Combined Regression and Ranking

D. Sculley
Google, Inc.
Pittsburgh, PA USA
dsculley@google.com

ABSTRACT

Many real-world data mining tasks require the achievement of two distinct goals when applied to unseen data: first, to induce an accurate preference *ranking*, and second to give good *regression* performance. In this paper, we give an efficient and effective Combined Regression and Ranking method (CRR) that optimizes regression and ranking objectives simultaneously. We demonstrate the effectiveness of CRR for both families of metrics on a range of large-scale tasks, including click prediction for online advertisements. Results show that CRR often achieves performance equivalent to the best of both ranking-only and regression-only approaches. In the case of rare events or skewed distributions, we also find that this combination can actually improve regression performance due to the addition of informative ranking constraints.

Categories and Subject Descriptors

H.3.3 [**Information Systems**]: Information Storage and Retrieval—*Information Search and Retrieval*

General Terms

Algorithms, Measurement, Performance

Keywords

ranking, regression, large-scale data

1. INTRODUCTION

This paper addresses the real-world scenario in which we require a model that performs well on two distinct families of metrics. The first set of metrics are *regression* based metrics, such as Mean Squared Error, which reward a model for predicting a numerical value y' that is near to the true target value y for a given example, and penalize predictions far from y. The second set are rank-based metrics, such as Area under the ROC curve (AUC), which reward a model for producing predicted values with the same pairwise ordering $y'_1 > y'_2$ as the true values $y_1 > y_2$ for a pair of given examples.

In many settings good performance on both families together is needed. An important example of such a setting is the prediction of clicks for sponsored search advertising. In real-time auctions for online advertisement placement, ads are ranked based on $bid * pCTR$ where $pCTR$ is the predicted click-through rate (CTR) an ad. Predicting a good ranking is critical to efficient placement of ads. However, it is also important that the $pCTR$ not only give good ranking value, but also give good regression estimates. This is because online advertisements are priced using next-price auctions, in which the price for a click on an ad at rank i is based on the expected $bid * pCTR$ for the ad at the next lower rank [1]. In such a setting, it is critical that the actual $pCTR$ estimate be as accurate as possible to enable fair and efficient pricing. Thus, CTR prediction must be performed with both ranking and regression performance in mind.

1.1 Ranking vs. Regression

At first, it may appear that simply learning a good regression model is sufficient for this task, because a model that gives perfect regression will, by definition, also give perfect ranking. However, a model with near-perfect regression performance may yield arbitrarily poor ranking performance. Consider the case of an extreme minority-class distribution, where nearly all examples have target value $y = 0$ and only a small fraction have value $y = 1$. Here, it is possible to achieve near-perfect regression performance by always predicting 0; this gives a useless ranking. Even in less extreme cases, small regression errors can cause large ranking errors.

Similarly, it is easy to see that even a perfect ranking model may give arbitrarily poor regression estimates. Scores produced by a perfect ranking model may include arbitrary order-preserving transformations of the true target values, resulting in predictions that are useless as regression values.

1.2 Benefits of Combination

In this paper, we propose to address these issues using a combined objective function that optimizes regression-based and rank-based objectives simultaneously. This combination guards against learning degenerate models that perform well on one set of metrics but poorly on another. In our experiments, we find that the combined approach often gives "best of both" performance, performing as well at regression as a regression-only method, and as well at ranking as a ranking-only method.

Additionally, we find that the combined approach can ac-

KDD'10, July 25–28, 2010, Washington, DC, USA.

tually improve regression performance in the case of rare events. This is because adding a rank-based term to a general regression objective function brings an additional set of informative constraints that are not available to standard regression-only methods. In the case of rare events, for example, knowing that $y_a < y_b < y_c$ may considerably restrict the set of possible values for y_b, making effective regression estimates for y_b possible with many fewer observations. This scenario is commonly encountered in extreme minority class distributions, examined in our text classification experiments. Long-tailed distributions are another important example; in these distributions a large fraction of the distribution is composed of very low frequency events.

One might object that it would be simpler to learn two separate models, one for ranking and one for regression. However, we would then be faced with the problem of how to combine these scores. This would require a joint criteria similar to our CRR approach, but without the benefit of sharing information between the problems during simultaneous optimization.

1.3 Organization

The remainder of this paper is organized as follows. Section 2 lays out our notation and gives background in supervised regression and ranking. We present the CRR algorithm in Section 3 and show how it may be efficiently optimized for large-scale data sets. Section 4 reports experimental results on public data sets for text classification and document ranking, and on a proprietary data set for click prediction in sponsored search. The final sections survey related work and report our conclusions.

2. BACKGROUND

This section lays out the notation used in this paper. We also give background on common loss functions and supervised methods for regression and ranking. These serve as the building blocks for our combined approach.

