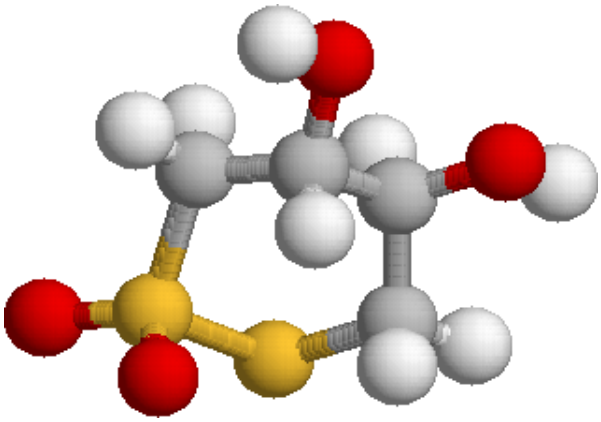
Rheinische Friedrich-Wilhems-Universität Bonn
Institut für Informatik III

Praktikumsbericht
– gekürzte Fassung –

Wissensentdeckung in Datenbanken

Sebastian Marius Kirsch
Betreuer: Thomas Gärtner
Lehrstuhl: Prof. Dr. Stefan Wrobel

Sommersemester 2004



Praktikum nach der Diplomprüfungsordnung für den Studiengang Informatik an der Rheinischen Friedrich-Wilhelms-Universität Bonn vom 14. März 2003.

Inhaltsverzeichnis

2. Eine Kernfunktion auf einer Menge von Punkten

2.1. Lernverfahren auf Basis von Kernfunktionen

Lernverfahren, die auf Kernfunktionen basieren, stellen in den letzten Jahren ein wichtiges Forschungsthema auf dem Gebiet des maschinellen Lernens und des Data Mining dar. Dabei ist eine Kernfunktion eine wie in Abschnitt 2.4 definierte Funktion, die implizit das innere Produkt bei einer Abbildung in einen höherdimensionalen (möglicherweise unendlich-dimensionalen) euklidischen Raum, dem sog. Merkmalsraum, berechnet.

Ein auf diese Weise implizit berechnetes inneres Produkt kann in einer Reihe von Verfahren benutzt werden, sofern diese sich so formulieren lassen, dass sie nur innere Produkte verwenden und niemals die Abbildung in den höherdimensionalen Raum explizit berechnen müssen. Beispiele für derartige Verfahren sind *Support Vector Machines* (Cortes und Vapnik, 1995), *Kernel Principal Component Analysis* (Schölkopf u. a., 1998, siehe auch Abschnitt 2.5), das Rosenblatt-Perceptron und das davon abgeleitete *Voted Perceptron* (Freund und Schapire, 1999).

Aus dieser Beschreibung und den notwendigen Eigenschaften einer Kernfunktion folgt, dass sich Kernfunktionen nicht nur auf Vektoren von reellen Zahlen definieren lassen, sondern auch auf strukturierten Daten. Durch die Kernfunktion wird implizit eine Abbildung in einen Raum definiert, in dem die Merkmale der Instanzen als Vektoren (von möglicherweise unendlicher Länge) dargestellt werden. (Ein Überblick über Kern-Methoden ist in Müller u. a. (2001) zu finden.)

Durch derartige Kernfunktionen lassen sich auch strukturierte Daten (wie Graphen, Bäume, endliche Automaten oder Punktmengen) direkt als Eingabeformat für Lernverfahren benutzen. Die Formulierung einer passenden Kernfunktion, die die Ähnlichkeit zwischen den Eingabedaten angemessen beschreibt, ist hier von grundlegender Bedeutung.

Nachdem in früheren Arbeiten an der Uni Bonn Kernfunktionen erforscht wurden, die auf der Graphenstruktur von Molekülen arbeiten (Horváth u. a., 2004), soll in diesem Praktikum ein Kern implementiert werden, der die 3D-Informationen des Moleküls benutzt. Die Kernfunktion soll auf einer Menge von Punkten im Raum (den Atomen eines Moleküls) arbeiten, wobei für jeden Punkt ein Label vorgegeben ist.

2.2. Verwandte Arbeiten

Kondor und Jebara (2003) verwenden Kernel-PCA, um Gauß-Verteilungen im höherdimensionalen Merkmalsraum an die Datenpunkte anzupassen; als Kernfunktion wird die Ähnlichkeit zwischen diesen Gauß-Verteilungen über Bhattacharyyas Ähnlichkeitsmaß berechnet.

Wolf und Shashua (2003) definieren einen Kernel durch die Haupt-Winkel zwischen zwei linearen Unterräumen, die durch Matrizen aufgespannt werden. Sie geben eine Möglichkeit an, diese Haupt-Winkel nur durch innere Produkte zu berechnen, was erlaubt, die Spaltenvektoren der aufspannenden Matrizen in einen hochdimensionalen Merkmalsraum abzubilden.

2.3. Notation

Im Folgenden werden Matrizen mit Großbuchstaben bezeichnet; $X \in M_{m\times n}(\mathbb{R})$ bezeichnet eine Matrix mit m Zeilen und n Spalten, wobei die Komponenten reelle Zahlen sind. $(X)_{ij}$ bezeichnet den Wert in Zeile i und Spalte j von X. $\mathbf{1}$ bezeichnet Matrizen, bei denen jede Komponente gleich 1 ist, also $(\mathbf{1})_{ij} \equiv 1 \; \forall\, i \; \forall\, j$.

Vektoren werden im Allgemeinen mit fettgedruckten Kleinbuchstaben bezeichnet: $\mathbf{x} \in \mathbb{R}^m$ bezeichnet einen Vektor mit m Komponenten.

Das innere Produkt in einem euklidischen Raum wird mit $\langle\cdot,\cdot\rangle$ bezeichnet.

Der komponentenweise Betrag eines Vektors oder einer Matrix wird mit $\mathrm{abs}(x)$ bezeichnet, d. h. $(\mathrm{abs}(x))_i = |(x)_i| \; \forall\, i$.

