

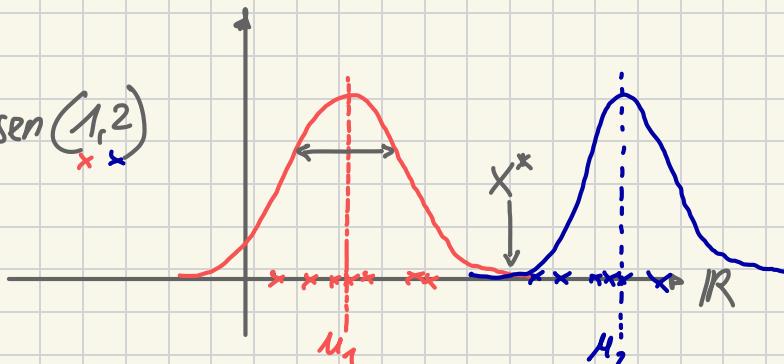
MACHINE LEARNING

14.10.29

LDA

$x \in \mathbb{R}$

zwei Klassen (1, 2)



$\bar{F}_1(x), \bar{F}_2(x)$ Verteilungsfunktionen (CDF)

$f_1(x) = \frac{\partial \bar{F}_1(x)}{\partial x}, f_2(x) = \frac{\partial \bar{F}_2(x)}{\partial x}$ Dichtefunktionen (PDF)

Annahme: 1-D Gaußverteilung für beide Klassen mit $\mu_1 < \mu_2$

$$x \sim \begin{cases} N(\underline{\mu_1}, \sigma^2_1), & \text{wenn } x \text{ aus Klasse 1} \\ N(\underline{\mu_2}, \sigma^2_2), & \text{wenn } x \text{ aus Klasse 2} \end{cases}$$

$$\Pr(\text{error}) = \Pr(x > x^*, x \in \text{Klasse 1}) + \Pr(x < x^*, x \in \text{Klasse 2})$$

$$\Pr(A, B) = \Pr(A|B) \cdot \Pr(B)$$

$$= \Pr(x > x^* | x \in \text{Klasse 1}) \cdot \Pr(x \in \text{Klasse 1}) +$$

$$\Pr(x < x^* | x \in \text{Klasse 2}) \cdot \Pr(x \in \text{Klasse 2})$$

wir wollen

minimiere $\Pr(\text{error})$
 x^*

Umschreiben mittels CDFs

$$\bullet \Pr(x < c \mid x \in \text{Klasse 1}) = F_1(c)$$

$$\Rightarrow \Pr(x > x^* \mid x \in \text{Klasse 1}) = 1 - F_1(x^*)$$

$$\bullet \Pr(x < x^* \mid x \in \text{Klasse 2}) = F_2(x^*)$$

Also, $\Pr(\text{error}) = (1 - F_1(x^*)) \cdot \bar{\pi}_1 + F_2(x^*) \cdot \bar{\pi}_2$

a-priori W-keit $\Pr(x \in \text{Klasse 1})$

Da $f_1(x) = \frac{dF_1(x)}{dx}$ und $f_2(x) = \frac{dF_2(x)}{dx}$, erhalten wir

$$\frac{\Pr(\text{error})}{\partial x^*} = -f_1(x^*) \cdot \bar{\pi}_1 + f_2(x^*) \cdot \bar{\pi}_2 = 0 \quad // 0 \text{ setzen}$$

$$\rightarrow f_1(x^*) \cdot \bar{\pi}_1 = f_2(x^*) \cdot \bar{\pi}_2 \quad (\times)$$

Alternativ (mit Bayes):

$$\Pr(x \in \text{Klasse 1} \mid X=x) = \Pr(x \in \text{Klasse 2} \mid X=x)$$

$f_1(x)$ A-posteriori W-keit dass x zu Klasse 1 gehört!

$$\text{Boyes: } \Pr(X=x \mid x \in \text{Klasse 1}) \cdot \Pr(x \in \text{Klasse 1}) = \Pr(X=x \mid x \in \text{Klasse 2}) \cdot \Pr(x \in \text{Klasse 2})$$

Normalisierung

Normalisierung

gleich

$$\Rightarrow f_1(x) \cdot \bar{\pi}_1 = f_2(x) \cdot \bar{\pi}_2$$

Einschub: Multivariate Normalverteilung; $x \in \mathbb{R}^d$

$$x \sim N(\mu, \Sigma) \quad \underbrace{\mu \in \mathbb{R}^d, \Sigma \in \mathbb{R}^{d \times d}}_{\text{Kovarianzmatrix}}$$

Dichtefunktion:

$$f(x) = \frac{1}{\sqrt{(2\pi)^d \cdot |\Sigma|}} \cdot e^{-\frac{(x-\mu)^T \Sigma^{-1} (x-\mu)}{2}}$$

Determinante von Σ

Wir nehmen an: f_1, f_2 multivariate Normalverteilungen mit $(\mu_1, \Sigma), (\mu_2, \Sigma)$

$$\frac{1}{\sqrt{(2\pi)^d \cdot |\Sigma|}} \cdot e^{-\frac{(x-\mu_1)^T \Sigma^{-1} (x-\mu_1)}{2}} \cdot \pi_1 = \frac{1}{\sqrt{(2\pi)^d \cdot |\Sigma|}} \cdot e^{-\frac{(x-\mu_2)^T \Sigma^{-1} (x-\mu_2)}{2}} \cdot \pi_2$$

$$\log \Rightarrow -\frac{1}{2} \cdot \underbrace{(x-\mu_1)^T \Sigma^{-1} (x-\mu_1)} + \ln(\pi_1) = -\frac{1}{2} \cdot \underbrace{(x-\mu_2)^T \Sigma^{-1} (x-\mu_2)} + \ln(\pi_2)$$

$$x^T \Sigma^{-1} x - \underbrace{x^T \Sigma^{-1} \mu_1 - \mu_1^T \Sigma^{-1} x}_{-2 \mu_1^T \Sigma^{-1} x} + \mu_1^T \Sigma^{-1} \mu_1$$

$$(x-\mu)^T = x^T - \mu^T$$

$$\Rightarrow -\frac{1}{2} \cdot x^T \Sigma^{-1} x + \mu_1^T \Sigma^{-1} x - \frac{1}{2} \cdot \mu_1^T \Sigma^{-1} \mu_1 + \ln(\pi_1) = -\frac{1}{2} \cdot x^T \Sigma^{-1} x + \mu_2^T \Sigma^{-1} x - \frac{1}{2} \cdot \mu_2^T \Sigma^{-1} \mu_2 + \ln(\pi_2)$$

Zusammenfassen ($\times 2$, links \rightarrow rechts):

$$2 \cdot \ln\left(\frac{\pi_2}{\pi_1}\right) + 2 \cdot \left(\Sigma^{-1}(\mu_2 - \mu_1)\right)^T x + (\mu_1 - \mu_2)^T \Sigma^{-1} (\mu_1 - \mu_2) = 0$$

$$\Leftrightarrow \underbrace{2 \cdot \left(\Sigma^{-1}(\mu_2 - \mu_1)\right)^T}_{Q^T} x + \underbrace{(\mu_1 - \mu_2)^T \Sigma^{-1} (\mu_1 - \mu_2)}_b + 2 \cdot \ln\left(\frac{\pi_2}{\pi_1}\right) = 0$$

\Rightarrow Lineare Entscheidungsspannen!
(da von der Form $Q^T x + b$)

Entscheidungsregel:

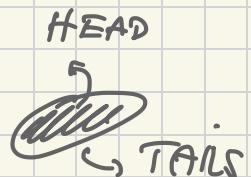
Def: $f(x) := 2 \cdot \left(\Sigma^{-1} (\mu_2 - \mu_1) \right)^T x + (\mu_2 - \mu_1)^T \Sigma^{-1} (\mu_1 - \mu_2) + 2 \ln \left(\frac{\pi_2}{\pi_1} \right)$

$\hat{C}(x) = \begin{cases} \text{Klasse 1, wenn } f(x) < 0 \\ \text{Klasse 2, wenn } f(x) > 0 \end{cases}$

Prediction

Einschub: MLE - Bernoulli Verteilung

Zufallsvariable $y = 0$ TAILS
 $y = 1$ HEAD



Annahme: $\Pr(y=1) = \theta$ $(0 \leq \theta \leq 1)$
 $\Pr(y=0) = 1-\theta$

Wir schreiben $y \sim \text{Ber}(\theta)$

$\mathcal{D} = \{y_n : n=1, \dots, N\}$

$\hookrightarrow \text{Ber}(y|\theta) = \theta^{\prod_{y=1}^N} \cdot (1-\theta)^{\prod_{y=0}^N}$

$\left(\prod_{y=1}^N = \begin{cases} 1, \text{ wenn } y=1 \\ 0, \text{ sonst} \end{cases} \right)$

$P(\mathcal{D}|\theta) = \prod_{n=1}^N \theta^{\prod_{y=1}^{y_n}} \cdot (1-\theta)^{\prod_{y=0}^{y_n}}$

$$\ln(p(D|\theta)) = \sum_{n=1}^N \underbrace{1}_{y_n=1} \cdot \ln(\theta) + \underbrace{1}_{y_n=0} \cdot \ln(1-\theta)$$

.

$$= \underbrace{N_1 \cdot \ln(\theta)}_{\hookrightarrow \text{Häufigkeit von } y=1} + N_0 \cdot \ln(1-\theta)$$

$$\underbrace{\text{NLL}(\theta)}_{\text{Negative Log-Likelihood}} = -\ln(p(D|\theta)) = -N_1 \cdot \ln(\theta) - N_0 \cdot \ln(1-\theta)$$

Negative Log-Likelihood

$$\frac{\partial \text{NLL}(\theta)}{\partial \theta} = \frac{-N_1}{\theta} + \frac{N_0}{1-\theta} \quad // 0 \text{ setzen}$$

$$\Rightarrow -\frac{N_1}{\theta} + \frac{N_0}{1-\theta} = 0 \quad \Rightarrow \quad 0$$

Beispiel (MLE): Kategoriale Verteilung (categorical distribution)

(z.B.: C-Seiten Würf/: Wir modellieren den Ausgang eines Wurfs als Zufallsvariable $Y_n \in \{1, \dots, C\}$)

$$Y_n \sim \text{Cat}(\Theta) \quad 0 \leq \Theta_c \leq 1, \sum_{c=1}^C \Theta_c = 1$$

$$(\Pr(Y=c) = \Theta_c)$$

Wahrscheinlichkeitsmesse-Funktion:

$$\text{Cat}(y|\Theta) = \prod_{c=1}^C \Theta_c^{\mathbf{1}_{y=c}}$$

Auf Basis von $D = \{y_1, \dots, y_N\}$ würden wir gerne den MLE von Θ bestimmen.

$$p(D|\Theta) = \prod_{n=1}^N \prod_{c=1}^C \Theta_c^{\mathbf{1}_{y_n=c}} \quad | \log$$

$$\ln p(D|\Theta) = \sum_n \sum_c \mathbf{1}_{y_n=c} \cdot \ln(\Theta_c)$$

$$\Rightarrow \text{NLL}(\Theta) = - \sum_n \sum_c \mathbf{1}_{y_n=c} \cdot \ln(\Theta_c)$$

$$= - \sum_c N_c \cdot \ln(\Theta_c) \quad \text{mit}$$

$$N_c = \sum_n \mathbf{1}_{y_n=c}$$

Hinweis: Minimieren von $\text{NLL}(\Theta)$ unter Nebenbedingung $\sum_c \Theta_c = 1$.

