Von überwachten zu unüberwachten Support-Vektor-Maschinen und Anwendungen in der Astronomie*

Fabian Gieseke

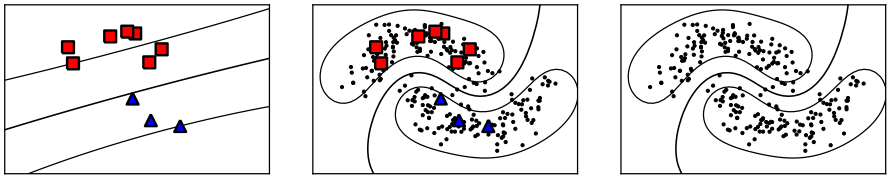
Department für Informatik
Carl von Ossietzky Universität Oldenburg
f.gieseke@uni-oldenburg.de

Abstract: Ein bekanntes Problem des maschinellen Lernens ist die Klassifikation von Objekten. Entsprechende Modelle basieren dabei meist auf Trainingsdaten, welche aus Mustern mit zugehörigen Labeln bestehen. Die Erstellung eines hinreichend großen Datensatzes kann sich für gewisse Anwendungsfälle jedoch als sehr kosten- oder zeitintensiv erweisen. Eine aktuelle Forschungsrichtung des maschinellen Lernens zielt auf die Verwendung von (zusätzlichen) ungelabelten Mustern ab, welche oft ohne großen Aufwand gewonnen werden können. In diesem Beitrag wird die Erweiterung von sogenannten Support-Vektor-Maschinen auf solche Lernszenarien beschrieben. Im Gegensatz zu Support-Vektor-Maschinen führen diese Varianten jedoch zu kombinatorischen Optimierungsproblemen. Die Entwicklung effizienter Optimierungsstrategien ist daher ein erstrebenswertes Ziel und soll im Rahmen dieses Beitrags diskutiert werden. Weiterhin werden mögliche Anwendungsgebiete der entsprechenden Verfahren erläutert, welche sich unter anderem im Bereich der Astronomie wiederfinden.

1 Einleitung

Das Gebiet des *maschinellen Lernens* [HTF09] hat innerhalb der letzten Jahren stark an Bedeutung gewonnen. Einer der Gründe hierfür ist die Tatsache, dass die Datenvolumina in den verschiedensten Bereichen dramatisch zugenommen haben. Dies ist beispielsweise in der Astronomie der Fall, wo aktuelle und zukünftige Projekte Daten im Tera- und Petabytebereich liefern und liefern werden [ITA+11, YAAA00]. Die Analyse dieser Datenmengen „per Hand“ ist im Allgemeinen nicht mehr möglich und erfordert maschinelle Lernverfahren zur (halb-)automatisierten Wissensextraktion. Die Klassifikation von Objekten zählt mit zu den wichtigsten Problemen des maschinellen Lernens. Als Ausgangspunkt stehen hier im Allgemeinen Trainingsdaten in Form von *Mustern* und zugehörigen *Labeln* zur Verfügung. Das Ziel des Lernprozesses besteht darin, auf Basis der bekannten Trainingsdaten entsprechende Modelle zu erstellen, welche bisher unbekannte Muster klassifizieren können. Diese Form wird als *überwachtes Lernen* [HTF09] bezeichnet.

*Der vorliegende Text stellt eine Kurzfassung meiner Dissertation [Gie11] mit dem Originaltitel *„From Supervised to Unsupervised Support Vector Machines and Applications in Astronomy“* dar.



(a) Überwacht (b) Halbüberwacht (c) Unüberwacht

Abbildung 1: Das Ziel einer SVM besteht darin, auf Basis gelabelter Muster (rote Quadrate und blaue Dreiecke) diejenige Hyperebene zu finden (mittlere Linie), welche den Abstand zu den Mustern maximiert, vgl. Abb. (a). Oft sind gelabelte Muster jedoch nur in geringem Umfang vorhanden; ungelabelte Muster (schwarze Punkte) können hingegen meist leicht erhalten werden. Halbüberwachte SVM können auf Basis dieser zusätzlichen Informationen zu besseren Modellen führen, vgl. Abb. (b). Sind keine gelabelten Muster vorhanden, so kann zwar die „Struktur“ der Daten erfasst werden, die Klassenzugehörigkeit ist allerdings nicht mehr ersichtlich, vgl. Abb. (c).