2.4. Kernfunktionen

Eine *Kernfunktion* auf einer Menge $\mathcal{X}$, dem *Instanzenraum*, ist eine symmetrische Funktion $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, die positiv definit ist: Für $n \in \mathbb{Z}^+, x_1, \ldots, x_n \in \mathcal{X}$ und $c_1, \ldots, c_n \in \mathbb{R}$ gilt

$$\sum_{i,j\in\{1,\ldots,n\}} c_i c_j k(x_i, x_j) \geq 0$$

Aus *Mercer's kernel theorem* folgt, dass für jede Kernfunktion k auf $\mathcal{X}$ eine Abbildung $\phi : \mathcal{X} \to \mathcal{H}$ in einen Merkmalsraum $\mathcal{H}$ existiert, so dass die Berechnung der Kernfunktion einem inneren Produkt in dem Merkmalsraum entspricht:

$$k(x, x') = \langle \phi(x), \phi(x') \rangle$$

Die Kernmatrix K von n Elementen $x_1, \ldots, x_n \in \mathcal{X}$ bezeichnet die Matrix

$$K \in M_{n\times n}(\mathbb{R}), (K)_{ij} = k(x_i, x_j)$$

2.5. *Kernel PCA*

Hauptkomponentenanalyse (*Principal Component Analysis*, PCA) ist ein bekanntes Verfahren zur Datenanalyse und -visualisierung. PCA basiert auf Eigenwertzerlegung der Kovarianzmatrix; dabei geben die Eigenvektoren (absteigend geordnet nach ihrem dazugehörigen Eigenwert) die Richtung der größten Varianz in den Datenpunkten an. Da alle Eigenvektoren linear unabhängig sind, bilden sie eine neue Basis des Vektorraums. Durch Projektion der Datenpunkte auf die $N \leq n$ Eigenvektoren mit größten Eigenwert wird eine Dimensionsreduzierung erreicht, bei der nur ein Minimum an Varianz der Datenpunkte verlorengeht.

Kernel PCA, vorgestellt von Schölkopf u. a. (1998), verallgemeinert das Verfahren der Hauptkomponentenanalyse auf Hauptkomponenten, die durch Projektion in einen Merkmalsraum entstehen. Dabei muss die Projektion der Datenpunkte in den Merkmalsraum nicht explizit berechnet werden; stattdessen genügt die implizite Angabe durch eine Kernfunktion.

Der durch die Kernfunktion implizit definierte Merkmalsraum kann unter Umständen unendlich-dimensional sein; deshalb können in diesem Merkmalsraum auch unendlich viele linear unabhängige Vektoren existieren. Hierbei hilft jedoch die Beobachtung, dass der durch Projektion von endlich vielen Datenpunkten in diesem Raum aufgespannte Teilraum nur endliche Dimension hat; somit ist die Zahl der Eigenvektoren mit Eigenwert ungleich Null in diesem Teilraum durch die Zahl der Datenpunkte beschränkt.

Für n Datenpunkte $x_1, \ldots, x_n \in \mathbb{R}^m$ wird statt der Kovarianzmatrix die Kernmatrix K der Datenpunkte für die Eigenwertzerlegung benutzt. Da eine herkömmliche Zentrierung der Datenpunkte in $\mathbb{R}^m$ nicht unbedingt einer Zentrierung im Merkmalsraum entspricht, wird stattdessen der Ausdruck

$$\tilde{K} = K - \frac{1}{n}K\mathbf{1} - \frac{1}{n}\mathbf{1}K + \frac{1}{n^2}\mathbf{1}K\mathbf{1}$$

zur Zentrierung im Merkmalsraum benutzt. (Für eine Herleitung siehe Schölkopf u. a., 1998)

Eigenwertwertzerlegung der zentrierten Kernmatrix $\tilde{K}$ bezeichnet die Zerlegung in eine Matrix $U = (\mathbf{u}_1 \cdots \mathbf{u}_n)$ von Eigenvektoren und eine Diagonalmatrix D mit Eigenwerten $\lambda_1, \ldots, \lambda_n$ auf der Diagonale, so dass

$$\tilde{K} = UDU^\top$$

Durch Sortierung der Eigenvektoren nach ihren Eigenwerten entsteht

$$\tilde{U} = (\tilde{\mathbf{u}}_1 \cdots \tilde{\mathbf{u}}_n) = (\mathbf{u}_{p_1} \cdots \mathbf{u}_{p_n})$$

mit $\lambda_{p_i} \geq \lambda_{p_j}$ für $i \leq j$ und $i, j \in \{1, \ldots, n\}$, wobei $(p_1, \ldots, p_n)$ eine Permutation von $(1, \ldots, n)$ darstellt.

Durch $\tilde{U}^\top \tilde{K}$ erhält man die Projektion der Datenpunkte auf die Hauptkomponenten im Merkmalsraum, wobei die erste Zeile der entstehenden Matrix die Projektion auf die Richtung der größten Varianz im Merkmalsraum enthält.

Im Fall des linearen Kerns $k_{\text{lin.}}(x_i, x_j) = \langle x_i, x_j \rangle = x_i^\top x_j$ entspricht die Projektion auf die Eigenvektoren mit Eigenwert ungleich Null genau der herkömmlichen Hauptkomponentenanalyse.

2.6. Die Kernfunktion

Bei den in diesem Praktikum betrachteten Instanzen handelt sich um Mengen von m-dimensionalen Vektoren $x = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}, \mathbf{x}_i \in \mathbb{R}^m$ mit Labeln $l_1, \ldots, l_n \in L$, L endlich. Der Einfachheit halber bezeichnen wir Instanzen auch als Matrizen

$$X = (\mathbf{x}_1 \cdots \mathbf{x}_n) \in M_{m \times n}(\mathbb{R})$$

mit einem Vektor von Labeln $\mathbf{l} = (l_1 \cdots l_n) \in L^m$. Die Kernmatrize $K(x)$ bezeichnet die $M_{n \times n}(\mathbb{R})$-Matrix mit $(K(x))_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$.

