MIT-CSAIL-TR-2008-045 July 23, 2008

A Projected Subgradient Method for Scalable Multi-Task Learning

Ariadna Quattoni, Xavier Carreras, Michael Collins, and Trevor Darrell

A Projected Subgradient Method for Scalable Multi-task Learning

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

Recent approaches to multi-task learning have investigated the use of a variety of matrix norm regularization schemes for promoting feature sharing across tasks. In essence, these approaches aim at extending the l_1 framework for sparse single task approximation to the multi-task setting. In this paper we focus on the computational complexity of training a jointly regularized model and propose an optimization algorithm whose complexity is linear with the number of training examples and $O(n \log n)$ with n being the number of parameters of the joint model. Our algorithm is based on setting jointly regularized loss minimization as a convex constrained optimization problem for which we develop an efficient projected gradient algorithm. The main contribution of this paper is the derivation of a gradient projection method with $l_{1-\infty}$ constraints that can be performed efficiently and which has convergence rates of $O(1/\epsilon^2)$ for any convex Lipschitz loss function.

1 Introduction

Multi-task learning has had a relatively long history in machine learning [Ando and Zhang, 2005, Argyriou et al., 2006, Baxter, 1997, Obozinski et al., 2006, Raina et al., 2006, Amit et al., 2001, Torralba et al., 2006, Quattoni et al., 2008]. Broadly speaking the goal of multi-task learning is to exploit commonality among tasks by training classifiers for related problems together with a shared representation.

In this paper we focus on convex formulations of multi-task learning based on jointly regularized loss minimization. The main idea of joint regularized loss minimization is to cast the multi-task learning problem as a penalized convex optimization. Under this framework one seeks to find parameters $W = [\mathbf{w}_1, \ldots, \mathbf{w}_m]$ for m tasks which minimize the sum of the losses of each classifier with an additional penalty term that controls the “joint complexity” of the coefficient matrix W. In general, the penalty term is designed to promote feature sharing across tasks and to minimize the total number of features used by any task.

Different penalty terms have been proposed in the literature, typically involving a composition of norms over the coefficient matrix. Recent work considered models regularized with an $l_{1-\infty}$ norm [Tropp, 2006, Quattoni et al., 2008]; this norm penalizes the sum of the maximum absolute values of the coefficients of each feature across tasks, and can be shown to minimize the number of non-zero rows in the coefficient matrix.

We believe that to take full advantage of multi-task learning the optimization algorithm should scale to large number of dimensions, training examples and tasks. Existing methods for solving problems involving $l_{1-\infty}$ regularizaiton have relied on general LP and QP solvers, which are not typically able to scale to handle large numbers of input points. In this paper we propose a projected gradient algorithm for $l_{1-\infty}$ regularized joint loss minimization whose complexity is linear with the number of training examples and $O(dm \log dm)$, where d is the number of input dimensions and m is the number of tasks.

Because of their scalability properties, projected subgradient methods have been recently revived in the machine learning community for solving constrained optimization problems involving large number of examples and dimensions. For example, Shalev-Shwartz et al. [2007] proposed a projected subgradient method for solving large scale support vector machines, and Duchi et al. [2008] proposed an analogous algorithm with l_1 regularization. Our approach follows this line of research: we cast jointly regularized loss minimization as a convex constrained optimization and develop an efficient projected gradient algorithm. Results from optimization theory allow us to guarantee convergence rates of $O(1/\epsilon^2)$ for any convex loss function with our method. The main challenge in developing a projected gradient algorithm for $l_{1-\infty}$ constraints resides on being able to efficiently compute Euclidean projections onto the $l_{1-\infty}$ ball. We show that this can be done in $O(dm \log dm)$ time, where d and m are the number of dimensions and tasks.

2 Previous Work

In recent years various approaches for joint regularized loss minimization have been proposed [Obozinski et al., 2006, Argyriou et al., 2006, Amit et al., 2001, Ando and Zhang, 2005, Quattoni et al., 2008], in this section we focus on the convex formulations of the problem that are most closely related to our work.