Lagrange Multiplizieren (2D); Funktion $f(x,y)$, Nebenbedingung $g(x,y)=0$

1. Lagrangepunkt: $L(x,y,\lambda) = f(x,y) - \lambda \cdot g(x,y)$

2. Wir suchen

$$\nabla_{x,y,d} L(x, y, d) = 0 \quad \text{unter } \nabla f \neq 0$$

Anm.: $\begin{cases} \frac{\partial L}{\partial x} = \frac{\partial f}{\partial x} - d \cdot \frac{\partial g}{\partial x} \\ \frac{\partial L}{\partial y} = \frac{\partial f}{\partial y} - d \cdot \frac{\partial g}{\partial y} \end{cases} = 0 \quad (\text{also } \nabla f = d \nabla g)$

und $\frac{\partial L}{\partial d} = 0 \rightarrow \underbrace{g(x, y)}_0 = 0$

unsere Nebenbedingung

3. Gleichungssystem lösen

Hinweis: zu sehen ob die gefundenen Extrema Minima od. Maxima sind, ist nicht immer unmittelbar klar (siehe Optimierungs-Lit.).

Bei uns (im Fall der kategorialen Verteilung):

$$\begin{aligned} L(\theta, d) &= -NLL(\theta) - d \cdot \left(1 - \sum_c \theta_c\right) \\ &= -\sum_c N_c \cdot \ln(\theta_c) - d \cdot \left(1 - \sum_c \theta_c\right) \end{aligned}$$

$$\frac{\partial L(\theta, d)}{\partial \theta_c} = -\frac{N_c}{\theta_c} + d \stackrel{\text{null setzen}}{\Rightarrow} N_c = d \cdot \theta_c \quad (\times)$$

$$\frac{\partial L(\theta, d)}{\partial d} = -\left(1 - \sum_c \theta_c\right) \stackrel{--}{\Rightarrow} \sum_c \theta_c = 1 \quad (\times)$$

Wir wissen $\sum_c N_c = N \stackrel{(\times)}{\Rightarrow} \underbrace{\sum_c N_c}_N = d \cdot \underbrace{\sum_c \theta_c}_1 = d = N$

Also $\hat{\theta}_c = \frac{N_c}{d} = \frac{N_c}{N}$

MLE

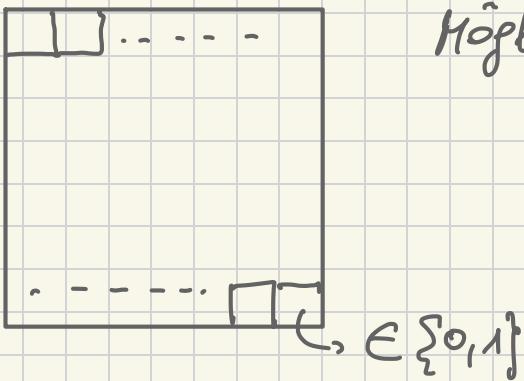
Betrachten wir einen Spezialfall:

$$p(y=c|x, \{\theta_j\}) = \frac{p(x|y=c, \theta) \cdot p(y=c|\pi)}{\text{Normierung}} ; x \in \mathbb{R}^d$$

$\Rightarrow \prod_{j=1}^d p(x_j|y=c, \theta_{jc})$

Ein solches Modell nennt man NAIVE BAYES Klassifizierer.

Beispiel: Binäre Bilddaten



Möglicher Name: "Bernoulli Naive Bayes"

LOGISTISCHE REGRESSION (LR)

Im Gegensatz zu LDA ein diskriminativer Ansatz!
 wir versuchen $p(y=c|x)$ direkt
zu modellieren!

1. Binärer Fall (also $y \in \{0,1\}$)

Da $y \in \{0,1\}$, bietet es sich an, $p(y|x)$ mittels einer Bernoulli-Verteilung zu modellieren, wobei wir den Parameter der Bernoulli-Verteilung abhängig vom Input (x) machen, z.B.:

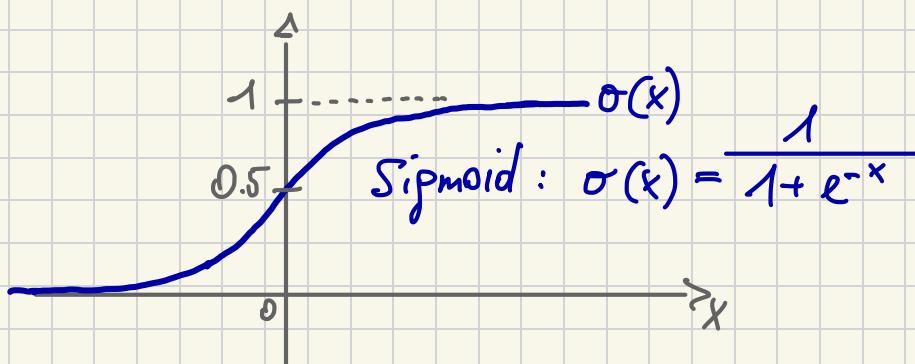
$$\text{Ber}(y | f(x; \alpha))$$

Funktion von x mit Parameter α
 (wir wollen $f(x; \alpha) \in [0, 1]$)

Nehmen wir an, dass $x \in \mathbb{R}^d$ und $y \in \{0,1\}$; in der LR haben wir folgendes Modell:

$$p(y|x, \theta) = \text{Ber}(y | \sigma(w^T x + b))$$

inkludiert Parameter $w \in \mathbb{R}^d$ und $b \in \mathbb{R}$



Setzen wir $\alpha = \omega^T x + b$; wir haben also konkret

$$p(y=1|x, \theta) = \sigma(\alpha) = \frac{1}{1+e^{-\alpha}}$$

$$p(y=0|x, \theta) = 1 - p(y=1|x, \theta)$$

$$= 1 - \frac{1}{1+e^{-\alpha}} = \frac{e^{-\alpha}}{1+e^{-\alpha}} = \frac{1}{1+e^{\alpha}} = \sigma(-\alpha)$$

Anm.: Setzen wir $p(y=1|x, \theta) = q$ und betrachten

$$\log \left(\underbrace{\frac{q}{1-q}}_{\text{"odds"}, \text{log-odds}} \right)$$

$$\log \left(\frac{q}{1-q} \right) = \log \left(\frac{e^\alpha}{1+e^\alpha} \cdot \frac{1+e^\alpha}{1} \right) = \log(e^\alpha) = \alpha (= \omega^T x + b)$$

Anmerkung: $P(y|x, \{w, b\}) = \text{Ber}(y | \sigma(w^T x + b))$, $x \in \mathbb{R}^d$, $w \in \mathbb{R}^d$, $b \in \mathbb{R}$

Nehmen wir nur

$$\hat{w} = \begin{pmatrix} b \\ w_1 \\ \vdots \\ w_d \end{pmatrix}, \quad \hat{x} = \begin{pmatrix} 1 \\ x_1 \\ \vdots \\ x_d \end{pmatrix} \quad \hat{x}, \hat{w} \in \mathbb{R}^{d+1}$$

$$\Rightarrow \langle w, x \rangle + b = \langle \hat{w}, \hat{x} \rangle$$

In Allgemeinen werden wir immer nur $\langle w, x \rangle$ ab jetzt schreiben.
 entweder als $\in \mathbb{R}^d$, oder $\in \mathbb{R}^{d+1}$

$$\begin{aligned} NLL(w) &= -\frac{1}{N} \log P(\mathcal{D}|w) \\ &= -\frac{1}{N} \cdot \log \left(\prod_{n=1}^N \text{Ber}(y_n | \sigma(w^T x_n)) \right) \end{aligned}$$

Wir setzen: $\mu_n = \sigma(w^T x_n)$

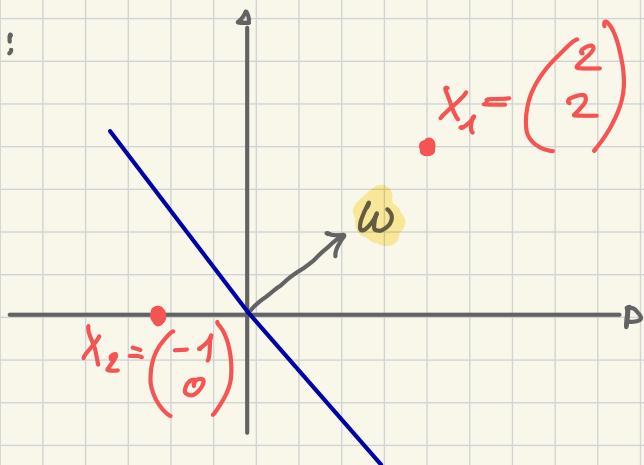
L_D

$$= -\frac{1}{N} \cdot \log \left(\prod_{n=1}^N \text{Ber}(y_n | \mu_n) \right) \in [0, 1]$$

$$= -\frac{1}{N} \cdot \sum_{n=1}^N \log (\text{Ber}(y_n | \mu_n))$$

Anmerkung:

$$\omega = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$



$$\begin{aligned} \omega^T x_1 &= \left\langle \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 2 \\ 2 \end{pmatrix} \right\rangle = 4 \\ \omega^T x_2 &= \left\langle \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} -1 \\ 0 \end{pmatrix} \right\rangle = -1 \end{aligned}$$

$$\begin{aligned} NLL(\omega) &= -\frac{1}{N} \cdot \sum_{n=1}^N \log (\mu_n^{y_n} \cdot (1-\mu_n)^{1-y_n}) \\ &= -\frac{1}{N} \cdot \sum_{n=1}^N \left[y_n \cdot \log(\mu_n) + (1-y_n) \cdot \log(1-\mu_n) \right] \\ &\quad \hookrightarrow \sigma(\omega^T x_n) \end{aligned}$$

