Theorie des maschinellen Lernens

Hans U. Simon

20. Juli 2016

17 Multiklassen-Kategorisierung und Rangordnungsprobleme

Ein Lernproblem mit einer Klasse von Hypothesen der Form $h : \mathcal{X} \to \mathcal{Y}$ heißt *Multiklassen-Kategorisierungproblem*, wenn $\mathcal{Y}$ eine endliche Menge mit $|\mathcal{Y}| \geq 2$ ist. Die Elemente von $\mathcal{Y}$ heißen dann *Kategorien* oder auch *Labels*. Ein binäres Klassifikationsproblem ist gewissermaßen der Spezialfall mit $|\mathcal{Y}| = 2$. Beispiele für Multiklassen-Kategorisierung wären etwa:

Thematische Kategorisierung von Dokumenten: Hier wäre $\mathcal{X}$ eine Menge von Dokumenten und $\mathcal{Y}$ eine Menge möglicher Themen.

Bilderkennung: Hier wäre $\mathcal{X}$ eine Menge von Bildern und $\mathcal{Y}$ eine Menge möglicher auf den Bildern dargestellter Objekte.

Ein *Rangordnungsproblem* besteht (informell gesprochen) darin, eine gegebene Folge von Instanzen gemäß ihrer „Relevanz“ zu ordnen. Beispiele für diese Art von Problemen wären etwa:

Suchmaschinen: Ordne die Resultate einer Suchmaschine bezüglich ihrer Relevanz für die Suchanfrage.

Betrugsaufdeckung: Ordne mitprotokollierte finanzielle online-Transaktionen gemäß ihres Verdachtsgrades auf Betrug.

Wir werden uns in den ersten drei Abschnitten dieses Kapitels mit Multiklassen-Kategorisierung und in den letzten beiden Abschnitten mit dem Rangordnungsproblem befassen.

17.1 Multiklassen-Kategorisierung vermöge Binärklassifikation

In diesem Abschnitt besprechen wir zwei mögliche Arten, Multiklassen-Kategorisierung auf binäre Klassifikation zu reduzieren. Im Folgenden nummerieren wir die $k = |\mathcal{Y}|$ Labels durch und identifizieren ein Label mit seiner Nummer. D.h., wir setzen $\mathcal{Y} = [k]$.

Die „Einer–gegen–Alle“-Reduktion. Für alle $i, j \in [k]$ sei

$$B_j(i) = \begin{cases} 1 & \text{falls } i = j \\ -1 & \text{sonst} \end{cases}$$

die Abbildung die das Label j durch das Label 1 und alle anderen Labels durch das Label -1 ersetzt. Es sei A ein Algorithmus für binäre Klassifikation der, angesetzt auf eine Trainingssequenz S, eine Voraussagefunktion $A(S) : \mathcal{X} \to \mathbb{R}$ berechnet, so dass sign $\circ A(S) : \mathcal{X} \to \{-1, 1\}$ eine binäre Voraussagefunktion darstellt.

Einschub: Natürlich ist nicht verboten, dass bereits $A(S)$ nur die Werte ± 1 annimmt. Falls aber beliebige reelle Werte angenommen werden können, dann kann (intuitiv) das Vorzeichen als das vorausgesagte Label und der Absolutbetrag als der Grad an Zuversicht in diese Voraussage interpretiert werden.

Für eine Trainingssequenz $S = [(x_1, y_1), \ldots, (x_m, y_m)] \in (\mathcal{X} \times [k])^m$ und $B : \mathcal{Y} \to \{-1, 1\}$ definieren wir $B(S)$ als die Trainingssequenz, die aus S durch die Labelsubstitution B hervorgeht, d.h.:

$$B(S) = [(x_1, B(y_1)), \ldots, (x_m, B(y_m))] \in (\mathcal{X} \times \{-1, 1\})^m \ .$$

Im Rahmen der Methode „Einer–gegen–Alle“ verarbeiten wir (mit Hilfe einer Prozedur A für Binärklassifikation) eine gegebene Trainingssequenz $S = [(x_1, y_1), \ldots, (x_m, y_m)] \in (\mathcal{X} \times [k])^m$ wie folgt:

1. Berechne $h_j = A(B_j(S))$ für $j = 1, \ldots, k$.
2. Wähle als Hypothese für die Multiklassen-Kategorisierung die folgende Abbildung $h : \mathcal{X} \to [k]$:
$$\forall x \in \mathcal{X} : h(x) = \text{argmax}_{j \in [k]} h_j(x) \ . \tag{1}$$

Salopp gesprochen entscheidet sich die Hypothese h bei einer gegebenen Instanz $x \in \mathcal{X}$ für das Label j, dessen Binärhypothese h_j die höchste Zuversicht ausstrahlt, dass j im Vergleich zu allen anderen Labels die bessere Wahl darstellt. Falls das beste Label in (1) nicht eindeutig ist, dann legen wir fest, dass unter den optimalen Labels dasjenige mit der kleinsten Nummer gewählt wird.

Das folgende Beispiel zeigt, dass die Methode „Einer–gegen–Alle“ versagen kann.

Beispiel 17.1 *Die Grundmenge $\mathcal{X}$ bestehe aus den drei Punkten x_1, x_2, x_3 der Ebene mit $x_1 = (-1, 1)$, $x_2 = (0, 1)$ und $x_3 = (1, 1)$. Es sei $\mathcal{Y} = [3]$, d.h., es gibt drei Labels. Die Wahrscheinlichkeitsverteilung D habe die Werte $D(x_1, 1) = D(x_3, 3) = 0.4$ und $D(x_2, 2) = 0.2$. Insbesondere wird also dem Punkt x_i fast sicher das Label i zugeteilt. Die Prozedur A liefere ein lineare Voraussagefunktionen h_w mit $h_w(x) = \langle w, x \rangle$. Wir ziehen dann für die Multiklassen-Kategorisierung Hypothesen von der Form*

$$h(x) = \text{argmax}_{i \in [3]} \langle \vec{w}_i, x \rangle$$

für Gewichtsvektoren $\vec{w}_i$ in Betracht. Man überprüft leicht, dass die Hypothese h für die folgende Wahl der drei (normierten) Gewichtsvektoren fehlerfrei ist:

$$\vec{w_1} = \frac{1}{\sqrt{2}}(-1,1) \ , \ \vec{w_2} = (0,1) \ , \ \vec{w_3} = \frac{1}{\sqrt{2}}(1,1) \ .$$

Wir behaupten, dass die Methode „Einer–gegen–Alle“ nicht *zu einer fehlerfreien Hypothese gelangen kann. Wenn wir nämlich die Prozedur A die Hypothese h_2 berechnen lassen, bei welcher das Label 2 alleine gegen die Labels 1 und 3 antritt, dann ist eine beste Wahl für h_2 eine Hypothese, deren Vorzeichen alle Instanzen auf -1 setzt (realisiert zum Beispiel durch den Gewichtsvektor $(0,-1)$).*
Begründung: *Aus Konvexitätsgründen ist es unmöglich mit einer linearen Voraussagefunktion x_2 auf Label 1 und x_1, x_3 beide auf Label -1 zu setzen. Wenn aber auf mindestens einem Punkt ein Fehler gemacht werden muss, dann am besten auf dem, der über die kleinste Wahrscheinlichkeitsmasse verfügt.*
Wenn nun Mehrdeutigkeiten bei argmax *zu Gunsten des Labels mit der kleinsten Nummer aufgelöst werden, dann wird die von „Einer–gegen-Alle“ ermittelte Hypothese h dem Punkt x_2 das Label 1 zuweisen. Damit gilt $L_D(h) \geq 0.2$.*

Die „Alle–Paare“-Reduktion. Für alle $1 \leq i < j \leq k$ sei

$$S'_{i,j} = ((x,y))_{(x,y)\in S \wedge y \in \{i,j\}}$$

die Trainingssequenz, welche aus S entsteht, indem wir alle Paare (x,y) mit $y \notin \{i,j\}$ aus S entfernen. Dann ist $S_{i,j} = B_i(S'_{i,j})$ die Trainingssequenz, die aus $S'_{i,j}$ entsteht, indem wir das Label i durch 1 und das Label j durch -1 ersetzen. Im Rahmen der Methode „Alle–Paare“ verarbeiten wir (mit Hilfe einer Prozedur A für Binärklassifikation) eine gegebene Trainingssequenz $S = [(x_1,y_1),\ldots,(x_m,y_m)] \in (\mathcal{X} \times [k])^m$ wie folgt:

1. Berechne $h_{\{i,j\}} = A(S_{i,j})$ für alle $1 \leq i < j \leq k$.

2. Wähle als Hypothese für die Multiklassen-Kategorisierung die folgende Abbildung $h : \mathcal{X} \to [k]$:
$$\forall x \in \mathcal{X} : h(x) = \text{argmax}_{i\in[k]} \sum_{j \in [k]\setminus\{i\}} \text{sign}(j-i) h_{\{i,j\}}(x) \ .$$

Die Intuition dahinter ist die folgende. Ein positiver Wert von $h_{\{i,j\}}(x)$ drückt aus, dass die Voraussagefunktion das Label $\min\{i,j\}$ (also das kleinere) dem Label $\max\{i,j\}$ vorzieht. Bei negativen Werten verhält es sich umgekehrt. Der Absolutbetrag von $h_{\{i,j\}}(x)$ ist ein Maß für die Zuversicht in diese Labelpräferenz. Die Summe $\sum_{j\in[k]\setminus\{i\}} \text{sign}(j-i) h_{\{i,j\}}(x)$ liefert ein Gesamtmaß für die Zuversicht, dass i im Vergleich zu den Labels $j \neq i$ eine bessere Wahl darstellt.