Obozinski et al. [2006] proposed a joint regularization framework based on an l_{1-2} matrix norm on the problems coefficients W. The proposed norm penalizes the sum of l_2-norms of the block of coefficients associated with each feature across tasks. To optimize their objective they proposed a blockwise boosting scheme based on boosted-lazzo, an optimization algorithm developed for l_1 regularization. They show that for twice-differentiable strongly convex loss functions their algorithm will converge in a finite number of iterations, but no bound on the rate of convergence is available. Their algorithm can in theory be extended to handle strongly convex losses regularized with other l_{1-p} norms, although no applications with $l_{1-\infty}$ have been reported.

Argyriou et al. [2006] developed a related formulation also incorporating an intermediate hidden representation, at the additional computational cost of retraining classifiers at each iteration. They show that their objective can also be expressed as a convex optimization for which they develop an an iterative algorithm which is guaranteed to converge to the optimal solution.

Amit et al. [2001] proposed an alternative joint regularization framework based on a trace-norm penalty on the coefficients matrix, where the trace-norm is defined as the sum of the singular values of W, and derived a method that performs gradient descent on a smooth approximation of the objective.

Quattoni et al. [2008] recently proposed a model for jointly regularized optimization with a norm penalty that minimizes the total number of features involved in the approximation. Their joint regularization exploits a norm derived from simultaneous sparse signal approximation methods [Tropp, 2006], the $l_{1-\infty}$, which they optimize with a general LP solver. The number of variables of the resulting linear program is equal to the total number of training examples plus the number of input dimensions, and the number of constraints is of similar magnitude. It is well known that best linear-program solvers typically could not handle problems with more than several thousand variables. Thus, while this approach is feasible for small problems, it becomes intractable for large data-sets with a large number of tasks, and thousands of examples and dimensions.

The main differences between our approach and previous optimization algorithms for joint regularized loss minimization is that for $l_{1-\infty}$ constraints and any convex loss we can guarantee convergence rates that do not depend on the number of the training examples. Furthermore, as long as the subgradient of the cost function can be computed efficiently the overall cost each of iteration is small.

3 Joint Regularization for Multi-task Learning

Following Quattoni et al. [2008] we assume that we have a collection of related tasks and a set of data samples for each of them: $D = \{T_1, \dots, T_m\}$ where: $T_k = \{(\mathbf{x}_1^k, y_1^k), (\mathbf{x}_2^k, y_2^k), \dots, (\mathbf{x}_{n_k}^k, y_{n_k}^k)\}$ for $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \{+1, -1\}$

Let $\mathbf{w}_k$ be the parameters for the k-th problem and $W = [\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_m]$ to be a $d \times m$ matrix where $W_{j,k}$ corresponds to the j-th coefficient of the k-th problem. We are interested in learning linear classifiers for each problem of the form $f_k(\mathbf{x}) = \mathbf{w}_k^{\mathrm{T}}\mathbf{x}$.

Consider learning a single sparse classifier; i.e. a classifier with a small number of non-zero coefficients. Given a training set with n examples, a natural way of expressing the problem as a constrained convex optimization is:

$$\min_{\mathbf{w}} \ \frac{1}{n}\sum_{i=1}^{n} \mathrm{loss}(f(\mathbf{x}_i), y_i) \qquad \text{s.t.} \quad \sum_{j=1}^{d} |w_j| \leq C \tag{1}$$

Here $\mathrm{loss}(f(\mathbf{x}), y)$ in the objective is some convex loss function that measures the loss incurred by the classifier on an example. For the experiments in this paper we use the hinge loss defined as $\mathrm{hinge}(f(\mathbf{x}), y) = \min(0, 1 - yf(\mathbf{x}))$. The constant C in the constraints is a regularization parameter that controls the amount of sparsity in the model.