Das Konzept einer *Support-Vektor-Maschine* (SVM) [SS01, SC08] zählt mit zu den erfolgreichsten Methoden im Bereich des maschinellen Lernens. Sind hinreichend viele Trainingsdaten vorhanden, so weisen die zugehörigen Modelle oft eine exzellente Klassifikationsgüte auf; die Erstellung eines solchen Trainingsdatensatzes erweist sich in vielen Anwendungsfällen jedoch oft als sehr aufwendig. Im Gegensatz zu gelabelten Trainingsdaten können ungelabelte Muster meist ohne großen Aufwand gewonnen werden. Um auch letztere in der Trainingsphase zu verwenden, wurden in der Literatur unter anderem *halb-* und *unüberwachte Support-Vektor-Maschinen* vorgestellt [BD99, Joa99, VS77, XNLS05]. In vielen Fällen ermöglichen diese Erweiterungen die Generierung von Modellen mit einer deutlich besseren Güte. Die konkrete Anwendung dieser Erweiterungen auf reale Fragestellungen wird jedoch dadurch erschwert, dass hierzu *kombinatorische Optimierungsprobleme* citeBoydV2004 gelöst bzw. möglichst gut approximiert werden müssen.

Im Folgenden werden Support-Vektor-Maschinen und deren Erweiterungen sowie Optimierungsansätze skizziert. Zudem werden Anwendungsgebiete der Verfahren beschrieben.

2 Support-Vektor-Maschinen

In ihrer ursprünglichen Form zielen Support-Vektor-Maschinen auf die *binäre Klassifikation* ab, welche in diesem Abschnitt näher erläutert werden soll.

2.1 Überwachte Lernszenarien

Für solche Lernszenarien stehen Muster zweier Klassen zur Verfügung, welche durch einen *Trainingsdatensatz* $T_L = \{(\mathbf{x}'_1, y'_1), \ldots, (\mathbf{x}'_l, y'_l)\} \subset \mathbb{R}^d \times \{-1, +1\}$ repräsentiert werden können. Das Ziel einer Support-Vektor-Maschine besteht darin, diejenige Hyperebene zu finden, welche die Muster beider Klassen am besten „trennt“, und zwar so, dass

die induzierte Distanz zwischen der Hyperebene und den Mustern maximal ist. Weiterhin sollen nur wenige Muster in dem induzierten Korridor liegen. Aus mathematischer Sicht führt dies zu einem effizient lösbaren *quadratischen Programm* [BV04] der Form

$$\begin{aligned} \underset{\mathbf{w}\in\mathbb{R}^d,\, b\in\mathbb{R},\, \boldsymbol{\xi}'\in\mathbb{R}^l}{\text{minimiere}} \; & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{l}\xi_i' \\ \text{s.d. } & y_i'(\langle \mathbf{w}, \mathbf{x}_i'\rangle + b) \geq 1 - \xi_i',\ \xi_i' \geq 0, \end{aligned} \tag{1}$$

wobei der erste Term mit der Maximierung des Abstandes und der zweite Term mit der „Bestrafung“ von Mustern im Korridor korrespondiert [SS01, SC08]. Der Parameter $C > 0$ bestimmt hierbei den Kompromiss zwischen diesen beiden Teilzielen.

Mit Hilfe von *Kernfunktionen* $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ kann dieses Konzept auf nicht-lineare Hyperebenen erweitert werden [SS01, SC08]. In Abb. 1 (a) ist eine solche nicht-lineare Support-Vektor-Maschine dargestellt. Hierbei stellen die roten Quadrate und die blauen Dreiecke die gelabelten Muster und die mittlere Linie die nicht-lineare Hyperebene dar. Der Rand ist durch die beiden äußeren Linien skizziert. Neue, bisher nicht betrachtete Muster können nun mittels dieses Modells in die beiden Klassen eingeteilt werden.

2.2 Halbüberwachte Lernszenarien

Support-Vektor-Maschinen zählen zu den überwachten Lernverfahren [HTF09], da nur auf gelabelte Muster zurückgegriffen wird. Die Verwendung von gelabelten und ungelabelten Mustern $T_U = \{\mathbf{x}_1, \ldots, \mathbf{x}_u\} \subset \mathbb{R}^d$ führt zu den *halbüberwachten Support-Vektor-Maschinen* [BD99, Joa99, VS77]. Ziel dieser Variante ist, diejenige Partition der ungelabelten Muster in zwei Klassen zu finden, so dass eine anschließende Anwendung einer (modifizierten) Support-Vektor-Maschine zu dem bestmöglichen Ergebnis führt, siehe Abb. 1 (b). Ähnlich zu dem überwachten Fall weist die Hyperebene einen großen Abstand zu den gelabelten Mustern auf, jedoch verläuft diese *nicht* durch die von den ungelabelten Mustern (schwarze Punkte) erzeugten Überdichten. In diesem Fall bieten die ungelabelten Muster also wertvolle zusätzliche Informationen über die „Struktur“ der Daten. Es ist direkt ersichtlich, dass das finale Modell eine bessere Qualität aufweist.

