Ranking von Schach-Evaluationsfunktionen

Ranking of Evaluation Functions in Chess
Bachelor-Thesis von Denny Dittmar
Mai 2011
Ranking von Schach-Evaluationsfunktionen
Ranking of Evaluation Functions in Chess

Vorgelegte Bachelor-Thesis von Denny Dittmar

Gutachten: Johannes Fürnkranz

Tag der Einreichung:
Erklärung zur Bachelor-Thesis

Hiermit versichere ich, die vorliegende Bachelor-Thesis ohne Hilfe Dritter nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den 25. Mai 2011

(Denny Dittmar)
Zusammenfassung

Inhaltsverzeichnis

1 Einführung und Grundlagen ................................................. 4
  1.1 Motivation und Zielsetzung ........................................... 4
  1.2 Ranking im maschinellen Lernen ................................... 4
  1.3 Evaluationsfunktionen bei Schachprogrammen ................. 4
  1.4 Problemspezifikation für diese Arbeit ......................... 6
  1.5 Spielstärkenvergleiche von Evaluationsfunktionen durch Schachspiele 7

2 Grundlagen ........................................................................... 9
  2.1 Grundlagen von Ranking-SVMs ..................................... 9
     2.1.1 Support Vector Machines ...................................... 9
     2.1.2 Ranking-SVMs ..................................................... 12
  2.2 Aktives Lernen ............................................................ 13
  2.3 Genetische Algorithmen ............................................... 13

3 Beschreibung der angewandten Ranking-Verfahren ............... 15
  3.1 Distanzmaß für Evaluationsvektoren ............................ 15
  3.2 Ranking-Verfahren 1 .................................................... 15
     3.2.1 Erweiterung des Ranking-SVM-Verfahrens durch aktives Lernen 16
  3.3 Ranking-Verfahren 2 .................................................... 17
  3.4 Ranking-Verfahren 3 .................................................... 18

4 Versuchsdurchführung und Evaluation der Ergebnisse .......... 20
  4.1 Programm zur Versuchsdurchführung ............................ 20
     4.1.1 Schachprogramm ................................................ 20
     4.1.2 Schachprogramminterface .................................... 20
     4.1.3 SVM-Ranking ..................................................... 21
     4.1.4 Genetische Algorithmen ...................................... 21
  4.2 Durchführung und Auswertung der Tests ....................... 21

5 Zusammenfassung ............................................................. 27
1 Einführung und Grundlagen

1.1 Motivation und Zielsetzung


Das Ziel dieser Arbeit soll es somit sein, mit verschiedenen Ansätzen zu prüfen, inwieweit ein solches Modell aus Trainingsdaten gewonnen werden kann, d.h. wie allgemein und zuverlässig es ist. Als Trainingsdaten sollen dafür Resultate von paarweisen Schachspielvergleichen dienen.

1.2 Ranking im maschinellen Lernen

Unter einem Ranking-Problem wird im maschinellen Lernen das Ziel verstanden, ein Vorhersagemodell für eine Reihung bzw. Ordnung (Ranking) von Objekten zu lernen und anzuwenden. Ein Objekt wird dabei in der Regel durch ein Label oder einen Vektor repräsentiert, dessen Komponenten bestimmte numerische Features des Objektes sind. Als Trainingsdaten für die Erstellung des Modells dienen Ordnungen von Objekten, die in Form von Objektlisten und/oder Objektpaaren (siehe beispielsweise [1]) gegeben sein können und für deren enthaltene Elemente die Ordnungen jeweils bekannt sind.

Unter Anwendung des Modells soll dann eine Testmenge von Objekten auf eine Reihung dieser Objekte abgebildet werden, in der sich möglichst das Ranking widerspiegelt, das auch vorliegt. Abbildung 1 verdeutlicht dieses Prinzip von Training und Test für Ranking-Probleme im maschinellen Lernen.


1.3 Evaluationsfunktionen bei Schachprogrammen

In diesem Kapitel soll darauf eingegangen werden, was Evaluationsfunktionen bei Schachprogrammen sind und wie sie verwendet werden. Dazu erfolgt hier eine Kurzeinführung in die Funktionsweise von Schachprogrammen.

Die Funktionsweisen für die Ermittlung des nächsten Zuges in einer Stellung folgen bei modernen Schachprogrammen meist dem gleichen Basisprinzip. Das sieht zunächst so aus, dass es einen Zuggenerator gibt, mit dessen Hilfe alle möglichen Züge in einer Stellung ermittelt werden können. Damit kann dann ein Suchbaum aufgebaut werden, dessen Knoten für
Abbildung 1: Ranking im maschinellen Lernen mit mehreren Trainingsinstanzen und einer Testinstanz [2]

Spielstellungen stehen. Die Wurzel des Baumes steht für die Ausgangstellung und ein Kindknoten eines beliebigen Knotens in dem Suchbaum stellt eine mögliche Nachfolgestellung dar, die durch Ausführung eines einzelnen regulären Zuges aus der Stellung dieses Knotens entsteht.


Der Suchbaum wird beim Minimax durch eine Tiefensuche aufgebaut, die von einem Knoten aus nur dann noch tiefer sucht, wenn dort kein Spielende ist oder wenn die vorgegebene Tiefe noch nicht erreicht wurde.


Die Bewertungen der Blattknoten erfolgt dann durch eine Evaluationsfunktion, die bestimmte materielle und positionelle Stellungsmerkmale für beide Spieler in der Spielsituation auf einen Score abbildet. Diese Stellungsmerkmale beinhalten beispielsweise die Anzahl der Bauern der jeweiligen Spieler und werden jeweils durch einen numerischen Wert repräsentiert. Die Evaluationsfunktion ist dann meist gegeben durch eine Linearkombination dieser Merkmale, wobei die Koeffizienten die Gewichte sind, die charakteristisch für die Evaluationsfunktion sind, also \( \text{eval}(s) = \sum_i w_i \ast f_i \) mit \( s \) als Stellung, mit \( w_i \) als \( i \)-tes Gewicht und \( f_i \) als das \( i \)-te Merkmal. Der Betrag eines Gewichts für ein Merkmal orientiert sich an der Bedeutung des Merkmals. Das Vorzeichen ist positiv, wenn es gut für den Max-Spieler ist, ansonsten negativ. Beispielsweise hat somit die Dame des Max-Spielers ein weitaus höheres Gewicht als der Bauer des Max-Spielers, während die Dame des Min-Spielers ein weitaus geringeres Gewicht hat als ein Bauer des Min-Spielers.

1.4 Problemspezifikation für diese Arbeit


Die Menge der Evaluationsfunktionen, die hier überhaupt betrachtet werden soll, wird im Voraus aus Komplexitätsgründen begrenzt. Das heißt Trainings- und Testpaare für die Rankingverfahren werden dann nur durch Evaluationsfunktionen aus dieser Menge gebildet.
Dazu werden die Anzahl und die Typen der Merkmale, die veränderbar sein sollen, sowie der Wertebereich ihrer Gewichte reduziert. Eine zu hohe Anzahl an veränderlichen Merkmalen könnte den Lernprozess für den Rahmen dieser Arbeit als zu aufwendig gestalten und eine zu geringe würde dazu führen, dass man die Resultate nicht verallgemeinern könnte auf Evaluationsfunktionen mit einer deutlich höheren Anzahl an Merkmalsgewichten. Die Anzahl der Gewichte wird auf daher auf 10 festgelegt.

