Blockwise Coordinate Descent Procedures for the Multi-task Lasso, with Applications to Neural Semantic Basis Discovery

Han Liu HANLIU@CS.CMU.EDU
Machine Learning Department, School of Computer Science, Carnegie Mellon University, PA 15213 USA

Mark Palatucci MPALATUC@CS.CMU.EDU
School of Computer Science, Carnegie Mellon University, 5000 Forbes Ave., Pittsburgh, PA 15213 USA

Jian Zhang JIANZHAN@STAT.PURDUE.EDU
Department of Statistics, Purdue University, 250 N. University St., West Lafayette, IN, 47907 USA

Abstract

We develop a cyclical blockwise coordinate descent algorithm for the multi-task Lasso that efficiently solves problems with thousands of features and tasks. The main result shows that a closed-form Winsorization operator can be obtained for the sup-norm penalized least squares regression. This allows the algorithm to find solutions to very large-scale problems far more efficiently than existing methods. This result complements the pioneering work of Friedman, et al. (2007) for the single-task Lasso. As a case study, we use the multi-task Lasso as a variable selector to discover a semantic basis for predicting human neural activation. The learned solution outperforms the standard basis for this task on the majority of test participants, while requiring far fewer assumptions about cognitive neuroscience. We demonstrate how this learned basis can yield insights into how the brain represents the meanings of words.

1. Introduction

The cyclical coordinate descent algorithm has been proposed to solve the ℓ_1-regularized least squares regression (or the Lasso) almost ten years ago (Fu, 1998), but not until very recently was their power fully appreciated (Friedman et al., 2007; Wu & Lange, 2008). In particular, Friedman et al.(2007) show that the coordinate descent method, if implemented appropriately, can be used to evaluate the entire regularization path remarkably faster than almost all the existing state-of-the-art methods. The main reasons for such a surprising performance of the coordinate descent algorithm can be summarized as: (i) During each iteration, the coordinate-wise update can be written as a closed-form soft-thresholding operator, thus an inner loop is avoided; (ii) If the underlying feature vector is very sparse, the soft-thresholding operator can very efficiently detect the zeros by a simple check, thus only a small number of updates are needed. (iii) Many computational tricks, like the *covariance update* or *warm start*, can be easily incorporated into the coordinate descent procedure (Friedman et al., 2008).

In this paper, we consider the computational aspect of the multi-task Lasso (Zhang, 2006), which generalizes the Lasso to the multi-task setting by replacing the ℓ_1-norm regularization with the sum of sup-norm regularization. A scalable cyclical blockwise coordinate descent algorithm is designed which can evaluate the entire regularization path efficiently. In particular, we show that the sub-problem within each iteration can be very efficiently solved by a *Winsorization* operator,[1] i.e. a proportion of the extreme values of the given vector are truncated while the others remain the same. This extends the result of (Friedman et al., 2007) to the multi-task setting. A similar result also appeared in (Fornasier & Rauhut, 2008) under the more general linear inverse problem framework.

The main contribution of this work is that we formulate a non-trivial learning task from the cognitive neuroscience community into a multi-task Lasso problem

Appearing in *Proceedings of the 26th International Conference on Machine Learning*, Montreal, Canada, 2009.

[1] After Charles P. Winsor, whom John Tukey credited with converting him from topology to statistics (Mallows, 1990)

and solve it using the obtained blockwise coordinate descent algorithm. Compared with the most state-of-the-art results (Mitchell et al., 2008), our solution outperforms the standard hand-crafted features in the majority of test participants while using far fewer assumptions. We also discuss how our methods can be used to refine current theory in cognitive neuroscience.

2. The Multi-task Lasso

In this section, we introduce the multi-task Lasso and some related work. We start with some notations. Consider a K-task linear regression, for $k = 1, \ldots, K$, $Y^{(k)} = \sum_{j=1}^{p} \beta_j^{(k)} X_j^{(k)} + \epsilon^{(k)}$ where $Y^{(k)}, X_j^{(k)}, \epsilon^{(k)} \in \mathbf{R}^{n_k}$. The superscript k indexes the tasks, p is the number of features, and n_k is the number of instances within the k-th task. We assume the data is standardized so the constant terms can be dropped, i.e. $X_j^{(k)}$ and $Y^{(k)}$ have mean 0 and $\|X_j^{(k)}\|_2 = 1$ where $\|\cdot\|_2$ is the ℓ_2-Euclidean norm. Let

$$\beta_j \equiv (\beta_j^{(1)}, \ldots, \beta_j^{(K)})^T \tag{1}$$

be the vector of all coefficients for the j^{th} feature across different tasks, the multi-task Lasso estimate is formulated as the solution to the optimization problem

$$\min_{\beta} \left\{ \frac{1}{2} \sum_{k=1}^{K} \|Y^{(k)} - \sum_{j=1}^{p} \beta_j^{(k)} X_j^{(k)}\|_2^2 + \lambda \sum_{j=1}^{p} \|\beta_j\|_\infty \right\}, \tag{2}$$

where $\|\beta_j\|_\infty \equiv \max_k |\beta_j^{(k)}|$ is the sup-norm in the Euclidean space. It has the effect of "grouping" the elements in β_j such that they can achieve zeros simultaneously. If all tasks share a common design matrix, the multi-task regression reduces to a multivariate-response regression. In this case, Turlach et al. (2005) proposes the same sum of sup-norm regularization and name the resulting estimate in (2) as the simultaneous Lasso. It's obvious that any solver for the multi-task Lasso also solves the simultaneous Lasso. Existing methods to solve (2) from the machine learning and statistics communities include the double coordinate descent method from (Zhang, 2006), the interior-point method from (Turlach et al., 2005), and the geometric solution path method from (Zhao et al., 2009). These methods, however, have difficulty scaling to thousands of features and tasks.

