Maximum Margin Matrix Factorization for Collaborative Ranking

Joint work with Quoc Le, Alexandros Karatzoglou and Markus Weimer

Alexander J. Smola

sml.nicta.com.au
Statistical Machine Learning Program
Canberra, ACT 0200 Australia
Alex.Smola@nicta.com.au

October 6, 2007, DMSS

Outline

Collaborative Ranking

Setting

- Internet retailer (e.g. Netflix) sells movies M to users U.
- Users rate movies if they liked them.
- Retailer wants to suggest some more movies which might be interesting for users.

Goal

Suggest movies that user will *like*. Pointless to recommend movies that users do not like since they are unlikely to rent.

Problems with Netflix contest

- Error criterion is uniform over all movies.
- Can only recommend a small number of movies at a time (probably no more than 10).
- Need to do well only on top scoring movies.

More Applications

Retail

eTailer (e.g. Amazon) wants to suggest new books and other products based on past purchase decisions and reviews.

Collaborative photo viewing site

Want to suggest some more photos user might want to see given past viewing behavior (e.g. Flickr.com).

Collaborative bookmark site

Suggest new bookmarks based on which ones users clicked at before. Do this in a personalized fashion. This immediately avoids click spam: only spammer is affected by spam clicks: each user gets his personalized view (e.g. Digg.com).

News site

Suggest new stories personalized to click behavior, such as news.{google, yahoo}.com.

Even More Applications

Blogs and Internet Discussions

Show mainly stories that the Specific User likes. E.g. a customized version of Slashdot or Groklaw.

Internet Dating

Suggest new dates based on previous viewing behavior, number of contact attempts, collaboratively.

Advertising

Use more fine-grained information on link following behavior, e.g. a) customer visits site, b) puts things into shopping basket, c) registers for an account, d) purchases item. Use the entire range of decisions to predict better.

Requirements

Direct Optimization

Want to optimize scoring function directly, or at least convex (or at least continuous) uppoer bound of this.

Featureless Estimation

Algorithm should not need inherent features of objects.

Incorporating Features

If features exist, algorithm should be able to include them.

Scalability

Implementation needs to scale to hundreds of millions of records. Parallel and multi-core implementation needed. Cheap deployment phase.

The Ingredients

Convex Upper Bound

Bound ranking loss, such as NDCG@k directly via convex ranking bound (e.g. Chapelle, Le and Smola, 2007).

Low Rank Matrix Factorization

Use tools of Srebro et al. and factorize the scoring matrix into a product of user and movie matrix: $F = MU$.

Features

Use movie specific, user specific, and (movie, user) specific information to add to factorization

$$F = MU + \underbrace{f_m w_m + w_u f_u + f_{mu} \cdot w_{mu}}_{\text{optional}}.$$

Bundle Method Solver

Scalable convex optimization for movie and user phase. Parallelize over users.

Outline

Ranking Movies for a User

Data

- (movie,user) pairs (i, j)
- ratings $Y_{ij} \in \{1, \ldots, 5\}$ of movie i by user j

Goal

- For a given user j rank unseen movies i such that the movies he likes most are suggested first.

