ML Konzepte: Teil 3 (Unbearbeitete Version)

Hayrettin Acar

December 14, 2023

1 Gauß'sche Diskriminanzanalyse (GDA)

Die bisherigen Ansätze der vorgestellten Konzepte werden als **diskriminative** Algorithmen bezeichnet. Mit ihnen soll anhand von Eingabedaten (x) das Wahrscheinlichkeitsmodell $p(y|x;\theta)$ erlernt werden. Bei einem Klassifikationsproblem soll beispielsweise anhand der Eingabewerte bestimmt, werden wie wahrscheinlich es ist, dass ein Testpunkt zu einer Klasse zugehörig ist.Im Gegensatz hierzu versuchen **generative** Algorithmen die Wahrscheinlichkeitsverteilungen der Merkmale zu erlernen, $p(x|y;\theta)$. Mithilfe des *Theorems von Bayes* kann dann eine Schlussfolgerung auf $p(y|x;\theta)$ erfolgen.

Die Gauß'sche Diskriminanzanalyse (GDA) ist ein Algorithmus, der zur Klassifizierung und Dimensionalitätsreduzierung verwendet wird. Er ist eng verwandt mit der gebräuchlicheren linearen Diskriminanzanalyse (LDA) und basiert auf den folgenden Grundannahmen der zugrunde liegenden Daten aus:

- Gauß'sche Verteilung: Die GDA geht davon aus, dass die Datenpunkte innerhalb jeder Klasse aus einer multivariaten Gauß'schen Normal-Verteilung stammen.
- Gleichheit der Klassenkovarianz: In ihrer einfachsten Form geht die GDA davon aus, dass alle Klassen die gleiche Kovarianzmatrix haben. Eine verallgemeinerte Version der GDA, die so genannte lineare Diskriminanzanalyse (LDA), lockert diese Annahme jedoch und lässt unterschiedliche Kovarianzmatrizen für jede Klasse zu. Die LDA wird häufig bevorzugt, wenn die Kovarianzmatrizen zwischen den Klassen signifikant unterschiedlich sind.
- Unabhängigkeit der Merkmale: Die GDA geht davon aus, dass die (reellwertigen) Merkmale innerhalb jeder Klasse statistisch unabhängig sind.

1.1 Binäres GDA Modell

Zur binären Klassifizierung werden folgende Grundannahmen getroffen:

$$\begin{aligned} y|x;\theta &\sim Bernoulli(\phi) \\ x|y=1 &\sim \mathcal{N}(\mu_1, \Sigma) \\ x|y=0 &\sim \mathcal{N}(\mu_0, \Sigma) \end{aligned}$$

Die normalverteilten Merkmale werden parametrisiert mithilfe des **Erwartungs-Vektors** $\mu \in \mathbb{R}^2$ und der **Kovarianz-Matrix** $\Sigma \in \mathbb{R}^{2\times 2}$. Die Wahrscheinlichkeiten für einen Trainingsdatensatz können bestimmt werden zu

$$p(x;y=1) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu_1)^T\Sigma^{-1}(x-\mu_1)\right) \tag{1}$$

$$p(x;y=0) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu_0)^T\Sigma^{-1}(x-\mu_0)\right) \tag{2}$$

$$p(y) = \phi^y(1-\phi)^{1-y}$$

Dabei beschreibt $|\Sigma|$ die Determinante der Kovarianzmatrix, welche für beide Fälle identisch ist. Weiterhin beschreibt ϕ die Bernoulli-Wahrscheinlichkeit.

Da die GDA ein generativer Algorithmus ist, muss die Bestimmung einer Optimierungsfunktion anhand der kombinierten Wahrscheinlichkeit $l(\phi, \mu_0, \mu_1, \Sigma)$ angesetzt werden. Mit der Annahme unabhängiger Wahrscheinlichkeiten findet man für die Likelihood-Funktion im Allgemeinen, dass

$$l(\phi, \mu_0, \mu_1, \Sigma) = \log \prod_{i=1}^{m} p(x^{(i)}, y^{(i)}; \phi, \mu_0, \mu_1, \Sigma) \tag{3}$$

$$= \log \prod_{i=1}^{m} p(x^{(i)}|y^{(i)}; \phi, \mu_0, \mu_1, \Sigma) p(y^{(i)}; \phi) \tag{4}$$

$$= \log \prod_{i=1}^{m} \left(\prod_{j=1}^{d} p(x^{(i)}|y^{(i)}; \phi, \mu_0, \mu_1, \Sigma) \right) p(y^{(i)}; \phi) \tag{5}$$

$$= \sum_{i=1}^{m} \left(\log p(y^{(i)}; \phi) + \sum_{i=1}^{d} \log p(x_j^{(i)}|y^{(i)}; \phi, \mu_0, \mu_1, \Sigma) \right) \tag{6}$$

wobei $j = 1, ...d$ die verschiedenen Merkmale bezeichnet. Mithilfe der Maximum Likelihood können die folgenden Parameter bestimmt werden zu

$$\phi = \frac{1}{m} \sum_{i=1}^{m} 1\{y^{(i)} = 1\} \tag{7}$$

$$\mu_0 = \frac{\sum_{i=1}^{m} 1\{y^{(i)} = 0\} x^{(i)}}{\sum_{i=1}^{m} 1\{y^{(i)} = 0\}} \tag{8}$$

$$\mu_1 = \frac{\sum_{i=1}^{m} 1\{y^{(i)} = 1\} x^{(i)}}{\sum_{i=1}^{m} 1\{y^{(i)} = 1\}} \tag{9}$$

$$\Sigma = \frac{1}{m} \sum_{i=1}^{m} (x^{(i)} - \mu_{y^{(i)}})(x^{(i)} - \mu_{y^{(i)}})^T \tag{10}$$

Die Erwartungsvektoren werden dabei intuitiver Weise als gemittelter Durchschnitt der Merkmale für jede Klasse gebildet.
Es kann gezeigt werden, dass für einer multivariaten Gauß'schen Normal-Verteilung $p(x|y)$ die entsprechende logistische Funktion $p(y = 1|x; \phi, \mu_0, \mu_1)$ gefunden werden kann zu

$$p(y = 1|x; \phi, \Sigma, \mu_0, \mu_1) = \frac{1}{1 + \exp(-\theta^T x)}$$

mit $\theta = \theta(\phi, \Sigma, \mu_0, \mu_1)$. Umgekehrt gilt dies jedoch nicht. Nicht jede logistische Funktion impliziert, dass die Merkmale dieser Normalverteilung entsprechen. Es werden also für die Gauß-Analyse strengere Annahmen getroffen als für die logistische Regression. Gelten diese Annahmen jedoch, ist GDA besonders effizienter Algorithmus im Vergleich zur logistischen Regression.