2.3 Unüberwachte Lernszenarien

Die sogenannten *unüberwachten Support-Vektor-Maschinen* greifen nur auf eine Menge $T_U = \{\mathbf{x}_1, \ldots, \mathbf{x}_u\} \subset \mathbb{R}^d$ ungelabelter Muster zurück, beziehen also keine gelabelten Muster ein. Das Lernziel dieser Variante besteht darin, diejenige Partition der ungelabelten Muster in zwei Klassen zu finden, so dass eine anschließende Anwendung einer Support-Vektor-Maschine zu dem optimalen Ergebnis führt, siehe Abb. 1 (c). Es wird also ein zu dem halbüberwachten Fall sehr ähnliches Ziel verfolgt. Ursprünglich wurde diese Modifikation unter dem Namen *Maximum Margin Clustering* [XNLS05] vorgestellt.

Algorithmus 1 Exakte Suche in polynomieller Zeit

1: Konstruiere den *Graphen* $\mathcal{G}$ (bestehend aus $\mathcal{O}(u^d)$ Knoten)
2: **for** jeden Knoten v in $\mathcal{G}$ **do**
3: Sei $\mathbf{y} \in \{-1, +1\}^u$ der zugehörige Partitionsvektor.
4: Teste $\mathbf{y}$ mittels Anwendung einer Support-Vektor-Maschine (in $\mathcal{O}(u^3)$ Zeit).
5: Merke Ergebnis, falls besser als vorher.
6: **end for**

3 Kombinatorische und nicht-konvexe Optimierung

Beide oben skizzierten Erweiterungen führen zu kombinatorischen Optimierungsproblemen, da die optimale Partition der ungelabelten Muster im Vorfeld nicht bekannt ist. Aus mathematischer Sicht erhält man für den unüberwachten Fall beispielsweise ein Optimierungsproblem der Gestalt

$$\begin{aligned} \underset{\substack{\mathbf{y}\in\{-1,+1\}^u,\\ \mathbf{w}\in\mathbb{R}^d,\, b\in\mathbb{R},\, \boldsymbol{\xi}\in\mathbb{R}^u}}{\text{minimiere}} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{u}\xi_i \\ \text{s.d.} \quad & y_i(\langle \mathbf{w}, \mathbf{x}_i\rangle + b) \geq 1 - \xi_i,\ \xi_i \geq 0, \end{aligned} \tag{2}$$

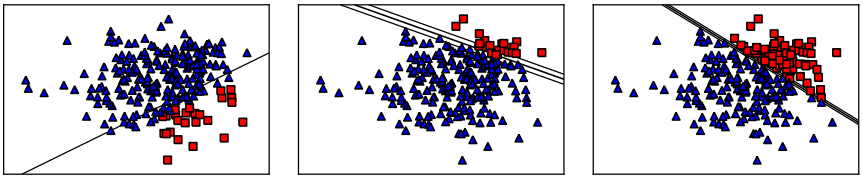
Eine mögliche Partition ergibt sich durch Zuordnung aller Muster zu genau einer Klasse. Diese unerwünschte Lösung kann durch eine Hinzunahme einer weiteren Nebenbedingung der Form $\left|\frac{1}{u}\sum_{i=1}^{u}\max(0, y_i) - b_c\right| < \varepsilon$ mit entsprechenden Parametern $b_c \in [0, 1]$ und $\varepsilon > 0$ verhindert werden.

Die Suche nach einer geeigneten Belegung des Vektors $\mathbf{y}$ stellt die wesentliche Herausforderung im Kontext des obigen Optimierungsproblems dar. Da es sich bei dieser Erweiterung um ein gemischt-ganzzahliges Optimierungsproblem [Flo95] handelt, kann zur Lösung auf Standardverfahren für diese Problemklasse zurückgegriffen werden. Da Probleme dieser Problemklasse jedoch im Allgemeinen NP-schwer sind [Flo95], ist die Bestimmung von Lösungen in polynomieller Zeit nicht sichergestellt. Ein trivialer Ansatz besteht zudem darin, jede mögliche Belegung des Partitionsvektors mittels einer Support-Vektor-Maschine zu „testen“. Diese Herangehensweise ist jedoch aufgrund der exponentiellen Laufzeit nur für sehr kleine Probleminstanzen geeignet. Aus theoretischer Sicht stellt sich deshalb die Frage, ob Lösungsverfahren mit polynomieller Laufzeit existieren.

3.1 Exakte Lösungen in polynomieller Laufzeit

Wie in diesem Abschnitt beschrieben wird, ist dies für den Spezialfall einer linearen Kernfunktion und Mustern aus $\mathbb{R}^d$ möglich. Der entsprechende Lösungsansatz greift dabei auf komplexe Ergebnisse aus dem Bereich der *Algorithmischen Geometrie* [Ede87] zurück.

Theorem 1 *[[Gie11, GVJ12]] Sei $T_U = \{\mathbf{x}_1, \ldots, \mathbf{x}_u\} \subset \mathbb{R}^d$. Dann kann für eine lineare*



(a) $C = 0.001$ (b) $C = 1$ (c) $C = 1000$

Abbildung 2: Unüberwachte Support-Vektor-Maschinen: Exakte Lösungen für eine Menge von Punkten im $\mathbb{R}^2$ und verschiedene Belegungen des Parameters C.