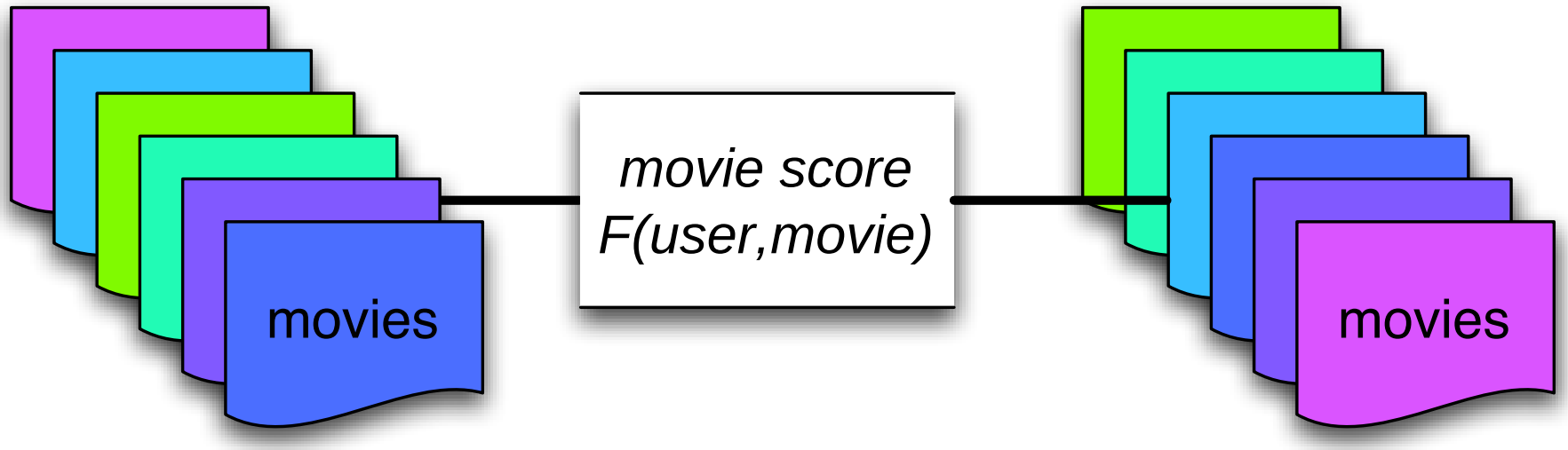
Modified goal

- Rank by assigning scores F_{ij} to (movie,user) pairs.
- This is fast at training time since it decouples the movies (only quicksort needed).

Ranking loss

- Multivariate performance score (couples the F_{ij} scores)
- We use Normalized Discounted Cumulative Gains truncated at 10 retrieved movies (NDCG@10).

Scoring Function



Goal is to find a scoring function F_{ij} which optimizes a user-defined ranking and performance score.

Ranking Scores

Normalized Discounted Cumulative Gain

$$\mathrm{DCG@}k(\pi, y) = \sum_{i=1}^{k} \frac{(2^{y_i} - 1)}{\log(\pi_i + 1)} \text{ and}$$

$$\mathrm{NDCG@}k(\pi, y) = \frac{\mathrm{DCG@}k(\pi, y)}{\mathrm{DCG@}k(\mathrm{argsort}(y), y)}.$$

π is permutation and y are user ratings. $\mathrm{Score} = a(\pi)^\top b(y)$

Extensions

- Alternatives are unnormalized, not truncated, different decay, different position weighting, ...
- We can take position specific ranking into account. E.g. last position in the list should be a romantic comedy, etc. But more expensive at prediction time!

A Convex Upper Bound

Problem

Finding F such that DCG@k is maximized is highly nonconvex (it is piecewise constant).

Solution

Structured estimation yields convex upper bound. We use

$$G(\pi, f) := \sum_j c_{\pi_j} f_j \text{ which is maximized for } \pi^* = \operatorname{argsort} f.$$

Here c_j is a monotonically decreasing function in j.

Theorem

The loss $\Delta(y, \pi^*)$ for choosing π^* is bounded by ξ where

$$G(\operatorname{argsort} y, f) - G(\pi, f) \geq \Delta(y, \pi) - \xi \text{ for all } \pi.$$

Putting it together

Per-user loss

$$l(y, f) = \max_{\pi} \Delta(y, \pi) + G(\pi, f) - G(\operatorname{argsort} y, f).$$

Maximization is carried out by solving a Linear Assignment Problem: the linear programming relaxation of the integer programming problem is totally unimodular. This allows us to compute *gradients* and *values* efficiently.
Use Joncker and Volgenant or Orlin and Yee algorithm.

Cumulative upper bound

$$L(Y, F) := \sum_{i \in \text{Users}} l(Y_{i\cdot}, F_{i\cdot})$$

We only sum over a subset of movies per user.