1.2 Beispielbetrachtung

Betrachtet sie die bereits bekannte Verteilung zweier Klassen in (1). Mithilfe der berechneten Parameter $\phi, \Sigma, \mu_0, \mu_1$ lässt sich 2D-Gaussverteilung bestimmen, welche aufgrund der gleichen Kovarianzmatrix identisch für beide Klassen aussieht. Lediglich die unterschiedlichen Erwartungsvektoren μ_0, μ_1 führen zu einer räumlichen Verschiebung beider Kurven.

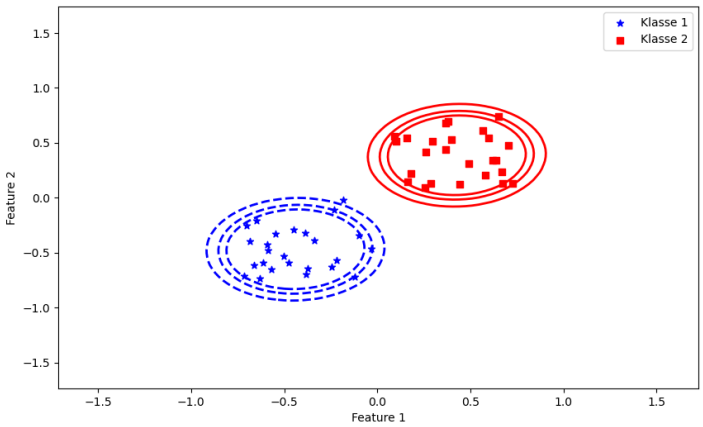

Figure 1: Die Verteilung von 2 verschiedenen Klassen in Abhängigkeit von zwei Eingabemerkmalen mit GDA Konturen.

Für jeden Datenpunkt können nun die Wahrscheinlichkeiten $P(x|y)$ und $P(y)$ bestimmt werden. Für einen Testpunkt $[-0.2, -0.4]$ kann dann mithilfe des Satzes von Bayes bestimmt werden, mit welcher Wahrscheinlichkeit dieser zur jeweiligen Klasse gehört. Die Wahrscheinlichkeiten können für den genannten Testpunkt bestimmt werden zu

$$P(y|x) = \begin{cases} 0.999989.. & \text{, Klasse 1} \\ 0.0000101.. & \text{, Klasse 2} \end{cases}$$

2 Naive Bayes

Die Naive Bayes Methode geht von diskreten Merkmalswerten aus und trifft vereinfachende Annahmen hinsichtlich der Wahrscheinlichkeiten. Sie wird häufig im Umfeld der Textklassifikation (z.B. E-Mail Spam-Filter) verwendet, welche hier beispielhaft beschrieben werden soll.

Die Merkmale werden zunächst mithilfe einer Abbildung auf ein Wörterbuch mit d Wörtern kodiert.

$$x = \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \begin{bmatrix} a \\ Auftrag \\ Arbeit \\ \vdots \\ Zaun \end{bmatrix}$$

Die Wahrscheinlichkeitsmodellierung ist zunächst gegeben durch

$$p(x_1, ...x_d|y) = p(x_1|y)p(x_2|y, x_1)p(x_2|y, x_1, x_2)..p(x_d|y, x_1....x_{d-1}) \tag{11}$$

Der Naive Bayes Algorithmus stellt nun die starke Annahme, dass die Wahrscheinlichkeiten für jedes Merkmal und einem Label unabhängig sind, d.h.

$$p(x_1, ...x_d|y) = p(x_1|y)p(x_2|y) = \prod_{j=1}^{d} p(x_j|y) \tag{12}$$

Im Zusammenhang der Spam-Klassifikation würde es bedeuteten, dass die Klassifikation in Spam (y=1) und dem auftauchenden Wort "Auftrag" keine Aussagekraft über die Wahrscheinlichkeit von anderen Wörtern z.B. "Arbeit" in einer Spam E-Mail hat.
Das Naive Bayes Modell wird parametriert durch

$$\begin{aligned} \phi_{j|y=1} &= p(x_j = 1|y = 1) \\ \phi_{j|y=0} &= p(x_j = 1|y = 0) \\ \phi_y &= p(y = 1) \end{aligned}$$

Angewendet auf das Beispiel des Spam-Filters wäre beispielsweise $\phi_{j|y=1}$ die Wahrscheinlichkeit für das Auftauchen des Wortes x_j, falls die E-Mail als Spam klassifiziert wurde. Mit ϕ_y ist die apriori-Wahrscheinlichkeit bezeichnet, dass eine einkommende E-Mail als Spam klassifiziert wird.
Zur Optimierung der Parameter wird, ähnlich zur bereits vorgestellten GDA, die kombinierte Wahrscheinlichkeit betrachtet

$$L(\phi_y, \phi_{j|y=0}, \phi_{j|y=1}) = \prod_{i=1}^{m} p(x^{(i)}, y^{(i)})$$

Man findet schnell, dass die maximum-likelihood Schätzung ergibt

$$\phi_y = \frac{\sum_{i=1}^{m} 1\{y(i) = 1\}}{m} \tag{13}$$

$$\phi_{j|y=0} = \frac{\sum_{i=1}^{m} 1\{x_j^{(i)} = 1 \wedge y^{(i)} = 0\}}{\sum_{i=1}^{m} 1\{y^{(i)} = 0\}} \tag{14}$$

$$\phi_{j|y=1} = \frac{\sum_{i=1}^{m} 1\{x_j^{(i)} = 1 \wedge y^{(i)} = 1\}}{\sum_{i=1}^{m} 1\{y^{(i)} = 1\}} \tag{15}$$

Die Interpretation der Parameter ist für das Beispiel der Spam-Klassifizierung sehr intuitiv. So berechnet sich beispielsweise $\phi_{j|y=1}$ durch die Anteil aller Spam-Mails, die das Wort x_j enthalten (Zähler), geteilt durch die Gesamtanzahl aller Spam E-Mails (Nenner).