 Auch die Wertebereiche für die Merkmalsgewichte werden aus Komplexitätsgründen begrenzt. Die oberen und unteren Grenzen für den Wertebereich eines Merkmalgewichts werden mit 75% bzw. 125% des Standardwertes des jeweiligen Gewichts festgesetzt. Mit Standardwert ist der Wert gemeint, welchen das verwendete Schachprogramm in den Default-Einstellungen benutzt.

Da alle Evaluationsfunktionen Linarkombinationen verschiedener Merkmalwerte sind, lassen sie sich jeweils auch durch einen Vektor \( \vec{v} = (w_1, ..., w_n)^T \) repräsentieren, welcher die einzelnen Gewichte \( w_i \) für die entsprechenden Merkmale der zu bewertenden Situation enthält, wobei wir zudem nur die Gewichte berücksichtigen brauchen, die veränderbar sein sollen, sodass sich für diesen Fall \( \vec{v} = (w_1, ..., w_{10})^T \) ergibt.

Die Menge aller Evaluationsvektoren, die man auf diese Weise erhält, sei mit \( V \) bezeichnet. Ziel ist es nun, für je 2 beliebige Evaluationsfunktionen aus \( V \) vorauszusagen, mit welcher das Schachprogramm besser spielt. Mit „besser spielt“ ist gemeint, mit welcher Evaluationsfunktion eine Instanz des verwendeten Schachprogramms gegen eine andere Instanz, das die andere Evaluationsfunktion verwendet, ein einzelnes Spiel wahrscheinlich gewinnen würde, d.h. wenigstens mit einer Wahrscheinlichkeit von über 50%.

Demnach soll eine asymmetrische binäre Relation \( \succ \subset V \times V \) gelernt werden, für die für 2 beliebige Evaluationsvektoren \( \vec{v}_i, \vec{v}_j \in V \) gilt, dass \( \vec{v}_i \succ \vec{v}_j \), d.h. \( \vec{v}_i \) eine höhere Spielstärke als \( \vec{v}_j \) hat. Man kann zudem annehmen, dass \( \succ \) transitiv ist und somit eine Striktordnung darstellt. Die durch die Voraussagen induzierte binäre Relation sei mit \( \succ^* \) bezeichnet. Es soll dann also für möglichst viele Evaluationsvektorenpaare \( \vec{v}_i, \vec{v}_j \) gelten, dass \( \vec{v}_i \succ^* \vec{v}_j \iff \vec{v}_i \succ \vec{v}_j \).

Als Trainingsinstanzen für die hier getesteten Ranking-Verfahren sollen Paare von Evaluationsvektoren \( \vec{v}_i, \vec{v}_j \) dienen, für die jeweils bekannt ist, welche Spielstärkenrelation vorliegt, d.h. ob \( \vec{v}_i \succ \vec{v}_j \) oder \( \vec{v}_j \succ \vec{v}_i \) gilt. Das Ermitteln dieser Relation soll mittels mehrerer Schachspiele erfolgen.

### 1.5 Spielstärkenvergleiche von Evaluationsfunktionen durch Schachspiele

Das Ziel ist es, für 2 Evaluationsfunktionen festzustellen, welche mit einer Wahrscheinlichkeit von mindestens 50% ein einzelnes Spiel gegen die andere gewinnt und damit als besser gilt. Dies soll durch Schachspiele erkannt werden, bei der 2 Instanzen eines Schachprogrammes mit jeweils einer der beiden Evaluationsfunktionen mehrere Partien gegeneinander spielen.

Nimmt man an, dass eine Reihe von Spielen zwischen beiden Instanzen durchgeführt wurde und die Ergebnisse der einzelnen Spiele vorliegen, so stellt sich die Frage, ob man eine von beiden Evaluationsfunktionen nun als die bessere annehmen kann.

Es soll hier zwischen 2 Möglichkeiten bzw. Hypothesen unterschieden werden. Zum einen ist das die Nullhypothese \( (H_0) \), die besagt, dass beide Evaluationsfunktionen gleich gut sind und zum andere eine Alternativhypothese \( (H_1) \), wonach diejenige mit mehr Siegen die bessere ist.
Es soll sich nun also für eine von beiden Hypothesen entschieden werden.
Als Traniingsinstanzen werden nur Paare verwendet, für die $H_1$ als gültig angenommen wurde.
Es ist von Interesse, dass die Wahrscheinlichkeit, dass $H_1$ angenommen wird, obwohl $H_0$ zutrifft, so gering wie möglich ist. Diese Art von Irrtum nennt man in der Statistik auch Fehler 1. Art.

Hierfür kann man einen sogenannten Vorzeichentest anwenden.
Dazu wird die Anzahl der durchgeführten Partien $N$, die nicht unentschieden endeten, sowie die Siege $k$ der Evaluationsfunktion, welche in diesen weniger Siege erzielte, betrachtet. Je geringer $k$ ist, desto eher ist $H_0$ falsch und $H_1$ richtig. Für ein gegebenes $N$ soll demnach bis zu einem maximalen $k$ die Nullhypothese $H_0$ verworfen werden.

Die Wahrscheinlichkeit, die Nullhypothese $H_0$ dabei zu verwerfen, obwohl sie zutrifft, ergibt sich wie folgt:
Unter der Annahme, dass $H_0$ zutrifft, gewinnt jede Evaluationsfunktion mit einer Wahrscheinlichkeit von 50%. Die Wahrscheinlichkeit, dass die Evaluationsfunktion mit weniger Siegen bei dieser Siegswahrscheinlichkeit genau $k$ Siege erzielt, ist gegeben durch die Binomialverteilung $p(k) = \binom{N}{k} (\frac{1}{2})^k (1 - \frac{1}{2})^{N-k} = \frac{1}{2^N} \binom{N}{k}$. Entsprechend ist die Wahrscheinlichkeit, dass sie maximal $k$ Siege erzielt bzw. dass $H_0$ falschlicherweise verworfen wird, gegeben durch $p_{\text{max}}(k) = \frac{1}{2^N} \sum_{i=1}^{k} \binom{N}{i}$.

Es sollen nun solange Schachspiele durchgeführt werden, bis ein Zwischenresultat (bestehend aus $N$ und $k$) erzielt wurde, bei dem diese Irrtumswahrscheinlichkeit möglichst gering ist, sodass man die Evaluationsfunktion mit mehr Siegen auch mit einer entsprechenden Sicherheit als die bessere annehmen kann. Die zu unterschreitende Grenze der Irrtumswahrscheinlichkeit, welche auch als Signifikanzniveau bezeichnet wird, wird hier intuitiv mit 15% festgelegt.
Da es auch sein kann, dass dieses Unterschreiten erst sehr spät eintritt und jedes Schachspiel mit Zeitaufwand verbunden ist, wird eine Obergrenze für die durchzuführenden Spiele festgelegt, die auch Spiele mit uneentschiedenem Ausgang beinhaltet.
Ist durch ein Zwischenresultat absehbar, dass in einer Vergleichsreihe nicht mehr bis zur Durchführung von 20 Spielen das benötigte Signifikanzniveau für die Verwerfung der Nullhypothese unterschritten werden kann, wird der Vergleich abgebrochen und keine der beiden Evaluationsfunktionen wird als besser angenommen und somit auch nicht in der Trainingsmenge berücksichtigt.

