Learning to Rank (CS635)

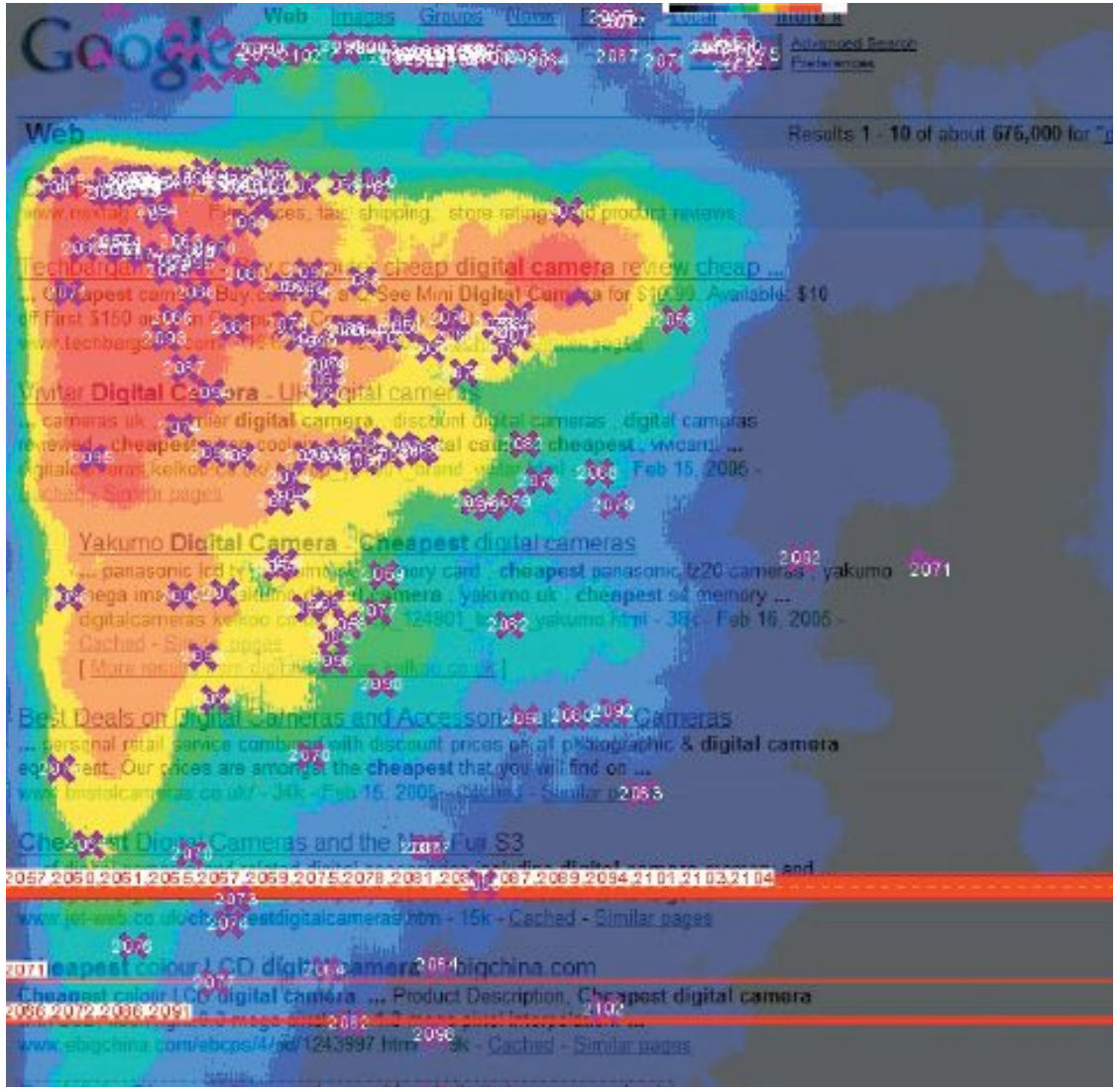
Soumen Chakrabarti
+ MRS slides from Stanford
+ Slides from Yisong Yue

Why is ML needed in ranking?

- Modern systems – especially on the Web – use a great number of features
- Arbitrary useful features – not a single unified model
  - Log corpus frequency of query word in anchor text?
  - Query word in color on page?
  - Number of images on page?
  - Number of (out) links on page?
  - PageRank of page?
  - URL length?
  - URL contains “~”?
  - Page edit recency?
  - Page loading speed
- The New York Times in 2008-06-03 quoted Amit Singhal as saying Google was using over 200 such features (“signals”)

Eye tracking on SERP

- SERP = search engine response page
- Scan down, inspect summary, (possibly) click, inspect document contents, update mental state, continue/abandon



Utility-based ranking principle

- Attention decays with rank
  - May be owing to fatigue or satisfaction
- Query q, whole corpus D
- Items indexed $i, j \in [|D|]$, called x_i, x_j, etc.
- Place most useful items (docs) first
- Utility of item i wrt query q is $f(q, x_i) \in \mathbb{R}$
  - Learnt scoring function
  - Induces total order on items
  - $f(q, x_i)$ is not a function of any other x_j — this is a big assumption

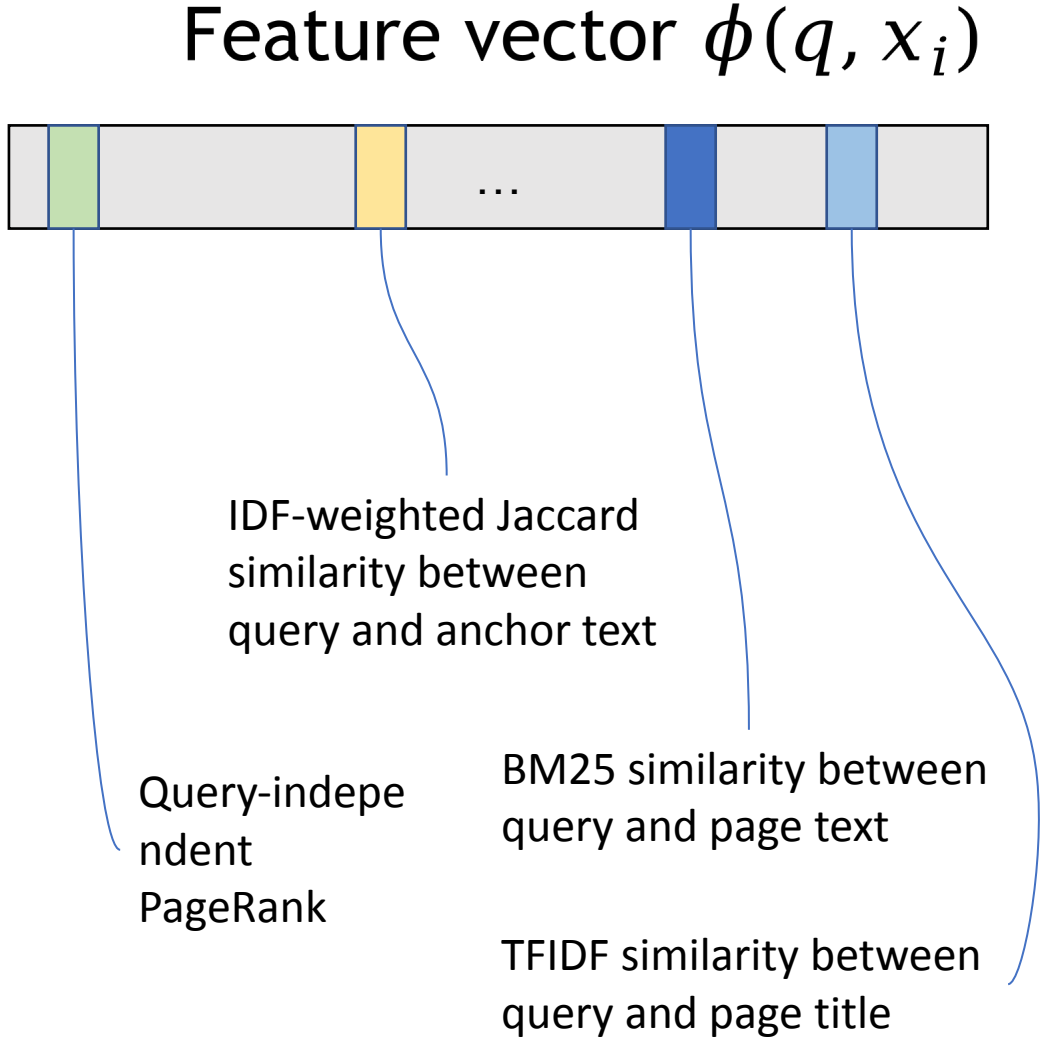
Interactions between items usually handled after a first round of filtering

