Maschine Learning
Nonparametric Techniques

Peter Hallama

11.11.2003

Inhaltsverzeichnis

1 Einleitung

Im Gegensatz zu den bisher kennengelernten parametrischen Techniken die eine Grundannahme über die Gesamtverteilung der betrachteten Werte zu grunde legen, beziehen sich nichtparametrische Techniken auf Verfahren, die keine Aussagen über einzelne Parameter der Grundgesamtheitsverteilung machen.

Es gibt viele Gründe, auch trotzt zum Teil effektiveren Algorithmen heutzutage, immer noch auf fast schon klassische nicht parametrische Verfahren zurückzugreifen. Zum einen sind die nichtparametrischen Techniken einfacher zu erlernen und zu handhaben als die parametrischen, zum anderen gibt es immer noch Spezialfälle in denen speziell angepasste nichtparametrische Verfahren die besten Ergebnisse erziehlen. Oder es werden in vielen Fällen gar nicht so genaue Ergebnisse benötigt, so dass gerne auf die effektiveren nichtparametrischen Techniken zurückgegriffen wird.

Nichtparametrische Techniken versuchen eine möglicherweise sehr komplexe Funktion durch lokale Teillösungen anzunähren und können so um den Preis einer geringen Abweichung sehr aufwendige Berechnungsschritte umgehen.

Somit ist für jeden leicht ersichtlich, warum auch jetzt noch nichtparametrische Techniken eine Rolle spielen.

2 Überblick

Im folgenden werde ich etwas genauer auf die wichtigsten nichtparametrischen Algorithmen eingehen den
- k-Nearest Neighbor,
- Locally weighted Regression,
- Radial Basis Functions
- und nicht zuletzt Case Based Reasoning.

Ich werde des weiteren einen kurzen Überblick über Learning Vector Quantization geben und anhand eines aktuellen Beispiels einen Leistungsvergleich zu anderen Algorithmen anführen.

3 Der k-Nearest Neighbor

Der k-Nearest Neighbor Algorithmus ist mit Sicherheit einer der einfachsten und zugleich oft effizientesten Algorithmen aus diesem Gebiet. Wie bei vielen nichtparametrischen Algorithmen werden hier die Instanzen als Punkte im mehrdimensionalen Raum dargestellt.

Eine Instanz wird über ihre Attribute definiert. Jedes Attribut stellt eine Achse im mehrdimensionalen Raum, die Anzahl des Auftretens der Attribute wird zu einem Punkt-Vektor zusammengefügt,

$$x = (a_1(x), a_2(x), ..., a_n(x))$$

a_r definiert die Anzahl des Vorkommens des Attributs x.

Entschieden wird bei der k-Nearest Neighbor Methode, wie der Name schon erahnen lässt, anhand der nächstliegenden Daten zum neu einzuordnenen Punkt. Definiert wird der nächste Nachbar über die einfache euklid'sche Abstandsfunktion

$$d(x_i, x_j) = \sqrt{\sum_{r=1}^{n}(a_i(x_r) - a_j(x_r))^2}$$

Als Rückgabewert beim k-Nearest Neighbor erhalten wir den am häufigsten vorkommenden Wert unter den in Betracht gezogenen Werten.

Wie dieses Beispiel verdeutlicht kann es beim k-Nearest Neighbor durchaus vorkommen, dass die Funktion, die nur einen Nachbarn mit einbezieht, einen anderen Wert liefert als die, die mehrere Nachbarn mit einbezieht. In unserem Beispiel würde der 1-NN unseren Punkt als Kreuz klassifizieren, der 3 sowohl 5-NN würden Ihn als Kreis einstufen.

