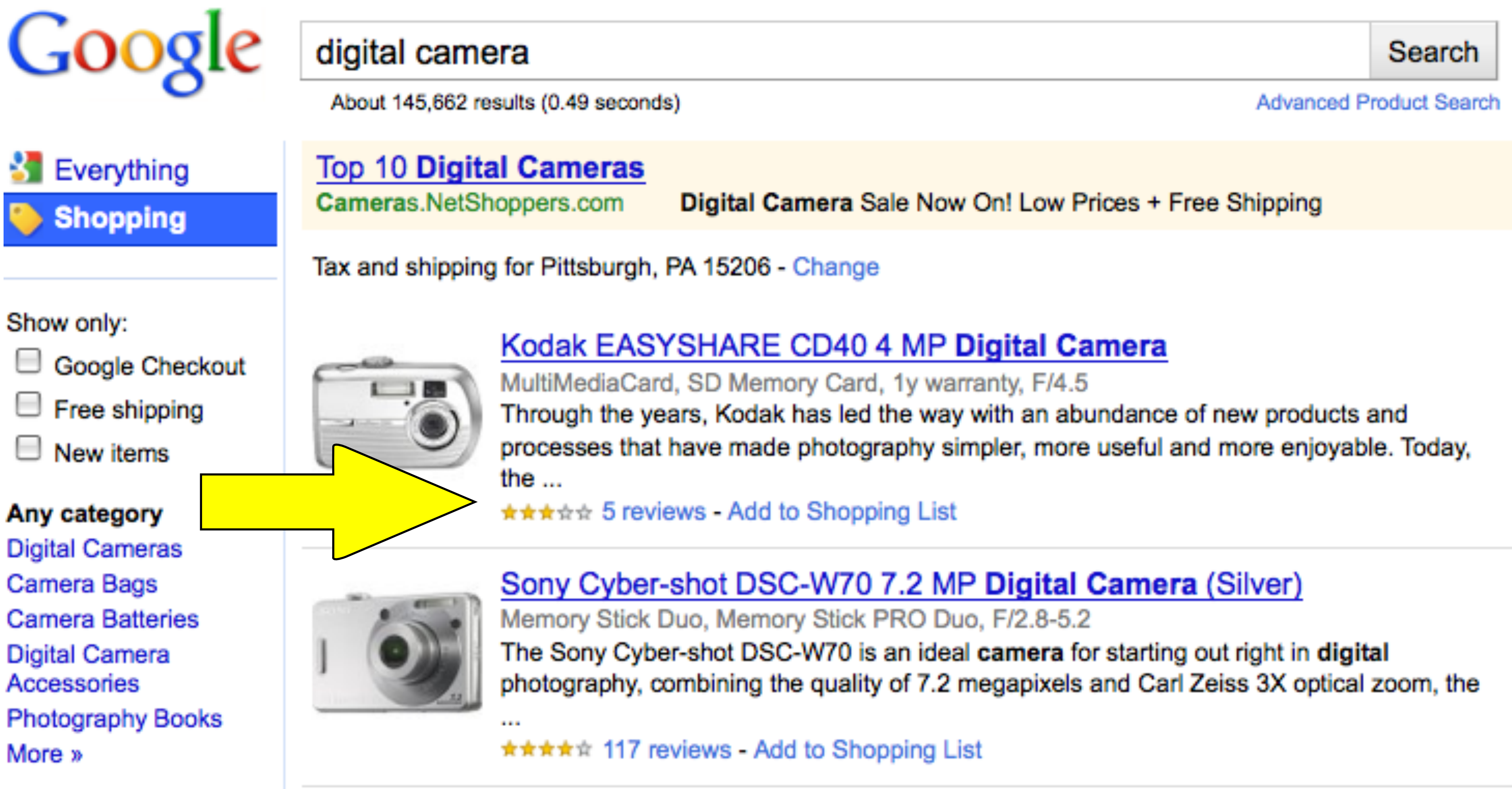
Combined Regression and Ranking

D. Sculley
Google Pittsburgh

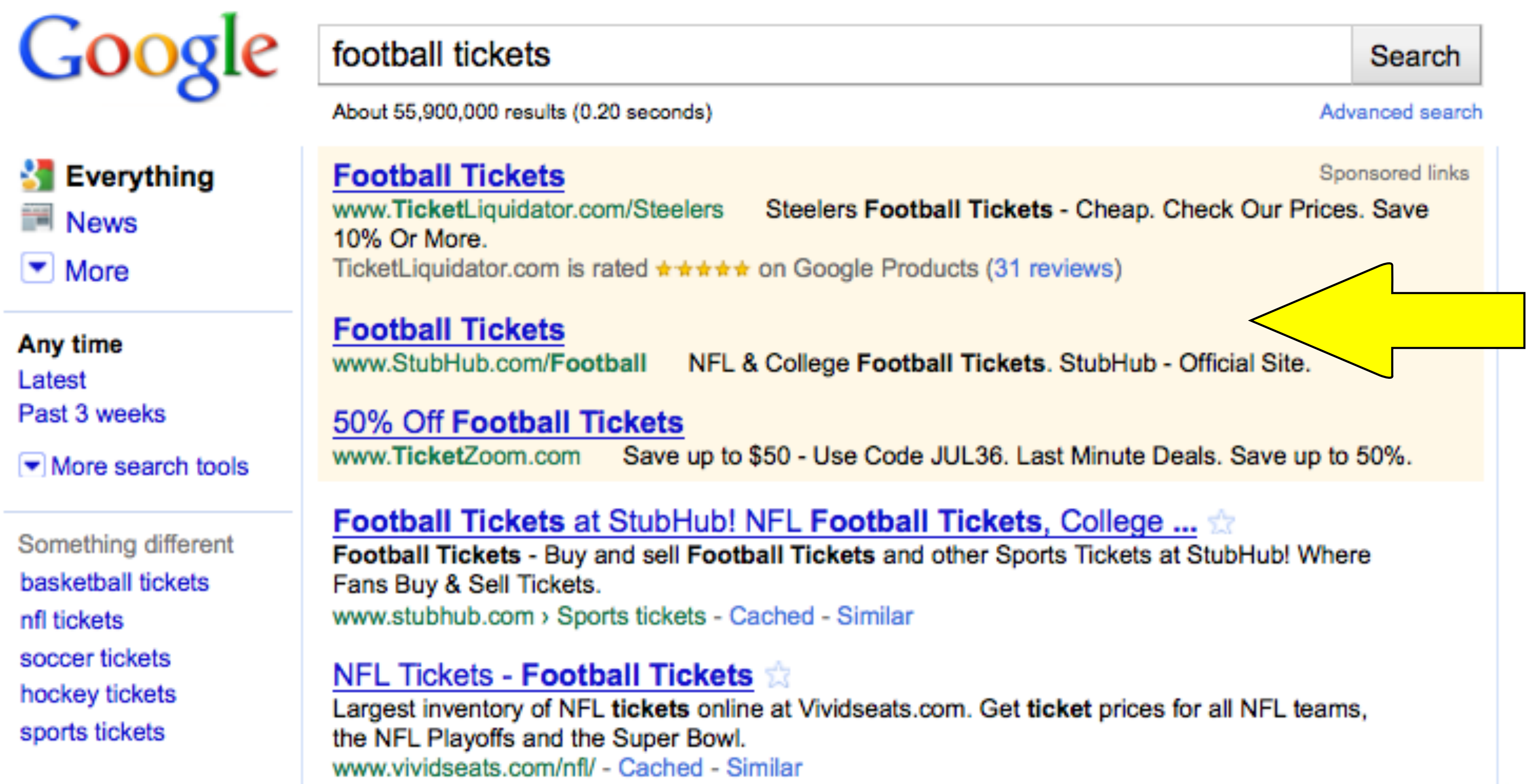
The Claim

- Many applications require models that give **both**:
  - good **regression** performance and
  - good **ranking** performance

Example: Predicting Star Ratings



Example: Click Prediction



- Why not just use existing methods?

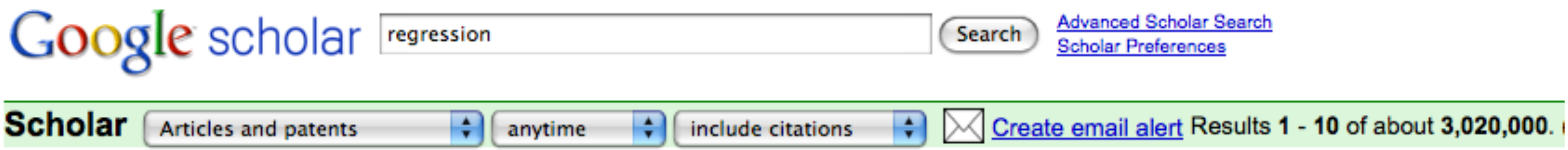
Standard Methods Can Fail Badly

- Rank-based models may do arbitrarily badly at regression
- Perfect regression implies perfect ranking, but...
- Even "good" regression can have bad ranking performance

Our Approach

- Novelty: optimize ranking and regression simultaneously
  - primary goal: try and get "best of both" performance
    - do as well at ranking as a ranking-only method
    - do as well at regression as a regression-only method
  - secondary goal: improved regression through ranking?
- We'll build this up in pieces