Quick review of classification

- M features and K labels
  - Features $\varphi(x) \in \mathbb{R}^M$ as properties of text x
  - Or both text and labels $\varphi(x, y) \in \mathbb{R}^D$
- Model weights per label $w_y \in \mathbb{R}^M$ or global $w \in \mathbb{R}^D$
- SVM; logistic regression
  - $\text{score}(y|x) = w \bullet \varphi(x, y)$ or $w_y \bullet \varphi(x)$
- Learn features as well as classifier?

Key step in L2R: Design and learn f

- Linear scoring function $f(q, x_i) = w \cdot \phi(q, x_i)$ with $w, \phi \in \mathbb{R}^M$
- Non-linear: regression tree, neural network, etc.
- Pre-aggregated features (as shown), exploiting human intuition about scoring
- Can also add lexicalized features (rediscover IDF?)



Key issue in L2R: Form of supervision

- Tradeoff between human effort vs. ranking quality
- Gold scores: continuous ($\mathbb{R}$, unrealistic) or ordinal ($[R] = \{0, \ldots, R\}$, more realistic)
- Complete or partial lists (unrealistic)
- Implicit pairwise preferences inferred from clicks (noisy but practical and cheap)

Key step in L2R: loss design

- Additive over single items: “itemwise”
  - Easy to write down and optimize
  - Unrealistic to get broad-coverage, absolute, itemwise judgment
- Additive over item pairs: “pairwise”
  - As easy to write down, more expensive to optimize
  - Practical to collect from clicks stats
  - Ignores absolute ranks: flipping 1&2 much more serious than flipping 12&13
- Non-additive, “listwise”
  - Optimize set selection or *collective* ranking
  - Hopefully closer to “true” (user satisfaction) objective
  - Much more difficult to model and optimize

Relevance scoring evolution

Hardwired

- TF and IDF
- BM25
  - Fielded
  - Length normalization

Generative[(1)]

- Model: document sampled to get query
- Document induces “language model”
- Extensions for cooccurrence and proximity

Loss-based

- Query + relevant docs
- System generates score and rank for all docs
- If relevant docs pushed down, update scoring model

(1) “Generative retrieval” now means something quite a bit different.

Loss-based learning to rank

- Characterize (query, doc) pair with a feature vector
  - (Many) sparse lexicalized features: scales with vocabulary
  - (Fewer) dense aggregated features: TFIDF, fielded matches, PageRank, etc.
- Scorer is a function $f_\theta(q, d) \in \mathbb{R}$ which induces a ranking
  - If gold documents have poor ranks, update scoring model parameters θ
  - Define a loss to measure "poor ranks"

Warmup: least-squares regression

- System outputs score $f_\theta(q, d) \in \mathbb{R}$
  - A simple choice is a linear function $f_\theta(q, d) = \theta \cdot \phi(q, d)$
  - Here $\theta,\ \phi(q, d) \in \mathbb{R}^M$
  - No trainable params inside 'encoder' ϕ (for now)
- Square loss is $\sum_{q,d} \left(f_\theta(q, d) - y_{q,d}\right)^2$
- Minimize this wrt θ
  - If f_θ is linear, then there is a closed form for θ
  - Otherwise, use gradient descent to train θ

Warmup: (Linear) support vector machine

- If binary labels $y_{q,d} \in \pm 1$ were provided, we could use a simple SVM
- Prediction $\widehat{y}_{q,d} = \text{sign}(\theta \cdot \phi(q,d))$
  - When $y_{q,d} = 1$, we want $\theta \cdot \phi(q,d) > 0$
  - When $y_{q,d} = -1$, we want $\theta \cdot \phi(q,d) < 0$
- Introduce margin and define loss
  - $[\![y_{q,d} = 1]\!] \, [\![\theta \cdot \phi(q,d) < 1]\!]$
  - $[\![y_{q,d} = -1]\!] \, [\![\theta \cdot \phi(q,d) > -1]\!]$
- Unified constraint $y_{q,d} \, (\theta \cdot \phi(q,d)) \geq 1$
- Leading to "hinge" (called "ReLU" in the AI era) loss

$$\min_{\theta} \lambda \|\theta\|_2^2 + \sum_q \frac{1}{|D|} \sum_{d \in D} \max\{0, 1 - y_{q,d} \, (\theta \cdot \phi(q,d))\}$$

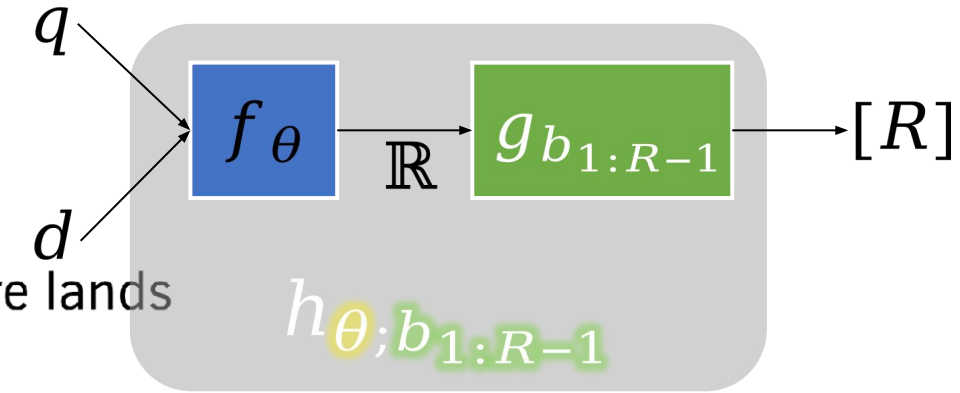
This is a well-studied quadratic program with linear constraints, for which super-optimized solvers like SVMlight are available

$\max\{0, a\}$ is also written as $[a]_+$

This loss is "itemwise"

Ordinal regression

- Star rating as in ecommerce, $y_{q,d} \in \{1, \ldots, R\}$ are ordinal
- System also outputs $h_{\theta;b_{1:R-1}}(q, d) \in \{1, \ldots, R\}$
- System parameters are $\theta; b_{1:R-1}$ — will describe these soon
- Perf score may be MAE, $\sum_{q,d} \left| h_{\theta;b_{1:R-1}}(q, d) - y_{q,d} \right|$
- Asking $h_{\theta;b_{1:R-1}}$ to directly target evenly spaced integer labels may be too demanding
- System predicts label in two stages
  - First apply f_θ to get a real score
  - Find bucket among $-\infty = b_0 < b_1 < \cdots < b_{R-1} < b_R = \infty$ where this score lands
  - Return index of this bucket



Ordinal (linear) regression

- Assume $f_\theta(q, d) = \theta \cdot \phi(q, d)$
- Let the ith instance in the rth relevance level be d_i^r
- Then we want $b_{r-1} < \theta \cdot \phi(q, d_i^r) < b_r$
- As before, use margins: $b_{r-1} + 1 \leq \theta \cdot \phi(q, d) \leq b_r - 1$
- If either does not happen, assess hinge losses
  - $[\![1 + b_{r-1} - \theta \cdot \phi(q, d) \leq 0]\!] \longrightarrow \mathrm{ReLU}(1 + b_{r-1} - \theta \cdot \phi(q, d))$
  - $[\![\theta \cdot \phi(q, d) - b_r + 1 \leq 0]\!] \longrightarrow \mathrm{ReLU}(\theta \cdot \phi(q, d) - b_r + 1)$
- Two hinges form a tub
- These are all linear in the variables $\theta; b_{1:R-1}$
- Large norm $\|\theta\|_2^2$ still needs to be penalized / regularized as in SVM
- Leads to a quadratic program with linear constraints (SVOR)