One alternative worth noting is the multi-task feature selection work of Argyriou, Evgeniou, and Pontil (2008). Compared with (2), although both methods can be used to learn features over many tasks, their work uses a different penalty term in the optimization problem. Our work, by contrast, focuses on the multi-task Lasso which uses the sum of sup-norm penalty.

Remark 1. Equation (2) treats all tasks equally, which can be sensitive to abnormal or outlier tasks. To address this, we can build an adaptive version of this algorithm. After obtaining the initial estimate by treating all the tasks equally, we could calculate the residual sum of squares for the fit of different tasks. A second step can then be conducted by weighting these tasks differently according to their initial fit. This provides extra performance gain and robustness.

3. Blockwise Coordinate Descent

For a fixed regularization parameter λ, the blockwise coordinate descent algorithm for the multi-task Lasso problem in Equation (2) is given in Figure 1, where $\langle \cdot, \cdot \rangle$ denotes the inner product operator of two vectors. Recall that β_j in (1) represents the coefficient vector of the j-th feature across all the K tasks. We call β_j a block. The algorithm consists of simultaneously updating the coefficients within each block while holding all the others fixed, then cycling through this process. Therefore, if the current estimates are $\widehat{\beta}_\ell$, $\ell = 1, \ldots, p$, then β_j is updated by the following sub-problem:

$$\widehat{\beta}_j = \arg\min_{\beta_j} \left\{ \frac{1}{2} \sum_{k=1}^{K} \left\| R_j^{(k)} - \beta_j^{(k)} X_j^{(k)} \right\|_2^2 + \lambda \|\beta_j\|_\infty \right\} \tag{3}$$

where $R_j^{(k)} \equiv Y^{(k)} - \sum_{\ell \neq j} \widehat{\beta}_\ell^{(k)} X_\ell^{(k)}$ denotes the partial residual vector.

If the regularization parameter $\lambda = 0$, the problem in (3) decouples and the least squares solution $\alpha_j^{(k)}$ is

$$\alpha_j^{(k)} = \langle R_j^{(k)}, X_j^{(k)} \rangle, \quad \text{for } \forall j, k. \tag{4}$$

Since $\langle R_j^{(k)}, X_j^{(k)} \rangle = \langle Y^{(k)}, X_j^{(k)} \rangle - \sum_{\ell \neq j} \beta_\ell^{(k)} \langle X_\ell^{(k)}, X_j^{(k)} \rangle$, we can pre-calculate the quantities $c_j^{(k)} = \langle Y^{(k)}, X_j^{(k)} \rangle$ and $d_{ij}^{(k)} = \langle X_i^{(k)}, X_j^{(k)} \rangle$. This is the same *covariance update* idea as in (Friedman et al., 2008) and corresponds to the first double loop in the algorithm in Figure 1. If we have a decreasing sequence of the regularization parameters λ's, the initial values of $\beta_j^{(k)}$ for each fixed λ comes from the solutions $\widehat{\beta}_j^{(k)}$ calculated from the previous λ value. This is the same *warm start* trick as in (Friedman et al., 2008) and can significantly speedup the algorithm performance for evaluating the entire solution path. Since the quantities $c_j^{(k)}$ and $d_{ij}^{(k)}$ do not depend on λ, they only need to be calculated once and can serve for the whole pathwise evaluation.

For $\lambda > 0$, (3) becomes a sup-norm penalized least squares regression. If we use the Newton's method or coordinate descent procedure to solve it as in (Zhang,

2006), an inner loop will be needed. This turns out not to be scalable if the number of tasks K is very large. Fortunately, Theorem 2 below shows that the solution to (3) is equivalent to a closed-form Winsorization operation applied on the previously calculated unpenalized least squares results $\alpha_j^{(k)}$'s. This corresponds to the main loop of the algorithm in Figure 1.

BLOCKWISE COORDINATE DESCENT ALGORITHM

Input: Data $(X_1^{(k)}, \ldots, X_p^{(k)}, Y^{(k)}), k = 1, \ldots, K$ and the regularization parameter λ;

Iterate over $k \in \{1, \ldots, K\}$ and $j \in \{1, \ldots, p\}$

(1) $c_j^{(k)} \leftarrow \langle Y^{(k)}, X_j^{(k)} \rangle$;

(2) $\beta_j^{(k)} \leftarrow$ initial values (either 0 or $\widehat{\beta}_j^{(k)}$ for the previous λ if doing a warm-start);

(3) For each $i \in \{1, \ldots, p\}$: $d_{ij}^{(k)} \leftarrow \langle X_i^{(k)}, X_j^{(k)} \rangle$;

End;

Iterate until convergence (Main Loop):

For each $j \in \{1, \ldots, p\}$:

(1) $\forall k \in \{1, \ldots, K\}$, $\alpha_j^{(k)} \leftarrow c_j^{(k)} - \sum_{i \neq j} \beta_i^{(k)} d_{ij}^{(k)}$;

(2) If $\sum_{k=1}^K |\alpha_j^{(k)}| \leq \lambda$ Then $\beta_j \leftarrow \mathbf{0}$ Else

(a) Sort the indices according to $|\alpha_j^{(k_1)}| \geq |\alpha_j^{(k_2)}| \geq \ldots \geq |\alpha_j^{(k_K)}|$;

(b) $m^* \leftarrow \arg\max_{1 \leq m \leq K} \left(\sum_{i=1}^m |\alpha_j^{(k_i)}| - \lambda \right) / m$;

(c) For each $i \in \{1, \ldots, K\}$

If $i > m^*$ Then $\beta_j^{(k_i)} \leftarrow \alpha_j^{(k_i)}$ Else

$$\beta_j^{(k_i)} \leftarrow \frac{\text{sign}(\alpha_j^{(k_i)})}{m^*} \left[\sum_{\ell=1}^{m^*} |\alpha_j^{(k_\ell)}| - \lambda \right];$$

End;

Output: $\widehat{\beta}_j^{(k)} \leftarrow \beta_j^{(k)}$ for $j = 1, \ldots, p$ and $k = 1, \ldots, K$;

Figure 1. The algorithm for the multi-task Lasso.

