



INTELLIGENTE DATENANALYSE IN MATLAB

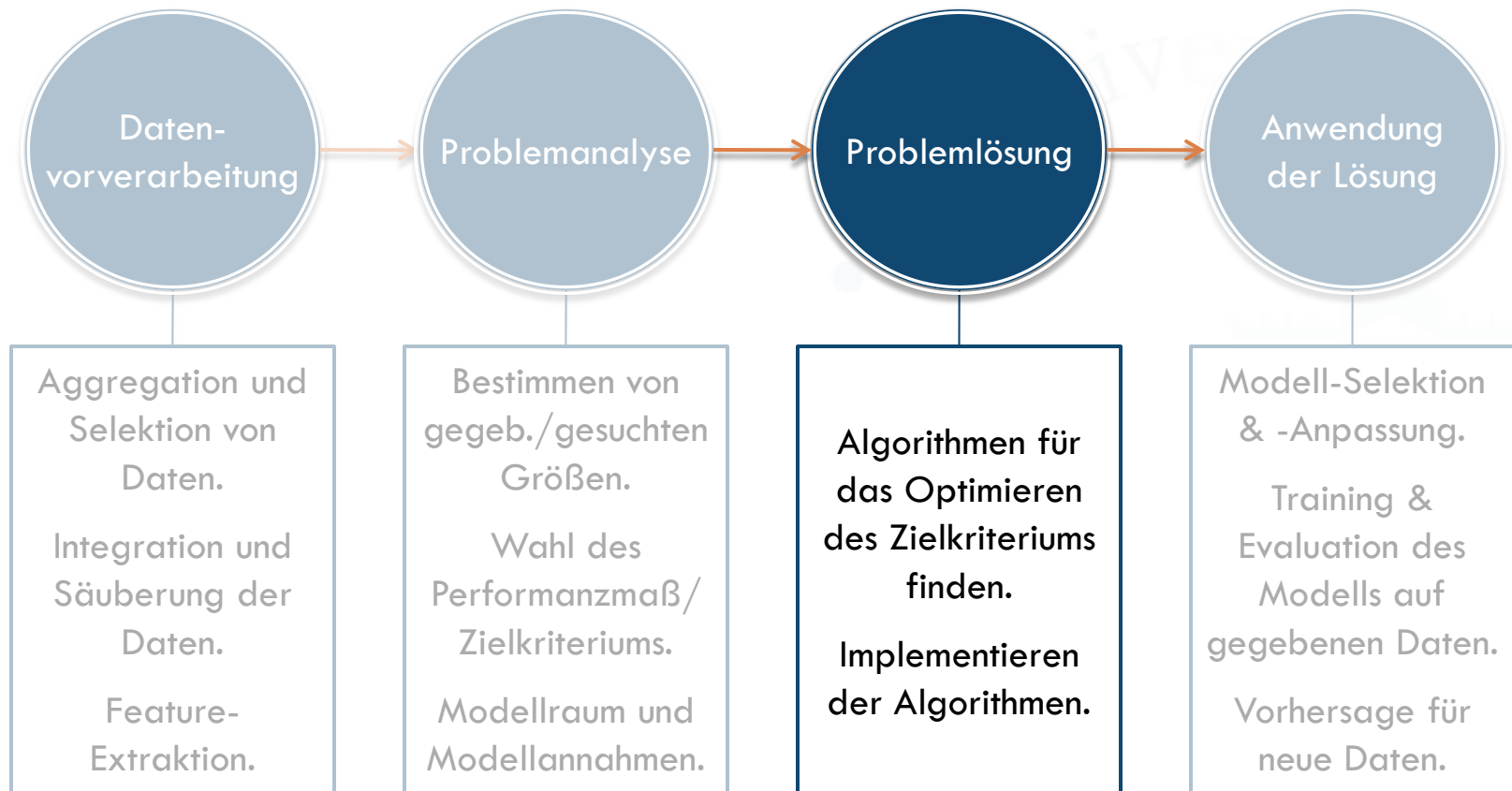
Lernen von Rankings

Literatur

- **Chris Burges: Learning to Rank for Web Search.**
<http://research.microsoft.com/en-us/um/beijing/events/lr4ir-2007>
- **Markus Weimer et al.: CoFiRank.**
<http://www.cofirank.org>
- **Yisong Yue: Learning to Rank.**
<http://www.yisongyue.com/research.php>
- **Lesley A. Ward et al.: Authority Rankings from HITS, PageRank, and SALSA.**
<http://www.math.hmc.edu/~ward/paperpdfs/hitsheaderbw6Jan05.pdf>