Pair preferences

- From clickthrough, can judge if $d_i \prec d_j$ (ith doc worse than jth doc) or $d_i \succ d_j$ (better) for a given query
  - 'Label' for a query is a set of pairs (i, j) ($\prec$ is implicit)
- Incomplete, noisy and inconsistent
- In case of a clean situation where gold good, bad doc sets $D_{q\oplus}$, $D_{q\ominus}$ are identified
  - May choose to exhaustively introduce pair preferences $g \succ b$ where $g \in D_{q\oplus}$ and $b \in D_{q\ominus}$
  - But this may be excessive, mey not be required

Pairwise eval and loss

- System score $f_\theta(q, d) \in \mathbb{R}$ used to rank
  - Assume no score ties, or resolve arbitrarily
- Gold good, bad doc sets $D_{q\oplus}$, $D_{q\ominus}$
- Say $N_{q\oplus} = |D_{q\oplus}|$, $N_{q\ominus} = |D_{q\ominus}|$
- Any good doc score should exceed any bad doc score
- I.e., want $f_\theta(q, g) \gg f_\theta(q, b)$
- Suggests a loss

$$\frac{1}{N_{q\oplus}\, N_{q\ominus}} \sum_{g \in D_{q\oplus}} \sum_{b \in D_{q\ominus}} [\![f_\theta(q, g) < f_\theta(q, b)]\!]$$

- Total or expected number of reversed pairs
- During test/inference, sort docs d by decreasing $f_\theta(q, d)$

RankSVM (linear f_θ)

- Again, employ a margin to rewrite the loss as

$$\min_\theta \lambda\|\theta\|_2^2 + \frac{1}{N_{q\oplus} N_{q\ominus}} \sum_{\substack{g \in D_{q,\oplus} \\ b \in D_{q\ominus}}} \mathrm{ReLU}(1 + \theta \cdot \phi(q, b) - \theta \cdot \phi(q, g))$$

- Another quadratic program with linear constraints
- A lot of constraints: $\sum_q N_{q\oplus} N_{q\ominus}$
- Many of these are redundant
- Clever tricks to avoid introducing unnecessary constraints
- Despite limitations of pairwise loss, RankSVM widely used
- In the AI age, $f_\theta(q, d)$ becomes non-linear, optimization becomes non-convex, and $b \subset D_{q\ominus}$ is carefully sampled

Can prove this sum is an upper bound on the number of discordant pairs in training fold

AP vs pairwise loss

- System score $f_\theta(q, d) \in \mathbb{R}$ used to rank D into $\overrightarrow{D}$
- Write $D_\oplus$ followed by $D_\ominus$ such that order among good (and bad) docs same as in $\overrightarrow{D}$
- Count the number of reversals (discordant pairs) Q
- With $|D_\oplus| = R$, can show that

$$\text{AP} \geq \frac{\left(\sum_{1 \leq i \leq R} \sqrt{i}\right)^2}{R\left(Q + \binom{R+1}{2}\right)}$$

- In other words, if Q is small, AP will be large

Recap

- (Query, document) featurization
- Itemwise decomposable learning to score
  - Every item has a relevance label
  - Regression, ordinal regression
- Pairwise decomposable scoring
  - Each pair has a relative relevance label
  - Hinge loss on margin-augmented difference of bad and good scores
- What if the label is a ranking of “all” items?

Slight notation overload ahead

(but not math complications)

Warmup: multiclass logistic regression

- Two-class LR is usually written as (with $\boldsymbol{w}, \boldsymbol{x} \in \mathbb{R}^M$)

$$\Pr(Y = 1|\boldsymbol{x}) = \frac{1}{1 + e^{-\boldsymbol{w}\cdot\boldsymbol{x}}} = \frac{e^{\boldsymbol{w}\cdot\boldsymbol{x}}}{e^{\boldsymbol{w}\cdot\boldsymbol{x}} + e^{\vec{0}\cdot\boldsymbol{x}}}$$

- At a slight overparameterization, for K classes, we can write

$$\Pr(Y = k|x) = \frac{e^{\boldsymbol{w}_k\cdot\boldsymbol{x}}}{\sum_{m'} e^{\boldsymbol{w}_{k'}\cdot\boldsymbol{x}}}$$

- One model weight vector $\boldsymbol{w}_k \in \mathbb{R}^M$ for each class $k \in [K]$ (total KM weights)
- Another view is that is $\boldsymbol{w} \in \mathbb{R}^{K\times M}$ now a weight matrix
- Feature map $\Phi(\boldsymbol{x}, y) \in \mathbb{R}^{K\times M}$ needed for compatibility
  - Choosing $\Phi(\boldsymbol{x}, k)[k, :] = \boldsymbol{x}$ and $\Phi(\boldsymbol{x}, k)[\neq k, :] = \vec{0}$ makes two notations equivalent
- Observe that $\boldsymbol{w} \cdot \Phi(\boldsymbol{x}, k) = \boldsymbol{w}[k, :] \cdot \boldsymbol{x}$ equivalent to $\boldsymbol{w}_k \cdot \boldsymbol{x}$
- Most likely class is $\underset{k\in[K]}{\operatorname{argmax}}\, \boldsymbol{w} \cdot \Phi(\boldsymbol{x}, k)$

Benefits of general notation

- Earlier we chose $\Phi(\mathbf{x}, k)[\underline{k}, :] = \mathbf{x}$ and $\Phi(\mathbf{x}, k)[k' \neq k, :] = 0$
- What if k', k are related?
  - E.g., k' is a descendant of k in a topic hierarchy
- More generally, can factor $\Phi(\mathbf{x}, k) = \varphi(x) \bullet \gamma(k)$
  - I.e., a document and a class can each have their own features
  - E.g., class can have textual description
- Or, each label k can index a whole permutation

Notes about notation

- Traditional supervised ML
  - Labeled iid instance $(\mathbf{x}_i, y_i)$ where $\mathbf{x}_i \in \mathbb{R}^M$
- Labeled data in learning to rank (L2R) is *nested*
  - $q \in Q$ is a set of queries
  - For each query q, a set of N_q candidate documents
  - Each document denoted as d_i
  - $\mathbf{x}_{q,i} = \varphi(d_i, q) \in \mathbb{R}^M$ is a query-dependent feature vector
  - All collected into matrix $\mathbf{X}_q \in \mathbb{R}^{N_q \times M}$
  - $\mathbf{X}_q[i, \star] = \mathbf{x}_{q,i} = \varphi(d_i, q)$
  - For collective labeling, system may have to see whole of $\mathbf{X}_q$

Simple to structured labels

- Regression has simple labels $y_i \in \mathbb{R}$
  - Can look at whole $\boldsymbol{X}_q$, but only $\boldsymbol{x}_i$ needed
- Ordinal regression has $y \in [R]$ for R-level ratings
- Multi-class classification has $y \in [K]$
- Subset retrieval: $y \in \{0,1\}^{N_q}$; gold $y_q[i] = [\![x_i \in D_{q\oplus}]\!]$
- Note, y or y_q is not the relevance estimate of just *one* item
- How about pairwise comparisons?
  - $N_{q\oplus} = |D_{q\oplus}|$, $N_{q\ominus} = |D_{q\ominus}|$, $N_q = |D_q| = |D_{q\oplus} \cup D_{q\ominus}|$
  - To predict $y_{q;\, i,j}$, enough to see $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$?
  - How about consistency across all of $\boldsymbol{X}_q$?