17.2 Multiklassen-Kategorisierung mit linearen Voraussagefunktionen

Wir beschränken uns in diesem Abschnitt auf den homogenen Fall, d.h., eine lineare Funktion ist durch einen Gewichtsvektor w gegeben (und das Bias b wird auf 0 gesetzt). Bei einem binären Klassifikationsproblem (mit Labels ± 1) und der Abbildung $\mathcal{X} \ni x \mapsto \psi(x) \in \mathcal{F}$ von einem Datenpunkt x auf den Merkmalsvektor $\psi(x)$ liefert ein Gewichtsvektor w die Klassifikation $h_w(x) = \text{sign}(\langle w, \psi(x)\rangle)$. Beachte, dass

$$\text{sign}(\langle w, \psi(x)\rangle) = \text{argmax}_{y\in\{-1,1\}}\langle w, y \cdot \psi(x)\rangle \ .$$

Man kann sich also alternativ vorstellen, dass bei der Berechnung von $h_w(x)$ zwei Bewertungsfunktionen gegeneinander antreten, nämlich $\langle w, y \cdot \psi(x)\rangle$ für $y = -1, 1$, und das Label, das zu einer höheren Bewertung führt, gewinnt. Das Schöne an dieser alternativen Vorstellung ist, dass sie sich mit Hilfe von „Label-sensitiven Feature Maps“ auf Multiklassen-Kategorisierung verallgemeinern lässt. Die Vorgehensweise ist dabei analog zur Vorgehensweise beim Einsatz von Support-Vektor-Maschinen (SVM) für die Zwecke der Binärklassifikation. Wir listen kurz die wichtigsten Konzepte auf, die im Kapitel über SVM und Kernels eine tragende Rolle spielten:

- „feature map“ $\psi : \mathcal{X} \to \mathcal{F}$
- Verlustfunktion ℓ und empirischer Verlust L_S
- Lineare Voraussagefunktionen h_w (Gewichtsvektor w)
- Separierende Hyperebene
- Hinge-Verlust
- Weiche SVM-Lernregel
- Kontrolle der Informationskomplexität
- Kernels und kernbasierte Verfahren

Alle diese Konzepte werden im Folgenden an die Spezifika der Multiklassen-Kategorisierung angepasst. Es wird sich zeigen, dass sowohl „feature maps“ als auch Verlustfunktionen in einer „Label-sensitiven“ Variante benötigt werden.

Label-sensitive Abbildungen in den Merkmalsraum. Das Zusammenspiel von Merkmals- und Gewichtsvektoren lässt sich im Falle der Multiklassen-Kategorisierung beschreiben wie folgt:

- Wir benutzen eine *Label-sensitive* Abbildung

$$\mathcal{X} \times \mathcal{Y} \ni (x, y) \mapsto \psi(x, y) \in \mathcal{F} \ .$$

- Die Hypothese zu einem Gewichtsvektor w wertet $\langle w, \psi(x,y)\rangle$ für alle $y \in \mathcal{Y}$ aus und entscheidet sich für das Label, das zu der höchsten Bewertung führt. D.h.,

$$h_w(x) = \text{argmax}_{y\in\mathcal{Y}} \langle w, \psi(x,y)\rangle \ . \tag{2}$$

Unsere übliche Herangehensweise bei binären Klassifikationproblemen entspricht dann dem Spezialfall $\mathcal{Y} = \{-1, 1\}$ und $\psi(x,y) = y \cdot \psi(x)$. Wir wollen anhand zweier Beispiele mögliche Wahlen von ψ ins Spiel bringen.

Beispiel 17.2 (Multivektor-Konstruktion) *Es sei $\mathcal{Y} = [k]$, $\mathcal{F} = \mathbb{R}^d$ und $\psi : \mathcal{X} \to \mathcal{F}$ eine herkömmliche (Label-insensitive) Abbildung. Wir wollen die Freiheit haben, das Paar (x, j) für jede Wahl von $j \in [k]$ mit einem separaten Vektor $\vec{w}_j \in \mathbb{R}^d$ zu bewerten, um uns dann für das Label j mit einem möglichst großen Wert von $\langle \vec{w}_j, \psi(x)\rangle$ zu entscheiden. Dies lässt sich mit einem einzigen Gewichtsvektor $\vec{w} \in \mathbb{R}^{dk}$ und einer Label-sensitiven Abbildung Ψ ereichen wie folgt:*

$$w = \begin{bmatrix} \vec{w_1} \\ \vdots \\ \vec{w_k} \end{bmatrix} \text{ und } \Psi(x,j) = \begin{bmatrix} \vec{z_1} \\ \vdots \\ \vec{z_k} \end{bmatrix} \ ,$$

wobei

$$\vec{z_i} = \begin{cases} \psi(x) & \text{falls } i = j \\ \vec{0} & \text{sonst} \end{cases} \ .$$

Offensichtlich gilt $\langle \vec{w}, \Psi(x,j)\rangle = \langle \vec{w}_j, \psi(x)\rangle$. Das ist der gewünschte Effekt.

In Beispiel 17.2 kann ψ anwendungsspezifisch gewählt sein, aber die zusammengesetzte Abbildung Ψ bringt kein zusätzliches Anwendungswissen ein. Die Dimension dk des Merkmalsraumes $\mathcal{F}$ wächst linear mit der Anzahl k der Labels. Im folgenden Beispiel geht Anwendungswissen in einer expliziteren Weise in die Konstruktion der Abbildung ψ ein und die Dimension der Merkmalsvektoren ist gänzlich unabhängig von der Anzahl der Labels.

Beispiel 17.3 (TF-IDF-Konstruktion) *Es sei $\mathcal{X}$ eine Menge von Dokumenten und $\mathcal{Y}$ eine Menge möglicher Themen. Es liege eine Kollektion $w_1, \ldots, w_d$ von Schlüsselwörtern vor. Es bezeichne $\text{TF}(j,x)$ die Anzahl der Vorkommen des Wortes w_j im Dokument x. „TF“ steht für „Term Frequency“. Weiter sei $\text{DF}(j,y)$ eine Schätzung für die relative Häufigkeit, mit der das Wort w_j in Dokumenten zu einem von y verschiedenen Thema vorkommt. „DF“ steht für „Document Frequency“. Es sei dann schließlich $\psi = (\psi_j)_{j\in[d]}$ die durch*

$$\psi_j(x,y) = \text{TF}(j,x) \cdot \log \frac{1}{\text{DF}(j,y)}$$

gegebene Abbildung. Der Term $\psi_j(x,y)$ wird als „Term-Frequency-Inverse Document-Frequency (TF-IDF)“ bezeichnet. Er ordnet dem Paar (x,y) einen hohen Wert zu, wenn w_j in dem Dokument x oft vorkommt, aber vergleichsweise selten in Dokumenten zu einem von y verschiedenen Thema.

Label-sensitive Kosten- und Verlustfunktion. Die 0–1-Verlustfunktion ist bei Multiklassen-Kategorisierung nicht immer adäquat. Nehmen wir als Beispiel das Problem, in einem Bild das dargestellte Objekt zu erkennen. Die Verwechslung von „Tiger“ und „Katze“ ist weniger gravierend als die Verwechslung von „Tiger“ und „Delphin“. Allgemein lässt sich das modellieren durch eine Kostenfunktion $\Delta : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$ und eine Verlustfunktion $\ell^\Delta(h,(x,y)) = \Delta(h(x),y)$. Der empirische Verlust einer Hypothese $h : \mathcal{X} \to \mathcal{Y}$ auf einer Trainingssequenz

$$S = [(x_1,y_1),\ldots,(x_m,y_m)] \in (\mathcal{X}\times\mathcal{Y})^m \tag{3}$$

ergibt sich dann gemäß der Formel

$$L_S^\Delta(h) = \frac{1}{m}\sum_{i=1}^m \ell^\Delta(h,(x_i,y_i)) = \frac{1}{m}\sum_{i=1}^m \Delta(h(x_i),y_i) \ .$$

Die 0–1-Verlustfunktion ist der Spezialfall, der sich durch die Setzung $\Delta(y,y') = \mathbb{1}_{[y'\neq y]}$ ergibt.