In the joint sparse approximation problem we are interested in learning m classifiers that are jointly sparse. By jointly sparse we mean that we wish only a few features to be non-zero in any of the m problems. It is easy to see that a jointly sparse solution would consist of a matrix W with only a small number of non-zero rows. Tropp [2006] showed that the $l_{1-\infty}$ regularization norm defined as:

$$||W||_{1-\infty} = \sum_{j=1}^{d} \max_k |w_{jk}| \tag{2}$$

is a convex relaxation of a pseudo-norm that counts the number of non-zero rows of W. Given this matrix norm a natural way of expressing the jointly sparse minimization as a constrained convex optimization problem would be:

$$\min_{W} \ \sum_{k=1}^{m} \frac{1}{n_k} \sum_{i=1}^{n_k} \mathrm{loss}(f(\mathbf{x}_i^k), y_i^k) \qquad \text{s.t.} \quad \sum_{j=1}^{d} \max_k |w_{jk}| \leq C \tag{3}$$

4 A Projected Subgradient Method for Joint Regularization

Our algorithm for optimizing equation (3) is based on the projected subgradient method for minimizing a convex function $F(\mathbf{z})$ subject to convex constraints of the form $\mathbf{z} \in Z$, where Z is a convex set [Bertsekas, 1999]. In our case $F(\mathbf{z})$ is the sum of average losses per task, $\mathbf{z}$ is the matrix W of coefficients for each problem and Z is the set of all matrixes with $||W||_{1-\infty} \leq C$.

A projected subgradient algorithm works by generating a sequence of solutions $\mathbf{z}^t$ via $\mathbf{z}^{t+1} = P_Z(\mathbf{z}^t - \eta\nabla^t)$. Here ∇^t is a subgradient of F at $\mathbf{z}^t$ and $P_Z(\mathbf{z}) = \min_{\mathbf{z}' \in Z} ||\mathbf{z} - \mathbf{z}'||$ is the Euclidean projection of $\mathbf{z}$ onto Z. Finally, η is the learning rate that controls the amount by which the solution changes at each iteration.

Standard results in optimization literature [Bertsekas, 1999] show that when $\eta = \frac{\eta_0}{\sqrt{t}}$ and $F(z)$ is a convex Lipschitz function the projected algorithm will converge to an ϵ-accurate solution in $O(1/\epsilon^2)$ iterations.

For the hinge loss, computing the subgradient of the objective of equation (3) at W is straightforward. The subgradient for the parameters of each task can be computed independently of the other tasks, and for the k-th task it is given by:

$$\nabla_k^t = \sum_{(\mathbf{x},y) \in T_k \ \text{s.t.} \ yf_k(\mathbf{x}) < 1} y\mathbf{x} \tag{4}$$

In the next section we show how to compute the projection onto the $l_{1-\infty}$ ball efficiently.

4.1 Euclidean Projection onto the $l_{1-\infty}$ Ball

In this section we describe an efficient method to project a matrix A to the $l_{1-\infty}$ ball. For now, we assume that all entries in A are non-negative, later we will show that this assumption imposes no limitation. The projection can be formulated as finding a matrix B as follows :

$$\min_{B,\boldsymbol{\mu}} \quad \frac{1}{2}\sum_{i,j}(B_{i,j} - A_{i,j})^2 \tag{5}$$

$$\text{s.t.} \quad B_{i,j} \le \mu_i \ , \quad \forall i,j \tag{6}$$

$$\sum_i \mu_i = C \tag{7}$$

$$B_{i,j} \ge 0 \ , \quad \forall i,j \ ; \quad \mu_i \ge 0 \ , \quad \forall i \tag{8}$$

In the above problem, the objective (5) corresponds to the Euclidean distance between A and B, whereas the constraints specify that B is in the boundary of the $l_{1-\infty}$ ball of radius C. To do so, there are variables $\boldsymbol{\mu}$ that stand for the the maximum coefficients of B for each feature i, as imposed by constraints (6), and that sum to the radius of the ball, as imposed by constraint (7). Constraints (8) stand for non-negativity of the new coefficients and maximum values.