Um eine Vorhersage für einen Testdatensatz zu treffen werden nun wieder mithilfe der Bayes Regel die Wahrscheinlichkeiten für beide Fälle berechnet,

$$p(y = 1|x) = \frac{\left(\prod_{j=1}^{d} p(x_j|y = 1)\right) p(y = 1)}{\left(\prod_{j=1}^{d} p(x_j|y = 1)\right) p(y = 1) + \left(\prod_{j=1}^{d} p(x_j|y = 0)\right) p(y = 0)}$$

Der Zähler beschreibt hierbei die Gesamtwahrscheinlichkeit aller auftretenden Wörter mit der Bedingung, dass ein eine Spam E-Mail vorliegt, multipliziert mit der Gesamtwahrscheinlichkeit, dass es sich um eine Spam handelt. Normiert wird dies im Nenner durch die Summe der Wahrscheinlichkeiten beider Fälle. Analog ist auch die Wahrscheinlichkeit $p(y = 0|x)$ zu berechnen. Die E-Mail würde dann anhand der höheren beider Wahrscheinlichkeiten entsprechend klassifiziert werden.
Ein Problem des Algorithmus ist die Behandlung von erstmaligen auftauchenden Wörtern, dessen Wahrscheinlichkeiten für beide Fälle mit 0 angeben werden und zu einer Division $\frac{0}{0}$ führen. Diesem Problem wird häufig mit dem **Laplace-Smoothing** Verfahren begegnet, welches die Wahrscheinlichkeitswerte in einen praktibaleren Wertebereich transformiert,

$$\phi_{j|y=0} = \frac{1 + \sum_{i=1}^{m} 1\{x_j^{(i)} = 1 \wedge y^{(i)} = 0\}}{2 + \sum_{i=1}^{m} 1\{y^{(i)} = 0\}} \tag{16}$$

$$\phi_{j|y=1} = \frac{1 + \sum_{i=1}^{m} 1\{x_j^{(i)} = 1 \wedge y^{(i)} = 1\}}{2 + \sum_{i=1}^{m} 1\{y^{(i)} = 1\}} \tag{17}$$

3 Support Vector Machines (SVM)

Der SVM Algorithmus gehört aufgrund seiner Robustheit und Effizienz zu den populärsten Algorthmen für Klassifikations- und Regressionsaufgaben. Er wird in verschiedenen Anwendungen wie beispielsweise Textklassifikation, Bilderkennung und Bioinformatik eingesetzt.

3.1 Optimale Margin Klassifikation

Betrachtet sei ein binäres Klassifizierungsproblem mit $y \in \{1, -1\}$, das vollständig linear separierbar ist, siehe (2). Die lineare Entscheidungsgrenze kann beschrieben werden durch

$$h_{w,b} = g(w^T x + b)$$

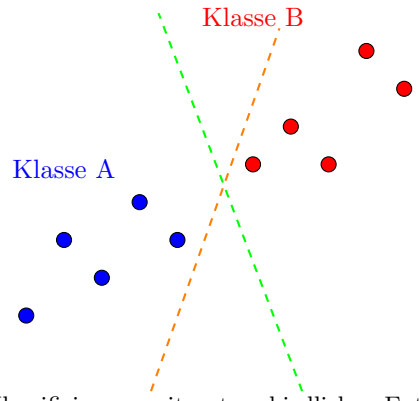

Figure 2: Binäres Klassifizierung mit unterschiedlichen Entscheidungsgeraden.

mit $g(z) = 1, z \geqslant 0$. Üblicherweise wird anstatt des Parametervektors θ nun die Notation mit w, b verwendet, um den additiven Term b gesondert behandeln zu können.

Grundsätzlich kann die Entscheidungsgerade unterschiedlich aussehen. Das Ziel der SVM besteht nun darin, die "optimale" Gerade zu finden (für höherdimenionsale Probleme handelt es sich um eine Entscheidungs-Hyperebene). Abhängig ist die Lage der Geraden von den Punkten, die am nächsten an der Entscheidungsgerade liegen (Support Vektoren). Diese sind auch am schwierigsten zu klassifizieren.

Die **funktionale** Spanne für einen Trainingsdatensatz kann nun dargestellt werden mit

$$\hat{\gamma}^{(i)} = y^{(i)}(w^T x^{(i)} + b). \tag{18}$$

Es gilt für beide Klassen, dass eine Vorhersage für den Fall $y^{(i)}(w^T x^{(i)} + b) > 0$ korrekt ist. Gleichzeitig ist auch die Zuversicht eine korrekte Vorhersage getroffen zu haben größer, falls $y^{(i)}(w^T x^{(i)} + b) \gg 0$. Problematisch ist allerdings, dass die Skalierung der Parameter w, b willkürlich ist und somit eine hohe Zuversicht mit entsprechender Skalierung immer künstlich herbeiführbar ist.

Die **geometrische** Spanne bezeichnet die räumliche Distanz eines Datenpunktes von der Grenze, (3). Mit dem Einheitsvektor $\vec{w} = \frac{w}{||w||}$ kann die geometrische Distanz bestimmt werden zu

$$\gamma^{(i)} = y^{(i)} \left(\left(\frac{w}{||w||} \right)^T x^{(i)} + \frac{b}{||w||} \right)$$

Für $||w||_2 = 1$ entspricht die funktionale Spanne der geometrischen Spanne. Weiterhin definieren wir die (niedrigste) geometrische Spanne eines gesamten Trainingsdatensatzes als

$$\gamma = \min_{i=0,\ldots,m} \gamma^{(i)}$$

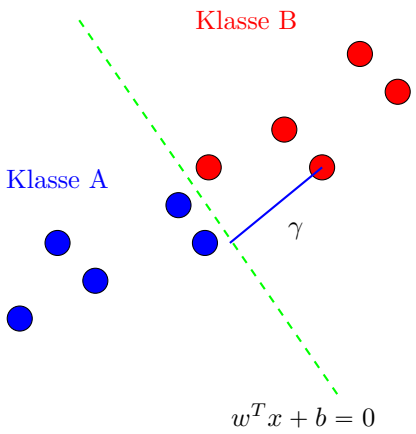

Figure 3: Geometrische Spanne γ.