Theorem 2. *Let $\alpha_j^{(k)}$ as defined in (4) and order the indices according to $|\alpha_j^{(k_1)}| \geq |\alpha_j^{(k_2)}| \geq \ldots \geq |\alpha_j^{(k_K)}|$. Then the solution to (3) is*

$$\widehat{\beta}_j^{(k_i)} = \frac{\text{sign}(\alpha_j^{(k_i)})}{m^*} \left[\sum_{i'=1}^{m^*} |\alpha_j^{(k_{i'})}| - \lambda \right]_+ \cdot \mathbf{1}_{\{i \leq m^*\}} + \alpha_j^{(k_i)} \cdot \mathbf{1}_{\{i > m^*\}}$$

where $m^* = \arg\max_m \frac{1}{m} \left(\sum_{i'=1}^m |\alpha_j^{(k_{i'})}| - \lambda \right)$, $\mathbf{1}_{\{\cdot\}}$ is the indicator function, and $[\cdot]_+$ denotes the positive part.

Proof: The proof proceeds by discussing several cases separately: (i) All the elements in the sup-norm are zeros; (ii) One unique element in the sup-norm achieves the maximum; (iii) At least two elements in the sup-norm achieve the maximum. These cases correspond to Propositions 5, 6, and 8 respectively.

Since the given objective function in (3) is convex, its solution can be characterized by the Karush-Kuhn-Tucker conditions as the following

$$\left(R_j^{(k)} - \widehat{\beta}_j^{(k)} X_j^{(k)} \right)^T X_j^{(k)} = \lambda \eta_k \quad \forall k \in \{1, \ldots, K\}, \quad (5)$$

where $\{\eta_k\}_{k=1}^K$ satisfy $\eta \equiv (\eta_1, \ldots, \eta_K)^T \in \partial \| \cdot \|_\infty \big|_{\beta_j}$. Here, $\partial \| \cdot \|_\infty \big|_{\beta_j}$ denotes the subdifferential of the convex functional $\| \cdot \|_\infty$ evaluated at β_j, it lies in a K-dimensional Euclidean space. Next, the following proposition from (Rockafellar & Wets, 1998) can be used to characterize the subdifferential of sup-norms.

Lemma 3. *The subdifferential of $\| \cdot \|_\infty$ in $\mathbf{R}^K$ is*

$$\partial \| \cdot \|_\infty \big|_x = \begin{cases} \{\eta : \|\eta\|_1 \leq 1\} & x = \mathbf{0} \\ \text{conv}\{\text{sign}(x_i) e_i : |x_i| = \|x\|_\infty\} & \text{o.w.} \end{cases} \quad (6)$$

where $\text{conv}(A)$ denotes the convex hull, and e_i is the i-th canonical unit vector in $\mathbf{R}^K$.

Proposition 4. *Let $\widehat{\beta}_j^{(k)}$ be solution to (3) and $\alpha_j^{(k)}$ in (4), if $\widehat{\beta}_j^{(k)} \neq 0$, then $\text{sign}(\widehat{\beta}_j^{(k)}) = \text{sign}(\alpha_j^{(k)})$.*

Proof to Proposition 4: Since $\widehat{\beta}_j^{(k)} \neq 0$, the result trivially follows from the convexity and continuity of the objective function in (3). □

Firstly, we consider the case that $\sum_{k=1}^K |\alpha_j^{(k)}| \leq \lambda$ and show that $\mathbf{0}$ must be a solution.

Proposition 5. *$\widehat{\beta}_j = \mathbf{0}$ if and only if $\sum_{k=1}^K |\alpha_j^{(k)}| \leq \lambda$.*

Proof to Proposition 5: From (5), we know that $\widehat{\beta}_j = \mathbf{0}$ if and only if $\exists \eta_1, \ldots, \eta_K$ such that $\sum_{k=1}^K |\eta_k| \leq 1$ and

$$\lambda \eta_k = R_j^{(k)^T} X_j^{(k)} = \alpha_j^{(k)}. \quad (7)$$

If $\sum_{k=1}^K |\alpha_j^{(k)}| \leq \lambda$, choosing η_k as in (7) would guarantee that $\sum_{k=1}^K |\eta_k| \leq 1$, therefore $\widehat{\beta}_j = \mathbf{0}$.

On the other hand, If $\widehat{\beta}_j = \mathbf{0}$, from (7), we know that $\lambda \eta_k = \alpha_j^{(k)}$, $k = 1, \ldots, K$ and $\sum_{k=1}^K |\eta_k| \leq 1$. This implies that $\sum_{k=1}^K |\alpha_j^{(k)}| \leq \lambda$. □

Next, we consider the case that $\sum_{k=1}^{K} |\alpha_j^{(k)}| > \lambda$ and $|\alpha_j^{(k_1)}| - \lambda > |\alpha_j^{(k_2)}|$. Here we show that $\widehat{\beta}_j^{(k)} = \alpha_j^{(k)}$ for $\forall k \neq k_1$, while $\widehat{\beta}_j^{(k_1)} = \text{sign}\left(\alpha_j^{(k_1)}\right)\left[|\alpha_j^{(k_1)}| - \lambda\right]$.