The Lagrangian of the projection is given by:

$$\begin{aligned}\mathcal{L}(B,\boldsymbol{\mu},\boldsymbol{\alpha},\theta,\boldsymbol{\beta},\boldsymbol{\gamma}) \ = \ & \frac{1}{2}\sum_{i,j}(B_{i,j} - A_{i,j})^2 \ + \ \sum_{i,j}\alpha_{i,j}(B_{i,j}-\mu_i) \\ & + \ \theta\Big(\sum_i \mu_i - C\Big) \ - \ \sum_{i,j}\beta_{i,j}B_{i,j} \ - \ \sum_i \gamma_i\mu_i\end{aligned}$$

Differentiating $\mathcal{L}$ with respect to $B_{i,j}$ gives the optimality condition $\frac{\partial \mathcal{L}}{\partial B_{i,j}} = B_{i,j} - A_{i,j} + \alpha_{i,j} - \beta_{i,j} = 0$. At the same time, the complementary slackness conditions related to $\boldsymbol{\beta}$ imply that whenever $B_{i,j} > 0$ then $\beta_{i,j} = 0$. Therefore, whenever $B_{i,j} > 0$ we have that $\alpha_{i,j} = A_{i,j} - B_{i,j}$.

Differentiating $\mathcal{L}$ with respect to μ_i gives the optimality condition $\frac{\partial \mathcal{L}}{\partial \mu_i} = \theta - \sum_j \alpha_{i,j} - \gamma_i = 0$. The complementary slackness conditions imply that whenever $\mu_i > 0$ then $\gamma_i = 0$ and therefore $\theta = \sum_j \alpha_{i,j}$. Thus, $\theta = \sum_j A_{i,j} - B_{i,j}$ whenever $\mu_i > 0$. This means that when projecting A the sum of magnitudes removed from a feature will be θ, and this quantity will be constant across all the features.

A final observation from the Lagrangian reveals how to obtain the coefficients of B given the optimal maximums μ. At the optimal solution, the following holds for non-zero coefficients:

$$A_{i,j} \ge \mu_i \implies B_{i,j} = \mu_i \tag{9}$$

$$A_{i,j} \le \mu_i \implies B_{i,j} = A_{i,j} \tag{10}$$

To prove (9), assume that $A_{i,j} \ge \mu_i$ but $B_{i,j} \ne \mu_i$. By (6), if $B_{i,j} \ne \mu_i$ then $B_{i,j} < \mu_i$, which means $\alpha_{i,j} = 0$ due to complementary slackness. This implies that $B_{i,j} = A_{i,j}$, and therefore $A_{i,j} < \mu_i$, which contradicts the assumption. To prove (10), assume that $A_{i,j} \le \mu_i$ but $B_{i,j} \ne A_{i,j}$. If $B_{i,j} \ne A_{i,j}$ then $\alpha_{i,j} > 0$, and so $B_{i,j} = \mu_i$ due to complementary slackness. But since $\alpha_{i,j} > 0$, $A_{i,j} > B_{i,j} = \mu_i$, which contradicts the assumption.

With these results, the problem of projecting into the $l_{1-\infty}$ ball can be reduced to the following problem, which finds the optimal maximums $\boldsymbol{\mu}$:

$$\text{find} \quad \boldsymbol{\mu}\ ,\ \theta \tag{11}$$

$$\text{s.t.} \quad \sum_i \mu_i = C \tag{12}$$

$$\sum_{A_{i,j} > \mu_i} A_{i,j} - \mu_i = \theta \ , \quad \forall i \ \text{ s.t. } \ \mu_i > 0 \tag{13}$$

With $\boldsymbol{\mu}$ we can create a matrix B using equations (9) and (10). Intuitively, the new formulation reduces finding the projection to the $l_{1-\infty}$ ball to finding a new vector of maximum absolute values that will be used to truncate the original matrix. The constraints express that the cumulative mass removed from the coefficients of a feature is kept constant across all features, except for those features whose coefficients vanish.

So far we have assumed that the input matrix A is non-negative. For the general case, it is easy to prove that the optimal projection never changes the sign of a coefficient [Duchi et al., 2008]. Thus, given the coefficient matrix W used by our learning algorithm, we can run the $l_{1-\infty}$ projection on A, where A is a matrix made of the absolute values of W, and then recover the original signs after truncating each coefficient.

