloor + 1 \right)$$

Faltung

als Matrix



$$\begin{pmatrix} w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} & 0 & 0 & 0 & 0 & 0 \\ 0 & w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} & 0 \\ 0 & 0 & 0 & 0 & 0 & w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} \end{pmatrix}$$

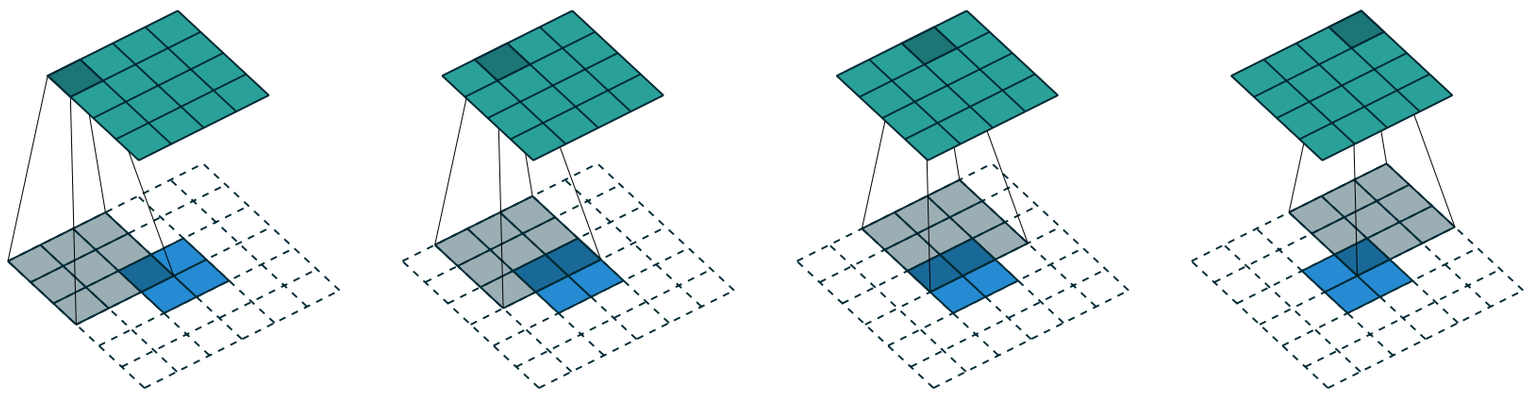
© Dumoulin, Visin

Faltung

transponiert