Proposition 6. *$|\widehat{\beta}_j^{(k_1)}| > |\widehat{\beta}_j^{(k)}|$ for $\forall k \neq k_1$ if and only if $|\alpha_j^{(k_1)}| - \lambda > |\alpha_j^{(k_2)}|$.*

Proof to Proposition 6: If $|\widehat{\beta}_j^{(k_1)}| > |\widehat{\beta}_j^{(k)}|$ for $\forall k \neq k_1$, this implies that $\partial \|\cdot\|_\infty\big|_{\beta_j} = e_{k_1}$, where e_{k_1} is the k_1-th canonical vector. Therefore, from (5), $\left(R_j^{(k)} - \widehat{\beta}_j^{(k)} X_j^{(k)}\right)^T X_j^{(k)} = \lambda \text{sign}(\widehat{\beta}_j^{(k_1)}) \mathbf{1}_{\{k=k_1\}}$.

Therefore, we know $\widehat{\beta}_j^{(k_1)} = \alpha_j^{(k_1)} - \lambda \text{sign}(\widehat{\beta}_j^{(k_1)})$ and $\widehat{\beta}_j^{(k)} = \alpha_j^{(k)}$ for $\forall k \neq k_1$. Combined with the fact $|\widehat{\beta}_j^{(k_1)}| > |\widehat{\beta}_j^{(k)}|$ for $\forall k \neq k_1$, we get $|\alpha_j^{(k_1)} - \lambda \text{sign}(\widehat{\beta}_j^{(k_1)})| > |\alpha_j^{(k)}|$ for $\forall k \neq k_1$.

From Proposition 4, we have $\text{sign}(\alpha_j^{(k_1)}) = \text{sign}(\widehat{\beta}_j^{(k_1)})$. Further, if $\widehat{\beta}_j^{(k_1)} > 0$, then $|\alpha_j^{(k_1)}| > \lambda$, we have $|\alpha_j^{(k_1)} - \lambda \text{sign}(\alpha_j^{(k_1)})| = |\alpha_j^{(k_1)}| - \lambda$. Therefore, $|\alpha_j^{(k_1)}| - \lambda > |\alpha_j^{(k_2)}|$. This is also true for $\widehat{\beta}_j^{(k_1)} < 0$.

On the other hand, assuming $|\alpha_j^{(k_1)}| - \lambda > |\alpha_j^{(k_2)}|$ but for some $n > 1$, there exist

$$|\widehat{\beta}_j^{(k_1)}| = \ldots = |\widehat{\beta}_j^{(k_n)}| = \|\widehat{\beta}_j\|_\infty. \tag{8}$$

Then, by (6) and (5), there must exist $a_1, a_2 \in [0,1]$ and $a_1 + a_2 \leq 1$, for $h = 1, 2$, $\left(R_j^{(k_h)} - \widehat{\beta}_j^{(k_h)} X_j^{(k_h)}\right)^T X_j^{(k_h)} = \lambda a_h \text{sign}(\widehat{\beta}_j^{(k_h)})$.

Combine this result and (8), we get $|\alpha_j^{(k_1)} - \lambda a_1 \text{sign}(\widehat{\beta}_j^{(k_1)})| = |\alpha_j^{(k_2)} - \lambda a_2 \text{sign}(\widehat{\beta}_j^{(k_2)})|$. By Proposition 4 and $|\alpha_j^{(k_1)}| > \lambda a_1$, we have $|\alpha_j^{(k_1)} - \lambda a_1 \text{sign}(\widehat{\beta}_j^{(k_1)})| = |\alpha_j^{(k_1)}| - \lambda a_1$. If $|\alpha_j^{(k_2)}| > \lambda a_2$, we get $|\alpha_j^{(k_1)}| - \lambda\left(\text{sign}(a_1 \widehat{\beta}_j^{(k_1)}) + a_2 \text{sign}(\widehat{\beta}_j^{(k_2)})\right) = |\alpha_j^{(k_2)}|$. Since $a_1 + a_2 \leq 1$, this obviously contradicts with the assumption that $|\alpha_j^{(k_1)}| - \lambda > |\alpha_j^{(k_2)}|$. The same result also hold for the case $|\alpha_j^{(k_2)}| \leq \lambda a_2$. □

Lastly, for the case $\sum_{k=1}^{K} |\alpha_j^{(k)}| > \lambda$ and $|\alpha_j^{(k_1)}| - \lambda \leq |\alpha_j^{(k_2)}|$. We start with an auxiliary proposition.

Proposition 7. *For $m > 1$, if there are precisely m entries $|\widehat{\beta}_j^{(k_1)}|, \ldots, |\widehat{\beta}_j^{(k_m)}|$ achieve $\|\widehat{\beta}_j\|_\infty > 0$, then*

$$\widehat{\beta}_j^{(k_i)} = \frac{\text{sign}(\alpha_j^{(k_i)})}{m}\left[\sum_{\ell=1}^{m} |\alpha_j^{(k_\ell)}| - \lambda\right] \quad \forall i = 1, \ldots, m.$$

Proof to Proposition 7: Since exactly m entries $|\widehat{\beta}_j^{(k_1)}|, \ldots, |\widehat{\beta}_j^{(k_m)}|$ achieve $\|\widehat{\beta}_j\|_\infty > 0$, by (6), there must exist m nonnegative numbers $a_1, \ldots, a_m$, such that $\sum_{\ell=1}^{m} a_\ell = 1$ and for each $\ell \in \{1, \ldots, m\}$, $\left(R_j^{(k_\ell)} - \widehat{\beta}_j^{(k_\ell)} X_j^{(k_\ell)}\right)^T X_j^{(k_\ell)} = \lambda a_\ell \text{sign}(\widehat{\beta}_j^{(k_\ell)})$.