2.1 Notation and Preliminaries

The data sets D described in this paper are composed of examples represented as tuples $(\mathbf{x}, y, q)$. Each tuple contains a feature vector $\mathbf{x} \in \mathbb{R}^m$ showing the location of the example in m-dimensional space, an associated label $y \in \mathbb{R}$, and an associated identifier $q \in \mathbb{N}$ denoting the query-shard for this example. The query-shard identifier q is useful in the case where each example in the data set is associated with a particular group, such as documents returned for a particular search query, and is commonly used in data sets for learning to rank [18]. In data sets that do not include query-shard identifiers, we can assume that $q = 1$ for all examples; this effectively recovers standard supervised-learning example-label pairs $(\mathbf{x}, y)$. We include a bias term in our feature set: every feature vector $\mathbf{x}$ has a fixed coordinate $\mathbf{x}_0 = 1$.

2.2 Supervised Regression Methods

The goal of supervised regression is to learn a model $\mathbf{w}$ that can predict a real-valued target $y' \in \mathbb{R}$ for a feature vector $\mathbf{x}$ using a prediction function $f(\mathbf{w}, \mathbf{x})$ with little loss with respect to a specified loss function $l(y, y')$. Regression models are useful when we care about the actual value of the prediction, as distinct from classification models that predict a discrete class label drawn from an unordered set of possible labels.

Because it is impossible to give a comprehensive overview of the wide range of regression methods in limited space, we focus on a particular form of regression using $L2$-regularized empirical risk minimization. (Readers wishing a broader review may refer to the text by Bishop [2] as an excellent starting point.) The goal of risk minimization is to incur little loss on unseen data. This aggregate loss $L(\mathbf{w}, D)$ is given by:

$$L(\mathbf{w}, D) = \frac{1}{|D|} \sum_{(\mathbf{x},y,q) \in D)} l(y, f(\mathbf{w}, \mathbf{x}))$$

Here, $l(y, y')$ is a loss function on a single example, defined on the true target value y and the predicted value y', and $f(\mathbf{w}, \mathbf{x})$ returns the predicted value y' using the model represented by $\mathbf{w}$. Specific regression loss functions $l(\cdot, \cdot)$ and their associated prediction functions $f(\cdot, \cdot)$ applied in this paper are described in Section 2.4.

The general formulation for $L2$ regularized empirical risk minimization is:

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

That is, we seek a linear model represented by weight vector $\mathbf{w}$ that both minimizes the loss of $\mathbf{w}$ on the training data D and also has low model complexity, represented by the squared norm of the weight vector [2]. The parameter λ controls the amount of regularization performed; tuning this parameter trades off the (possibly conflicting) goals of finding a model that is simple and finding a model that fits the data with little loss.

2.3 Supervised Ranking Methods

The goal of a supervised ranking method is to learn a model $\mathbf{w}$ that incurs little loss over a set of previously unseen data, using a prediction function $f(\mathbf{w}, \mathbf{x})$ for each previously unseen feature vector in the set, with respect to a rank-based loss function. Supervised learning to rank has been an active area of recent research.

A simple and successful approach to learning to rank is the *pairwise* approach, used by RankSVM [12] and several related methods [14, 10, 11]. (Exploring the use of other learning to rank methods in a combined ranking and regression framework is left to future work.) In this pairwise approach, the original distribution of training examples D is expanded into a set P of *candidate pairs*, and learning proceeds over a set of pairwise example vectors.

Formally, the set of candidate pairs P implied by a fixed data set D is the set of example pairs $(\mathbf{a}, y_a, q_a), (\mathbf{b}, y_b, q_b)$ drawn from all examples in D where $y_a \neq y_b$ and $q_a = q_b$. When $y_a > y_b$, then $\mathbf{a}$ is preferred over $\mathbf{b}$ (or, equivalently, ranked better than $\mathbf{b}$). In general for fixed D, $|P|$ is $O(|D|^2)$, but sharding by query identifier can result in $|P| \ll |D|^2$.

With P defined, we find $\mathbf{w}$ by optimizing a pairwise objective function:

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

Here, the loss function $L(\mathbf{w}, P)$ is defined over pairwise difference vectors from P:

$$L(\mathbf{w}, P) = \frac{1}{|P|} \sum_{((\mathbf{a},y_a,q_a),(\mathbf{b},y_b,q_b)) \in P)} l(t(y_a - y_b), f(\mathbf{w}, \mathbf{a} - \mathbf{b}))$$

The transformaton function $t(y)$ transforms the difference of the labels, and is instantiated differently for different loss functions (see Section 2.4); for squared loss $t(\cdot)$ is simply the identity function. Standard loss functions $l(\cdot,\cdot)$ are applicable on these pairwise difference vectors, given an appropriate transform $t(\cdot)$. For example, the RankSVM method by Joachims [12] uses hinge-loss and $t(y) = sign(y)$. Section 2.4 gives the loss functions and associated transformation functions that we apply in this paper.

2.4 Loss Functions

We explore two standard convex loss functions in this paper: squared loss and logistic loss. Our methods are general in that other convex loss functions could also be used. Here, we review these loss functions and give their associated prediction functions $f(\mathbf{w}, \mathbf{x})$, and the associated transformation functions $t(y)$ required for the computation of pairwise loss.

2.4.1 Squared Loss

The classical Least Mean Squares regression method [2] (also called Ridge Regression when used with $L2$ regularization) uses the squared loss functon. The squared loss for a single predicted value y' compared with true label y is given by $l(y, y') = (y - y')^2$. This loss function is convex. The associated transform function is the identity function $t(y) = y$. The associated prediction function is the standard dot product: $f(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle$.