Lineare Multiklassen-Voraussagefunktionen. Es sei $\psi : \mathcal{X}\times\mathcal{Y} \to \mathcal{F}$ eine Label-sensitive Abbildung und $W \subseteq \mathcal{F}$ eine Teilmenge aller Merkmalsvektoren aus $\mathcal{F}$. Das Paar (ψ, W) spezifiziert dann die folgende Familie von Multiklassen-Voraussagefunktionen:

$$\mathcal{H}_{\psi,W} = \{x \mapsto h_w(x) = \mathrm{argmax}_{y\in\mathcal{Y}}\langle w,\psi(x,y)\rangle : w \in W\} \ .$$

Das Pendant zur separierenden Hyperebene. Es sei S gemäß (3) eine Trainingssequenz und $\Delta : \mathcal{Y}\times\mathcal{Y} \to \mathbb{R}_+$ eine Kostenfunktion. Dem Gewichtsvektor, der eine separierende Hyperebene repräsentiert, entspricht bei Multiklassen-Kategorisierung die folgende Wahl von w^*:

$$w^* = \mathrm{argmin}_{w\in W} L_S^\Delta(h_w) \ . \tag{4}$$

In Worten: wähle einen Gewichtsvektor mit dem kleinstmöglichen empirischen Δ-Verlust auf S. Angenommen $\mathcal{F} = W = \mathbb{R}^d$ und wir sind im realisierbaren Fall, dann lässt sich ein Gewichtsvektors w mit $L_S^\Delta(h_w) = 0$ mit Hilfe von linearer Programmierung finden. Wir suchen nämlich einen Vektor w, der folgende linearen Randbedingungen erfüllt:

$$\forall i \in [m], \forall y \in \mathcal{Y}\setminus\{y_i\} : \langle w,\psi(x_i,y_i)\rangle > \langle w,\psi(x_i,y)\rangle \ .$$

Da wir w hochskalieren können, dürfen die Ungleichungen $\langle w,\psi(x_i,y_i)\rangle > \langle w,\psi(x_i,y)\rangle$ durch $\langle w,\psi(x_i,y_i)\rangle \geq \langle w,\psi(x_i,y)\rangle + 1$ ersetzt werden, so dass wir lineare Randbedingungen in der zulässigen $\geq$-Form erhalten..

Verallgemeinerter Hinge-Verlust. Im nicht realisierbaren Fall ist die Lernregel (4) nicht effizient implementierbar (außer wenn P = NP). Wir suchen einen ähnlichen Ausweg

wie seinerzeit beim Übergang von der harten zur weichen SVM-Lernregel. Als Surrogat-Kostenfunktion dient uns die *verallgemeinerte Hinge-Verlustfunktion*, die definiert ist wie folgt:

$$\ell^{hinge}(h_w,(x,y)) = \max_{y'\in\mathcal{Y}} \left(\Delta(y',y) + \langle w, \psi(x,y') - \psi(x,y)\rangle\right) \ .$$

Der empirische Hinge-Verlust auf einer Trainingsmenge S ergibt sich demnach aus

$$L_S^{hinge}(h_w) = \frac{1}{m}\sum_{i=1}^m \ell^{hinge}(h_w,(x_i,y_i)) = \frac{1}{m}\sum_{i=1}^m \max_{y'\in\mathcal{Y}} \left(\Delta(y',y_i) + \langle w, \psi(x,y') - \psi(x_i,y_i)\rangle\right) \ .$$

Wir machen folgende Beobachtung:

Lemma 17.4 *Es gilt*

$$\ell^{\Delta}(h_w,(x,y)) = \Delta(h_w(x),y) \leq \ell^{hinge}(h_w,(x,y)) \ ,$$

d.h., der Δ-Verlust von h_w auf (x,y) lässt sich mit dem verallgemeinerten Hinge-Verlust von h auf (x,y) nach oben beschränken.

Beweis Aus der Definition von $h_w(x)$ ergibt sich

$$\langle w, \psi(x,y)\rangle \leq \langle w, \psi(x,h_w(x))\rangle \ .$$

Es folgt

$$\Delta(h_w(x),y) \leq \Delta(h_w(x),y) + \langle w, \psi(x,h_w(x)) - \psi(x,y)\rangle \leq \ell^{hinge}(h_w,(x,y)) \ .$$

qed.

Aus Lemma 17.4 folgt natürlich direkt die Ungleichung $L_S^{\Delta}(h_w) \leq L_S^{hinge}(h_w)$.

Das Pendant zur weichen SVM-Lernregel. Die weiche SVM-Lernregel entspricht im Falle von Multiklassen-Kategorisierung der folgenden Wahl des Gewichtsvektors:

$$w^* = \mathrm{argmin}_{w\in\mathcal{F}} \left(\lambda\|w\|^2 + L_S^{hinge}(h_w)\right) \ . \tag{5}$$

Es lässt sich leicht nachrechnen, dass im Spezialfall von Binärklassifikation, $\Delta(y,y') = \mathbb{1}_{[y'\neq y]}$ und $\psi(x,y) = yx/2$, die allgemeine Hinge-Verlustfunktion das uns vertraute Aussehen hat, d.h., $\ell^{hinge}(h_w,(x,y)) = \max\{0, 1 - y\langle w,x\rangle\}$. Die Lernregel (5) stimmt in diesem Spezialfall dann mit der uns bekannten weichen SVM-Lernregel überein.

Kontrolle der Informationskomplexität. Ähnlich wie im Fall der Binärklassifikation lässt sich der Generalisierungsfehler auch im Falle von Multiklassen-Kategorisierung unabhängig von der Dimension des Hilbert-Raumes $\mathcal{F}$ nach oben beschränken. S. etwa Korollar 17.1 im Lehrbuch. Auf Beweistechniken für solche Resultate können wir im Rahmen dieser Vorlesung nicht näher eingehen.

Kernbasierte Verfahren. Ähnlich wie bei der Binärklassifikation ist auch im Fall der Multiklassen-Kategorisierung der Einsatz von Kernels und von kernbasierten Verfahren nötig, um die Berechnungskompelxität zu beherrschen (insbesondere wenn die Merkmalsvektoren in einem extrem hoch-dimensionalen Hilbert-Raum leben). Im Rahmen dieser Vorlesung können wir auf Details hierzu nicht eingehen.

17.3 Multiklassen-Kategorisierung mit strukturierter Objekten

Spezielle Probleme entstehen bei Multiklassen-Kategorisierung, wenn die Labelmenge $\mathcal{Y}$ strukturierte Objekte enthält und $|\mathcal{Y}|$ exponentiell mit einem Komplexitätsparameter r wächst. Als Beispiel mag das Problem der Erkennung handgeschriebener Zeichen[1] dienen. Jedes einzelne Zeichen eines handgeschriebenen Textes stehe als Bild zur Verfügung. Der Datenraum $\mathcal{X}$ besteht somit aus Sequenzen von Bildern. Die Labelmenge $\mathcal{Y}$ besteht aus Zeichensequenzen, also aus Wörtern über einem Grundalphabet A. Wir nehmen der Einfachheit halber an, dass alle Sequenzen die gleiche Länge r haben. Somit gilt $\mathcal{Y} = A^r$ und $|\mathcal{Y}| = |A|^r$ (eine Zahl, die exponentiell mit der Wortlänge r wächst). Eine fehlerfreie Hypothese h würde jedes Bild der Bildersequenz auf das in ihm dargestellte Zeichen abbilden.

Die Kontrolle der Informationskomplexität wird durch große Labelmengen $\mathcal{Y}$ nicht beeinträchtigt, weil die Schranken für den Generalisierungsfehler i.A. nicht von $|\mathcal{Y}|$ abhängen. Es stellt sich aber die Frage, ob die Berechnungskomplexität noch beherrschbar ist. Zum Beispiel könnte alleine schon die Berechnung von $h_w(x)$ gemäß (2) mindestens $|\mathcal{Y}|$ Rechenschritte kosten.[2]

Wir werden im Folgenden am OCR-Beispiel zeigen, wie (zumindest in konkreten Anwendungsszenarien) das Problem der effizienten Berechnung von $h_w(x)$ gelöst werden kann. Wir setzen $q = |A|$, d.h., unser Grundalphabet A hat q Zeichen. Wir nehmen an, dass jedes Bild einer Bildersequenz aus n Pixeln besteht. Ein Bildersequenz x habe die Form einer $(n \times r)$-Matrix, d.h., $x \in \mathbb{R}^{n \times r}$. Der Eintrag $x_{i,j} \in \mathbb{R}$ repräsentiere den Grauton des i-Pixels vom j-ten Bild der Sequenz. Es folgt die Beschreibung einer denkbaren Label-sensitiven Abbildung $\mathcal{X} \times \mathcal{Y} \to \mathbb{R}$. Die Merkmale eines Merkmalsvektors $\psi(x)$ zerfallen in qn Merkmale vom Typ 1 und q^2 Merkmale vom Typ 2.

Typ 1: Für jedes $i \in [n]$ und jedes $a \in A$ definieren wir das Merkmal

$$\psi_{i,a,1}(x, y) = \frac{1}{r} \sum_{j=1}^{r} x_{i,j} \mathbb{1}_{[y_j = a]} \ .$$

Dieses Merkmal könnte nützlich sein, wenn bestimmte Grauwerte des i-ten Pixels das abgebildete Zeichen a von anderen Zeichen unterscheiden.