Outline

Low Rank Factorization

Estimating F

Srebro and coworkers define regularizer for F

$$\|F\| := \inf_{MU=F} \frac{1}{2}\left[\|M\|^2 + \|U\|^2\right]$$

Semidefinite Convex Problem

Replace with semidefinite construction for matrix via

$$\begin{bmatrix} Z_M & F^\top \\ F & Z_U \end{bmatrix} \succeq 0 \text{ and } \|F\| \to \frac{1}{2}\left[\operatorname{tr} Z_M + \operatorname{tr} Z_U\right].$$

Problem

The optimization problem is huge. Even storing full F is impossible (10^{10} entries).

Solution

Factorize $F = MU$ or low rank M and U (10-200 dimensions).

Optimization Problem

Objective Function

$$L(Y, MU) + \frac{\lambda}{2}\left[\|U\|^2 + \|M\|^2\right]$$

Algorithm

Objective function is convex in U for fixed M and vice versa.

1. Minimize $L(Y, MU) + \frac{\lambda}{2}\|M\|^2$
2. Minimize $L(Y, MU) + \frac{\lambda}{2}\|U\|^2$

Repeat until converged.

Extensions

- Add user, movie specific features
- Different regularization for different movie, user numbers
- Different regularization for each dimension

Features

Improved low rank factorization

Use movie specific, user specific, and (movie, user) specific information to add to factorization

$$F = MU + \underbrace{f_m w_m + w_u f_u + f_{mu} \cdot w_{mu}}_{\text{optional}}.$$

Optimization

- Optimize over all parameters except for M or U.
- Problem is still convex.

Domain knowledge

The right place for feature engineering. In particular f_{mu} contains (movie, user) features.

- Christmas movies are not popular in August.
- Die Hard will not sell well on Good Friday.
- Soccer movies are popular in worldcup years . . .

Outline

Basic Constraints

Empirical Risk

- Evaluating it is expensive (linear assignment is killer).
- Almost equally expensive to compute value or value and gradient.