Structured labels

- All-pairs comparison: $y \in \{-1, 0, +1\}^{N_q \times N_q}$
  - Zero means no preference (diagonals and item pairs with same relevance level)
  - The matrix of y is skew-symmetric: $y[i, j] = -y[j, i]$
  - The number of such matrices is exponential in N_q, but only a subset is consistent with a total order
  - Instead of ± 1, could also be *counts*: how many times item i has been preferred to item j
- Permutation: $y \in \mathcal{S}_{N_q}$, the set of all permutations of N_q items
  - The number of possible 'labels' is now $N_q!$
  - y can be represented by an $N_q \times N_q$ permutation matrix

Structured loss

- Non-negative loss $\Delta(y_q, y)$
  - y_q is the gold collective label for query q
  - y or $\widehat{y}$ is a candidate collective label (such as output by a system)
  - y_q and y may not always have the same type signature
- $\Delta = 0$ if $y = y_q$
  - If y is close to gold, Δ is small; if y is terrible, Δ is large
- Regression
  - $y_q,\ y \in \mathbb{R}^{N_q}, \Delta(y_q, y) = \sum_{i \in D_q} (y_q[i] - y[i])^2$
- Pairwise preferences
  - $y_q, y \in \{-1, +1\}^{N_q \times N_q}, \Delta(y_q, y) = \sum_{i,j} [\![y_q[i,j] \neq y[i,j]]\!]$
  - Discordant pairs

Structured loss (pairwise)

- What if gold $y_q \in \mathbb{Z}^{N_q \times N_q}$ and system $y \in \{-1, +1\}^{N_q \times N_q}$?
- How about $\sum_q \sum_{i,j} \mathrm{ReLU}(-y_q[i,j]\ y[i,j])$
  - If signs agree, no penalty
  - Otherwise penalty scales with $y_q[i,j]$
  - Related to $\sum_q \sum_{i \prec j} [\![y_q[i,j] > 0]\!]$, summing over multisets $\{\{i \prec j\}\}$ (including potential contradictions)

Structured loss (listwise)

- Listwise preferences: $y_q, y \in \mathcal{S}_{N_q}$
- Can use various rank correlations
  - Unlikely to be given full gold permutation over all D_q
  - More likely that $y \in \mathcal{S}_{N_q}$ but $y_q \in \{-1, 0, +1\}^{N_q}$
- We have seen how to compute MAP, MRR and NDCG as procedures
  - Subtracting from 1, we can get $\Delta_{\text{MAP}}, \Delta_{\text{MRR}}, \Delta_{\text{NDCG}}$

Structured loss (setwise)

- Now y, y_q are subsets of candidate docs
  - May be encoded as bit vectors $\{0,1\}^{N_q}$
- Losses may be $1 -$ recall, $1 -$ precision, $1 - F_1$, etc.
- Most interesting collective losses do not have smooth functional forms capable of being directly plugged into neural networks
  - Design surrogates or employ other tricks

Feature map and inference

- Feature map $\Phi(X_q, y) \in \mathbb{R}^D$ (how to design Φ?)
- Note, candidate structured label y is an *input* to feature map
- Different from individual item's feature vector $\phi(x, q) \in \mathbb{R}^M$ (no y) with item score $\theta \cdot \phi(x, q)$
- Often rows of X_q are $\phi(x, q)$ for $x \in D_q$, i.e., $X_q \in \mathbb{R}^{N_q \times M}$ $(D = N_q M)$
- Model parameters $w \in \mathbb{R}^D$ to be consistent with $\Phi(X_q, y)$
- Score $w \cdot \Phi(X_q, y) \in \mathbb{R}$ tells us how compatible observed features X_q are with proposed *structured* label y, as estimated using model weights w
- Inference amounts to finding $\underset{y}{\operatorname{argmax}}\, w \cdot \Phi(X_q, y)$
- Space of collective structured labels y can be very large
  - E.g., all permutations, all matrices with $\{-1,\ 0, +1\}$ elements

Learning w via distributions

- Standard learning objective (empirical loss minimization)

$$\underset{w}{\operatorname{argmin}} \sum_q \Delta\left(y_q,\ \underset{y}{\operatorname{argmax}}\{w \cdot \Phi(X_q, y)\}\right)$$

- Objective usually not differentiable (or even continuous) wrt w
  - Smooth surrogates
  - Convex surrogates
- Use score to induce a probability distribution

$$\Pr(y|X_q; w) \propto \exp(w \cdot \Phi(X_q, y))$$

- Calculating normalizing denominator $Z_w(X_q) = \exp\left(\sum_{y'} w \cdot \Phi(X_q, y')\right)$ generally intractable
- Expected loss objective

$$\underset{w}{\operatorname{argmin}} \sum_q \sum_y \Delta(y_q,\ y) \Pr(y|X_q; w)$$

- Not convex, but can try gradient descent

Learning w via score margins

- Recall inference means finding $\underset{y}{\operatorname{argmax}}\{w \cdot \Phi(X_q, y)\}$
- Therefore, we want $w \cdot \Phi(X_q, y_q) \gg w \cdot \Phi(X_q, y)$ for all $y \neq y_q$
- Let the gap/margin depend on how bad y is, compared to y_q
- Want $w \cdot \Phi(X_q, y_q) \geq w \cdot \Phi(X_q, y) + \Delta(y_q, y)$
  - Includes special case of $y = y_q$ with $\Delta = 0$
- To protect against infeasibility, introduce slack $\xi_q \geq 0$

$$\xi_q + w \cdot \Phi(X_q, y_q) \geq w \cdot \Phi(X_q, y) + \Delta(y_q, y)$$

- If y_q beats y without help, $\xi_q = 0$
- Needing $\xi_q > 0$ amounts to defeat, so include in the objective the term $(1/Q)\sum_q \xi_q$ (averaging normalizes the loss scale)

Structured SVM

- Including SVM regularization, the objective is

$$\min_{w,\{\xi_q \geq 0\}} \|w\|_2^2 + \frac{C}{Q} \sum_q \xi_q$$

- Objective is quadratic (p.s.d.) in variables
- Subject to *lots* of constraints:
  for each q, and for each structured label y,

$$\xi_q + w \cdot \Phi(X_q, y_q) \geq w \cdot \Phi(X_q, y) + \Delta(y_q, y)$$

- Constraints are all half-spaces wrt variables
- Does not look like much progress yet

Incremental constraint inclusion

- Maintain working set of constraints $\mathcal{C}$
  - Starting with $\mathcal{C} = \emptyset$
- Find w (and thereby $\{\xi_q\}$) for this $\mathcal{C}$
- Find if any constraint is violated
- Want $\xi_q + w \cdot \Phi(X_q, y_q) \geq w \cdot \Phi(X_q, y) + \Delta(y_q, y)$ violated
- Therefore maximize the rhs over all y:

$$\breve{y} = \operatorname*{argmax}_y \; w \cdot \Phi(X_q, y) + \Delta(y_q, y)$$

Loss-augmented inference

t the loss
or $\breve{y}$ is large

scores $\breve{y}$ as a good label

- "Loss augmented inference"
- $\breve{y}$ is a potential certificate that w is not perfectly trained yet
- Introduce new constraint $\xi_q + w \cdot \Phi(X_q, y_q) \geq w \cdot \Phi(X_q, \breve{y}) + \Delta(y_q, \breve{y})$ into $\mathcal{C}$