Supervised Regression (birds eye view)



- Goal: learn a model ***w*** that predicts a *real valued* target ***y***

- Examples:
  - Least mean squares
  - Ridge Regression
  - LASSO

- Often solved using empirical risk minimization

Supervised Regression (review)

$$\min_{\mathbf{w} \in \mathbb{R}^m} L(\mathbf{w}, D) + \frac{\lambda}{2} ||\mathbf{w}||_2^2$$

Supervised Regression (review)

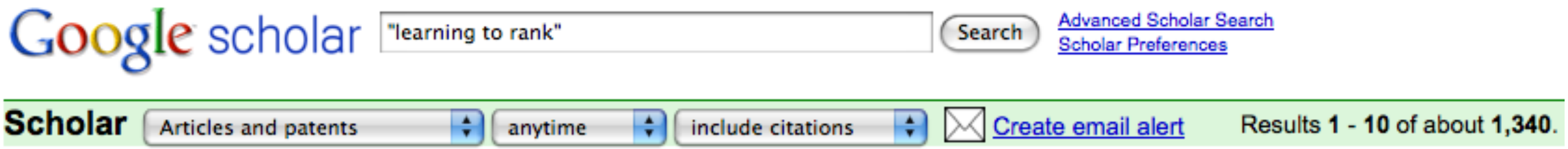
Loss Function

$$\min_{\mathbf{w} \in \mathbb{R}^m} L(\mathbf{w}, D) + \frac{\lambda}{2} ||\mathbf{w}||_2^2$$

Examples: Squared Loss, Logistic Loss, etc.

Regularization Term

Supervised Ranking (review)



- Goal: learn a model ***w*** that puts unseen data in the correct preference order

- Several known methods:
  - RankSVM (Joachims, 2002)
  - Voted Perceptron variant (Elsas et al., 2008)
  - Boosting variants: AdaRank-MAP, AdaRank-NDCG (Xu and Li, 2007)
  - Listwise approach (Cao et al., 2007)

Supervised Ranking (review)