Kernfunktion $k(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{x}_i, \mathbf{x}_j \rangle$ eine exakte Lösung (bis auf Maschinengenauigkeit) für das Optimierungsproblem (2) in $\mathcal{O}(u^{d+3})$ Laufzeit bestimmt werden.

Beweisidee: Die Kernidee besteht darin, eine Bijektion zwischen allen (überhaupt) in Frage kommenden Partitionen der ungelabelten Muster und den Knoten eines speziellen Graphen $\mathcal{G}$ herzustellen. Jede möglich optimale Partition $\mathbf{y}$ korrespondiert also mit genau einem Knoten v von $\mathcal{G}$. Ein bekanntes (nicht-triviales) Ergebnis aus der Algorithmischen Geometrie ist, dass dieser Graph höchstens $\mathcal{O}(u^d)$ Knoten enthalten kann. Dadurch ergibt sich das in Algorithmus 1 dargelegte Verfahren. Die Konstruktion des Graphen benötigt dabei $\mathcal{O}(u^d)$ Laufzeit. Da jede induzierte Partition mittels einer Support-Vektor-Maschine in $\mathcal{O}(u^3)$ Laufzeit getestet werden kann, ergibt sich die obere Laufzeitschranke. □

Das obige Ergebnis ist vor allem aus theoretischer Sicht interessant. Der zugrundeliegende Algorithmus kann jedoch auch zur Generierung von Benchmark-Datensätzen verwendet werden, siehe Abb. 2. Für hochdimensionale Datensätze ist das Verfahren jedoch von wenig praktischer Relevanz. Im Folgenden werden daher zwei Heuristiken entwickelt, welche möglichst gute Lösungen in praktikabler Laufzeit bestimmen können.

3.2 Matrizenbasierte beschleunigte lokale Suche

Aus anwendungstechnischer Sicht ist vor allem der halbüberwachte Fall interessant, der nachfolgend betrachtet werden soll. Diese Variante führt erneut zu einem gemischt-ganzzahligen Optimierungsproblem der Gestalt:

$$\begin{aligned} \underset{\substack{\mathbf{y}\in\{-1,+1\}^u,\\ \mathbf{w}\in\mathbb{R}^d,\, b\in\mathbb{R},\, \boldsymbol{\xi}'\in\mathbb{R}^l,\, \boldsymbol{\xi}\in\mathbb{R}^u}}{\text{minimiere}} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C'\sum_{i=1}^{l}\xi_i' + C\sum_{i=1}^{u}\xi_i \\ \text{s.d.} \quad & y_i'(\langle \mathbf{w}, \mathbf{x}_i' \rangle + b) \geq 1 - \xi_i',\ \xi_i' \geq 0, \\ \text{und} \quad & y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 - \xi_i,\ \xi_i \geq 0, \end{aligned} \tag{3}$$

mit Modellparametern $C' > 0$ und $C > 0$ [VS77]. Einer der ersten Optimierungsansätze wurde von Joachims [Joa99] vorgestellt und ist in Algorithmus 2 skizziert: Nach

Algorithmus 2 Beschleunigte lokale Suche

1: Initialisiere Partitionsvektor $\mathbf{y} \subseteq \{-1, +1\}^n$.
2: **while** nicht konvergiert **do**
3: $\quad \bar{\mathbf{y}} \leftarrow \mathbf{y}$ mit einzelner „vertauschter“ Koordinate
4: $\quad$ **if** $F(\bar{\mathbf{y}}) < F(\mathbf{y})$ **then**
5: $\quad\quad \mathbf{y} \leftarrow \bar{\mathbf{y}}$
6: $\quad$ **end if**
7: **end while**

der Initialisierung des Partitionsvektors $\mathbf{y}$ mit einer sinnvollen Belegung wird diese erste „Schätzung“ iterativ verbessert, indem pro Iteration eine Koordinate vertauscht wird.[1] Hierbei bezeichne $F(\mathbf{y})$ die Zielfunktion (3) für einen fest-gewählten Partitionsvektor $\mathbf{y}$.