Das Ziel ist nun die Maximierung von $\max_{\gamma,w,b} \gamma$ mit den Nebenbedingungen

$$y^{(i)}(w^T x^{(i)} + b) \geqslant \gamma, i = 1, ..., m$$

$$||w|| = 1$$

zu finden. Mit $||w|| = 1$ wird garantiert, dass die funktionale Spanne der geometrischen Spanne entspricht und somit das Ergebnis des Klassfikators unmittelbar interpretierbar wird. Allerdings erschwert diese Bedingung auch gleichzeitig die Optimierung, da sie einen nicht-konvexen Charakter einbringt. Mit

$$\gamma = \frac{\hat{\gamma}}{||w||}$$

kann alternativ eine Maximierung von

$$\max_{\hat{\gamma},w,b} = \frac{\hat{\gamma}}{||w||}$$

gesucht werden. Der Vorteil hierbei ist nun, dass die Parameter w, b frei wählbar sind, da diese keinen Einfluss auf die Lage und Ausrichtung der Entscheidungsgrenze haben. Damit kann die Bedingung

$$\hat{\gamma} = 1$$

definiert werden. Weiterhin kann das Maximierungsproblem reziprok zu einem Minimierungsproblem formuliert werden zu

$$\min_{w,b} \frac{1}{2}||w||^2$$

mit der Nebenbedingung $y^{(i)}(w^T x^{(i)} + b) \geqslant 1, i = 1, ..., m$. Dies ist ein konvexes Optimierungsproblem mit einer linearen Nebenbedingung und kann mithilfe von **quadratischer Optimierung** gelöst werden.

3.2 Lagrange Dualität

Allgemein bietet eine SVM in ihrer regularisierten Form die Lösung für die folgende Optimierungsfunktion

$$\min_{w \in \mathbb{R}^d, b \in \mathbb{R}} \frac{1}{2}||w||^2 + \frac{C}{m} \sum_{i=1}^{m} \max(0, 1 - y^{(i)}(w^T x^{(i)} + b)).$$

Hierbei wird die sog. **Hinge-Verlustfunktion** verwendet, $\max(0, 1 - f(x)y))$. Allerdings ist diese Funktion nicht differenzierbar und damit schwierig zu optimieren. Es stellt sich heraus, dass die Formulierung des Optimierungsproblems mithilfe der **Lagrange Dualität** neue Möglichkeiten eröffnet. Zunächst sei das allgemeine Optimierungsproblem definiert als

$$\min_{x \in \mathbb{R}^n} f(x)$$

mit den Nebenbedingungen

$$g_i(x) \leqslant 0, i = 1..., k$$
$$h_i(x) = 0, i = 1..., p$$

wobei gilt $x \in \mathbb{R}^n, f : \mathbb{R}^n \mapsto \mathbb{R}$, die Funktionen $g_i : \mathbb{R}^n \mapsto \mathbb{R}$ sind differenzierbar und konvex und die Funktionen $h_i : \mathbb{R}^n \mapsto \mathbb{R}$ sind affin. Weiterhin sei der optimale Wert p^* definiert als

$$p^* = \inf\{f(x)| \text{ x erfüllt alle Nebenbedingungen (feasible point)}\}$$

Nun kann die Lagrange Funktion $\mathcal{L} : \mathbb{R}^n \times \mathbb{R}^k \times \mathbb{R}^p \longrightarrow \mathbb{R}$ definiert werden zu

$$\mathcal{L}(x, \alpha, \beta) = f(x) + \sum_{i=1}^{k} \alpha_i g_i(x) + \sum_{i=1}^{p} \beta_i h_i(x) \tag{19}$$

Wir bezeichnen x als die **primale** Variable und α, β als **duale** Variablen (oder Lagrange-Multiplikatoren). Die Idee ist nun, dass die Maximierung der Lagrange-Funktion das ursprüngliche Optimierungsproblem formuliert. Wir definieren die **primale Funktion** zu

$$\theta_{\mathcal{P}} = \sup_{\alpha,\beta:\alpha_i \geqslant 0,\forall i} \{\mathcal{L}(x, \alpha, \beta)\}$$
$$= f(x) + \sup_{\alpha,\beta:\alpha_i \geqslant 0,\forall i} \Big[\sum_{i=1}^{k} \alpha_i g_i(x) + \sum_{i=1}^{p} \beta_i h_i(x) \Big] \tag{20}$$

Mit den definierten Nebenbedingungen gilt, dass der optimale Wert von $\alpha_i = 0$ sein muss und $h_i = 0$ unabhängig von β sein muss . Damit gilt

$$\theta_{\mathcal{P}} = f(x) + \begin{cases} 0 & \text{, für x erfüllt Nebenbedingungen} \\ \infty & \text{, für x erfüllt Nebenbedingungen nicht} \end{cases} \tag{21}$$

Das ursprüngliche Minimierungsproblem kann nun als **primales Problem** formuliert werden zu

$$\inf_x \theta_{\mathcal{P}}(x) = \inf_x \sup_{\alpha,\beta:\alpha_i \geqslant 0,\forall i} \{\mathcal{L}(x, \alpha, \beta)\}$$

$$p^* = \theta_{\mathcal{P}}(x^*).$$

Das **duale Problem** kann nun durch Tausch der Maximierung und Minimierung definiert werden als

$$\sup_{\alpha,\beta:\alpha_i \geqslant 0,\forall i} \{\inf_x \mathcal{L}(x, \alpha, \beta)\} = \sup_{\alpha,\beta:\alpha_i \geqslant 0,\forall i} \theta_{\mathcal{D}}(\alpha, \beta)$$

$$d^* = \theta_{\mathcal{D}}(\alpha^*, \beta^*).$$

Dabei wird $\theta_{\mathcal{D}}$ als **duale Funktion** bezeichnet. Das duale Problem unterscheidet sich grundsätzlich vom ursprünglichen Optimierungsproblem. Die Differenz $p^* - d^*$ wird als **Dualitäts-Lücke** bezeichnet. Für **schwache** Dualität kann gezeigt werden, dass

$$p^* \geqslant d^*.$$

Für konvexe Optimierungsprobleme gilt häufig die **starke** Dualität, $p^* = d^*$. Mithilfe der **Slater Bedingung** kann auf die hinreichenden Voraussetzungen für eine starke Dualität geprüft werden. Hiernach ist eine starke Dualität gegeben wenn mindestens ein Punkt x in der Definitionsmenge existiert, welcher alle Nebenbedingungen strikt erfüllt (z.B. $g_i(x) < 0, \forall i$). Darüber hinaus erfüllen Optimierungsprobleme mit starker Dualität notwendiger Weise auch die sog. **Karush-Kuhn-Tucker (KKT)** Bedingungen:

- $g_i(x^*) \leqslant 0, i = 1, ..., k$ und $h_i(x^*) = 0, i = 1, ..., p$
- $\alpha_i^* \geqslant 0, i = 1, ..., k$
- $\alpha_i^* g_i(x^*) = 0, i = 1, ..., k$ (Komplementaritätsbedingung)
- $\nabla_x \mathcal{L}(x^*, \alpha^*, \beta^*) = \vec{0}$

3.3 Optimale Margin Klassifikation: Duale Formulierung

Betrachtet werden soll nun die duale Funktion der optimalen Margin Klassifikation. Zu optimieren ist die bereits bekannte Funktion

$$\min_{w,b} \frac{1}{2}||w||^2$$

mit der umformulierten Nebenbedingung $g_i(w) = -y^{(i)}(w^T x^{(i)} + b) + 1 \leqslant 0$. Für jedes Trainingspaar i liegt solch eine Nebenbedingung vor. Es ist zu bemerken, dass die vorliegende (quadratische) Optimierungsfunktion konvex ist und eine affine Nebenbedingung besitzt. Für strenge Dualität reicht es also aus, wenn ein Punkt im Definitionsbereich gefunden werden kann. Mit $w = \vec{0}, b = 1$ ist dies immer möglich, weshalb starke Dualität angenommen werden kann. Die Optimierungsfunktion kann also mithilfe der dualen Formulierung äquivalent betrachtet werden. Nun kann die Lagrange-Funktion formuliert werden zu

$$\mathcal{L}(w, b, \alpha) = \frac{1}{2}||w||^2 - \sum_{i=1}^{m} \alpha_i [y^{(i)}(w^T x^{(i)} + b) - 1] \tag{22}$$

$$= \sum_{i=1}^{m} \alpha_i + \frac{1}{2}||w||^2 - w^T \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)} \tag{23}$$

Da für die Margin Klassifikation keine Gleichheitsbedingungen vorliegen, gibt es keine β-Terme.

Mit der Komplementaritätsbedingung ist zu bemerken, dass $\alpha_i > 0$ nur für diejenigen Traingsdaten auftritt, für die gilt $g_i(w) = 0$ und damit $y^{(i)}(w^T x^{(i)} + b) = 1$. Diese Punkte bilden die **Support Vektoren**, (4). Dabei ist zu bemerken, dass die Anzahl der Support-Vektoren wesentlich kleiner sein kann, als die Gesamtanzahl der Trainingsdaten.

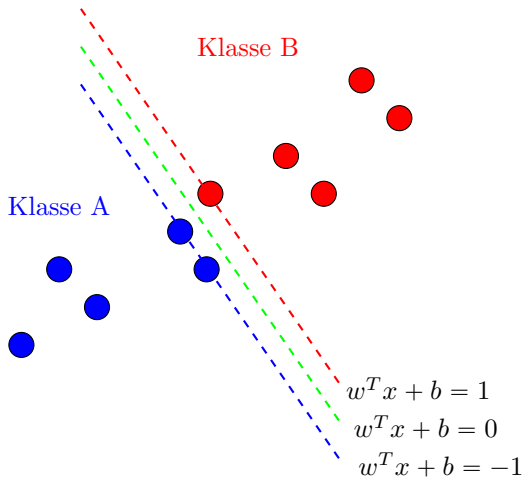

Figure 4: Support Vektoren.

Um das duale Problem zu formulieren, muss zunächst

$$\theta_{\mathcal{D}}(\alpha, \beta) = \inf_{w,b} \mathcal{L}(w, b, \alpha)$$

gefunden werden. Da die Lagrange Funktion quadratisch mit w verläuft kann die Minimierung mithilfe der partiellen Ableitung bestimmt werden. Hierbei ist

$$\frac{\partial}{\partial w}\mathcal{L}(w, b, \alpha) = 0 \iff w - \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)} = 0$$

wobei impliziert wird, dass

$$w = \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)}. \tag{24}$$

Zu bemerken ist hier, dass die Gewichtsparameter w also als Linearkombination der Eingabe-/ Ausgabewerte und der dualen Variable α gebildet werden. Damit kann die Lösung für das primale Problem sehr einfach über die Lösung des dualen Problems gefunden werden

$$w^* = \sum_{i=1}^{m} \alpha_i^* y^{(i)} x^{(i)}.$$

Weiterhin gilt

$$\frac{\partial}{\partial b}\mathcal{L}(w, b, \alpha) = 0 \Longleftrightarrow \sum_{i=1}^{m} \alpha_i y^{(i)} = 0 \tag{25}$$

Nun kann durch einsetzen in die Lagrange Funktion das duale Problem formuliert werden zu

$$\sup_{\alpha} \mathcal{L}(w, b, \alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j \langle x^{(i)}, x^{(j)} \rangle$$

mit den Nebenbedingungen $\alpha_i \geqslant 0, i = 1, ..., m$ und $\sum_{i=1}^{m} \alpha_i y^{(i)} = 0$ Der entscheidende Vorteil ist nun, dass die Optimierung des dualen Problems nicht mehr von w, b abhängig ist. Darüber hinaus kann das Skalarprodukt $\langle x^{(i)}, x^{(j)} \rangle$ sehr effizient berechnet werden.

Mit der gefundenen Bedingungn für w kann eine neue Vorhersage für den Testpunkt x ebenfalls sehr effizient berechnet werden zu

$$\begin{aligned} w^T x + b &= \left(\sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)} \right)^T x + b \\ &= \sum_{i=1}^{m} \alpha_i y^{(i)} \langle x^{(i)}, x \rangle + b \end{aligned} \tag{26}$$

3.4 Soft-Margin SVM

Nicht immer sind zwei Klassen exakt linear separierbar. Damit entsprechende Ausreißer keinen übermäßigen Einfluss auf die Entscheidungsgrenze haben, muss ein Regularisierungsterm hinzugefügt werden. Das neue Minimierungsproblem wird üblicherweise formuliert als

$$\min_{w,b} \frac{1}{2}||w||^2 + \frac{C}{m} \sum_{i=1}^{m} \xi_i \tag{27}$$

mit den Nebenbedingungen $y^{(i)}(w^T x^{(i)} + b) \geqslant 1 - \xi_i, i = 1, ..., m$ und $\xi_i \geqslant 0$. Mithilfe der Konstanten C kann nun die Regularisierung des Terms bestimmt werden.