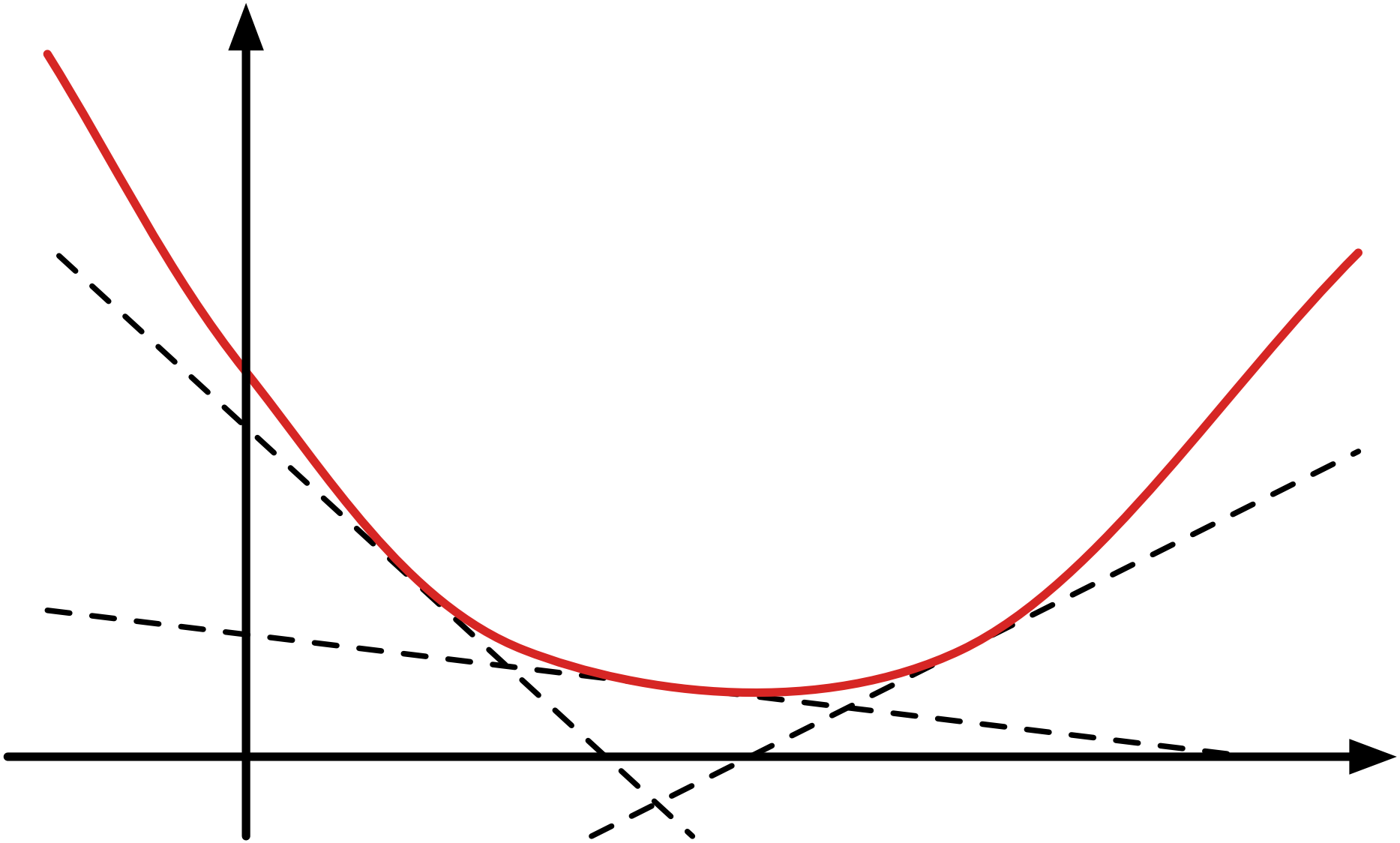
Regularizer

- Cheap to compute
- Easy to minimize over regularizer plus piecewise linear function

Idea

- Use past gradients to build up successively improving lower bound on empirical risk.
- Solve regularized lower bound problem successively.

Bundle Approximation



Algorithm

Lower bound

$$R_t[w] := \max_{j \le t} \langle a_t, w \rangle + b_t \le R_{\text{emp}}[w]$$

where $a_t = \partial_w R_{\text{emp}}[w_{t-1}]$ and $b_t = R_{\text{emp}}[w_{t-1}] - \langle a_t, w_{t-1} \rangle$.

Pseudocode

Initialize $t = 0, w_0 = 0, a_0 = 0, b_0 = 0$

repeat

Find minimizer $w_t := \text{argmin}_w R_t(w) + \frac{\lambda}{2} \|w\|^2$

Compute gradient a_{t+1} and offset b_{t+1}.

Increment $t \leftarrow t + 1$.

until $\epsilon_t \le \epsilon$

Upper Bound

Note that $R_{t+1}[w_t] = R_{\text{emp}}[w_t]$. Hence $R_{t+1}[w_t] - R_t[w_t]$ upper bounds gap size. We get a cheap convergence monitor.

Dual Problem

Good News

Dual optimization problem is Quadratic Program regardless of the choice of the empirical risk.

Details

$$\underset{\beta}{\text{minimize}} \ \frac{1}{2\lambda}\beta^\top AA^\top\beta - \beta^\top b$$

$$\text{subject to } \beta_i \geq 0 \text{ and } \sum_i \beta_i = 1$$

The primal coefficient w is given by $w = -\lambda^{-1}A^\top\beta$.

Very Cheap Variant

Can even use simple line search for update (almost as good).