0	1
2	3

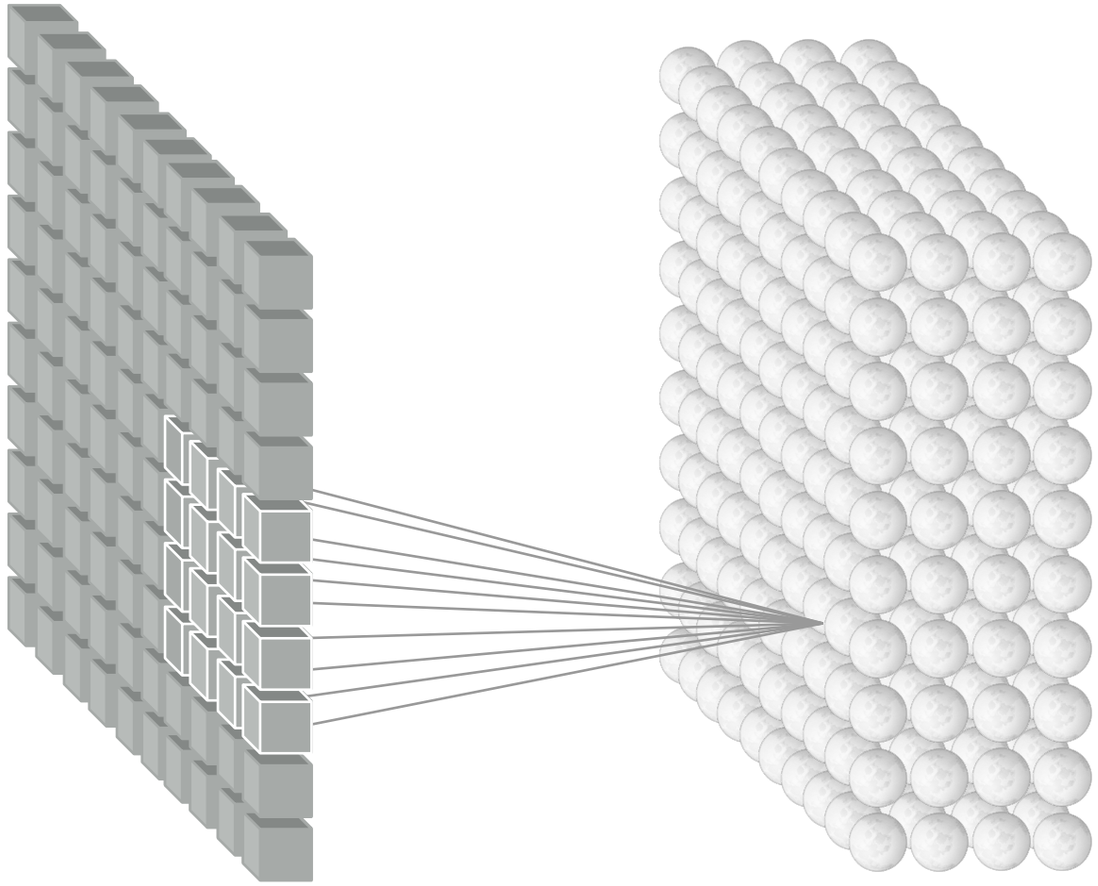
0	1	5
0	2	6
2	0	3



© Dumoulin, Visin

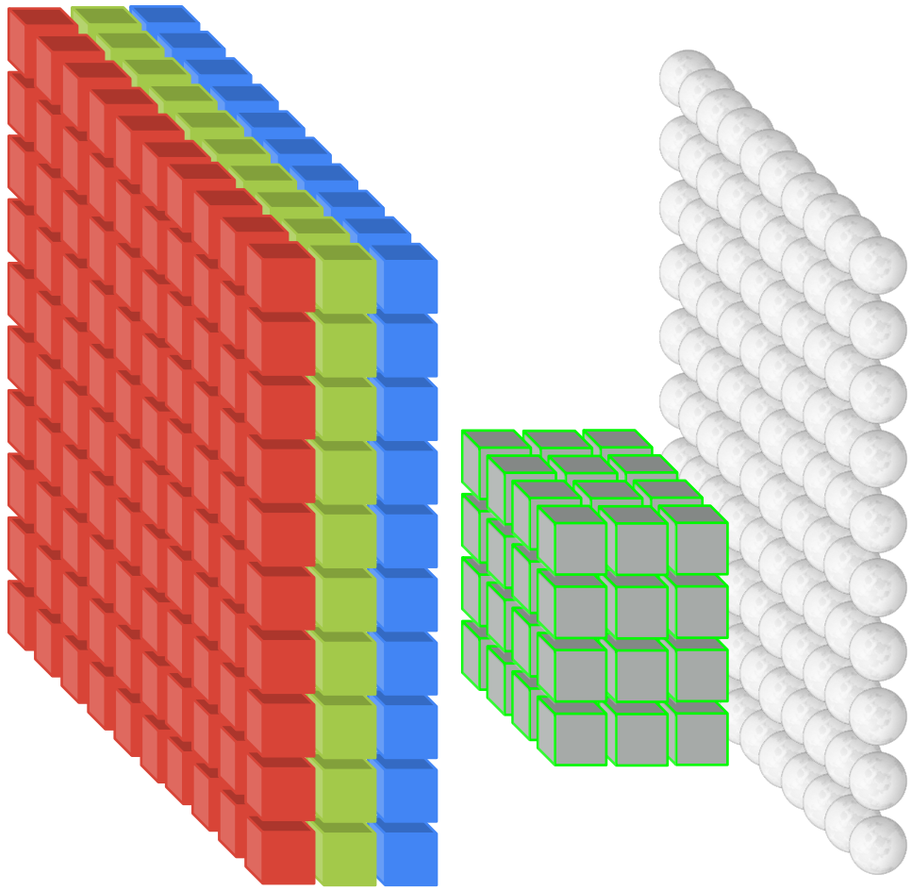
Faltung

als Schicht mit mehreren Faltungskernen



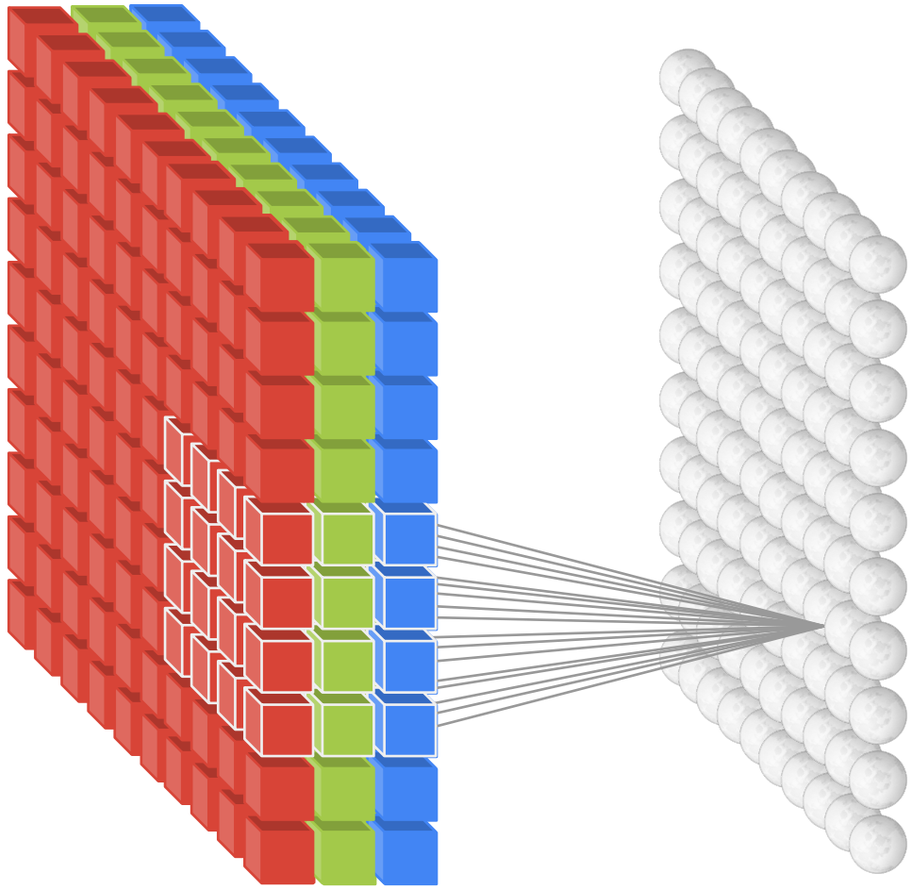
Faltungsschicht

Farbbilder



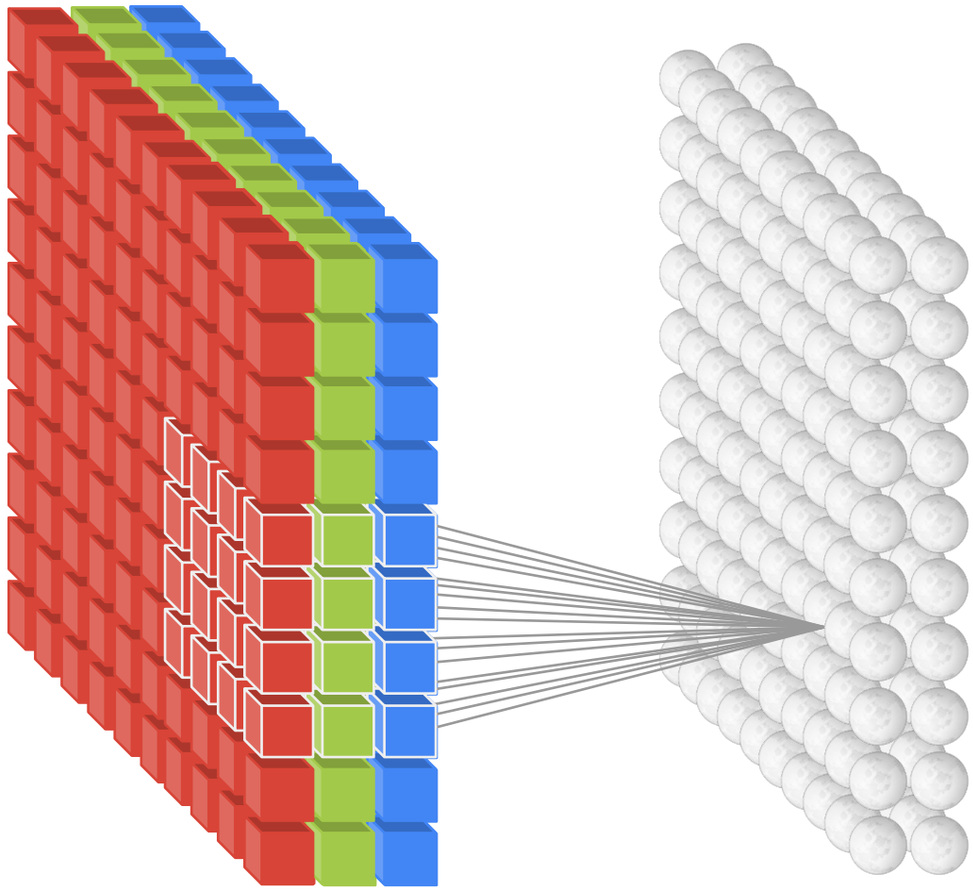
Faltungsschicht

Farbbilder



Faltungsschicht

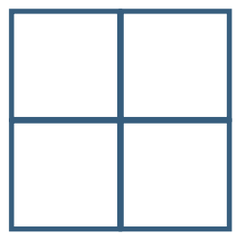
