Seminar Information Retrieval
Sommersemester 2012
Learning to Rank

Lorenz Heikenfeld

1 Einleitung

Ranking-Funktionen, also Funktionen welche Ordnungen über eine Menge von Objekten definieren, sind von zentraler Bedeutung für die Disziplin des Information Retrievals. (Internet-)Suchmaschinen sind wohl das beste Beispiel von Systemen, die als Dienstleistung die Ordnungen von Objekten zur Verfügung stellen und mittlerweile im alltäglichem Leben quasi-unverzichtbar sind.

Das Erzeugen solcher Ordnungen in Abhängigkeit der Relevanz bezüglich thematischer Suchbegriffe ist Objekt intensiver Forschungsbemühen. Seitdem relevante Literatur in großen Bibliotheken automatisiert auffindbar ist, stellt insbesondere die Datenfülle des Internets die Motivation für die automatisierte Bestimmung der Relevanz von Dokumenten dar. In der Vergangenheit wurde eine Reihe von bekannten Maßen für die Relevanz von Dokumenten entwickelt. Maße wie etwa `TF-IDF` oder `BM25` sind die Ergebnisse intensiver heuristischer Problemlösung. Dabei stützte man sich auf ein theoretisches Konzepte der Relevanz, welche etwa durch Probabilistische Modelle oder Sprachmodelle begründet werden. Man konnte mittels diesen Konzepten durch Observationen auf Beispielen geeignete Verallgemeinerungen treffen, welche sowohl auf umfangreichen Testdaten als auch in der Praxis zu nützliche Ergebnisse führten.

Sowohl die Fülle an Daten, welche das Internet bietet, und das Bestreben immer neue Aspekte der Dokumente in Betracht zu ziehen machen es wünschenswert auch das Erlernen von Ranking-Funktionen zu automatisieren. Die Anzahl der Eigenschaften einzelner Dokumente (oft im Kontext der Suchanfrage) kann dabei die Größenordnung von 500 Werten sprengen. Der Effekt des Overfittings stellt ebenfalls eine große Gefahr dar, wenn man mittels einer Auswahl von Beispielen versucht eine allgemeine, nützliche Ranking-Funktion zu finden: Die Spezialisierung auf auffällige Muster, welche innerhalb der Menge von Beispielen vorkommen, könnten zwar auf diesen Beispielen zum Erfolg führen, aber im Allgemeinen für das zu erlernende Konzept irrelevant sein.

Mittlerweile ist es nicht mehr praktikabel Ranking-Funktionen für moderne Suchmaschinen hauptsächlich manuell zu optimieren. In den letzten 10-15 Jahren greift man für dieses Problem auf erprobte Verfahren des Machine Learnings zurück. Es stellt sich dabei heraus, dass eine Vielzahl von überwachten Lernverfahren neben üblichen Klassifikationsaufgaben auch für Rankingprobleme einsetzen lassen.

Das Dokument ist folgend gegliedert: Abschnitt 2 beschäftigt sich mit der Abstraktion der zu ordnenden Dokumente durch aussagekräftige Eigenschaften.

Abschnitt 3 stellt eine Reihe von Lernverfahren für Ranking-Funktionen und Möglichkeiten der Kategorisierung von Lernverfahren vor. Anschließend werden Evaluationsergebnisse der vorgestellten Verfahren im Abschnitt 4 betrachtet. Eine abschließende Diskussion folgt im Abschnitt 5 an.

2 Document Features

Merkmale oder auch Features dienen der Abstraktion von Objekten, in diesem Fall HTML-Dokumenten. Gewünscht sind insbesondere solche Features, welche aussagekräftige Eigenschaften in Form von reellen Zahlen repräsentieren. Diese Abstraktion erlaubt es die selben Lernverfahren in anderen Problemdomänen wiederzuverwenden, sofern man auch dort aussagekräftige Features findet.

Im Learning to Rank Kontext werden Features von Dokumenten in drei Gruppen kategorisiert: (i) Query-spezifische Features, (ii) statische Features und (iii) Nutzer-spezifische Features. (i) Sind Merkmale welche abhängig von den Suchparametern der Query sind. Beispiele für query-spezifische Features sind die Summe der `term frequencies` der Suchbegriffe im Dokument, oder auch erprobte IR Relevanzmaße wie `TD-IDF` oder `BM25`. Dokumenten-spezifische Eigenschaften (ii) sind z.B. die `Document Length`, Maße der Komplexität der eingesetzten Sprachmittel oder auf der Struktur der Hyperlinks basierte Werte wie `PageRank` oder Bewertungen der Performanz der jeweiligen Hosts in Form von `HostRank` oder anderen Kriterien. Der "Learning to Rank"-Ansatz erlaubt es eine Vielzahl von Features zu verwenden. So wird in manchen Ansätzen das Dokument als eine Ansammlung von mehreren Streams angesehen, welche sich auf verschiedene Teilmengen des Dokumentes beschränken; Somit kann etwa die `Document Length` individuell angewendet werden auf das Gesamte Dokument, das Größte Textfragment, die Header oder andere strukturell interessante Regionen. Nutzerspezifische Features (iii) erlauben es weiterhin Eigenschaften der Anwender zu berücksichtigen. Dies können demographische Eigenschaften sein oder auch die Browser-Umgebung (Bildschirmauflösung, Betriebssystem, Browserversion ect.) beschreiben. Diese Art von Features erfordert zuverlässige Data Mining Strategien im Einsatz von populären Suchmaschinen und Aufgrund der damit verbundenen wirtschaftlichen Interessen existiert nur wenig Literatur zum Einsatz von Nutzer-spezifischen Features.