Wie minimieren wir $NLL(\omega)$?

$$\nabla_{\omega} NLL(\omega) = 0$$

$$\nabla_{\omega} NLL(\omega) = \begin{pmatrix} \frac{\partial NLL(\omega)}{\partial \omega_1} \\ \vdots \\ \frac{\partial NLL(\omega)}{\partial \omega_d} \end{pmatrix}$$

Erinnerung: $q_n = \omega^T x_n$

$$\mu_n = \sigma(q_n) = \sigma(\omega^T x_n)$$

$$\text{Wir betrachten: } \frac{\partial \mu_n}{\partial \omega_d} = \sigma(q_n) \cdot (1 - \sigma(q_n))$$

$$\begin{aligned} \frac{\partial \mu_n}{\partial \omega_d} &= \frac{\partial}{\partial \omega_d} \sigma(\omega^T x_n) = \frac{\partial}{\partial q_n} \sigma(q_n) \cdot \underbrace{\frac{\partial}{\partial \omega_d} q_n}_{=} \\ &= \sigma(q_n) \cdot (1 - \sigma(q_n)) \cdot \underbrace{x_{nd}}_{\text{d-te Koordinate von } x_n} \end{aligned}$$

$$\begin{aligned} \text{Wir erhalten also } \frac{\partial}{\partial \omega_d} \mu_n &= \underbrace{\sigma(q_n)}_{\mu_n} \cdot \underbrace{(1 - \sigma(q_n))}_{(1 - \mu_n)} \cdot x_{nd} \\ &= \mu_n \cdot (1 - \mu_n) \cdot x_{nd} \end{aligned}$$

Jetzt können wir den Gradienten $\nabla_{\omega} \log(\mu_n)$

$$\nabla_{\omega} \log(\mu_n) = \frac{1}{\mu_n} \cdot \nabla_{\omega} \mu_n$$

$$= \begin{pmatrix} \frac{1}{\mu_n} \cdot \frac{\partial}{\partial \omega_1} \mu_n \\ \vdots \\ \frac{1}{\mu_n} \cdot \frac{\partial}{\partial \omega_d} \mu_n \end{pmatrix}$$



$$\Rightarrow \nabla_w \log(m_n) = \frac{1}{m_n} m_n \cdot (1-m_n) \cdot \begin{pmatrix} x_{n1} \\ x_{nd} \end{pmatrix}$$

$= (1-m_n) \cdot \begin{pmatrix} x_{n1} \\ \vdots \\ x_{nd} \end{pmatrix}$
(x)

Auf gleiche Art u. Weise erhalten wir

$$\nabla_w \log(1-m_n) = -m_n \cdot \begin{pmatrix} x_{n1} \\ \vdots \\ x_{nd} \end{pmatrix}$$
(xx)

Für den Gradienten $\nabla_w NLL(w)$ folgt

$$\begin{aligned} \nabla_w NLL(w) &= -\frac{1}{N} \cdot \sum_{n=1}^N \left[y_n \cdot \underbrace{(1-m_n) \cdot \vec{x}_n}_{(x)} - (1-y_n) \cdot \underbrace{m_n \vec{x}_n}_{(xx)} \right] \\ &= -\frac{1}{N} \cdot \sum_{n=1}^N \left[y_n \vec{x}_n - y_n m_n \vec{x}_n - m_n \vec{x}_n + y_n m_n \vec{x}_n \right] \\ &= -\frac{1}{N} \cdot \sum_{n=1}^N \left[y_n \vec{x}_n - m_n \vec{x}_n \right] \\ &= \frac{1}{N} \cdot \sum_{n=1}^N (m_n - y_n) \vec{x}_n \end{aligned}$$

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

$$\Theta(w^T x_n) = \frac{1}{1+e^{-w^T x_n}}$$

In Matrizen Schreibweise: wir definieren

$$X = \begin{pmatrix} x_1 & \dots & x_d \\ \vdots & & \vdots \\ x_N & \dots & x_{Nd} \end{pmatrix} \in \mathbb{R}^{N \times d}$$

("Design Matrix")

Datenpunkte als Zeilenvektoren

Also,

$$\nabla_{\omega} NLL(\omega) = X^T (\vec{\mu} - \vec{y}) \frac{1}{N},$$

$\underbrace{d \times N}_{d \times 1} \quad \underbrace{N \times 1}_{d \times 1}$

$$\vec{\mu} = \begin{pmatrix} \mu_1 \\ \vdots \\ \mu_N \end{pmatrix}, \vec{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_N \end{pmatrix}$$

Betrachten wir die Hesse Matrix:

$$\underbrace{\nabla_{\omega}^2 NLL(\omega)}_{\text{nur eine Komponente}} = \frac{1}{N} \cdot \sum_{n=1}^N (\mu_n - y_n) \vec{x}_n$$

$$\text{nur eine Komponente: } \frac{\partial}{\partial \omega_j} NLL(\omega) = \frac{1}{N} \cdot \sum_{n=1}^N (\mu_n - y_n) \cdot x_{nj}$$

(Ignorieren wir $\frac{1}{N}$):

wissen wir bereits

$$\frac{\partial}{\partial \omega_j \partial \omega_k} NLL(\omega) = \sum_{n=1}^N x_{nj} \cdot \frac{\partial}{\partial \omega_k} \mu_n$$

$$= \sum_{n=1}^N x_{nj} \cdot x_{nk} \cdot (1 - \mu_n) \cdot \mu_n$$

$$z_j = (x_{1j}, \dots, x_{Nj})^T$$

$$z_k = (x_{1k}, \dots, x_{Nk})^T$$

$$B = \begin{pmatrix} \mu_1(1-\mu_1) & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \mu_N(1-\mu_N) \end{pmatrix}$$

(im Buch S)

Quadratische Form

$$\Rightarrow H(\omega) = \nabla_{\omega}^2 NLL(\omega) = X^T B X \cdot \left(\frac{1}{N}\right)$$

Da $B = \begin{pmatrix} \mu_1 \cdot (1-\mu_1) & & & \\ & \ddots & & \\ & & \ddots & \mu_N \cdot (1-\mu_N) \\ & & & \end{pmatrix}$ nur positive Einträge hat ($\neq 0$)

können wir $\nabla_{\omega}^2 NLL(\omega)$ schreiben als

$$\begin{aligned} \nabla_{\omega}^2 NLL(\omega) &= X^T B^{\frac{1}{2}} B^{\frac{1}{2}} X \\ &= (B^{\frac{1}{2}} X)^T \cdot (B^{\frac{1}{2}} X) \end{aligned}$$

\Rightarrow positiv semi-definit (PSD) (siehe Buch)

Eine 2-mal differenzierbare Funktion $f: \mathbb{R}^n \rightarrow \mathbb{R}$ ist konvex falls $\nabla^2 f(x)$ positiv semi-definit ist für alle $x \in \mathbb{R}^n$

Einschub: Lösen des Minimierungsproblems ($NLL(\omega)$) mittels "Stochastic Gradient Descent" (SGD)

wir hatten $\nabla_{\omega} NLL(\omega) = \frac{1}{N} \sum_{n=1}^N (\mu_n - y_n) \cdot \vec{x}_n$

Gradient descent: $\omega^{(t+1)} = \omega^{(t)} - \eta_t \cdot \nabla_{\omega} NLL(\omega)$

(\downarrow Lernrate (Schrittweite))

SGD Variante:

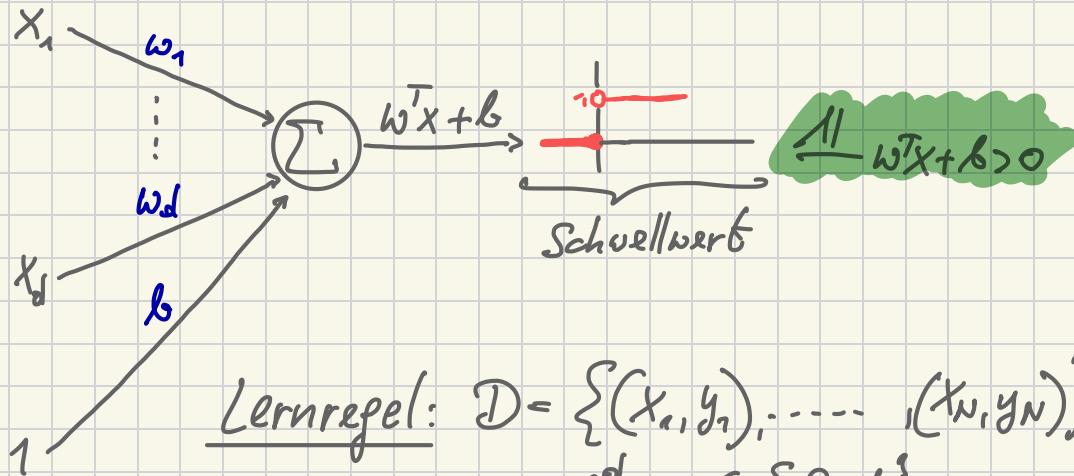
für $t=0, \dots, T$
Schritte

$$\tilde{n} \sim \text{Uniform}(\{1, \dots, N\})$$

$$\omega^{(t+1)} = \omega^{(t)} - \eta_t \cdot (\mu_{\tilde{n}} - y_{\tilde{n}}) \cdot \vec{x}_{\tilde{n}}$$

$$\Theta(w^T x_n + b) \in [0, 1]$$

Einschub: Perception (Rosenblatt)



Lernregel: $\mathcal{D} = \{(x_1, y_1), \dots, (x_N, y_N)\}$,
 $x \in \mathbb{R}^d, y \in \{0, 1\}$

$$\hat{y}_n = \begin{cases} 1, & w^T x_n + b > 0 \\ 0, & \text{sonst} \end{cases}$$

$$\begin{cases} 1, & w^T x + b > 0 \\ 0, & \text{sonst} \end{cases}$$

1. $y_n = 1, \hat{y}_n = 0$
 $w^{(t+1)} = w^{(t)} + x_n$

2. $y_n = 0, \hat{y}_n = 1$
 $w^{(t+1)} = w^{(t)} - x_n$

Solangen bis kein Fehler!