Farbbilder



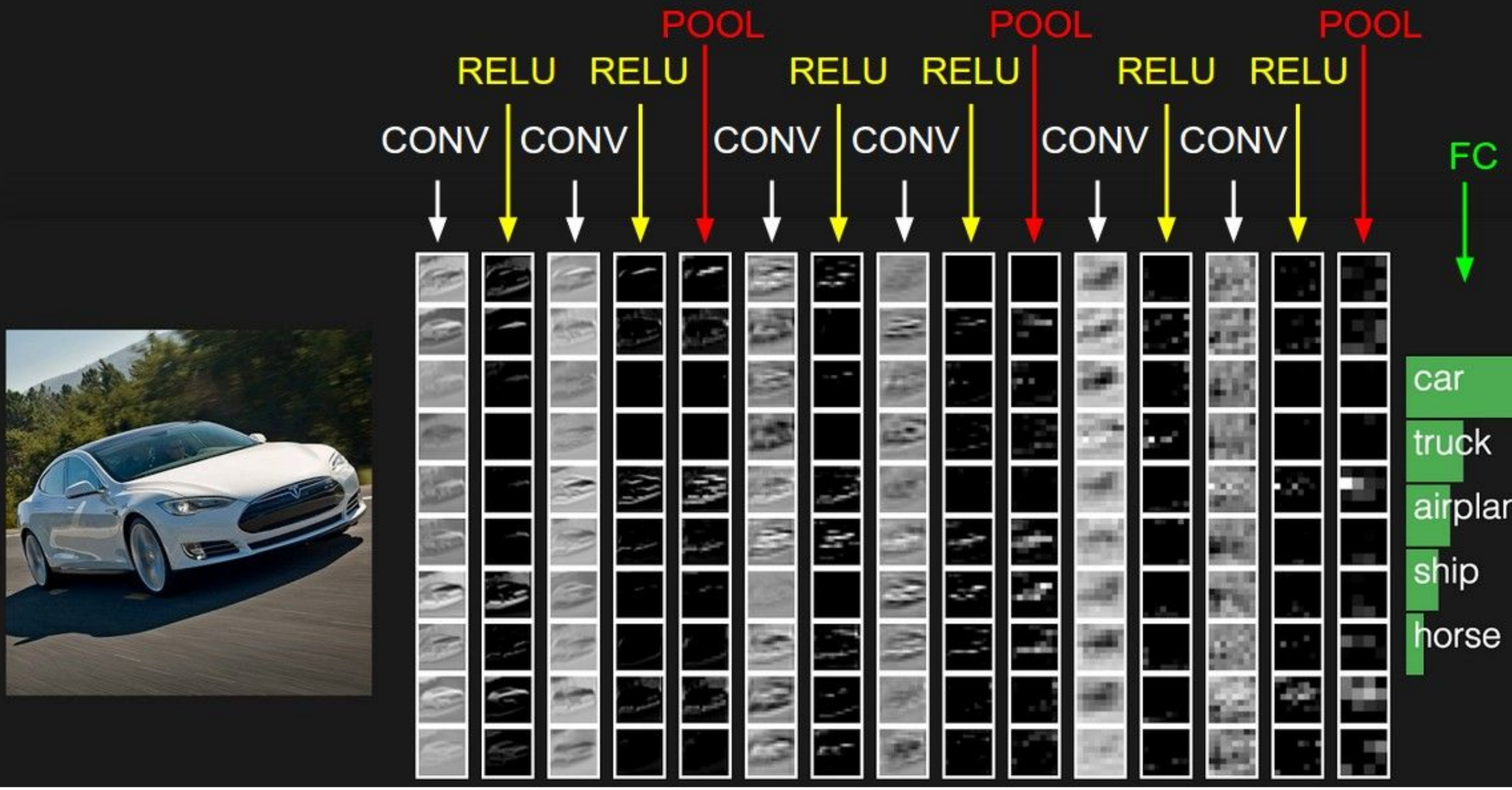
Pooling Schicht

Max Pooling

1	1	2	4
5	6	7	8
3	2	1	0
1	2	3	4

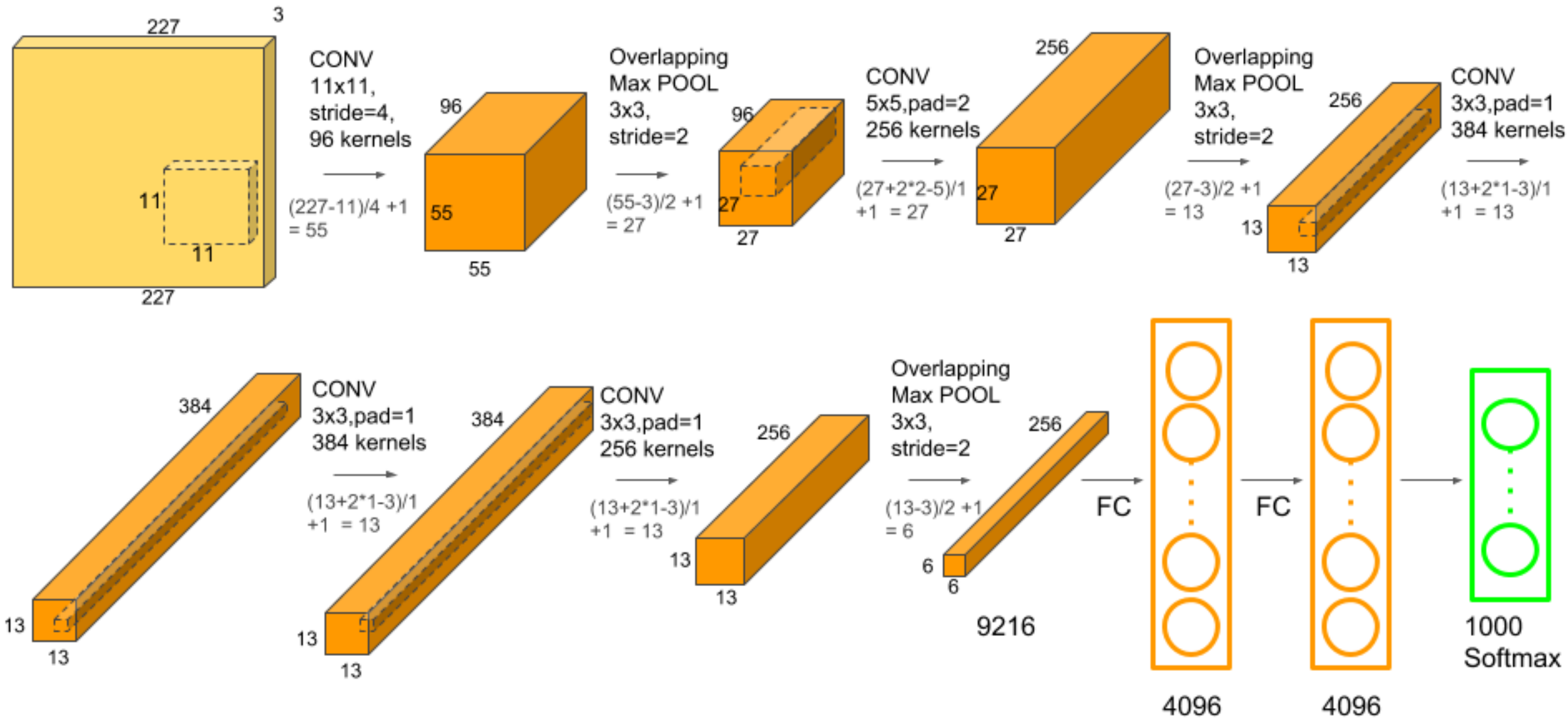


Faltungsnetzwerke



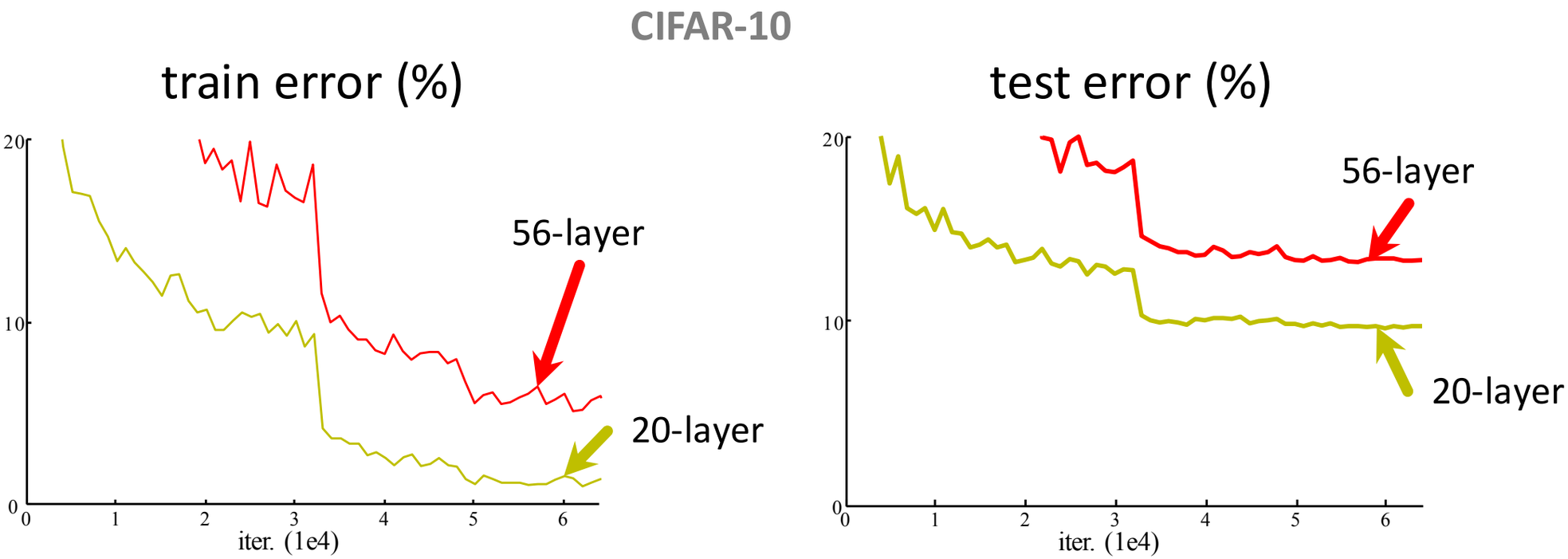
© Li, Karpathy, Johnson

AlexNet

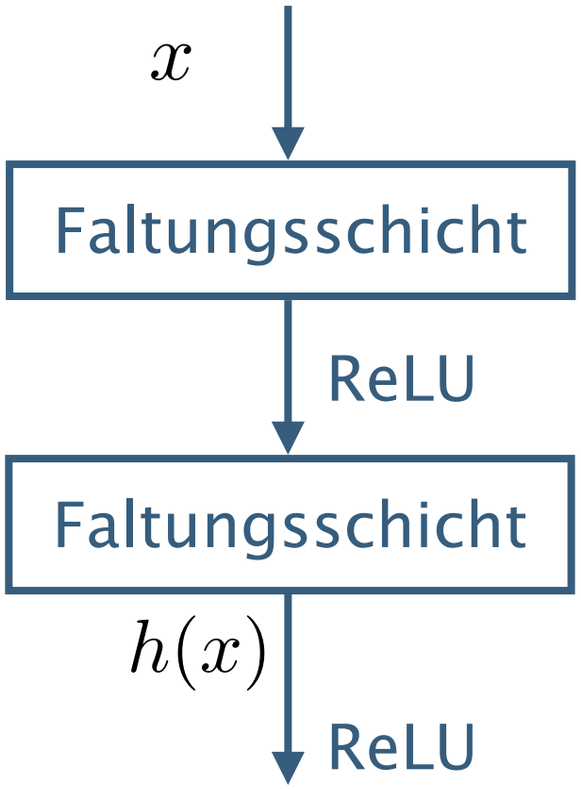


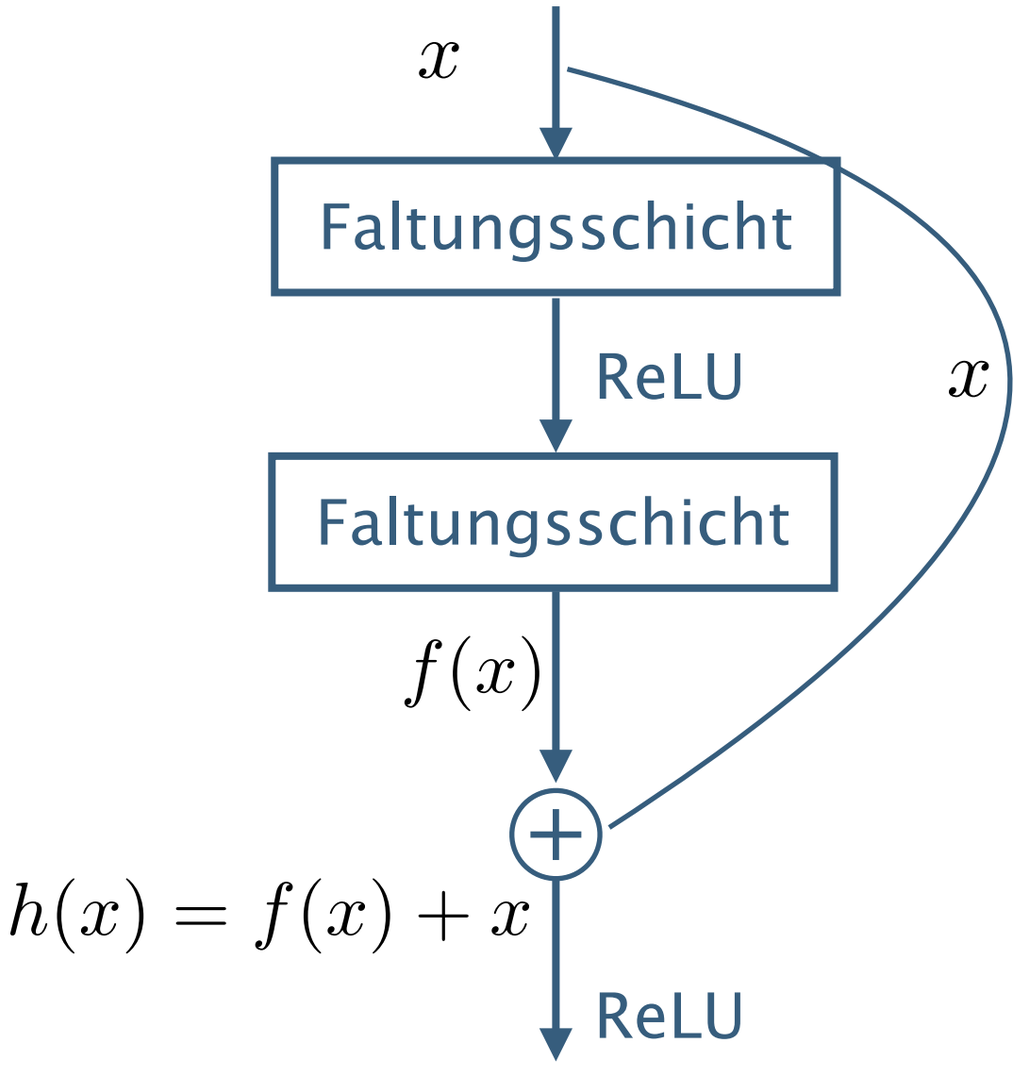
Residual Nets

je tiefer, desto besser?