3 Überwachte Lernverfahren

Überwachte Lernverfahren sind Algorithmen welche mittels einem Satz von beispielhaften Trainingsdaten versuchen Muster in den Beispielen zu erkennen und auf bisher unbeobachteten Daten zu verallgemeinern. Es findet eine Suche nach einer geeigneten Lösung innerhalb einer sehr großen, möglicherweise auch unendlichen Menge von Hypothesen statt. Üblicherweise kann man eine Hypothese durch eine Kostenfunktion bewerten. Durch die Optimierung einer geeigneten Kostenfunktion findet man zugleich die Parameter der optimalen Hypothese. Im Falle

von Klassifikationsproblemen sucht man meist nach mathematischen Funktionen, welche geeignete Entscheidungslinien in einem Feature Raum definieren.

"Learning to Rank" setzt erprobte Lernverfahren in einem neuem Kontext ein. Es werden nicht länger Entscheidungslinien für Klassifikationsprobleme gesucht, sondern Funktionen welche Präferenzen zwischen Objekten bestimmen. Ist man also in der Lage die Kostenfunktionen geschickt zu formulieren und nutzt geeignete Features, so kann man die erprobten Verfahren auch für das Problem des Rankings nach Relevanz einsetzen.

Lernverfahren im Kontext des Erlernens von Präferenz- oder Ranking-Funktionen können im wesentlichen nach ihren Kostenfunktionen unterschieden werden. Dabei unterscheidet man zwischen Verfahren, welche Kosten zwischen der tatsächlichen Bewertung und der gewünschten Bewertung einzelner Dokumente zuweisen (Pointwise Optimization), Verfahren welche anstatt der Bewertung eines Dokumentes die Präferenz innerhalb eines gegebenen Paares bewerten (Pairwise Optimization), sowie Kostenmaße welche die Fehler auf einer vollständigen Rangordnung bewerten (Listwise Optimization). Publikationen auf dem Gebiet enthalten erfolgreiche Ansätze basierend auf Support Vector Machines, Neuronal Nets, Genetic Programming, Boosting und vielen weiteren Verfahren. Es ist dabei bemerkenswert, dass die Listwise Optimization Strategien es ermöglichen, typische Evaluationsmaße des Information Retrievals direkt als Kostenfunktionen einzusetzen. Während Point- und Pairwise Verfahren weiterhin verfolgt werden, sind die Entwicklungen der Listwise Optimierung erst in den letzten 5-7 Jahren entstanden.

3.1 Ordinal Regression: Large Margin Boundaries

Large Margin Rank Boundaries for Ordinal Regression[1] ist einer der ersten Learning to Rank Ansätze. Es handelt sich dabei um ein Verfahren aus der Kategorie der Pointwise Optimization. Es wird dabei angenommen, dass ein Nutzer nach einer initialen Query eine relativ kleine Anzahl (ca 6-15 Dokumente) von Query-Ergebnissen in ordinale Relevanzkategorien einordnet. Somit erhält man für jede Query neue und individuelle Trainingsdaten und wendet das Lernverfahren darauf an um eine Query-spezifische Ranking-Funktion zu erhalten.

X sei die Menge an Dokumenten welche als initiales Query-Ergebnis zurück gegeben wurde. Y sei eine endliche Menge von Rang-Kategorien ordinaler Natur mit einer der Relevanz entsprechenden Ordnung $\succeq$. Die Abstände zwischen den Rang-Kategorien ist dabei nicht definiert, jedoch ist bekannt welche Ränge einer höheren bzw geringeren Relevanz entsprechen. Mehrere Dokumente können dabei der selben Rang-Kategorie, beispielsweise "Sehr Relevant" oder "Irrelevant" zugeordnet sein. Gesucht ist eine Ranking-Funktion $g : X \to Y$ sodass $y_1 \succeq y_2 \Leftrightarrow g(x_1) \succeq g(x_2)$.