- C $\searrow$: Für kleine Werte von C (hohe Regularisierung) priorisiert der Algorithmus die Maximierung der geoemtrischen Spanne. Es wird also einzelnen Punkten "erlaubt" auszureißen, wobei auch die Wahrscheinlichkeit von Missklassifikation erhöht wird. Der Algortihmus tendiert also zum Underfitting.
- C $\nearrow$: Mit großen Werten von C (niedriger Regularisierung) wird eine kleinere geoemtrische Spanne in Kauf genommen, um möglichst viele Datenpunkte zu berücksichtigen und eine Missklassifikation zu vermeiden. Die mögliche Spanne für Ausreißer wird also erniedrigt. Gleichzeitig wird der Algorithmus anfälliger für stärkere Datenvariationen.

Die Lagrange Funktion kann nun formuliert werden zu

$$\mathcal{L}(w, b, \xi, \alpha, \lambda) = \frac{1}{2}||w||^2 + \frac{C}{m} \sum_{i=1}^{m} \xi_i + \sum_{i=1}^{m} \alpha_i (1 - \xi_i - y^{(i)}(w^T x^{(i)} + b)) - \sum_{i=1}^{m} \lambda_i \xi_i \tag{28}$$

Dabei sind w, b, ξ die primalen Variablen und α, λ die dualen Variablen. Wie bereits in der dualen Form der Margin Klassifikation gezeigt, kann die duale Form durch Minimierung

$$\theta_{\mathcal{D}}(\alpha, \lambda) = \inf_{w,b,\xi} \mathcal{L}(w, b, \alpha)$$

gefunden werden. Die partiellen Ableitungen ergeben sich zu

$$\frac{\partial}{\partial w}\mathcal{L}(w,b,\xi,\alpha,\lambda) = 0 \Longleftrightarrow w - \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)} = 0 \tag{29}$$

$$\frac{\partial}{\partial b}\mathcal{L}(w,b,\xi,\alpha,\lambda) = 0 \Longleftrightarrow \sum_{i=1}^{m} \alpha_i y^{(i)} = 0 \tag{30}$$

$$\frac{\partial}{\partial \xi}\mathcal{L}(w,b,\xi,\alpha,\lambda) = 0 \Longleftrightarrow \frac{C}{m} - \alpha_i - \lambda_i = 0 \tag{31}$$

Zusammenfassend kann das duale Problem formuliert werden als

$$\sup_{\alpha} \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j \langle x^{(i)}, x^{(j)} \rangle \tag{32}$$

mit den Nebenbedingungen $\alpha_i \in [0, \frac{C}{m}]$, $\sum_{i=1}^{m} \alpha_i y^{(i)} = 0$ und $\lambda_i = C - \alpha_i$. Der einzige **Unterschied** zur optimalen Margin Klassifikation ist also die neue Bedingung für α. Man beachte, dass hier der Effekt des Regularisierungsparameters C eindeutig ersichtlich wird. Kleine Werte von C bestimmen die Grenzen von α_i, womit der Anteil jedes einzelnen Datenpunktes auf die Gesamtlösung minimiert wird, da auch hier gilt $w^* = \sum_{i=1}^{m} \alpha_i^* y^{(i)} x^{(i)}$. Mit der Komplementaritätsbedingung kann Zusammenhang zwischen den dualen Koeffizienten α_i und der Regularisierung gefunden werden

$$\begin{aligned}
\alpha_i = 0 &\Rightarrow y^{(i)}(w^T x^{(i)} + b) \geqslant 1 \\
\alpha_i \in [0, \frac{C}{m}] &\Rightarrow y^{(i)}(w^T x^{(i)} + b) = 1 \\
\alpha_i = [0, \frac{C}{m}] &\Rightarrow y^{(i)}(w^T x^{(i)} + b) \leqslant 1
\end{aligned}$$

Nachdem die Optimierung durchgeführt wurde (z.B. durch SMO-Verfahren) kann der übrige Parameter b^* mithilfe der Komplementaritätsbedingung ebenfalls bestimmt werden. Man findet schnell, dass gilt

$$b^* = y^{(i)} - w^{*T} x^{(i)}$$

Damit kann jeder beliebige Supportvektor i verwendet werden, um den Parameter b zu bestimmen.

3.5 Kernel-Funktionen

Die Idee hinter den Kernel-Funktionen ist eine wesentliche Vereinfachung der Berechnung von hochdimensionierten Problemen. Wird die Berechnung innerhalb eines Skalarproduktes benötigt, z.B. $\langle \phi(x), \phi(y) \rangle$ kann der sog. **Kernel-Trick** eine sehr effiziente Methode zur Berechnung hervorbringen. Der Ansatz ist hierbei, dass das hochdimensionierte Problem nicht explizit berechnet wird. Stattdessen wird eine sog. **Kernelfunktion** $K(x,y)$ gefunden, welche den Wert des Skalarproduktes berechnet,

$$K(x,y) = \langle \phi(x), \phi(y) \rangle \tag{33}$$

Der erhebliche Vorteil besteht nun darin, dass $K(x,y)$ direkt berechnet werden kann, d.h. $\phi(x)$ selbst und das dazugehörige Skalarprodukt muss nicht explizit berechnet werden .

Als **Beispiel** kann die Transformation $\phi(x) : \mathbb{R}^2 \longrightarrow \mathbb{R}^3$ betrachtet werden, wobei gelten soll

$$\phi(x) = \begin{bmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}x_1 x_2 \end{bmatrix}$$

Anstatt nun ein entsprechendes Skalarprodukt $\langle \phi(x), \phi(y) \rangle$ direkt zu berechnen, kann gesehen werden, dass gilt

$$\phi(x)^T \phi(y) = (x^T y)^2 = K(x,y)$$

Der Wert des Skalarproduktes in $\mathbb{R}^3$ kann für dieses Beispiel also über ein simples Quadrieren in $\mathbb{R}^2$ bestimmt werden.