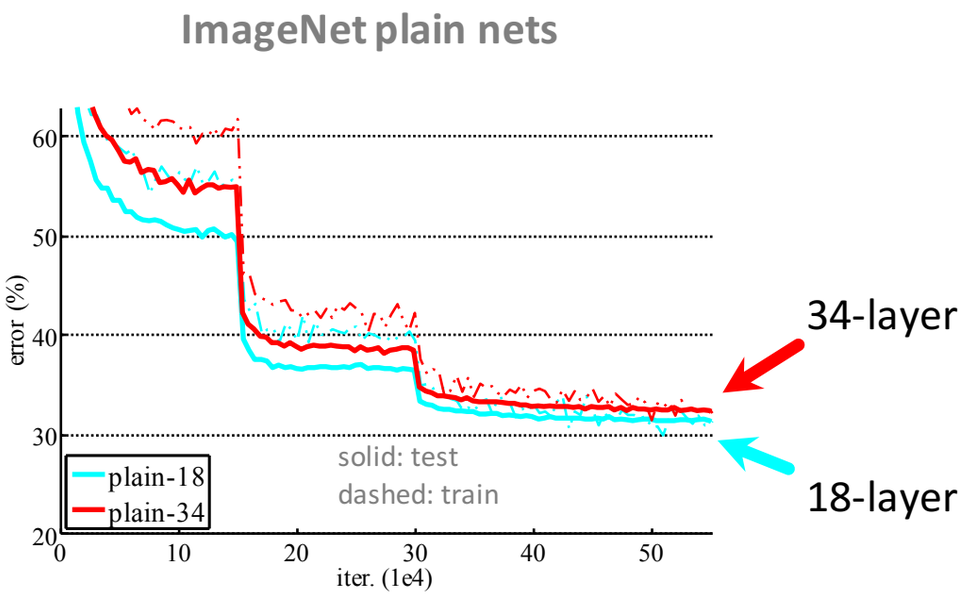


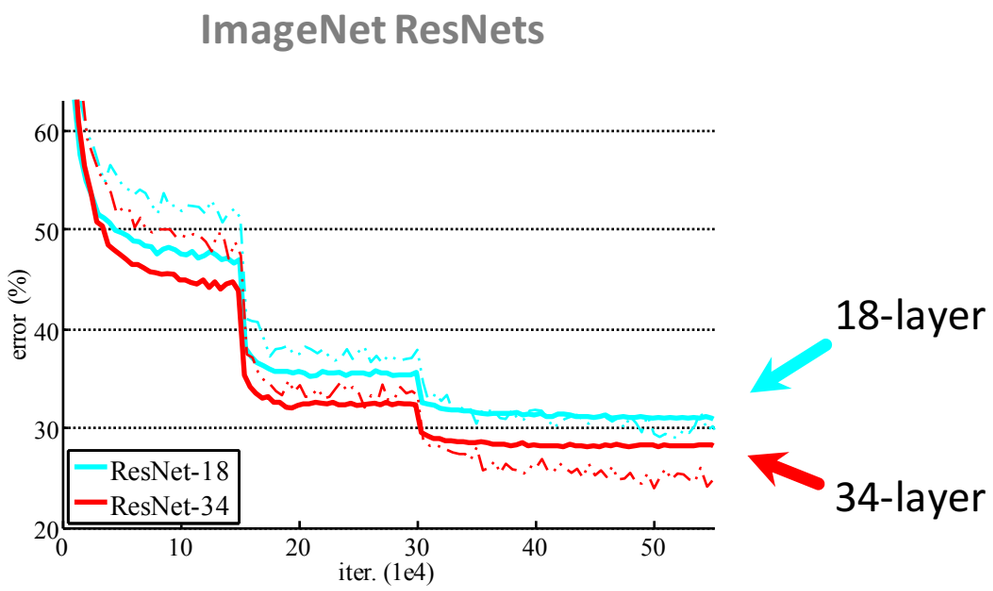
Residual Nets





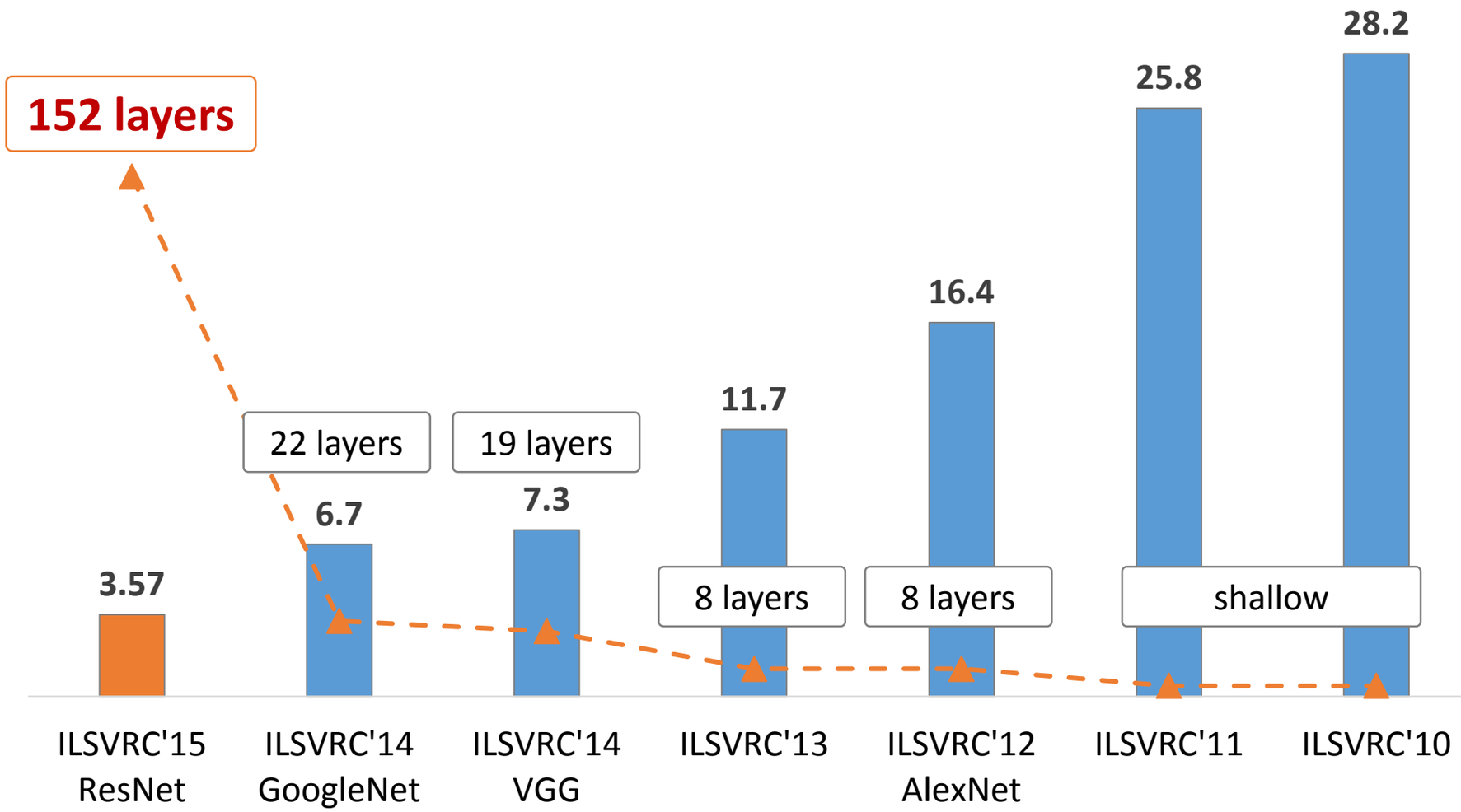
Residual Nets





© Kaimin He

Residual Nets

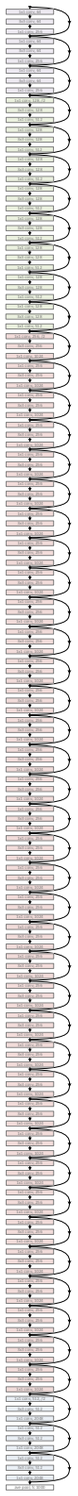


Residual Nets

AlexNet, 8 layers
(ILSVRC 2012)

VGG, 19 layers
(ILSVRC 2014)

ResNet, 152 layers
(ILSVRC 2015)



© Kaimin He

Augmentation

Simulation von Variationen der Daten

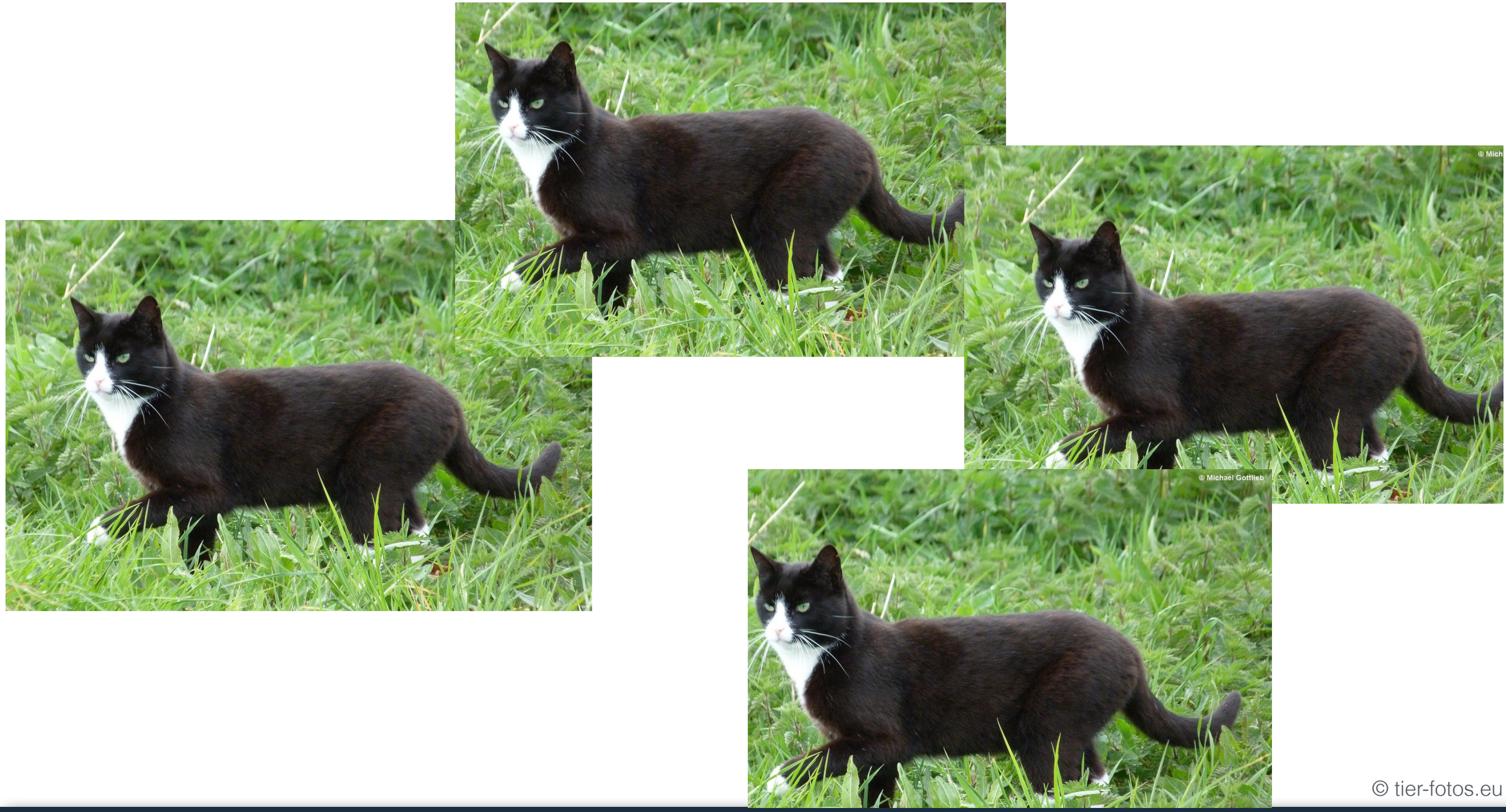
manche Variationen lassen sich gut simulieren…

© tier-fotos.eu

Augmentation

Simulation von Variationen der Daten

manche Variationen lassen sich gut simulieren…



© tier-fotos.eu

Augmentation

Simulation von Variationen der Daten

manche Variationen lassen sich gut simulieren…

© tier-fotos.eu

Augmentation

Simulation von Variationen der Daten

manche Variationen lassen sich gut simulieren…

© tier-fotos.eu

Augmentation

Simulation von Variationen der Daten

manche Variationen lassen sich gut simulieren…

© tier-fotos.eu

Augmentation

Simulation von Variationen der Daten

manche Variationen lassen sich gut simulieren..