Kostenfunktion Als Kostenfunktion für die Ordinal Regression wird für jedes Paar von Trainingssamples welche nach den Labels in unterschiedliche Ränge kategorisiert wurden untersucht, ob die von der Ranking-Funktion dem selben

Dokument eine höhere Relevanz zuordnet. Dazu wird eine Rank-Differenzfunktion $\ominus$ definiert (1), welche von Null verschiedene Werte bei ungleichen Rängen annimmt. Für jedes Paar, welches nicht den Trainingsdaten entspricht können die Kosten (2) c_{pref} berechnet werden, die abhängig von der Rankzuordnung einzelner Dokumente sind. Der Erwartungswert (3) R_{pref} und somit die Regressions-Kosten erhöhen sich für jedes falsch bewertete Dokument A um so stärker, je mehr andere Dokumente B fälschlicherweise in Relation zu A als (ir-)relevanter bestimmt wurden.

$$\ominus : Y \times Y \to \mathbb{R}$$

$$r_i \ominus r_j := i - j \tag{1}$$

$$c_{pref}(x_1, x_2, y_1, y_2, g) = \begin{cases} 1 & \text{if } y_1 \ominus y_2 > 0 \geq g(x_1) \ominus g(x_2) \\ 1 & \text{if } y_2 \ominus y_1 > 0 \geq g(x_2) \ominus g(x_1) \\ 0 & \text{sonst} \end{cases} \tag{2}$$

$$R_{pref}(g) = \mathbf{E}_{x_1, y_1, x_2, y_2}[c_{pref}(x_1, x_2, y_1, y_2, g)] \tag{3}$$

Das "Equivalence of Risk Functionals"-Theorem besagt, dass die Minimierung von $R_{pref}(g)$ der Minimierung des Klassifikationsfehlers eines äquivalenten Klassifikationsproblemes entspricht. Es findet dabei für jedes Paar (A, B) von Trainingssamples eine Klassifikation in drei mögliche Klassen $\{\succ, \prec, \sim\}$ bezüglich der Relevanzrelation, also "relevanter als", "irrelevanter als" und "gleichermaßen relevant wie", statt.

Die Trainingsdaten ϵ für dieses Klassifikationsproblem sind dann Trainingssamples z ungleicher Ränge sowie die Richtung der Präferenz y'.

$$\epsilon := \{(z, y') | z = (x_i, x_j) \in X \times X \wedge y' = \text{sign}(y_k, y_l) \wedge |y_k \ominus y_l| > 0, y_k, y_l \in Y \tag{4}$$

Algorithmus Als Modell für die Ranking-Funktion g wird nun eine latente, lineare Funktion $f : X \to \mathbb{R}$ gewählt.

$$f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} \tag{5}$$

Schwellenwerte θ werden eingesetzt um die Funktionswerte von f in Intervalle zu zerlegen, welche den Ordinalen Rängen der Ranking-Funktion g entsprechen.

$$g(\mathbf{x}) := r_i \Leftrightarrow f(\mathbf{x}) \in [\theta(r_{-i}), \theta(r_i)[\tag{6}$$

Die kleinsten Abstände der Abbilder $f(\mathbf{x})$ der Samples zu den Grenzen der Rang-Intervalle sollen maximiert werden (siehe Abbildung 3.1). Um dies zu erreichen werden Support Vector Machines eingesetzt um die Gewichte $\mathbf{w}$ für f optimal zu bestimmen. Man koppelt zusätzlich die Decision Lines zwischen den einzelnen Kategorien aneinander, indem für Dokumentenpaare ungleicher Ränge jeweils die Differenz der Merkmale als Input der SVM verwendet wird. Es lässt

sich zeigen, dass dies der Differenz der Funktionsbilder von f entspricht und die selben Gewichte $\mathbf{w}$ resultieren.

$$f(\mathbf{x_i}) - f(\mathbf{x_j}) = \mathbf{w} \cdot (\mathbf{x_i} - \mathbf{x_j}) \tag{7}$$

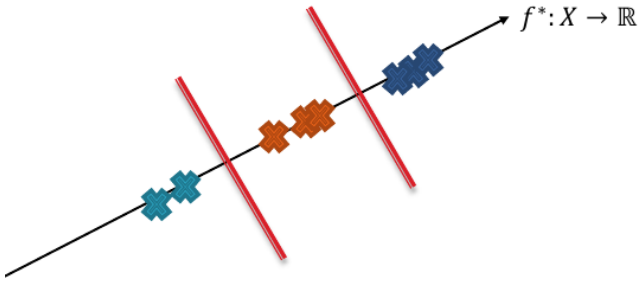

Fig. 1. Idee: SVM maximiert die minimalen Abstände der Abbilder der Samples zu Decision Lines

Die Karush-Kuhn-Tucker Kriterien erlauben es die optimalen Gewichte $\mathbf{w}^*$ aus den Trainingsdaten zu berechnen [2]. Man erhält ein typisches Optimierungsproblem für Support Vector Maschines, wie sie für binäre Klassifikationsaufgaben vorkommen und sucht nach den optimalen Lagrange Multiplikatoren a_i.

$$\mathbf{w}^* = \sum \mathbf{a}^* y'(\mathbf{x_i} - \mathbf{x_j}) \tag{8}$$

$$\mathbf{a}^* = \arg\max_{a_i} \sum_{i}^{|Y'|} a_i - \frac{1}{2} \sum_{i,j=1}^{|Y'|} a_i a_j y'_i y'_j K(\mathbf{x_i} - \mathbf{x_j}))] \tag{9}$$

$$a_i \geq 0 \tag{10}$$

$$\sum_{i=1}^{|Y'|} a_i y'_i = 0 \tag{11}$$

Um nicht linear separierbare Merkmale zu unterstützen wird ein Mercer Kernel K, üblicherweise ein Gauss-Kernel, eingesetzt um die Samples in einen linear separierbaren Feature Space abzubilden.