Typ 2: Für jedes Paar $a, b \in A$ definieren wir das Merkmal

$$\psi_{a,b,2}(x, y) = \frac{1}{r} \sum_{j=2}^{r} \mathbb{1}_{[y_{j-1} = a]} \mathbb{1}_{[y_j = b]} \ .$$

[1] auch OCR genannt, wobei „OCR“ für „Optical Character Recognition“ steht

[2] Ähnliche Probleme stellen sich beim Lösen des Optimierungsproblems (5).

Hohe Werte eines solchen Merkmals zeigen an, dass der Buchstabe a häufig von einem b gefolgt wird. Zum Beispiel folgt im Deutschen auf den Buchstaben „q“ häufig ein „u“.

Von diesen insgesamt $qn+q^2$ Merkmalen wird sich vermutlich nur ein kleiner Teil als nützlich erweisen, aber dies bekommen Algorithmen zum Lernen linearer Voraussagefunktionen mit der Wahl des Gewichtsvektors w in den Griff.
Wir definieren jetzt einen Vektor Φ, dessen Komponenten in einem 1–zu–1 Verhältnis zu den Komponenten von ψ stehen:

$$\Phi_{i,a,1}(x_{i,j}, y_j) = x_{i,j}\mathbb{1}_{[y_j=a]} \text{ und } \Phi_{a,b,2}(y_{j-1}, y_j) = \mathbb{1}_{[y_{j-1}=a]}\mathbb{1}_{[y_j=b]}$$

und beobachten, dass

$$h_w(x) = y^* = \text{argmax}_{y\in\mathcal{Y}}\langle w, \psi(x,y)\rangle = \text{argmax}_{y\in\mathcal{Y}} \sum_{j=1}^{r} \langle w, \Phi(x_j, y_{j-1}, y_j)\rangle \ .$$

Dabei bezeichnet x_j die j-te Spalte der Matrix x. Die Variable y_{j-1} wird nur für $j \geq 2$ wirklich benötigt. Es sei $g_w(x)$ der durch das Label $h_w(x)$ erzielte „Profitwert“, d.h.,

$$g_w(x) = \max_{y\in\mathcal{Y}} \langle w, \psi(x,y)\rangle = \max_{y\in\mathcal{Y}} \sum_{j=1}^{r} \langle w, \Phi(x_j, y_{j-1}, y_j)\rangle \ .$$

Wir beschreiben im Folgenden, wie sich $g_w(x)$ effizient berechnen lässt. Wir verwenden dynamische Programmierung und benutzen die folgende 2-dimensionale Tabelle:

$$\forall b \in A, \forall k \in [r] : M_{b,k}(x|w) = \max_{y_1\ldots y_k : y_k=b} \sum_{j=1}^{k} \langle w, \Phi(x_j, y_{j-1}, y_j)\rangle \ .$$

Auch bei dieser Tabelle kommt eine Abhängigkeit von y_{j-1} erst ab $j \geq 2$ ins Spiel. Offensichtlich gilt

$$g_w(x) = \max_{b\in A} M_{b,k}(x|w) \ .$$

Die Tabelle $M(x|w)$ lässt sich spaltenweise mit Hilfe der folgenden Rekursionsformel leicht ausfüllen:

$$M_{b,1}(x|w) = \langle w, \Phi(x_1, y_0, b)\rangle \text{ und } M_{b,k+1}(x|w) = \max_a \left(M_{a,k}(x|w) + \langle w, \Phi(x, a, b)\rangle\right) \ .$$

Es gibt zwei Möglichkeiten $h_w(x)$ effizient zu berechnen:

- Wenn beim Ausfüllen der Tabelle $M(x|w)$ ein paar Hilfsgrößen mitgeschleppt werden, dann kann man so nebenbei die beste Zeichensequenz $y^* = (y_1^*, \ldots, y_r^*)$ ausrechnen. S. das Lehrbuch zu den Details dieser Variante.

- Man kann mit Hilfe der Tabelle $M(x|w)$ und „Backtracking“ die beste Sequenz y^* ermitteln. „Backtracking“ bedeutet hier, dass die Tabelle von hinten nach vorn (Spalte r zuerst und Spalte 1 zuletzt) durchmustert wird. Ebenso wird die Sequenz y^* von hinten nach vorn berechnet. Details dazu könnten in den Übungen diskutiert werden.

17.4 Das allgemeine Rangordnungsproblem

Ein Rangordnungsproblem besteht darin, eine gegebene Folge von Instanzen nach ihrer Relevanz zu ordnen. S. dazu auch die zwei eingangs in diesem Kapitel aufgeführten Beispiele (Stichworte: Suchmaschinen, finanzielle online-Transaktionen).

Wir kommen nun zu einer formalen Beschreibung dieser Art von Problemen. Es bezeichne $\mathcal{X}^* = \cup_{n\geq 1}\mathcal{X}^n$ die Menge aller endlichen, nichtleeren Sequenzen über einer Grundmenge $\mathcal{X}$. Somit ist $\mathbb{R}^* = \cup_{r\geq 1}\mathbb{R}^r$ die Menge aller endlichen, nichtleeren Sequenzen über der Menge der reellen Zahlen. Wir betrachten Hypothesen der Form $h : \mathcal{X}^* \to \mathbb{R}^*$, welche Sequenzen $x = (x_1, \ldots, x_r)$ der Länge r stets auf Sequenzen $h(x) = (y_1, \ldots, y_r) \in \mathbb{R}^r$ abbilden. Eine Ungleichung $y_j > y_i$ bedeutet intuitiv, dass die Hypothese h die Instanz x_j in der Sequenz x für relevanter hält als die Instanz x_i. Falls die Komponenten in y paarweise verschieden sind, ergibt sich dadurch eine lineare Ordnung auf den Instanzen. Andernfalls erhalten wir lediglich eine partielle Ordnung.[3] Markierte Trainingsbeispiele stammen bei Rangordnungsproblemen über der Grundmenge $\mathcal{X}$ aus der Menge $\cup_{r\geq 1}(\mathcal{X}^r \times \mathbb{R}^r)$, d.h., ein markiertes Beispiel besteht aus einer Sequenz x und einem Vektor y, welcher numerische Bewertungen der in x vorkommenden Instanzen enthält. Wenn es nur darum ginge, die Komponenten von $x = (x_1, \ldots, x_r)$ nach ihrer Relevanz zu ordnen, dann könnten wir an Stelle der numerischen Bewertungen $y = (y_1, \ldots, y_r)$ Rangnummern von 1 bis r einführen:

Definition 17.5 *Es sei $y = (y_1, \ldots, y_r) \in \mathbb{R}^r$. Wenn y_i die k-kleinste Zahl in y ist, dann setzen wir $\bar{y}_i = k$. Wir nennen $\bar{y}_i$ die* Rangnummer *von x_i gemäß y. Wenn mehrere k-kleinste Zahlen $y_{i(1)} = y_{i(2)} = \ldots = y_{i(s)}$ existieren und es gilt $i(1) < i(2) < \ldots < i(s)$, dann vergeben wir die Rangnummern $k, k+1, \ldots, k+s-1$ (in dieser Reihenfolge an die Komponenten $i(1), i(2), \ldots, i(s)$, d.h., wir setzen $\overline{y_{i(j)}} = k + j - 1$ für $j = 1, \ldots, s$.*[4]

Zum Beispiel erhalten wir für $y = (2, 1, 6, -1, 1/2)$ die Gleichung $\bar{y} = (4, 3, 5, 1, 2)$. Intuitiv können wir uns vorstellen, dass höhere Rangnummern eine höhere Relevanz anzeigen. Die numerischen Bewertungen in y enthalten aber mehr Information als die Rangnummern in $\bar{y}$. Zum Beispiel würde auch der Vektor $y' = (50, 1, 100, -1, 1/2)$ auf $\overline{y'} = (4, 3, 5, 1, 2) = \bar{y}$ abgebildet, die numerischen Werte 50 und 100 an Stelle von 2 und 6 könnten aber als nachdrücklichere Voten für die Vergabe der zwei höchsten Rangnummern gewertet werden.

Übungsaufgabe. Entwerfe einen Algorithmus, der zur Berechnung von $\bar{y}$ aus $y \in \mathbb{R}^r$ Zeit $O(r \log r)$ benötigt.

Es bezeichne V_r die Menge aller Tupel aus $[r]^r$, welche paarweise verschiedene Komponenten besitzen (also die Menge der Rangnummern-Vektoren mit r Komponenten). Offensichtlich ist $y \mapsto \bar{y}$ eine Abbildung von $\mathbb{R}^*$ nach $\cup_{r\geq 1}V_r$, die Sequenzen aus $\mathbb{R}^r$ stets auf Elemente von V_r abbildet. Da jede Zahl $k \in [r]$ trivialerweise die k-kleinste Nummer in $[r]$ ist, folgt

[3] In diesem Abschnitt können wir annehmen, dass Gleichheiten der Form $y_i = y_j$ nur selten auftreten. Dies wird sich in Abschnitt 17.5 aber ändern.