Um möglichst eindeutige Klassifizierungen zu erhalet liegt es nahe, den Trainingsdaten eine Gewichtung in Abhängigkeit vom Abstand von unserer Instanz zu geben. Diese Gewichtungsfunktion können wir relativ frei wählen, sie sollte natürlich auf unsere Trainingsdaten zugeschnitten sein. Eine naheliegende Möglichkeit wäre, unsere Trainingsdaten mit einer quadratischen Abstandsfunktion einfliessen zu lassen.

$$w_i = \frac{1}{d(x_q, x_i)^2}$$

Somit werden nahe am Punkt liegende Daten erheblich stärker gewichtet, als weit entfernte Daten, da sehr weit entfernte Daten sogar fast gar keinen

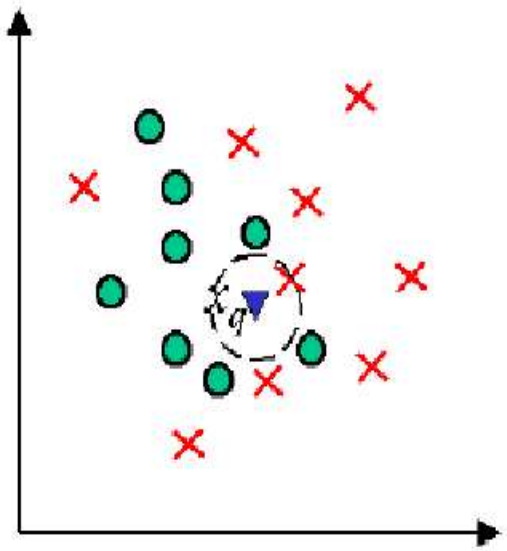

Abbildung 1: k-Nearest Neighbor

Einfluss mehr auf unsere Instanz haben bietet es sich hierfür sogar an den gesamten Trainingsraum mit in die Klassifizierung einfliessen zu lassen.

Eine Tatsache, die dabei noch ausser Acht gelassen wurde wäre, dass der Abstand für unseren Punkt immer über alle Attribute berechnet wird. Hierbei kann es bei höherer Anzahl von Dimensionen leicht zu Verzerrungen der Ergebnisse durch nicht relevante Attribute kommen. Eine mögliche Lösung für dieses Problem wäre den einzelnen Attributen ebenfalls ein Gewicht zu geben. Dies wird praktisch realisiert indem man einfach die Attribute mit einem Gewichtungsfaktor versieht und so räumlich gesehen eine Streckung der Achsen erreicht. Schwächer gewichtete Achsen sind also kürzer und gehen somit auch weniger stark in den Gesamtabstand zweier Punkte mit ein.

Ein anderes Problem des k-NN wie auch vieler anderer nichtparametrischer Algorithmen ist der hohe Speicherbedarf, welcher durch die Speicherung aller Trainingsdaten entsteht, und vor allem die damit verbundenen hohen Speicherzugriffszeiten. Eine oft für den k-NN gewählte Lösung ist der kd-tree, der nahe beieinander liegende Punkte bereits in nahe beieinander liegenden Knoten einer Baumstrucktur speichert und so bei minimalen Speichermehrbedarf zu einer deutlichen Erhöhung der Effektivitaet führt. Ge-

nauer untersucht wird der Nearest Neighbour mit der kd-tree Methode in dem Paper "The Labeled Cell Classifier: A Fast Approximation to k Nearest Neighbors"[1].

Im allgemeinen liegen die Vorteile des k-NN in dem schnellen Training, dieses besteht nämlich nur aus der Speicherung der neu eingestuften Instanzen. Des weiteren gehen bei dieser Methode keinerlei Daten verlohren, da alle Trainingsdaten gespeichert bleiben. Dazu kommen die allgemeinen Vorteile der nichtparametrischen Techniken. Ein Nachteil des k-NN ist allerdings, das er ein sogenannter fauler Algorihtmus ist.

Faule Algorithmen machen alle Berechnungen erst beim Auftreten eines neuen Problems, was vorallem bei entsprechend hoher Dimensionenzahl zu Verzögerungen führt. Das und das bereits erwähnte Problem, dass der k-NN alle Attribute zur Abstandsfindung heranzieht, kann aber noch durch Algorithmen zur Dimensionsreduktion behandelt werden.

4 Locally Weighted Regression

Die Lernaufgabe ist bei der Regression der Klassifikation (ä)hnlich. Aber es ist statt einer Vorhersage einer bestimmten Klasse eine numerische Vorhersagen aus Attribut Wertpaaren zu leisten.