Nachdem f optimiert wurde, müssen noch die Schwellenwerte für angrenzende Rangintervalle individuell angepasst werden. Weil der Umfang der Trainingsdaten eher klein ist und die optimale Abbildung f bereits bekannt ist, kann man hier die Paare von Samples suchen, deren Funktionswerte den minimalen Abstand zwischen zwei Rängen bilden. Der korrespondierende Schwellenwert zur Trennung beider Ränge wird dann als der Mittelwert beider Funktionswerte definiert.

3.2 RankNet: Gradient Descent

RankNet [3] stellt ein Exemplar der Paarweisen Kostenoptimierung dar. Es handelt sich hier um eine logische Weiterentwicklung des Large Margin Boundary Ordinal Regression Ansatzes. Als Trainingsdaten dient hier eine Reihe von Dokumentenpaaren $(\mathbf{x_i}, \mathbf{x_j})$. Es können Samples aus mehreren Beispiel-Queries verwendet werden, jedoch müssen die Elemente der Paare jeweils aus der selben Query stammen. Ein Neuronales Netz wird eingesetzt um die Relevanz von Dokumenten zu bewerten.

Modell Die Relevanz eines Dokumentes wird durch die Funktion $h : \mathbb{R}^d \to \mathbb{R}$ modelliert.

$$h(\mathbf{x_1}) > h(\mathbf{x_2}) \Leftrightarrow x_1 \succ x_2 \text{ im Modell h} \tag{12}$$

Die Funktion h wird wiederum mittels der Logistischen Funktion auf ein Probabilistisches Modell zurückgeführt. Sei $P(\mathbf{x_1} \succ \mathbf{x_2})$ die Wahrscheinlichkeit dafür, dass $\mathbf{x_1}$ relevanter als $\mathbf{x_2}$ sei. Die Differenz der Funktionswerte von h führt nach Anwendung der logistischen Funktion zu der gesuchten Wahrscheinlichkeit. Große Werte der Differenz $h(\mathbf{x_1}) - h(\mathbf{x_2})$ streben dabei gegen 1, negative Werte gegen 0, während für gleiche Funktionswerte die Präferenzen mit 0.5 als gleich wahrscheinlich modelliert werden.

$$P(\mathbf{x_1} \succ \mathbf{x_2}) = \frac{e^{h(\mathbf{x_1})-h(\mathbf{x_2})}}{1 - e^{h(\mathbf{x_1})-h(\mathbf{x_2})}} \tag{13}$$

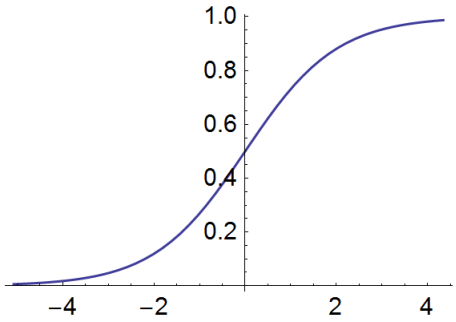

Fig. 2. Die Logistische Funktion $f_l(x) := \frac{e^x}{1-e^x}$

Als zu trainierende Zielwerte $\bar{P}(\mathbf{x_1} \succ \mathbf{x_2})$ für ein Paar (x_1, x_2) kann entsprechend 0.0, 0.5 oder 1.0 gewählt werden. Als Trainingswerte sind ausschließlich diese drei Möglichkeiten zulässig, weil die Kombination von Wahrscheinlichlichkeiten dieser Form im Allgemeinen die Eigenschaften der Asymmetrie und Transitivität der Relevanzrelation verletzen. Allein die zulässigen Werte 0.0, 0.5 und 1.0 erlauben es die Präferenzen bzw. Ambivalenz bei der Kombination von Wahrscheinlichkeiten ohne mögliche Widersprüche zu propagieren.