Überblick

□ Schritte der Datenanalyse:



Lernen von Rankings

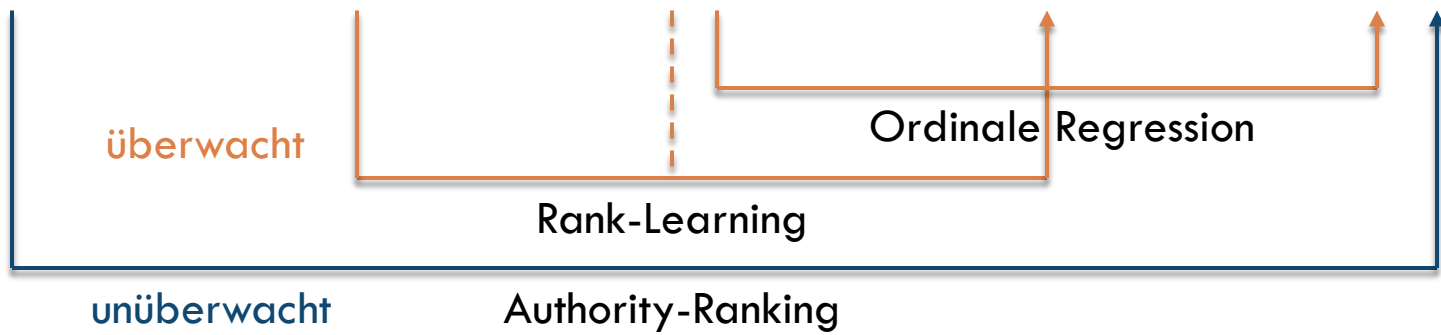
Problemstellungen

- Gegeben: Trainingsdaten mit (un-)bekannter Sortierung, paarweisen Ordnung oder Relevanz.
- Eingabe: Menge von Instanzen.
- Ausgabe: Sortierung der Instanzen.
 - Ordinale Regression: Sortierung bekannt.
 - Rank-Learning: (lokale/paarweise) Sortierung bekannt.
 - Authority-Ranking: Lokale „Kompetenz“-Bewertung bekannt, z.B. durch Verlinkungsstruktur der Instanzen.

Lernen von Rankings

Problemstellungen

Instanz	Echte Belegung		Vorhergesagte Belegung	
	Ranking Score s (inverse Relevanz)	Position (Rank y)	Ranking Score s'	Position (Rank y')
\mathbf{x}_1	$s_1 \in \mathbb{R}_0^+$	$y_1 \in \{1..n\}$	$s'_1 \in \mathbb{R}$	$y'_1 \in \{1..n\}$
\mathbf{x}_2	$s_2 \in \mathbb{R}_0^+$	$y_2 \in \{1..n\}$	$s'_2 \in \mathbb{R}$	$y'_2 \in \{1..n\}$
...
\mathbf{x}_n	$s_n \in \mathbb{R}_0^+$	$y_n \in \{1..n\}$	$s'_n \in \mathbb{R}$	$y'_n \in \{1..n\}$



Lernen von Rankings

Problemstellungen

- Ordinale Regression.
 - Modellierung als Multiklassen-Problem.
 - Modellierung als numerisches Regressions-Problem.
 - Optimierung eines Multivariaten Performanzmaßes.
- Rank-Learning.
 - Optimierung eines Ranking-Kriteriums bzgl. Instanzpaaren oder bzgl. eines globalen Rankings.
- Authority-Ranking.
 - Aus lokalen Kompetenz-Bewertungen zwischen den Instanzen globale Kompetenz-Bewertung lernen.

Rank-Learning

Motivation

- Beispiele:
 - ▣ Sortieren von Dokumenten.
 - ▣ Ranking von neuen Produkten.

Reifenmodell	Preis in EUR	Bremsweg (trocken)	Bremsweg (nass)	Geräusch	Platzierung
Fulda Carat Progresso	44 bis 70	2,0	1,8	3,0	1
Continental PC 2	59 bis 77	1,3	2,0	3,5	2
Bridgestone Turanza	51 bis 75	1,3	2,2	2,9	3
Uniroyal Rain Expert	52 bis 75	1,9	1,9	3,7	4
Semperit Comfort Life	46 bis 66	2,2	2,3	3,3	5
Firestone TZ300 a	48 bis 65	1,8	2,3	3,3	6
Dunlop SP Sport	51 bis 77	1,4	2,8	3,1	7
Vredestein Hi-Trac	37 bis 68	4,1	2,1	4,2	8
Hankook Optimo	43 bis 66	2,1	3,2	3,2	9
Yokohama C.Drive	48 bis 69	1,6	3,2	3,1	10
Goodyear DuraGrip	46 bis 72	2,0	2,8	3,0	?
Michelin Energy Saver	61 bis 86	2,1	2,5	3,0	?
Pneumant PN550	39 bis 65	2,4	3,2	3,6	?
Kumho Solus KH17	42 bis 61	1,8	2,2	3,0	?