[4] Dies ist eine völlig willkürliche Festlegung, die nur dazu dient, die Abbildung $y \mapsto \bar{y}$ auf eindeutige Weise zu definieren.

$\bar{v} = v$ für alle $v \in V_r$, d.h., der Rangnummern-Vektor zu einem Rangnummern-Vektor v ist v selber. Wir schreiben im Folgenden meist einfach V statt V_r (in dem Verständnis, dass die Anzahl der Komponenten in y bzw. $\bar{y}$ immer mit Länge der zugrunde liegenden Sequenz x übereinstimmt).

17.4.1 Kosten- und Verlustfunktionen

Es sei $\Delta : \cup_{r\geq 1}(\mathbb{R}^r \times \mathbb{R}^r) \to \mathbb{R}_0^+$ eine Funktion, die einem Paar (y', y) nicht-negative Kosten zuweist. Wir erinnern daran, dass eine Kostenfunktion Δ eine Verlustfunktion induziert:

$$\ell^{\Delta}(h, (x, y)) = \Delta(h(x), y) \ .$$

Wir schreiben meist einfach ℓ statt ℓ^{Δ}, da die zugrunde liegende Kostenfunktion Δ in der Regel aus dem Kontext ersichtlich ist. Im Folgenden diskutieren wir einige Kostenfunktionen.

0–1 Kostenfunktion. Sie ist gegeben durch $\Delta(y', y) = \mathbb{1}_{[\overline{y'} \neq \bar{y}]}$, d.h., sie weist dem Paar (y', y) Kosten 0 zu, falls die von y' und y induzierten Rangnummern-Vektoren übereinstimmen, und Kosten 1 andernfalls. Offensichtlich gilt $\Delta(y', y) = \Delta(\overline{y'}, \bar{y})$, d.h., die numerischen Bewertungen gehen in die Kosten nur indirekt über die betreffenden Rangnummern ein. Die 0–1 Kostenfunktion ist *keine* gute Wahl, da sie nicht zwischen extrem abweichenden und fast identischen Rangordnungen unterscheidet.

Kendall–Tau Kostenfunktion. Sie zählt[5] die mittlere Anzahl der Paarvergleiche, die bei y' anders ausgehen als bei y:

$$\Delta(y', y) = \frac{2}{(r-1)r} \sum_{1 \leq i < j \leq r} \mathbb{1}_{[\mathrm{sign}(y'_j - y'_i) \neq \mathrm{sign}(y_j - y_i)]} \ .$$

Auch hier gilt $\Delta(y', y) = \Delta(\overline{y'}, \bar{y})$.

NDCG Kostenfunktion. Wir setzen voraus, dass alle numerischen Bewertungsvektoren y nicht-negativ und verschieden vom Nullvektor sind. Weiter sei $D : \mathbb{N} \to \mathbb{R}^+$ eine monoton steigende Funktion. Die folgende Funktion repräsentiert einen kumulativen Gewinn (Cumulative Gain = CG), den y' bezüglich y erzielt:

$$G(y', y) = \sum_{i=1}^{r} D(\overline{y'}) y_i \in \mathbb{R}^+ \ . \tag{6}$$

Es ist nicht schwer zu zeigen[6], dass $G(y', y)$ zu gegebenem y maximal wird, wenn $\overline{y'} = \bar{y}$ (insbesondere also, wenn $y' = y$). Daher repräsentiert die Größe

$$0 < \frac{G(y', y)}{G(y, y)} \leq 1$$

[5] zumindest bei Vektoren y, y', die jeweils paarweise verschiedene Komponenten besitzen

[6] Das wäre eine gute Übungsaufgabe.

so etwas wie einen normierten kumulativen Gewinn (Normalized Cumulative Gain = NCG). Durch Negieren dieser Größe erhalten wir die NDCG Kostenfunktion:

$$\Delta(y', y) = 1 - \frac{G(y', y)}{G(y, y)} = \frac{1}{G(y, y)} \left(\sum_{i=1}^{r} y_i (D(\overline{y_i}) - D(\overline{y'_i})) \right) \quad . \tag{7}$$

Es gilt $\Delta(y', y) = \Delta(\overline{y'}, y)$, d.h., die numerischen Bewertungen in y' gehen in die Kosten nur indirekt über die Rangnummern in $\overline{y'}$ ein. Die numerischen Bewertungen in y hingegen haben einen direkten Einfluss auf $\Delta(y', y)$.
Die Aufgabe der Funktion D ist im Übrigen, den großen Rangnummern in $\overline{y'}$ einen (etwas) höheren Einfluss einzuräumen als den kleinen. Dies wird der Tatsache gerecht, dass bei Anwendungen in der Regel nur die Instanzen von hohem Rang wirklich relevant sind. Eine typische Wahl von D wäre etwa

$$D(i) = \begin{cases} \frac{1}{\log_2(r-i+2)} & \text{falls } i \in \{r-k+1, \ldots, r\} \\ 0 & \text{falls } i \in \{1, \ldots, r-k\} \end{cases} \quad .$$

Die resultierende Kostenfunktion berücksichtigt nur die k Spitzenränge. Für den Spitzenrang r gilt $D(r) = 1$. Für den kleinsten noch berücksichtigten Rang $r - k + 1$ gilt $D(r - k + 1) = 1/\log_2(k + 1)$.

17.4.2 Lineare Voraussagefunktionen und konvexe Surrogat-Zielfunktionen

Wir setzen $\mathcal{X} = \mathbb{R}^d$ voraus. Eine Sequenz $x = (x_1, \ldots, x_r) \in (\mathbb{R}^d)^r$ setzt sich dann aus r Vektoren $x_1, \ldots, x_r$ des Euklidischen Raumes $\mathbb{R}^d$ zusammen. Die folgende Hypothese ermittelt ihre numerischen Bewertungen der Instanzen $x_1, \ldots, x_r$ mit einer linearen Funktion (gegeben durch einen Gewichtsvektor $w \in \mathbb{R}^d$):

$$h_w(x) = (\langle w, x_1 \rangle, \ldots, \langle w, x_r \rangle) \quad .$$

Das Auffinden eines Gewichtsvektors w mit einem minimalen empirischen Verlust

$$L_S^{\Delta}(h_w) = \frac{1}{m} \sum_{i=1}^{m} \Delta(h_w(\vec{x_i}), \vec{y_i})$$

auf einer gegebenen Trainingsmenge

$$S = [(\vec{x_1}, \vec{y_1}), \ldots, (\vec{x_m}, \vec{y_m})]$$

(mit $(\vec{x_i}, \vec{y_i}) \in (\mathbb{R}^d)^r \times \mathbb{R}^r$) ist für die meisten nicht-trivialen Kostenfunktionen Δ ein NP-hartes Problem. Einen Ausweg bieten (wie schon oft in der Vorlesung) konvexe obere Schranken einer nicht konvexen Kostenfunktion, die zu konvexen Surrogat-Zielfunktionen für die eigentliche Zielfunktion $L_S(h_w)$ führen:

Bemerkung 17.6 *Falls $\Delta'(h_w(x), y)$ eine in w konvexe Funktion ist und falls $\Delta'(h_w(x), y) \geq \Delta(h_w(x), y)$, dann ist auch $L_S^{\Delta'}(h_w)$ eine in w konvexe Funktion mit $L_S^{\Delta'}(h_w) \geq L_S^{\Delta}(h_w)$.*

Es genügt daher im Folgenden, eine konvexe obere Schranken Δ' von Δ zu finden. Das führen wir für die Kendall–Tau und für die NDCG Kostenfunktion vor. In beiden Fällen ist Δ' eine Variante der Hinge Kostenfunktion.

Konvexe obere Schranke für die Kendall–Tau Kostenfunktion. Wir beobachten zunächst, dass

$$\mathbb{1}_{[\text{sign}(y_j-y_i)\neq\text{sign}(y'_j-y'_i)]} \leq \mathbb{1}_{[\text{sign}(y_j-y_i)\cdot(y'_j-y'_i)\leq 0]} \ .$$

Für $y' = h_w(x)$ ergibt sich

$$\mathbb{1}_{[\text{sign}(y_j-y_i)(y'_j-y'_i)\leq 0]} = \mathbb{1}_{[\text{sign}(y_j-y_i)\cdot\langle w,x_j-x_i\rangle\leq 0]} \leq \max\{0, 1-\text{sign}(y_j-y_i)\cdot\langle w, x_j-x_i\rangle\} \ .$$

Für die Kendall–Tau Kostenfunktion Δ gilt daher

$$\Delta(h_w(x), y) \leq \frac{2}{(r-1)r} \sum_{1\leq i<j\leq r} \max\{0, 1-\text{sign}(y_j-y_i)\cdot\langle w, x_j-x_i\rangle\} =: \Delta'(h_w(x), y) \ .$$

$\Delta'(h_w, y)$ ist die gesuchte in w konvexe obere Schranke der Kendall–Tau Kostenfunktion.