Neuronales Netz Um die Funktion h zu modellieren wird ein Two-Layer Feed-Forward Neuronal Net eingesetzt. Nach Mitchell [4] ist ein solches Netz in der Lage beliebige kontinuierliche und beschränkte Funktionen zu approximieren, sofern ausreichend viele Knoten auf den beiden Ebenen vorhanden sind. In dieser Anwendung wird nur ein Output Knoten benötigt und für die Anzahl der Knoten im Hidden Layer 10 gewählt. Jedoch ist die Anzahl der Dokumenten-Features in der Größenordnung von ca 600 Merkmalen. Das Erlernen einer geeigneten Bewertungsfunktion h wird auf das Back Propagation Verfahren zurückgeführt. Die Ebenen werden von 1 bis 3 von Inputs über Hidden Layer zu Output Layer numeriert. Sei g^i eine differenzierbare Aktivierungsfunktion der Knoten auf Ebene i, b_a^i der Bias des a-ten Knotens in Layer i und w_{ab}^{ij} das Gewicht zwischen a-ten Knoten in Ebene i und b-tem Knoten in Ebene j. Der Wert des Output Knotens o_i entspricht dann der Gleichung 14.

$$o_i = g^3\left(\sum_j w_{ij}^{32} g^2\left(\sum_k w_{jk}^{21} \cdot x_k + b_j^2\right) + b_i^3\right) \tag{14}$$

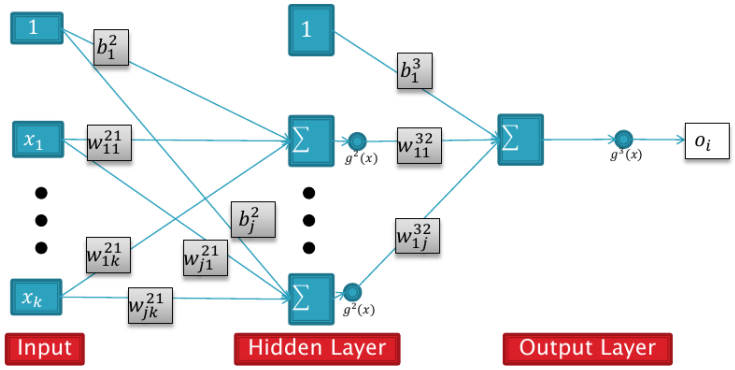

Fig. 3. Struktur des Neuronalen Netzes

Kostenfunktion Als Kostenfunktion $C(P(\mathbf{x_i} \succ \mathbf{x_j}), \bar{P}(\mathbf{x_i} \succ \mathbf{x_j}))$ wird die Cross Entropy verwendet. Die Cross Entropy ist ein für Neuronale Netze übliches Kostenmaß, das Fehler relativ anstatt absolut bewertet [5].

$$C_{ij} = -\bar{P}(x_i \succ x_j)(h(\mathbf{x_i}) - h(\mathbf{x_j})) + \log(1 + e^{h(\mathbf{x_i}) - h(\mathbf{x_j})}) \tag{15}$$

Die Trainingsdaten werden eingeschränkt auf den Fall $\bar{P}(x_{t+1} \succ x_t) = 0$ also auf Dokumentenpaare in absteigender Relevanz. Die Cross Entropy kann nun, für diesen eingeschränkten Fall, durch eine monoton steigende Funktion f

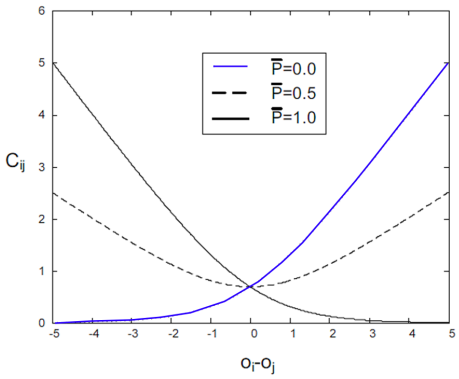

Fig. 4. Cross Entropy für verschiedene Zielwerte. Die blau markierte Funktion wird als Kostenfunktion approximiert

approximiert werden. Die Differenz $o_{t+1} - o_t$ der Relevanzbewertungen konsekutiver Dokumente werden mit dem Zielwert $\bar{P}(x_{t+1} \succ x_t) = 0$ verglichen. Paare von gleichermaßen relevanten Dokumenten werden hier ebenfalls nicht für das Training verwendet. Weil beide Fälle von Präferenz eines Dokumentes zueinander symmetrisch sind, handelt es sich hier nur noch um eine Konvention bezüglich der Reihenfolge der Trainingsbeispiele. Zusätzlich können Gewichtungsfaktoren angewendet werden, um die Fehler auf bezüglich der Rang-Positionen in den Beispielen zu gewichten.

Der Gradient der Kostenfunktion (16) wird jeweils für Dokumentenpaare bestimmt. Dazu werden je die Features eines Dokumentes als Input für das Neuronale Netz mit den aktuellen Gewichten verwendet und die g_j^i Werte aller Knoten berechnet. Diese Aktivierungswerte der Knoten müssen für beide Forward-Propagation Anwendungen eines Dokumentenpaares gespeichert werden. Die erste Ableitung der Approximation der Cross Entropy muss bekannt sein. Nach der Kettenregel kann der Gradient des Neuronalen Netzes zurückgeführt werden auf die Differenz der Gradienten beider Anwendungen des Netzes. Die Ableitungen nach den Gewichtungs- und Bias-Werten sind hier gleich den Ableitungen wie sie bei der Back-Propagation bereits bekannt sind.