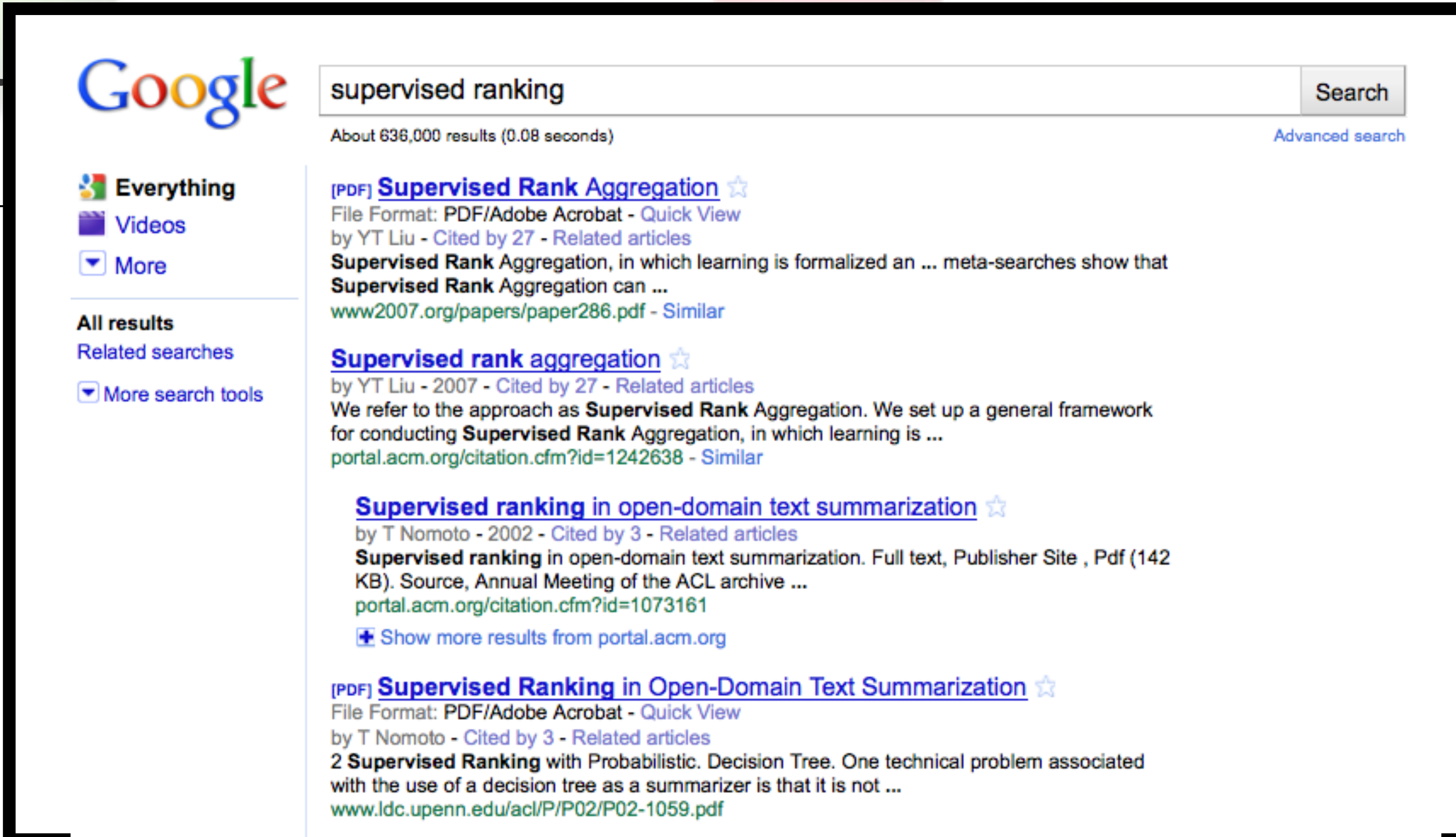
$$\min_{\mathbf{w} \in \mathbb{R}^m} L(\mathbf{w}, P) + \frac{\lambda}{2} ||\mathbf{w}||_2^2$$

Supervised Ranking (review)

$$\min_{\mathbf{w}\in\mathbb{R}^m} L(\mathbf{w}, P) + \frac{\lambda}{2}||\mathbf{w}||_2^2$$

Candidate Pairs:
pairs (a,b) of comparable examples with different ranks

Su



Candidate Pairs: pairs (a,b) of comparable examples with different ranks

Supervised Ranking (review)

$$\min_{\mathbf{w} \in \mathbb{R}^m} L(\mathbf{w}, P) + \frac{\lambda}{2} ||\mathbf{w}||_2^2$$

Warning: P is quadratic in |D|

- Joint optimization...

Combined Ranking and Regression

$$\min_{\mathbf{w} \in \mathbb{R}^m} \alpha L(\mathbf{w}, D) + (1 - \alpha) L(\mathbf{w}, P) + \frac{\lambda}{2} ||\mathbf{w}||_2^2$$

Combined Ranking and Regression

Regression Term

Ranking Term

$$\min_{\mathbf{w} \in \mathbb{R}^m} \alpha L(\mathbf{w}, D) + (1 - \alpha) L(\mathbf{w}, P) + \frac{\lambda}{2} ||\mathbf{w}||_2^2$$

Tradeoff Parameter

Regularization Term

Google™

Combined Ranking and Regression

$$\min_{\mathbf{w}\in\mathbb{R}^m} \alpha L(\mathbf{w}, D) + (1-\alpha) L(\mathbf{w}, P) + \frac{\lambda}{2}||\mathbf{w}||_2^2$$

Convexity Maintained

- What about dealing with size of P? This is quadratic in |D|.