3. $y_n = \hat{y}_n$
 $w^{(t+1)} = w^{(t)}$

Def.: $err_n = (\hat{y}_n - y_n)$

\Rightarrow Allg. Updateregel

$$w^{(t+1)} = w^{(t)} - err_n \cdot x_n$$

$$= w^{(t)} - (\hat{y}_n - y_n) \cdot x_n$$

$\in \{0, 1\}$

$\in \{0, 1\}$

■ Terminiert nur dann, wenn Punkte linear trennbar!

! ABER, Logistische Regression findet hingegen den Maximum-Likelihood Schätzer auch im nicht linear separierbaren Fall!

Logistische Regression - Mehrklassen Fall

#klassen

wir versuchen $p(y|x, \theta)$ zu modellieren (wobei $y \in \{1, \dots, C\}$)

Wie vorher: $x \in \mathbb{R}^D$

$$p(y|x, \{w_i, b_i\}) = \text{Cat}\left(y \mid \text{softmax}\left(\tilde{w}^T x + b\right)\right)$$

$$\tilde{w} \in \mathbb{R}^{D \times C}$$

$$\text{softmax}: \mathbb{R}^C \rightarrow [0, 1]^C$$

$$a \mapsto \text{softmax}(a) = \left[\frac{e^{a_1}}{\sum_{c'} e^{a_{c'}}}, \dots, \frac{e^{a_C}}{\sum_{c'} e^{a_{c'}}} \right]^T$$

$$1. \quad 0 < \text{softmax}(a)_i < 1 \quad \text{für alle } i \in \{1, \dots, C\}$$

$$2. \quad \sum_c \text{softmax}(a)_c = 1$$

Maximum-lik. Schätzung

Zuerst schreiben wir $\text{Cat}(y|\theta)$ leicht um: wir haben $y \in \{1, \dots, C\}$, also ein Integer. Wir können aber auch $y=c'$ folgendermaßen repräsentieren:

$$[0, 0, 0, \dots, 0, 1, 0, \dots, 0] \in \mathbb{R}^C$$

↳ Position c'

// One-hot encoding

Wir schreiben:

$$\text{Cat}(\vec{y} | \theta) = \prod_{c=1}^C \theta_c^{y_c}$$

\Rightarrow

$$\begin{aligned} NLL(w) &= -\frac{1}{N} \log \left(\prod_{n=1}^N \prod_{c=1}^C \alpha_{nc}^{y_{nc}} \right) \quad (\text{ohne } b) \\ &= -\frac{1}{N} \cdot \sum_{n=1}^N \sum_{c=1}^C y_{nc} \cdot \log(\alpha_{nc}) \\ &= \frac{1}{N} \cdot \sum_{n=1}^N H_{CE}(\vec{y}_n, \vec{\alpha}_n) \end{aligned}$$

mit $\alpha_{nc} = \text{softmax}(h(x_n))_c$

\hookrightarrow Cross-Entropy (Kreuzentropie)

Wir wollen $\nabla_w NLL(w) = 0$ nach w lösen. Wir erhalten

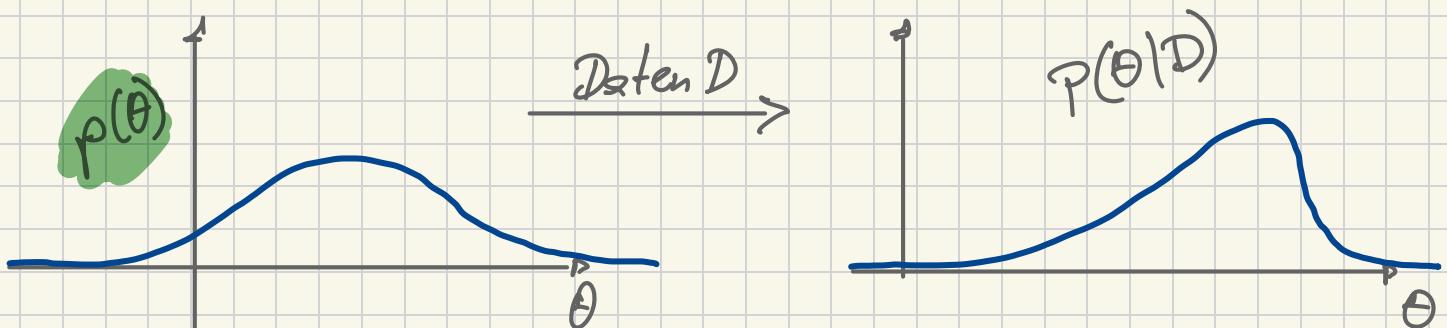
$$\boxed{\nabla_w NLL(w) = \frac{1}{N} \cdot \sum_{n=1}^N \vec{x}_n \cdot (\vec{\alpha}_n - \vec{y}_n)^T}$$

$\hookrightarrow \in \mathbb{R}^{D \times 1}$

$$(\vec{\alpha}_n - \vec{y}_n)^T \in \mathbb{R}^{1 \times C}$$

Wir können nun $\nabla_w NLL(w)$ in SGD benutzen und so unser w updaten!

MAP (maximum a-posteriori) Schätzung



A-priori Verteilung von Parameter Θ

Bislang haben wir versucht Θ so zu schätzen, dass $p(D|\Theta)$ maximiert wird (od. $-\log p(D|\Theta)$ minimiert wird) - also MLE!!!

Sagen wir nun wir hätten Vorwissen, in der Form von $p(\Theta)$ zur Verfügung. Dann könnten wir versuchen Θ anhand von

$$\hat{\Theta} = \underset{\Theta}{\operatorname{argmax}} p(\Theta|D)$$

zu schätzen.

Wir haben $p(\Theta|D) = \frac{p(\Theta) \cdot p(D|\Theta)}{\text{Normalisierung}}$ (Bayes)

$$\Rightarrow \hat{\Theta}_{MAP} = \underset{\Theta}{\operatorname{argmax}} p(\Theta) \cdot p(D|\Theta)$$

Alternativ:

$$\hat{\Theta}_{MAP} = \underset{\Theta}{\operatorname{argmin}} -\log p(\Theta) - \underbrace{\log p(D|\Theta)}_{\text{hatten wir bereits bei MLE}}$$

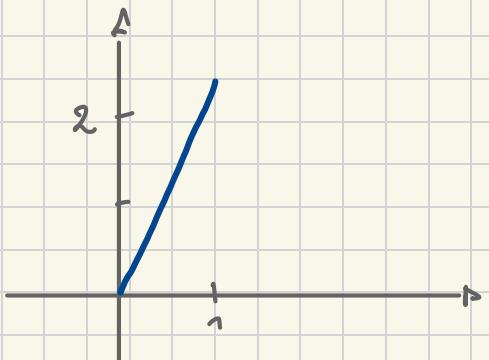
Beispiel (Geometrische Verteilung): $\text{Geom}(\theta)$, $\theta \in (0,1)$

$y \sim \text{Geom}(\theta)$ modelliert die Wahrscheinlichkeit des man genau n Versuche benötigt bis zum ersten Erfolg:

$$P(y=n | \theta) = \theta \cdot (1-\theta)^{y-1}$$

Als Prior könnten wir

$$p(\theta) = \begin{cases} 2\theta, & \text{wenn } 0 < \theta < 1 \\ 0, & \text{sonst} \end{cases}$$



Sagen wir beispielsweise $D = \{y\}, y=8$:

$$\begin{aligned} p(\theta | D) &= p(\theta) \cdot p(D | \theta) \\ &= 2\theta \cdot \theta \cdot (1-\theta)^7 \\ &= 2\theta^2 \cdot (1-\theta)^7 \end{aligned}$$

Wir bestimmen direkt (ohne Lop.) Minima / Maxima:

$$\frac{d}{d\theta} 2\theta^2(1-\theta)^7 = 4\theta \cdot (1-\theta)^6 - 4\theta^2(1-\theta)^5$$

θ -setzen und auflösen nach θ ergibt

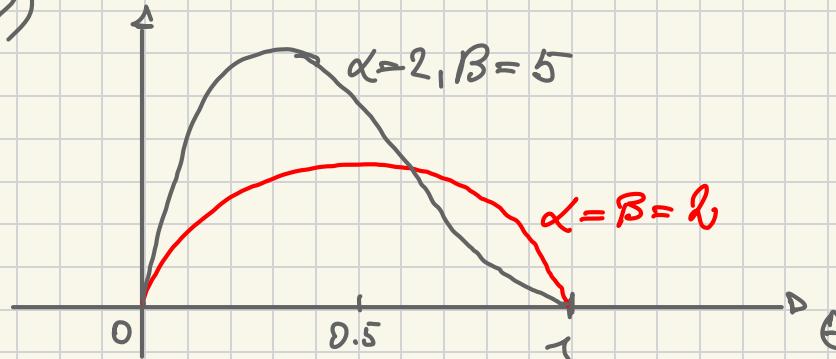
$$\hat{\theta}_{MAP} \in \{0, 1, \frac{1}{2}\}$$

$$\frac{d^2}{d\theta^2} 2\theta^2(1-\theta)^7 = 4 \cdot (6\theta^2 - 6\theta + 1)$$

\rightarrow für $\theta=0$ ist 2-te Abl. positiv } Minima
 für $\theta=1$ ————— positiv }
 für $\theta=\frac{1}{2}$ ————— negativ } Maxima

Beispiel (Künzwurf), $y \sim \text{Ber}(\theta)$

Als Prior nehmen wir eine Beta Verteilung $\text{Beta}(\alpha, \beta)$
 $(p(\theta))$



$$p(\theta) = \frac{1}{B(\alpha, \beta)} \cdot \theta^{\alpha-1} \cdot (1-\theta)^{\beta-1}$$

wobei $B(\alpha, \beta)$ Beta-Funktion
 $\alpha > 0, \beta > 0$.