2.4.2 Logistic Loss

The logistic loss function is most commonly applied in Logistic Regression, a method that is often thought of as a classification method but which can also be seen as a regression method for predicting real-valued probability scores [2]. The logistic loss function for $y \in [0, 1]$ and $y' \in [0, 1]$ is $l(y, y') = y \log y' + (1 - y) \log(1 - y')$. This loss function is convex. The associated prediction function is $f(\mathbf{w}, \mathbf{x}) = \frac{1}{1+e^{-\langle \mathbf{w}, \mathbf{x} \rangle}}$. The transformation function we use in this paper when computing pairwise loss is $t(y) = \frac{1+y}{2}$, which ensures that the resulting value of $t(y - y')$ is always in the range $[0, 1]$ when y and y' are also in $[0, 1]$.

2.4.3 Other Loss Functions

As noted above, other convex loss function could be applied in this framework; convex loss functions ensure that the gradient-based algorithm given in Section 3.1 converge to a globally optimal value. In particular, we explored the use of hinge loss in preliminary experiments, but found that it was out-performed by the other loss functions on the data sets in this paper.

3. ALGORITHM: CRR

In this section, we detail our proposed combined regression and ranking (CRR) approach, giving a general framework and an efficient algorithm suitable for massive data sets. An implementation of this algorithm in C++ is freely available at `http://code.google.com/p/sofia-ml`

3.1 General Framework

We build off the regression and ranking approaches described in Section 2 to create an optimization problem with terms for regression loss $L(\mathbf{w}, D)$ and pairwise ranking loss $L(\mathbf{w}, P)$. The combined CRR optimization problem is:

$$\min_{\mathbf{w} \in \mathbb{R}^m} \alpha L(\mathbf{w}, D) + (1 - \alpha) L(\mathbf{w}, P) + \frac{\lambda}{2} ||\mathbf{w}||_2^2 \qquad (3)$$

Here, the parameter $\alpha \in [0, 1]$ trades off between optimizing regression loss and optimizing pairwise loss. Note that setting $\alpha = 1$ recovers the standard regression problem given in Section 2.2, and setting $\alpha = 0$ recovers the pairwise ranking problem given in Section 2.3. Setting α to an intermediate value forces the optimization to consider both regression and ranking loss terms. We have found that CRR is not overly sensitive to specific values of α.

This combined framework is suitable for use with a range of convex loss functions. As described in Section 2.4, we apply logistic loss and squared loss in our experiments.

Algorithm 1 Combined Regression and Ranking.
Given: tradeoff parameter α, regularization parameter λ, training data D, iterations t

$\mathbf{w_0} \leftarrow \oslash$
for $i = 1$ to t **do**
  pick z uniformly at random from $[0, 1]$
  if $z < \alpha$ **then**
    $(\mathbf{x}, y, q) \leftarrow \text{RandomExample}(D)$
  else
    $((\mathbf{a}, y_a, q), (\mathbf{b}, y_b, q)) \leftarrow \text{RandomCandidatePair}(P)$
    $\mathbf{x} \leftarrow (\mathbf{a} - \mathbf{b})$
    $y \leftarrow \text{t}(y_a - y_b)$
  end if
  $\eta_i \leftarrow \frac{1}{i\lambda}$
  $\mathbf{w}_i \leftarrow \text{StochasticGradientStep}(\mathbf{w}_{i-1}, \mathbf{x}, y, \lambda, \eta_i)$
end for
return $\mathbf{w}_t$

3.2 Efficient Computation

A naive algorithm for optimizing the CRR objective function would enumerate the full set P of candidate pairs. Because $|P|$ is quadratic in $|D|$, this would be intractable for large-scale data sets. Joachims gave a $O(|D| \log |D|)$ method [13], but even this is impractical as $|D|$ grows large. Instead, we take the approach of [21] and sample from P rather than constructing P explicitly.

Algorithm 1 gives a method for efficiently solving the CRR optimization problem using stochastic gradient descent. Stochastic gradient descent methods have proven to be extremely practical for massive data sets, reducing training times by several orders of magnitude over more sophisticated optimization methods [3, 23].

The method StochasticGradientStep is instantiated differently for different loss functions. For squared loss, with $y \in \mathbb{R}$, the method returns:

$$(1 - \eta_i \lambda) \mathbf{w}_{i-1} + \eta_i \mathbf{x} (y - \langle \mathbf{w}_{i-1}, \mathbf{x} \rangle)$$

For logistic loss (assuming $y \in \{0, 1\}$), the method StochasticGradientStep returns [20]:

$$(1 - \eta_i \lambda) \mathbf{w}_{i-1} + \eta_i \mathbf{x} \left(y - \frac{1}{1 + e^{-\langle \mathbf{w}_{i-1}, \mathbf{x} \rangle}} \right)$$

Finally, it is helpful to represent $\mathbf{w}$ as a scalar-vector product, allowing the $L2$ regularization step to be performed in constant time by modifying the scalar rather than modifying each element of $\mathbf{w}$ [23].