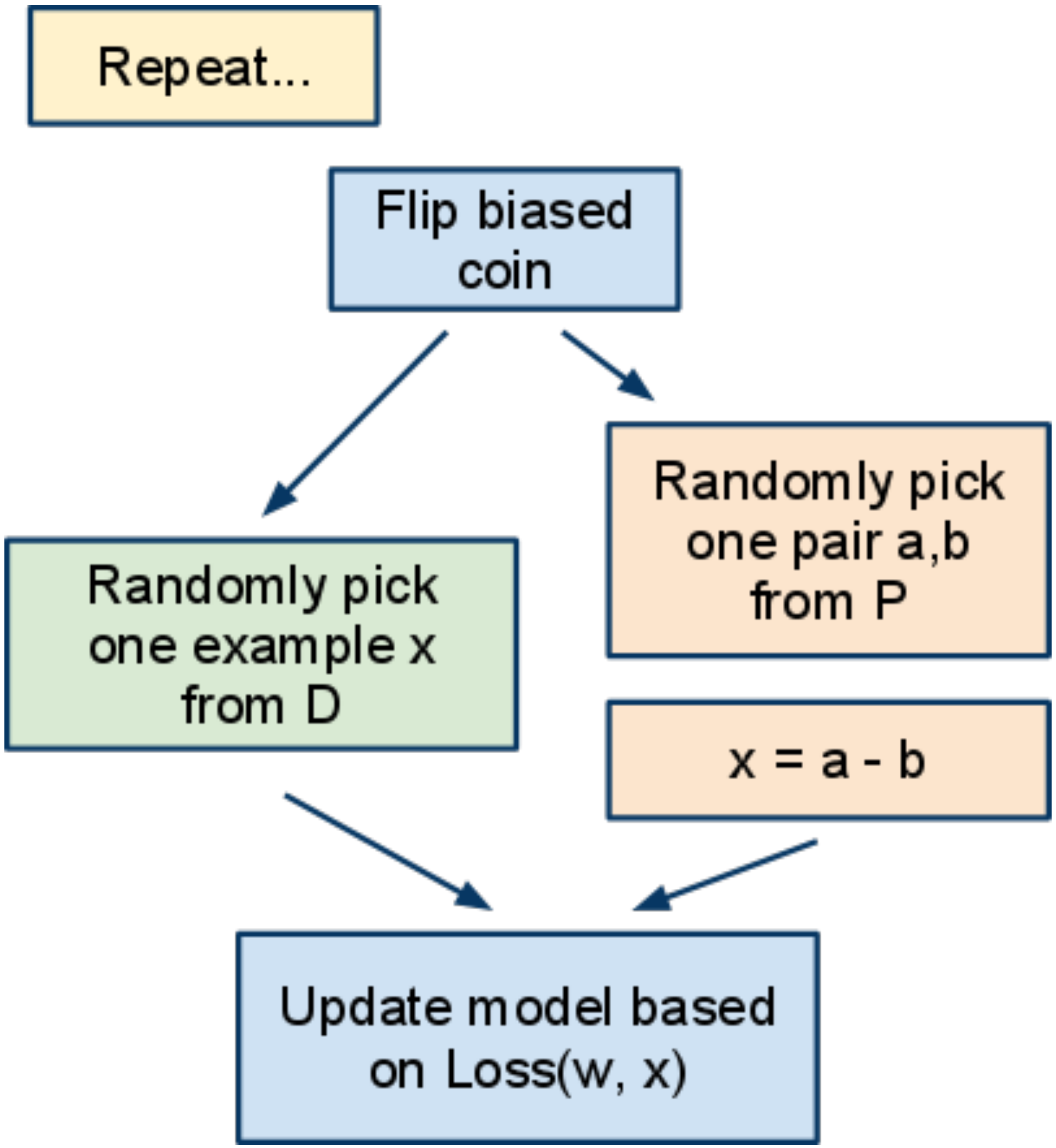
Efficient Sampling from *P*

- We don't want to look at O(n^2) training pairs
- How to sample pairs from *P*?

- Fastest solution is to index the training data:
  - O(log|Q| + log|Y|) in general
  - O(1) for common scenarios

- When data is too large to index, can use rejection sampling

Solving CRR Efficiently



Scalability

- Like other stochastic gradient descent algorithms, CRR is fast for large data

- RCV1 experiments
    - 780,000 training examples
    - Less than 3 CPU sec's on normal laptop

Non-linear Models

- CRR optimization problem is defined using a linear model w
- If we want non-linearity, use a trick from Balcan and Blum:
    - Pick a set of *k* reference examples ***r**_1*, ..., ***r**_k*
    - Map each example ***x*** into a new feature space of dimension *k*
    - Value for feature *i* in new space is *kernel*(***x***, ***r***_i)
- Still efficient

Experimental Overview

- Data sets:
  - RCV1 text classification
  - LETOR learning to rank benchmark data
  - Click prediction data for sponsored search (private)

- Comparison methods:
  - Regression-only, Ranking-only
  - Parameters tuned with cross validation on training data or on separate validation data

- Evaluation metrics:
  - Mean Squared Error (MSE)
  - AUC Loss (1 - Area Under ROC Curve)
  - Normalized Discounted Cumulative Gain (NDCG)
  - Mean Average Precision (MAP)

RCV1 Setup

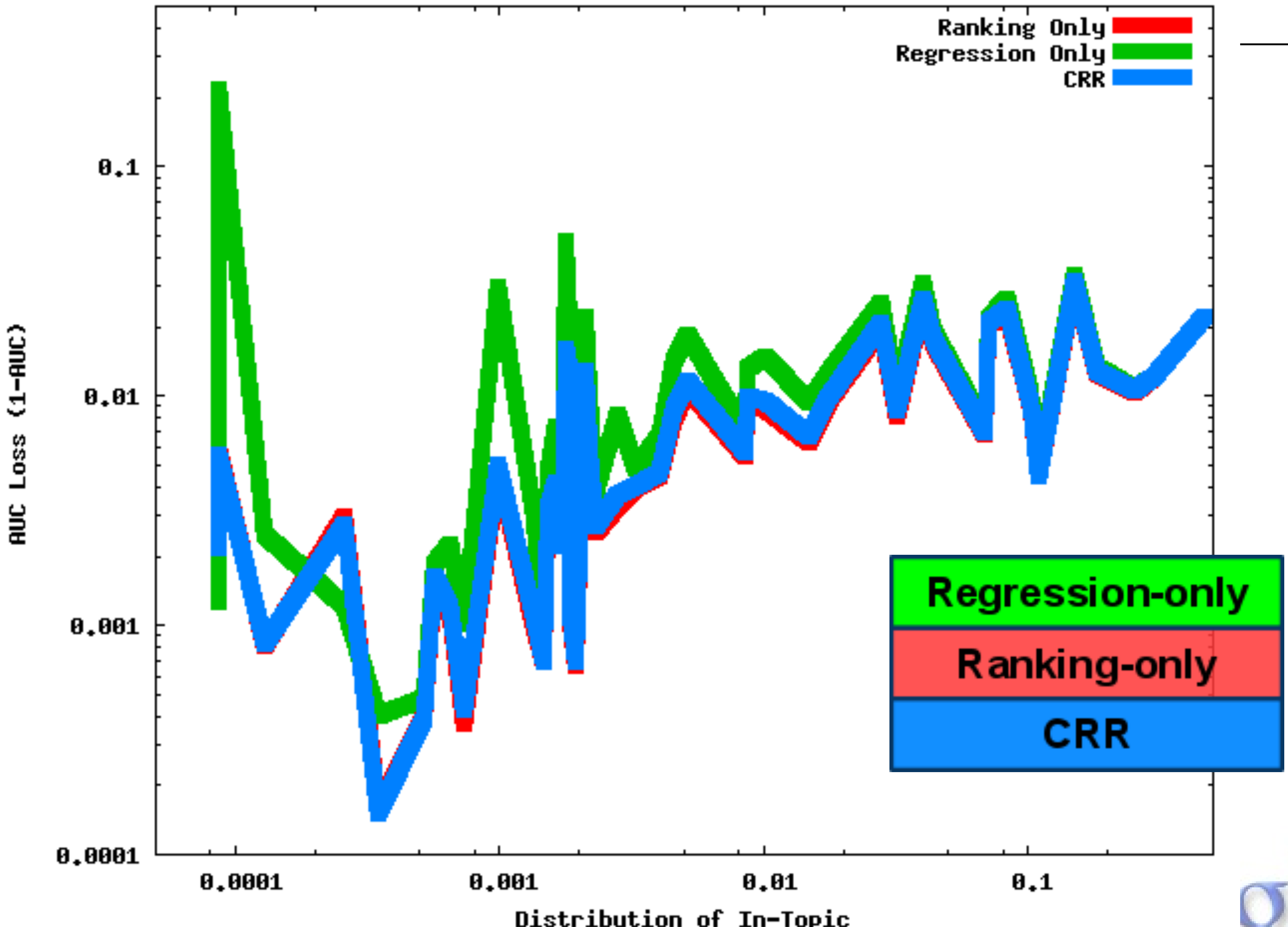
- Benchmark text mining data set
- Tested 40 per-topic tasks