Trainingsdaten Testdaten

Zielgröße

Rank-Learning

Problemstellung

- Gegeben: Trainingsdaten $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$ mit gegebenem Ranking $y_1 < y_2 < \dots < y_n$ bzw. Scores $0 \leq s_1 \leq s_2 \leq \dots \leq s_n$.
- Gesucht: Modell $f : \mathbf{x} \in \mathbb{R}^m \mapsto s \in \mathbb{R}$ welches für Instanz \mathbf{x} Ranking-Score s liefert.
- Ziel: Optimierte Ranking-Kriterium.
 - Pairwise Ranking: Ranking-Kriterium = Summe über paarweisen Vergleich der Instanzen/Scores.
 - Global Ranking: Kriterium definiert für gesamtes Ranking.

Gutes Kriterium?

Rank-Learning

Ranking-Kriterien



- Qualität einer Vorhersage $s' = \{s'_1, s'_2, \dots, s'_n\}$ für gegebene Ranking Scores $s_1 \leq s_2 \leq \dots \leq s_n$ bzgl. top- k Instanzen ($1 \leq k \leq n$):

- (Pairwise) Agreement at k -th Position:

$$A(s', k) = \frac{1}{k} \sum_{i=1}^k \sum_{l=k+1}^n [s'_i > s'_l]$$

- Mean Pairwise Agreement at k -th Position:

$$MPA(s', k) = \frac{1}{k} \sum_{j=1}^k A(s', j)$$

- (Negative) Mean Squared Error at k -th Position:

$$MSE(s, s', k) = -\frac{1}{k} \sum_{i=1}^k (s_i - s'_i)^2$$

Rank-Learning

Ranking-Kriteria

- Qualität einer Vorhersage $y' = \{y'_1, y'_2, \dots, y'_n\}$ für gegebenes Ranking $y_1 = 1, y_2 = 2, \dots, y_n = n$ bzgl. top- k Instanzen ($1 \leq k \leq n$):

- Average Precision at k -th Position:

$$P(y', k) = \frac{1}{k} \sum_{i=1}^k [k > y'_i]$$

- Mean Average Precision at k -th Position:

$$MAP(y', k) = \frac{1}{k} \sum_{j=1}^k P(y', j)$$

- (Negative) Distortion at k -th Position:

$$D(y', k) = -\frac{1}{k} \sum_{i=1}^k |y_i - y'_i|$$

Rank-Learning

Ranking-Kriteria

- Qualität einer Vorhersage $y' = \{y'_1, y'_2, \dots, y'_n\}$ für gegebene Ranking Scores $\{s_1, s_2, \dots, s_n\}$ bzgl. top- k Instanzen ($1 \leq k \leq n$):

- Discounted Cumulative Gain at k -th Position:

$$DCG(s, y', k) = \sum_{i=1}^k \frac{2^{(\mu - s_{y'_i})} - 1}{\log(i+1)} \quad 0 \leq s_1 \leq s_2 \leq \dots \leq s_n = \mu$$

- Normalized Discounted Cumulative Gain at k -th Position:

$$NDCG(s, y', k) = \frac{DCG(s, y', k)}{DCG(s, y, k)}$$

- (Unnormierte) Cosinus-Ähnlichkeit bzgl. \mathbf{c} mit $0 < c_1 < \dots < c_n$:

$$C(s, y', \mathbf{c}) = \sum_{i=1}^n s_i (c_{y'_i} - c_i)$$

Rank-Learning

Lösungsansatz



- Approximation des Ranking-Kriteriums durch konvexe Verlustfunktion:
 - (Pairwise) Agreement (z.B. RankSVM).
 - Normalized DCG (z.B. CoFiRank).
 - Mean Average Precision (z.B. Multivariate SVM).
- Approximation des Gradienten des Ranking-Kriteriums:
 - Mean Average Precision, Normalized DCG (z.B. SPSA, LambdaRank).
- Heuristische Suche bzw. Graph-basierte Verfahren:
 - Normalized DCG (z.B. Xrank).