Which can be re-written as

$$\alpha_j^{(k_\ell)} = \lambda a_\ell \text{sign}(\widehat{\beta}_j^{(k_\ell)}) + \widehat{\beta}_j^{(k_\ell)} \quad \forall \ell \in \{1, \ldots, m\}. \tag{9}$$

Using the fact that $|\widehat{\beta}_j^{(k_1)}| = \ldots = |\widehat{\beta}_j^{(k_m)}|$, summing over the absolute value of both sides of all the equations in (9), we obtain $\sum_{\ell=1}^{m} |\alpha_j^{(k_\ell)}| = \sum_{\ell=1}^{m} |\lambda a_\ell \text{sign}(\widehat{\beta}_j^{(k_\ell)}) + \widehat{\beta}_j^{(k_\ell)}|$. Since $|\lambda a_\ell \text{sign}(\widehat{\beta}_j^{(k_\ell)}) + \widehat{\beta}_j^{(k_\ell)}| = \lambda a_\ell + |\widehat{\beta}_j^{(k_\ell)}|$ and $\sum_{\ell=1}^{m} a_\ell = 1$, we have

$$|\widehat{\beta}_j^{(k_i)}| = \frac{1}{m}\left[\sum_{\ell=1}^{m} |\alpha_j^{(k_i)}| - \lambda\right]. \quad \forall i \in \{1, \ldots, m\}. \tag{10}$$

Finally, by Proposition 4, we know that $\text{sign}(\alpha_j^{(k_i)}) = \text{sign}(\widehat{\beta}_j^{(k_i)})$ for $i = 1, \ldots, m$, therefore

$$\widehat{\beta}_j^{(k_i)} = \frac{\text{sign}(\alpha_j^{(k_i)})}{m}\left[\sum_{\ell=1}^{m} |\alpha_j^{(k_\ell)}| - \lambda\right] \quad i = 1, \ldots, m. \quad \square$$

To finish the proof of Theorem 2, we still need to describe the exact conditions that there are exactly $m > 1$ elements that achieve the sup-norm. This is given in the following Proposition 8. A similar result was also given in (Fornasier & Rauhut, 2008).

Proposition 8. *For $m > 1$, there exist precisely m entries $|\widehat{\beta}_j^{(k_1)}|, \ldots, |\widehat{\beta}_j^{(k_m)}|$ that achieve $\|\widehat{\beta}_j\|_\infty > 0$ if and only if $|\alpha_j^{(k_1)}| - \lambda \leq |\alpha_j^{(k_2)}|$ and $|\alpha_j^{(k_m)}| \geq \frac{1}{m-1}\left(\sum_{\ell=1}^{m-1} |\alpha_j^{(k_\ell)}| - \lambda\right)$ and $|\alpha_j^{(k_{m+1})}| < \frac{1}{m}\left(\sum_{\ell=1}^{m} |\alpha_j^{(k_\ell)}| - \lambda\right)$.*

Proof to Proposition 8: Assume exactly $m > 1$ entries $|\widehat{\beta}_j^{(k_1)}|, \ldots, |\widehat{\beta}_j^{(k_m)}|$ achieve $\|\widehat{\beta}_j\|_\infty > 0$, from Proposition 7, we know that for $i = 1, \ldots, m$,

$$\widehat{\beta}_j^{(k_i)} = \frac{\text{sign}(\alpha_j^{(k_i)})}{m}\left[\sum_{\ell=1}^{m} |\alpha_j^{(k_\ell)}| - \lambda\right]. \tag{11}$$

By (9) and Proposition 4, we have

$$a_\ell = \frac{\alpha_j^{(k_\ell)} - \widehat{\beta}_j^{(k_\ell)}}{\lambda \text{sign}(\widehat{\beta}_j^{(k_\ell)})} = \frac{|\alpha_j^{(k_\ell)}| - |\widehat{\beta}_j^{(k_\ell)}|}{\lambda} \quad \ell \in \{1, \ldots, m\}.$$

Plugging (11) into a_m, since $a_m \geq 0$, we get

$$|\alpha_j^{(k_m)}| \geq \frac{1}{m-1}\left(\sum_{\ell=1}^{m-1}|\alpha_j^{(k_\ell)}| - \lambda\right). \quad (12)$$

Further, since $|\widehat{\beta}_j^{(k_m)}| > |\widehat{\beta}_j^{(k_{m+1})}|$, the necessity follows from $|\alpha_j^{(k_{m+1})}| = |\widehat{\beta}_j^{(k_{m+1})}| < |\widehat{\beta}_j^{(k_m)}| = \frac{1}{m}\left[\sum_{\ell=1}^{m}|\alpha_j^{(k_\ell)}| - \lambda\right]$.

For the sufficiency, we assume that precisely $n \neq m$ entries $|\widehat{\beta}_j^{(k_1)}|, \ldots, |\widehat{\beta}_j^{(k_n)}|$ achieve $\|\widehat{\beta}_j\|_\infty > 0$, then follow exactly the same argument as the necessity part to obtain a contradiction. □

To prove Theorem 2, we know from Proposition 8 there are precisely m^* entries in $\widehat{\beta}_j$ that achieve $\|\widehat{\beta}_j\|_\infty > 0$ if and only if $m^* = \arg\max_m \frac{1}{m}\left(\sum_{\ell=1}^{m}|\alpha_j^{(k_\ell)}| - \lambda\right)$. The result follows by combining Propositions 5 and 6.