Es kann also jedes Skalarprodukt in ein höher-dimensionales Problem überführt werden, indem das i, j-te Skalarprodukt durch eine entsprechende Kernelfunktion ersetzt wird. Für m Punkte ist dann die **Kernel-Matrix** $K \in \mathbb{R}^{m \times m}$ gegeben mit $K_{i,j} = K(x^{(i)}, x^{(j)})$. Mit dem **Theorem von Mercer** gilt, dass eine gültige Kernel-Matrix notwendiger Weise symmetrisch und positiv semi-definit sein muss.

Insgesamt ist nun der Vorteil, dass ein Algorithmus auf Basis einer Kernel-Matrix durch simplen Austausch der Matrix verändert bzw. angepasst werden kann (sog. **Kernel-Trick**). Beispielsweise entspricht die Kernelfunktion

$$K(w, x) = (1 + \langle w, x \rangle)^M$$

eines Feature-Mappings auf alle Monome des M-ten Grades.

Eine hinlänglich bekannte Kernel-Funktion ist der sog. **Gauss-Kernel (Radial Basis Function (RBF))**, welche die Ähnlichkeit zweier Eingaben quantifiziert,

$$K(w, x) = \exp(-\frac{||w - x||^2}{2\sigma^2})$$

Dabei wird σ als der Bandbreitenparameter bezeichnet. Tatsächlich entspräche diese Kernel-Funktion sogar einem unendlich-dimensionalen Skalarpodukt, welches herkömmlich nicht berechenbar wäre.

3.6 SVM: Zusammenfassung

Die Margin-Klassifikation kann mithilfe von Lagrange in seine duale Form überführt werden.

$$\sup_{\alpha} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j \langle x^{(i)}, x^{(j)} \rangle$$

Diese Darstellung erlaubt nun die Verwendung von Kernel-Funktionen, da das Optimierungsproblem auf ein Skalarprodukt reduziert wird. Üblicherweise wird erst die Kombination dieser beiden Elemente als **Support Vector Machine** bezeicnet.

4 Ensemble Methoden

Ensemble-Methoden kombinieren mehrere maschinelle Lernmodelle, um die Gesamtleistung und die Vorhersage-Genauigkeit zu verbessern. Sie nutzen die Stärken verschiedener Modelle, um individuelle Schwächen auszugleichen, was zu robusteren und genaueren Vorhersagen führt. Man unterscheidet allgemein **parallele** Methoden in denen die Stärken der individuellen Modelle genutzt und letztlich kombiniert werden. Es existieren aber auch **sequentielle** Ansätze, in denen eine schrittweise Verbesserung des Modells erzielt wird.

4.1 Bagging und Random Forest

Bei der sog. **bootstrap-aggregation (bagging)** wird der zur Verfügung stehende Datensatz in B Untermengen aufgeteilt wird. Die Idee dahinter ist, Algorithmen mit hoher Varianz robuster zu machen. Entscheidungsbäume beispielsweise neigen zur Überanpassung, da ihre Konstruktion stark von den verfügbaren Daten abhängig ist. Mit dem Bagging kann die Varianz näherungsweise verringert werden zu

$$\mathbb{E}\hat{\mu}^2 \approx \frac{1}{B} \sum_{i=1}^{B} [\hat{\mu}(\mathcal{D}^{(i)})]^2.$$

Da die Untermengen nicht völlig unabhängig voneinander sind, ist die Reduktion der Varianz allerdings nicht ideal. Für jede Untermenge wird dann die Hypothese $\hat{f}_1, \hat{f}_2 ..., \hat{f}_B$ gebildet und anschließend zu einer einzelnen kombiniert,

$$\hat{f}_{ges} = f(\hat{f}_1, \hat{f}_2 ..., \hat{f}_B).$$

Auf welche Art die verschiedenen Ergebnisse zusammengefasst werden (sog. **Ensemble**-Methode), ist nicht direkt vorgegeben und kann unterschiedlich aussehen, z.B. Bildung des arithmetischen Mittelwerts.

Bei Entscheidungsbäumen beispielsweise wird üblicherweise zusätzlich zum bagging, in jedem Knoten zur Aufteilung eine zufällige Untermenge der verfügbaren Merkmale d in Betracht gezogen (häufig

$\approx \sqrt{d}$). Dieser Ansatz ist als **Random Forest** bekannt und trägt zusätzlich dazu bei, die Korrelation zwischen den Merkmalen zu verringern.

4.2 Boosting Methoden

Boosting ist eine sequentielle Methode der Verbesserung einer Hypothese. Eine *schwache* Hypothese wird hierzu schrittweise durch verschiedene Techniken (z.B. Gewichtsanpassungen etc.,) verbessert.

Es sei zunächst für ein Regressionsproblem angenommen, dass dem Eingaberaum $\mathcal{X}$ verschiedene Funktionen der Vorhersage (sog. **Basis-Funktion**) zur Verfügung stehen,

$$h_1, ...h_N : \mathcal{X} \to \mathbb{R}$$

Das Vorhersagemodell ist dann die lineare Kombination aus den Basis-Funktionen,

$$f(x) = \sum_{n=1}^{N} \nu_n h_n. \tag{34}$$

Die Klasse der Funktionen ist dann gegeben mit

$$\mathcal{F}_N = \left\{ \sum_{n=1}^{N} \nu_n h_n(x) | \nu_n \in \mathbb{R}, h_n \in \mathcal{H}, n = 1, ..., N \right\} \tag{35}$$

Der Lernprozess besteht dann aus der Bestimmung der Koeffizienten ν_n und der Funktionen h_n.

4.2.1 Forward Stagewise Additive Modeling (FSAM)

FSAM ist ein iterativer Prozess zur Regression, bei der zunächst mit einer einfachen Vorhersagefunktion (z.B. $f_0 \equiv 0$) angefangen wird und weitere Funktionen additiv hinzugefügt werden. Nach $n-1$ Etappen liegt dann die Funktion

$$f_{n-1} = \sum_{i=1}^{n-1} \nu_i h_i$$

vor. Das Ziel ist dann für die n-te Runde die Funktion

$$f_n = f_{n-1} + \nu_n h_n$$

zu bestimmen und hierbei

- die **Schrittweite** in Form von $h_n \in \mathcal{H}$ zu finden
- die **Schrittgröße** ν_n zu bestimmen.