- ~780k training examples
- ~23k test examples
- ~50k sparse features

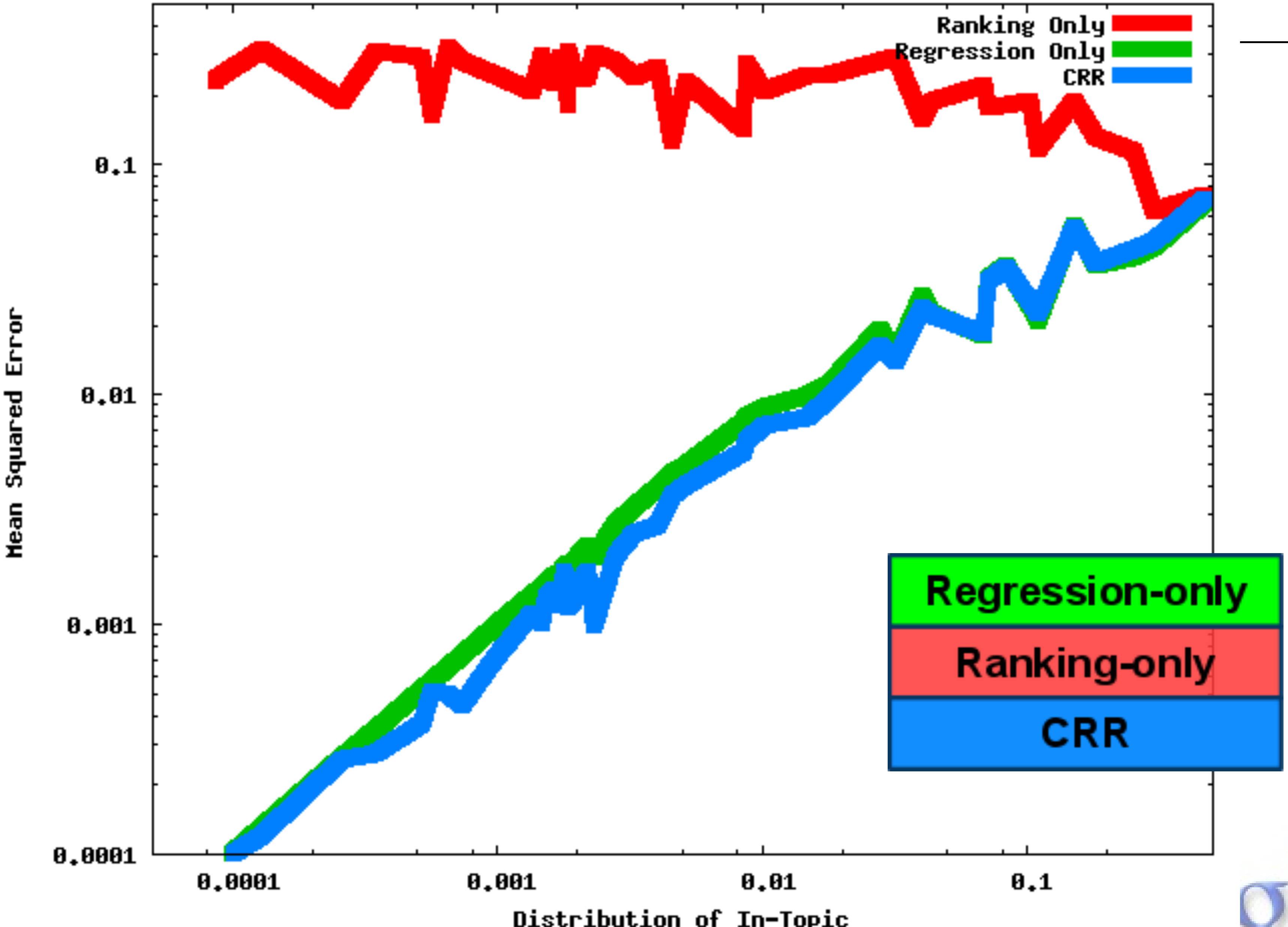
- Some topics contain extreme minority class distributions, with only 0.02% "positive"

- Used logistic loss on {0, 1} targets

RCV1 Ranking Results



RCV1 Regression Results



RCV1 Results

- CRR achieves "best of both" metrics on 16 out of 40 tasks
    - Within 0.001 of best on 19 additional tasks
    - Always gives best performance on at least one of the two metrics
- Adding rank-based constraints can help regression:
    - CRR out-performs regression-only on MSE on 20 of 33 extreme minority class topics
    - gives equal performance on remainder

Why Would Ranking Help Regression?