Augmentation

Simulation von Variationen der Daten

manche Variationen lassen sich gut simulieren… aber machen keinen Sinn

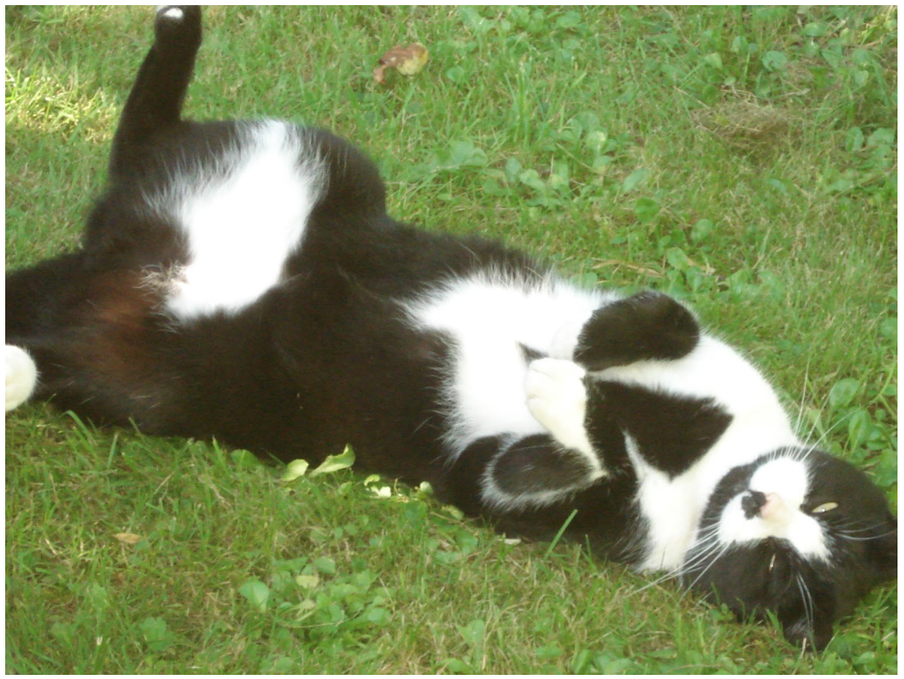
© tier-fotos.eu

Augmentation

Simulation von Variationen der Daten

manche Variationen lassen sich gut simulieren… aber machen keinen Sinn

… obwohl …



© tier-fotos.eu

Augmentation

Simulation von Variationen der Daten

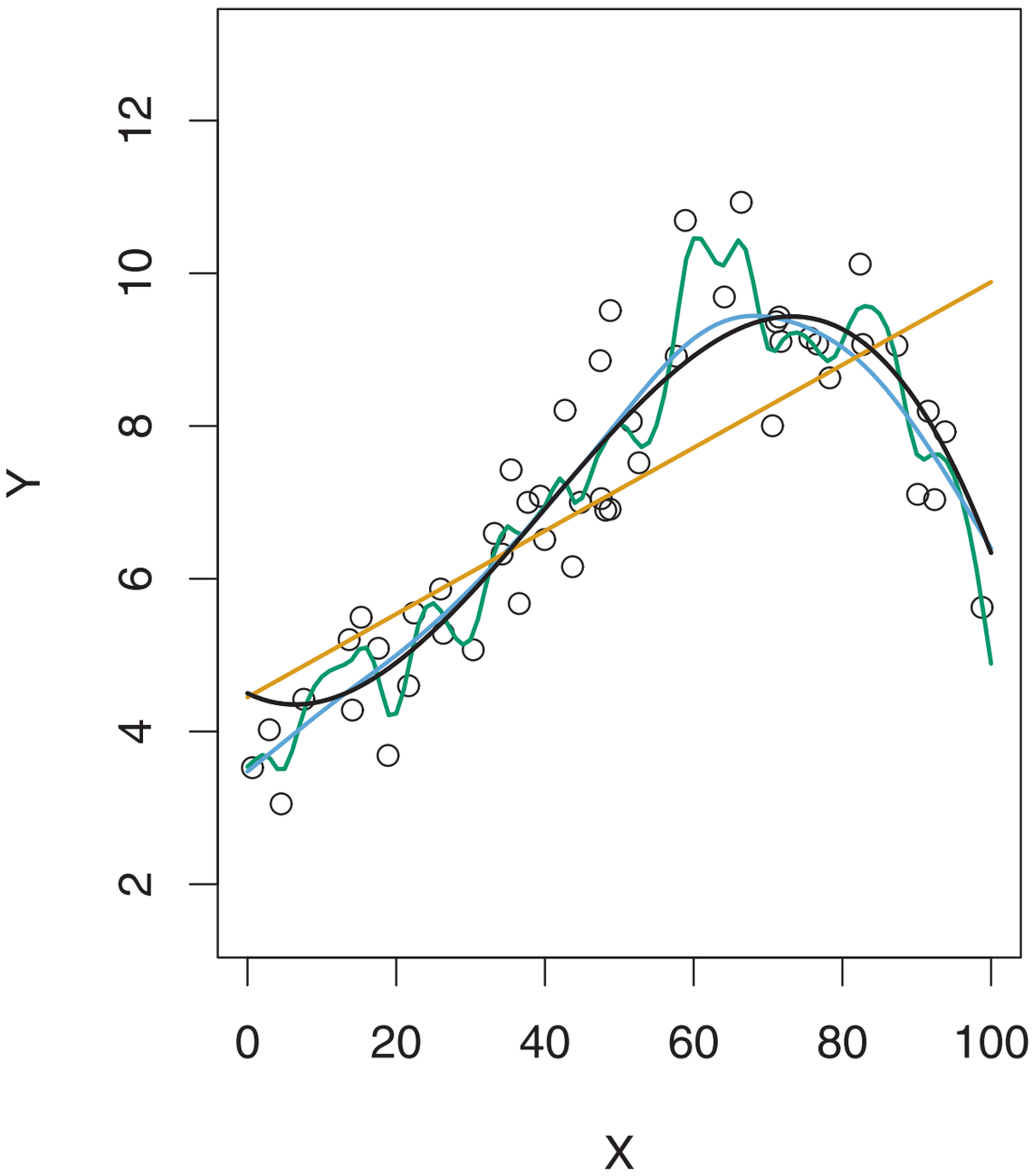
manche Variationen lassen sich kaum simulieren…