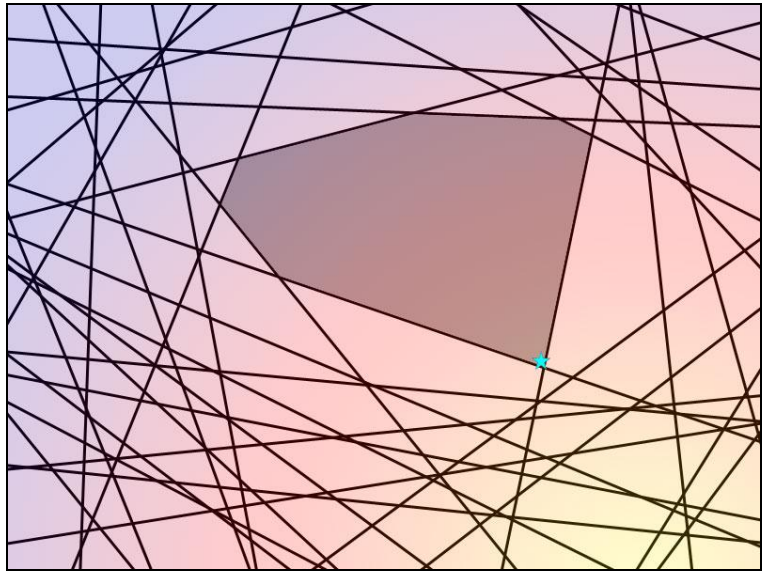
Unless violation is small

- Find next w with augmented $\mathcal{C}$ and repeat

This constraint is linear wrt w

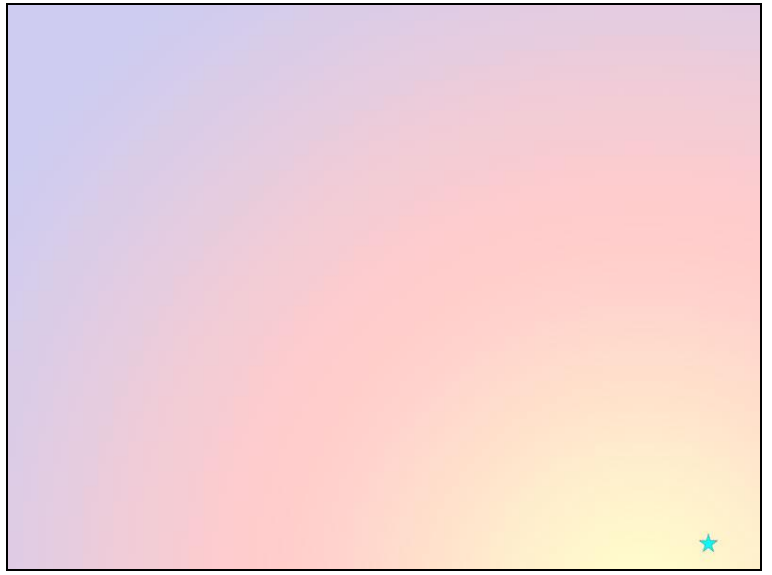
The "cutting plane method"

Illustrative Example



Original SVM Problem

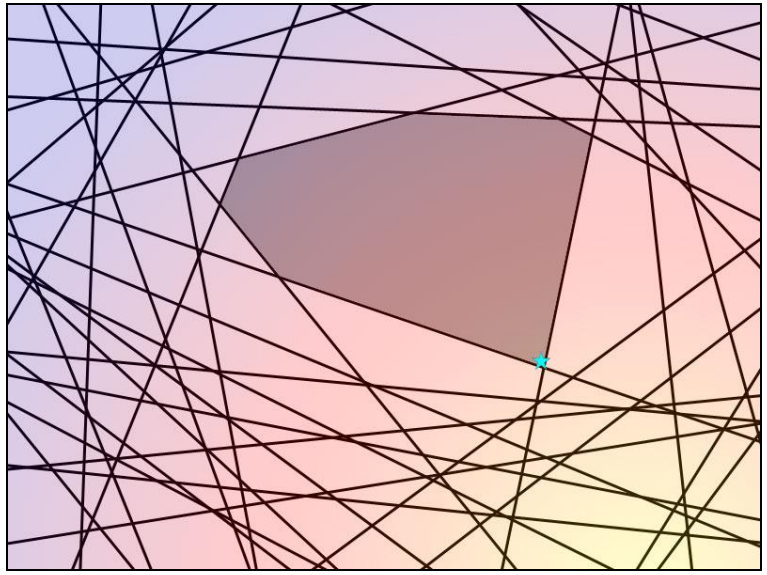
- Exponential constraints
- Most are dominated by a small set of “important” constraints



Structural SVM Approach

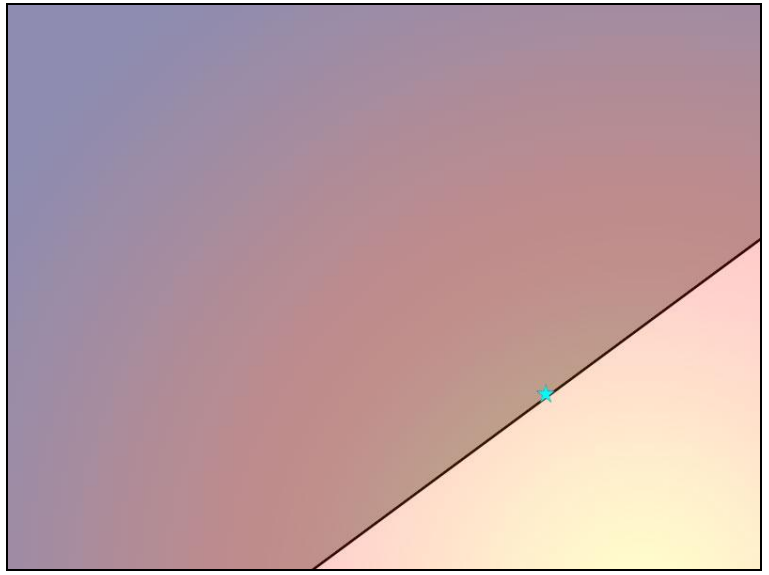
- Repeatedly finds the next most violated constraint…
- …until set of constraints is a good approximation.

Illustrative Example



Original SVM Problem

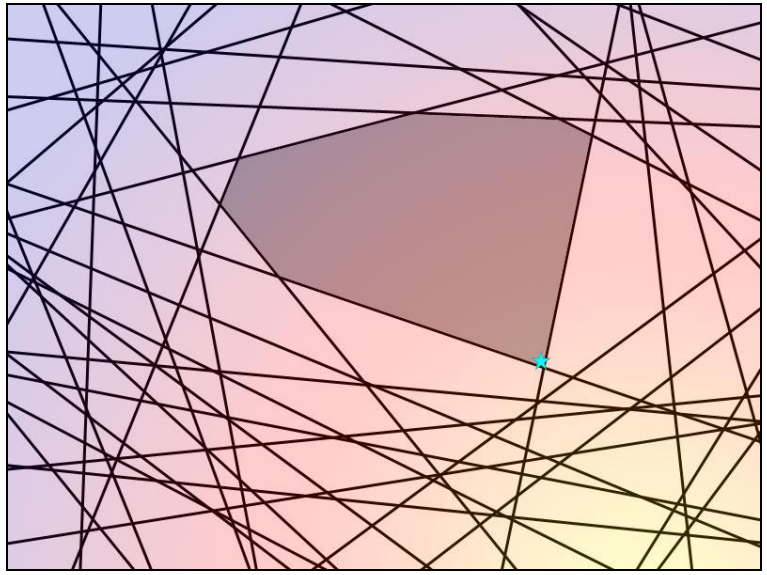
- Exponential constraints
- Most are dominated by a small set of “important” constraints



Structural SVM Approach

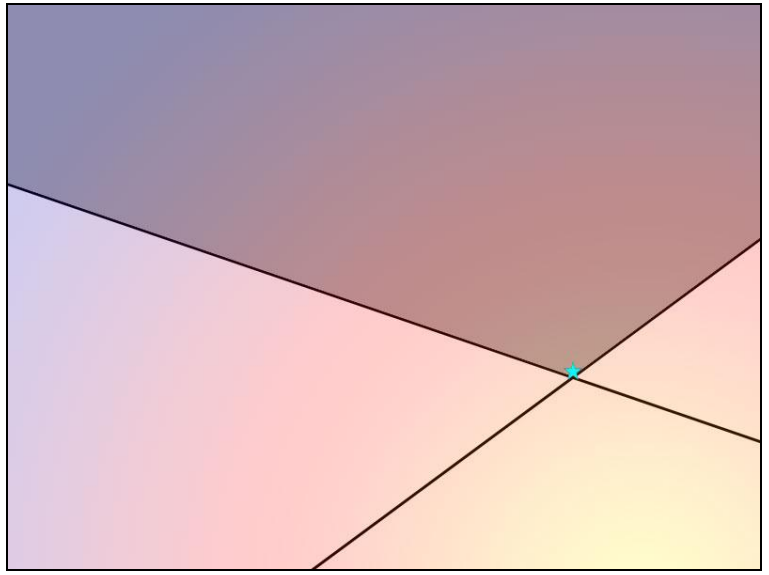
- Repeatedly finds the next most violated constraint…
- …until set of constraints is a good approximation.