Eine direkte Implementation ist jedoch sehr rechenintensiv, da im Allgemeinen eine große Anzahl an Vertauschungen vorgenommen werden muss bis das Verfahren konvergiert (ein einzelner Funktionsaufruf in Schritt 4 benötigt $\mathcal{O}(n^3)$ Laufzeit, wobei $n = l + u$). Die lokale Suchstrategie lässt sich allerdings sehr effizient umsetzen; Kernidee ist hierbei, Varianten der ursprünglichen Support-Vektor-Maschinen zu betrachten, welche auf der quadratischen Verlustfunktion basieren [Rif02]. Dies führt zu einer Zielfunktion der Form

$$F(\bar{\mathbf{y}}) = \bar{\mathbf{y}}^{\mathrm{T}} \mathbf{\Lambda} \underbrace{\left(\mathbf{I} - \overline{\mathbf{K}}\mathbf{G} - \mathbf{G}\overline{\mathbf{K}} + \mathbf{G}\overline{\mathbf{K}}\overline{\mathbf{K}}\mathbf{G} + \lambda \mathbf{G}\overline{\mathbf{K}}\mathbf{G}\right)}_{=:\mathbf{H}} \mathbf{\Lambda}\bar{\mathbf{y}}, \tag{4}$$

wobei $\mathbf{K} \in \mathbb{R}^{n \times n}$ die Kernelmatrix, $\mathbf{\Lambda} \in \mathbb{R}^{n \times n}$ eine Diagonalmatrix mit Einträgen der Form $[\mathbf{\Lambda}]_{i,i} = \sqrt{\frac{1}{l}}$ für $i = 1, \ldots, l$ und $[\mathbf{\Lambda}]_{i,i} = \sqrt{\frac{\gamma}{u}}$ für $i = l+1, \ldots, n$, und $\overline{\mathbf{K}} := \mathbf{\Lambda}\mathbf{K}\mathbf{\Lambda}$ und $\mathbf{G} := \left(\mathbf{\Lambda}\mathbf{K}\mathbf{\Lambda} + \lambda \mathbf{I}\right)^{-1}$ entsprechende Matrizen sein. Diese Zerlegung ermöglicht eine effizientes „Updaten“ der Funktionswerte pro Iteration:

Theorem 2 *([Gie11, GPK09, GKAP12, GKAP11]) Jede Iteration (Schritte 3–6) von Algorithmus 2 kann in $\mathcal{O}(n)$ Laufzeit getätigt werden.*

Beweisidee: Die Kernidee besteht darin, zunächst die Matrix $\mathbf{H}\mathbf{\Lambda}$ in $\mathcal{O}(n^3)$ Zeit zu berechnen und anschließend den Vektor $\mathbf{H}\mathbf{\Lambda}\mathbf{y} \in \mathbb{R}^n$ pro Iteration mittels

$$\mathbf{H}\mathbf{\Lambda}\bar{\mathbf{y}} = \mathbf{H}\mathbf{\Lambda}\mathbf{y} - 2y_j(\mathbf{H}\mathbf{\Lambda})_{[n],\{j\}} \tag{5}$$

zu erneuern, wobei j die vertauschte Koordinate und $(\mathbf{H}\mathbf{\Lambda})_{[n],\{j\}}$ die j-te Spalte von $\mathbf{H}\mathbf{\Lambda}$ sei. Dieses Update kann in linearer Zeit getätigt werden. Anschließend kann $F(\bar{\mathbf{y}})$ ebenfalls in linearer Laufzeit bestimmt werden (Multiplikation zweier Vektoren) [Gie11]. □

Die obige Auswertung der Zielfunktion in linearer Laufzeit ermöglicht somit eine hocheffiziente Umsetzung der lokalen Suche, welche pro Iteration nur $\mathcal{O}(n)$ anstatt $\mathcal{O}(n^3)$ Laufzeit benötigt. Dies ermöglicht eine deutlich bessere Exploration des Suchraums und somit eine bessere Güte der bestimmten Lösung.

[1] Als Schätzung kommt z.B. die durch Anwendung einer auf den gelabelten Mustern trainierten Support-Vektor-Maschine induzierte Partition in Frage; zudem können jeweils mehrere Koordinaten vertauscht werden.

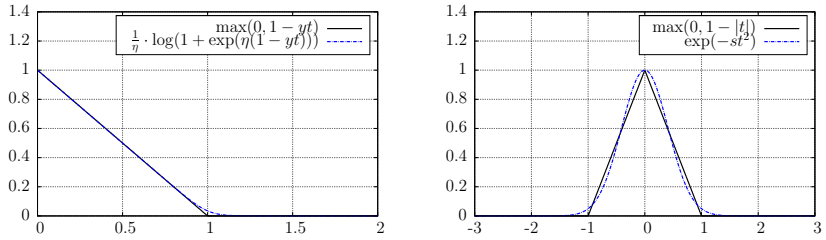


Abbildung 3: In Abb. (a) ist der *hinge loss* $\mathcal{L}(y,t) = \max(0, 1-yt)$ und seine differenzierbare Ersatzfunktion $\mathcal{L}(y,t) = \frac{1}{\eta}\log(1+\exp(\eta(1-yt)))$ mit $y = +1$ und $\eta = 20$ gezeigt. Die zugehörige effektive Verlustfunktion $\mathcal{L}(t) = \max(0, 1-|t|)$ und deren Ersatzfunktion $\mathcal{L}_e(t) = \exp(-st^2)$ mit $s = 3$ sind in Abb. (b) gezeigt.