Konvexe obere Schranke für die NDCG Kostenfunktion. Es sei $D : \mathbb{N} \to \mathbb{R}^+$ eine monoton wachsende Funktion und $G(y', y)$ die in (6) definierte kumulative Gewinnfunktion. Aus der uns bereits bekannten Ungleichung

$$G(y', y) = \sum_{i=1}^{r} D(\overline{y'_i}) y_i \leq G(y, y)$$

lässt sich leicht folgende Gleichung für $\bar{y}$ ableiten[7]:

$$\bar{y} = \text{argmax}_{v\in V} \sum_{i=1}^{r} v_i y_i \ . \tag{8}$$

Im Spezialfall $x = (x_1, \ldots, x_r) \in (\mathbb{R}^d)^r$ und $y = h_w(x) = (\langle w, x_1\rangle, \ldots, \langle w, x_r\rangle)$ und mit der Definition $\psi(x, v) = \sum_{i=1}^{r} v_i x_i$ ergibt sich aus (8) die Gleichung

$$\overline{h_w(x)} = \text{argmax}_{v\in V} \sum_{i=1}^{r} v_i \langle w, x_i\rangle = \text{argmax}_{v\in V} \left\langle w, \sum_{i=1}^{r} v_i x_i \right\rangle = \text{argmax}_{v\in V} \langle w, \psi(x, v)\rangle \ . \tag{9}$$

Für eine beliebige Kostenfunktion Δ mit $\Delta(y', y) = \Delta(\overline{y'}, y)$ ergibt sich folgende Rechnung:

$$\begin{aligned}
\Delta(h_w(x), y) &= \Delta(\overline{h_w(x)}, y) \\
&\overset{(9)}{\leq} \Delta(\overline{h_w(x)}, y) + \left\langle w, \psi\left(x, \overline{h_w(x)}\right)\right\rangle - \langle w, \psi(x, \bar{y})\rangle \\
&\leq \max_{v\in V} \left(\Delta(v, y) + \langle w, \psi(x, v)\rangle - \langle w, \psi(x, \bar{y})\rangle\right) \\
&\overset{*}{=} \max_{v\in V} \left(\Delta(v, y) + \sum_{i=1}^{r} (v_i - \bar{y}_i)\langle w, x_i\rangle\right) =: \Delta'(h_w(x), y)
\end{aligned}$$

[7]Das wäre eine gute Übungsaufgabe.

In der mit "$*$" markierten Gleichung wurde einfach die Definition von $\psi(x, v)$ bzw. $\psi(x, \bar{y})$ expandiert. $\Delta'(h_w(x), y)$ ist die gesuchte in w konvexe obere Schranke für die NDCG Kostenfunktion.

Wir stellen uns in diesem Abschnitt abschließend die Aufgabe, die Funktion Δ', die sich als konvexe obere Schranke der NDCG Kostenfunktion ergeben hat, zu berechnen. Dies führt uns zu folgendem Optimierungsproblem. Zu gegebenen Parametern x, y, w finde

$$v^* = \text{argmax}_{v \in V} \left(\Delta(v, y) + \sum_{i=1}^{r} (v_i - \overline{y_i}) \langle w, x_i \rangle \right) \quad . \tag{10}$$

Ein Durchmustern aller $v \in V = V_r$ (exhaustive search) wäre wegen $|V_r| = r!$ hochgradig ineffizient. Wir werden das durch (10) gegebene Optimierungsproblem effizient auf das sogenannte „Zuweisungsproblem" (assignment problem) reduzieren. Für letzteres existieren effiziente Algorithmen. Details folgen.

Aus der Gleichung (7) für die NDCG Kostenfunktion $\Delta(y', y)$ ergibt sich im Spezialfall $y = v \in V$ wegen $\bar{v} = v$ die folgende Gleichung:

$$\Delta(v, y) = \frac{1}{G(y, y)} \sum_{i=1}^{r} (D(\overline{y_i}) - D(v_i)) y_i \quad .$$

Wenn wir dies in (10) einsetzen und ausnutzen, dass die Terme $\overline{y_i}$ und $D(\overline{y_i})$ nur einen konstanten (von v unabhängigen) Beitrag zur Zielfunktion leisten, erhalten wir folgendes:

$$v^* = \text{argmax}_{v \in V} \sum_{i=1}^{r} \left(v_i \langle w, x_i \rangle - \frac{D(v_i) y_i}{G(y, y)} \right) = \text{argmin}_{v \in V} \sum_{i=1}^{r} \left(\frac{D(v_i) y_i}{G(y, y)} - v_i \langle w, x_i \rangle \right) \quad . \tag{11}$$

Es bezeichne $K_{r,r}$ den vollständigen bipartiten Graphen mit den zwei Knotenmengen $A = \{a_1, \ldots, a_r\}$ und $B = \{b_1, \ldots, b_r\}$ und den r^2 Kanten $\{a_i, b_j\}$ mit Kantenkosten $c_{i,j}$ für $1 \leq i, j \leq r$. Wir können uns $v \in V$ vorstellen als ein perfektes Heiratssystem (perfect matching), wobei a_i mit b_{v_i} „verheiratet" wird.[8] Die Gesamtkosten des Heiratssystems v betragen dann

$$C(v) = \sum_{i=1}^{r} c_{i, v_i} \quad .$$

Setzen wir

$$c_{i,j} = \frac{D(j) y_i}{G(y, y)} - j \langle w, x_i \rangle \quad ,$$

so können wir das Minimierungsproblem in (11) interpretieren als das Problem, im bipartiten Graphen $K_{r,r}$ mit Kantenkosten $(c_{i,j})_{1 \leq i,j \leq r}$ ein perfektes Heiratssystem v mit minimalen Gesamtkosten zu bestimmen. Im Englischen spricht man von einem „Minimum Weight Perfect Matching" oder auch von einem „Assignment" Problem, im Deutschen von einem

[8] Die Definition von $V = V_r$ impliziert, dass jeder der $2r$ Knoten über die „Heiratsvorschrift" v exakt einen Partner erhält.

Zuweisungsproblem. Der schnellste bekannte Algorithmus reduziert das Zuweisungsproblem auf das sogenannte „Minimum Cost Flow“ Problem und verwendet für letzteres den sogenannten „Successive Shortest Path“ Algorithmus. Die Darstellung dieses Verfahrens würde den Rahmen unserer Vorlesung sprengen. Wir behandeln aber im nächsten Abschnitt eine Reduktion des Zuweisungsproblems auf „Lineare Programmierung“.

17.4.3 Reduktion des Zuweisungsproblems auf Lineare Programmierung

Eine Matrix $B \in \{0,1\}^{r\times r}$ heißt *Permutationsmatrix*, wenn in jeder Zeile und jeder Spalte exakt ein Eintrag den Wert 1 hat. Eine Matrix $B \in [0,1]^{r\times r}$ heißt *doppelt stochastisch*, wenn jede Zeilensumme und jede Spaltensumme in B exakt den Wert 1 liefert. Wir zitieren ohne Beweis das folgende Resultat:

Theorem 17.7 (Birkhoff, 1946) *Jede doppelt stochastische Matrix lässt sich als konvexe Kombination von Permutationsmatrizen darstellen.*

Jeder Rangnummernvektor $v \in V_r$ kann mit einer Permutationsmatrix B vermöge der Vorschrift

$$\forall 1 \leq i,j \leq r : b_{i,j} = 1 \Leftrightarrow j = v_i$$

identifiziert werden. Die Elemente aus V_r und die $(r \times r)$-Permutationsmatrizen entsprechen sich 1–zu–1. Offensichtlich entspricht das Zuweisungsproblem für den Graphen $K_{r,r}$ mit Kantenkosten $(c_{i,j})$ dem folgenden Optimierungsproblem. Finde eine $(r \times r)$-Matrix B, die die Kostenfunktion

$$C(B) = \sum_{i,j=1}^{r} c_{i,j} b_{i,j}$$

minimiert und dabei folgende Randbedingungen einhält:

1. Für alle $i \in [r]$ gilt $\sum_{j=1}^{r} b_{i,j} = 1$.
2. Für alle $j \in [r]$ gilt $\sum_{i=1}^{r} b_{i,j} = 1$.
3. Für alle $1 \leq i,j \leq r$ gilt $B_{i,j} \in \{0,1\}$.

Die erste (bzw. zweite) Randbedingung kontrolliert, dass alle Zeilensummen (bzw. Spaltensummen) den Wert 1 ergeben. Die ersten beiden Bedingungen erzwingen demnach gerade, dass B eine doppelt stochastische Matrix ist. Zusammen mit der dritten Bedingung wird dann erzwungen, dass B eine Permutationsmatrix ist. Ohne die dritte Randbedingung hätten wir ein lineares Programmierungsproblem vorliegen. Wir können das Zuweisungsproblem nun lösen wie folgt:

- Finde eine optimale Basislösung zur Minimierung von $C(B)$ unter den ersten beiden Randbedingungen (zum Beispiel mit dem Simplexverfahren).