Gib Daten y_1, \dots, y_N ($y_i \in \{0, 1\}$), sei N_1 die Anzahl an y_i mit $y_i = 1$ und N_0 die Anzahl an y_i mit $y_i = 0$.

Als "Negativer Log-Posterior" $(-\log p(\theta | D))$ erhalten wir

$$-\log p(\theta | D) = -\underbrace{\log p(\theta)}_{\text{Beta Prior}} - \underbrace{\log p(D | \theta)}_{\text{Bernoulli}} + \text{CONST}$$

Einsetzen ergibt:

$$-\log p(\theta | D) = - \left[N_1 \cdot \log(\theta) + N_0 \cdot \log(1-\theta) \right] \quad (\times)$$

$$- \left[(\alpha-1) \cdot \log(\theta) + (\beta-1) \cdot \log(1-\theta) \right] + \text{CONST}$$

Minimieren von (x) bzgl. Θ ergibt:

$$\hat{\Theta}_{MAP} = \frac{N_1 + \alpha - 1}{N_0 + N_1 + \alpha + \beta - 2}$$

Mit $\alpha = \beta = 2$:

$$\Theta_{MAP} = \frac{N_1 + 1}{N_0 + N_1 + 2}$$

"Add-one
smoothing"

Im Kontext der logistischen Regression (2-klassen Fall),
haben wir $w \in \mathbb{R}^d$ als Parameter (ohne Bias b).

Wir nehmen als $p(w)$ folgendes:

$$p(w) = \mathcal{N}(w; 0, s^2 I_d)$$

$$I_d = \begin{pmatrix} 1 & & & & \\ & \ddots & & & \alpha \\ & & \ddots & & \\ & & & \ddots & 1 \\ & & & & 1 \end{pmatrix}, s > 0$$

$d \times d$

$$\mathcal{N}(w; 0, s^2 I_d) = (2\pi)^{-\frac{d}{2}} \cdot \det(s^2 I_d)^{-\frac{1}{2}} \cdot e^{-w^T \begin{pmatrix} s^2 & 0 \\ 0 & s^2 \end{pmatrix}^{-1} w} \cdot \underbrace{\frac{1}{2s^2} w^T w}_{l} = e^{-\frac{1}{2s^2} \|w\|_2^2}$$

Um also \hat{w}_{MAP} zu bestimmen (bei LR), minimieren wir

$$\hat{w}_{MAP} = \underset{w}{\operatorname{argmin}} NLL(w) + \underbrace{\frac{1}{2s^2} \|w\|_2^2}_{=: \lambda > 0}$$

Regularisierungsterm

Lineare Regression (Kap. 11 im Buch) (LinReg)

Ziel: Wir versuchen IR-wertige Outputs vorherzusagen.
($y \in \mathbb{R}$)

1. "Least-Squares" LinReg.

Modell: $P(y | x, \underbrace{\{w_0, w, \sigma^2\}}_{\theta}) = N(y | w^T x + w_0, \sigma^2)$

$x \in \mathbb{R}^D, w \in \mathbb{R}^D, w_0 \in \mathbb{R}$

Ahm.: w_0 können wir in w inkludieren mit

$$w' = [w_0, w_1, \dots, w_d]^T \in \mathbb{R}^{d+1}$$

$$x' = [1, x_1, \dots, x_d]^T \in \mathbb{R}^{d+1}$$

$$NLL(w, \sigma^2) = - \sum_{n=1}^N \log \left[\frac{1}{2\pi\sigma^2} \cdot e^{-\frac{1}{2\sigma^2} \cdot (y_n - w^T x_n)^2} \right] \quad \text{mit}$$

Daten $\mathcal{D} = \{(x_1, y_1), \dots, (x_N, y_N)\}$

$$\Rightarrow NLL(w, \sigma^2) = - \sum_{n=1}^N \frac{1}{2} \cdot \log \left(\frac{1}{2\pi\sigma^2} \right) - \frac{1}{2\sigma^2} \cdot (y_n - \underbrace{w^T x_n}_{\hat{y}_n})^2$$

$$= \frac{N}{2} \cdot \log(2\pi\sigma^2) + \frac{1}{2\sigma^2} \cdot \sum_{n=1}^N (y_n - \hat{y}_n)^2 \quad (\times)$$

Nehmen wir σ^2 fix.

1. ohne additive Konstanten:
(ohne Skalierung)

$$\sum_{n=1}^N (y_n - w^T x_n)^2 = \text{RSS}(w)$$

"Residual sum of squares"

Also

$$RSS(\omega) = \sum_{n=1}^N r_n^2$$

2. ohne additive Konstanten (u. Skalierung mit σ^2), aber mit Faktor $\frac{1}{N}$ multipliziert:

$$MSE(\omega) = \frac{1}{N} \cdot \sum_{n=1}^N r_n^2$$

"Mean-Squared Error"

3. gleich wie 2., aber mit Wurzel $\sqrt{}$

$$RMSE(\omega) = \sqrt{MSE(\omega)}$$

"Root Mean-Squared Error"

Nehmen wir

$$\hat{\omega}_{MLE} = \underset{\omega}{\operatorname{argmin}} RSS(\omega)$$

(ginge auch mit MSE, RMSE, NLL;)

D.h. wir versuchen $\nabla_{\omega} RSS(\omega) = 0$ nach ω zu lösen.

$$RSS(\omega) = \sum_{n=1}^N r_n^2 = \sum_{n=1}^N (y_n - \omega^T x_n)^2$$

$x_n \in \mathbb{R}^d$
 $\omega \in \mathbb{R}^d$

$$X = \begin{pmatrix} x_1 & \dots & x_d \\ \vdots & & \vdots \\ x_m & \dots & x_{Nd} \end{pmatrix}, \quad \omega = \begin{pmatrix} \omega_1 \\ \vdots \\ \omega_d \end{pmatrix}, \quad X\omega = \begin{pmatrix} \langle \omega, x_1 \rangle \\ \vdots \\ \langle \omega, x_N \rangle \end{pmatrix}$$

$y_n \in \mathbb{R}$

$$\bar{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_N \end{pmatrix} \Rightarrow \text{mit } X, \omega \text{ und } \bar{y} \text{ erhalten wir}$$

$$RSS(\omega) = \|X\omega - \bar{y}\|_2^2 = (X\omega - \bar{y})^T (X\omega - \bar{y})$$

$$\begin{aligned}
 RSS(\omega) &= (\bar{X}\omega - \bar{y})^T \cdot (\bar{X}\omega - \bar{y}) \\
 &= (\bar{X}\omega)^T (\bar{X}\omega) - (\bar{X}\omega)^T \bar{y} - \bar{y}^T \bar{X}\omega + \bar{y}^T \bar{y} \\
 &= \underbrace{\omega^T \bar{X}^T \bar{X} \omega}_A - 2\omega^T \bar{X}^T \bar{y} + \bar{y}^T \bar{y}
 \end{aligned}$$

$$\frac{\partial}{\partial x} x^T A x = (A + A^T)x$$

$$\begin{aligned}
 \nabla_{\omega} RSS(\omega) &= (\bar{X}^T \bar{X} + \bar{X} \bar{X}^T) \omega - 2\bar{X}^T \bar{y} \\
 &= \boxed{(2\bar{X}^T \bar{X}) \omega - 2\bar{X}^T \bar{y} = 0 \quad \cdot \frac{1}{2}} \\
 &\quad \bar{X}^T \bar{X} \omega - \bar{X}^T \bar{y} = 0
 \end{aligned}$$

\Rightarrow nach ω aufgelöst:

$$\widehat{\omega}_{MLE} = \underbrace{(\bar{X}^T \bar{X})^{-1}}_{\text{Pseudo-Inverse von } X} \bar{X}^T \bar{y}$$

Anm.: Hat X vollen Rang, dann erhalten wir mit $\widehat{\omega}_{MLE}$ ein eindeutiges globales Minimum von $RSS(\omega)$.

Anm.: MLE für σ^2 ist $\widehat{\sigma}_{MLE}^2 = \frac{1}{N} \cdot \| \bar{X}\widehat{\omega}_{MLE} - \bar{y} \|^2$

(\hookrightarrow zuerst ausrechnen)

(Hesse-Matr. für σ^2 nicht fix)

$$\begin{pmatrix} 1 & \dots & \sigma \\ \vdots & \ddots & \vdots \\ \sigma & \dots & 1 \end{pmatrix}$$

MAP-Schätzer

(mit Prior $p(\omega) = N(\omega; 0, s^2 I_d)$, $s^2 > 0$)

Wir formulieren die Negative-Log-Posterior

$$\frac{1}{2\theta} \cdot (\bar{X}\omega - \bar{y})^T \cdot (\bar{X}\omega - \bar{y}) + \underbrace{\frac{1}{2s^2} \cdot \omega^T \omega}_{\text{Beitrag von Prior } p(\omega)} + \text{CONST.} = J(\omega)$$

$$\frac{1}{2\sigma^2} \cdot \left[w^T X^T X w - y^T X w - \left(X w \right)^T y + y^T y \right] + \frac{1}{2s^2} \overbrace{w^T w}^{||w||^2} + \text{const.} = J(w)$$

$$\begin{aligned} \frac{\partial J(w)}{\partial w} &= -\frac{1}{\sigma^2} \left[-2y^T X + 2w^T X^T X \right] + \frac{1}{s^2} \cdot 2w^T \\ &= \frac{1}{\sigma^2} \cdot \left[w^T X^T X - y^T X \right] + \frac{1}{s^2} w^T \end{aligned}$$

$\Rightarrow 0$ setzen und nach w auflösen:

$$\begin{aligned} w^T \cdot \left[\frac{1}{\sigma^2} X^T X + \frac{1}{s^2} \cdot I_d \right] &= y^T X \frac{1}{\sigma^2} && | \cdot \sigma^2 \\ w^T \cdot \left[X^T X + \frac{\sigma^2}{s^2} \cdot I_d \right] &= y^T X \\ \Rightarrow \widehat{w}_{MAP} &= y^T X \left(X^T X + \frac{\sigma^2}{s^2} \cdot I_d \right)^{-1} \end{aligned}$$

$\sigma^2 = \frac{\sigma^2}{s^2}$

(Regularized Linear Regression)

Wir hatten bereits folgende Modelle:

Klassifizierung

$$\begin{cases} (1) \quad p(y|x,w) = \text{Ber}(y|\sigma(w^T x)) & \text{2-klassen LogReg.} \\ (2) \quad p(y|x,W) = \text{Cat}(y|\text{softmax}(Wx)) & \text{K-klassen LogReg.} \end{cases}$$

$$(3) \quad p(y|x,w) = \mathcal{N}(y|w^T x, \sigma^2) \quad \text{Lin. Regression}$$

Wir könnten $\phi: \mathbb{R}^d \rightarrow \mathbb{R}^e$ per Hand definieren und

$$f(x;\theta) = W\phi(x) + b \quad , \quad \Theta = \{W, b\}$$

statt von Wx in (2) verwenden. Oder wir verwenden

$$f(x;\theta) = W\phi(x;\theta_2) + b \quad \theta_1 = \{W, b\} \quad \theta = \{\theta_1, \theta_2\}$$

D.h. wir parametrisieren die "Feature Transformation" ϕ .