Daher wird bei der Durchführung einer Vergleichsreihe nach jedem Spiel immer die Farbe gewechselt, damit jede Evaluationsfunktion jede Farbe in etwa gleich oft hatte mit einer maximalen Differenz von 1.
2 Grundlagen

Dieses Kapitel soll sich weitergehenden Grundlagen widmen.

2.1 Grundlagen von Ranking-SVMs

In diesem Abschnitt soll die Funktionsweise von Ranking-SVMs erläutert werden, die ebenfalls als Ranking-Verfahren verwendet werden sollen. Zum besseren Verständnis wird dazu zuerst auf standardmäßige SVMs eingegangen.

2.1.1 Support Vector Machines

Support Vector Machines oder kurz SVMs sind Klassifikationssverfahren für 2 Klassen. Sie dienen der Vorhersage, welcher von 2 möglichen Klassen ein gegebener Vektor angehört, wobei mittels Trainingsdaten in Form einer Vektorenmenge, für deren Elemente die Klasse bekannt ist, versucht wird ein Modell zu lernen.

Dazu wird versucht die Trainingsvektoren durch eine Hyperebene so zu trennen, sodass auf jeder Seite der Hyperebene nur Vektoren einer Klasse liegen, wobei vorausgesetzt wird, dass diese auch durch eine Hyperebene linear trennbar sind.

Eine Trainingsinstanz ist gegeben durch ein Paar \((x_i, y_i)\), wobei \(x_i \in \mathbb{R}^n\) ein Vektor und \(y_i \in \{-1, 1\}\) seine Klasse ist. Die Hyperebene ist definiert durch 2 Vektoren \(\vec{w}, \vec{b} \in \mathbb{R}^n\), so dass \(\text{sgn}(\vec{w}^T x_i + b)\) die Seite der Hyperebene angibt, auf der der Vektor \(x_i\) liegt. Für die Klassen der Trainingsdaten soll gelten \(\forall i \ y_i = \text{sgn}(\vec{w}^T x_i + b)\). Sind die Klassen linear trennbar, so gibt es im Allgemeinen mehrere Hyperebenen, für die dies erfüllt ist. Um jedoch auch die allgemeine Zuverlässigkeit für das Klassifizieren unbekannter Daten möglichst zu maximieren, soll für die Trennung der Klassen genau die Hyperebene gefunden und verwendet werden, welche den maximalen Abstand zu dem nächstliegenden Vektor hat, wobei dieser Abstand auch als Margin bezeichnet wird.

Der Abstand eines beliebigen Vektors \(x_i\) zur Hyperebene ist gegeben durch \(\frac{\vec{w}^T x_i + b}{||\vec{w}||_2}\). Dieser Abstand bleibt gleich, wenn \(\vec{w}\) und \(\vec{b}\) jeweils mit dem gleichen Faktor skaliert werden. Wird für \(\vec{w}\) und \(\vec{b}\) eine Skalierung gewählt, sodass der Abstand für den nächsten Punkt zur Hyperebene 1 ist, dann ist der Abstand für alle anderen Vektoren mindestens 1 und der Margin ist dann gegeben durch \(\frac{1}{||\vec{w}||_2}\). Alle Vektoren mit dem Abstand 1 bzw. -1 zur Hyperebene sind die Stützvektoren (Support Vectors), welche allein ausreichend sind, um die Hyperebene nach genannten Kriterien eindeutig zu bestimmen (siehe Abbildung 3).

Die Maximierung des Margins ist damit äquivalent zur Minimierung von \(\frac{1}{2}||\vec{w}||_2^2\).

Das Finden einer Hyperebene mit maximalem Margin lässt sich somit durch die Lösung des folgenden Optimierungsproblems erreichen:

\[
\begin{align*}
\arg\min_{\vec{w}, \vec{b}} & \quad \frac{1}{2}||\vec{w}||_2^2 \\
\text{s.t.} & \quad \forall i \ y_i(\vec{w}^T x_i + \vec{b}) \geq 1
\end{align*}
\]

Es gibt jedoch auch 2-Klassen-Mengen von Vektoren, die nur annährend linear trennbar sind. Zu diesem Zweck kann man auch einige Verletzungen der Constraints (siehe Abbildung 4) des

Optimierungsproblems zulassen, indem man es um sogenannte Schlupfvariablen $\xi_i$ erweitert. Für die Constraints muss dann nur noch gelten, dass $y_i(\bar{w}^T x + b) - 1 + \xi_i \geq 0$, wobei $\xi_i \geq 0$. Um die Verletzungen der Constraints nicht zu groß werden zu lassen, wird nun auch versucht, parallel zur Maximierung des Margins die Summe der Schlupfvariablen $C \sum_i \xi_i$ zu minimieren, welche mit einem Faktor $C$ gewichtet ist, der das Toleranzniveau für Constraintverletzungen bestimmt. Das erweiterte Optimierungsproblem mit Schlupfvariablen sieht damit folgendermaßen aus:

$$
\begin{align*}
\text{argmin}_{\bar{w}, \bar{b}} & \frac{1}{2}||\bar{w}||^2 + C \sum_i \xi_i \\
\text{s.t.} & \forall i \ y_i(\bar{w}^T x_i + \bar{b}) + \xi_i \geq 1, \xi_i \geq 0
\end{align*}
$$

Durch die Anwendung von Lagrange-Multiplikatoren lässt sich das Optimierungsproblem auch umformen, sodass man die sogenannte Duale Form erhält:

$$
\begin{align*}
\text{argmax}_\alpha & \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j (\bar{x}_i^T \bar{x}_j) \\
\text{s.t.} & \sum_i \alpha_i y_i = 0, 0 \leq \alpha_i \leq C
\end{align*}
$$

Für die Hyperebene gilt dann $\bar{w} = \sum_i \alpha_i y_i \bar{x}_i$ und die Klassifikation ist gegeben durch $y_j = \text{sgn}(\bar{w}^T \bar{x}_j + \bar{b}) = \text{sgn}(\sum_i \alpha_i y_i (\bar{x}_i^T \bar{x}_j) + \bar{b})$

Die meisten $\alpha_i$ haben dabei den Wert 0, nur für die Stützvektoren haben sie einen Wert ungleich 0.

Sind 2-Klassen-Mengen von Vektoren auch nicht annährend linear trennbar sind, gibt es die Möglichkeit, diese mit einer Transformationsfunktion $\phi: \mathbb{R}^n \rightarrow \mathbb{R}^m$, $m > n$ auf einen höherdimensionalen Raum abzubilden, in welchem diese transformatierten Vektoren dann linear trennbar werden(siehe Abbildung 5).