Remark 9. We also conducted experiments to quantitatively compare the performance of the blockwise coordinate descent algorithm with the Log-barrier interior-point method in a similar setting as in (Friedman et al., 2007). The blockwise coordinate descent algorithm is significantly faster. Due to the lack of space, we do not report the simulation details here.

The complexity analysis of the algorithm is straightforward. Within the main loop, the most expensive step is sorting the K elements, which can be done in $O(K \log K)$. This makes the algorithm scalable to very large number of tasks. From the Winsorization operator, we do not need to update a block if $\sum_{k=1}^{K}|\alpha_j^{(k)}| \leq \lambda$, which happens frequently if the problem is sparse. This makes the algorithm scalable to very large number of features. Furthermore, since many quantities can be pre-calculated, the algorithm can be also applied to large sample datasets. The numerical convergence of this algorithm is summarized in the following theorem.

Theorem 10. *The solution sequence generated by the blockwise coordinate descent algorithm in Figure 1 is bounded and every cluster point is a solution of the multi-task Lasso defined in* (2)*.*

Proof The optimization objective function in (2) is continuous on a compact level set, and is convex (but not strictly convex) and nondifferentiable. Furthermore, notice that the nondifferentiable part $\lambda\sum_{j=1}^{p}\|\beta_j\|_\infty$ is separable, i.e. it can be decomposed into a sum of individual functions, one for each block of variables. By Theorem 4.1 in (Tseng, 2001) we obtain the claimed results. □

4. Neural Semantic Basis Discovery

In this section, we present a case study of the multi-task Lasso by applying it to a problem in cognitive neuroscience. Specifically, we consider the task of predicting a person's neural activity in response to an arbitrary word in English as described in (Mitchell et al., 2008). Their approach is to predict the neural image that would be recorded using *functional magnetic resonance imaging* (fMRI) when a person thinks about a given word. In their work, the semantic meaning of a word is encoded by co-occurrence statistics with other words in a very large text corpus. They use a small number of training words to learn a linear model that maps these co-occurrence statistics to images of neural activity recorded while a person is thinking about those words. Their model can then predict images for new words that were not included in the training set and shows that the predicted images are similar to the observed images for those words.

Table 1. The semantic basis used in Mitchell et al. (2008)

See	Eat	Run	Say	Enter
Hear	Touch	Push	Fear	Drive
Listen	Rub	Fill	Open	Wear
Taste	Approach	Move	Lift	Break
Smell	Manipulate	Ride	Near	Clean

In their initial model each word is encoded by a vector of co-occurrences with 25 sensory-action verbs (e.g. eat, ride, wear). For example, words related to foods such as "apples" and "oranges" would have frequent co-occurrences with the word "eat" but few co-occurrences with the word "wear". Conversely, words related to clothes such as "shirt" or "dress" would co-occur frequently with the word "wear" but not the word "eat". Thus "eat" and "wear" are example *basis words* used to encode relationships of a broad set of other words. These 25 sensory-motor verbs (shown in Table 1) were hand-crafted based on domain knowledge from the cognitive neuroscience literature and are considered a *semantic basis* of latent word meaning. A natural question is: *What is the optimal basis of words to represent semantic meaning across many concepts?*

Rather than relying on models that require manual selection of a set of words, our research tries to build models that will perform *variable selection* to automatically learn a semantic basis of word meaning. In this way, we not only want to predict neural activity well, but also give insights into how the brain represents the meaning of different concepts. *The hope is that learning directly from data could lead to new theories in cognitive neuroscience.*

For our study, we utilize the two datasets described in (Mitchell et al., 2008). The first dataset was collected using fMRI. Nine participants were presented with 60 different words and were asked to think about each word for several seconds while their neural activities were recorded. The 60 words are composed of nouns from 12 categories with 5 exemplars per category. For example, a *bodypart* category includes Arm, Eye, Foot, Hand, Leg, a *tools* category includes the words Chisel, Hammer, Pliers, Saw, Screwdriver, and a *furniture* category includes Bed, Chair, Dresser, Desk, Table, etc.

The second dataset is a symmetric matrix of text co-occurrences between the 50,000 most frequent words in English. These co-occurrences are derived from the Google Trillion Word Corpus[2]. The goal is to use these co-occurrences to construct a feature vector that represents word meaning. The hope is that two words that cause similar neural activites would also have high inner product between their co-occurrence vectors. One simple representation of each word is to use the raw 50,000 dimensional feature vector of co-occurrences (normalized to unit length row norm). Typically a much smaller representation is desired such as the hand-crafted 25-word basis described above.

For each participant, there are altogether $n = 60$ fMRI images taken[3], each one corresponds to one stimulus word. A typical fMRI image contains over $K = 20,000$ different neural responses. Each neural response is the activity in a *voxel* (volume-element) in the brain. Each voxel represents a 1-3 mm^3 volume in the brain and is the basic spatial unit of measurement in fMRI. Here we show the problem of learning a semantic basis can be formulated into a multi-task Lasso as in (2). Since the goal of learning a semantic basis is to find a common set of predictor variables that will predict the neural response well across multiple voxels, where each predictor variable is the text co-occurences with a particular word from the Google Trillion Word Corpus. Therefore, for each participant, we have roughly $K = 20,000$ tasks, all these tasks share the common design columns $\{X_j\}_{j=1}^{p} \in \mathbf{R}^n$, representing the co-occurrences of $n = 60$ training words with $p = 50,000$ other English words in the Google Corpus. The response vector $Y^{(k)}$ for each task contains the neural activations for a single voxel (volume-element) across the $n = 60$ fMRI images. Therefore, this is a multi-task learning problem with a very large number of tasks $K = 20,000$ and a very large number of features $p = 50,000$. While the algorithm we develop can solve a problem of this scale in only a few minutes, our primary results use a smaller dataset where $p = 5,000$ and $K = 500$, so that our experiments are directly comparable with other published results. We also provide additional results where $K = 4500$.