4.2 An Efficient Algorithm to Find $\boldsymbol{\mu}$

In this section we describe an efficient algorithm to solve problem (11). Given a matrix A and a ball of radius C, the goal is to find a vector $\boldsymbol{\mu}$ of maximums for the new projected matrix, such that $C = \sum_i \mu_i$. We start by defining the following function:

$$\mathrm{N}(\boldsymbol{\mu}) = ||A||_{1-\infty} - \sum_i \mu_i = \sum_i \max_j A_{i,j} - \sum_i \mu_i = \sum_i \max_j (A_{i,j} - \mu_i)$$

This function measures how much a candidate vector $\boldsymbol{\mu}$ reduces the norm of A. Note that by constraint (12) the optimal maximums must accomplish that $\mathrm{N}(\boldsymbol{\mu}) = ||A||_{1-\infty} - C = \delta$. Note also that this function can be decomposed feature-wise as $\mathrm{N}(\boldsymbol{\mu}) = \sum_i \mathrm{N}_i(\mu_i)$, where each component is $\mathrm{N}_i(\mu_i) = \max_j A_{i,j} - \mu_i$.

The maximums $\boldsymbol{\mu}$ truncate A causing a cumulative reduction in magnitude that must be constant across features, as expressed by constraints (13). We define a function that computes the cumulative reduction for a feature i, given a maximum value μ_i for the coeffiecients of that feature:

$$\mathrm{R}_i(\mu_i) = \sum_{A_{i,j} > \mu_i} (A_{i,j} - \mu_i)$$

Thus, we are interested in finding $\boldsymbol{\mu}$ and θ such that $\mathrm{N}(\boldsymbol{\mu}) = \delta$ and $\forall i \;\; \mathrm{R}_i(\mu_i) = \theta$. Our strategy is to first find θ, and then find $\boldsymbol{\mu}$ by building the inverse function of R_i, so $\mu_i = \mathrm{R}_i^{-1}(\theta)$.

To gain insight of the relation between $\mathrm{N}(\boldsymbol{\mu})$, $\mathrm{R}_i(\mu_i)$ and θ, we start by looking at the simpler relationship between the functions N_i and R_i and a new maximum value μ_i. Let $\mathbf{s}_i$ be a vector of the coefficients of feature i in A sorted in decreasing order, with an added 0 at position $m+1$, $s_{i,1} \geq s_{i,2} \geq \ldots \geq s_{i,m} \geq s_{i,m+1} = 0$. It is easy to see that R_i is piecewise linear with intervals $[s_{i,k}, s_{i,k+1}]$, and that the slope at the k-th interval is $-k$. Let us define a vector $\mathbf{r}_i$ where each component is $r_{i,k} = \mathrm{R}_i(s_{i,k}) = \sum_{j=1}^{k-1}(s_{i,j} - s_{i,k}) = \sum_{j=1}^{k-1} s_{i,j} - (k-1)s_{i,k}$.

Finally, consider the function $F_i(\theta) = \mathrm{N}_i(\mathrm{R}_i^{-1}(\theta))$, that given a reduction of the coefficients mass of feature i computes the reduction of the matrix norm. This function is also piecewise linear, with intervals $[r_{i,k}, r_{i,k+1}]$ for $1 \leq k \leq m$. Furthermore, we know that $F_i(r_{i,k}) = \mathrm{N}_i(s_{i,k}) = s_{i,1} - s_{i,k}$. Thus, the gradient of F_i can be computed as :

$$\frac{\mathrm{N}_i(s_{i,k+1}) - \mathrm{N}_i(s_{i,k})}{r_{i,k+1} - r_{i,k}} = \frac{s_{i,1} - s_{i,k+1} - s_{i,1} + s_{i,k}}{\sum_{j=1}^{k} s_{i,j} - k s_{i,k+1} - \sum_{j=1}^{k-1} s_{i,j} + (k-1)s_{i,k}} = \frac{1}{k}$$

Recall that N is the sum of N_i. Similarly, we can consider the sum of F_i, which is also a piecewise linear function, with intervals given by merging the values of all vectors $\mathbf{r}_i$. The algorithm in Figure 1 builds the pieces of each F_i incrementally, until it finds a θ such that $\sum_i F_i(\theta) = \delta$:

The cost of the algorithm is dominated by the sort and merge operations. The initial sorting of d vectors takes $O(dm \log m)$. Merging the vectors and visiting its elements takes $O(dm \log d)$. Thus, the total cost of the projection algorithm is $O(dm \log dm)$. Notice that the projection algorithm only needs to consider non-zero rows of A. Thus, in this complexity cost, d can be interpreted as the number of non-zero rows, rather than the actual number of features. This property is particulary attractive for learning methods that maintain sparse coefficients, such as online learning methods.

```
Algorithm   Given a matrix $A \in \mathbb{R}^{d \times m}$, and $\delta$ :
    • Create the vectors $\mathbf{s}_i$ by sorting the rows of $A$ in decreasing order. Then, with each $\mathbf{s}_i$,
      create vectors $\mathbf{r}_i$, sorted increasingly by construction.
    • Run a merge algorithm on the $d$ sorted vectors $\mathbf{r}_i$. At each iteration $t$ the merge algorithm
      visits an element $\langle i, k, r_{i,k} \rangle$, ensuring that elements are visited in increasing order in $r_{i,k}$.
      While merging:
        – Maintain a vector of gradients g for each feature, with $g_i = 1/k$, where $k$ is the
          number of values of $\mathbf{r}_i$ that have been visited.
        – Incrementally build a list of points $(r^t, n^t)$, where $r^t$ is the value visited at step $t$ of
          merge, and $n^t$ is the reduction in $l_{1-\infty}$ norm when $\theta = r^t$. Initially $n^0 = 0$, then:
              $n^t = n^{t-1} + \sum_i g_i (r^t - r^{t-1})$
    • Stop when $n^{t+1} > \delta$. With the gradient values after the update in step $t$, set $\theta$ as
              $\theta = r^t + (\delta - n^t) / \sum_i g_i$
      Let $k$ be the last visited value of $\mathbf{r}_i$ until step $t$, then set
              $\mu_i = \max \{ \, s_{i,k} - (\theta - r_{i,k})/k \; , \; 0 \, \}$
    Return $\boldsymbol{\mu}$
```

Figure 1: An algorithm to find optimal maximums $\boldsymbol{\mu}$.

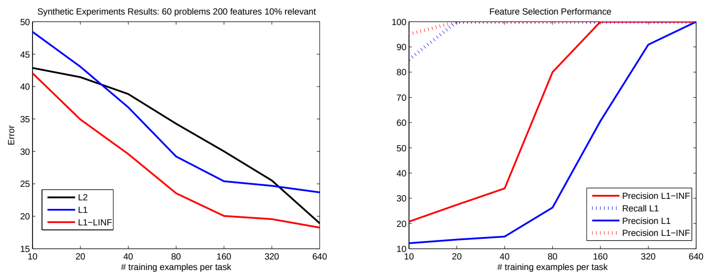

Figure 2: Synthetic experiments. Left: test error. Right: feature selection performance.

5 Experiments

For all experiments we compared the $l_{1-\infty}$ projection with both independent l_2 projections for each task and independent l_1 projections, To compute the independent l_1 projections we used the algorithm of Duchi et al. [2008]. In all cases we used a projected subgradient method, thus the only difference is in the projection step. For all the experiments we used the sum of average hinge losses per task as our objective function. The learning rate was set to $\eta_0/\sqrt{t}$, where η_0 was chosen to minimize the objective on the training data. All models were run for 200 iterations.

5.1 Synthetic Experiments

To create data for these experiments we first generated parameters $W = [\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_m]$ for all tasks, where each entry was sampled from a normal distribution with 0 mean and unit variance. To generate jointly sparse vectors we randomly selected 10% of the features to be the relevant feature set V. Then for each task we randomly selected a subset v and zeroed all parameters outside v.[1]

Each of the dimensions of the training points $\mathbf{x}_i^k$ for each task was also generated from a normal distribution with 0 mean and unit variance. All vectors were then normalized to have unit norm. The corresponding labels y_i^k were set to $\text{sign}(\mathbf{w}_k \mathbf{x}_k^i)$. The test data was generated in the same fashion. The number of dimensions for these experiments was set to 200 and the number of problems to 60.