Eine optimale Basislösung entspricht nämlich den Ecken des Polytops, das durch die linearen Randbedingungen gegeben ist. In unserem konkreten Problem entsprechen die Ecken des Polytops gerade den Permutationsmatrizen.[9] Mit anderen Worten: wenn wir nicht irgendeine optimale Lösung berechnen, sondern eine optimale *Basislösung*, dann wird die dritte (und problematische) Randbedingung automatisch eingehalten!

17.5 Das bipartite Rangordnungsproblem

In diesem Abschnitt betrachten wir Rangordnungsprobleme mit markierten Beispielen der Form $(x, y) \in \mathcal{X}^r \times \{-1, 1\}^r$, d.h., die Instanzen in x werden entweder mit 1 oder mit -1 bewertet. Rangordnungsprobleme mit dieser Einschränkung werden *bipartit* genannt.

Intuitiv können wir die Bewertung „1“ (bzw. „-1“) als Indikator für „Relevanz“ (bzw. „Irrelevanz“) auffassen. Das Erlernen einer bipartiten Rangordnung sieht auf den ersten Blick aus wie ein binäres Klassifikationsproblem. Während aber die 0–1 Kostenfunktion eine gute Wahl für binäre Klassifikationsprobleme ist, ist sie i.A. ungeeignet, um die bei binären Rangordnungsproblemen vorliegende Zielsetzung zu erfassen.

Beispiel 17.8 (Aufdeckung von Betrug) *Bei der Überwachung von finanziellen online-Transaktionen werden betrügerische Aktivitäten i.A. nur einen kleinen Bruchteil ausmachen, sagen wir* $1/1000$. *Die konstante* -1*-Hypothese, die jede Transaktion als „kein Betrug“ bewertet, hätte dann die Fehlerrate* $1/1000$. *Dies ist für ein binäres Klassifikationsproblem kein schlechtes Ergebnis! Jedoch würde eine solche Hypothese* keinen *Beitrag zum Aufspüren betrügerischer Transaktionen leisten.*

17.5.1 Kostenfunktionen

Wir wollen nun erarbeiten, welche Kostenfunktionen statt der 0–1 Kosten verwendet werden sollten. Wenn wir $y \in \{-1, 1\}^r$ als den Vektor mit den korrekten ± 1-Werten ansehen und $y' = h(x) \in \mathbb{R}^r$ als die von einer Hypothese h vorgenommenen Bewertungen, dann können folgende Fälle auftreten:

Korrekte Positivbewertungen (true positives): $y_i = 1$ und $y'_i = 1$. Wir setzen

$$a = |\{i \in [r] : y_i = 1 \wedge \operatorname{sign}(y'_i) = 1\}| \ .$$

Falsche Positivbewertungen (false positives): $y_i = -1$ und $y'_i = 1$. Wir setzen

$$b = |\{i \in [r] : y_i = -1 \wedge \operatorname{sign}(y'_i) = 1\}| \ .$$

Falsche Negativbewertungen (false negatives): $y_i = 1$ und $y'_i = -1$. Wir setzen

$$c = |\{i \in [r] : y_i = 1 \wedge \operatorname{sign}(y'_i) = -1\}| \ .$$

[9] Der Beweis dieser Behauptungen soll hier nicht geführt werden, da wir im Rahmen dieser Vorlesung nicht tiefer in die Grundlagen zur Theorie der linearen Programmierung einsteigen wollen.

Korrekte Negativbewertungen (true negatives): $y_i = -1$ und $y'_i = -1$. Wir setzen

$$d = |\{i \in [r] : y_i = -1 \wedge \text{sign}(y'_i) = -1\}| \ .$$

Definition 17.9 *Es seien a, b, c, d die Zählparameter für die oben genannnten vier Fälle, die sich beim Abgleich von $y' \in \mathbb{R}^r$ mit $y \in \{-1, 1\}^r$ ergeben. Dann heißt der Quotient $\frac{a}{a+c}$ (bzw. der Quotient $\frac{a}{a+b}$) die* Sensitivität (sensitivity) *(bzw. die* Präzision (precision)*) von y' bezüglich y. Der Quotient $\frac{d}{d+b}$ heißt im Englischen „specificity". Wir nennen im folgenden $1 - \frac{d}{d+b} = \frac{b}{d+b}$ die* Hypersensitivität.

Erwünscht sind Hypothesen h, die auf Beispielen (x, y) Voraussagen $y' = h(x)$ treffen, welche bezüglich y hohe Präzisions- und Sensitivitätswerte und niedrige Hypersensitivitätswerte erzielen.

Beispiel 17.10 (Aufdeckung von Betrug — fortgesetzt) *Nehmen wir an, dass die* 1*-Einträge in y Betrugsfälle markieren. Dann liefert $\frac{a}{a+c}$ den Bruchteil der Betrugsfälle, welche durch entsprechende* 1*-Einträge in $y' = h(x)$ erfolgreich angezeigt werden. Die Hypothese h muss in diesem Sinne* sensitiv *auf relevante Instanzen (hier: Betrugsfälle) reagieren. Eine Sensitivität von* 1 *lässt sich auf triviale Weise erzielen, indem y' als die konstante* 1*-Hypothese gewählt wird, die jedes Mal auf „Betrug" wettet. In diesem Fall wäre die Präzision $\frac{a}{a+b}$ sehr gering (insbesondere wenn Betrugsfälle relativ selten vorkommen). Ebenso wäre die Hypersensitivität gleich* 1*, da wir auch bei jedem Nicht-Betrugsfall auf „Betrug" wetten.*

Wir führen die folgende Kurzschreibweise ein:

$$\text{sensitivity} = \frac{a}{a+c} \ , \ \text{precision} = \frac{a}{a+b} \text{ und specificity} = \frac{d}{d+b} \ .$$

Wir behalten dabei im Kopf, dass die Parameter a, b, c, d Funktionen in y' und y sind. Die Parameter sensitivity, precision und specificity sind Funktionen in a, b, c, d (und können daher indirekt als Funktionen in y', y aufgefasst werden).

Wir betrachten Kostenfunktionen Δ von der Form $\Delta(y', y) = 1 - F$ wobei F durch eine Formel in den Variablen sensitivity, precision und specificity (oder alternativ als Formel in den Variablen a, b, c, d) ausgedrückt werden kann. Wir listen ein paar populäre Wahlen von F auf:

1. Setze F gleich dem arithmetischen Mittel von sensitivity und specificity:

$$\bar{F} = \frac{1}{2}(\text{sensitivity} + \text{specificity}) = \frac{1}{2}\left(\frac{a}{a+c} + \frac{d}{d+b}\right) \ .$$

2. Setze F_1 gleich dem harmonischen Mittelwert von sensitivity und precision:

$$F_1 = \frac{2}{\text{sensitivity}^{-1} + \text{precision}^{-1}} = \frac{2a}{2a+b+c} \ .$$

3. Es sei $\beta > 0$. Wir erhalten eine Verallgemeinerung von F_1, der Beitrag von sensitivity^{-1} zum harmonischen Mittel mit β^2 gewichtet wird:

$$F_\beta = \frac{1+\beta^2}{\beta^2 \cdot \text{sensitivity}^{-1} + \text{precision}^{-1}} = \frac{(1+\beta^2)a}{(1+\beta^2)a + b + \beta^2 c} \ .$$

17.5.2 Lineare Voraussagefunktionen und konvexe Surrogat-Zielfunktionen

Wir setzen wieder $\mathcal{X} = \mathbb{R}^d$ voraus. Für $w \in \mathbb{R}^d$, $\theta \in \mathbb{R}$ und $x = (x_1, \ldots, x_r) \in (\mathbb{R}^d)^r$ definieren wir

$$h_{w,\theta}(x) = (\langle w, x_1\rangle + \theta, \ldots, \langle w, x_r\rangle + \theta) \ .$$

Statt $h_{w,0}$ schreiben wir einfach h_w (und diese Definition von h_w deckt sich mit der in Abschnitt 17.4.2 verwendeten Definition). Die Funktion $\vec{b}_\theta : \mathbb{R}^* \to \{-1,1\}^*$ ist definiert wie folgt:

$$\vec{b}_\theta(a_1, \ldots, a_r) = (\text{sign}(a_1 + \theta), \ldots, \text{sign}(a_r + \theta)) \ .$$

Statt $\vec{b}_0$ schreiben wir einfach $\vec{b}$. Es gilt

$$\vec{b}_\theta(h_w(x)) = \vec{b}_\theta(\langle w, x_1\rangle, \ldots, \langle w, x_r\rangle) = (\text{sign}(\langle w, x_1\rangle + \theta), \ldots, \text{sign}(\langle w, x_r\rangle + \theta)) \ .$$

Somit liefert $\vec{b}_\theta(h_w(x))$ die von $h_{w,\theta}$ induzierten ± 1-Bewertungen der Instanzen $x_1, \ldots, x_r$.