[2]http://googleresearch.blogspot.com/2006/08/all-our-n-gram-are-belong-to-you.html

[3]Each image is actually the average of 6 different recordings of each word.

5. Experimental Results

To evaluate our methods and compare them to existing results, we use exactly the same experimental protocols described in (Mitchell et al., 2008). As a small additional step we use the multi-task Lasso to perform a variable selection. Note that the multi-task Lasso is used in this context only to learn *which* variables should be input into the final model. Specifically, the *leave-two-out-cross-validation* procedure is as follows:

a Create a $60 \times 5,000$ design matrix of semantic features using co-occurences of the 5,000 most frequent words in English (minus 100 stop words). A stop word is a high frequency common word like "the".

b Select 2 words out of 60 for testing and use the other 58 words for training. Using (2), learn the coefficients β by setting each X_j to be the 58×1 vector of co-occurences for each of the 5000 basis words and each $Y^{(k)}$ to be the 58×1 column vector for each of the top $K = 500$ voxel responses selected using the stability criterion score described in (Mitchell et al., 2008). In the language of multi-task Lasso, this problem corresponds to the scale $n = 58, p = 5000, K = 500$. The regularization parameter here can be set to choose the desired number of non-zero coefficients. We set this parameter to yield basis sets with 25, 140, and 350 elements so the model is easier to interpret and compare to existing results.

c Create a new matrix of semantic features that is $58 \times d$, where d is the number of selected feature blocks in β. In our case d will be either 25, 140, or 350. Each non-zero block should correspond to a word selected from the original set of 5,000. This word shall now become a semantic feature in the new basis.

d Train a linear model using ridge regularization to predict each of the 500 voxels from the semantic feature basis. The solution can be obtained using the standard normal equations for ridge regression.

e For each of the two test examples, predict the neural response of the 500 selected voxels. Compute the cosine similarity of each prediction with each of the held out images. Based on the combined similarity scores, choose which prediction goes with each held out image. Test if the joint labeling was correct. This leads to an output of 0 or 1. For more details, see (Mitchell et al., 2008).

f Repeat steps b-e for all $\binom{60}{2}$ possible pairs of words (1,770 total). Count the number of incorrect labelings in step e to determine the accuracy of the basis set.

Figure 2. The leave-two-out-cross-validation protocols

Table 2. Accuracies for 9 fMRI Participants. Learned *per-subject*

	Subject 1	Subject 2	Subject 3	Subject 4	Subject 5	Subject 6	Subject 7	Subject 8	Subject 9
Handcraft	0.749718	0.705085	0.726554	0.715254	0.792655	0.771751	0.684746	0.729379	0.787006
MTL25	0.863842	0.713559	0.718079	0.608475	0.787006	0.649153	0.714124	0.730508	0.679661
MTL140	0.863842	0.741243	0.727119	0.545763	0.831638	0.654237	0.733333	0.751977	0.717514
MTL350	0.881921	0.757627	0.754802	0.567232	0.840678	0.69209	0.762147	0.784746	0.738983

Table 3. Accuracies for 9 fMRI Participants. Learned with *combined fMRI*

	Subject 1	Subject 2	Subject 3	Subject 4	Subject 5	Subject 6	Subject 7	Subject 8	Subject 9
MTL25	0.815254	0.718079	0.679096	0.588701	0.732203	0.661017	0.737288	0.755932	0.690960
MTL140	0.849718	0.736158	0.733333	0.553107	0.810734	0.678531	0.761017	0.753107	0.732768
MTL350	0.880791	0.761017	0.758757	0.575141	0.845198	0.691525	0.762712	0.783051	0.738983

We repeat this experiment for each of the nine different participants in the fMRI study and use the same method in Mitchell et al. (2008) to ensure consistency while testing various semantic features. The regularization parameter for the ridge regression in step d is chosen automatically using the cross-validation error score described in (Hastie et al., 2001, Page 216).

We performed several experiments using the above protocols to compare the performance of the semantic feature sets learned using the multi-task Lasso with the hand-crafted features described earlier. Using these results we now pose and answer several questions:

Q1: Can we automatically learn a semantic basis?

As in Figure 2, we use the multi-task Lasso to learn a semantic basis from the 5,000 most frequent words in English and adjust the regularization parameter to obtain different basis sizes. We denote MTL25, MTL140, MTL350 as models that have 25,140, and 350 words respectively in their basis sets each time they were trained. Note that we train the models on each iteration of the cross-validation and keep the regularization parameter the same between iterations. As a result, there can be a slight variation in the actual number of features selected on each iteration.

We see in Table 2 the results of the 4 models. The result for the 25 features hand-crafted by cognitive neuroscientists is also reported. Chance accuracy for this prediction task is 0.5 and statistically significant accuracy at the 5% level is 0.61 (Mitchell et al., 2008). We see that in 8 of 9 subjects, all three multi-task Lasso models learn a semantic basis that leads to statistically significant accuracies. This suggests that we can indeed learn a semantic basis directly from data.

The MTL140 and MTL350 models achieve higher accuracies than the hand-crafted features in 6 of 9 participants. It is exciting that the model can often meet or exceed the performance of the hand-crafted features *using far fewer assumptions about neuroscience*. It is also useful that our learned basis is different than the handcrafted features, which suggests potential benefits from a model averaging approach. For the MTL25 model, we report accuracies higher than the handcrafted features in 4 of 9 participants. We also see that two larger basis sets outperform the MTL25 set, suggesting that more than 25 features are necessary to capture the variance of the data.