Gleiche Label teilen die Elemente $\mathbf{x}_1, \ldots, \mathbf{x}_n$ einer Instanz x in Klassen mit ähnlichen Eigenschaften ein. Im aktuellen Beispiel kann die Ordnungszahl der Atome als Label dienen; ein weiterer Ansatz zur Wahl von Label wird in Abschnitt 2.8 vorgestellt.

Für jedes Label $l \in L$ bezeichnet $\mathbf{1}_l$ den Vektor

$$(\mathbf{1}_l)_i = \begin{cases} 1 & \text{falls } l_i = l \\ 0 & \text{sonst} \end{cases}$$

Für jede Instanz x wird in einem vorbereitenden Schritt die Kernfunktion zwischen ihren Elementen berechnet. Die entstehende Kernmatrize $K = K(x)$ wird durch $\tilde{U}^\top \tilde{K}$ auf ihre Hauptkomponenten im Merkmalsraum projeziert (wie im letzten Abschnitt beschrieben.)

Die Menge der Label einer Instanz ist

$$L(x) = \bigcup_{i=1}^{n} \{l_i\}$$

Für alle $l \in L(x)$ definieren wir den Vektor $\phi_l(x)$

$$\phi_l(x) = \text{abs}(\tilde{U}^\top \tilde{K} \mathbf{1}_l)$$

Die Wahl von Absolutwerten für die Komponenten der $\phi_l(x)$-Vektoren ist aus folgendem Grund notwendig: Bei der Berechnung der Eigenvektoren erhält man zwar Vektoren in Richtung der größten Varianz im Merkmalsraum; ihre Orientierung ist jedoch nicht festgelegt. Um die Projektion auf die Eigenvektoren vergleichbar zu machen, wird

die Summe der Projektionen auf die einzelnen Eigenvektoren auf positive Werte normiert.(Zu den Ergebnissen ohne Absolutwerte siehe Abschnitt 3.3.)

Sei

$$m = \max_{x \in \mathcal{X}} \operatorname{rang}(K(x))$$

Um sicherzustellen, dass alle $\phi_l(x)$ die selbe Länge haben, werden die $\phi_l(x)$-Vektoren mit Nullen auf die Länge m aufgefüllt, d. h. es gilt

$$(\phi_l(x))_i = 0 \; \forall \operatorname{rang}(K(x)) < i \leq m \; \forall \, l \in L \; \forall \, x \in \mathcal{X}$$

Wie bereits erwähnt, kann der Merkmalsraum unendlich viele linear unabhängige Vektoren enthalten; die Zahl der Eigenvektoren mit Eigenwert ungleich Null im aufgespannten Teilraum ist jedoch durch den Rang der Kernmatrix nach oben beschränkt. Durch Auffüllen der $\phi_l(x)$-Vektoren mit Null wird also nur die Projektion auf die Eigenvektoren auf Null gesetzt, deren Eigenwert Null beträgt.

Die Kernfunktion zwischen zwei Instanzen x und x' ergibt sich als

$$k_{set}(x, x') = \sum_{l \in L(x) \cap L(x')} \phi_l(x)^\top \phi_l(x')$$

Da zur Berechnung von k_{set} nur innere Produkte verwendet werden, ist k_{set} ein positivdefinite Kernfunktion.

2.7. Berechnung von Kernfunktionen mit Matrizen

Da für die Eigenwertzerlegung die komplette Kernmatrix berechnet werden muss, bietet sich an, auch die Berechnung der Kernfunktion mit Matrixmethoden durchzuführen. Entsprechende Formulierungen für häufig benutzte Kernfunktionen finden sich in diesem Abschnitt.

2.7.1. Linearer Kern

In der Matrixschreibweise läßt sich die Kernmatrize für den linearen Kern $k_{lin.}(x_i, x_j) = \langle x_i, x_j \rangle$ durch

$$K_{lin.} = X^\top X$$

berechnen, da $(K)_{ij} = x_i^\top x_j = \langle x_i, x_j \rangle = k_{lin.}(x_i, x_j)$

2.7.2. Gaußscher Kern

Zur Berechnung der Gaußschen Kernfunktion $k_{rbf}(x_i, x_j) = e^{-g\|x_i - x_j\|^2}$ mit Matrizen benötigt man eine Hilfsmatrix

$$D \in M_{n \times n}(\mathbb{R}), (D)_{ij} = (K)_{ii},$$

womit sich die neue Kernmatrize aus der ursprünglichen Kernmatrize wie folgt ergibt:

$$K' = D - 2K + D^\top$$
$$(K_{rbf})_{ij} = e^{-g(K')_{ij}}$$

bzw. $k_{rbf}(x_i, x_j) = e^{-g(x_i - x_j)^2} = e^{-g(k(x_i,x_i) - 2k(x_i,x_j) + k(x_j,x_j))}$

2.7.3. Polynomieller Kern

Der polynomielle Kern $k_{poly}(x_i, x_j) = (1 + k(x_i, x_j))^p$ besitzt keine spezielle Formulierung in Matrixschreibweise.

2.8. Knotenfärbung

Die Zahl der möglichen Label ist im Beispiel der Moleküldaten durch die Ordnungszahlen der Atome stark eingeschränkt; insgesamt kommen in den betrachteten Wirkstoffen nur 63 verschiedene Elemente vor.

Um die Zahl der möglichen Label und damit die Unterscheidungsfähigkeit zu erhöhen, wird in einem Vorverarbeitungsschritt jedem Atom ein Label zu gewiesen, das aus dem Symbol des Atoms und einer Liste der Symbole seiner Nachbarn (in lexikografischer Ordnung) besteht:

$$l_i = (\text{Symbol von } i, \langle \text{Symbol von Nachbar } 1, \ldots, \text{Symbol von Nachbar } j \rangle)$$

Durch diese Art der Knotenfärbung erhöht sich die Zahl der verschiedenen Label in den verwendeten Daten von 63 auf 1897.