3.3 Gradientenbasierte Optimierung

Eine spezielle Eigenschaft des Optimierungsproblems (3) ist die Tatsache, dass es als *kontinuierliches* Optimierungsproblem formuliert werden kann, und zwar in der Form

$$\underset{\mathbf{c}\in\mathbb{R}^n,\, b\in\mathbb{R}}{\text{minimiere}} \qquad \frac{1}{l}\sum_{i=1}^{l}\mathcal{L}\big(y_i', f(\mathbf{x}_i')+b\big) + \gamma\frac{1}{u}\sum_{i=1}^{u}\mathcal{L}_e\big(f(\mathbf{x}_i)+b\big) + \lambda\mathbf{c}^{\mathrm{T}}\mathbf{K}\mathbf{c} \tag{6}$$

wobei $\bar{\mathbf{x}}_i = \mathbf{x}_i'$ für $i = 1,\ldots,l$ und $\bar{\mathbf{x}}_i = \mathbf{x}_{i-l}$ für $i = l+1,\ldots,n$, und $f(\cdot) = \sum_{j=1}^{n} c_j k(\cdot, \bar{\mathbf{x}}_j)$ sei. Die Qualität des Modells wird dabei mittels zweier Verlustfunktionen bestimmt, wobei $\mathcal{L}$ der sogenannte *hinge loss* und $\mathcal{L}_e(t) = \mathcal{L}(\mathrm{sgn}(t), t)$ die zugehörige *effektive Verlustfunktion* ist, siehe Abb. 3.

Da die Funktion $\mathcal{L}_e$ nicht konvex ist, ist auch die Zielfunktion nicht konvex. Weiterhin existieren die partiellen Ableitungen nicht; dies schließt eine direkte Anwendung effizienter gradientenbasierter Optimierungsstrategien aus. Eine mögliche Herangehensweise besteht darin, auf differenzierbare Ersatzfunktionen für $\mathcal{L}$ und $\mathcal{L}_e$ zurückzugreifen [CZ05], siehe Abb. 3. Dies führt zu einer alternativen (aber sehr ähnlichen) Zielfunktion der Gestalt

$$\begin{aligned} F_\gamma(\mathbf{c}) = {} & \frac{1}{l}\sum_{i=1}^{l}\frac{1}{\eta}\log\left(1+\exp(\eta(1-y_i' f(\bar{\mathbf{x}}_i)))\right) \\ & + \frac{\gamma}{u}\sum_{i=1}^{u}\exp(-3(f(\bar{\mathbf{x}}_{l+i}))^2) + \lambda\sum_{i=1}^{n}\sum_{j=1}^{n} c_i c_j k(\bar{\mathbf{x}}_i, \bar{\mathbf{x}}_j), \end{aligned} \tag{7}$$

Das folgende Theorem zeigt, dass sowohl die Ersatzfunktion $F_\gamma(\mathbf{c})$ als auch deren Gradient $\nabla F_\gamma(\mathbf{c})$ effizient für eine beliebige Zwischenlösung $\mathbf{c} \in \mathbb{R}^n$ bestimmt werden kann:

Theorem 3 *([Gie11, GAPK12]) Für eine lineare Kernfunktion mit Mustern aus $\mathbb{R}^d$ und Datenmatrix $\mathbf{X} \in \mathbb{R}^{n\times d}$ mit $s \ll nd$ von Null verschiedenen Einträgen kann $F_\gamma(\mathbf{c})$ und $\nabla F_\gamma(\mathbf{c})$ in $\mathcal{O}(s)$ Laufzeit und Speicherplatz für ein beliebiges $\mathbf{c} \in \mathbb{R}^n$ bestimmt werden.*

Datensatz	l	u	LIBSVM	UniverSVM	S^{2}RLSC	QN-S^{3}VM
USPS(2,5)	16	806	10.5 ± 4.7	5.6 ± 3.6	6.3 ± 2.8	5.4 ± 1.7
USPS(2,7)	17	843	4.9 ± 2.9	2.3 ± 1.0	1.4 ± 0.2	3.6 ± 3.3
USPS(3,8)	15	751	12.9 ± 8.3	8.0 ± 3.5	6.2 ± 2.7	10.8 ± 8.9
USPS(8,0)	22	1108	5.0 ± 2.0	2.8 ± 2.2	1.8 ± 0.6	3.4 ± 1.3
TEXT	48	924	24.8 ± 9.6	6.7 ± 0.9	6.2 ± 0.9	8.2 ± 3.2
TEXT	389	584	4.9 ± 0.7	4.7 ± 0.4	5.0 ± 1.1	4.6 ± 0.6

Tabelle 1: Klassifikationsfehler von überwachten (LIBSVM) und halbüberwachten Support-Vektor-Maschinen (UniverSVM, S^{2}RLSC, QN-S^{3}VM) für Ziffern- und Texterkennung. Für die Trainingsphase kann dabei auf jeweils l gelabelte und u ungelabelte Muster zurückgegriffen werden. Angegeben ist der mittlere Fehler sowie die Standardabweichung über 10 Partitionen der Trainingsmuster.