An interesting observation comes from participant 4. On this participant all three learned models performed worse than the hand-crafted model. The very abnormal behavior of the learned models on this participant versus the other participants suggests that this particular participant might be an outlier or a very noisy observation (as is common in fMRI studies). However, the hand-crafted feature does not suffer from this case. More investigation is suggested to study this.

Q2: Is there any advantage to learning the basis across multiple subjects simultaneously?

An interesting question is whether we could ameliorate this problem by learning the basis simultaneously across all subjects at once. Table 3 shows the results of learning the basis by combining the fMRI data for all participants (thereby yielding a 58 x 4500) target matrix. This corresponds to a multi-task Lasso where $K = 4500$. On average, we found a slight advantage on the MTL140 and MTL350 models, and slight disadvantage for the MT25 model. Although encouraging, this is hardly conclusive evidence, and we feel this is an interesting direction for future work. In particular, it would be worth studying the impact of noisy data by removing an outlier such as participant 4.

Q3: What is the learned semantic basis?

Table 4 shows one example of 25 basis words learned using the MTL25. It is easy to see relationships between many of the words in the basis set and the 60 stimulus words described before. For example, the model learned Tools as a basis word, which is interest-

ing because 5 different instances of tools were shown (e.g. Screwdriver, Hammer, etc.). The basis word Bedroom clearly refers to words in the furniture category (Bed, Dresser, etc.) and Arms is related to body parts (Leg, Hand, etc.).

Table 4. An example of 25 learned semantic basis words.

Tools	Car	Dog	Wine	Pottery
Model	Station	Bedroom	Breakfast	Cup
Mad	Rentals	Fishing	Cake	Tip
Arms	Walk	Cleaning	Cheese	Gay
Right	White	Front	Contents	Result

For a given basis word, we can train a simple linear model to predict neural activations across all 20,000 voxels in the brain from this single basis word. Note that this is a multiple output regression and each learned coefficient corresponds to a physical location in the brain. By plotting the coefficients, we can discover how different basis words activate different regions of the brain. Figure 3 shows a 3D map of these coefficients for the basis word Tools. We see strong activation (red) in the superior temporal sulcus which is believed to be associated with the perception of biological motion. We also see strong activation in the postcentral gyrus which is believed to be associated with premotor planning.

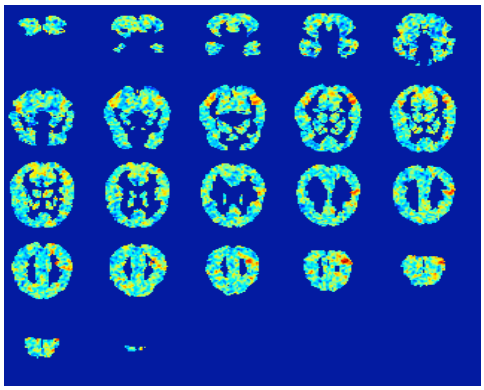

Figure 3. Regression weights to each voxel for word Tools.

6. Conclusion

We present a blockwise coordinate descent algorithm to fit the entire regularization path of the multi-task Lasso in a highly efficient way. This algorithm uses a closed-form Winsorization operator which allows it easy to implement and perform far more efficiently than prevous methods. We believe that the multi-task Lasso is useful for a large class of sparse, multi-task regression problems and demonstrated its power on a neural semantics discovery problem.

References

Argyriou, A., Evgeniou, T., & Pontil, M. (2008). Convex multi-task feature learning. *Machine Learning*, *73*, 243–272.

Fornasier, M., & Rauhut, H. (2008). Recovery algorithms for vector valued data with joint sparsity constraints. *SIAM Journal of Numerical Analysis*, *46*, 577–613.

Friedman, J., Hastie, T., Hüdotofling, H., & Tibshirani, R. (2007). Pathwise coordinate optimization. *The Annals of Applied Statistics*, *1*, 302–332.

Friedman, J. H., Hastie, T., & Tibshirani, R. (2008). Regularized paths for generalized linear models via coordinate descent. *Technical report, Stanford University.*

Fu, W. J. (1998). Penalized regressions: The bridge versus the lasso. *Journal of Computational and Graphical Statistics*, *7*, 397–416.

Hastie, T., Tibshirani, R., & Friedman, J. H. (2001). *The elements of statistical learning: Data mining, inference, and prediction.* Springer-Verlag.

Mallows, C. L. (Ed.). (1990). *The collected works of john w. tukey. volume vi: More mathematical, 1938–1984.* Wadsworth & Brooks/Cole.

Mitchell, T., et al. (2008). Predicting human brain activity associated with the meanings of nouns. *Science*, *320*, 1191–1195.

Rockafellar, R. T., & Wets, R. J.-B. (1998). *Variational analysis.* Springer-Verlag Inc.

Tseng, P. (2001). Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of optimization theory and applications*, *109*, 475–494.

Turlach, B., Venables, W. N., & Wright, S. J. (2005). Simultaneous variable selection. *Technometrics*, *27*, 349–363.

Wu, T. T., & Lange, K. (2008). Coordinate descent algorithms for lasso penalized regression. *The Annals of Applied Statistics*, *2*, 224–244.

Zhang, J. (2006). *A probabilistic framework for multi-task learning* (Technical Report CMU-LTI-06-006). Ph.D. thesis, Carnegie Mellon University.

Zhao, P., Rocha, G., & Yu, B. (2009). The grouped and hierarchical model selection through composite absolute penalties. *The Annals of Statistics (to appear).*