$$\frac{\partial f}{\partial \alpha} = \left(\frac{\partial o_{t+1}}{\partial \alpha} - \frac{\partial o_t}{\partial \alpha} \right) f' \tag{16}$$

Wurde der Gradient $\frac{\partial f}{\partial \alpha_k}$ für alle Gewichtsparameter α_k bestimmt, wird zum aktuellen Wert von α_k der Gradient gewichtet mit der Lernrate η addiert. Somit findet also eine Gradientensuche nach einer kosten-minimalen Belegung der Gewichte statt. Es wären eine Reihe von Heuristiken denkbar, um das Feststecken in lokalen Minima zu vermeiden, etwa durch zufälliges "Losrütteln" aus einem lokalem Optimum. RankNet setz als Heuristik gegen das Oszillieren um ein Optimum herum die Halbierung der Lernrate bei steigendem Epochen-Fehler ein.

3.3 RankGP: Genetic Programming

RankGP [6] ist ein Beispiel für die direkte Optimierung einer im Information Retrieval üblichen Listenbewertungsfunktion, der Mean Average Precision (MAP). Sei die Precision einer Menge von Dokumenten bezüglich einer Query das Verhältnis der Anzahl der relevanten Dokumente zur Gesamtzahl der Elemente. Sei Q die Menge von (Beispiels-)Queries, k_{ij} die Position des j-relevantesten (laut Trainingsdaten) Dokumentes zu Query q_i nach einem Ranking und m_i die Anzahl der zu q_i relevanten Dokumente und $P(r)$ die Precision innerhalb der Dokumente auf den obersten r Rangpositionen.

$$\text{MAP} := \frac{1}{|Q|} \sum_{i=1}^{|Q|} \left(\frac{1}{m_i} \sum_{j=1}^{m_i} P(k_{ij}) \right) \tag{17}$$

Mittels Genetischer Programmierung wird ein Hypothesenraum möglicher Rankingfunktionen nach einer solchen Funktion durchsucht, welche die besten MAP Ergebnisse liefert. Das Vorgehen entspricht einer Simulation der Evolutionstheorie, bei welcher der MAP-Wert die individuelle 'Fitness' darstellt.

Feature-Normierung Die Features jeweils werden pro Query auf das Intervall $[0, 1]$ normiert. Dazu werden für jedes Feature und Query das jeweilige Minimum und Maximum benötigt. Die Features werden folgendermaßen durch Verschiebung und Stauchung auf ihren normierten Wert abgebildet:

$$f_i' = \frac{f_i - \max(f_i, q)}{\max(f_i, q) - \min(f_i, q)} \tag{18}$$

Hypothesenraum Elemente des Hypothesenraumes, also der Ranking-Funktionen welche in Betracht gezogen werden, bestehen aus folgenden Bausteinen:

1. Variablen $S_v = \{f_i | f_i \in \text{Features}\}$
2. Konstanten $S_c = \{0.0, 0.1, 0.2, \ldots, 1.0\}$
3. Lineare Operatoren $S_{op} = \{+, -, \cdot, /\}$

Der Hypothesenraum wird auf Binärbäume mit einer maximalen Tiefe von acht Ebenen begrenzt. Es werden ausschließlich lineare Operatoren verwendet.

Turnierselektion Die Turnierselektion ist eine von mehreren Möglichkeiten der Genetischen Programmierung um fitte Individuen für die Reproduktion zu selektieren. Es werden hierzu $n = 5$ Kandidaten zufällig gezogen (mit gleicher Wahrscheinlichkeit) und das fitteste bzw. die zwei fittesten Individuen nach Veränderung mittels Genetischer Operatoren in die neue Generation überführt.

Im Gegensatz zu Selektionsstrategien wie etwa Roulette Selektion, bei welcher die Auswahlwahrscheinlichkeit proportional zur individuellen Fitness ist, ist der

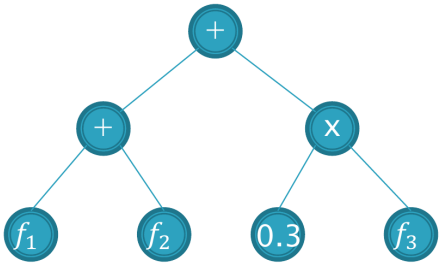

Fig. 5. Beispiel für eine Ranking-Funktion im RankGP-Hypothesenraum

Wert der Fitness bei der Turnierselektion weniger dominant. Es existiert weiterhin ein Bias zu fitten Individuen, jedoch ist die Wahrscheinlichkeit bei der Turnierselektion kleiner, dass ein sogenanntes Overcrowding auftritt. Als Overcrowding bezeichnet man die Tendenz zu großen Gruppen genetisch sehr ähnlichen Individuen mit relativ hoher Fitness, welche wie ein lokales Optimum agieren und ihre Konkurrenten verdrängen. Die Turnierselektion führt also zu einer größeren genetischen Diversität innerhalb der Populationen auch nach vielen Generation [4].