Beweisidee: Der Gradient kann geschrieben werden als

$$\nabla F_\gamma(\mathbf{c}) \quad = \quad \mathbf{K}\mathbf{a} + 2\lambda\mathbf{K}\mathbf{c} \tag{8}$$

mit einem entsprechenden Vektor $\mathbf{a} \in \mathbb{R}^n$ [GAPK12]. Aufgrund der linearen Kernfunktion kann $\mathbf{K}\mathbf{c} = \mathbf{X}(\mathbf{X}^\mathrm{T}\mathbf{c})$ in $\mathcal{O}(s)$ Laufzeit bestimmt werden. Dies ermöglicht die effiziente Berechnung von $F_\gamma(\mathbf{c})$ und $\nabla F_\gamma(\mathbf{c})$ in $\mathcal{O}(s)$ Laufzeit, siehe [Gie11] für Details. □

Ähnliche Berechnungen können auch für andere Datenszenarien (und nicht-lineare Kernfunktionen) erzielt werden. Hinsichtlich der lokalen Suche kann anschließend auf Black-box-Verfahren zur gradientenbasierten Optimierung zurückgegriffen werden. Als besonders geeignet haben sich hier spezielle Varianten des Newton-Verfahrens erwiesen, welche die Hesse-Matrix nicht explizit bestimmen sondern nur approximieren [Gie11, GAPK12].

4 Anwendungen und Ausblick

In Tabelle 1 sind die Klassifikationsergebnisse einer überwachten Support-Vektor-Maschine (LIBSVM) und drei halbüberwachten Implementationen gezeigt (S^{2}RLSC ist die beschleunigte lokale Suche und QN-S^{3}VM das gradientenbasierte Verfahren; Universvm eine weitere halbüberwachte Vergleichsmethode). Die betrachteten Datensätze stammen aus zwei Anwendungsgebieten: Handschriftenerkennung (USPS, siehe Abb. 4) und Textklassifikation (Text). Die Ergebnisse zeigen, dass ein deutlich geringerer Klassifikationsfehler mit den halbüberwachten Varianten erzielt werden kann (um teilweise ca. 20%).

Exemplarisch sei noch ein Beispiel aus der Astronomie skizziert: Abb. 4 zeigt Bilddaten des *Sloan Digital Sky Survey* [YAAA00], wobei eine der Klassifikationsaufgaben darin besteht, elliptische Galaxien von Spiralgalaxien zu trennen. Falls nur wenige gelabelte Muster zur Verfügung stehen (hier: 10), so liefert eine Support-Vektor-Maschine einen Fehler von mehr als 30%; durch Hinzunahme von (117) ungelabelten Mustern kann dieser

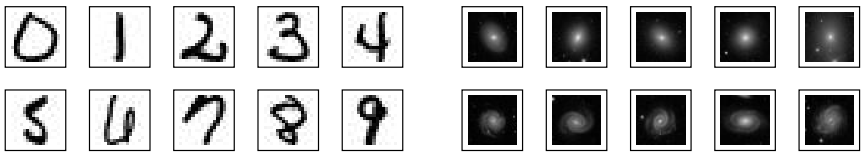


Abbildung 4: USPS-Zifferndatensatz [HTF09] und photometrische Daten von Galaxien [YAAA00].

mit der halbüberwachten Variante (QN−S^{3}VM) auf ca. 20% gesenkt werden.[2]

Da solche gelabelten Muster im Bereich der Astronomie für spezielle Fragestellungen oft nur in einem geringen Umfang zur Verfügung stehen, bieten halbüberwachte Ansätze ein enormes Potential. Dies trifft nicht nur auf Klassifikations- sondern auch auf z.B. Regressionsszenarien zu, wie aktuelle Arbeiten im Kontext der Schätzung der Rotverschiebung von sogenannten *Quasaren* [PZG13] eindrucksvoll zeigen. Der Bedarf entsprechender Techniken wird in naher Zukunft stark ansteigen, beispielsweise im Rahmen des *Large Synoptic Survey Telescope* [ITA$^+$11], welches *pro Nacht* 50 Terabyte an Daten liefern wird. Neben den riesigen Datenmengen wird hier vor allem der Mangel an gesicherten Label-Informationen ein großes Problem darstellen.

Danksagung. Teile der in diesem Beitrag vorgestellten Ergebnisse wurden von der *Deutschen Forschungsgemeinschaft* (DFG) unterstützt (KR 3695/2-1).