**Abbildung 5:** SVM: Lineare Trennung nur in höherdimensionalem Raum möglich [5]

Unter Verwendung der Duale Form ergibt sich damit:

$$
\begin{align*}
\text{argmax}_\alpha & \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j (\phi(\bar{x}_i))^T \phi(\bar{x}_j))
\end{align*}
$$
s.t. \( \forall i \sum a_i = 0, 0 \leq a_i \leq C \)

Die Klassifizierung erfolgt mit

\[
y_j = \text{sgn}(\vec{w}^T \phi(x_j) + b) = \text{sgn}(\sum_i a_i y_i (\phi(x_i)^T \phi(x_j))) + \vec{b}.
\]

Sowohl bei der Optimierung als auch bei der Klassifizierung kommen die Trainingsdaten hierbei nur in den Skalarprodukten \( \phi(x_i)^T \phi(x_j) \) vor. Statt die Transformationen und Skalarprodukte dabei immer explizit zu berechnen, was sehr aufwendig sein kann, lässt sich auch eine sogenannte *Kernel-Funktion K* verwenden, von der man weiß, dass \( K(x_i, x_j) = \phi(x_i)^T \phi(x_j) \) gilt und diese leichter zu berechnen ist. Man bezeichnet dieses Vorgehen auch als *Kernel-Trick*.

### 2.1.2 Ranking-SVMs


Dazu wird eine lineare Ranking-Funktion \( f \) gelernt, die jedem Vektor \( \vec{x}_i \) einen Score zuweist, durch den eine totale Ordnung auf der Menge aller Vektoren impliziert wird. Die Ranking-Funktion ist hierbei gegeben durch \( f(\vec{x}_i) = \vec{w}^T \vec{x}_i \), wobei \( \vec{w} \) der zu lernende Gewichtsvektor ist. Wenn für zwei Vektoren \( \vec{x}_i, \vec{x}_j \) der Vektor \( \vec{x}_i \) in der durch \( f \) gegebenen Ordnung vor dem Vektor \( \vec{x}_j \) liegt, so gilt demnach: \( f(\vec{x}_i) > f(\vec{x}_j) \) bzw. \( \vec{w}^T \vec{x}_i > \vec{w}^T \vec{x}_j \).

Grafisch lässt sich dieses Ordnen der Vektoren durch ihre Projektionen auf \( \vec{w} \) veranschaulichen (siehe Abbildung 6). In dem Beispiel werden die Vektoren, bezeichnet durch \( 1, 2, 3, 4 \), in der Reihenfolge (1, 2, 3, 4) sortiert und für \( \vec{w}_1 \) in der Reihenfolge (2, 3, 1, 4). Der Margin ist im Fall der Ranking-SVMs der kleinste Abstand, der zwischen 2 Vektoren liegt und ist gegeben durch \( \frac{1}{||\vec{w}||_2} \). Für die Gewichtsvektoren \( \vec{w}_1 \) und \( \vec{w}_2 \) im Beispiel ist dieser jeweils gegeben mit \( \delta_1 \) bzw. \( \delta_2 \).

Soll nun von einer gegebenen binären Ordnungsrelation \( R \), die die Trainingsmenge ist, gelernt werden, wird hierzu versucht, ein solches \( \vec{w} \) zu finden, sodass für möglichst viele Paare \( (\vec{x}_i, \vec{x}_j) \in R \Rightarrow \vec{w}^T \vec{x}_i > \vec{w}^T \vec{x}_j \) gilt.

Da dieses Problem sehr komplex (NP-hart) ist, wird die Lösung approximiert. Hierzu wird auf den normalen SVM-Algorithmus unter Verwendung von Schlupfvariablen \( \xi_{ij} \) zurückgegriffen, wobei versucht wird \( C \sum \xi_{ij} \) zu minimieren, um den Fehler auf den Trainingsdaten gering zu halten. Dies lässt sich ähnlich zu dem Vorgangen bei Standard-SVMs als folgendes Optimierungsproblem formulieren:

\[
\begin{align*}
\min \frac{1}{2}||\vec{w}||_2^2 + C \sum \xi_{ij} \\
\text{s.t. } \forall i,j \ (\vec{x}_i, \vec{x}_j) \in R \Rightarrow \vec{w}^T \vec{x}_i > \vec{w}^T \vec{x}_j + 1 - \xi_{ij} \\
\forall ij \ \xi_{ij} \geq 0
\end{align*}
\]

Durch die Minimierung von \( \frac{1}{2}||\vec{w}||_2^2 \), was der Maximierung des Margins \( \frac{1}{||\vec{w}||_2} \), entspricht, wird eine Generalisierung der Ranking-Funktion erreicht. \( C \) dient hierbei dann dem Trade-Off zwischen Generalisierung und dem Fehler auf den Trainingsdaten. Die Constraints des Optimierungsproblems \( \vec{w}^T \vec{x}_i > \vec{w}^T \vec{x}_j + 1 - \xi_{ij} \) lassen sich umformen zu \( \vec{w}^T (\vec{x}_i - \vec{x}_j) > 1 - \xi_{ij} \). Durch diese Umformung resultiert ein Optimierungsproblem, welches dem für Klassifikations-SVMs für paarweise Vektorendifferenzen entspricht. Da für die Lösung eines solchen Optimierungsproblems bereits Algorithmen existieren, kann auf diese hierfür zurückgegriffen werden.
Auch für Ranking-SVMs lässt sich der Vektor $\vec{w}$ als Linearkombination der Trainingsdaten darstellen. Analog zu normalen SVMs ergibt sich unter Verwendung einer Transformationsfunktion $\phi: \mathbb{R}^n \rightarrow \mathbb{R}^m$, mit $m > n$ dann $\vec{w} = \sum_i \alpha_i \phi(\vec{x}_i)$, wobei die Koeffizienten $\alpha_i$ aus der Lösung der dualen Form des Optimierungsproblems abgeleitet werden können. Die Ranking-Funktion ist dann gegeben durch $f(\vec{x}) = \vec{w}^T \phi(\vec{x}) = \sum_i \alpha_i \phi(\vec{x}_i)^T \phi(\vec{x}) = \sum_i \alpha_i K(\vec{x}_i, \vec{x})$. Folglich lassen sich für Ranking-SVMs auch nichtlineare Kernel verwenden.

Abbildung 6: verschiedene Reihungen für verschiedene Gewichtsvektoren [6]

2.2 Aktives Lernen

Im maschinellen Lernen gibt es Szenarien, in denen es mit Kosten verbunden ist, die Klasse eines Trainingsbeispiels zu ermitteln, da dies beispielsweise Zeit oder Expertenwissen erfordert. Insofern ist es hier durchaus lohnenswert, nur für die Trainingsbeispiele die Klasse zu ermitteln, welche bei Hinzufügung zur Trainingsmenge die Entwicklung eines geeigneten Modells in Hinblick auf Generalisierung am Besten vorantreiben. Aktives Lernen stellt eine Form des maschinellen Lernens dar, bei dem diese Idee in das Lernverhalten eines Lernens mit einbezogen wird, das heißt der Lerner sucht sich selbst aus, welche Trainingsbeispiele er seiner Trainingsmenge hinzufügen möchte.