[1]We make sure that $|v|$ is at least half the size of V

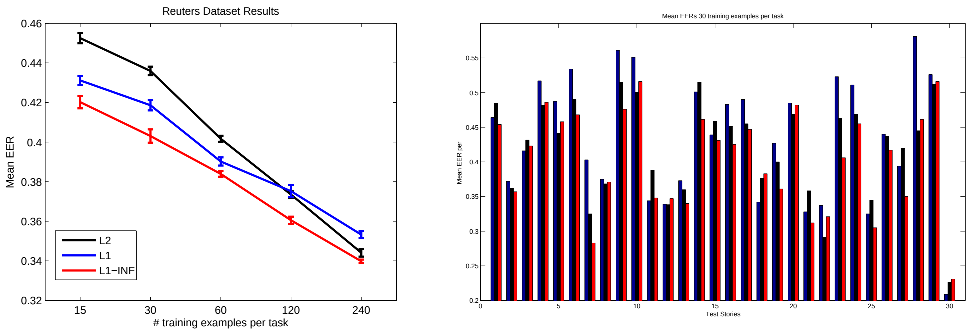

Figure 3: Average EERs on the Reuters Dataset.

We evaluated three types of projections: $l_{1-\infty}$, independent l_2 and independent l_1. For each projection the ball constraints C were set to be the true norm of the corresponding parameters. That is for the $l_{1-\infty}$ norm we set $C = ||W||_{1-\infty}$, for the independent l_2 norms we set $C_k = ||\mathbf{w}||_2$ and for the independent l_1 norms we set $C_k = ||\mathbf{w}||_1$. We trained models with different number of training examples ranging from 10 to 640 examples per task and evaluated the classification error of the resulting classifier on the test data.

Figure 2 shows the results of these experiments. As we would expect, given the amount of feature sharing between tasks, the $l_{1-\infty}$ projection results in better generalization than both independent l_1 and independent l_2 projections. Since we know the relevant feature set V, we can evaluate how good the l_1 and $l_{1-\infty}$ projections are in recovering these features. For each model we take the coefficient matrix W learnt and select all the features corresponding to non-zero coefficients for at least one task. The right plot shows precision and recall of feature selection for each model, as we increase the number of training examples per task. As we can see both the l_1 model and the $l_{1-\infty}$ can easily recognize that a feature is in the relevant set : the recall for both models is high even with very few training examples. The main difference between the two models is in the precision at recovering relevant features: the $l_{1-\infty}$ model returns significantly sparser solutions.

5.2 Visual Category Recognition Experiments

For this experiments we collected a dataset from the Reuters news-website. Images on the Reuters website have associated story or topic labels, which correspond to different stories in the news. There were a total of 25,589 images and 316 stories, an image can belong to one or more stories. The experiments involved the binary prediction of whether an image belonged to one of the 40 most frequent stories, these are all the stories for which we had at least 80 positive images in the training partition. Story examples are: Golden Globes, Australian Open, Danish Cartoons or Afghanistan.

We partitioned the data into four sets: 5,000 images were used as unlabeled data to compute an image representation, 5,000 images were reserved as validation data, and 10,589 images as training data. The remaining 5,000 images where used for testing. For each of the 40 most frequent topics we created multiple training sets of different sizes, $n = \{15, 30, 60, 120, 240\}$: each training set contained n positive examples and $2n$ negative examples. All examples were randomly sampled from the supervised training data. We performed 10 runs of the experiments by randomizing the selection of training examples.

For all the experiments we computed and image representation in two stages. In the first stage we create and initial bag of words representation, containing 3,000 visual words. In the second stage we used the unlabeled data $U = [x_1, x_2..x_u]$, where x_i is the bag of words representation of an image, to compute our final representation. We do this by performing SVD on U to obtain a projection matrix P, where each column of U corresponds to an unlabeled data sample. Finally, the new image representation is built by projecting the initial bag of words representation to the space spanned by the columns of P. The size of the final image representation is given by the rank of U which in these experiments was 3,000.