Ein Algorithmus für kanonische stabile Färbungen (siehe z. B. Immerman und Lander, 1990; Fürer, 1995) wurde in diesem Praktikum nicht evaluiert.

3. Experimente

3.1. Verwendete Daten

Die Experimente zur Evaluation der in Abschnitt 2.6 definierten Kernfunktion k_{set} beruhen auf den Moleküldaten des NCI und deren Evaluation bzgl. ihrer Wirksamkeit gegen das AIDS-Virus HIV. Dabei wurden folgende Quellen benutzt:

- Ergebnisse des AIDS-Screening-Tests des National Cancer Institute von März 2002 (`http://dtpsearch.ncifcrf.gov/ftp/aids/aids_conc_mar02.txt`)
- Chemische 3D-Strukturdaten der beteiligten Moleküle (`http://dtpsearch.ncifcrf.gov/FTP/AIDO99SD.BIN`)

Insgesamt sind für 42687 Wirkstoffe sowohl Ergebnisse des AIDS-Screening-Tests als auch 3D-Daten bekannt. Dabei wurden 41184 Wirkstoffe als inaktiv getestet, 1081 Wirkstoffe sind mässig aktiv, und 422 Wirkstoffe wurden als aktiv getestet.

Die Moleküle liegen ursprünglich ohne 3D-Daten vor; die 3D-Daten wurden deshalb mit Hilfe des Programms Corina der Arbeitsgruppe von Prof. Gasteiger an der Universität Nürnberg-Erlangen erstellt. Da Angaben über Stereoisomere nicht in den Ursprungsdaten vohanden sind, können die berechneten Koordinaten fehlerhaft sein.

Die Daten liegen im SDFile-Format vor; nach Experimenten mit dem Openbabel-Paket (siehe Stahl) wurde darauf verzichtet, sie direkt in die Anwendung einzulesen. Stattdessen werden sie durch eine selbst implementiertes Perl-Script in das RARFF-Format umgewandelt, das direkt eingelesen werden kann. Die Konvertierung geschieht mit Hilfe einer Klasse zur Behandlung von SDF-Daten, die soweit implementiert wurde, wie es für die aktuelle Aufgabe notwendig ist. (Zur Beschreibung des Eingabeformat siehe Abschnitt 4.1.)

3.2. Evaluation

Die durchgeführten Experimente lehnen sich an die Vorgehensweise in Horváth u. a. (2004) an. Für die Implementierung wurde das *gk*-Framework von Thomas Gärtner benutzt; dieses wiederum setzt auf SVM^{light}[1] von Thorsten Joachims als SVM-Implementierung auf. Zum Vergleich der Ergebnisse dient das Mass *area under ROC curve* (Hanley und McNeil, 1982), sprich die Fläche unter der ROC-Kurve für den gefundenen Klassifikator.

[1] `http://svmlight.joachims.org/`

Bei Verwendung des `pointSet`-Kerns muss an zwei Stellen eine Kernfunktion ausgewählt werden: Zum Einen wird *Kernel PCA* verwendet; hier muss eine Kernfunktion zwischen den einzelnen Komponenten einer Instanz ausgewählt werden. Ausserdem kann auch die Kernfunktion zwischen den Instanzen $k(x_i, x_j)$ mit Hilfe eines RBF-Funktion oder eines Polynoms modifiziert werden.

In den Experimenten hat sich gezeigt, dass wir mit der Kombination aus RBF-Kern zwischen den Komponenten einer Instanz und unmodifizierten (linearen) Kern zwischen den Instanzen die höchste *area under ROC curve* erreichen konnten. Wählt mal zweimal einen linearen Kern, findet der verwendete SVM-Algorithmus oft gar keine Lösung; wählt man zweimal einen RBF-Kern, erzielt man wiederum schlechtere Ergebnisse. (Der polynomielle Kern wurde bei den Experimenten aussen vor gelassen.)

Auf dem kleinsten Problem (Unterscheidung zwischen CA und CM) wurde ein geeigneter Parameter für den Regularisierungs-Parameter c der verwendeten SVM gesucht; eine Visualisierung der Ergebnisse findet sich in Abbildung 3.1. Für alle weiteren Experimente wurde $c = 0.1$ gewählt.

Im nächsten Experiment wurde der Einfluss von Knotenfärbung (siehe Abschnitt 2.8) untersucht; die Ergebnisse in Abbildung 3.2 zeigen eine wesentliche Verbesserung durch den Einsatz von Knotenfärbung.

Die Ergebnisse aus diesem Test konnten direkt für die Wahl eines passenden Parameters g für die Breite der Gaußschen Funktionen des RBF-Kerns verwendet werden. Experimente in den Größenordnungen von 10^{-5} bis 10^{3} (siehe Abbildung 3.2) ergaben den Wert $g = 0.1$ als vielversprechende Wahl.

Mit den so gefundenen Werten ($c = 0.1$, $g = 0.1$, Kantenfärbung) wurden Experimente auf den größeren Problemen durchgeführt. Die Ergebnisse sind in Tabelle 3.1 zusammengestellt.

Zum Vergleich der Kernfunkion auf 3D-Daten der Moleküle mit spektralen Methoden wurden ebenfalls Experimente mit dem Kern k_{set} auf den Adjazenzmatrizen der Molekülgraphen durchgeführt, d. h. Instanzen x mit Matrizen $X \in M_{n\times n}(\{0,1\})$ mit

$$(X)_{ij} = \begin{cases} 1 & \text{falls Bindung zwischen Atom } i \text{ und Atom } j \\ 0 & \text{sonst} \end{cases}$$

Beim Vergleich der Ergebnisse sind zwei Tatsachen zu beobachten: Zum Einen zeigt die Kernfunktion auch auf Adjazenzlisten sehr gute Ergebnisse, obwohl ihre Entwicklung nicht unter diesem Einsatzgebiet motiviert wurde. Zum Zweiten sind die Ergebnisse auf Adjazenzmatrizen durchweg besser als die Ergebnisse auf 3D-Daten. Eine Erklärung für dieses Verhalten steht bislang noch aus.