3.2.1 Convergence

It is easy to see that Algorithm 1 converges to a globally optimal value for the CRR objective function. Observe that although D and P are different sets, this optimization problem can be viewed as an optimization over a single distribution DP, which is composed of both original labeled examples $(\mathbf{x}, y, q) \in D$ and pairwise-difference examples $(\mathbf{a} - \mathbf{b}, t(y_a - y_b), q)$ implied by the candidate pairs in P, in proportion α to $1 - \alpha$. Because we employ convex loss functions, stochastic gradient descent will converge to a globally optimum value. And because Algorithm 1 samples from DP directly, the solution to which this algorithm converges is the solution to the CRR optimization problem.

We set the learning rate η_i for step i using the schedule for the Pegasos algorithm: $\eta_i = \frac{1}{i\lambda}$. We tested other methods of setting learning rate such as $\eta_i = \frac{c}{c+i}$ and $\eta_i = c$ for fixed constants c without observing improvement.

3.2.2 Efficient Sampling from P

Efficient sampling from P is possible by indexing D. We borrow these indexed sampling methods from the fast learning to rank methods in [21].

In the case where there are exactly two unique values for $y \in \{0, 1\}$ and all examples belong to the same query shard, it is possible simply to divide D into D_0 containing all examples with $y = 0$ and D_1 containing all examples with $y = 1$. To sample from P, pick one example uniformly at random from D_0 and one example uniformly at random from D_1. This sampling can be performed in constant time.

When there are many query shards and no restrictions on y, then we can build an index of D, with an index for each query shard mapping from each possible y value to the set of examples in that query shard with that y value [21]. The sampling weights for each query shard depend on the number of possible candidate pairs in that shard; this can be computed in closed form given the number of examples for each unique y value in that shard. Exact sampling from P in this case requires binary searches to select a candidate pair. This can be performed in $O(\log |Q| + \log |Y|)$ time, where $|Q|$ is the number of query shards and $|Y|$ is the maximum number of unique y values in any query shard. Approximate sampling from P may be achieved more quickly, for example, by assigning equal weight to each query shard [21].

3.2.3 Scalability

Our algorithm completes training in time $O(ts + |D|)$, where t is the number of iterations and s is the maximum number of non-zero values in any feature vector $\mathbf{x}$. The exact value of t needed for convergence depends on the data set. As an example, 10^6 iterations were more than sufficient for state of the art results in AUC Loss on the RCV1 data set. Training times, completed in less than three CPU seconds on a normal laptop for both RCV1 and LETOR tasks.

The computational complexity stated above assumes that we can sample a candidate pair from P in constant time; the training cost increases slightly if we need to use binary search to sample from P. This extra fixed cost can be avoided if we are willing to use rejection sampling from D to find candidate pairs; however, for data sets that fit in main memory on a single machine, it is faster simply to index the data.

3.3 Non-Linear Models

We have described linear methods thus far, using predictions that depend on dot products $\langle \mathbf{w}, \mathbf{x} \rangle$. CRR may be extended to non-linear methods using a trick from Balcan and Blum [19]. The idea is to select a kernel or similarity measure $k(\mathbf{x}, \mathbf{x}')$, sample s examples $\mathbf{x}^1, \ldots, \mathbf{x}^s$ from D, and map each feature vector $\mathbf{x}$ to a new feature vector $\mathbf{x}'$ where the the value of feature i is the value of $k(\mathbf{x}, \mathbf{x}^i)$ This allows non-linear models to be learned in the CRR framework using the same generic algorithm. We leave exploration of this non-linear extension to future work.

4. EXPERIMENTS

In this section, we test the effectiveness of the CRR method against its natural competitors: regression-only methods using the same loss function as the regression-based component of CRR, and ranking-only methods using the same loss function as the rank-based component of CRR. Our experiments are conducted on public benchmark data sets for text mining and document ranking, and on a proprietary data set for click prediction drawn from sponsored search advertisements on Google Search.

We find that CRR often gives "best of both" performance, yielding performance on regression-based metrics approaching or exceeding that of regression-only methods, and approaching or exceeding the performance of rank-only methods on rank-based methods.

4.1 Performance Metrics

We evaluate each learned model $\mathbf{w}$ using the following regression-based and rank-based performance metrics on examples in held out test data sets D_t.

4.1.1 Mean Squared Error (MSE)

This classical measure is our primary regression-based performance metric. MSE is computed as:

$$\frac{1}{|D_t|} \sum_{(\mathbf{x}, y, q) \in D_t} (y - f(\mathbf{w}, \mathbf{x}))^2$$

Values closer to zero show better performance.

4.1.2 AUC Loss

For problems with target values in $\{0, 1\}$, we use AUC Loss as our ranking-based performance metric, computed as $1 - AUC$, where AUC is the area under the ROC curve [4]. AUC Loss can be interpreted as the probability that a randomly selected example with $y = 0$ is mistakenly ranked above a randomly selected example with $y = 1$. Values closer to zero show better performance.