Rank-Learning

RankSVM

- Ziel: Maximiere paarweise Übereinstimmung

$$A(\mathbf{s}', k) = \frac{1}{k} \sum_{i=1}^k \sum_{j=k+1}^n [s'_j > s'_i]$$

für lineares Modell $f_{\mathbf{w}}(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$ mit $s'_i = f_{\mathbf{w}}(\mathbf{x}_i)$.

- Umformung: Minimierung einer (regularisierten) Verlustfunktion

$$\min_{\mathbf{w}} \sum_{i=1}^k \sum_{j=k+1}^n l_{0/1}(f_{\mathbf{w}}(\mathbf{x}_i), f_{\mathbf{w}}(\mathbf{x}_j)) + \Omega(\mathbf{w})$$

mit $l_{0/1}(f_{\mathbf{w}}(\mathbf{x}_i), f_{\mathbf{w}}(\mathbf{x}_j)) = \begin{cases} 0 & f_{\mathbf{w}}(\mathbf{x}_j) > f_{\mathbf{w}}(\mathbf{x}_i) \\ 1 & f_{\mathbf{w}}(\mathbf{x}_j) \leq f_{\mathbf{w}}(\mathbf{x}_i) \end{cases}$

Konvexer Regularisierer

Nicht konvexe Verlustfunktion

- Konvexe Approximation der Verlustfunktion

$$l_{0/1}(f_w(\mathbf{x}_i), f_w(\mathbf{x}_j)) = \begin{cases} 0 & f_w(\mathbf{x}_j) > f_w(\mathbf{x}_i) \\ 1 & f_w(\mathbf{x}_j) \leq f_w(\mathbf{x}_i) \end{cases} = \begin{cases} 0 & f_w(\mathbf{x}_j - \mathbf{x}_i) > 0 \\ 1 & f_w(\mathbf{x}_j - \mathbf{x}_i) \leq 0 \end{cases}$$

bspw. durch Hinge-Loss:

$$l_h(f_w(\mathbf{x}_j - \mathbf{x}_i), +1) = \begin{cases} 0 & 1 - f_w(\mathbf{x}_j - \mathbf{x}_i) < 0 \\ 1 - f_w(\mathbf{x}_j - \mathbf{x}_i) & 1 - f_w(\mathbf{x}_j - \mathbf{x}_i) \geq 0 \end{cases} = \max(0, 1 - f_w(\mathbf{x}_j - \mathbf{x}_i))$$

Neue Beispiele $\mathbf{x}_{ji} = \mathbf{x}_j - \mathbf{x}_i$ mit Label $y_{ji} = +1$

- RankSVM = Minimierung des regularisierten Hinge-Loss mit Instanzen $\{(\mathbf{x}_j - \mathbf{x}_i, +1) : i = 1 \dots n, j = i + 1 \dots n\}$.

Rank-Learning

RankSVM



- Lösung des Optimierungsproblems:
 - Bspw. durch Gradientenabstieg (siehe: lineare Modelle).
- Ranking für neue Instanzen:
 - Berechnen und Sortieren der Scores $s = f_{\mathbf{w}}(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$.
- Erweiterungen:
 - Kernelisierte Variante durch Structural SVM: Quadratische Optimierung mit „Most Violating Constraint“-Ansatz.
 - Approximative Optimierung des Mean Pairwise Agreement at k -th Position durch verwenden der Instanzen $\{(\mathbf{x}_j - \mathbf{x}_i, +1) : i = 1 \dots k, j = k + 1 \dots n\}$.

Rank-Learning

CoFiRank



- Ziel: Maximiere Normalized Discounted Cumulative Gain bzgl. der ersten k Positionen

$$NDCG(s, y', k) = \frac{1}{DCG(s, y, k)} \sum_{i=1}^k \frac{2^{\mu - s_{y'_i}} - 1}{\log(i+1)}$$

für lineares Modell $f_w(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$ mit $y' = \arg \text{sort}(f_w(\mathbf{x}_i))$,
 $y = \arg \text{sort}(s_i)$ und $\mu = \max_i(s_i)$.

- Problem: $NDCG(s, \arg \text{sort}(f_w(\mathbf{x}_i)), k)$ ist nicht konvex in \mathbf{w} !
- Idee: Konvexe obere Schranke für NDCG-Loss finden.

Rank-Learning

CoFiRank



- NDCG-Verlustfunktion für $\mathbf{y}' = \arg \text{sort}(s'_i)$ mit $s'_i = f_{\mathbf{w}}(\mathbf{x}_i)$:

$$l_{NDCG}(\mathbf{s}, \mathbf{y}') = 1 - NDCG(\mathbf{s}, \mathbf{y}', k)$$

- Es gilt $\mathbf{y}' = \arg \text{sort}(s'_i) = \arg \max_{\mathbf{p} \in \text{perm}(n)} C(\mathbf{s}', \mathbf{p}, \mathbf{c}) = \arg \max_{\mathbf{p} \in \text{perm}(n)} \sum_{i=1}^n s'_i (c_{p_i} - c_i)$.