Für einen Datensatz $\mathcal{D} = \{(x_1, y_1), ..., (x_m, y_m)\}$ wäre dann die folgende Optimierung durchzuführen

$$(\nu_n, h_n) = \arg \min_{\nu \in \mathbb{R}, h \in \mathcal{H}} \frac{1}{m} \sum_{i=1}^{m} l\left(y^{(i)}, f_{n-1}(x^{(i)}) + \nu h(x^{(i)})\right)$$

Beim sog. L^2**-Boosting** wird der quadratische Fehler zwischen dem Fehler der aktuellen Funktion $(y^{(i)} - f_{n-1}(x^{(i)}))$ und der neuen Funktion h minimiert,

$$J(h) = \frac{1}{m} \sum_{i=1}^{m} \left([y^{(i)} - f_{n-1}(x^{(i)})] - h(x^{(i)}) \right)^2 \tag{36}$$

4.2.2 Adaptive Boosting (Ada Boosting)

Eine bekannte Form von FSAM ist das sog. **Ada-Boosting**, bei dem eine schwache Hypothese mithilfe von fortlaufend angepassten Gewichten der Trainingsdaten schrittweise verbessert. Wir nehmen zunächst ein binäres Klassifizierungsproblem an, $h : \mathbb{R}^n \to \{-1, 1\}$. Es sei weiterhin angenommen, für unseren

Datensatz $\mathcal{D} = \{(x_1, y_1), ..., (x_m, y_m)\}$ existiere eine Gewichtung $p = (p^{(1)}, p^{(2)}, ..., p^{(m)})$, üblicherweise mit

$$\sum_{i=1}^{m} p^{(i)} = 1.$$

Ein sog. **schwache** Hypothese ist nun jede, welche mit einer Marge $\gamma > 0$ besser entscheidet als ein rein zufallsbasierter Klassifizierer (Fehlerrate ≈ 0.5) und damit eine leicht bessere Fehlerrate erreicht,

$$\sum_{i=1}^{m} p^{(i)} 1\{y^{(i)} \neq h_n(x^{(i)})\} \leqslant \frac{1}{2} - \gamma \tag{37}$$

Zu Beginn wird nun jedem Trainingsbeispiel dasselbe Gewicht zugeordnet, $p^{(i)} = \frac{1}{m}$. Die Idee ist nun, jedem falsch klassifizierten Trainigspaar eine höhere Gewichtung zuzuteilen, sodass dieses im nächsten Iterationsschritt stärker von der Hypothese berücksichtigt wird.

In jedem Iterationsschritt n wird zunächst die Gesamtfehlerrate ϵ_n bestimmt,

$$\epsilon_n = \frac{\sum_{i=1}^{m} p^{(i)} 1\{y^{(i)} \neq h_n(x^{(i)})\}}{\sum_{i=1}^{m} p^{(i)}}$$

Damit wird der Hypothese einer jeden Iteration eine eigene Gewichtung α_n zugewiesen, welches in der Gesamtbetrachtung bestimmt, wie hoch der Einfluss jeweils ausfällt,

$$\alpha_n = \frac{1}{2} \ln(\frac{1 - \epsilon_n}{\epsilon_n}) \tag{38}$$

Die Gewichte werden nun jeweils abhängig von der sog. **funktionalen** Spanne $y^{(i)} h(x^{(i)})$ mithilfe einer exponentiellen Verlustfunktion bestimmt,

$$p_{n+1}^{(i)} = p_n^{(i)} \exp(-\alpha_n y^{(i)} h_n(x^{(i)}))$$

Je schlechter die Hypothese in der n-ten Iteration abschneidet, desto höher fällt bei einem falsch klassifiziertes Trainingspaar ($y^{(i)} h_n(x^{(i)}) < 0$) eine neue Gewichtung für die darauf folgende Iteration $(n + 1)$ aus. Die finale Klassifizierung erfolgt dann anhand der Summe aller bestimmten Hypothesen,

$$\hat{y}^{(i)} = \sum_{n=1}^{N} \alpha_n h_n(x^{(i)}).$$

4.3 Gradienten Boosting

AdaBoost kann empfindlich auf verrauschte Daten und Ausreißer reagieren, da es versucht, die falsch klassifizierten Stichproben in nachfolgenden Iterationen anzupassen. Darüber hinaus ist es schwierig, immer eine passende Schrittrichtung h zu bestimmen.

Beim **Gradienten Boosting** wird der Ansatz (ähnlich zu Gradient Descent) verfolgt, eine *lokal* günstige Funktion durch Gradientenbildung bezogen auf die Funktion selbst zu finden,

$$-g = -\nabla_h \sum_{i=1}^{m} l(y^{(i)}, h(x^{(i)}))$$

$$-g = -(\partial_{h(x^{(1)})} l(y^{(1)}, h(x^{(1)})), ..., \partial_{h(x^{(m)})} l(y^{(m)}, h(x^{(m)})))$$

Letztlich ist dann das Ziel, diese Funktion so gut wie möglich anzunähern, beispielsweise mithilfe einer quadratischen Fehlerfunktion,

$$\min_{h \in \mathcal{H}} \sum_{i=1}^{m} (-g_i - h(x^{(i)}))^2$$

Als Beispiel sei das **Binomial Booosting** mit der logistischen Verlustfunktion

$$l(y, h(x)) = \log(1 + \exp(-y h(x)))$$

betrachtet. Der dazugehörige Gradient bestimmt sich zu

$$g(x^{(i)}) = -\partial_h(x^{(i)})[\log(1 + \exp(-yh(x^{(i)})))]$$

$$= \frac{y^{(i)}}{1 + \exp(-yh(x^{(i)}))}$$

Damit kann der additive Anteil für die $n - te$ Iteration berechnet werden mit

$$h_n = \arg\min_{h \in \mathcal{H}} \sum_{i=1}^{m} \frac{1}{m} [\frac{y^{(i)}}{1 + \exp(-yh_{n-1}(x^{(i)}))} - h(x^{(i)})]^2$$

wobei immernoch gilt

$$f_n = f_{n-1} + \nu_n h_n.$$