4.1.3 Mean Average Precision (MAP)

Mean Average Precision is a rank-based metric defined for *relevant* and *non-relevant* examples across a set of query shards, and is based on the $P@n$ metric and $AP(q)$, the Average Precision for a single query [18].

The metric $P@n$ shows the precision achieved by considering only the top n examples in the ranked list. If there are r_n relevant documents in the top n examples, then $P@n = \frac{r_n}{n}$. The $AP(q)$ metric averages the $P@n$ metric over possible values of n. Let r_q be the total number of relevant examples for this query shard, and $|Q|$ be the total number of examples in this query shard, and $r(n)$ be a function returning 1

if the n-th ranked example is relevant and 0 otherwise.

$$AP(q) = \frac{1}{r_q} \sum_{n=1}^{|Q|} P@n * r(n)$$

MAP is then the arithmetic mean of $AP(q)$ for all query shards q in the data set [18]. Values closer to 1.0 show better performance.

4.1.4 NDCG: Normalized Discounted Cumulative Gain

The NDCG metric is appropriate for assessing the quality of a ranking when multiple levels of relevance, such as *not relevant*, *relevant*, and *extremely relevant*. Briefly, NDCG is defined as:

$$Z_n \sum_{j=1}^{n} \frac{2^{y_j} - 1}{\log(1+j)}$$

Here, y_j is the target value for the j-th example (with higher values of y showing stronger relevance), and Z_n is a normalization constant ensuring that the perfect $NDGC$ score for the given set of examples is 1.0 [18].

4.2 RCV1 Experiments

Our first set of experiments was run on the publicly available RCV1 benchmark corpus for text mining [17]. We use this data set because it has several qualities of interest: it is relatively large, has a sparse high dimensional feature-set, and contains learning tasks (topic identification) with a range of class distribution levels including balanced distributions and extreme minority class distributions.

In this setting, the motivation for the CRR approach is that we may desire to predict excellent rankings of relevant (in topic) versus non-relevant (out of topic), and also to return relevance scores that have consistent meaning from task to task for end users. Thus, both ranking and regression performance are important here.

4.2.1 RCV1 Experimental Setup

Because we are primarily interested in large-scale learning tasks, we used the larger of the two standard splits (781,265 examples) for training, and the smaller (23,149 examples) for testing. We selected 40 topics (from the 103 total topics available) from this set for binary one-versus-all topic prediction, assigning a label of 1.0 for an example of the given topic and a label of 0.0 for all other examples. Topics were selected for representative coverage of a range of class distribution levels, as shown in Table 1. The class distribution ranged from 0.05% positive for the smallest topic we tested to nearly 50% for the largest.

Parameter tuning of the regularization parameter λ was performed using cross validation on the training data for each method. The CRR parameter α was set to a default value of 0.5. We used 1,000,000 stochastic gradient descent steps for all methods; training completed within three CPU seconds for all methods on a normal 2.4GHz laptop.

We used logistic loss as the loss function for these experiments; thus, the regression-only method is $L2$-regularized Logistic Regression. We also experimented with hinge-loss (*i.e.* linear SVM), but found the methods using logistic loss to give stronger performance. To our knowledge, the results for AUC loss are the best reported results for this data set in the literature.

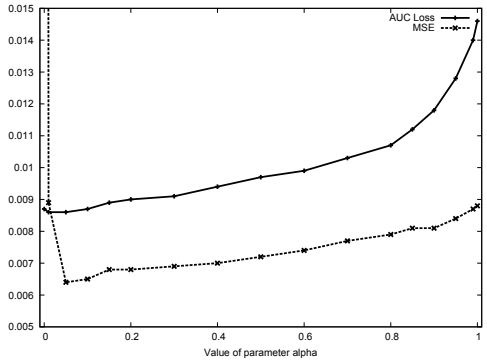

Figure 1: Effect of varying CRR tradeoff parameter α on GDEF task from RCV1. A range of intermediate values give MSE results that are better than either rank-only ($\alpha = 0$) or regression-only ($\alpha = 1$), while AUC Loss monotonically decreases as α approaches 0.

4.2.2 RCV1 Results

The results for all 40 topics are given in Table 1, and show several important trends. First, the ranking-only method does, indeed, give stronger AUC Loss performance than the regression-only method. Likewise, the regression-only method strongly out-performs the ranking-only method on MSE, giving orders of magnitude reduction in comparison.

Second, the CRR method gives the desired effect of achieving excellent performance in both metrics. The CRR method achieves both best AUC Loss and best MSE together on 16 of the 40 tasks. On another 19 tasks, CRR achieves best performance on one metric and performance within 0.001 (absolute difference) of best on the other. Finally, CRR always achieves best performance on at least one metrics, and is never worse than 0.004 (absolute difference) of best on any metric for any task.

CRR gives especially strong MSE performance on minority class distributions of less than 10%. Here, CRR outperformed the regression-only method on 20 of the 33 topics in this range, and equalled the performance of regression-only on the remaining topics. A Wilcoxon signed rank test showed this improvement to be statistically significant with $p = 0.002$. This result shows that adding ranking information to minority-class regression problems can, indeed, improve regression performance as suggested in Section 1.2.