Permutation
von $\{1, \dots, n\}$

Maximal falls \mathbf{p}
sortiert ist wie \mathbf{s}'

- Relaxierte NDCG-Verlustfunktion:

$$l'_{NDCG}(\mathbf{s}, \mathbf{s}') = \max_{\mathbf{p} \in \text{perm}(n)} l_{NDCG}(\mathbf{s}, \mathbf{p}) + C(\mathbf{s}', \mathbf{p}, \mathbf{c})$$

□ Für die relaxierte Verlustfunktion gilt:

$$\square l'_{NDCG}(\mathbf{s}, \mathbf{s}') = l_{NDCG}(\mathbf{s}, \hat{\mathbf{p}}) + C(\mathbf{s}', \hat{\mathbf{p}}, \mathbf{c}) \geq l_{NDCG}(\mathbf{s}, \mathbf{y}') + C(\mathbf{s}', \mathbf{y}', \mathbf{c}) \geq l_{NDCG}(\mathbf{s}, \mathbf{y}')$$

$$\text{mit } \hat{\mathbf{p}} = \arg \max_{\mathbf{p} \in \text{perm}(n)} l_{NDCG}(\mathbf{s}, \mathbf{p}) + C(\mathbf{s}', \mathbf{p}, \mathbf{c}).$$

Nicht-negativ

□ $l'_{NDCG}(\mathbf{s}, \mathbf{s}')$ ist konvex in \mathbf{s}' und damit auch konvex in \mathbf{w} mit

$$\text{Ableitung } \frac{\partial l'_{NDCG}(\mathbf{s}, \mathbf{s}')}{\partial \mathbf{s}'} = \frac{\partial C(\mathbf{s}', \hat{\mathbf{p}}, \mathbf{c})}{\partial \mathbf{s}'} = \mathbf{c}_{\hat{\mathbf{p}}} - \mathbf{c}.$$

□ Optimierungsproblem $\hat{\mathbf{p}} = \arg \max_{\mathbf{p} \in \text{perm}(n)} l_{NDCG}(\mathbf{s}, \mathbf{p}) + C(\mathbf{s}', \mathbf{p}, \mathbf{c})$ kann für gegebenes \mathbf{w} (und damit festes \mathbf{s}') effizient berechnet werden.

□ Algorithmus (RegERM mit relaxiertem NDCG-Loss):

RegNDCG(*Instanzen* \mathbf{x}_i , *echte Ranking Scores* s_i und *Schranke* k)

Setze $k = 0$, $\mu^0 = 1$, $\mathbf{w}^0 = \mathbf{0}$

Wähle \mathbf{c} mit $0 < c_1 < \dots < c_n$

DO

$$s_i'^k = \mathbf{x}_i^T \mathbf{w}^k \quad \forall i$$

$$\hat{\mathbf{p}}^k = \arg \max_{\mathbf{p} \in \text{perm}(n)} l_{\text{NDCG}}(\mathbf{s}, \mathbf{p}) + C(\mathbf{s}'^k, \mathbf{p}, \mathbf{c})$$

$$\mathbf{g}^k = \sum_{i=1}^n (c_{\hat{p}_i^k} - c_i) \mathbf{x}_i + \Omega'(\mathbf{w}^k)$$

IF $k > 0$ THEN

$$\mu^k = \mu^{k-1} (\mathbf{g}^{k-1T} \mathbf{g}^{k-1}) / ((\mathbf{g}^{k-1} - \mathbf{g}^k)^T \mathbf{g}^{k-1})$$

$$\mathbf{w}^{k+1} = \mathbf{w}^k - \mu^k \mathbf{g}^k$$

$$k = k + 1$$

WHILE $\|\mathbf{w}^k - \mathbf{w}^{k-1}\| > \varepsilon$

RETURN $\mathbf{w}^k, \mathbf{s}'^k$

Authority-Ranking

Motivation

□ Beispiele:

- Ranking von Webseiten, z.B. durch Analyse der Verlinkungsstruktur.
- Reputation von Benutzern in sozialen Netzwerken.
- Finden geeigneter Literatur-Referenzen und Domain-Experten (zugehörige Autoren).