Häufig läuft dieser Prozess so ab, dass der Lerner in jeder Runde auf Basis des aktuellen Modells neue Beispiele auswählt, diese zur Trainingsmenge hinzufügt und das neue Modell auf Basis der neuen Trainingsmenge ermittelt. Metriken für den Lernnutzen eines bestimmten Trainingsspiels orientieren sich zum Beispiel daran, mit welcher Sicherheit der Lerner dieses Beispiel mit dem aktuellem Modell klassifiziert oder wie weit die Bereiche mit Unsicherheit bzw. der Hypothesenraum für die einzelnen Klassifizierungsmöglichkeiten des Beispiels eingegrenzt würden. Für SVMs existieren hierfür beispielsweise Ansätze, wobei nur für solche Beispiele die Klasse abgefragt wird, deren Distanz zur aktuellen Hyperebene am geringsten ist, da so der Raum möglicher Hyperebenen am effizientesten eingegrenzt werden kann. [7]

2.3 Genetische Algorithmen

Genetische Algorithmen stellen eine Möglichkeit dar, um Optimierungsprobleme zu lösen, das heißt um für eine Gütefunktion $f: \mathbb{R}^n \rightarrow \mathbb{R}$ einen Vektor $v \in \mathbb{R}^n$ zu finden, sodass $f(v)$ einen
möglicherweise größten Wert annimmt, also $argmax_v f(v)$. Sie orientieren sich dabei an natürlichen, darwinistischen Prinzipien, wonach die Angepasstheit von Lebewesen bzw. Individuen einer Population an ihre Umwelt auf ihre Überlebens- und Fortpflanzungschancen zurückgeführt wird, die mit der Angepasstheit steigen, wobei Individuen auch mutieren und bei der Fortpflanzung die Gene der Eltern weitergegeben werden.


Auf diese Weise nähert man sich einem meist lokalen Optimum. Als Abbruchkriterium wird in der Regel eine vorgegebene Anzahl an Generationen oder eine Vorgabe für die Güte der besten Individuen verwendet.

![Abbildung 7: Anwendung genetischer Operatoren: Selektion, Crossover, Mutation [3]](image-url)
3 Beschreibung der angewandten Ranking-Verfahren

In diesem Kapitel werden die verschiedenen Verfahren zum Lernen des Vorhersageprädikats \( \vec{v}_i \succ \vec{v}_j \) beschrieben, für das für möglichst viele Paare von Evaluationsvektoren \((\vec{v}_i, \vec{v}_j)\) gelten soll:

\[ \vec{v}_i \succ \vec{v}_j \iff \vec{v}_i \succ \vec{v}_j. \]

Dabei handelt es sich um 1 Verfahren, welches auf einer Ranking-SVM basiert, sowie 2 weitere Ansätze, um auch verschiedene Verfahren gegenüberstellen zu können.

3.1 Distanzmaß für Evaluationsvektoren

Für einige der angewandten und später beschriebenen Verfahren wird ein Distanzmaß für Evaluationsvektoren benötigt. Eine direkte Anwendung z.B. der euklidischen Distanz ist hierfür ungünstig, da beispielsweise eine Veränderung des Merkmalgewichts für einen Bauern um einen bestimmten absoluten Wert die Spielstärke einer Evaluationsfunktion im Durchschnitt stärker beeinflusst als die Veränderung des Gewichts für eine Dame um denselben Wert. Es wird daher angenommen, dass die Spielstärkenbeeinflussung durch eine absolute Wertveränderung eines Merkmalgewichts in etwa umgekehrt proportional zum Standardwert eines Merkmalgewichts ist, d.h. der Wert, welcher im verwendeten Schachprogramm für dieses Gewicht standardmäßig verwendet wird.

Daher wird als Distanzmaß eine gewichtete euklidische Distanz verwendet, bei der alle Merkmalsgewichte noch einmal durch ihr Standardgewicht geteilt werden.

Als Distanzmaß \( \text{dist} \) ergibt sich für 2 Evaluationsfunktionen \( \vec{v}_i, \vec{v}_j \) somit:

\[ \text{dist}(\vec{v}_i, \vec{v}_j) = \sqrt{\sum_{k=1}^{10} \left( \frac{\vec{v}_{i,k} - \vec{v}_{j,k}}{s_k} \right)^2}, \]

wobei \( s_k \) der Standardwert des k-ten Evaluationsgewichts ist.

Wenn in den folgenden Abschnitten von Distanz die Rede ist, so bezieht sich das auf dieses Maß.

3.2 Ranking-Verfahren 1

Es soll insbesondere eine Ranking-SVM getestet werden, um Spielstärkenrelationen von Evaluationsfunktionspaaren vorherzusagen.

Eine Ranking-SVM trainiert auf Basis der Trainingsdaten eine Ranking-Funktion \( f(\vec{x}) = \vec{w}^T \phi(\vec{x}) \), die Vektoren einen Score zuweist, aus welchem sich das Ranking dieser ergibt. Dies erfolgt, wie bereits in Kapitel 2 gezeigt, durch Lösung eines Optimierungsproblems, dass an Constraints gebunden ist, die aus den Trainingsdaten abgeleitet werden. Die Trainingsmenge \( Tr \) besteht in diesem Fall aus Paaren von Evaluationsvektoren \((\vec{v}_i, \vec{v}_j)\), wobei \( \vec{v}_i, \vec{v}_j \) zufällig gewählt wurden und aufgrund von durchgeführten Schachspielen \( \vec{v}_i \succ \vec{v}_j \) als wahrscheinlich gilt, sodass \( Tr \subset \succ \) angenommen werden kann.

Die Voraussage, welcher von 2 Evaluationsvektoren besser ist, erfolgt bei diesem Ranking-Verfahren bei gegebener Ranking-Funktion \( f \) demnach wie folgt:

\[ \vec{v}_i \succ \vec{v}_j \iff f(\vec{v}_i) > f(\vec{v}_j) \]

Da für Ranking-SVMs auch die Verwendung von nichtlinearen Kernels möglich ist, wird versucht
neben dem standardmäßigen linearen Kernel auch einige andere populärere zu verwenden und zu testen. Das betrifft einen Polynom-, Radial-Basis-Function- und Sigmoid-Kernel, welche wie nachfolgend definiert sind:

Polynom-Kernel: \( K(\vec{x}_i, \vec{x}_j) = (\vec{x}_i^T \vec{x}_j + c)^d \)

RBF-Kernel: \( K(\vec{x}_i, \vec{x}_j) = \exp(-\gamma ||\vec{x}_i - \vec{x}_j||^2_2) \)

Sigmoid-Kernel: \( K(\vec{x}_i, \vec{x}_j) = \tanh(\kappa \vec{x}_i^T \vec{x}_j + c) \)

### 3.2.1 Erweiterung des Ranking-SVM-Verfahrens durch aktives Lernen

Da die Erstellung einer Trainingsinstanz aufgrund der hierfür durchzuführenden Schachspiele zeitaufwendig ist und es sein kann, dass die mit einer zufälligen Trainingsmenge trainierte Funktion noch keine zuverlässigen Vorhersagen bei gegebenem Umfang der Trainingsmenge trifft, soll speziell für das Ranking-SVM-basierte Verfahren auch der Ansatz versucht werden, für das Lernen der Funktion auf aktives Lernen zurückzugreifen und die Evaluationsfunktionenpaare nicht zufällig, sondern gezielt zu wählen, damit die Ranking-Funktion mit weniger Trainingsinstanzen bereits bessere Voraussagen macht.