Table 1 shows that the benefits of CRR tend to diminish as the class distribtion becomes more evenly balanced. The regression-only method gives "best of both" performance on three of the four most balanced tasks. Here, the balanced distribution across a large amount of training data makes the addition of ranking information superfluous.

After these tests were completed, we went back to examine the effect of varying α. Results for the GDEF topic are shown in Figure 1. For this topic, we see that a wide range of intermediate values for α give improved performance on ranking metrics compared with regression-only methods ($\alpha = 1$) and give improved performance on regression metrics compared with ranking-only methods ($\alpha = 0$). (Although our default value of $\alpha = 0.5$ was not optimal for CRR for this task, it still gave good results on both metrics.)

Finally, although we are primarily interested in regression and ranking metrics in this paper, we also looked at

Task	% positive	Regression AUC Loss	MSE	Ranking AUC Loss	MSE	CRR AUC Loss	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

Table 1: RCV1 Experimental Results. The regression method tends to out-perform the ranking-only method in Mean Squared Error (MSE), while the rank-only method out-performs the regression-only method in AUC loss. The combined CRR method tends to do nearly as well as the best of both, simultaneously. These differences are most pronounced on minority class problems.

Method	MQ2007 MAP	MQ2007 NDCG	MQ2007 MSE	MQ2008 MAP	MQ2008 NDGC	MQ2008 MSE
Regression	0.456	0.492	**0.182**	0.464	0.476	**0.144**
RankSVM (baseline [18])	0.464	0.497	-	0.470	0.483	-
Ranking	**0.466**	**0.500**	0.498	**0.479**	**0.490**	0.850
CRR	0.465	0.498	0.214	**0.479**	0.489	0.229

Table 2: LETOR 4.0 Results. The regression-only method performs best on Mean Squared Error (MSE), while the ranking-only method performs best on the rank-based metrics Mean Average Precision (MAP) and Normalized Discounted Cumulative Gain (NDCG). The combined CRR method performs nearly as well as the ranking-only method on MAP and NDCG, with MSE approaching that of the regression-only method. The baseline results for RankSVM are previously reported benchmark results [18].

classification-based metrics for this data. We found that both the CRR and ranking-only method out-performed logistic regression for classification accuracy, when using decision thresholds set to maximize accuracy for each task. CRR and ranking-only both gave macro-averaged accuracy of 0.882, while logistic regression gave macro-averaged accuracy of 0.861.

4.3 LETOR Experiments

Another typical setting in which we may wish to have both good regression values and good ranking performance is the scenario in which we wish to return relevant documents to an end user, and attach with them a graphical representation of the relevance, such as a number of "star" icons (with fractional stars allowed). Here, it is important that the ranking be effective, and also that the graphical representation have a consistent meaning across queries. For example, a rating of $1\frac{1}{2}$ stars should have a query-independent interpretation.

We explore this scenario using public benchmark data from the LETOR 4.0 learning to rank data set [18].

4.3.1 LETOR Experimental Setup

There are two tasks in this data set, MQ2007 and MQ2008, drawn from TREC challenge data sets in information retrieval. The examples in this data set have relevance labels in $\{0, 1, 2\}$ with 2 being most relevant. The examples are sharded by query identifier, and represent candidate documents that a search engine might return for the associated query. The provided feature set is a dense set of various information retrieval features such as TF-IDF similarity measures between query and document, PageRank, and HITS [18].

This benchmark data set is most commonly used to evaluate ranking-only methods [18] with the goal of ranking relevant documents above non-relevant ones. We consider the additional task of predicting the actual relevance value as an associated regression task, in addition to the ranking task. Predicting the actual relevance labels would allow the search engine to annotate the documents with between 0 and 2 stars, for example. It would also allow the search engine to set a consistent query-independent quality threshold, so that documents not meeting a standard quality level were not shown to users.

We used squared loss as the loss function for our methods, in order to predict the actual relevance value. Thus, our regression-only method is the classical Ridge Regression method, also described as $L2$-Regularized Least Mean Squares [2]. We also experimented with hinge loss and logistic loss (thresholding the y values) without improvement.

Each task provides canonical splits for 5-fold cross validation, including specified tuning, training, and test data for each fold. We followed standard parameter tuning practices for each method using coarse grid search, and report the mean test values across all 5 folds. The CRR parameter $\alpha = 0.5$ was set as a default.

For evaluation, we use MAP and NDCG (see Section 4.1) as the ranking-based metrics. These metrics are standard for these benchmark tasks, allowing comparison with previously published results. Our regression-based metric is MSE.

4.3.2 LETOR Results

The results for our LETOR experiments are given in Table 2. The CRR method gives ranking performance that is statistically indistinguishable from the rank-only method using t-tests for both MAP and NDGC, but which is significantly better than that of the regression-only method ($p < 0.005$). The ranking performance of CRR is better than the previously published results for RankSVM [18], although not as strong as methods designed specifically to optimize MAP or NDGC that go beyond simple pairwise learning [18].

For regression-based performance, the CRR method improves sharply over the MSE performance of the rank-only method, but only approaches the performance of the regression-only method. On MQ2007, CRR improves MSE by a factor of 2 compared with rank-only, and exceeds the MSE of the regression-only method by just 18%. On MQ2008, CRR improves MSE by a factor of 4 compared with rank-only, but remains 60% above that of regression-only.