3.3. Unvorhergesehene Ergebnisse

Versäumt man es, bei der Berechnung der $\phi_l(x)$-Vektoren in Abschnitt 2.6 Absolutwerte zu verwenden, ergibt sich eine Kernfunktion, durch die auf unseren Daten ein

Tabelle 3.1.: *area under ROC curve* für den `pointSet`-Kern auf verschiedenen Problemen; alle Experimente wurden mit den Parametern $c = 0.1$ und $g = 0.1$ unter Einsatz von Knotenfärbung durchgeführt. Zum Vergleich sind die Ergebnisse aus Horváth u. a. (2004) für *walk-based kernels* und *cyclic pattern kernels* ebenfalls abgedruckt. (Der Eintrag „—“ bezeichnet ein Experiment, bei der die SVM kein Ergebnis lieferte.)

problem	cost	point set kernels on 3D data	point set kernels on adj. matrix	walk-based kernels	cyclic pattern kernels
CA vs. CM	1.0	0.774 ± 0.014	0.796 ± 0.010	0.818 ± 0.024	0.813 ± 0.014
CA vs. CM	2.5	0.767 ± 0.022	0.798 ± 0.022	0.825 ± 0.32	0.827 ± 0.013
CA vs. CM+CI	1.0	0.859 ± 0.023	0.858 ± 0.018	0.926 ± 0.015	0.908 ± 0.024
CA vs. CM+CI	100.0	0.840 ± 0.023	0.882 ± 0.022	0.928 ± 0.013	0.921 ± 0.026
CA+CM vs. CI	1.0	0.735 ± 0.017	0.732 ± 0.013	0.815 ± 0.015	0.775 ± 0.017
CA+CM vs. CI	35.0	—	0.751 ± 0.013	0.799 ± 0.011	0.801 ± 0.017
CA vs. CI	1.0	0.876 ± 0.026	0.873 ± 0.033	0.942 ± 0.015	0.919 ± 0.011
CA vs. CI	100.0	0.851 ± 0.030	0.886 ± 0.027	0.944 ± 0.015	0.929 ± 0.01

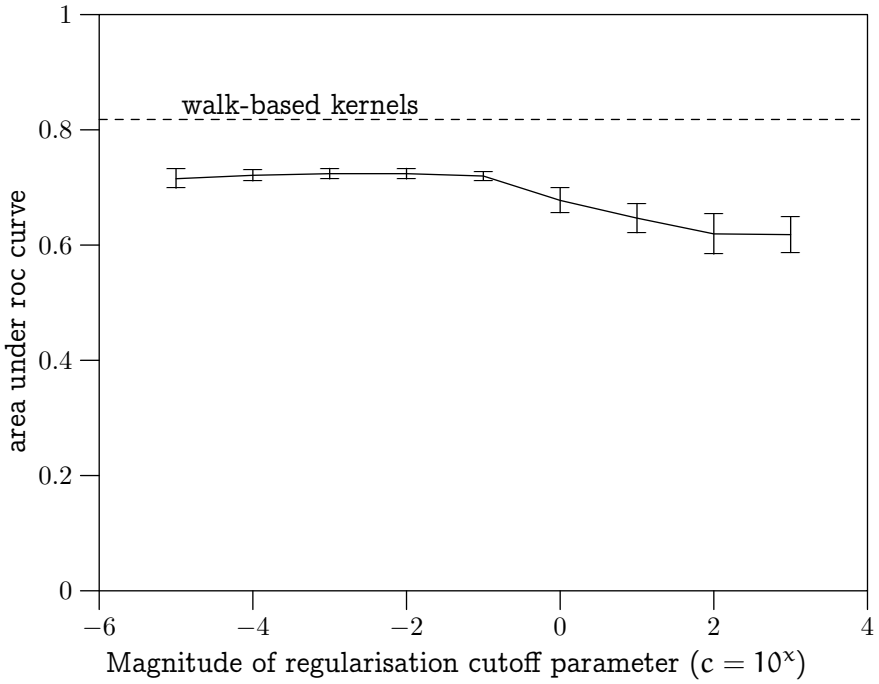


Abbildung 3.1.: Einfluss des Regularisierungs-Parameters auf die *area under roc curve* (Daten: CA vs. CM ohne Knotenfärbung, fünffache Kreuzvalidierung)

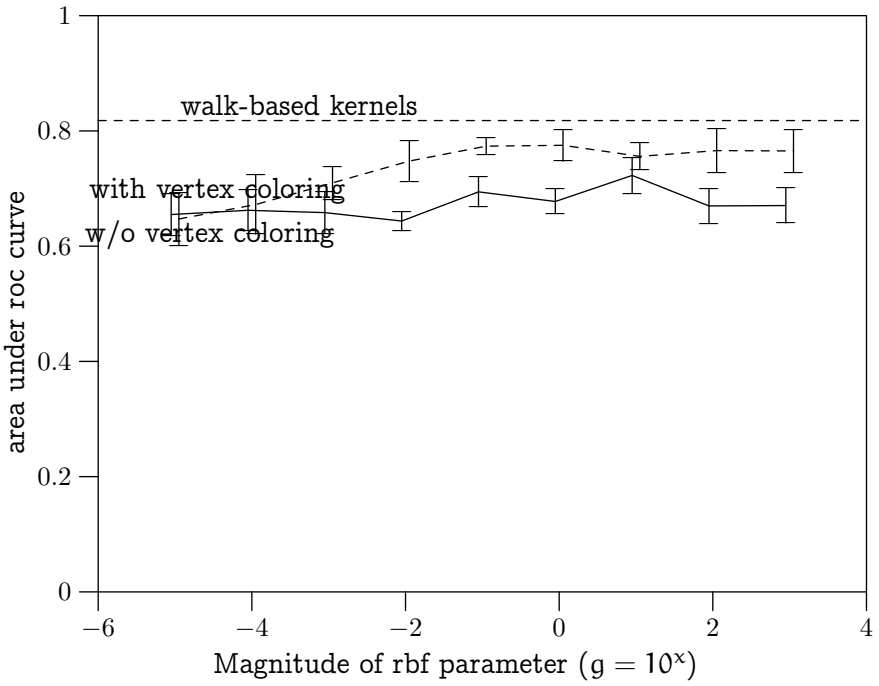


Abbildung 3.2.: Einfluss von Knotenfärbung auf die *area under roc curve* (Daten: CA vs. CM, fünffache Kreuzvalidierung)