Genetische Operatoren Um neue Individuen aus einer älteren Generation zu erzeugen, werden nach Turnierselektion folgende Operationen angewendet. Dabei wird gewährleistet, dass die maximale Baumtiefe niemals verletzt ist.

- `Reproduktion`
  Die Reproduktion ist die unveränderte Übernahme eines Individuums in eine nachfolgende Generation
- `Crossover`
  Es findet ein Austausch von Genen zwischen zwei Individuen statt. Hier werden zufällige Interne Knoten ausgewählt und gegeneinander ausgetauscht.
- `Mutation`
  Unterbäume an zufälligen Internen Knoten werden durch einen neuen zufällig generierten Baum ersetzt. Dabei ist aufgrund der Anzahl von Knoten pro Baumebene ein Austausch auf tieferen Ebenen wahrscheinlicher. Ob die Baumstruktur (Tiefe und Verzweigungen/Blattknoten-Struktur) selbst geändert wird ist unklar dokumentiert.

Algorithmus RankGP setzt die vorgestellten Operationen in folgender Weise ein:

- `Input`
  Aufteilung der Trainingsdaten in Trainings-Set ($\frac{2}{3}$ der Daten) und Validation-Set ($\frac{1}{6}$ der Daten), sowie Evaluations-Set ($\frac{1}{6}$ der Daten)
- `Initialisierung`
  Zufällige generierung der initialen Population P_0 von $P_{size} = 600$ Individuen. Als Constraint wird verlangt, dass mindestens die Hälfte der Individuen Bäume maximaler Tiefe sind. Es wird eine leere Outputmenge O erzeugt.
- `Iteration`
  Es wird in jedem Iterationsschritt eine neue Generation P_{i+1} erzeugt. Dies geschieht 200 mal.
  1. `Reproduktion`
     Das Individuum mit der höchsten Fitness aus P_i wird nach O und P_{i+1} übernommen.
  2. `Auffüllung der Population`
     Mittels Turnierselektion werden Individuen in P_i ausgewählt und mit gegebener Wahrscheinlichkeit durch Crossover ($p_c = 0.95$) und Mutation ($p_m = 0.05$) modifiziert. Dies erfolgt solange, bis Generation P_{i+1} die gewünschte Größe hat.
- `Output`
  Aus den in ihrer Generation jeweils besten Ranking-Funktionen der Menge O wird nun die Hypothese gewählt, welche die Summe der Fitness (MAP auf Trainings-Set) und der MAP auf dem Validation-Set maximiert. Dies wirkt dem Overfitting entgegen.

4 Evaluationen und Vergleich

Als Evaluationsmaße dienen zum einen das Normalized Discounted Cumulative Gain Measure (NDCG@R) und die Precision (P@R) der ersten R Rangpositionen.

$$\text{NDCG@R} := N_i \sum_{j=1}^{R} \frac{2^{r(j)-1}}{ld(1+j)} \tag{19}$$

Dabei ist $r(j)$ eine Relevanzbewertungsfunktion (nach Evaluationsdaten) für Dokument j. Üblicherweise wählt man $r(j) = 0$ für irrelevante Dokumente und positive Werte für relevante Dokumente. Bei binärer Relevanz würde man die Werte 0 und 1 verwenden, während man bei fein granularen Kategorien die Werte 0 bis 5 zuweisen würde. Der Normalisierungsfaktor N_i wird so gewählt, dass ein perfektes Ergebnis den Endwert 1 erzielt. Auf Grund des exponentiell steigenden Einflusses der Relevanzbewertung und der logarithmischen Abgewichtung niedriger Ränge ist es leider recht schwer die Endwerte des NDCG@R-Maßes miteinander zu vergleichen, falls die Parameter $r(j)$ und R unterschiedlich gewählt wurden.

RankNet wurde auf einem nicht näher spezifiziertem Datensatz von 17000 englisch-sprachiger Queries evaluiert. NDCG wird an Rang 15 gemessen, wobei die Relevanzbewertung die Werte 0 bis 5 annimmt. Auf der Evaluationsmenge wurden NDCG@15-Werte von 0.488 ± 0.01 erreicht.

Für RankGP wurden die NDCG- und Precision-Werte der Ränge 1 bis 15 als Durchschnitt nach 10 maliger 5-fold Cross Validation auf den Datensätzen TD2003 und TD2004 berechnet. Abbildung 4 zeigt die Werte von RankGP, BM25, sowie zwei weiteren Listwise-Verfahren RankBoost und RankSVM. Hier wurde als Relevanzbewerungen die binäre Unterscheidung mit Werten 0 bzw 1 gewählt. Besonders auffällig ist das deutlich bessere Abschneiden aller getesteten "Learning to Rank"-Verfahren im Vergleich zum `BM25`-Ranking.

