ELLA: An Efficient Lifelong Learning Algorithm

Paul Ruvolo PRUVOLO@CS.BRYNMAWR.EDU
Eric Eaton EEATON@CS.BRYNMAWR.EDU
Bryn Mawr College, Computer Science Department, 101 North Merion Avenue, Bryn Mawr, PA 19010 USA

Abstract

The problem of learning multiple consecutive tasks, known as *lifelong learning*, is of great importance to the creation of intelligent, general-purpose, and flexible machines. In this paper, we develop a method for online multi-task learning in the lifelong learning setting. The proposed Efficient Lifelong Learning Algorithm (ELLA) maintains a sparsely shared basis for all task models, transfers knowledge from the basis to learn each new task, and refines the basis over time to maximize performance across all tasks. We show that ELLA has strong connections to both online dictionary learning for sparse coding and state-of-the-art batch multi-task learning methods, and provide robust theoretical performance guarantees. We show empirically that ELLA yields nearly identical performance to batch multi-task learning while learning tasks sequentially in three orders of magnitude (over 1,000x) less time.

1. Introduction

Versatile learning systems must be capable of efficiently and continually acquiring knowledge over a series of prediction tasks. In such a lifelong learning setting, the agent receives tasks sequentially. At any time, the agent may be asked to solve a problem from any previous task, and so must maximize its performance across all learned tasks at each step. When the solutions to these tasks are related through some underlying structure, the agent may share knowledge between tasks to improve learning performance, as explored in both transfer and multi-task learning.

Despite this commonality, current algorithms for transfer and multi-task learning are insufficient for lifelong learning. Transfer learning focuses on efficiently modeling a new target task by leveraging solutions to previously learned source tasks, without considering potential improvements to the source task models. In contrast, multi-task learning (MTL) focuses on maximizing performance across *all* tasks through shared knowledge, at potentially high computational cost. Lifelong learning includes elements of both paradigms, focusing on efficiently learning each consecutive task by building upon previous knowledge while optimizing performance across all tasks. In particular, lifelong learning incorporates the notion of *reverse transfer*, in which learning subsequent tasks can improve the performance of previously learned task models. Lifelong learning could also be considered as online MTL.

In this paper, we develop an Efficient Lifelong Learning Algorithm (ELLA) that incorporates aspects of both transfer and multi-task learning. ELLA learns and maintains a library of latent model components as a shared basis for all task models, supporting soft task grouping and overlap (Kumar & Daumé III, 2012). As each new task arrives, ELLA transfers knowledge through the shared basis to learn the new model, and refines the basis with knowledge from the new task. By refining the basis over time, newly acquired knowledge is integrated into existing basis vectors, thereby improving previously learned task models. This process is computationally efficient, and we provide robust theoretical guarantees on ELLA's performance and convergence. We evaluate ELLA on three challenging multi-task data sets: land mine detection, facial expression recognition, and student exam score prediction. Our results show that ELLA achieves nearly identical performance to batch MTL with three orders of magnitude (over 1,000x) speedup in learning time. We also compare ELLA to a current method for online MTL (Saha et al., 2011), and find that ELLA has both lower computational cost and higher performance.

2. Related Work

Early work on lifelong learning focused on sharing distance metrics using task clustering (Thrun & O'Sullivan, 1996), and transferring invariances in neu-

Proceedings of the 30th International Conference on Machine Learning, Atlanta, Georgia, USA, 2013. JMLR: W&CP volume 28. Copyright 2013 by the author(s).