Illustrative Example



Original SVM Problem

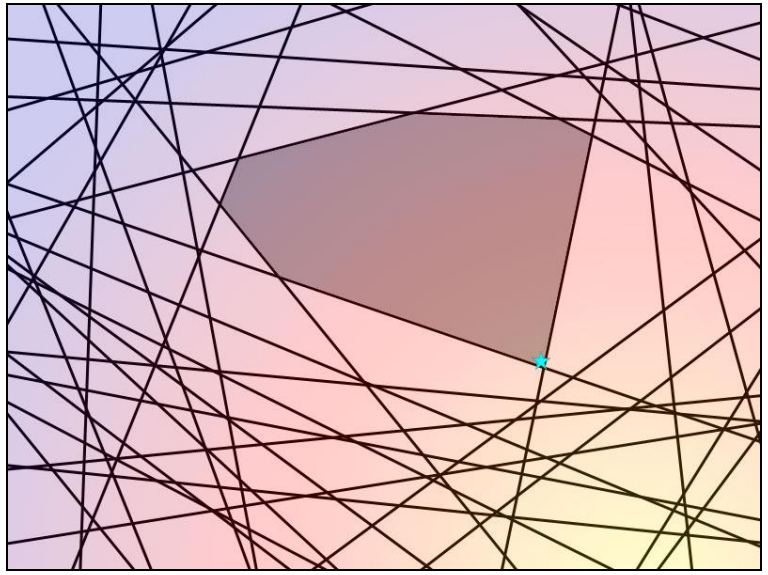
- Exponential constraints
- Most are dominated by a small set of “important” constraints



Structural SVM Approach

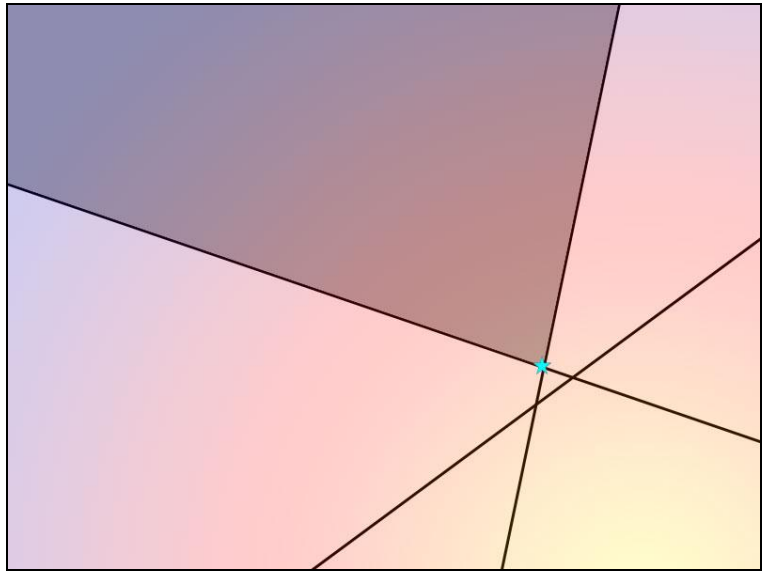
- Repeatedly finds the next most violated constraint…
- …until set of constraints is a good approximation.

Illustrative Example



Original SVM Problem

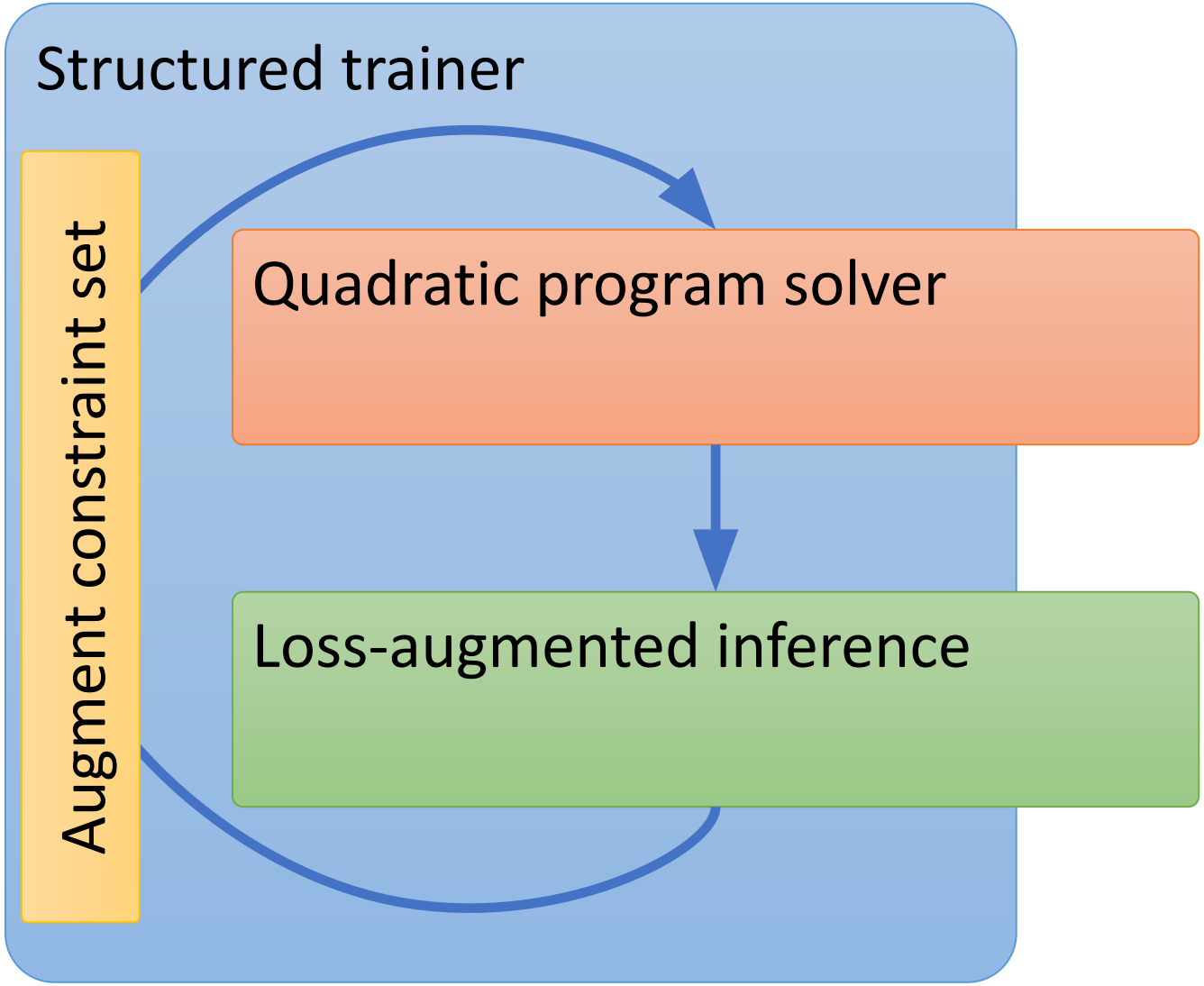
- Exponential constraints
- Most are dominated by a small set of “important” constraints



Structural SVM Approach

- Repeatedly finds the next most violated constraint…
- …until set of constraints is a good approximation.