Convergence

Theorem

The number of iterations to reach ϵ precision is bounded by

$$n \leq \log_2 \frac{\lambda R_{\text{emp}}[0]}{G^2} + \frac{8G^2}{\lambda\epsilon} - 4$$

steps. If the Hessian of $R_{\text{emp}}[w]$ is bounded, convergence to any $\epsilon \leq H/2$ takes at most the following number of steps:

$$n \leq \log_2 \frac{\lambda R_{\text{emp}}[0]}{4G^2} + \frac{4}{\lambda}\max\left[0, 1 - 8G^2H^*/\lambda\right] - \frac{4H^*}{\lambda}\log 2\epsilon$$

Advantages

- Linear convergence for smooth loss
- For non-smooth loss almost as good in practice (as long as smooth on a course scale).
- General solver (works for any loss)

Outline

Datasets

Data set sizes

Dataset	Users	Movies	Ratings
EachMovie	61265	1623	2811717
MovieLens	983	1682	100000
Netflix	480189	17770	100480507

Very sparse matrix. Ratings between 1 and 5.

Comparators

Regression

- Plain regression on the labels
- Very easy to implement, only solve least mean squares problems iteratively.
- This is what the Netflix contest wants.

Ordinal Regression

- Retain absolute ordering between users
- Relax the actual scores (retain only margin)
- Suggested by Herbrich and Gräpel, 2000

Protocol

- Same protocol for all solvers (same function space)
- Weak generalization: new movies for the same user
- Strong generalization: new movies for a new user (do not optimize for the user we want to test on).

Weak Generalization (NDCG@10)

	Method	N=10	N=20
EachMovie	NDCG	**0.6673 ± 0.0015**	**0.7589 ± 0.0006**
	Ordinal	0.5592 ± 0.0040	0.6619 ± 0.0062
	Regr.	0.5349 ± 0.0214	0.6291 ± 0.0161
		p = 2.7e-14	**p = 3.0e-12**
MovieLens	NDCG	0.6364 ± 0.0064	**0.7153 ± 0.0038**
	Ordinal	0.6291 ± 0.0004	0.6601 ± 0.0013
	Regression	**0.6404 ± 0.0057**	0.7015 ± 0.0056
	MMMF	0.6061 ± 0.0037	0.6937 ± 0.0039
		p = 0.011	**p = 6e-5**
Netflix	NDCG	0.6081	0.6204
	Regression	0.6082	0.6287

Strong Generalization (NDCG@10)

	Method	N=10	N=20
EachMovie	NDCG	$\mathbf{0.6367 \pm 0.001}$	$\mathbf{0.6619 \pm 0.0022}$
	GPR	0.4558 ± 0.015	0.4849 ± 0.0066
	CGPR	0.5734 ± 0.014	0.5989 ± 0.0118
	GPOR	0.3692 ± 0.002	0.3678 ± 0.0030
	CGPOR	0.3789 ± 0.011	0.3781 ± 0.0056
	MMMF	0.4746 ± 0.034	0.4786 ± 0.0139
MovieLens	NDCG	$\mathbf{0.6237 \pm 0.0241}$	$\mathbf{0.6711 \pm 0.0065}$
	GPR	0.4937 ± 0.0108	0.5020 ± 0.0089
	CGPR	0.5101 ± 0.0081	0.5249 ± 0.0073
	GPOR	0.4988 ± 0.0035	0.5004 ± 0.0046
	CGPOR	0.5053 ± 0.0047	0.5089 ± 0.0044
	MMMF	0.5521 ± 0.0183	0.6133 ± 0.0180

Discussion

Open Question

- Why is strong generalization with our solver so much better? Not much difference for weak generalization.

Parallelization

- User optimization easy: embarassingly parallel.
- Movie optimization almost as easy: precompute gradient of loss in parallel.
- MMMF is very slow (up to 1 day) vs. 20 minutes for our implementation.

Improvements

- Adaptive regularization for dimensions, sample sizes (movies, users)
- Better initialization
- Better sparse matrix library
- Use QuickMatch (faster than current implementation)

Summary

1. Collaborative Ranking
2. A Convex Upper Bound for Ranking
3. Low Rank Factorization
4. Convex Optimization via Bundle Methods
5. Experiments