Auch bei der Locally Weighted Regression wird keine Grundannahme (ü)ber die Gesamterteilung der Werte getroffen und so z(ä)hlt sie zu den nichtparametrischen Techniken.

Die Locally Weighted Regression versucht, im Gegensatz zum k-NN, die gesuchte Zielfunktion möglichst genau in der näheren Umgebung des Anfragepunktes anzunähren. Hierbei kann sie sich allen Arten von Funktionen oder auch neuronalen Netzwerken bedienen. Da aber der Rechenaufwand für komplexere Funktionen schnell zu hoch wird werden lineare oder quadratische Annährungsfunktionen am häufigsten verwendet. Es wird versucht, den qadratischen Fehler über die k-NN der Funktion so gering wie möglich zu halten.

Ein häufig verwendeter Spezialfall der LWR ist die Locally Weighted Linear Regression. Hier wird versucht, die Zielfunktion lokal linear anzunähren, also eine Linie zu finden, die unsere Punkte am besten unterscheidet. Hierzu versucht man, den quadratischen Abstand aller Punkte zur Linie zu minimieren, da in diesem Fall der Abstand genau dem Fehler entspricht.

Abbildung 2 zeigt, wie eine LWLR eine Funktion lokal annährt, die Kurve

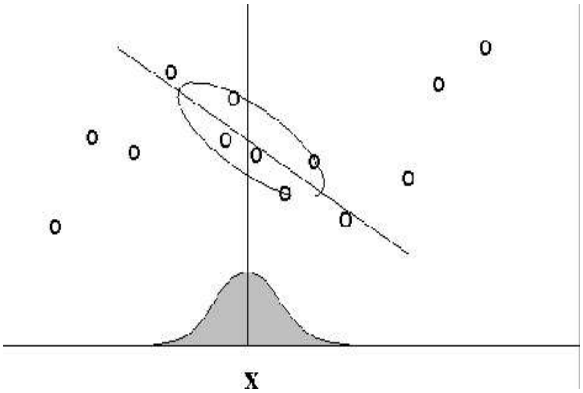

Abbildung 2: Locally Weighted Linear Regression

verdeutlicht die Gewichtung der Punkte. Die Vorteile der LWLR liegen vor allem darin, dass die Zielfunktion explizit angenährt wird.

Auch bei dieser Methode gehen keine Daten verloren, da auch hier neu klassifizierte Punkte einfach den Trainingsdaten hinzugefügt werden, aber sie hat ebenfalls den Nachteil das uninteressante Attribute mit entscheidend sind. Sie ist ebenfalls eine Lazy-Method, also langsam zur Anfragezeit, dies führt vor allem zu Problemen, wenn komplexere Funktionen zur Annährung verwendet würden. Aus eben diesem Grund beschränkt man sich in der Praxis häufig auf lineare oder quadratische Funktionen.

5 Radial Basis Functions

Radial Basis Functions versuchen durch Kombination verschiedener distanzgewichteter Funktionen die gesamte Zielfunktion anzunähren. Die Gesamtfunktion lässt sich also als Summe der Teilfunktionen schreiben.

$$\hat{f}(x) = \omega_0 + \sum_{u=1}^{k} \omega_u K_u(d(x_u, x))$$

Sie besteht aus einer einfachen Gewichtung und der Summe verschiedener Kernelfunktionen K, die abhängig vom Abstand abnehmen. So kann ein Wert über alle Funktionen angenährt werden, da auch hier wieder das Prinzip des Distance-Weighted NN vorkommt und weit vom Punkt entfernte Kernel-Funktionen nicht mehr spührbar in die Berechnung eingehen.