Bei Verwendung der im vorigen Abschnitt genannten Kostenfunktionen wird die Klasse der linearen Voraussagefunktionen meistens durch die Festlegung $\theta = 0$ eingeschränkt. Die Verwendung der zusätzlichen Variable θ kann aber sinnvoll sein in folgenden Fällen:

- Die Hypothese $\vec{b}_\theta \circ h_w$ soll höchstens k Instanzen in $x \in (\mathbb{R}^d)^r$ mit 1 bewerten. Dann kann nach Ermittlung eines Gewichtsvektors w der Parameter θ solange verkleinert werden, bis höchstens k innere Produkte $\langle w, x_i\rangle$ oberhalb der Schwelle $-\theta$ liegen.

- Die Hypothese $\vec{b}_\theta \circ h_w$ soll höchstens k Instanzen in $x \in (\mathbb{R}^d)^r$ mit -1 bewerten. Dann kann nach Ermittlung eines Gewichtsvektors w der Parameter θ solange vergrößert werden, bis höchstens k innere Produkte $\langle w, x_i\rangle$ unterhalb der Schwelle $-\theta$ liegen.

Eine Beschränkung auf höchstens k viele 1-Bewertungen kann zum Beispiel sinnvoll sein, wenn jeder 1-Eintrag einer aufwändigen Nachbearbeitung bedarf (wie es zum Beispiel bei verdächtigen finanziellen online-Transaktionen der Fall wäre). Eine analoge Bemerkung gilt für die Beschränkung der Anzahl der -1-Bewertungen.

Aus Effizienzgründen bietet es sich wieder an die eigentliche Kostenfunktion $\Delta(h_{w,\theta}(x), y)$ durch eine in w konvexe obere Schranke $\Delta'(h_{w,\theta}(x), y)$ zu ersetzen. Wir setzen dabei voraus, dass Δ von $h_{w,\theta}(x)$ nur indirekt über $\vec{b}_\theta(h_w(x))$ abhängt, d.h., es gilt die Gleichung

$$\Delta(h_{w,\theta}(x), y) = \Delta(\vec{b}_\theta(h_w(x)), y) \ .$$

Zur Herleitung einer konvexen oberen Schranke für die Kostenfunktion Δ benötigen wir die folgende Formel für $\vec{b}_\theta(h_w(x))$:

Übungsaufgabe. Es sei $V = \{-1, 1\}^r$ und $V_{\leq k}$ (bzw. $V_{\geq k}$) sei die Menge der Vektoren aus V mit höchstens (bzw. mindestens) k 1-Komponenten. Weise nach, dass folgendes gilt:

1. $\vec{b}(h_w(x)) = \text{argmax}_{v \in V} \sum_{i=1}^{r} v_i \langle w, x_i \rangle$.
2. Wenn θ so gewählt werden muss, dass $\vec{b}_\theta(h_w(x))$ höchstens k 1-Komponenten hat, dann gilt $\vec{b}_\theta(h_w(x)) = \text{argmax}_{v \in V_{\leq k}} \sum_{i=1}^{r} v_i \langle w, x_i \rangle$.
3. Wenn θ so gewählt werden muss, dass $\vec{b}_\theta(h_w(x))$ mindestens k 1-Komponenten hat, dann gilt $\vec{b}_\theta(h_w(x)) = \text{argmax}_{v \in V_{\geq k}} \sum_{i=1}^{r} v_i \langle w, x_i \rangle$.

Wir werden daher voraussetzen, dass bei einer geeigneten Wahl von $V' \subseteq V$ die Formel

$$\vec{b}_\theta(h_w(x)) = \text{argmax}_{v \in V'} \sum_{i=1}^{r} v_i \langle w, x_i \rangle \tag{12}$$

gültig ist. Die folgende Rechnung liefert als Ergebnis die gesuchte in w konvexe obere Schranke Δ' von Δ:

$$\begin{aligned} \Delta(h_{w,\theta}(x), y) &= \Delta(\vec{b}_\theta(h_w(x)), y) \\ &\overset{(12)}{\leq} \Delta(\vec{b}_\theta(h_w(x)), y) + \left(\sum_{i=1}^{r} (\bar{b}_\theta(h_w(x))_i - y_i) \langle w, x_i \rangle \right) \\ &\leq \max_{v \in V'} \left(\Delta(v, y) + \sum_{i=1}^{r} (v_i - y_i) \langle w, x_i \rangle \right) =: \Delta'(h_{w,\theta}(x), y) \ . \end{aligned}$$

Wir stellen uns in diesem Abschnitt abschließend die Aufgabe, die Funktion Δ', die sich als konvexe obere Schranke der Kostenfunktion Δ ergeben hat, zu berechnen. Zu diesem Zweck machen wir zwei Voraussetzungen:

1. Der Wert der Kostenfunktion ist durch die Parameter sensitivity, precision und specificity vollständig bestimmt.
2. Es bezeichne $P = P(y)$ (bzw. $N = N(y)$) die Anzahl der Komponenten $i \in [r]$ von y mit $y_i = 1$ (bzw. $y_i = -1$). Für $0 \leq a \leq P$ und $0 \leq b \leq N$ bezeichne $V_y[a, b]$ die Menge aller Vektoren aus $V = \{-1, 1\}^r$ mit genau a 1-Komponenten im Bereich $\{i \in [r] : y_i = 1\}$ und genau $b \leq N$ 1-Komponenten im Bereich $\{i \in [r] : y_i = -1\}$. Wir setzen voraus, dass V' sich mit einer geeigneten Auswahl der Paare (a, b) darstellen lässt als eine Vereinigung von Mengen der Form $V_y[a, b]$.

Die erste Voraussetzung ist für alle von uns im Abschnitt 17.5.1 genannten Kostenfunktionen erfüllt. Die zweite Voraussetzung ist zum Beispiel erfüllt für die wichtigen Fälle $V' \in \{V, V_{\leq k}, V_{\geq k}\}$. Für die Menge $V = \{-1, 1\}^r$ ist das offensichtlich. $V_{\leq k}$ lässt sich (für jede mögliche Wahl von y) darstellen als

$$V_{\leq k} = \bigcup_{(a,b): a+b \leq k, 0 \leq a \leq P(y), 0 \leq b \leq N(y)} V_y[a, b] \ .$$

Eine analoge Darstellung gilt für $V_{\geq k}$.

Widmen wir uns jetzt dem Optimierungsproblem, das sich bei der Auswertung der Kostenfunktion Δ' stellt: Zu gegebenen Parametern x, y, w finde

$$v^* = \text{argmax}_{v \in V'} \left(\Delta(v, y) + \sum_{i=1}^{r} (v_i - y_i) \langle w, x_i \rangle \right) = \text{argmax}_{v \in V'} \left(\Delta(v, y) + \sum_{i=1}^{r} v_i \langle w, x_i \rangle \right) \quad . \tag{13}$$

Hier sind die entscheidenden Beobachtungen:

- Da V' sich als Vereinigung von Mengen der Form $V_y[a, b]$ darstellen lässt, können wir für jede Wahl von a, b den besten Vektor $v^*_{a,b}$ in $V_y[a, b]$ ermitteln.
- Für eine feste Wahl von a und b gilt (unabhängig von der Auswahl von $v \in V_y[a, b]$)

  $$\text{sensitivity} = \frac{a}{P} \ , \ \text{precision} = \frac{a}{a+b} \text{ und specificity} = \frac{b}{N} \quad .$$

  Daher liefert der Kostenterm $\Delta(v, y)$ in (13) für alle Wahlen von $v \in V_y[a, b]$ den gleichen Beitrag. Dadurch wird das Optimierungsproblem runtergekocht auf

  $$v^*_{a,b} = \text{argmax}_{v \in V_y[a,b]} \left(\sum_{i=1}^{r} v_i \langle w, x_i \rangle \right) \quad .$$

- Die Berechnung von $v^*_{a,b}$ ist einfach:
  - Sortiere die Paare $(i, \langle w, x_i \rangle)$ absteigend nach der zweiten Komponente.
  - Ermittle daraus die Folge $i_1, \ldots, i_P$ der Indizes i mit $y_i = 1$, so dass $\langle w, y_{i_k} \rangle \geq \langle w, y_{i_{k+1}} \rangle$ für $k = 1, \ldots, P-1$, und ebenso die Folge $j_1, \ldots, j_N$ der Indizes j mit $y_j = -1$, so dass $\langle w, y_{j_k} \rangle \geq \langle w, y_{j_{k+1}} \rangle$ für $k = 1, \ldots, N-1$.
  - Verteile die a Einsen von v im Bereich $\{i \in [r] : y_i = 1\}$ auf die Komponenten $i_1, \ldots, i_a$; verteile die b Einsen von v im Bereich $\{i \in [r] : y_i = -1\}$ auf die Komponenten $j_1, \ldots, j_b$. Setze die restlichen Komponenten von v auf den Wert -1.
- Aus der Kollektion der Vektoren $v^*_{a,b}$ wählen wir den „Champion“ v^* aus, also denjenigen unter den Vektoren $v^*_{a,b}$, dessen Wert bei der Zielfunktion $\Delta(v, y) + \sum_{i=1}^{r} v_i \langle w, x_i \rangle$ am größten ist.

Weitere Implementierungsdetails sind leicht einzufüllen. Es ist offensichtlich, dass dieser Algorithmus einen besten Vektor v^* des Optimierungsproblems (13) ausgibt.