Overall, these results are promising considering that the class distribution in these tasks is relatively balanced. Nearly 26% of the examples in MQ2007 and 24% of the examples in MQ2008 are at least *relevant*, making these tasks ones we expected CRR to give less benefit on. While CRR does not achieve "best of both" results, it does yield ranking results equivalent to the rank-only method with much less MSE.

4.4 Predicting Clicks in Sponsored Search

We now turn our attention to a commercial application, predicting clicks in sponsored search advertisement.

4.4.1 Regression and Ranking for Sponsored Search

In a sponsored search setting, the search engine is faced with the problem of which advertisements (if any) to show when a user enters a search query. As mentioned in the introduction, we believe that the CRR approach is useful for this task for two key reasons.

First, because real-time virtual auctions for advertisement

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

Table 3: Click-Prediction Results. The CRR method gives "best of both" performance on click-prediction data, giving MSE equivalent to the regression-only method with 0.8% less AUC Loss.

placement are resolved by ranking ads based on $bid * pCTR$, achieving good ranking performance is critical for effective auction resolution.

Second, because online advertisement pricing is controlled by a next-price auction in which the price paid for a click is determined by the $bid * pCTR$ of the ad at the next lower rank [1], it is equally important to achieve accurate regression estimates of the $pCTR$ value.

4.4.2 Click-Prediction Experimental Setup

For this set of experiments, we use a data set of several million advertisement impressions sampled from live traffic on Google Search. Each example in the data set consists of a high-dimensional representation of the advertisement and a binary label of 1 *clicked* or 0 for *not clicked*.

We held out a portion of this data to use as separate tuning data, and used this data to tune parameters for each method using coarse grid search. To better imitate a real-world setting, we split the remaining data temporally, using the earlier portion of the data for training and the later portion for testing.

Because we assume that CTR is best expressed as a probability, we used the logistic loss function. Thus, the regression method is equivalent to $L2$-regularized logistic regression.

Our evaluation metrics were AUC Loss to assess ranking performance and MSE to assess regression performance.

4.4.3 Click-Prediction Results

The results, given in Table 3 show that the CRR method again gives "best of both" performance. The AUC Loss is equal to that of the rank-only method, and MSE performance is equal to the regression-only method. Although the relative improvement in AUC Loss may appear small, it is statistically significant with $p < 0.0001$. This 0.8% reduction in pairwise preference errors would have a large absolute impact if applied to queries for millions of users.

5. RELATED WORK

We are not aware of previous work in the literature that explicitly combines regression and ranking objectives for learning. Thus, we review related work in regression-only and ranking-only methods here.

5.1 Regression

As noted in Section 2.2, it is impossible to give a full overview of the wide field of regression in limited space. The techniques of Least Mean Square (LMS), Ridge Regression (*i.e.* LMS with $L2$-regularization), and Logistic Regression are all well studied and widely applied [2].

Although we only apply $L2$-regularization in this paper, it is worth noting that *sparsity* can be achieved using $L1$-norm regularization. Common techniques here are the LASSO ($L1$-regularized LMS) and $L1$-regularized logistic regression. Solving this $L1$-regularization optimization problem efficiently for large data sets is an area of active research, due to the discontinuity of the $L1$ penalty term. Contemporary methods include the use of interior point methods [15], truncated gradient descent descent [16], and projected gradient descent [9]. The CRR framework could easily be modified to include $L1$ regularization following similar approaches; we leave such investigations to future work.

The use of regression methods in sponsored search has been an area of active recent research. Ciaramita *et al.* found that a multi-layer regression model out-performed a ranking-only model for click prediction [8], but this work did not explore the idea of combined ranking and regression. Click feedback was used in conjunction with relevance information for placing contextual advertisements by Chakrabarti *et al.* [7]. This approach used an approximate logistic regression method with a special sparsity-enforcing constraint. Sculley *et al.* applied logistic regression in the related task of predicting *bounce rates*, which are indicative of user satisfaction with the advertisement landing page [22]; the CRR approach would be interesting to compare for this task.

5.2 Ranking

One standard method for supervised learning to rank is RankSVM [12], which employs the same basic pairwise approach given in Section 2.3 with hinge-loss as the loss function. RankSVM also allows direct use of kernels for non-linearity. Elsas *et al.* employ a similar loss function in their modification of the Voted Perceptron algorithm for ranking problems [10]. We found logistic loss and squared loss to be more effective for the data sets in this paper.

A related learning task is the problem of directly optimizing AUC; this is equivalent to solving a rank optimization problem with exactly two ranks over a single query shard. Joachims gives an efficient $O(n \log n)$ method for evaluating AUC loss, and provides an iterative framework for solving an SVM with this loss function, based on the idea of iteratively finding and optimizing the most violated constraint [13]. Sampling methods for pairwise optimization were given by Sculley for both AUC and general ranking problems [21],

Several other methods for learning to rank have been proposed. For example, Burgess *et al.* proposed the use of non-linear neural networks, applying a probabilistic pairwise cost function [5]. Freund *et al.* gave a boosting approach for learning to rank [11]. A list-wise approach to ranking, rather than a pairwise approach, was explored by Cao *et al.* [6].