Authority-Ranking

Problemstellung

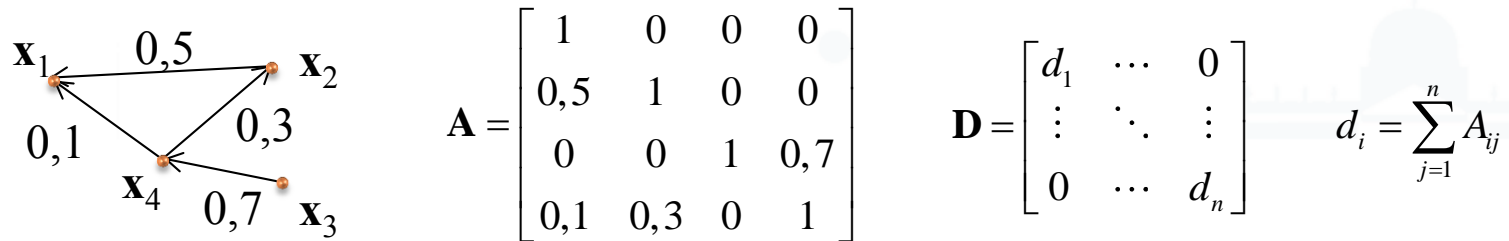
- Gegeben: Trainingsdaten $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$ mit gegebenen lokalen Authority Scores (Kompetenz-Bewertungen):
 - Authority Score $A_{ij} = a(\mathbf{x}_i, \mathbf{x}_j)$ gibt an wie „kompetent“ \mathbf{x}_j aus Sicht von \mathbf{x}_i ist. Authority Matrix
- Gesucht: Modell $f : \mathbf{A} \in \mathbb{R}^{n \times n} \mapsto \mathbf{s} \in \mathbb{R}^n$ welches für Instanzen $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$ globale Authority Scores (Ranking) s_j liefert.
- Annahme: Je kompetenter \mathbf{x}_i und je höher der Authority Score A_{ij} , desto kompetenter \mathbf{x}_j .

Authority-Ranking

Lösungsansatz

□ Modellierung der (nicht-symmetrischen) Kompetenz-Bewertungen als ungerichteten Graphen:

- Instanzen sind Knoten, Authority Scores sind Kanten-Gewichte
 \Rightarrow Authority Matrix = Adjazenzmatrix \mathbf{A} .



- Intuition: (normalisierter) Graph beschreibt mit welcher Wahrscheinlichkeit Knoten x_i Knoten x_j als „Experten“ nennen würde.
- Beispiel: x_4 hält x_2 für 3-mal so kompetent wie x_1 , x_3 ist aus seiner Sicht kein „Experte“.

Authority-Ranking

Lösungsansatz



□ Random Surfer:

- Beginnend bei einem beliebigen Knoten „fragt“ man sich (unendlich lang) durch.
- Wahrscheinlichkeit mit der ein Knoten insgesamt gefragt wurde = globaler Authority Score.

□ Hubs & Authorities:

■ Jeder Knoten besitzt...

- *Hub Score*: Wie gut sind seine Verweise auf „kompetente“ Knoten = ausgehende Kanten.
- *Authority Score*: Wie kompetent ist der Knoten, d.h. wie viele kompetente Knoten verweisen auf ihn = eingehende Kanten.