As in the synthetic experiments we compare the $l_{1-\infty}$ projection with independent l_1 projections and independent l_2 projections. To validate the constant C for each model we assume that we have 10 topics for which we have validation data. We chose the C that minimized the average Equal Error Rate on these topics, for tried values of $C = \{1, 10, 100\}$.

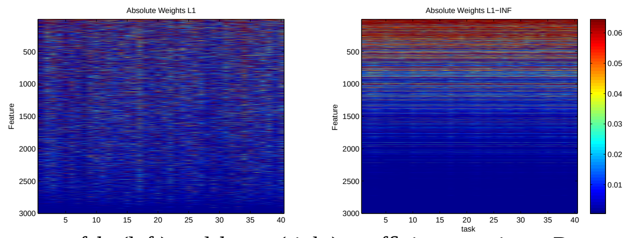

Figure 4: Sparsity patterns of l_1 (left) and $l_{1-\infty}$ (right) coefficient matrices. Rows corresponds to features, whereas columns correspond to tasks. The right legend indicates the map weights to colors.

Figure 3 shows results on the Reuters dataset. The left plot shows the average EER over the 30 non-validation topics for the three different types of projections. It can be seen that independent l_1 projection results in better performance than independent l_2 projection but they are both outperformed by $l_{1-\infty}$ projection. The right graph shows the average EER for each of the 30 stories for models trained with 30 examples. It can be observed that the performance of the $l_{1-\infty}$ model is quite robust as it improves performance for all but a few topics. Figure 4 shows the absolute values of the coefficient matrices learnt with 30 examples for l_1 and $l_{1-\infty}$ regularization. Clearly, $l_{1-\infty}$ produces a joint sparsity pattern.

6 Conclusion

In this paper we have presented a simple and effective algorithm for training joint models with $l_{1-\infty}$ constraints. The proposed algorithm has a convergence rate of $O(1/\epsilon^2)$ and a computational cost that scales linearly with the number of examples and with $O(dm \log dm)$ with the number of problems and dimensions, d and m respectively. The experiments show that this algorithm can find solutions that are jointly sparse, resulting in lower test error.

One of the advantages of our approach is that it can be easily extended to work on an online setting, where the subgradients would be computed with respect to one or few examples. Indeed, our algorithm can be easily modified to fit the framework of online convex optimization proposed by Zinkevich [2003]. This would result in online algorithm for jointly-regularized multi-task learning with provable regret bounds and computational const which can scale to real-sized datasets. Future work should explore this possibility.

References

Y. Amit, M. Fink, N. Srebro, and S. Ullman. Uncovering shared structures in multiclass classification. In *Proceedings of ICML*, 2001.

R. Ando and T. Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6:1817–1853, 2005.

A. Argyriou, T. Evgeniou, and M. Pontil. Multi-task feature learning. In *Proceedings of NIPS*, 2006.

J. Baxter. A bayesian/information theoretic model of learning to learn via multiple task sampling. *Machine Learning*, 28, 1997.

D. Bertsekas. *Nonlinear Programming*. Athena Scientific, 1999.

J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra. Efficient projections onto the l1-ball for learning in high dimensions. In *Proceedings of International Conference on Machine Learning*, 2008.

G. Obozinski, B. Taskar, and M. Jordan. Multi-task feature selection. In *Technical Report*, 2006.

A. Quattoni, M. Collins, and T. Darrell. Transfer learning for image classification with sparse prototype representations. In *Proceedings of CVPR*, 2008.

R. Raina, A. Y. Ng, and D. Koller. Constructing informative priors using transfer learning. In *Proceedings of the 23rd International Conference on Machine learning*, pages 713–720, 2006.

S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal estimated sub-gradient solver for svm. In *Proceedings of International Conference on Machine Learning*, 2007.

A. Torralba, K. Murphy, and W. Freeman. Sharing visual features for multiclass and multiview object detection. *Pattern Analysis and Machine Intelligence*, In press, 2006.

J. Tropp. Algorithms for simultaneous sparse approximation, part ii: convex relaxation. In *Signal Process. 86 (3) 589-602*, 2006.

M. Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of ICML*, 2003.