Finally, several methods have been proposed for optimizing MAP and NDGC directly. These methods include the boosting approaches AdaRank-MAP and AdaRank-NDGC by Xu and Li [24], and an SVM approach for optimizing a relaxation of MAP by Yue *et al.* [25]. The CRR approach uses the simple pairwise method for efficiency on large data sets; it would be worth investigating combining regression with methods in this vein for optimizing ranking functions such as MAP and NDGC.

6. CONCLUSIONS

We have presented a combined regression and ranking method, CRR, that gives strong performance on both regression and ranking metrics. The use of stochastic gradient descent makes the algorithm easy to implement, and efficient

for use on large-scale data sets. We have found that CRR is especially effective on minority class distributions, and have demonstrated its applicability to the problem of CTR prediction in sponsored search. Given this abundance of data mining tasks involving rare events or long-tailed distributions, there are a wide range of application areas to which CRR may be applied.

In this paper, we have pointed out several areas for future work, including the use of non-linear prediction functions and applying $L1$ regularization to achieve sparse models. Perhaps the most interesting area for future work is the exploration of more sophisticated ranking functions to use in conjunction with regression functions. The pairwise approach we employ is simple and efficient; however, list-wise approaches and approaches for directly optimizing MAP or NDCG may yield additional benefit.

7. ACKNOWLEDGMENTS

We gratefully thank Gary Holt, H. Brendan McMahan, Andrew W. Moore, Sajid M. Siddiqi, Matthew Streeter, and Douglas L. Vail for their insightful comments on this work.

8. REFERENCES

[1] G. Aggarwal, A. Goel, and R. Motwani. Truthful auctions for pricing search keywords. In *EC '06: Proceedings of the 7th ACM conference on Electronic commerce*, 2006.

[2] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag New York, Inc., 2006.

[3] L. Bottou and O. Bousquet. The tradeoffs of large scale learning. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems*, volume 20, pages 161–168. NIPS Foundation (http://books.nips.cc), 2008.

[4] A. P. Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30, 1997.

[5] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, 2005.

[6] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li. Learning to rank: from pairwise approach to listwise approach. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, 2007.

[7] D. Chakrabarti, D. Agarwal, and V. Josifovski. Contextual advertising by combining relevance with click feedback. In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, 2008.

[8] M. Ciaramita, V. Murdock, and V. Plachouras. Online learning from click data for sponsored search. In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, 2008.

[9] J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra. Efficient projections onto the l1-ball for learning in high dimensions. In *ICML '08: Proceedings of the 25th international conference on Mach ine learning*, 2008.

[10] J. L. Elsas, V. R. Carvalho, and J. G. Carbonell. Fast learning of document ranking functions with the committee perceptron. In *WSDM '08: Proceedings of the international conference on Web search and web data mining*, 2008.

[11] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *J. Mach. Learn. Res.*, 4, 2003.

[12] T. Joachims. Optimizing search engines using clickthrough data. In *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2002.

[13] T. Joachims. A support vector method for multivariate performance measures. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, 2005.

[14] T. Joachims. Training linear svms in linear time. In *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2006.

[15] K. Koh, S.-J. Kim, and S. Boyd. An interior-point method for large-scale l1-regularized logistic regression. *J. Mach. Learn. Res.*, 8, 2007.

[16] J. Langford, L. Li, and T. Zhang. Sparse online learning via truncated gradient. *J. Mach. Learn. Res.*, 10, 2009.

[17] D. D. Lewis, Y. Yang, T. G. Rose, and F. Li. RCV1: A new benchmark collection for text categorization research. *J. Mach. Learn. Res.*, 5:361–397, 2004.

[18] T.-Y. Liu, T. Qin, J. Xu, W. Xiong, and H. Li. LETOR: Benchmark dataset for research on learning to rank for information retrieval. In *LR4IR 2007: Workshop on Learning to Rank for Information Retrieval, in conjunction with SIGIR 2007*, 2007.

[19] M.-F. M.F. Balcan and A. Blum. On a theory of learning with similarity functions. In *ICML '06: Proceedings of the 23rd international conference on Machine learning*, 2006.

[20] T. M. Mitchell. Generative and discriminative classifiers: Naive bayes and logistic regression. In *Machine Learning*. http://www.cs.cmu.edu/∼tom/mlbook/NBayesLogReg.pdf, 2005.

[21] D. Sculley. Large scale learning to rank. In *NIPS 2009 Workshop on Advances in Ranking*, 2009.

[22] D. Sculley, R. G. Malkin, S. Basu, and R. J. Bayardo. Predicting bounce rates in sponsored search advertisements. In *KDD '09: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2009.

[23] S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal estimated sub-gradient solver for svm. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, 2007.

[24] J. Xu and H. Li. Adarank: a boosting algorithm for information retrieval. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, 2007.

[25] Y. Yue, T. Finley, F. Radlinski, and T. Joachims. A support vector method for optimizing average precision. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, 2007.