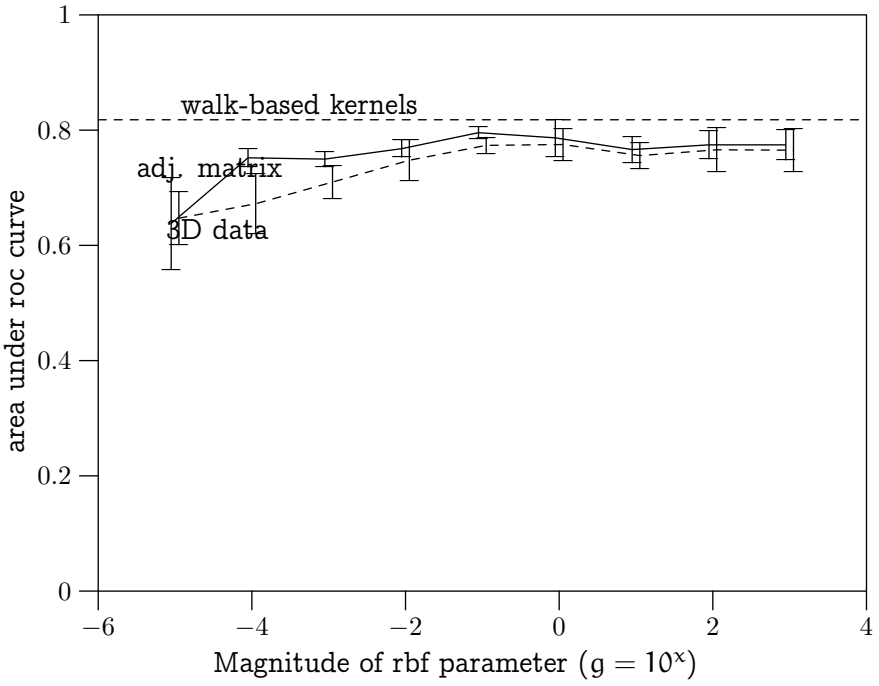


Abbildung 3.3.: Vergleich des `pointSet`-Kerns auf 3D-Daten gegenüber Adjazenzlisten (Daten: CA vs. CM mit Knotenfärbung, fünffache Kreuzvalidierung)

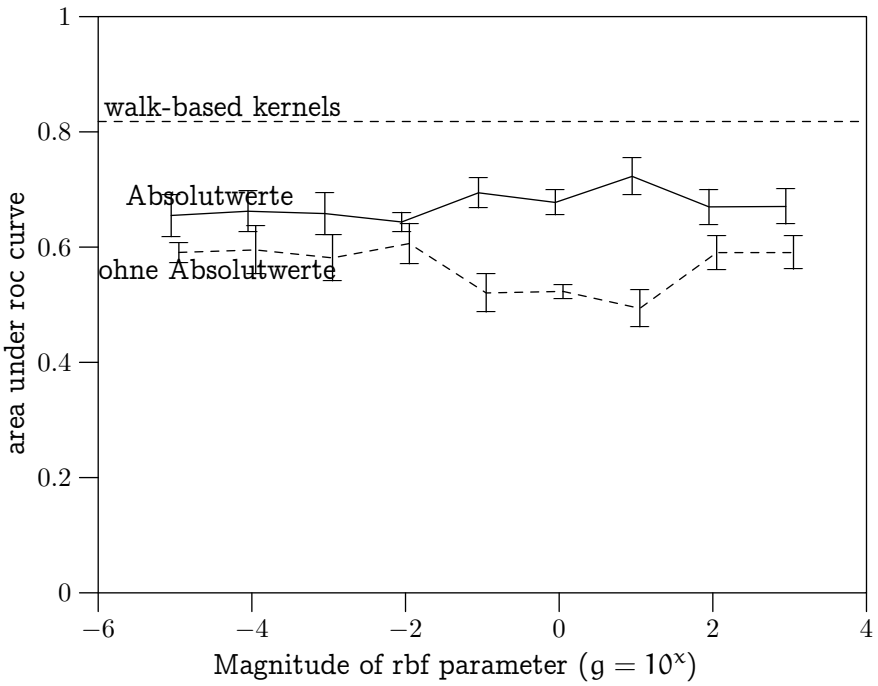


Abbildung 3.4.: *area under roc curve* gegen RBF-Parameter (Daten: CA vs. CM ohne Knotenfärbung, fünffache Kreuzvalidierung)

Klassifikator mit einer *area under roc curve* wie in Abbildung 3.4 gefunden wird: Im schlechtesten Fall ist die Klassifizierung schlechter als raten, und es ist kein eindeutiger Bereich festzustellen, in dem die Kernfunktion bessere Resultate liefert.

5. Fazit

Die Ergebnisse dieses Praktikums haben gezeigt, dass eine Kombination von Standardverfahren eine Kernfunktion ergibt, die es auf den untersuchten Daten erlaubt, das Klassifikationsproblem zufriedenstellend zu lösen. Die eingesetzten Verfahren sind Kernel-PCA zur Transformation der Eingabepunkte, Knotenfärbung und Aufsummieren der inneren Produkte gleich gefärbter Knotenvektoren. Die Performanz der in Horváth u. a. (2004) vorgestellten Kernfunktionen wird jedoch nicht erreicht.

Eine Abschätzung für die Laufzeit bei der Berechnung der Kernfunktion wurde nicht erbracht.

Eine Erklärung für das Verhalten der Kernfunktion auf Adjazenzmatrizen (wie in Abschnitt 3.2 beschrieben) wurde noch nicht gefunden. Eine genauere Untersuchung dieser Tatsache könnte zu einer Klasse von spektralen Kern-Methoden auf Graphen führen.