“Structured learning API”



Steps:

1. From application needs, design Δ and Φ
2. Design loss-augmented inference procedure

Pairwise RankSVM

- We 'implement' RankSVM using "structured API"
- Query q, good docs $D_{q\oplus}$, bad docs $D_{q\ominus}$
- $$y_q[i,j] = \begin{cases} +1, & x_i \in D_{q\oplus},\ x_j \in D_{q\ominus} \\ -1, & x_i \in D_{q\ominus},\ x_j \in D_{q\oplus} \\ 0, & \text{otherwise} \end{cases}$$
- $y_q[i,j] = 0$ means inferred $y[i,j]$ can be any value without incurring a loss

Feature map $\Phi(X_q, y)$

- Usually designed keeping loss Δ in mind
- Suppose we define

Can ignore $[i, j]$ where $y_q[i, i] = 0$ (acts as a mask)

$$\Phi(X_q, y) = \frac{1}{2N_{q\oplus}N_{q\ominus}} \sum_{i,j:y_q[i,j]\neq 0} y[i, j](X_q[i] - X_q[j])$$

- If $X_q[\star] \in \mathbb{R}^m$ then $\Phi(X_q, y) \in \mathbb{R}^m$ as well, i.e, $M = m$
- Structured model w in "$w \cdot \Phi(X_q, y)$" is the same as λ in itemwise score "$\lambda \cdot \phi(q, X_q[\star])$"
- Note that

$$w \cdot \Phi(X_q, y) \propto \sum_{i,j:y_q[i,j]\neq 0} y[i, j](\lambda \cdot X_q[i] - \lambda \cdot X_q[j])$$

- (Not "good score" and "bad score" but just "two scores"
- To maximize lhs (inference), choose $y[i, j] = \text{sgn}(\lambda \cdot X_q[i] - \lambda \cdot X_q[j])$

ivalent to
nputing all
n scores $\lambda \cdot$
$[\star]$ and sorting
lecreasing
re order;
owledge of y_q
not actually
eded during
erence

RankSVM as structured loss

- Loss is

$$\Delta(y_q, y) = \frac{1}{2N_{q\oplus}N_{q\ominus}} \sum_{i,j:y_q[i,j]\neq 0} [\![y[i,j] \neq y_q[i,j]]\!]$$

- Can rewrite as

$$\Delta(y_q, y) = \frac{1}{2N_{q\oplus}N_{q\ominus}} \sum_{i,j:y_q[i,j]\neq 0} \frac{1 - y_q[i,j]\ y[i,j]}{2}$$

- $y_q[i,j] \neq 0$ holds for $2N_{q\oplus}N_{q\ominus}$ elements out of the N_q^2 elements in the matrix y_q
- Common to normalize losses so that regularization term $\|w\|_2^2$ is uniform across training batches of diverse sizes

Loss-augmented inference for RankSVM

- $\breve{y} = \underset{y}{\text{argmax}} \quad w \cdot \Phi(X_q, y) + \Delta(y_q, y)$
- Objective is

$$\underset{y}{\text{argmax}} \sum_{\substack{i,j: \\ y_q[i,j] \neq 0}} y[i,j](\lambda \cdot X_q[i] - \lambda \cdot X_q[j]) + \frac{1 - y_q[i,j]\ y[i,j]}{2}$$

- Same as

$$\underset{y}{\text{argmax}} \sum_{\substack{i,j: \\ y_q[i,j] \neq 0}} y[i,j](\lambda \cdot X_q[i] - \lambda \cdot X_q[j]) - \frac{y_q[i,j]\ y[i,j]}{2}$$

- Same as

$$\underset{y}{\text{argmax}} \sum_{\substack{i,j: \\ y_q[i,j] \neq 0}} y[i,j] \left(\lambda \cdot X_q[i] - \lambda \cdot X_q[j] - \frac{y_q[i,j]}{2} \right)$$

- Can set each $y[i,j]$ independent of others
- Still a difference of scores, minus a correction induced by y_q via loss Δ

Interpretation of $\breve{y}$

- Consider again

$$\operatorname*{argmax}_{y} \sum_{\substack{i,j: \\ y_q[i,j] \neq 0}} y[i,j] \left(\lambda \cdot X_q[i] - \lambda \cdot X_q[j] - \frac{y_q[i,j]}{2} \right)$$

- Suppose gold $y_q[i,j] = +1$ i.e., i better than j
- Unless i-score beats j-score by $\geq \frac{1}{2}$, we still set $\breve{y}[i,j] = -1$
- Introducing the inequality *component*

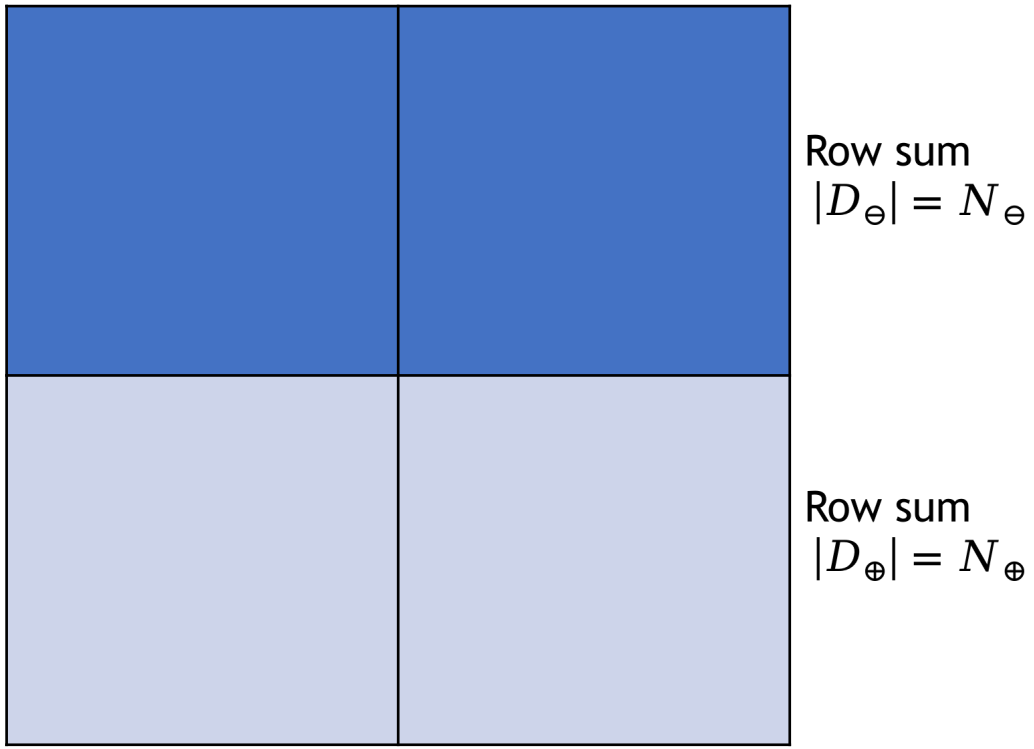
$$2(\lambda \cdot X_q[i] - \lambda \cdot X_q[j]) + \xi_q \geq [\![y_q[i,j] \neq \breve{y}[i,j]]\!]$$

- ...and *averaging* over all such i, j to get a single new linear constraint
- Note that $\breve{y}$ need not be consistent with a total order