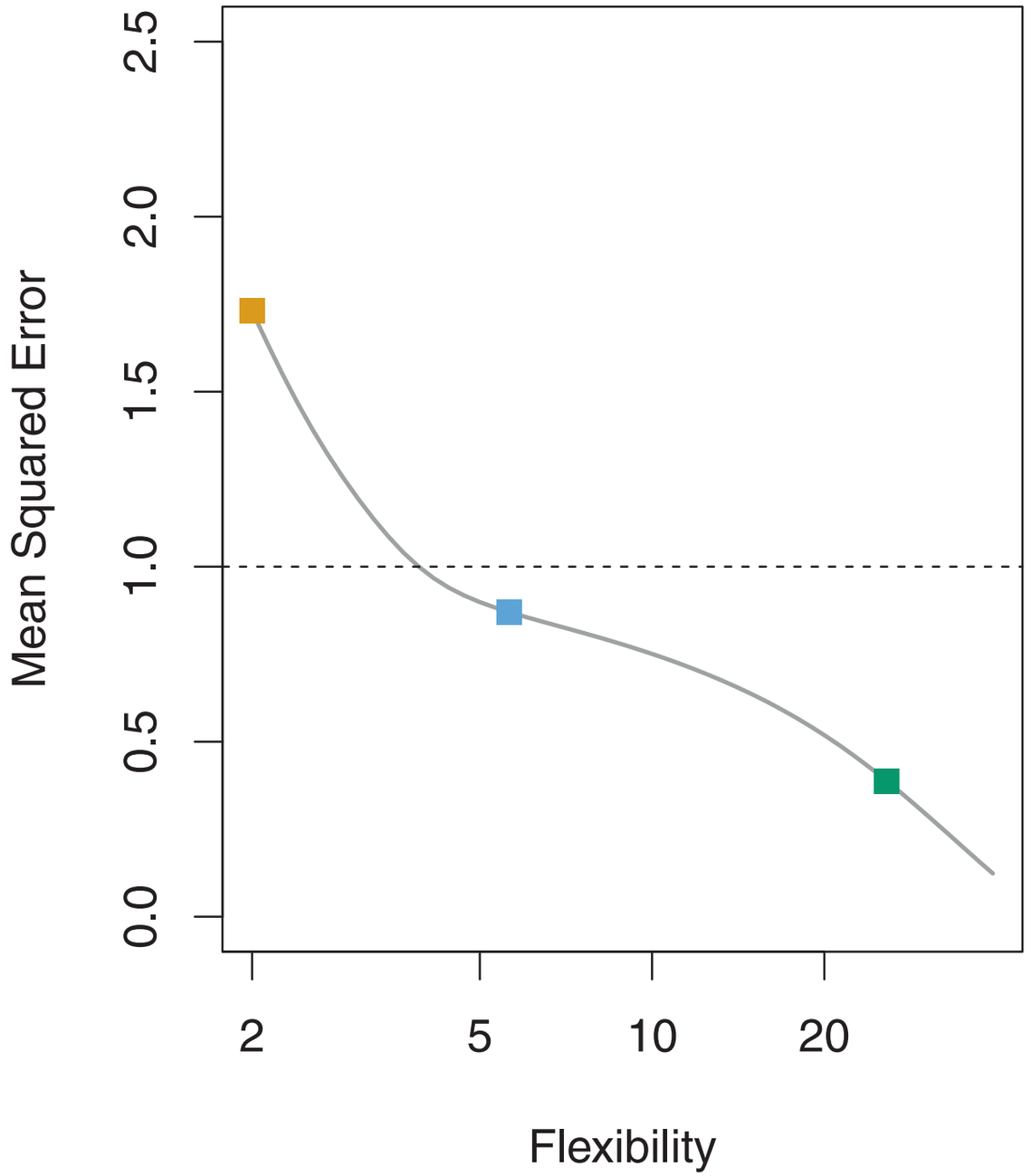
© tier-fotos.eu

Wiederholung der Grundlagen

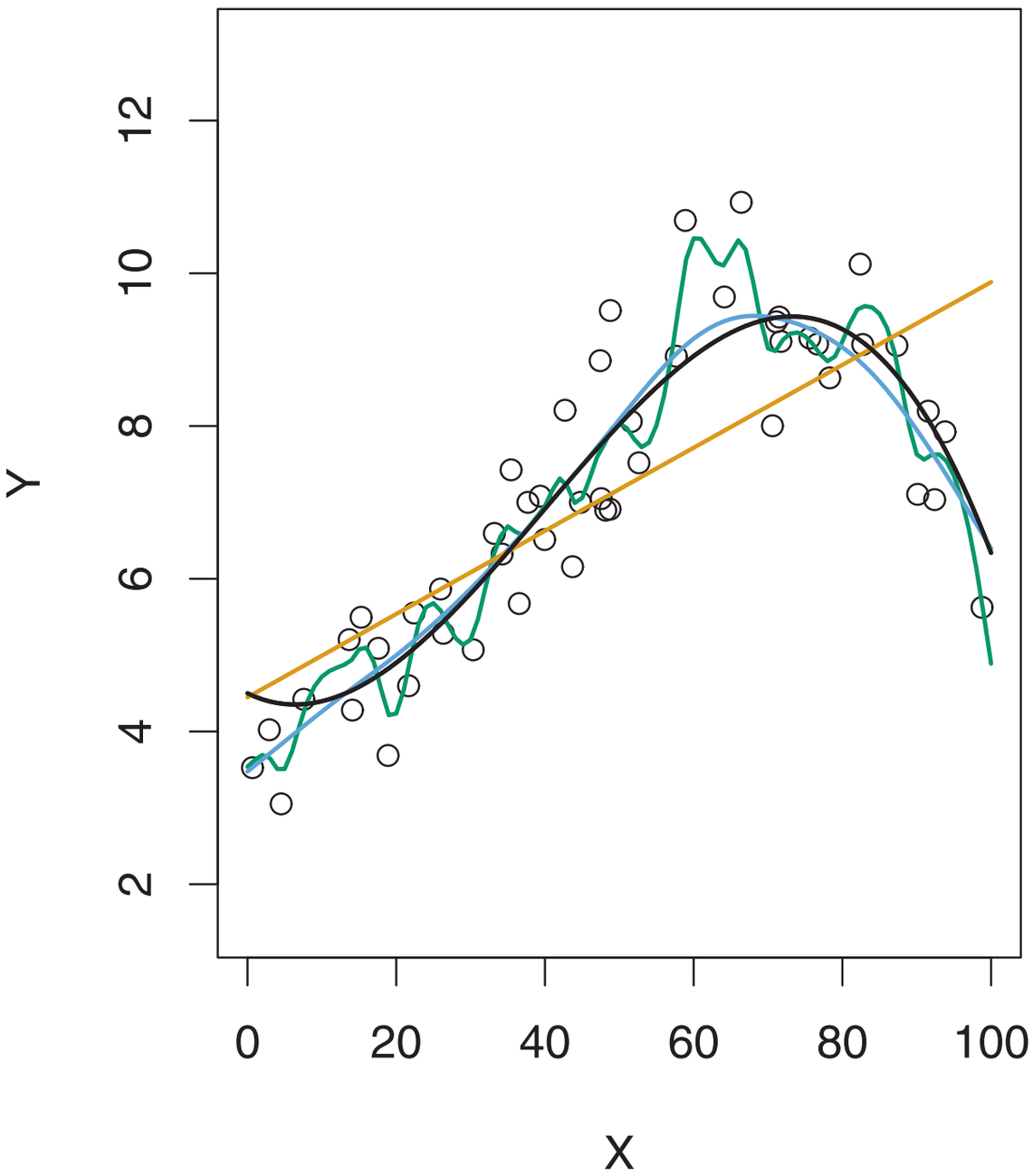
1. Regression und Klassifikation
2. Aufbau und Training von tiefen Neuronalen Netzen
3. Faltungsnetze
4. **Fehlermaße der Klassifikation**

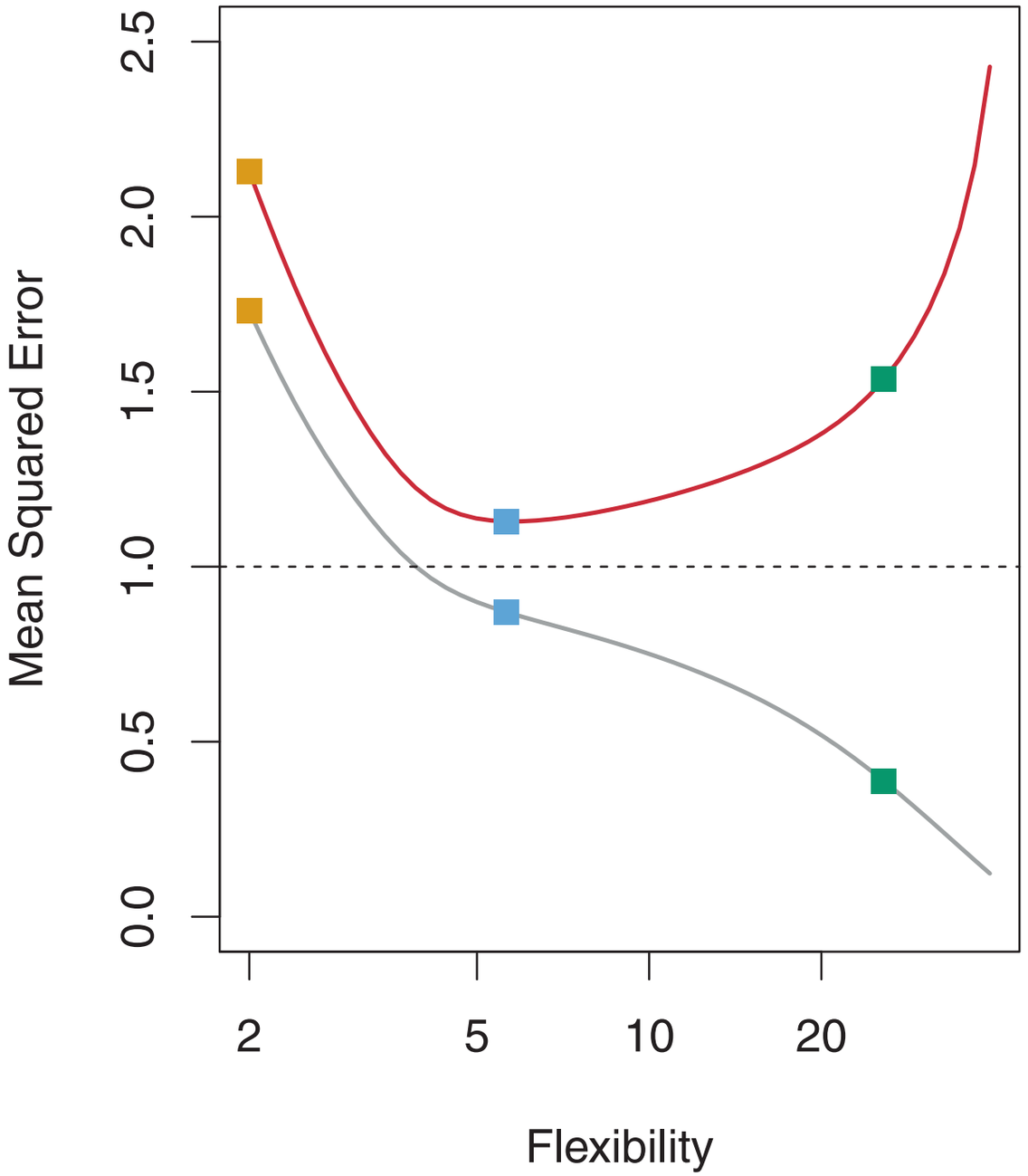
Regression





Regression





Aufteilung einer Menge von Daten

Trainingsdaten

Menge von Daten, die zum Lernen der Parameter des Klassifikators verwendet werden

Validierungsdaten

Menge von Daten, die zum Lernen der Hyperparameter eines Klassifikators verwendet werden

Testdaten

Menge von Daten, die ausschließlich zur Messung der Performanz eines vollständig spezifizierten Klassifikators verwendet werden

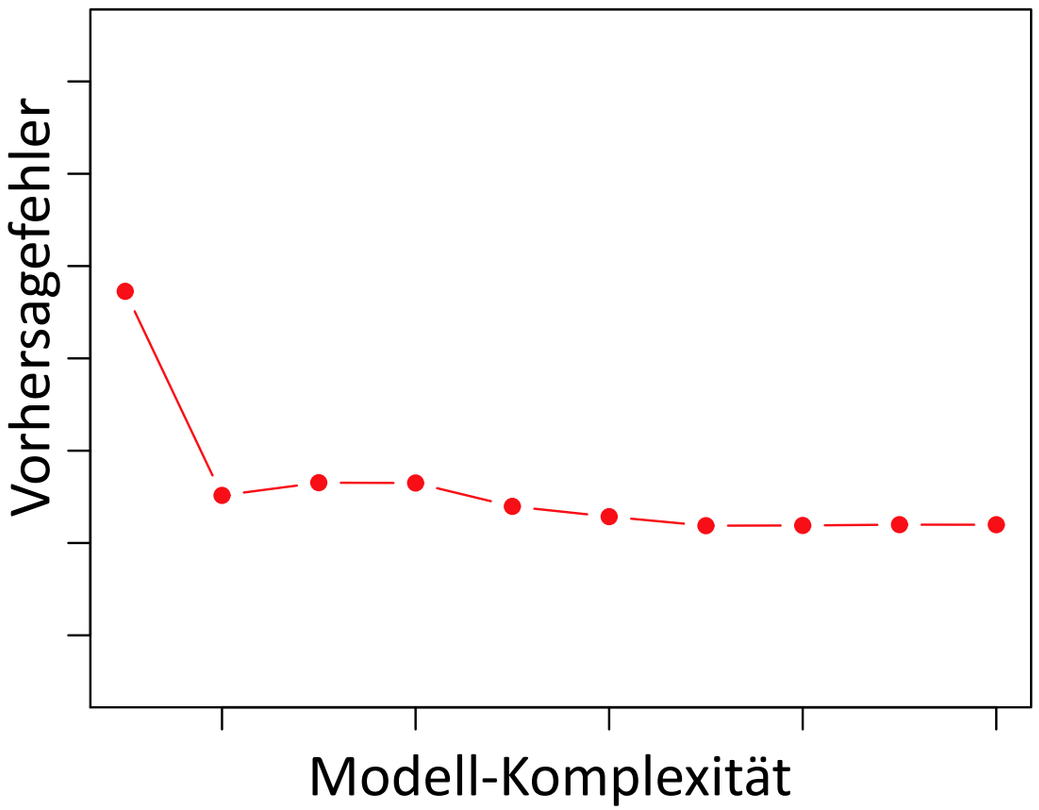
Definition by Brian Ripley, Pattern Recognition and Neural Networks, 1996