Authority-Ranking

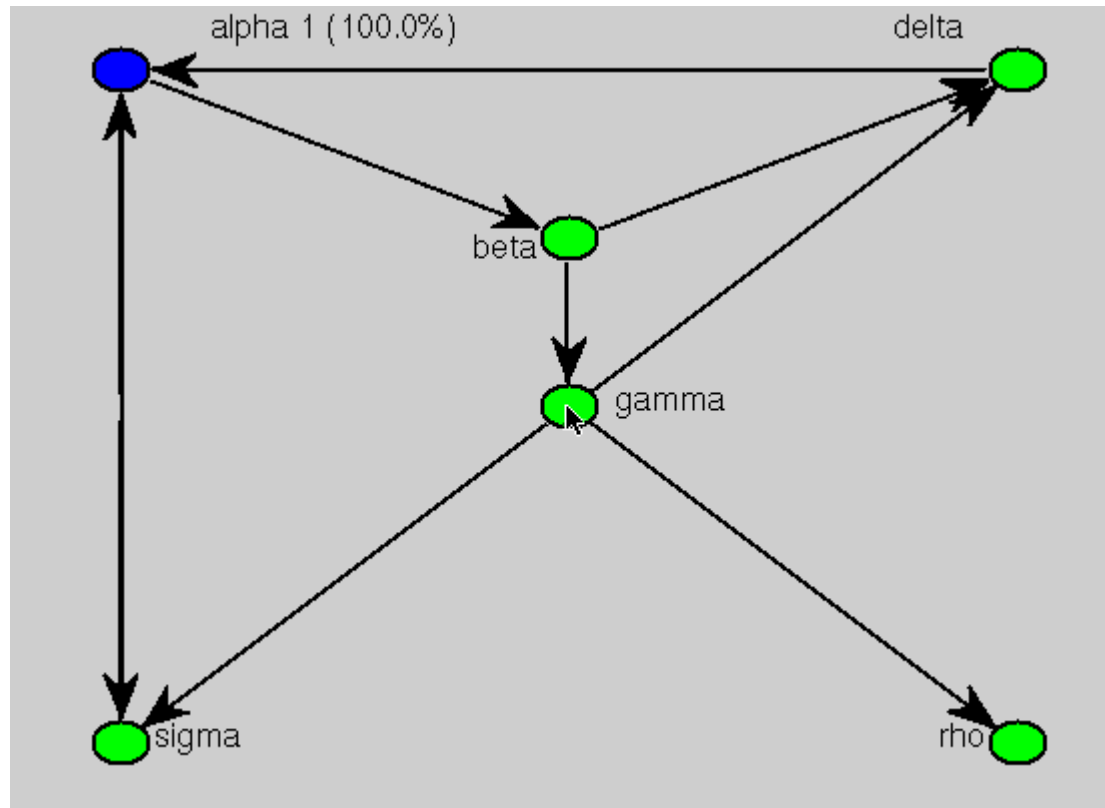
Random Surfer: PageRank

- PageRank: Random Surfer Modell zum Ranking von Webseiten.
 - (Ursprünglicher) Ranking-Algorithmus von Google, Fireball etc.
 - Abhängig von Query werden relevante Webseiten gefunden und nach ihrem globalen Authority Score sortiert.
- Annahmen:
 - Link auf Webseite x_i verweist auf „kompetente“ Webseite x_j , d.h. Adjazenzmatrix des Webgraphen = Authority Matrix mit $A_{ij} = 1$.
 - Mit Wahrscheinlichkeit $1 - \varepsilon$ folgt der Nutzer (Random Surfer) einem Link auf der Webseite.
 - Mit Wahrscheinlichkeit ε wechselt er auf eine zufällige Webseite.

Authority-Ranking

Random Surfer: PageRank

□ Beispiel:



Authority-Ranking

Random Surfer: PageRank

- Gegeben: Wahrscheinlichkeit dafür, dass der Nutzer von Webseite x_i zu Webseite x_j wechselt ist

$$P_{ij} = p(\mathbf{x}_j | \mathbf{x}_i) = (1 - \varepsilon) \frac{A_{ij}}{\sum_{k=1}^n A_{ik}} + \varepsilon \frac{1}{n}$$

Transitionswahrscheinlichkeit

und somit $\mathbf{P} = (1 - \varepsilon)\mathbf{D}^{-1}\mathbf{A} + \varepsilon\mathbf{U}$ mit $U_{ij} = \frac{1}{n}$.

- Gesucht: Wahrscheinlichkeit dafür, dass man auf Webseite x_i ist, d.h. $s_i = p(\mathbf{x}_i)$.

Aufenthaltswahrscheinlichkeit

Authority-Ranking

Random Surfer: PageRank

- Algorithmus: Beginnend mit initialen Ranking Scores s iterativ neue Scores bestimmen mit

$$s' = \frac{1}{c} \mathbf{P}^T s \quad \text{wobei} \quad c = \|\mathbf{P}^T s\| \Rightarrow \|s'\| = 1$$

- Konvergenz von PageRank bei $s' = s$, sodass gilt

$$s = \frac{1}{\lambda} \mathbf{P}^T s$$

d.h. s ist ein Eigenvektor von \mathbf{P}^T mit Eigenwert λ .

- Man kann zeigen, dass s der Eigenvektor mit größtem Eigenwert λ ist (Analogie zu Spectral Clustering).

Authority-Ranking

Random Surfer: PageRank

□ Vorteile:

- Leicht und effizient berechenbar.
- Existenz & Eindeutigkeit der Lösung sowie Konvergenz des Algorithmus ist garantiert für $0 < \varepsilon < 1$.

□ Nachteile:

- Links können schlechte Indikatoren für Kompetenz-Bewertung sein:
 - Kompetenz zweier Instanzen kann sich gegenseitig verstärken.
 - Automatisch generierte Links haben kaum Aussagekraft.
 - „Künstliche“ Links (z.B. Link-Spam) verändern Ranking.
- Eigenschaften der Instanzen fließen nicht ins Ranking ein.