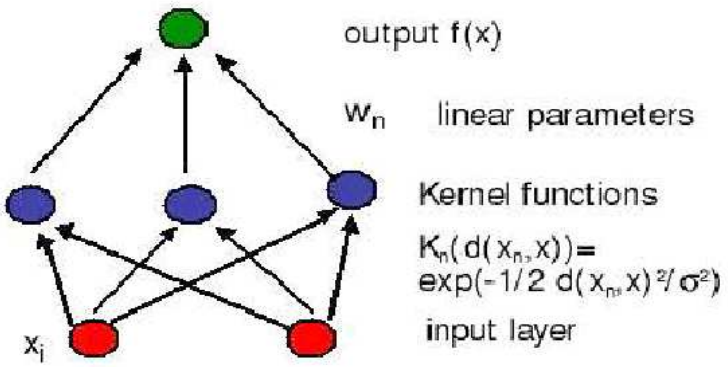

Abbildung 3: Radial Basis Functions

Die RBF sind somit sehr nah verwand mit der Locally Weighted Regression, haben aber noch den Vorteil, dass Sie "ëager" und keine Lazy Methods sind. Die Hauptrechenzeit wird also bereits vor Auftreten eines neuen Falls ausgeführt.

Die Nachteile dieser Funktion sind, dass es nicht einfach ist, die Kernelfunktionen richtig zu wählen oder ihre Gewichtung zu trainieren.

6 Case Based Reasoning

Einen gewissen Sonderfall der Nonparametric Techniques stellt Case Based Reasoning dar. Der stärkste Unterschied besteht in der Speicherung der Daten. Diese werden nicht mehr im n-dimensionalen Raum dargestellt, sondern indem man ihre Struktur und Funktionsweise beschreibt. Ein Problem wird gelöst, indem man es auf bereits bekannte Probleme zurückführt oder versucht, bekannte Lösungen zu kombinieren.

Ein Beispiel für die Anwendungsmöglichkeiten des Case Based Reasoning stellt das CADET System dar. Es kann zur Erstellung von Wasserleitungssystemen verwendet werden. Eine Rohrkonstruktion wird zum einen in ihrem Aussehen beschrieben zum anderen in ihrer Funktionsweise. Wird z.B. einem T-Stük von 2 Seiten Wasser zugeführt, so ergänzt sich die Menge additiv am Ausgang. Das CADET System besteht aus über 75 solcher Trainingsfälle und kann mit deren Hilfe auch komplexe Anforderungen bewältigen.

Der gesamte Ablauf von CBR wird am besten durch unten stehendes

Diagramm verdeutlicht.

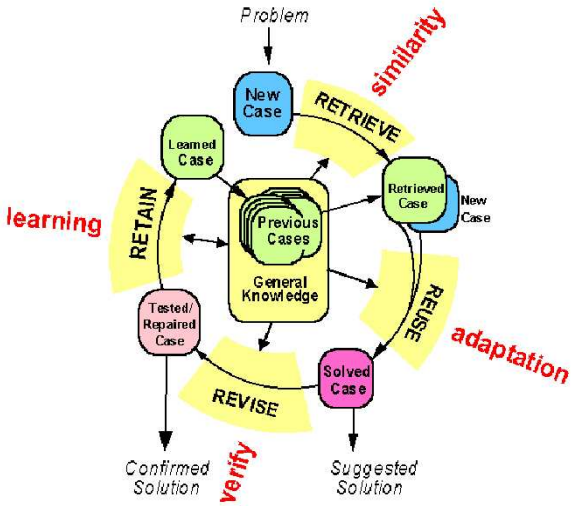

Abbildung 4: Case Based Reasoning

Das Diagramm zeigt den Ablauf des Case Based Reasoning. Zu einem neuen Problem sucht man zuerst (ä)hnliche F(ä)lle in der Datenbank der bisherigen Best(ä)nde um sie dann auf das gestellte Problem anzupassen und wiederzuverwenden. Dies f(ü)hrt zu einem L(ö)sungsvorschlag der durch (Ü)berpr(ü)fung zu einer best(ä)tigten L(ö)sung wird, und im Lernschritt dann den bekannten L(ö)sungen hinzugef(ü)gt werden kann.

7 Learning Vector Quantization

Die Learning Vector Quantization will ich hier kurz ansprechen, da Sie eine sehr große Ähnlichkeit mit den bisher vorgestellten Modellen hat, obwohl sie nicht zu den nichtparametrischen Techniken zhlt.