B. Bibliotheken für lineare Algebra

Eine Recherche im Internet bzgl. Bibliotheken für Matrix-Operationen, Eigenwertprobleme und ähnliches ergab folgende Softwarepakete:

LAPACK LAPACK ist der Urvater aller Bibliotheken für lineare Algebra; in Fortran-77 geschrieben, existieren heute Anbindungen an viele andere Sprachen. LAPACK benutzt eine Sammlung von Routinen namens BLAS für grundlegende Operationen auf Matrizen und Vektoren.

Der Funktionsumfang und die Geschwindigkeit von LAPACK dürften schwer zu übertreffen sein; seine Komplexität und die Tatsache, dass keine modernen Programmierparadigmen wie objektorientierte Programmierung unterstützt werden, erschweren jedoch die Einarbeitung und die Einbindung in aktuelle Projekte.

LAPACK und BLAS dienen als Grundlage für viele andere Pakete im Bereich lineare Algebra. (http://www.netlib.org/lapack/)

LAPACK und BLAS existieren in optimierten Versionen für viele Architekturen und Vektor-Befehlssätze:

Sun Performance Library (SUNWperf) für UltraSPARC im 32- und 64-Bit-Modus enthält optimierte Varianten von LAPACK 3.0 und BLAS Level 1–3. Sie ist Bestandteil der Sun Studio-Entwicklungsumgebung; Lizenzkosten belaufen sich auf ca. $3000. (http://docs.sun.com/db/doc/817-6701)

Apple MacOS X Vector Libraries enthalten BLAS Level 1–3 in einer Version, die auf AltiVec aufbaut, der Vektor-Einheit des G4- und G5-Prozessors; weiterhin werden LAPACK und CLAPACK ausgeliefert. Diese Libraries sind Bestandteil des „Accelerate“-Frameworks und gehören zum Lieferumfang von Mac OS X. (http://developer.apple.com/hardware/ve/vector_libraries.html)

AMD Core Math Library 2.0 ist erhältlich für 32- und 64-Bit-Linux auf AMD Opteron und Athlon64-Prozessoren. BLAS und LAPACK 3.0 sind in optimierten Varianten enthalten. Die Lizenz ist kostenfrei. (http://www.developwithamd.com/apppartnerprog/acml/home/index.cfm?action=home)

Intel Math Kernel Libraries für Windows und Linux enthalten BLAS und LAPACK in Versionen für Pentium 4, Xeon und Itanium. Die Lizenzkosten belaufen sich auf $99–$399 pro Arbeitsplatz; nichtkommerzieller Einsatz ist kostenlos. (http://www.intel.com/software/products/mkl/index.htm)

LAPACK++ LAPACK++ ist eine Anpassung von LAPACK an C++; das Projekt wurde inzwischen zugunsten von TNT eingestellt. (http://math.nist.gov/lapack++/)

TNT Das *Template Numerical Toolkit* ist der designierte Nachfolger von LAPACK++ und einer Reihe von weiteren Paketen. In der aktuellen Version kann es jedoch nur grundlegende Operationen auf Matrizen durchführen; ausserdem sind Algorithmen für Eigenwertzerlegung, Cholesky-, LU- und QR-Faktorisierung sowie SVD implementiert.

Die Zusammenarbeit zwischen Vektor- und Matrixklassen ist schlecht; so werden Vektor-Matrix-Multiplikation oder Erzeugung einer $n \times 1$-Matrix aus einem Vektor nicht unterstützt. Grundlegende Operationen wie Transponieren einer Matrix wird von der aktuellen Bibliothek nicht unterstützt.

Das Projekt ist laut Webseite nicht unter aktiver Entwicklung. (http://math.nist.gov/tnt/)

GMM++ GMM++ ist Teil des Pakets GetFEM, das Operationen für finite Elemente bereitstellt. GMM++ implementiert Operationen für dicht und spärlich besetzte Matrizen; grundlegende Operationen lassen sich durch setzen eines Flags beim Compilieren von LAPACK oder ATLAS durchführen. Eine Einbindung von SuperLU zum Lösen von spärlich besetzten Matrizen ist vorhanden. GMM++ lehnt sich an andere C++-Bibliotheken wie MTL und ITL an.

GMM++ sieht sich weniger als eigenständige Bibliothek denn als „Klebstoff“ zwischen vorhandenen Klassen. So wird zur Speicherung von Vektoren die Klasse `std::vector` benutzt. GMM++ kann mit beliebigen Datentypen als Basistyp umgehen, sofern alle arithmetischen Operationen auf diesem Typ implementiert sind.

Das Paket macht extensiven Gebrauch von Templates und stellt so eine einfache Anpassbarkeit an neue Datentypen sicher. (http://www.gmm.insa-tlse.fr/getfem/gmm_intro)

newmat Das Paket von Robert Davies enthält mehrere Matrixtypen (Dense, Symmetric, Diagonal, Triangular etc.) und Operationen auf diesen Typen. Algorithmen zur Faktorisierung sind enthalten, werden jedoch nur für kleine Probleme empfohlen.

Die Dokumentation zu diesem Paket ist sehr gut und enthält nicht nur Beispielprogramme, sondern auch Hintergründe, die zur aktuellen Implementation geführt haben. Auch die Optimierungsverfahren werden detailliert erklärt; sie scheinen sich an die in (Stroustrup, 1997, Kap. 22) vorgeschlagenen Techniken anzulehnen.