Für eine Abschätzung des Qualitätsgewinns, den eine neue Instanz mit sich bringt, soll unter anderem auf aktive Lernansätze für Ranking-SVMs [8] [9] zurückgegriffen werden, die mitunter vor allem auch im Information Retrieval verwendet werden.

Dabei wird für das effektive Lernen einer Trainingsmenge nach solchen Vektorenpaaren in dieser gesucht, deren Scoredifferenz bzgl. der aktuellen Rankingfunktion am geringsten ist bzw. deren Projektionen auf den aktuellen Gewichtsvektor \( \vec{w} \) den geringsten Abstand haben. Für ein solches Vektorenpaar ist die Vorhersage ihrer Relation aufgrund eines ähnlichen Scores mit einer größeren Unsicherheit verbunden. Die Ermittlung der Relation des Vektorenpaares, der Einbezug zur aktuellen Trainingsmenge und die Aktualisierung der Rankingfunktion hätte den Effekt, dass die daraus resultierende Neuanschluss des Gewichtsvektors \( \vec{w} \) zu einer Vergrößerung des Abstandes der Projektionen dieser Vektoren führt und damit verbessert sich die Generalisierung. Dies ist analog zu aktivem Lernen bei normalen SVMs, wo jedoch die Vektoren mit dem geringsten Abstand zur Hyperebene gesucht werden.

Damit dieses Vorgehen erwartungsgemäß funktioniert, ist es notwendig, dass es auch einen Gewichtsvektor \( \vec{w} \) gibt, sodass die Projektionen der Vektoren auf diesem dem gesuchten Ranking entsprechen. Für verschiedene Transformationsfunktionen kann dies sehr unterschiedlich aussehen. Daher ist auch das Testen verschiedener Kernels angebracht.

Der Abstand zwischen den Projektionen zweier Vektoren auf \( \vec{w} \) ist gegeben durch \( |\frac{\vec{w}^T(\phi(\vec{v}_i) - \phi(\vec{v}_j))}{||\vec{w}||_2}| \).

Will man demnach das Vektorenpaar \( (\vec{v}_i, \vec{v}_j) \) mit dem geringsten Abstand ermitteln, muss man \( |\frac{\vec{w}^T(\phi(\vec{v}_i) - \phi(\vec{v}_j))}{||\vec{w}||_2}| \) minimieren, was äquivalent zur Minimierung von \( |\vec{w}^T(\phi(\vec{v}_i) - \phi(\vec{v}_j))| = |\vec{w}^T(\phi(\vec{v}_i) - \vec{w}^T\phi(\vec{v}_j))| = |f(\vec{v}_i) - f(\vec{v}_j)| \) ist.

Es soll also versucht werden, im gegebenem Raum ein Vektorenpaar zu finden, für das diese Scoredifferenz möglichst gering ist. Da eine optimale Lösung durch 2 identische Vektoren gegeben ist, deren Vergleich jedoch weniger von Interesse ist, soll auch versucht werden, den Abstand zwischen diesen größer zu halten. Auf diese Weise wird auch erreicht, besser über den gesamten Raum der Evaluationsfunktionen zu generalisieren.
Man kann dieses Größerhalten des Abstandes dadurch berücksichtigen, indem man das bishe-
rigere Optimierungsproblem erweitert und nun stattdessen versucht den Quotienten aus Abstand
und Scoredifferenz zu maximieren.

Als zu lösendes Optimierungsproblem ergibt sich damit:

\[
\argmax_{(\vec{v}_i, \vec{v}_j)} \frac{\text{dist}(\vec{v}_i, \vec{v}_j)}{f(\vec{v}_i) - f(\vec{v}_j)}
\]

Um ein gutes Vektorenpaar hierfür zu finden, kann man auf genetische Algorithmen zurück-
gegreifen, mit denen sich ein lokales Maximum für dieses Optimierungsproblem approximieren
lässt.

Nachdem ein Vektorenpaar mit möglichst hoher „Fitness“ gefunden wurde, wird dessen Rela-
tion durch Schachspiele ermittelt. Zusammen mit dieser Relation ergibt sich dann eine neue
Trainingsinstanz. Fällt der Vergleich für ein Vektorenpaar jedoch unentschieden aus, bringt
es wenig, das Optimierungsproblem bei gleichen Bedingungen neu zu starten, da mit nicht
geringer Wahrscheinlichkeit das gleiche Paar gefunden wird. Ein Vergleichen bis ein Nichtun-
entschieden zustande kommt, kann bei annährend gleich starken Evaluationsfunktionen sehr
lange dauern und damit teuer sein. Stattdessen wird zufällig eine Komponente beider Vektoren
zufällig mit einem neuen Wert besetzt und der Vergleich dann wiederholt, bis ein Vergleich nicht
unentschieden endete.

Auf diese Weise erhält man in schätzungsweise akzeptabler Zeit zumindest für ein ähnliches
Vektorenpaar eine Relation, die man als die des ursprünglichen Vektorenpaares annehmen kann.

### 3.3 Ranking-Verfahren 2

Auch für dieses Ranking-Verfahren soll die Vorhersage der Relation, welche für 2 gegebene Eva-
luationsvektoren gilt, auf der Basis einer zufällig generierten Trainingsmenge \( Tr \) erfolgen, für
welche \( Tr \subset \succ \) angenommen wird.

Die Grundidee dieses Verfahrens soll sein, für ein gegebenen Paar \( \vec{v}_i, \vec{v}_j \) von Evaluationsvek-
toren \( \vec{v}_i \succ \vec{v}_j \) vorherzusagen, gdw. das Paar \( (\vec{v}_i, \vec{v}_j) \) zu den Paaren in der Trainingsmenge eine
höhere Ählichkeit als hat \( (\vec{v}_j, \vec{v}_i) \) und somit eher \( \vec{v}_i \succ \vec{v}_j \) gilt als \( \vec{v}_j \succ \vec{v}_i \). Dafür soll zunächst
ein Ähnlichkeitsmaß definiert werden, welches für 2 Paare von Evaluationsvektoren angibt, wie
ähnlich diese zueinander sind.

Man kann 2 Paare als ähnlich annehmen, wenn die ersten, sowie die zweiten Elemente zueinan-
der jeweils eine geringe Distanz haben. Es bietet sich hierfür an, ein Distanzmaß für Paare auf
der Basis des bereits für einzelne Vektoren definierten zu definieren, indem man die euklidische
Norm der beiden Distanzen bildet.