Wir können auch

$$f(x;\theta) = f_L(f_{L-1}(\dots f_1(x)) \dots) \quad \text{mit}$$

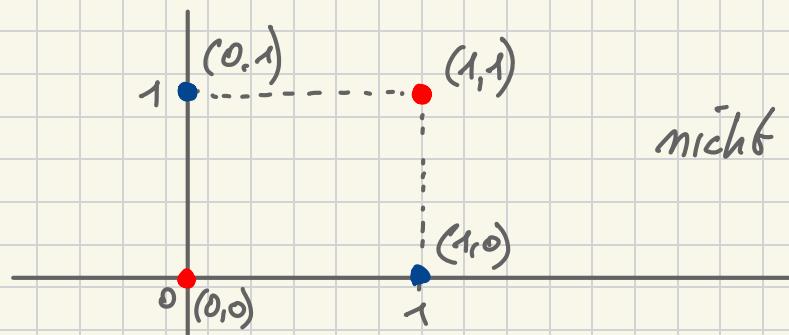
$$f_L(x) = f(x;\theta_L) \quad \text{als "Feature Transformation nutzen."}$$

Einschub (XOR und Perceptron)

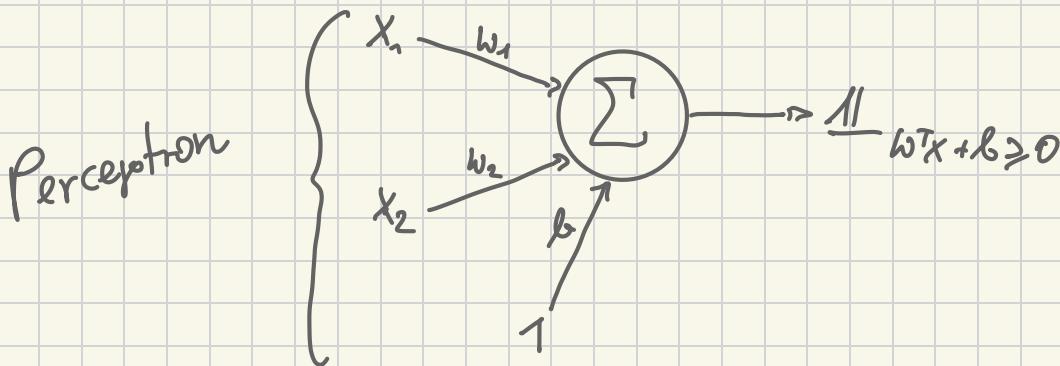
$$f(x; \Theta) = \frac{1}{1 + e^{-\sum w_i x_i - b}} - \begin{cases} 1, & \text{wenn } \sum w_i x_i + b \geq 0 \\ 0, & \text{sonst.} \end{cases}$$

XOR:

x_1	x_2	$x_1 \text{ XOR } x_2$
0	0	0
0	1	1
1	0	1
1	1	0



nicht lin. trennbar!



Aber, mit

$$f(x; \Theta) = \varphi(\sum w_i x_i + b)$$

$$\varphi(x) = \begin{cases} 1, & \text{wenn } x \geq 0 \\ 0, & \text{sonst} \end{cases}$$

Multilayer
Perceptron (MLP)

und $W = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$, $b = \begin{pmatrix} -1.5 \\ -0.5 \end{pmatrix}$, $w = \begin{pmatrix} -1 \\ 1 \end{pmatrix}$, $c = -0.5$ ist die XOR

Funktion sehr wohl realisierbar.

(Ann.: φ wird komponentenweise ausgewertet.)

Was hier eigentlich realisiert wird, ist $(x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix})$

$$\overline{(x_1 \wedge x_2)} \wedge (x_1 \vee x_2)$$

$(\wedge \dots \text{ AND})$
 $(\vee \dots \text{ OR})$.

Bsp.: $x = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$

$$wx = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 2 \\ 2 \end{pmatrix}$$

$$wx + b = \begin{pmatrix} 2 \\ 2 \end{pmatrix} + \begin{pmatrix} -1.5 \\ -0.5 \end{pmatrix} = \begin{pmatrix} 0.5 \\ 1.5 \end{pmatrix}$$

$$\varphi(wx + b) = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

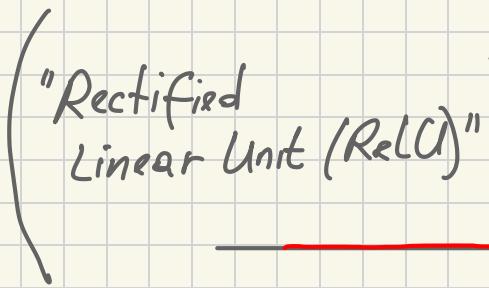
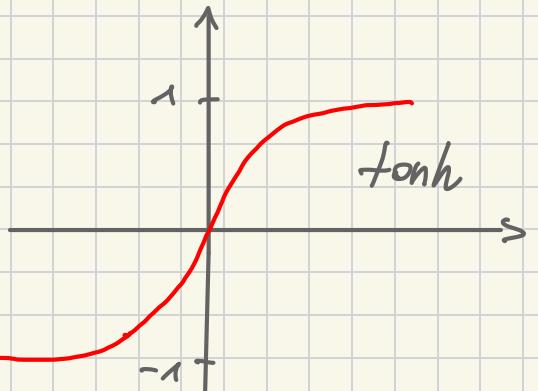
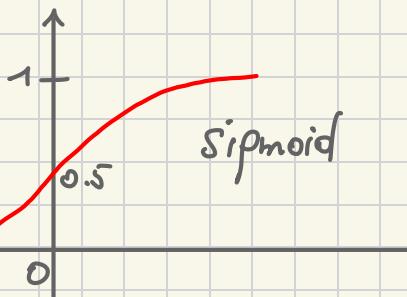
$$w^T \varphi(wx + b) + c = \langle \begin{pmatrix} -1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix} \rangle - 0.5 = -0.5$$

$$\varphi(w^T \varphi(wx + b) + c) = 0 \quad \checkmark$$

In weiterer Folge, ersetzen wir $\varphi = 1$... durch eine differenzierbare Funktion $\varphi: \mathbb{R} \rightarrow \mathbb{R} \Rightarrow$ wir können somit Gradienten ausrechnen (sogar automatisch).

Beispiele für φ :

(auch Aktivierungsfunktionen genannt)



Bsp. (für ein MCP, $x \in \mathbb{R}^2$, $y \in \{0,1\}$)

$$p(y|x, \theta) = \text{Ber}(y | \theta(Q_3)) \text{ mit}$$

$$Q_3 = w^\top z_2 + b_3$$

$$z_2 = \sigma(w_2 z_1 + b_2)$$

$$z_1 = \sigma(w_1 x + b_1)$$

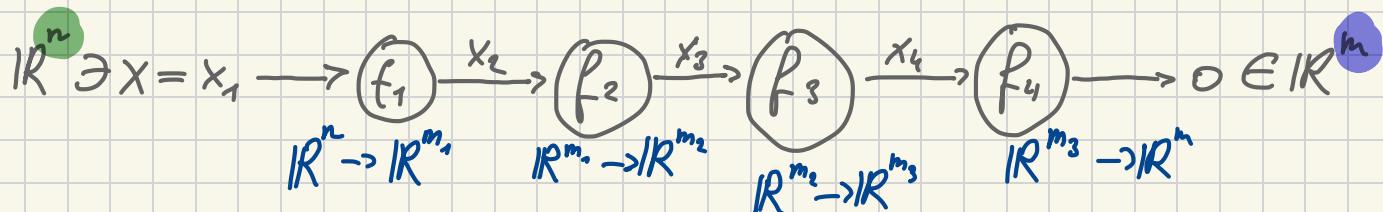
$$(\sigma \equiv \text{Sigmoid})$$

$$\Theta = \{w, b_3, w_2, b_2, w_1, b_1\}$$

Backpropagation (18.3) → im Buch

Input $x \in \mathbb{R}^n$, Output $o \in \mathbb{R}^m$

$$o = f(x) \text{ wobei } f = f_4 \circ f_3 \circ f_2 \circ f_1$$



$$\frac{\partial o}{\partial x} = \frac{\partial o}{\partial x_4} \cdot \frac{\partial x_4}{\partial x_3} \cdot \frac{\partial x_3}{\partial x_2} \cdot \frac{\partial x_2}{\partial x} \quad // \text{Kettenregel}$$

$$\cdot \underbrace{\begin{pmatrix} J_{f_4}(x_4) & J_{f_3}(x_3) & J_{f_2}(x_2) & J_{f_1}(x) \end{pmatrix}}_{\frac{\partial o}{\partial x_4} \in \mathbb{R}^{m \times m_3}} // \text{Produkt von Jacobi-Matrizen}$$