Aufteilung einer Menge von Daten

zufällige Aufteilung in Trainingsmenge und Testmenge

Beispiel

392 Daten werden aufgeteilt in 196 Trainingsdaten und 196 Testdaten



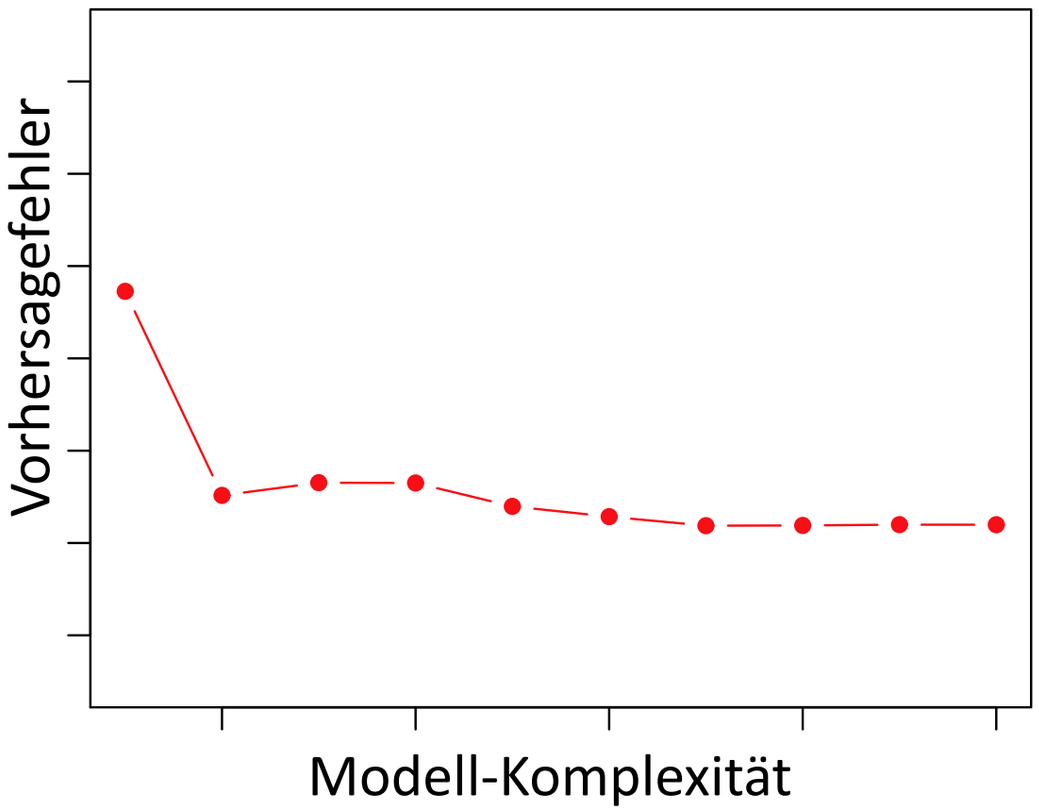
© James, Witten, Hastie, Tibshirani

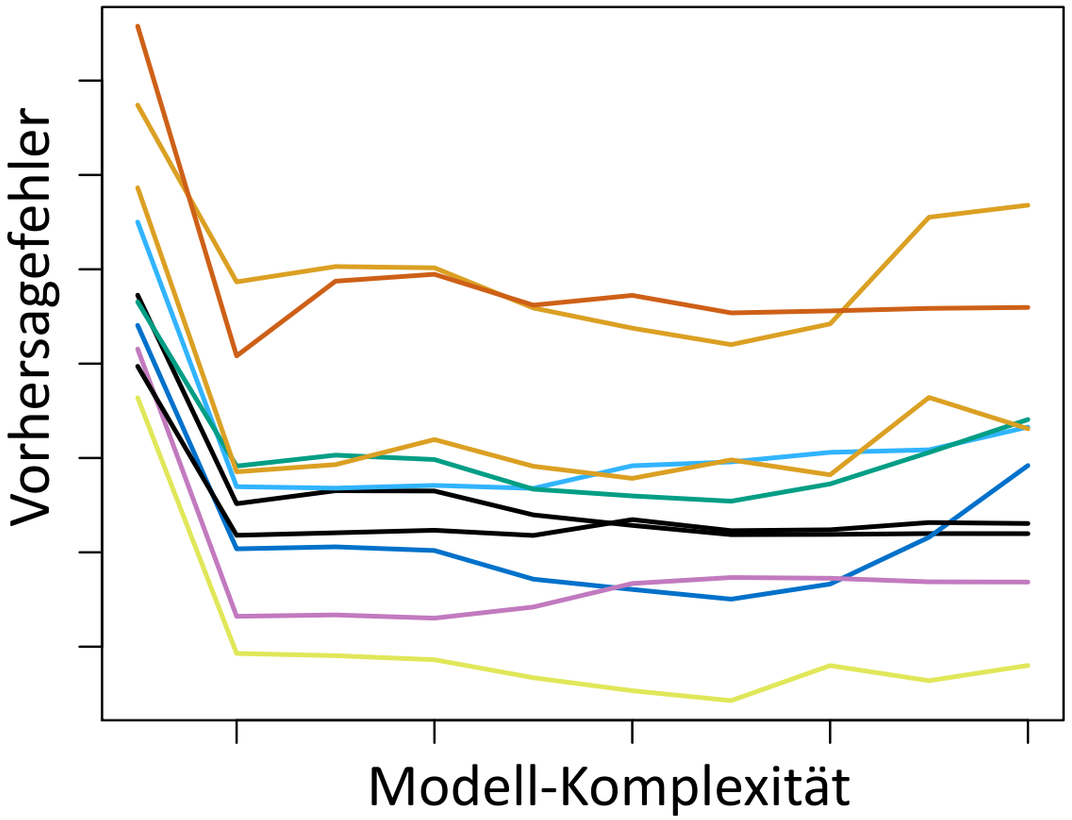
Aufteilung einer Menge von Daten

mehrfache zufällige Aufteilung in Trainingsmenge und Testmenge

Beispiel

392 Daten werden aufgeteilt in 196 Trainingsdaten und 196 Testdaten





© James, Witten, Hastie, Tibshirani

Cross Validation

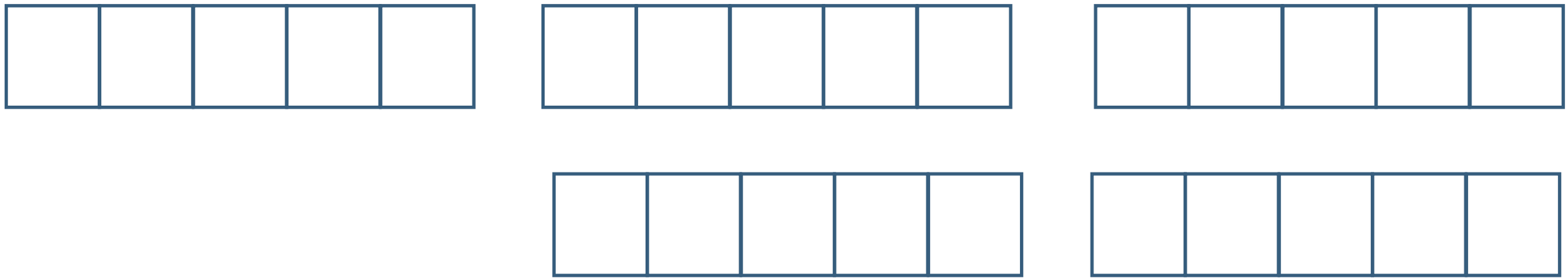
Systematische Kombination der Test- und Trainingsmengen

(1) Zufällige Aufteilung der Daten in V Teile

(2) Training mit $V-1$ Teilen, Testen mit dem verbleibenden Teil

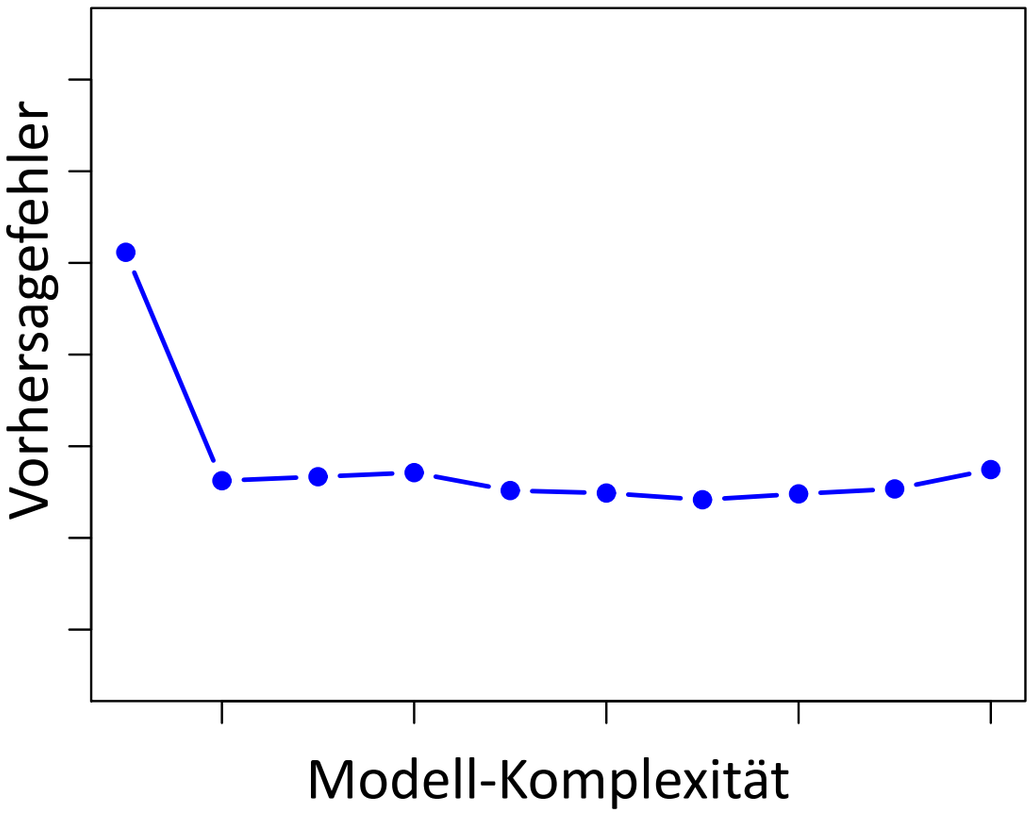
(3) Wiederhole (1)+(2) V mal

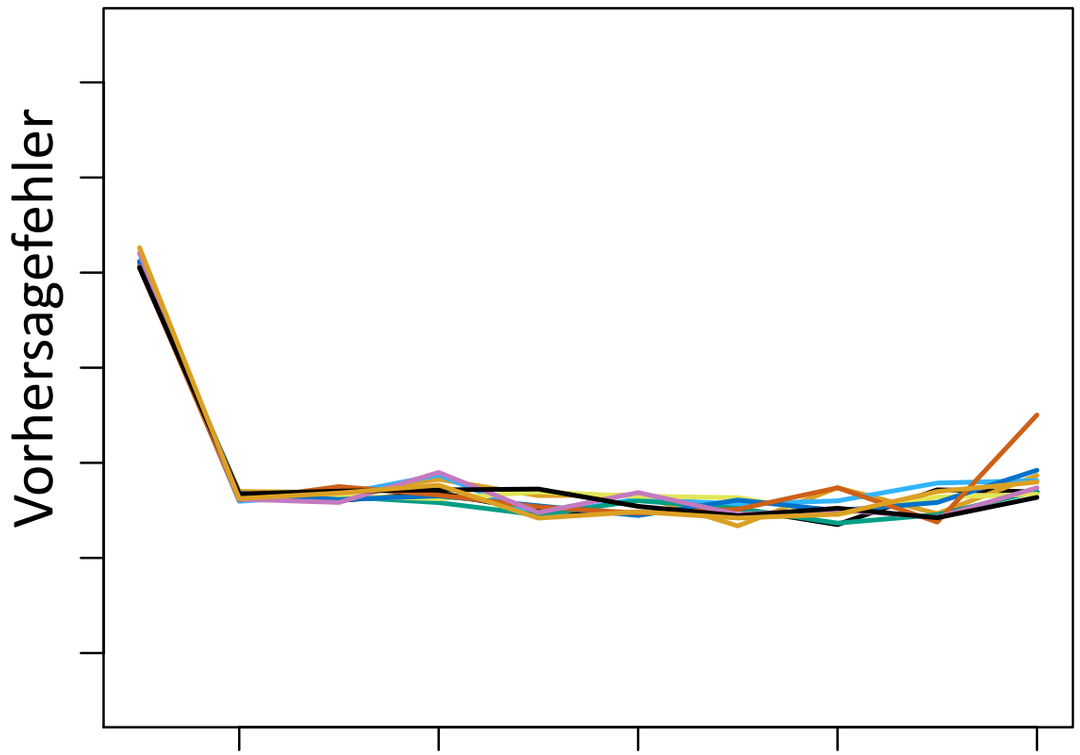
(4) berechne den mittleren Vorhersagefehler