Als Distanzmaß für 2 Paare ergibt sich somit:

\[
\text{dist}((\vec{v}_i, \vec{v}_j), (\vec{v}_k, \vec{v}_l)) = \sqrt{\text{dist}(\vec{v}_i, \vec{v}_k)^2 + \text{dist}(\vec{v}_j, \vec{v}_l)^2}
\]

Für das Ähnlichkeitsmaß wird nun einfach der Kehrwert dieser Distanz gebildet, sodass diese-
ses umso größer ist, je kleiner die Distanz zwischen 2 Paaren ist.
\[
\text{sim}((\vec{v}_i, \vec{v}_j), (\vec{v}_k, \vec{v}_l)) = \frac{1}{\text{dist}((\vec{v}_i, \vec{v}_j), (\vec{v}_k, \vec{v}_l))} = \frac{1}{\sqrt{\text{dist}(\vec{v}_i, \vec{v}_k)^2 + \text{dist}(\vec{v}_j, \vec{v}_l)^2}}
\]

Um nun die Zugehörigkeit eines Paares zu den Paaren einer Menge zu ermitteln, wird die Summe der jeweiligen Ähnlichkeiten des Paares zu allen Paaren in der Relation gebildet. Somit ergibt sich ein Zugehörigkeitsmaß eines Paares zu einer Menge:

\[
\text{sim}_{Tr}(\vec{v}_i, \vec{v}_j) = \sum_{(\vec{v}_k, \vec{v}_l) \in Tr} \text{sim}((\vec{v}_i, \vec{v}_j), (\vec{v}_k, \vec{v}_l))
\]

Für die Vorhersage ergibt sich somit:

\[
\vec{v}_i \succ^* \vec{v}_j \iff \text{sim}_{Tr}(\vec{v}_i, \vec{v}_j) > \text{sim}_{Tr}(\vec{v}_j, \vec{v}_i)
\]

### 3.4 Ranking-Verfahren 3

Während in den letzten Verfahren die Trainingsmenge aus zufälligen Paaren mit ermittelter Relation bestand, soll in diesem Ansatz versucht werden, aus paarweise verglichenen Evaluationsfunktionen nach und nach eine Rankingliste von Evaluationsvektoren aufzubauen und mit dieser eine Vorhersage für die Relation eines neuen Evaluationsfunktionspaares durchzuführen.

Um die Liste zu erstellen, werden einzelne Evaluationsvektoren zufällig ausgewählt und anschließend durch Vergleiche mit bereits in der Liste enthaltenen Evaluationsvektoren verglichen und eingeordnet, indem zuerst ein Vergleich mit dem mittleren Element der gesamten Liste erfolgt, und abhängig vom Resultat, das Einfügen in die Teilliste ober- bzw. unterhalb dieses Elements rekursiv fortgesetzt wird. Im Unterschied zu den anderen Verfahren wird eine zufällig ausgewählte Evaluationsfunktion bei der Erstellung der Trainingsmenge mit mehr als nur einer anderen Evaluationsfunktion verglichen und diese Vergleiche erfolgen dann auch nicht zufällig. Die Anzahl der Vergleiche pro einzuordnende Evaluationsfunktion hat die Komplexität \( \Theta(\log_2(n)) \), wobei \( n \) die aktuelle Listengröße ist, was akzeptabel erscheint.

Mit einer gegebenen Liste soll dann für ein neues Paar eine Abschätzung getroffen werden, wo die beiden Vektoren des Paares in der Liste eingeordnet werden müssten und daraus eine Vorhersage für die Spielstärkenrelation abgeleitet werden, da man einen Vektor genau dann als den besseren annehmen kann, wenn er in der Rankingliste höher eingeordnet wird.

Das Vorgehen für die Abschätzung der Position, wo ein neuer Evaluationsvektor in der Liste stehen würde, soll wie folgt aussehen:

Zunächst soll versucht werden, auch hier ein Ähnlichkeitsmaß zu definieren, dass für 2 Elemente angeben soll, wie stark sie dazu neigen, an einer ähnlichen Position in der Liste eingeordnet zu werden. Es wird angenommen, dass Vektoren mit geringer Distanz eine ähnliche Spielstärke haben. Auch hier wird daher als Ähnlichkeitsmaß der Kehrwert der Distanzen der Evaluationsvektoren zueinander verwendet. Es ergibt sich also als Ähnlichkeitsmaß \( \text{sim}(\vec{v}_i, \vec{v}_j) = \frac{1}{\text{dist}(\vec{v}_i, \vec{v}_j)} \).

Damit das Resultat für zu geringe Distanzen hier nicht zu groß wird, erfolgt noch eine Aufaddition von 0.1 im Nenner, sodass sich \( \text{sim}(\vec{v}_i, \vec{v}_j) = \frac{1}{\text{dist}(\vec{v}_i, \vec{v}_j) + 0.1} \) ergibt.

Für die Abschätzung der Position \( \text{pos} \) eines einzuordnenden Elements \( \vec{v} \) kann man nun die Summe aus allen Positionsnummern \( i \) nehmen, die mit \( \text{sim}(\vec{v}, \vec{v}_i) \) (\( \vec{v}_i \) ist i-tes Listenelement)
gewichtet wurden und anschließend normalisieren, also:

\[ \text{pos}(\vec{v}) = \frac{\sum_{i=1}^{n} \text{sim}(\vec{v}, \vec{v}_i) \cdot i}{\sum_{i=1}^{n} \text{sim}(\vec{v}, \vec{v}_i)}, \] mit n als aktuelle Listengröße und \( \vec{v}_i \) als i-tes Element in der Liste.

Nimmt man an, dass die Positionsnummer eines Elements in der Liste umso höher ist, je größer die Spielstärke ist, so lautet also die Vorhersage:

\[ \vec{v}_i \succ^* \vec{v}_j \iff \text{pos}(\vec{v}_i) > \text{pos}(\vec{v}_j), \] wobei man für diesen Vergleich auf die Normalisierung auch verzichten kann.
4 Versuchsdurchführung und Evaluation der Ergebnisse

In diesem Abschnitt sollen die 3 genannten Ranking-Verfahren getestet werden, indem diese für gegebene Trainingsmengen daraufhin geprüft wurden, wie zuverlässig die Relationsvorhersagen jeweils für eine Testmenge sind, die aus zufällig gewählten Paaren von Evaluationsfunktionen besteht, deren angenommen Relation durch Schachspiele ermittelt wurde.

4.1 Programm zur Versuchsdurchführung

Um diesen Test der 3 Verfahren durchzuführen, d.h. um unter anderem automatisch die Trainings- und Testmengen zu erstellen, sowie die Vorhersagen der Verfahren mit den gelernten Modelle zu testen, wurden Programme in Java geschrieben. Über bereits existierende Programme, die dabei verwendet wurden, wird nachfolgend ein Überblick gegeben.

4.1.1 Schachprogramm


4.1.2 Schachprogramminterface


1 http://www.craftychess.com/
4.1.3 SVM-Ranking

Als Implementierung für einen Ranking-SVM-Algorithmus wurde \( \text{SVM\_rank} \) verwendet [6]. Abgesehen von verschiedenen Kernel wurden hier ausschließlich die Standardeinstellungen verwendet, was auch die veränderbaren Parameter der einzelnen Kernelfunktionen betrifft.

4.1.4 Genetische Algorithmen

Für die Anwendung genetischer Algorithmen zur Optimierung wurde das Open-Source Java-Framework \( \text{JGAP} \) für genetische Algorithmen benutzt.