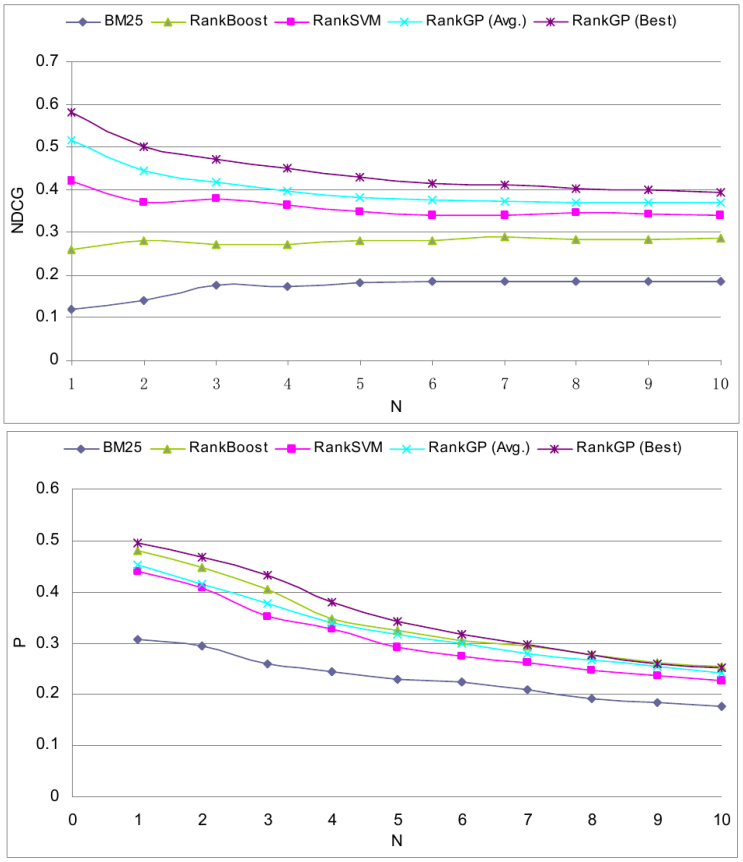

Fig. 6. NDCG@N auf TD2003 (oben) Precision@N auf TD2004 (unten)

5 Diskussion

Es ist offensichtlich, dass der Einsatz von Lernverfahren es ermöglicht Ranking-Funktionen zu erlernen, welche bessere Ergebnisse als traditionelle Verfahren wie z.B. `BM25` liefern. Besonders interessant ist jedoch die Fähigkeit eine sehr große Anzahl von Dokumenteneigenschaften zu berücksichtigen. Mit dem Einsatz von Lernverfahren benötigt man prinzipiell nur noch ausreichend aussagekräftige Features für das jeweilige Ranking-Problem. Somit könnten die vorgestellten Verfahren auch in anderem Kontext erfolgreich eingesetzt werden um Rangordnungen zu bestimmen. "Learning to Rank" liefert zudem interessante Beispiele für den Einsatz von Lernverfahren für die Lösung anderer Problemstellungen als Klassifikations-Probleme. Die "Learning to Rank"-Ansätze bieten somit eine neue Sichtweise auf bereits bekannte Verfahren.

Bedenkt man die hohe Anzahl von eingesetzten Features (ca. 600 bei RankNet) könnte man sich Fragen, ob Verfahren der Feature Selektion oder Dimension Reduction/Basentransformation (z.b. Principle Component Analysis, Singulärwertszerlegung) im "Learning to Rank" Kontext Einsatz finden. Während die Adaption von Lernverfahren zum Ranking in der Literatur gut dokumentiert ist, erwecken die Auswahl der Features und viele der Evaluationen jedoch eher den Eindruck fehlender Spezifikation bzw. einen Mangel an Reproduzierbarkeit. Weil aussagekräftige Features so kritisch für den Erfolg von Lernverfahren sind, wäre es interessant diesen Aspekt des "Learning to Rank" genauer zu untersuchen.

References

1. Herbrich, R., Graepel, T., Obermayer, K.: Large margin rank boundaries for ordinal regression. In Smola, Bartlett, Schoelkopf, Schuurmans, eds.: Advances in Large Margin Classifiers. Volume 88. MIT Press (2000) 115–132
2. Smola, A., Bartless, P., Schoelkopf, B., Schuurmans, D.: Introduction to large margin classifiers. In Smola, Bartlett, Schoelkopf, Schuurmans, eds.: Advances in Large Margin Classifiers. (2000) 89–114
3. Burges, C.J.C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., Hullender, G.N.: Learning to rank using gradient descent. In: Machine Learning, Proceedings of the Twenty-Second International Conference (ICML 2005), Bonn, Germany, August 7-11, 2005. Volume 119 of ACM International Conference Proceeding Series., ACM (2005) 89–96
4. Mitchell, T.: Machine Learning. McGraw-Hill (1997)
5. Theodoridis, S., Koutroumbas, K.: Pattern Recognition. 4. edn. Academic Press (2009)
6. Yeh, J.Y., Lin, J.y., Ke, H.r., Yang, W.P.: Learning to rank for information retrieval using genetic programming (2007)
7. Liu, T.Y.: Learning to rank for information retrieval, tutorial slides, www-conference 2009