Die Daten können wieder als Punkte im Raum dargestellt werden und es besteht das Problem der Klassifizierung. Ein Unterschied zu den bisherigen Methoden ist, das eine Klasse explizit durch mehrere Prototypen definiert wird. Diese dienen dann zur Klassifizierung neuer Punkte.

LVQ1 lernt dabei dadurch, dass sobald ein neuer Punkt eingestuft wurde, dieser den ihm am nächsten liegenden Prototypen seiner Klasse um einen gewissen Prozentsatz zu sich hinzieht. LVQ2 würde noch Prototypen der anderen Klassen abstoßen. LVQ3 behandelt einige dabei auftretenden Probleme. Zur Verdeutlichung hier noch eine Grafik. Die 2 Klassen werden durch

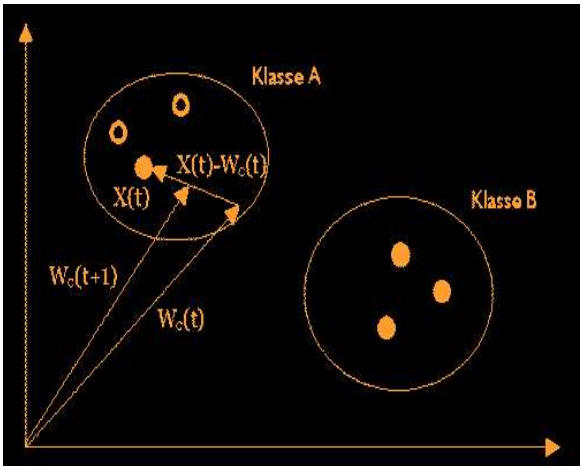

Abbildung 5: LVQ1

jeweils 3 Prototypen definiert, der neue Punkt würde den nächstliegenden zu sich hin verschieben.

8 Beispiele

8.1 Benchmark für Spam Filter

Ein mögliches Einsatzgebiet der hier vorgestellten Techniken stellen Spam filter dar. Um die Konkurenzfähigkeit zu zur Zeit gängigen Methoden zu testen wurde ein umfangreicher Testkörper zusammengestellt. Er besteht aus 2412 regulären Mails einer Linguistik-Newsgroup und 481 Spam Nachrichten, und stellt so einen Anteil von 16,6% Spam Nachrichten, was ihn auch mit früheren Tests vergleichbarer macht.

Aufgeteilt werden diese Nachrichten gleichmässig in 10 Teilgruppen, von denen 9 zum Training der Algorithem verwendet werden. Dieser Spam Corpus ist bekannt unter dem Namen LingSpam und auch frei herunterzuladen. [2]

Verglichen wird hier der TiMBL Algorithmus, eine Modifikation des k-NN, mit dem Naive Baysian und den Standard Outlook Pattern, die anhand starrer Regeln entscheiden. Der TiMBL Algorithmus ist im wesentlichen identisch mit dem k-NN, nur werden bei ihm nicht die k-nächsten Nachbarn zur Entscheidungsfindung benutzt sondern alle Nachbarn die sich in k-nächster Umgebung befinden, also deren Abstand vom Anfragepunkte kleiner k ist. Diese Definition kann zwar zu einer sehr großen Anzahl mit einfließender Daten führen, ist wohl aber am geeignetsten für dieses spezielle Problem.[6]

Die Bewertung erfolgt anhand einer Total Cost Ratio. Um diese zu definieren muss man zuerst mögliche Fehlerquoten definieren.

$$Acc = \frac{N_{L\to L} + N_{S\to S}}{N_L + N_S} \qquad Err = \frac{N_{L\to S} + N_{S\to L}}{N_L + N_S}$$

Der Fehler im Allgemeinen ist definiert durch die Anzahl der Spam Nachrichten die den Filter passieren, also als legal eingestuft werden, plus die Anzahl der Nachrichten, die fälschlicherweise vom Filter geblockt werden. Natürlich relativ zur Gesamtmenge der Nachrichten.