4.2 Durchführung und Auswertung der Tests

Die verwendete Testmenge \( T \) hat einen Umfang von ca. 1000 Evaluationsfunktionspaaren, die allesamt zufällig gewählt wurden und in ca. 4 Wochen Rechenzeit erstellt wurden. Die Testmenge entspricht ungefähr 40% aller paarweisen Vergleiche, die in dieser Zeit durchgeführt wurden. Bei den restlichen Vergleichen wurden die Evaluationsfunktionspaare als gleich stark eingestuft und die Resultate somit nicht berücksichtigt. Es wird hier davon ausgegangen, dass die Relation, die für alle Evaluationsfunktionspaare ermittelt wurde, der tatsächlichen Relation \( \succ \) entspricht, also \( T \subset \succ \) gilt, wobei hier natürlich eine Irrtumswahrscheinlichkeit vorhanden ist.

Beim Vergleich der 3 Ranking-Verfahren wurde die Güte der Vorhersagen in Abhängigkeit von der Anzahl der erstellten Trainingsdaten ermittelt. Als Gütekriterium für die Vorhersagen wurde das Evaluationsmaß \( \text{Kendalls} \ \tau \) verwendet, welches die Übereinstimmungen der Relationsvorhersagen \( \succ^* \) mit der tatsächlichen Relation \( \succ \) für alle Testpaare \( (\vec{v}_i, \vec{v}_j) \in T \), auf \([-1,1]\) abbildet und gegeben ist durch \( \tau(P,Q) = \frac{p-q}{p+q} \), wobei \( P \) die Anzahl der Testpaare ist, bei der die Relationsvorhersage mit der ermittelten tatsächlichen übereinstimmt (d.h. \( \vec{v}_i \succ \vec{v}_j \)) und \( Q \) die Anzahl an Testpaaren mit Nichtübereinstimmung (d.h. \( \vec{v}_i \not\succ \vec{v}_j \)). Je höher also der Wert ist, desto besser die Vorhersagen. Ein Wert von 0 bedeutet hier, dass die richtigen und falschen Vorhersagen identisch sind und das Verfahren folglich nicht brauchbar als ein zufälliges Entscheidungsverfahren ist, das jeweils mit einer Wahrscheinlichkeit von 50% eine von 2 Relationen zuweist. Wünschenswert wäre es demnach, mit diesem Maß Evaluationswerte zu erhalten, die zumindest deutlich größer als 0 sind.

Nachfolgende Grafiken zeigen die Entwicklung des Evaluationsmaßes für alle angewandten Verfahren in Abhängigkeit vom Lernprozess. Für die Verfahren 1 (Ranking-SVM) ohne und mit aktivem Lernen wurden verschiedene Kernel verwendet. Die Trainingsmenge für die SVM-Verfahren und das Ranking-Verfahren 2 umfassen bis zu 200 Paare, die für das Ranking-Verfahren 3 umfasst eine Liste mit bis zu 200 Elementen:

2 \hspace{1cm} \text{http://www.cs.cornell.edu/people/tj/svm_light/svm_rank.html}

3 \hspace{1cm} \text{http://jgap.sourceforge.net/}
Abbildung 8: Ranking-SVM-Verfahren für verschiedene Kernel

Abbildung 9: Ranking-SVM-Verfahren mit aktivem Lernen für verschiedene Kernel
Aus den Grafiken ist ersichtlich, dass bei allen Verfahren (und für verschiedene Einstellungen) der Lernverlauf auffällig in der Nähe des Wertes 0 pendelt, was bedeutet, dass die Anzahl korrekter und falscher Relationsvoraussagen im Durchschnitt in etwa gleich bleibt und die Voraussagen in etwa auf dem Niveau eines zufälligen Entscheiders liegen. Selbst bei dem Ranking-Verfahren 1 (Ranking-SVM) mit und ohne aktivem Lernen ergibt sich dieses Bild. Einzig bei dem Ranking-
Verfahren 2 scheinen hierbei die Werte zumindest einigermaßen stabil etwas über 0 zu liegen, während bei allen anderen der Durchschnittswert in etwa gleich 0 ist.

Ein möglicher Grund für die jeweiligen Resultate ist, dass die Trainingsmengen nicht groß genug waren, um mit den entsprechenden Verfahren zuverlässig zu generalisieren, um für von den Trainingsdaten unabhängige Testdaten korrekte Voraussagen zu treffen. Eine weitere Möglichkeit ist, dass es mit den entsprechenden Verfahren grundsätzlich nur schwer möglich bis unmöglich ist, zuverlässig zu generalisieren, auch wenn die benötigten Trainingsmengen in großem Umfang vorhanden sind.


Man könnte ebenfalls einwenden, dass ein zu hoch angesetztes Signifikanzniveau bei den Spielvergleichen eine Rolle für dieses Resultat spielt, da zu vielen Paaren in den Trainings- und Testmengen falsche Relationen zugewiesen worden. Da es sich dabei aber nur um eine verhältnismäßig geringe Anzahl handeln dürfte, kann dies kein bestimmender Faktor für dieses Resultat sein.

Weiterhin ist denkbar, dass die Gewichte der verwendeten Distanzfunktion \( \vec{s} = (s_1, ..., s_{10})^T \) nicht gut genug gewählt waren, die am Standardwert der Merkmalsgewichte orientiert wurden. Diese Distanzfunktion wurde bei allen Verfahren außer dem SVM-Verfahren ohne aktives Lernen verwendet. Möglicherweise könnte auch eine Optimierung dieser Distanzgewichte bei diesen Verfahren zu besseren Ergebnissen führen.

Es soll daher noch nachfolgend versucht werden, nach besseren Distanzgewichten zu suchen.


Nachfolgende Grafik zeigt den Trainingsverlauf für die Validierungsmenge sowohl mit den bisher verwendeten Gewichten als auch mit den optimierten Gewichten, bei denen sich die größte Fläche unter der Lernkurve innerhalb des Optimierungszeitraumes ergab:
Abbildung 12: Verfahren 2 auf Validierungsmenge mit und ohne optimierte Distanzwichte

Für die Validierungsmenge lässt sich hierbei eine deutliche Steigerung mit den optimierten Gewichten erkennen. Dennoch sind die auch Resultate mit den optimierten Gewichten zu ungenau. Für die Testmengen ergeben sich folgende Veränderungen mit den optimierten Gewichten, wobei hier nur die Ranking-Verfahren 2 und 3 getestet wurden:

Abbildung 13: Verfahren 2 auf Testmenge mit und ohne optimierte Distanzwichte

Getestet wurden die verschiedenen Verfahren, indem der Fehler auf einer Testmenge in Abhängigkeit vom Lernprozess auf den Trainingsdaten untersucht wurde. Die Verfahren 2 und 3 wurden zudem auch unter Verwendung einer Distanzfunktion mit optimierten Gewichten getestet. Das Resultat war, dass sich für alle Verfahren im Durchschnitt nahezu zufällige Vorhersagen ergaben, weshalb der Versuch, ein zuverlässiges Prädikat zu finden, unter diesen Vorgaben somit nicht erreicht werden konnte. Diese Gemeinsamkeit bei allen 3 Verfahren, die recht unterschiedlich sind, belegt, dass die Trainingsdaten offenbar auch generell nur schwer bzw. nicht generalisierbar sind.

Literatur


[9] YU, Hwanjo: *SVM Selective Sampling for Ranking with Application to Data Retrieval*. 2005