- Rank-based constraints are informative, especially when observations are rare

- Imagine you had two biased coins
  - A comes up heads with probability 0.02
  - B comes up heads with probability 0.03

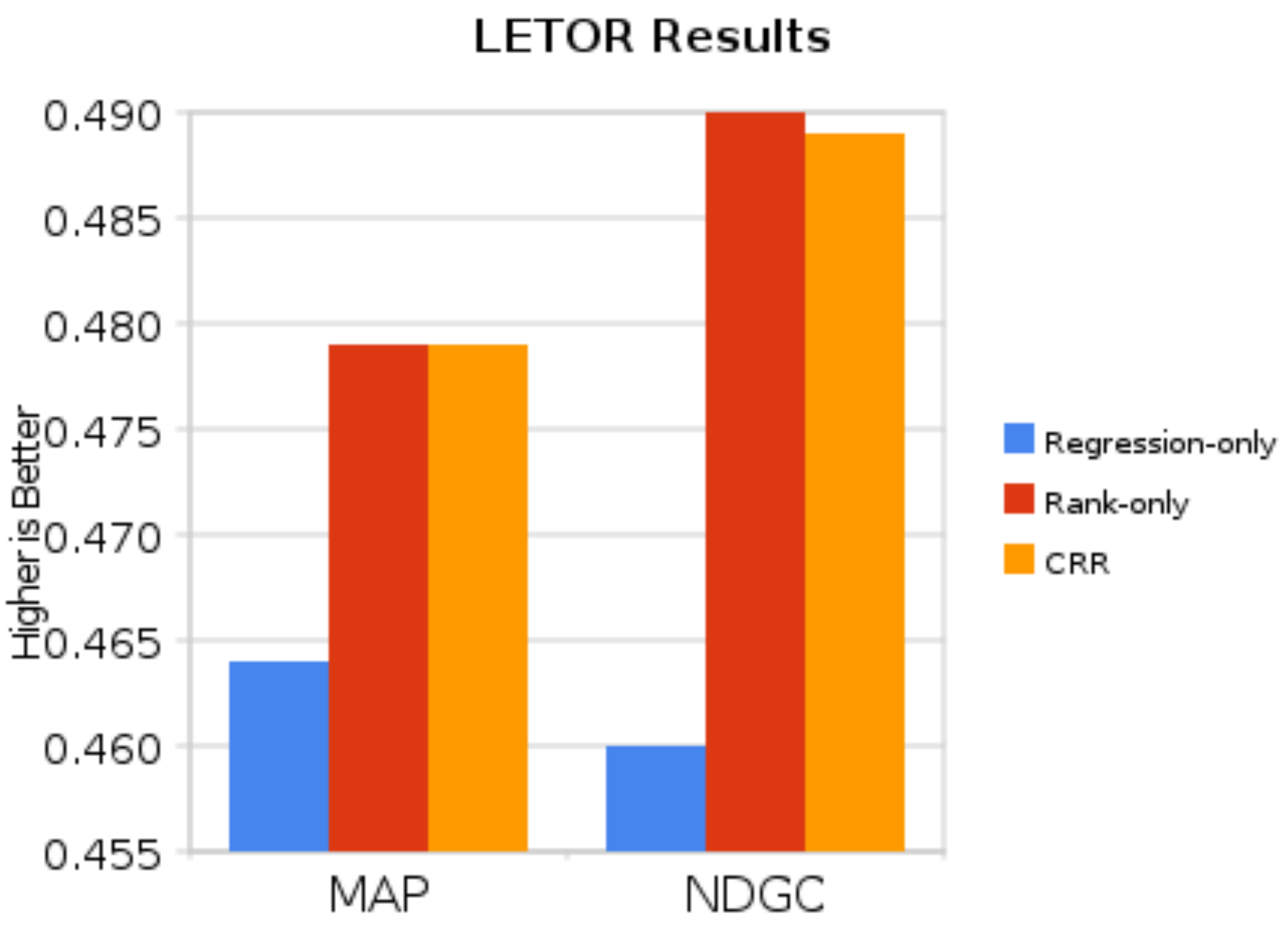
- Knowing that coin C is between A and B is extremely helpful if we don't have much other data