Also ist $\frac{\partial o}{\partial x} \in \mathbb{R}^{m \times n}$; bezeichnet als $J_f(x)$

$$\begin{array}{c}
 \xrightarrow{n} \\
 \left(\begin{array}{c} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \vdots & & \vdots \\ \frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial x_2} & \cdots & \frac{\partial f_m}{\partial x_n} \end{array} \right) \\
 m \downarrow \\
 = \\
 \left(\begin{array}{c} \nabla f_1(x)^T \\ \nabla f_2(x)^T \\ \vdots \\ \nabla f_m(x)^T \end{array} \right)
 \end{array}$$

Partielle Ableitung der 1-ten Komponente
des Outputs von f bzgl. x_n

Anm.: In den Spalten (z.B. i-te Spalte) steht $\frac{\partial f}{\partial x_i}$

Wollen wir z.B. die i-te Zeile von J_f :

$$e_i^T J_f(x) = (0, \dots, 0, 1, 0, \dots) \cdot \left(\begin{array}{c} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{array} \right)$$

e_i^T $1 \times m$ Vektor der Form
 $(0, 0, \dots, \underset{i\text{-te Stelle}}{1}, \dots, 0)$
i-te Stelle

Wollen wir z.B. die j-te Spalte von J_f :

$$J_f(x) \cdot e_j$$

e_j $n \times 1$ Vektor der Form
 $(0, \underset{j\text{-te Stelle}}{1}, 0, \dots, 0)$
j-te Stelle

$e_i^T J_f(x) \dots \text{VJP (Vector Jacobi Product)}$
 $J_f(x) e_j \dots \text{JVP (Jacobi Vector Product)}$

Fall $n < m$: besser $J_f(x)$ anhand $j=1:n$ JVPs zu berechnen.

$$\text{Im Beispiel: } J_f(x) \cdot v = J_{f_4}(x_4) \cdot J_{f_3}(x_3) \cdot J_{f_2}(x_2) \cdot J_{f_1}(x) \cdot v$$

Algorithmus ("Forward-Mode" Automatic Differentiation)

$$x_1 = x$$

$$v_j = e_j \text{ for } j=1:m$$

for $k=1:K$ do

$$x_{k+1} = f_k(x_k)$$

$$v_j = J_{f_k}(x_k) \cdot v_j \text{ for } j=1:m$$

end

$$\text{Return } o = x_{K+1}, \underbrace{\left[J_f(x) \right]_{:,j}}_{j\text{-te Spalte von } J_f \text{ mit } v_j \text{ befüllen}} = v_j \text{ for } j=1:m$$

j-te Spalte von J_f mit v_j befüllen

Algorithmus ("Reverse-Mode" Automatic Differentiation)

$$x_1 = x$$

for $k=1:K$ do

$$x_{k+1} = f_k(x_k)$$

end

$$u_i = e_i \in \mathbb{R}^m \text{ for } i=1:n$$

for $k=K:1$ do

$$u_i^T = u_i^T J_{f_k}(x_k) \text{ for } i=1:n$$

end

$$\text{Return } o = x_{K+1}, \left[J_f(x) \right]_{i,:} = u_i^T \text{ for } i=1:n$$

} Forward pass

} Backward pass

Numerisches Beispiel :

$$f: \mathbb{R}^3 \rightarrow \mathbb{R}^2, \quad x \mapsto f(x) = \begin{pmatrix} x_1^2 + x_2 + x_3 \\ x_1 - x_2 + x_3 \end{pmatrix}$$

Jacobi Matrix ist 2×3 Matrix:

$$\begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \frac{\partial f_1}{\partial x_3} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \frac{\partial f_2}{\partial x_3} \end{pmatrix} = \begin{pmatrix} 2x_1 & 1 & 1 \\ 1 & -1 & 1 \end{pmatrix}$$

Auswerten von J_f an Stelle $x = \begin{pmatrix} 2 \\ 3 \\ 1 \end{pmatrix}$

$$\Rightarrow J_f(x) = \begin{pmatrix} 4 & 1 & 1 \\ 1 & -1 & 1 \end{pmatrix}$$

$$\text{Mit } u = \begin{pmatrix} 1 \\ 0 \end{pmatrix}: \quad u^T J_f(x) = (4 \ 1 \ 1)$$

$$u = \begin{pmatrix} 1 \\ 1 \end{pmatrix}: \quad u^T J_f(x) = (5 \ 0 \ 2)$$

Beispiel mit Parameter

$$\mathcal{L}((x, y), \Theta) = \frac{1}{2} \|y - \underbrace{w_2 \varphi(w_1 x)}_{\text{green}}\|_2^2, \quad \Theta = \{w_1, w_2\}$$

mit Daten $D = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$

$$\mathcal{L} = f_4 \circ f_3 \circ f_2 \circ f_1$$

$$x_2 = f_1(x; w_1) = \underbrace{w_1 x}_{\text{yellow}}$$

$$L = f_4(x_4, y) = \|y - x_4\|_2^2$$

$$x_3 = f_2(x_2; \{ \}) = \underbrace{\varphi(x_2)}_{\text{green}} \quad // \text{keine Parameter}$$

$$x_4 = f_3(x_3; w_2) = \underbrace{w_2 x_3}_{\text{blue}}$$

Seien wir $\Theta_1 = \omega_1$, $\Theta_2 = \omega_2$

$$\frac{\partial L}{\partial \Theta_2} = \frac{\partial L}{\partial x_4} \cdot \boxed{\frac{\partial x_4}{\partial \Theta_2}}$$

$$\frac{\partial L}{\partial \Theta_1} = \underbrace{\frac{\partial L}{\partial x_4}}_{\text{Jacob-Matrizen}} \cdot \underbrace{\frac{\partial x_4}{\partial x_3}}_{\text{Jacob-Matrizen}} \cdot \underbrace{\frac{\partial x_3}{\partial x_2}}_{\text{Jacob-Matrizen}} \cdot \boxed{\frac{\partial x_2}{\partial \Theta_1}}$$

Algorithmus (für MLP mit K Schichten/Loyer und Skalar Output)
(wie im Beispiel)

$x_1 = x$
 for $k=1:K$ do
 $x_{k+1} = f_k(x_k)$
 end } Forward pass

$u_{K+1} = 1$ // da $m=1$, Skalar Output

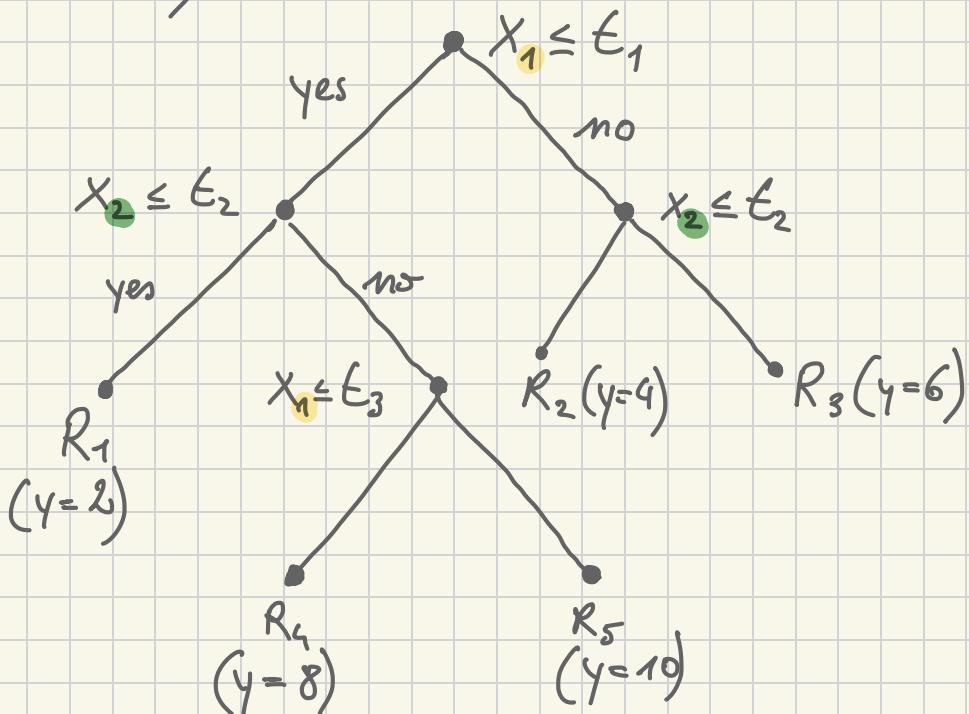
for $k=K:1$ do
 $f_k = u_{k+1}^T \cdot \frac{\partial f_k(x_k; \Theta_k)}{\partial \Theta_k}$ // neu
 $u_k^T = u_{k+1}^T \cdot \frac{\partial f_k(x_k; \Theta_k)}{\partial x_k}$
 end

Return x_{K+1} , $\nabla_x L = u_1$, $\left\{ \nabla_{\Theta_k} L = f_k \text{ für } k=1:K \right\}$

Trees, Forests (& Boosting)

Grundidee: "Classification and Regression Trees (CART)"
partitionieren den Input Raum rekursiv.