Authority-Ranking

Hubs & Authorities: HITS

- Hypertext Induced Topic Search (HITS):
 - Query-abhängige Auswahl relevanter Webseiten R .
 - Webseiten R , alle auf R verweisenden Webseiten und alle von R direkt erreichbaren Webseiten bilden Teilgraphen mit Adjazenzmatrix A .
 - Für diesen Teilgraphen Hub- und Authority-Scores bestimmen und Webseiten bzgl. Authority-Scores sortieren.
- Annahmen:
 - Hubs zeigen auf viele Authorities.
 - Authorities werden von vielen Hubs erwähnt.

Authority-Ranking

Hubs & Authorities: HITS

- *Hub Score* h_i einer Webseite x_i ist (normierte) Summe der *Authority Scores* s_j der Webseiten x_j auf welche x_i verweist:

$$h_i = \frac{1}{d} \sum_{j=1}^n A_{ij} s_j \Rightarrow \mathbf{h} = \frac{1}{d} \mathbf{A} \mathbf{s} \quad \text{wobei} \quad d = \|\mathbf{A} \mathbf{s}\| \Rightarrow \|\mathbf{h}\| = 1$$

- *Authority Score* s_i einer Webseite x_i ist (normierte) Summe der *Hub Scores* h_j der Webseiten x_j welche auf x_i verweisen:

$$s_i = \frac{1}{c} \sum_{j=1}^n A_{ji} h_j \Rightarrow \mathbf{s} = \frac{1}{c} \mathbf{A}^T \mathbf{h} \quad \text{wobei} \quad c = \|\mathbf{A}^T \mathbf{h}\| \Rightarrow \|\mathbf{s}\| = 1$$

Authority-Ranking

Hubs & Authorities: HITS



- Gegenseitiges Einsetzen ergibt:

$$\mathbf{h} = \frac{1}{d} \mathbf{A} \mathbf{s} = \frac{1}{d} \mathbf{A} \left(\frac{1}{c} \mathbf{A}^T \mathbf{h} \right) = \frac{1}{\lambda} \mathbf{A} \mathbf{A}^T \mathbf{h}$$

$$\mathbf{s} = \frac{1}{c} \mathbf{A}^T \mathbf{h} = \frac{1}{c} \mathbf{A}^T \left(\frac{1}{d} \mathbf{A} \mathbf{s} \right) = \frac{1}{\lambda} \mathbf{A}^T \mathbf{A} \mathbf{s}$$

- Analog zu PageRank ist \mathbf{h} der Eigenvektor von $\mathbf{A} \mathbf{A}^T$ und \mathbf{s} der Eigenvektor von $\mathbf{A}^T \mathbf{A}$ mit jeweils größtem Eigenwert.
- Algorithmus: Iterative, abwechselnde Berechnung der Hub- und Authority-Scores.

Authority-Ranking

Hubs & Authorities: HITS

- Vorteile:
 - Effizient berechenbar.
 - Unterscheidung in Hubs & Authorities.
- Nachteile:
 - Links können schlechte Indikatoren für Kompetenz-Bewertung sein; anfällig für Link-Spam.
 - Lösung muss nicht eindeutig sein (Lösung ist dann abhängig von Initialisierung).
 - Leichte Veränderung des Graphen kann unter Umständen große Änderung des Ranking verursachen.

Authority-Ranking

Erweiterungen



- Kombination beider Ansätze:
 - Stochastic Approach for Link Structure Analysis (SALSA).
- Modell für Transitionswahrscheinlichkeiten:
 - Eigenschaften der verlinkten Seite (z.B. Aktualität, inhaltliche Ähnlichkeit zur verlinkenden Seite, Relevanz bzgl. der Query).
 - Eigenschaften der Verlinkung (z.B. Position, Anchor-Text).
 - Struktur des umgebenden Teilgraphen.
- Personalisierung:
 - Benutzerabhängige Transitionswahrscheinlichkeiten.
 - Re-Ranking nach User-Feedback (Click-Stream-Analyse).

Zusammenfassung

- Rank-Learning: Lernen eines Modells basierend auf Daten mit gegebenem Ranking.
 - Zahlreiche (nicht-konvexe) Ranking-Kriterien.
 - Ein Lösungsansatz: Konvexe Approximation für Kriterium finden und diese optimieren (z.B. RankSVM, CoFiRank).
- Authority-Ranking: Lernen eines Modells basierend auf lokalen Relevanz-Bewertungen.
 - Modellierung als Graph.
 - Finden eines globalen Rankings durch Lösen eines Eigenwertproblems (z.B. PageRank, HITS, SALSA).