Nun ist es aber sinnvoll hier noch eine Wertung einzuführen da es um ein vielfaches schlimmer ist, normale Emails zu blocken, als Spam Mails zu übersehen. Wir führen also die Gewichtungsvariable Lambda ein und wichten somit die normalen Mails stärker, was zu den gewichteten Raten führt.

$$WAcc = \frac{\lambda N_{L\to L} + N_{S\to S}}{\lambda N_L + N_S} \qquad WErr = \frac{\lambda N_{L\to S} + N_{S\to L}}{\lambda N_L + N_S}$$

Um einen Vergleich zu bekommen führen wir jetzt noch eine Baseline ein, die Baseline ist gleichbedeutend mit einer Auswertung ohne jeglichen Filter, der Fehler besteht hier also einfach aus allen Spam Nachrichten, da auch gar keine normale Nachricht geblockt werden kann.

$$WAcc^b = \frac{\lambda N_L}{\lambda N_L + N_S} \qquad WErr^b = \frac{N_S}{\lambda N_L + N_S}$$

Als TCR können wir also jetzt das Verhältnis von dieser Baseline zum gewichteten Fehler definieren, es ist leicht ersichtlich das eine möglichst hohe TCR angestrebt wird da diese gleichbedeutend mit einem niedrigen gewichteten Fehler ist.

$$TCR = \frac{WErr^b}{WErr} = \frac{N_S}{\lambda N_{L\to S} + N_{S\to L}}$$

Bei der Auswertung erkennt man, dass zwar der Naive Baysian Filter nahezu immer leichte Vorteile hat, der TiMBL aber gut mithalten kann und beide die Outlook Patterns klar übertreffen, welche bei höherem Lambda auch schnell unter die Baseline von 1 fallen, was bedeutet das Problem für den Anwender sogar noch verschlechtern. Im Verhältniss zum Programmier- und

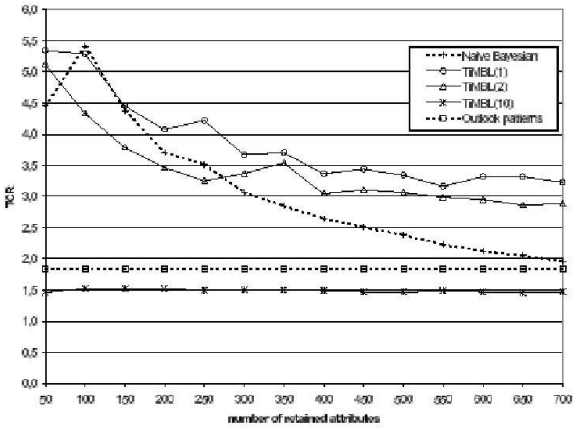

Figure 1. TCR scores for λ=1

Abbildung 6: Teilergebniss Benchmark

Rechenaufwandt stellt der NN also eine gute Alternative zum Bayesian Filter dar.

In einem weiteren Testsetup wurde der Einfluss verschiedener Attribut- und Abstandsgewichtungen auf das Testergebniss des TiMBL getestet, um so die optimale Konfiguration für das Problem des Spamfilters zu erhalten. Es wurden die Attribute über ihren Information Gain(IG) bemessen und nur die x am höchsten eingestuften Attribute zum Testen verwendet [7]. Des weiteren wurden die Attribute nach IG gewichtet und mit einer normalisierten Form, der Gain Ratio (GR), und der ungewerteten Form verglichen.

Hier ist die Methode über IG vor allem bei größeren Attributmengen klar im Vorteil und so wird sie im weiteren verwendet.

Als nächstes werden Distanzgewichtungen verglichen und nach Testauswertung eine Funktion mit $\frac{1}{d^3}$ gewählt. Im letzten Optimierungsschritt werden dann wieder die verschiedenen k-NN miteinander verglichen und so die optimale Gesamtkonfiguration gefunden.