Next: predicting a relevant *set*

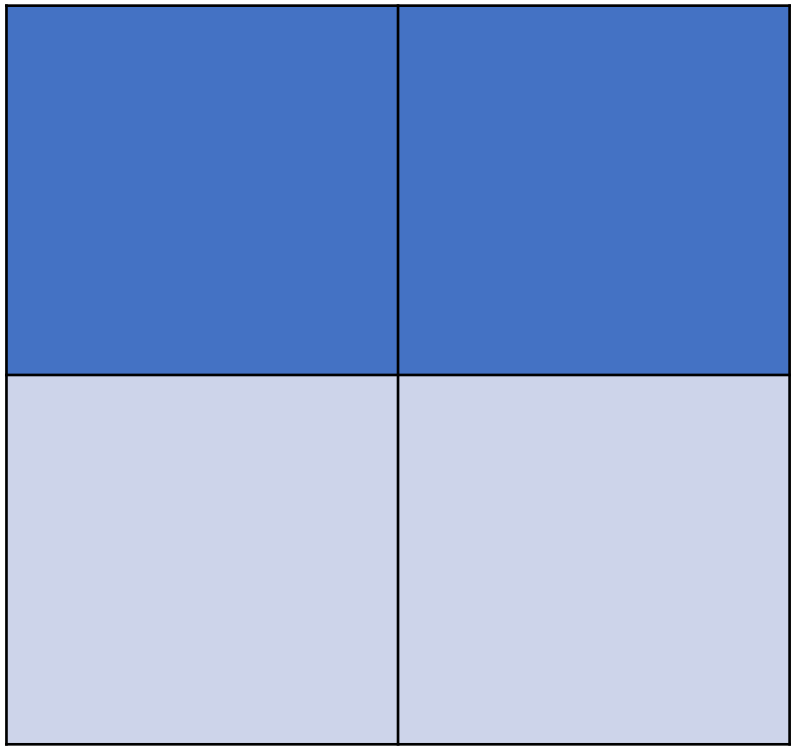
- Query q, gold set $D_{q\oplus}$, system prediction $\widehat{D_q}$
- How good is $\widehat{D_q}$ wrt $D_{q\oplus}$?
  - Use recall, precision, F1, …
  - Generally, any function of the contingency table
- Loss $\Delta(D_{\oplus}, \widehat{D})$
- More familiar notation $D_{q\oplus} \rightarrow y_q$, $\widehat{D_q} \rightarrow \widehat{y_q}$
- Where $y,\ y_q \in \{-1, +1\}^{N_q}$
- Itemwise decision, but collective loss
- $w \cdot \Phi(X_q, y) = \sum_{i \in [N_q]} y[i](w \cdot \phi(q, x_i))$
- Inference amounts to just

$$y[i] = \operatorname{sgn}(w \cdot \phi(q, x_i))$$



Contingency table → loss measure

- 0/1 error $\Delta_{0/1} = \frac{b+c}{a+b+c+d}$
- $\Delta_{\text{recall}} = 1 - R = 1 - \frac{d}{c+d}$
- $\Delta_{\text{precision}} = 1 - P = 1 - \frac{d}{b+d}$
- $\text{F1} = \frac{2PR}{P+R}$
- $\Delta_{\text{F1}} = 1 - \text{F1} = \frac{b+c}{2d+b+c}$



$a + b = N_{\ominus}$
$c + d = N_{\oplus}$

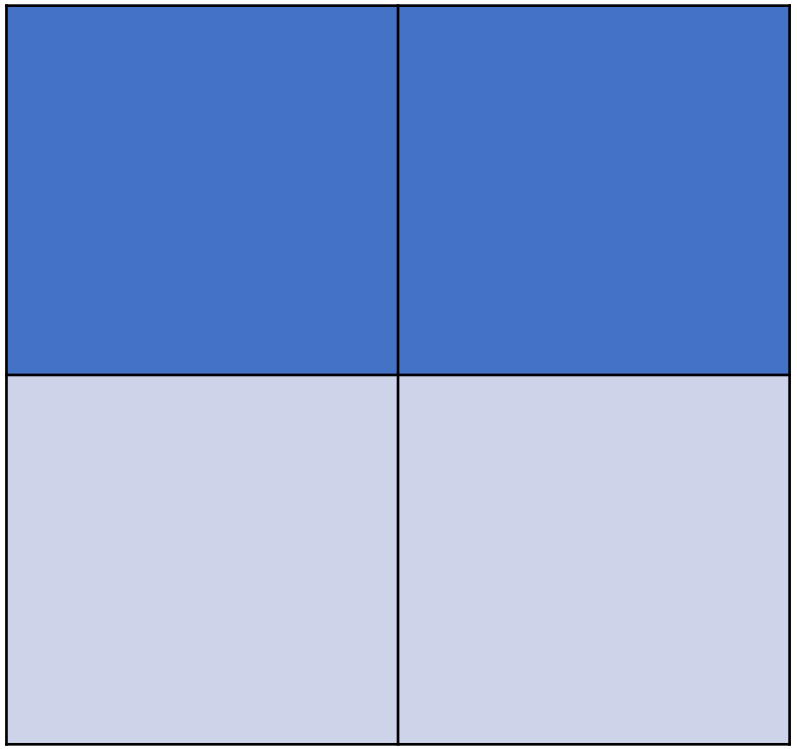
Balancing Δ against feature-based scores

- Recall $a + b = N_{\ominus}$ and $c + d = N_{\oplus}$
- For $a = 0, 1, \ldots, N_{\ominus}$
  - $b = N_{\ominus} - a$
  - For $c = 0, 1, \ldots, N_{\oplus}$
    - $d = N_{\oplus} - c$
      - Calculate $\Delta(y^*, y) = \Delta(a, b, c, d)$
      - Largest possible value of $w \cdot \Phi(X_q, y = \langle a, b, c, d \rangle)$
      - Remember largest yet value of $w \cdot \Phi(X_q, y) + \Delta(y_q, y)$ and corresponding y
- Check worst y yet and designate $\breve{y}$ if bad enough

Which items to assign $y = -1$ (thus, others to $y = +1$) so that contingency table $\langle a, b, c, d \rangle$ is satisfied, and $w \cdot \Phi(X_q, y)$ is maximized?

Satisfying $\langle a, b, c, d \rangle$

- We have $N_{q\ominus}$ items that we must split into $a + b$
- Each item has score $\lambda \cdot X_q[i]$
- a items will get $y[i] = -1$, remaining b will get $y[i] = +1$
- Recall collective score has terms like $y[i](\lambda \cdot X_q[i])$
- Assign b highest-scoring items label $+1$, remaining a items label -1
- Even if label $y[i] \neq \text{sgn}(w \cdot \phi(q, x_i))$
- Similarly divide up $D_{q\oplus}$ items into $c + d$ items



$$a + b = N_{\ominus}$$
$$c + d = N_{\oplus}$$

Predicting a list/ordering

- Consider MRR and MRR-loss $\Delta_{\mathrm{MRR}} = 1 - \mathrm{MRR}$
- Suppose there are R_q relevant docs
- If there are N_q items to rank, first relevant doc can be placed at positions $1, 2, \ldots, N_q - R$
- Accordingly, Δ_{MRR} can take discrete values $0, {}^{1}/_{2}, {}^{2}/_{3}, {}^{3}/_{4}, \ldots$ etc.
- Suppose first relevant doc is at rank p_1
- Given scores $\lambda \cdot \phi(q, X_q[i])$ for all $i \in [N_q]$, how to place the items to maximize (say)

$$\sum_{i,j:y_q[i,j]\neq 0} y[i,j](\lambda \cdot X_q[i] - \lambda \cdot X_q[j])$$

- Sort all items by decreasing score
- Place highest-scoring $p_1 - 1$ irrelevant items at ranks $1, 2, \ldots, p_1 - 1$
- Then place the highest-scoring relevant item at rank p_1
- Finally, place all remaining items in decreasing score order

Exercise: extend to Δ_{NDCG} and Δ_{MAP}

(Hint: sort $D_{q\oplus}$ and $D_{q\ominus}$ separately, then carefully choose a merge

Conclusion to first pass on L2R

- Itemwise, pairwise, listwise losses
- Non-linear decision function vs. accurate modeling of list-wise loss
- Convex upper bound vs. non-convex approximations

Adding proximity credit to vector space scoring

An exploration of proximity measures in information retrieval

A general scoring model for cooccurrence

A Markov Random Field Model for Term Dependencies