Damit empfiehlt sich newmat für den Einsatz in der Lehre: Der Funktionsumfang genügt für kleine Probleme, die Dokumentation erleichtert die schnelle Einarbeitung, und das gute Design ermöglicht die Erstellung von gut lesbaren, kompakten Programmen. (http://www.robertnz.net/nm_intro.htm)

LinAl LinAL enthält Interfaces zu BLAS, LAPACK, ARPACK und SuperLU; es ist jedoch seit zwei Jahren ohne neues Release im Beta-Stadium. (http://sourceforge.net/projects/linal)

MTL Die *Matrix Template Library* ist ein umfassendes Paket für Matrixoperationen. Sie orientiert sich bzgl. des Programmierparadigmas an der Standard Template Library: Es werden z. B. Container, Iteratoren und Funktionsobjekte angeboten. Operatoren auf Matrizen und komplexe Algorithmen werden über ein Interface zu LAPACK und BLAS realisiert. (http://www.osl.iu.edu/research/mtl/)

GSL Die *GNU Scientific Library* ist eine Bibliothek des GNU-Projekts für numerische Probleme. Sie umfasst auch Matrix- und Vektoperationen, sowie einfache Algorithmen für Eigenwertprobleme. Die Autoren empfehlen jedoch LAPACK für grössere Probleminstanzen.

GSL ist eine C-Bibliothek, kann aber auch in C++ eingebunden werden. Der Programmierstil orientiert sich am Stil der *GNU Multiple Precision Arithmetic Library* GMP.

Nach meiner Einschätzung empfiehlt sich GSL für Projekte in C, die sich auch sonst aus dem Fundus des GNU-Projekts bedienen. (http://www.gnu.org/software/gsl/)

Weiterführende Informationen finden sich an folgenden Stellen:

The Object-Oriented Numerics Page (http://www.oonumerics.org/oon/)

Robert Davies' List of Libraries Vom Autor von newmat (http://www.robertnz.net/cpp_site.html#Library)

Aufgrund der so gewonnen Übersicht über die vorhandenen Bibliotheken habe ich mich für den Einsatz von GMM++ entschieden. Ausschlaggebend hierfür waren der Funktionsumfang, die gute Interaktion zwischen Matrix- und Vektortypen und die Möglichkeit, durch Einbindung von LAPACK und ARPACK gegebenenfalls schnelle Matrixoperationen zu realisieren.

Die von Thomas Gärtner bisher benutzte Bibliothek TNT/JAMA++ scheint aufgrund des geringen Funktionsumfangs nicht geeignet für weitergehende Experimente.

Literaturverzeichnis

[Cortes und Vapnik 1995] Cortes, C. ; Vapnik, V. N.: Support vector networks. In: *Machine Learning* 20 (1995), S. 273–297

[Fawcett und Mishra 2003] Fawcett, Tom (Hrsg.) ; Mishra, Nina (Hrsg.): *Proceedings of the Twentieth International Conference on Machine Learning*. AAAI Press, 2003

[Flach 2003] Flach, Peter A.: The geometry of ROC space: understanding machine learning metrics through ROC isometrics. In: (Fawcett und Mishra, 2003), S. 194–201

[Freund und Schapire 1999] Freund, Yoav ; Schapire, Robert E.: Large margin classification using the perceptron algorithm. In: *Machine Learning* 37 (1999), Nr. 3, S. 277–296

[Fürer 1995] Fürer, Martin: Graph isomorphism testing without numerics for graphs of bounded eigenvalue multiplicity. In: *Proceedings of the sixth annual ACM-SIAM symposium on Discrete algorithms*, Society for Industrial and Applied Mathematics, 1995, S. 624–631. – ISBN 0-89871-349-8

[Hanley und McNeil 1982] Hanley, James ; McNeil, Barbara: The Meaning and Use of the Area under Receiver Operating Characteristics (ROC) Curve. In: *Radiology* 143 (1982), Nr. 1, S. 29–36. – URL http://www.med.mcgill.ca/epidemiology/hanley/software/Hanley_McNeil_Radiology_82.pdf

[Horváth u. a. 2004] Horváth, Tamás ; Gärtner, Thomas ; Wrobel, Stefan: Cyclic pattern kernels for predictive graph mining. In: *Proceedings of the 2004 ACM SIGKDD international conference on Knowledge discovery and data mining*, 2004, S. 158–167. – ISBN 1-58113-888-9

[Immerman und Lander 1990] Immerman, Neil ; Lander, Eric: *Describing Graphs: A First-Order Approach to Graph Canonization*. Kap. 4, S. 59–81. In: Selman, Alan L. (Hrsg.): *Complexity Theory Retrospective*, Springer, 1990

[Kondor und Jebara 2003] Kondor, Risi ; Jebara, Tony: A Kernel between Sets of Vectors. In: (Fawcett und Mishra, 2003)

[Müller u. a. 2001] Müller, Klaus-Robert ; Mika, Sebastian ; Rätsch, Gunnar ; Tsuda, Koji ; Schölkopf, Bernhard: An introduction to kernel-based learning algorithms. In: *IEEE Transactions on Neural Networks* 12 (2001), Nr. 2, S. 181–202

[Provost und Fawcett 2001] Provost, Foster J. ; Fawcett, Tom: Robust Classification for Imprecise Environments. In: *Machine Learning* 42 (2001), Nr. 3, S. 203–231. – URL http://citeseer.ist.psu.edu/article/provost89robust.html

[Schölkopf u. a. 1998] Schölkopf, Bernhard ; Smola, Alexander ; Müller, Klaus-Robert: Nonlinear component analysis as a kernel eigenvalue problem. In: *Neural Computation* 10 (1998), S. 1299–1319. – URL http://citeseer.ist.psu.edu/sch98nonlinear.html

[Stahl] Stahl, Matthew T.: *OELib Primer: An Introduction to Programming with the OpenEye Library*. – URL http://openbabel.sourceforge.net/Primer.shtml

[Stroustrup 1997] Stroustrup, Bjarne: *The C++ Programming Language*. 3rd. Addison-Wesley, 1997. – ISBN 0-201-88954-4

[Witten und Frank 2000] Witten, Ian H. ; Frank, Eibe: *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Kap. 8, S. 265–320, Morgan Kaufman, 2000

[Wolf und Shashua 2003] Wolf, Lior ; Shashua, Ammon: Learning over Sets using Kernel Principal Angles. In: *Journal of Machine Learning Research* 4 (2003), Nr. 10, S. 913–931