Google™

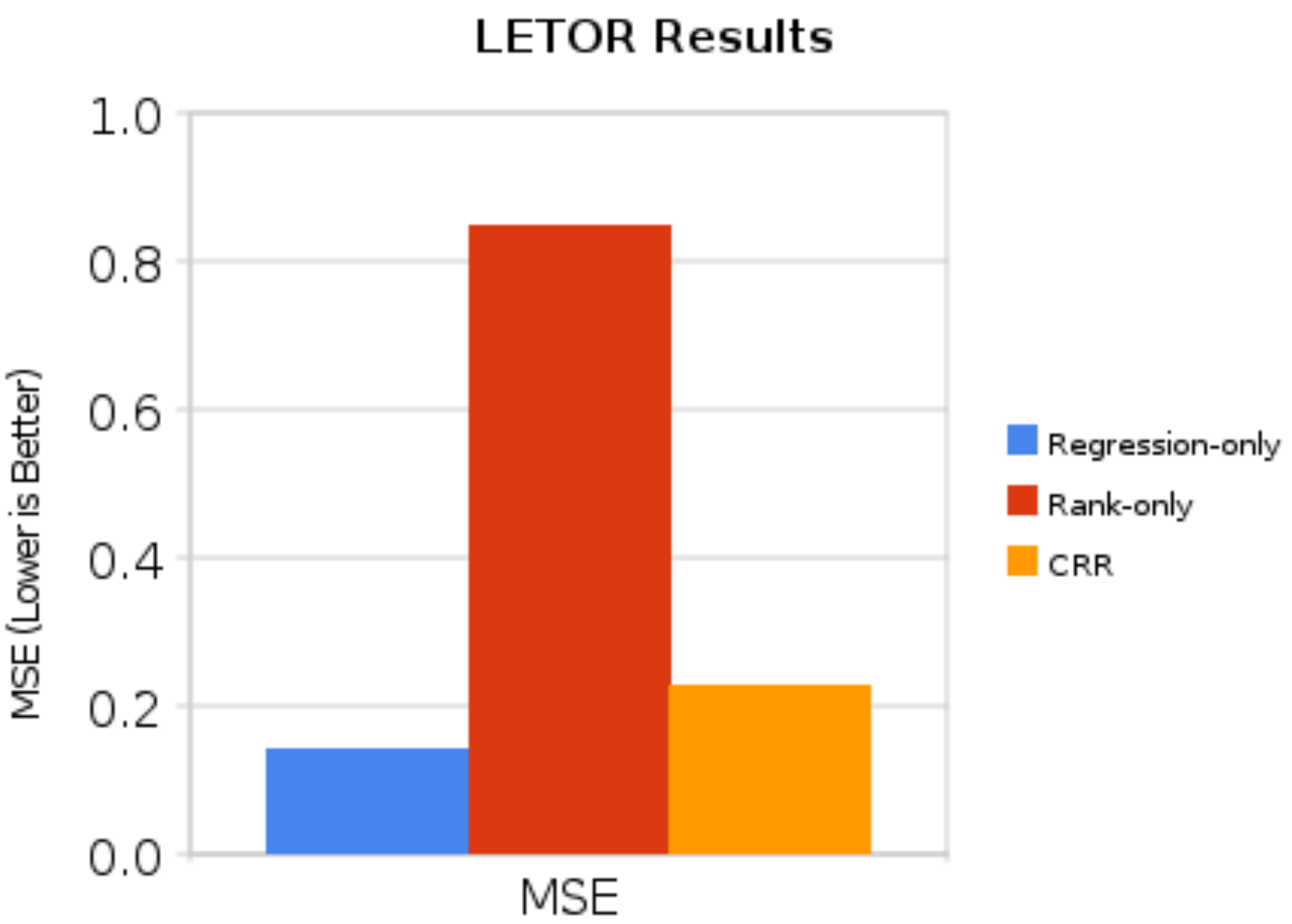
LETOR Experiments

- LETOR: benchmark learning to rank data
- Tasks with multiple relevance levels: 1, 2, or 3 stars
- Used squared loss; regression predicts ordinal values

LETOR Ranking Results



LETOR Regression Results



Click Prediction Experiments

- Test data set of several million ads
- Labels of "clicked" and "not clicked"
- Very high dimensional feature space
- Logistic loss used

Click Prediction Results

Method	Mean Sq. Error	AUC Loss
Ranking-only	0.0935	0.1325
Regression-only	0.0840	0.1334
CRR	0.0840	0.1325

Click Prediction Results

Method	Mean Sq. Error	AUC Loss
Ranking-only	0.0935	0.1325
Regression-only	0.0840	0.1334
CRR	0.0840	0.1325

11% better than ranking-only

0.8% better than regression-only

Improvements are statistically significant

- How sensitive is the tradeoff parameter alpha?