Bsp.: $X \in \mathbb{R}^2$, $X = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$
(Illustration)



Z.B.: $R_1 = \left\{ X \in \mathbb{R}^2 : x_1 \leq t_1 \wedge x_2 \leq t_2 \right\}$

Schwellwerte

Formal hören wir:

$$f(x; \Theta) = \sum_{j=1}^J w_j \cdot \mathbb{1}_{x \in R_j}$$

Regionen an den "leaf nodes" (Blattknoten)

Θ ... Parameter

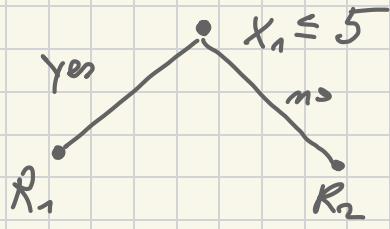
$$\Theta = \left\{ (\omega_j, R_j) : j = 1, \dots, J \right\}$$

$$w_j = \frac{\sum_{n=1}^N y_n \cdot \mathbb{1}_{x_n \in R_j}}{\sum_{n=1}^N \mathbb{1}_{x_n \in R_j}}$$

Vorhergesagter Wert in
 R_j

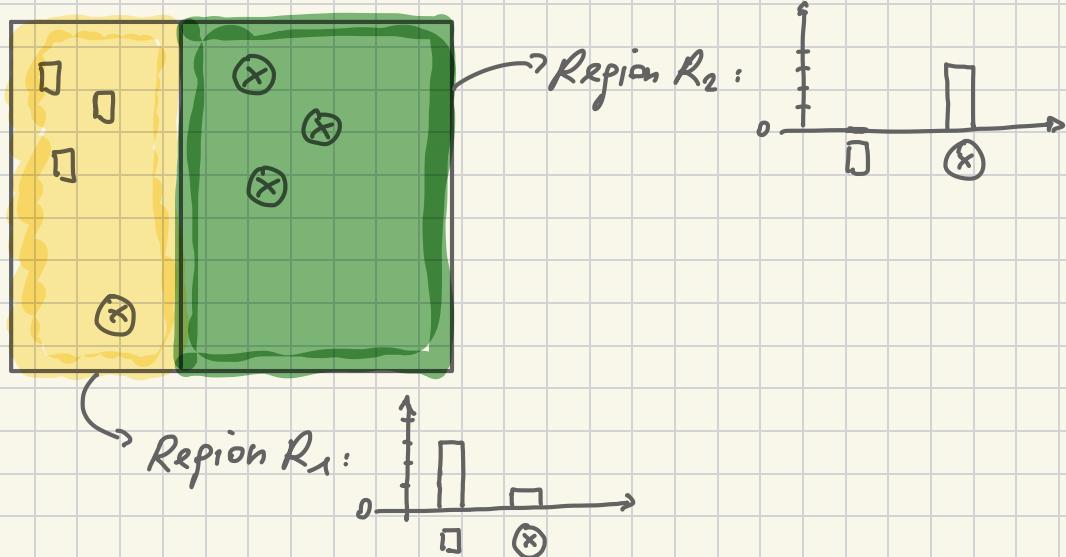
Bsp.:

$$\text{Datensatz } D = \{(1, 2, 6), x_1 \\ (2, 6, 4), x_2 \\ (4, 10, 6)\}, x_3$$



$$w_1 = \frac{6 \cdot \mathbb{1}_{x_1 \in R_1} + 4 \cdot \mathbb{1}_{x_2 \in R_1} + 6 \cdot \mathbb{1}_{x_3 \in R_1}}{\mathbb{1}_{x_1 \in R_1} + \mathbb{1}_{x_2 \in R_1} + \mathbb{1}_{x_3 \in R_1}} = \frac{12}{3} = 6$$

Wichtig: In Klassifikationsproblemen beinhalten die Blatt-Knoten eine Verteilung über die Klassen-Labels (onstall eines Mittels über die Targets (y)).



Wie finden wir θ anhand der Daten?



$$L(\theta) = \sum_{n=1}^N l(g_n, f(x_n; \theta))$$

→ Loss Funktion

$$= \sum_{j=1}^J \left[\begin{array}{l} l(g_n, \omega_j) \\ x_n \in R_j \end{array} \right]$$

$\left. \begin{array}{l} \text{nicht differenzierbar} \\ \text{Loss Funktion} \end{array} \right\}$

(Anm.: Finden der optimalen Partitionierung des Input Raums ist
NP-vollständig)

Ansatz: Greedy - Strategie

↳ also Baum Knoten für Knoten aufbauen!

Nehmen wir Knoten i und sei

$$\mathbf{x}_n = [x_{n1}, \dots, x_{nD}]^T$$

$$D_i = \{(x_n, y_n) \in N_i\},$$

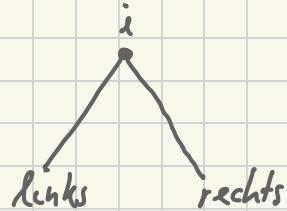
also die Menge an Daten-Tupel die an Knoten i ankommen.

Fall 1: Merkmal j ist reellwertig. z.B. $j=1$ mit Werten $\{4.5, -12, 72, -12\}$ (als Multimenge zu verstehen).

Sortiert $T_1 = (-12, 4.5, 72)$! Dies sind die möglichen Schwellwerte um zu splitten.

$$D_i^{\text{links}}(j, t) = \{(x_n, y_n) \in N_i, x_{nj} \leq t\}$$

$$D_i^{\text{rechts}}(j, t) = \{(x_n, y_n) \in N_i, x_{nj} > t\}$$



Fall 2: Merkmal j ist **kategorisch**. z.B. $j=1$ mit K_1 möglichen Ausprägungen $\rightarrow K_j$ mögliche Splits

$$D_i^{\text{links}}(j, t) = \left\{ (x_n, y_n) \in N_i : x_{nj} = t \right\}$$

$$D_i^{\text{rechts}}(j, t) = \left\{ (x_n, y_n) \in N_i : x_{nj} \neq t \right\}$$

Haben wir jetzt $D_i^{\text{links}}(j, t)$ und $D_i^{\text{rechts}}(j, t)$ für alle möglichen j 's und t 's bestimmt (am Knoten i), wählen wir "das Beste" j und t wie folgt:

$$(j_i, t_i) = \underset{j \in \{1, \dots, d\}}{\operatorname{arg\,min}} \left[\frac{|D_i^{\text{links}}(j, t)|}{|D_i|} \cdot \text{COST}(D_i^{\text{links}}(j, t)) + \frac{|D_i^{\text{rechts}}(j, t)|}{|D_i|} \cdot \text{COST}(D_i^{\text{rechts}}(j, t)) \right]$$

Wie wählt man $\text{COST}(\cdot)$?

Fall 1 (Regression):

$$\text{COST}(D_i) = \sum_{n: (x_n, y_n) \in D_i} (y_n - \bar{y})^2 \quad \text{mit} \quad \bar{y} = \frac{1}{|D_i|} \sum_{n: (x_n, y_n) \in D_i} y_n$$

Fall 2 (Klassifikation):

$$(i) \quad \frac{1}{|D_i|} \cdot \sum_{n: (x_n, y_n) \in D_i} \underbrace{\frac{1}{C} y_n}_{\text{alle Ausprägungen der kategorischen Variable}} = c \quad \text{für alle } c \in \{1, \dots, C\}$$

$$\left(\begin{array}{c} \frac{1}{|D_i|} \cdot \sum_{n: (x_n, y_n) \in D_i} \frac{1}{y_n=1} \\ \vdots \\ \frac{1}{|D_i|} \cdot \sum_{n: (x_n, y_n) \in D_i} \frac{1}{y_n=c} \end{array} \right) = \left(\begin{array}{c} \hat{\pi}_{i,1} \\ \vdots \\ \hat{\pi}_{i,c} \end{array} \right),$$

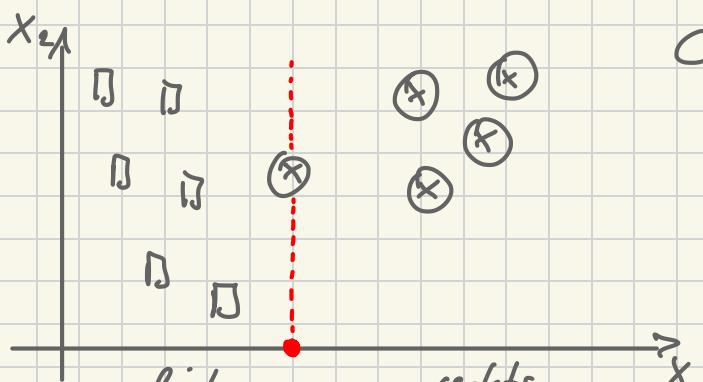
(ii) Gini-Index G_i

$$G_i = \sum_{c=1}^C \hat{\pi}_{ic} \cdot (1 - \hat{\pi}_{ic})$$

$$= 1 - \sum_{c=1}^C (\hat{\pi}_{ic})^2$$

$$\text{Cost}(D_i) = G_i$$

Bsp. (Gini Index)



(Split entlang x_1 -Achse)

$C = 2$ ⚡ ... Klasse 0

⊗ ... Klasse 1

Links: $\hat{\pi}_{i,0} = \frac{1}{7} \cdot 6, \hat{\pi}_{i,1} = \frac{1}{7} \cdot 1$
 $\Rightarrow \frac{6}{7} \cdot (1 - \frac{6}{7}) + \frac{1}{7} \cdot (1 - \frac{1}{7}) \approx 0.24$

Rechts: 0

BOOSTING (Ensemble Methode)

Ensembles von Bäumen sind im Allg. Modelle der Form

$$(x) \quad f(x; \Theta) = \sum_{m=1}^M \beta_m \cdot F(x; \Theta_m) \quad M \dots \text{Ensemble Größe}$$

Boosting ist eine Methode um ein solches Modell rezipentiel aufzubauen, wobei F binäre Klassifizierer (also mit Output $\{-1\}$) sind.

Grundidee: Wir "fitten" zuerst $F(x; \Theta_1)$ auf den originalen Trainingsdaten und gewichten dann die Daten um, sodass jene von $F(x; \Theta_1)$ als falsch klassifizierte Instanzen MEHR Gewicht erhalten. Dann "fitten" wir $F(x; \Theta_2)$ und gewichten wieder um, etc.

Passiert i.A. M -mal ($M > 0$) wobei M frei wählbar ist ($M \dots \# \text{Boosting Runden}$).

Man kann zeigen: hat jedes $F(\cdot; \Theta_i)$ einen Fehler $< \frac{1}{2}$, dann hat das Ensemble (x) höhere Genauigkeit als alle seiner Komponenten (im Kontext von ADABOOST).

Im Allgemeinen:

$$L(f) = \sum_{n=1}^N \ell(y_n, f(x_n; \theta))$$

differenzierbar
wobei

$$f(x; \theta) = \sum_{m=1}^M \beta_m \cdot F(x; \theta_m)$$

z.B. einfache Decision Trees
(von vorher)

Algorithmus: $f_0(x) = 0$

(Template)

for $m=1 \dots M$

$$(\beta_m, \theta_m) = \underset{\beta, \theta}{\operatorname{arg\,min}} \sum_{n=1}^N \ell(y_n, f_{m-1}(x_n) + \beta \cdot F(x_n; \theta_m))$$

$$f_m(x) = f_{m-1}(x) + \beta_m \cdot F(x; \theta_m)$$

end

(1. Termin : 6. Feb (Do) 0:00 - 12:00)
 (2. Termin : 28. Feb (Fr) 10:00 - 13:00)