Weiterführende verwandte Links:

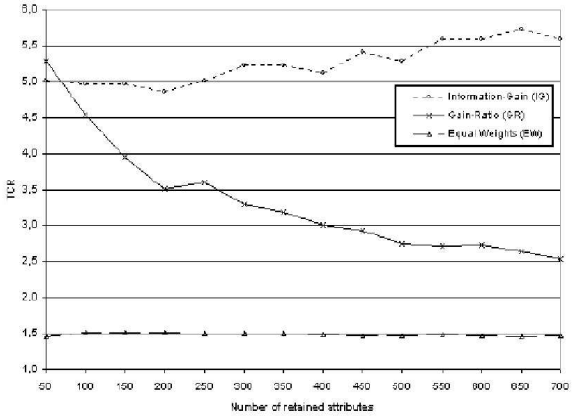

Figure 1: *TCR* of 10-NN for $\lambda = 1$ and three attribute-weighting functions.

Abbildung 7: IG / GR

A Decision Tree based Spam Filtering Agent [3]
Stacking classifiers for anti-spam filtering of e-mail[4]
Automatic Spam Detection as a Text Classification Task[5]

8.2 Schrifterkennung mit k-NN

Ein anderes Einsatzgebiet für den k-NN in dem er zum Teil noch bessere Ergebnisse erzielen kann ist die Schrifterkennung. In einem besonderen Verfahren wird ein Trainingsbuchstabe nicht nur in seiner Standardform gespeichert, sondern ebenfalls um eine geringe Gradzahl nach links und rechts gedreht.

Die so entstehende Kurve im mehrdimensionalen Raum wird ebenfalls mit neu erstellten Kurven vom Anfragebuchstaben verglichen und als Entscheidungskriterium der kürzeste Abstand zweier Tangenten gewählt. Diese Methode erzielt sehr gute Trefferquoten bei der Texterkennung.

8.3 Spezielle Hochburgen der NPs

Für spezielle Fälle wie einfache Bildklassifizierung von Satellitenaufnahmen sind spezielle k-NN Methoden immer noch unschlagbar.

Ebenfalls sehr gute Ergebnisse erziehlt der Nearest Neighbor mit stark verrauschten Daten bei genügend großer Trainingsanzahl.

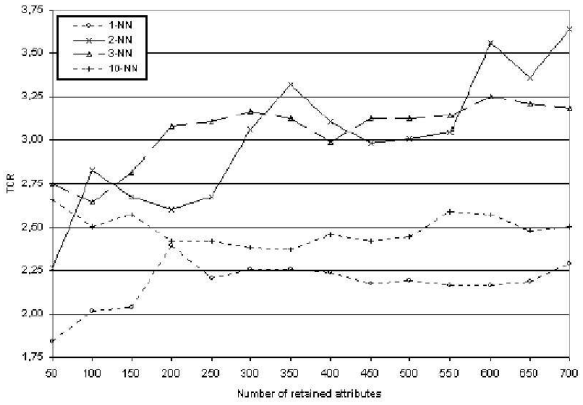

Figure 5: *TCR* of *k*-NN for $\lambda = 9$ and different *k* values (using *IG* for attribute weighting and $1/d^3$ for distance weighting).

Abbildung 8: Vergleich mit optimalen Wichtungen

9 Zusammenfassung

Nonparametric Techniques stellen trotz einer Vielzahl modernerer Technik auch in der heutigen Zeit noch echte Alternativen dar. Viele Methoden sind so gut erforscht und optimiert, dass sie in ihrer Effizienz kaum zu übertreffen sind. Vorallem da sie leicht zu implementieren sind erfreuen sie sich einer grossen Beliebtheit, auch in aktuellen Programmen, um z.B. Benutzervorlieben zu erfassen. Im Allgemeinen kann man sagen, dass Sie vor allem bei geringer Attribut- und somit Dimensionszahl Nonparametric Techniques immer noch eine echte Alternative sind, die auch gerne gewählt wird.

Literatur

[1] http://www.citeseer.nj.nec.com/palau97labeled.html

[2] http://www.aueb.gr/users/ion/lingspam_public.tar.gz

[3] http://www.cs.mu.oz.au/481/2001_projects/gntr/index.html

[4] http://www.arxiv.org/pdf/cs.CL/0106040

[5] http://www.faure.iei.pi.cnr.it/~fabrizio/Smadja.pdf

[6] http://citeseer.nj.nec.com/androutsopoulos00learning.html

[7] http://citeseer.nj.nec.com/512377.html