Literatur

[BD99] Kristin P. Bennett und Ayhan Demiriz. Semi-supervised support vector machines. In *Advances in Neural Information Proc. Systems 11*, Seiten 368–374. MIT Press, 1999.

[BV04] Stephen Boyd und Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[CZ05] Olivier Chapelle und Alexander Zien. Semi-Supervised Classification by Low Density Separation. In *Proc. of the 10th Int. Work. on Art. Intell. and Stat.*, Seiten 57–64, 2005.

[Ede87] Herbert Edelsbrunner. *Algorithms in Combinatorial Geometry*. Springer-Verlag, New York, NY, USA, 1987.

[Flo95] Christodoulos A. Floudas. *Nonlinear and Mixed-Integer Optimization: Fundamentals and Applications*. Oxford University Press, USA, 1995.

[GAPK12] Fabian Gieseke, Antti Airola, Tapio Pahikkala und Oliver Kramer. Sparse Quasi-Newton Optimization for Semi-Supervised Support Vector Machines. In *Proc. of the 1st Int. Conf. on Pattern Recognition Applications and Methods*, Seiten 45–54, 2012.

[Gie11] Fabian Gieseke. *From Supervised to Unsupervised Support Vector Machines and Applications in Astronomy*. Dissertation, Department für Informatik, Carl von Ossietzky Universität Oldenburg, 2011.

[2] Eine Support-Vektor-Maschine trainiert mit allen 127 Bildern liefert hier einen Fehler von ca. 15%.

[GKAP11] Fabian Gieseke, Oliver Kramer, Antti Airola und Tapio Pahikkala. Speedy Local Search for Semi-Supervised Regularized Least-Squares. In *Proc. of the 34th Annual German Conf. on Artificial Intelligence*, Seiten 87–98. Springer, 2011.

[GKAP12] Fabian Gieseke, Oliver Kramer, Antti Airola und Tapio Pahikkala. Efficient Recurrent Local Search Strategies for Semi- and Unsupervised Regularized Least-Squares Classification. *Evolutionary Intelligence*, 5(3):189–205, 2012.

[GPK09] Fabian Gieseke, Tapio Pahikkala und Oliver Kramer. Fast Evolutionary Maximum Margin Clustering. In *Proceedings of the 26th International Conference on Machine Learning*, Seiten 361–368, 2009.

[GVJ12] Fabian Gieseke, Jan Vahrenhold und Xiaoyi Jiang. On the Complexity of Exact Linear Semi- and Unsupervised Support Vector Machines. 2012. Manuskript.

[HTF09] Trevor Hastie, Robert Tibshirani und Jerome Friedman. *The Elements of Statistical Learning*. Springer, 2. Auflage, 2009.

[ITA+11] Z. Ivezic, J. A. Tyson, E. Acosta, R. Allsman und andere. LSST: from Science Drivers to Reference Design and Anticipated Data Products. 2011.

[Joa99] Thorsten Joachims. Transductive Inference for Text Classification using Support Vector Machines. In *Proc. of the Int. Conf. on Machine Learning*, Seiten 200–209, 1999.

[PZG13] Kai Lars Polsterer, Peter-Christian Zinn und Fabian Gieseke. Finding New High-Redshift Quasars by Asking the Neighbours. *Monthly Notices of the Royal Astronomical Society*, 428(1):226–235, 2013.

[Rif02] Ryan Michael Rifkin. *Everything Old is New Again: A Fresh Look at Historical Approaches in Machine Learning*. Dissertation, MIT, 2002.

[SC08] Ingo Steinwart und Andreas Christmann. *Support Vector Machines*. Springer, 2008.

[SS01] Bernhard Schölkopf und Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2001.

[VS77] V. Vapnik und A. Sterin. On structural risk minimization or overall risk in a problem of pattern recognition. *Automation and Remote Control*, 10(3):1495–1503, 1977.

[XNLS05] Linli Xu, James Neufeld, Bryce Larson und Dale Schuurmans. Maximum margin clustering. In *Adv. in Neural Inf. Proc. Systems 17*, Seiten 1537–1544, 2005.

[YAAA00] Donald G. York, J. Adelman, John E. Anderson und Scott F. Anderson. The Sloan Digital Sky Survey: Technical Summary. *The Astron. Journal*, 120(3):1579–1587, 2000.

Fabian Gieseke wurde am 4. April 1981 in Emsdetten, Deutschland, geboren. Er erhielt seine Diplome in Mathematik und Informatik von der Westfälischen Wilhelms-Universität Münster und den Doktor der Naturwissenschaften von der Carl von Ossietzky Universität Oldenburg. Während seines Promotionsstudiums beschäftige er sich mit den in diesem Beitrag vorgestellten Erweiterungen von Support-Vektor-Maschinen auf unüberwachte und halbüberwachte Lernszenarien, sowie mit Klassifikations- und Regressionsmodellen für konkrete Fragestellungen aus dem Bereich der Astronomie.