Cross Validation

Systematische Kombination der Test- und Trainingsmengen





© James, Witten, Hastie, Tibshirani

Fehlermaße

Konfusionsmatrix

Beispiel

True negative

True positive

False negative

False positive

		vorhergesagt: nein	vorhergesagt: ja
tatsächlich	**nein**	50	10
tatsächlich	**ja**	5	100

$\sum : 165$

Fehlermaße

Konfusionsmatrix

Beispiel

True negative

True positive

False negative

False positive

		vorhergesagt: nein	vorhergesagt: ja
tatsächlich	**nein**	50	10
tatsächlich	**ja**	5	100

$\sum : 165$

$$\text{Accuracy: } \frac{\text{TP+TN}}{\text{total}}$$

$$\text{Error rate: } \frac{\text{FP+FN}}{\text{total}}$$

Fehlermaße

Konfusionsmatrix

Beispiel

True negative

True positive

False negative

False positive

		vorhergesagt: nein	vorhergesagt: ja
tatsächlich: nein		50	10
tatsächlich: ja		5	100

$\sum : 165$

$$\text{Precision: } \frac{\text{TP}}{\text{TP+FP}}$$

$$\text{Recall: } \frac{\text{TP}}{\text{TP+FN}}$$

Fehlermaße

Konfusionsmatrix

Beispiel

True negative

True positive

False negative

False positive

		vorhergesagt: nein	vorhergesagt: ja
tatsächlich: nein		50	10
tatsächlich: ja		5	100

$\sum : 165$

$$\text{Precision: } \frac{\text{TP}}{\text{TP+FP}} \qquad \text{Specificity: } \frac{\text{TN}}{\text{TN+FP}}$$

$$\text{Recall: } \frac{\text{TP}}{\text{TP+FN}} \qquad \text{Sensitivity: } \frac{\text{TP}}{\text{TP+FN}}$$

Fehlermaße

Konfusionsmatrix

Beispiel

True negative

True positive

False negative

False positive

		vorhergesagt: nein	vorhergesagt: ja
tatsächlich	**nein**	50	10
tatsächlich	**ja**	5	100

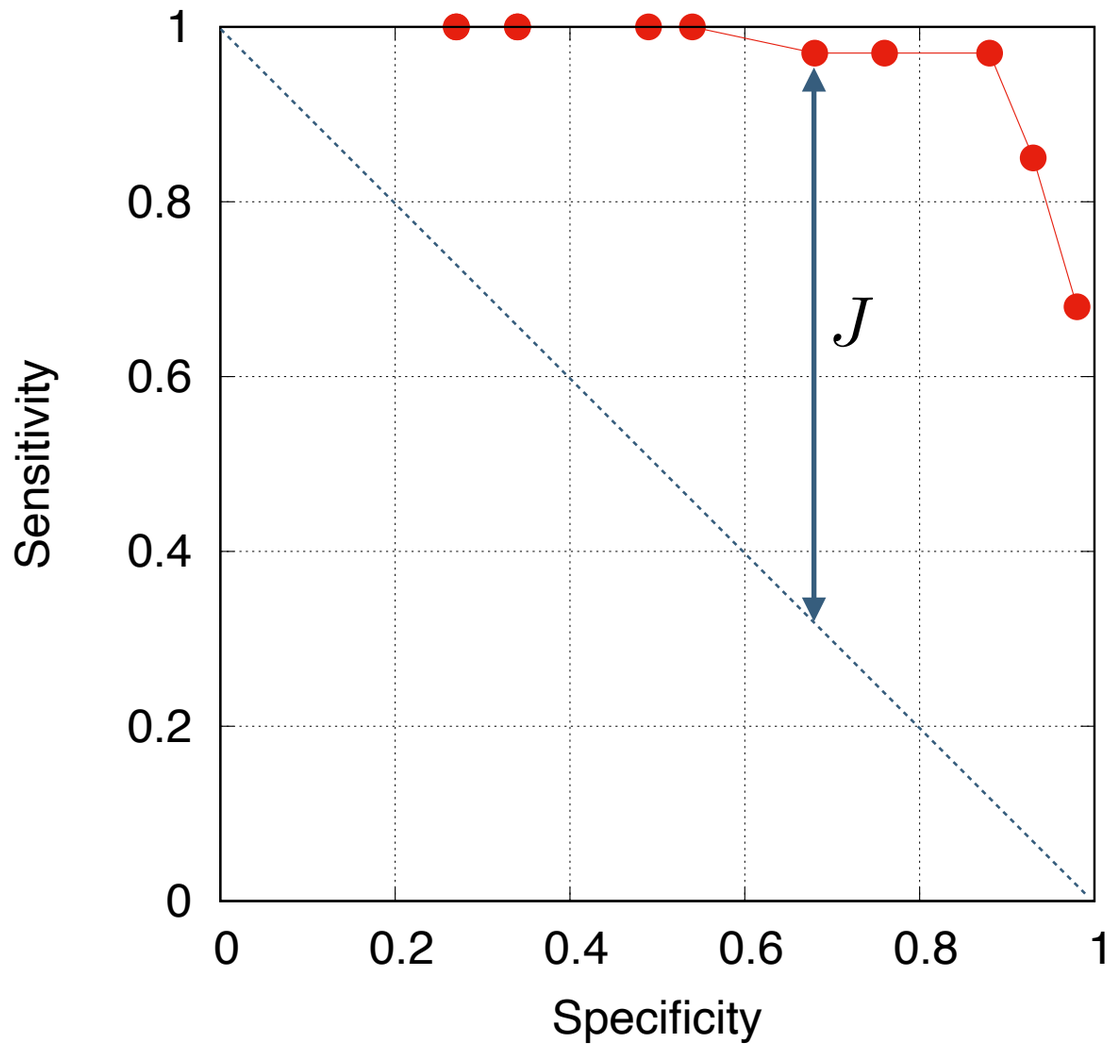
$\sum : 165$

$$\text{F1 measure: } 2 \cdot \frac{\text{Sensitivity} \cdot \text{Specificity}}{\text{Sensitivity} + \text{Specificity}}$$

ROC curve

Receiver Operator Characteristic

Youden's J statistic: $J = \text{Sensitivity} + \text{Specificity} - 1$

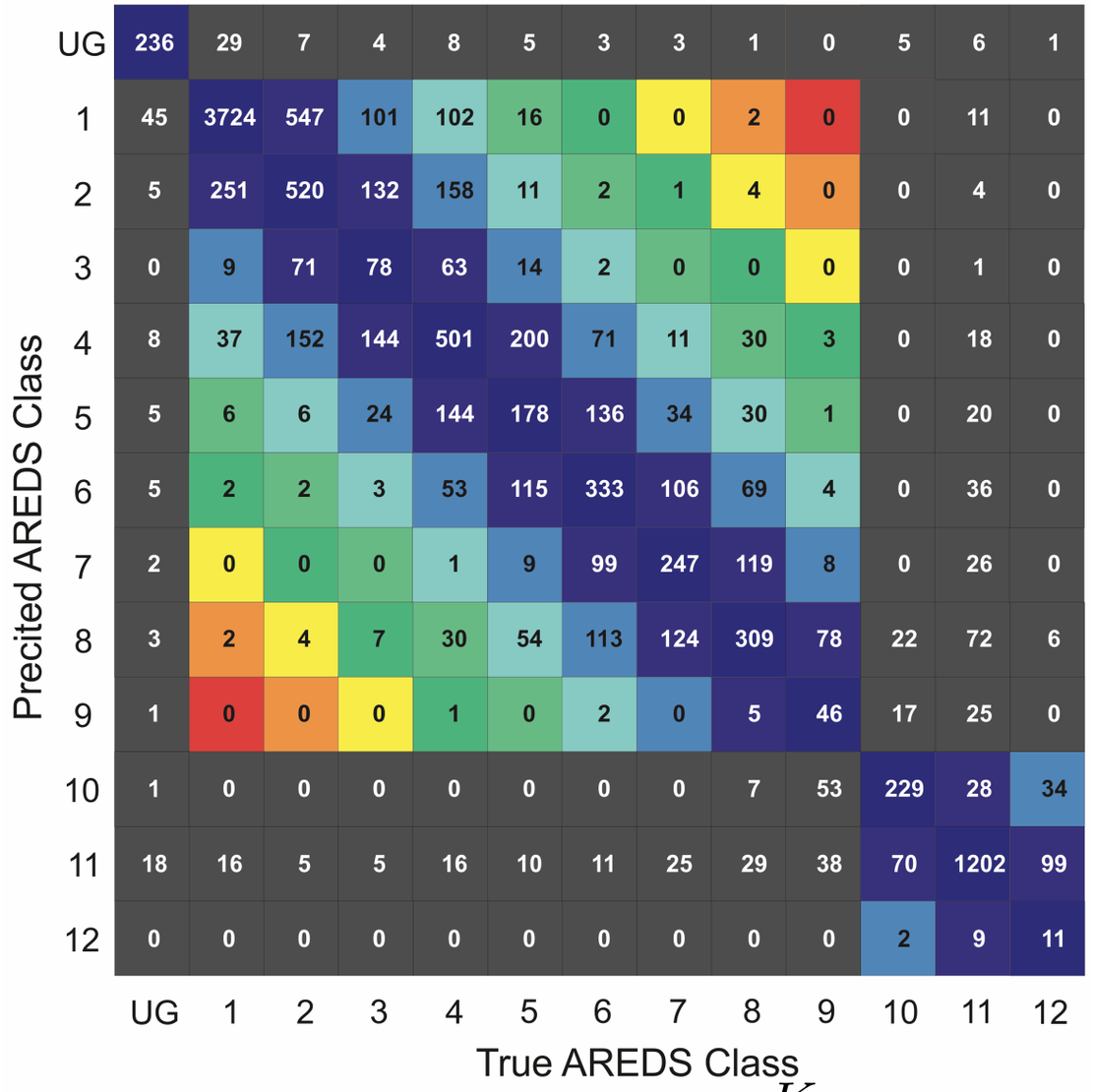


Fehlermaße

Konfusionsmatrix

für mehrere Klassen

Beispiel



$$\text{Macro-Average Precision: } \frac{1}{K}\sum_{k=1}^{K}\frac{\text{TP}_k}{\text{TP}_k + \text{FP}_k}$$

$$\text{Micro-Average Precision: } \frac{\sum_k \text{TP}_k}{\sum_k \text{TP}_k + \sum_k \text{FP}_k}$$