Combined Ranking and Regression

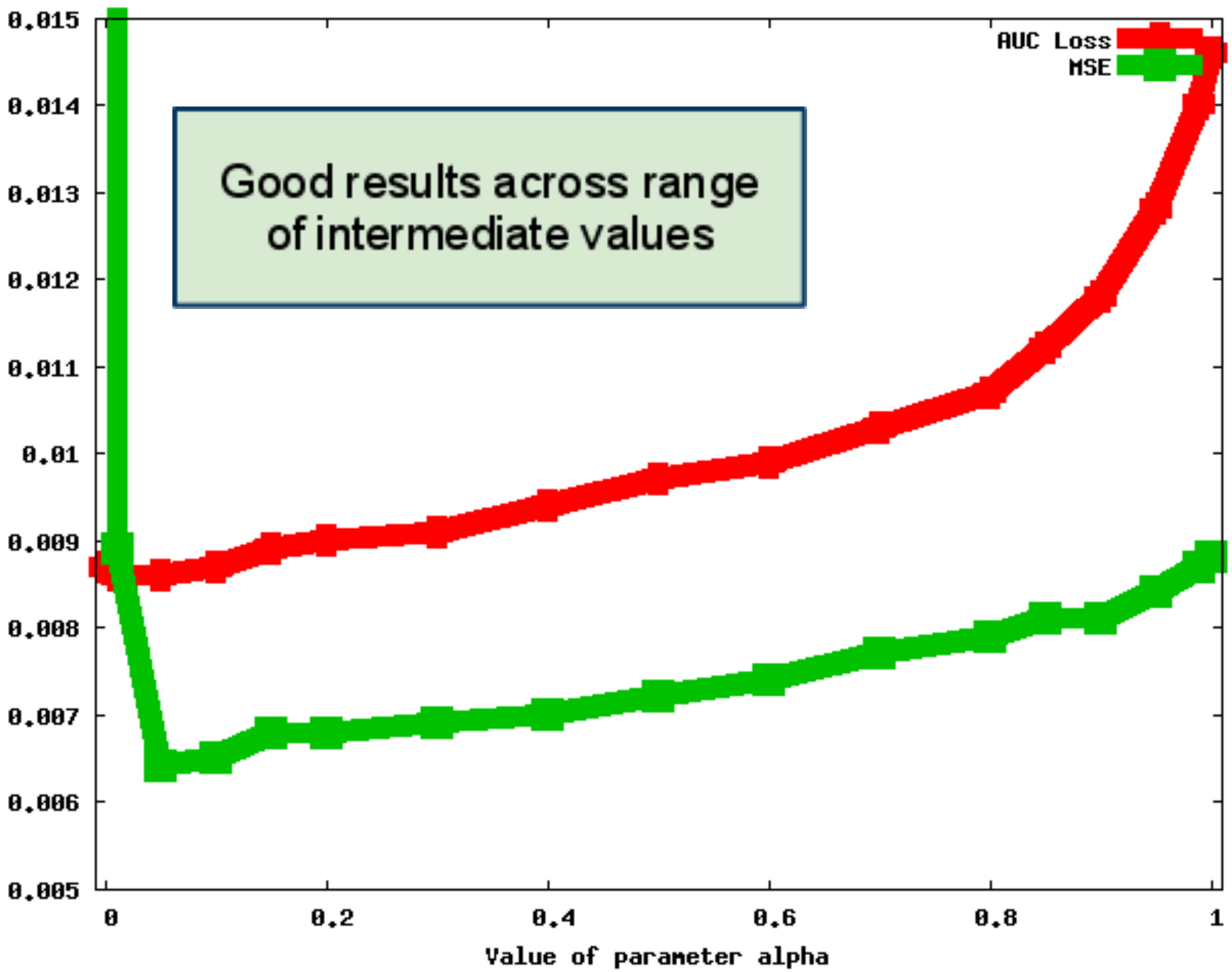
Regression Term

Ranking Term

$$\min_{\mathbf{w} \in \mathbb{R}^m} \alpha L(\mathbf{w}, D) + (1 - \alpha) L(\mathbf{w}, P) + \frac{\lambda}{2} ||\mathbf{w}||_2^2$$

Tradeoff Parameter

Looking at Tradeoff Parameter, alpha



Wrapping Up...

- Combined Ranking and Regression often gives "best of both" performance

- This algorithm uses pairwise method for rank-based component

- Simple, scalable, and robust

- Promising area for additional work
    - consider joint optimizations including MAP or NDCG optimization for ranking component

Thank you!

Questions?

Open Source Code: http://code.google.com/p/sofia-ml

Email: dsculley@google.com

RCV1

Task	% positive	Regression AUC Loss	Regression MSE	Ranking AUC Loss	Ranking MSE	CRR AUC Loss	CRR MSE
E141	0.05%	**0.000**	0.001	**0.000**	0.293	**0.000**	**0.000**
GOBIT	0.06%	0.002	**0.001**	**0.001**	0.162	0.002	**0.001**
E61	0.06%	0.002	**0.001**	**0.001**	0.320	**0.001**	**0.001**
GTOUR	0.10%	0.030	**0.001**	**0.005**	0.245	**0.005**	**0.001**
C331	0.13%	0.003	**0.001**	**0.001**	0.205	**0.001**	**0.001**
E143	0.15%	**0.001**	**0.001**	**0.001**	0.296	**0.001**	**0.001**
G152	0.15%	0.005	**0.001**	**0.003**	0.239	**0.003**	**0.001**
G155	0.16%	0.007	0.002	**0.004**	0.223	**0.004**	**0.001**
E411	0.17%	**0.002**	0.002	**0.002**	0.289	**0.002**	**0.001**
C313	0.18%	0.047	**0.002**	**0.014**	0.281	0.016	**0.002**
E311	0.19%	**0.001**	0.002	**0.001**	0.311	**0.001**	**0.001**
C32	0.19%	0.019	**0.002**	**0.012**	0.180	0.013	**0.002**
G157	0.19%	**0.001**	0.002	**0.001**	0.254	**0.001**	**0.001**
C16	0.21%	0.022	**0.002**	**0.012**	0.234	0.013	**0.002**
GWELF	0.22%	0.010	**0.002**	**0.005**	0.236	0.006	**0.002**
E513	0.23%	0.004	0.002	**0.003**	0.300	**0.003**	**0.001**
E14	0.28%	0.008	0.003	**0.003**	0.281	0.004	**0.002**
C173	0.33%	0.005	0.003	**0.004**	0.237	**0.004**	**0.002**
E121	0.41%	0.007	0.004	**0.004**	0.261	0.005	**0.003**
GENT	0.46%	0.014	**0.004**	**0.008**	0.126	**0.008**	**0.004**
C34	0.52%	0.018	0.005	**0.011**	0.231	0.012	**0.004**
GHEA	0.85%	0.007	0.008	**0.005**	0.140	0.006	**0.006**
C183	0.87%	0.013	0.008	**0.009**	0.275	0.010	**0.006**
GDEF	1.01%	0.015	0.009	**0.009**	0.208	**0.009**	**0.007**
C42	1.48%	0.009	0.010	**0.006**	0.242	0.007	**0.008**
E211	1.76%	0.013	0.011	**0.010**	0.245	**0.010**	**0.009**
E51	2.77%	0.025	0.019	**0.019**	0.280	0.021	**0.016**
M12	3.16%	0.010	0.015	**0.008**	0.288	0.009	**0.014**
C24	3.98%	0.031	0.027	**0.025**	0.157	0.026	**0.024**
GDIP	4.34%	0.019	0.023	**0.017**	0.188	0.018	**0.022**
M13	6.89%	**0.007**	**0.018**	**0.007**	0.221	**0.007**	**0.018**
GPOL	7.11%	0.021	**0.031**	**0.020**	0.175	0.021	**0.031**
C152	8.34%	0.026	0.036	**0.023**	0.178	0.024	**0.035**
C151	10.22%	0.010	**0.024**	**0.009**	0.188	**0.009**	0.025
M14	10.98%	0.005	0.021	**0.004**	**0.115**	**0.004**	0.022
ECAT	14.90%	0.033	0.054	**0.030**	0.188	0.031	**0.053**
C15	18.05%	**0.013**	**0.036**	**0.013**	0.132	**0.013**	0.037
MCAT	25.41%	0.011	**0.039**	**0.010**	0.113	**0.010**	0.043
GCAT	30.11%	**0.012**	**0.043**	**0.012**	0.062	**0.012**	0.046
CCAT	46.59%	**0.022**	**0.067**	**0.022**	0.073	**0.022**	0.070

Click Prediction Results

Method	AdSet1 AUC Loss	AdSet1 MSE
Regression	0.133	**0.084**
Ranking	**0.132**	0.094
CRR	**0.132**	**0.084**

0.8% improvement in AUC loss with same MSE
Difference is statistically significant