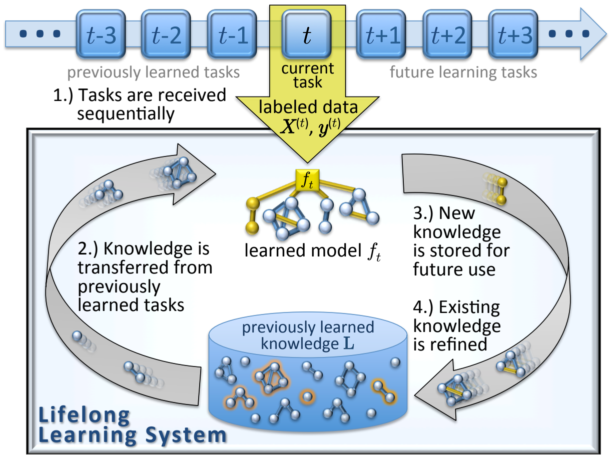
ral networks (Thrun, 1996). Lifelong learning has also been explored for reinforcement learning (Ring, 1997; Sutton et al., 2007) and learning by reading (Carlson et al., 2010). In contrast, ELLA is a general algorithm that supports different base learners to learn continually, framed in the context of current MTL methods.

Recently, MTL research has considered the use of a shared basis for all task models to improve learning over a set of tasks. Several formulations of this idea have been proposed, including a probabilistic framework (Zhang et al., 2008) and a non-parametric Bayesian method that automatically selects the number of bases (Rai & Daumé III, 2010). These methods assume that each model is represented as a parameter vector that is a linear combination of these bases. By using a common basis, these approaches share information between learning tasks and account for task relatedness as the models are learned in tandem with the basis. The GO-MTL algorithm (Kumar & Daumé III, 2012) also uses a sparsely shared basis for multi-task learning, with the advantage that it automatically learns (potentially) overlapping groups of tasks to maximize knowledge transfer. We employ this rich model of underlying task structure as the starting point for developing ELLA.

Few papers have focused on the development of very computationally efficient methods for MTL. Simm et al. (2011) present a model for learning multiple tasks that is efficient in the case when the number of tasks is very large. However, their approach suffers from significant drawbacks in comparison with ELLA: (1) their approach is not an online algorithm, limiting its use in the lifelong learning setting, and (2) their underlying model of shared task structure is significantly less flexible than our model. Another approach, OMTL by Saha et al. (2011), is designed to provide efficient performance when instances and new tasks arrive incrementally. However, OMTL is only applicable to classification tasks (not regression) and relies on perceptron learning, which we found to perform poorly in comparison to other base learners (see Section 4).

3. Approach

We begin by describing the lifelong learning problem and why lifelong learning algorithms must be able to efficiently learn new tasks and incorporate new training data from previous tasks. Then, we introduce ELLA, which efficiently handles both of these operations through a two-stage optimization procedure. We show that ELLA encompasses the problem of online dictionary learning for sparse coding as a special case, and finish by proving robust convergence guarantees.



t-3
t-2
t-1
t
t+1
t+2
t+3
previously learned tasks
current task
future learning tasks
1.) Tasks are received sequentially
labeled data $X^{(t)}, y^{(t)}$
f_t
learned model f_t
2.) Knowledge is transferred from previously learned tasks
3.) New knowledge is stored for future use
4.) Existing knowledge is refined
previously learned knowledge L
Lifelong Learning System

Figure 1. An illustration of the lifelong learning process.

This paper uses the following conventions: matrices are denoted by bold uppercase letters, vectors are denoted by bold lowercase letters, scalars are denoted by normal lowercase letters, and sets are denoted using script typeface (e.g., $\mathcal{A}$). Parenthetical superscripts denote quantities related to a particular task (e.g., matrix $\mathbf{A}^{(t)}$ and vector $\mathbf{v}^{(t)}$ are related to task t).

3.1. The Lifelong Learning Problem

A lifelong learning agent (Figure 1) faces a series of supervised learning tasks $\mathcal{Z}^{(1)}, \mathcal{Z}^{(2)}, \ldots, \mathcal{Z}^{(T_{\max})}$, where each task $\mathcal{Z}^{(t)} = \left(\hat{f}^{(t)}, \mathbf{X}^{(t)}, \mathbf{y}^{(t)}\right)$ is defined by a (hidden) mapping $\hat{f}^{(t)} : \mathcal{X}^{(t)} \mapsto \mathcal{Y}^{(t)}$ from an instance space $\mathcal{X}^{(t)} \subseteq \mathbb{R}^d$ to a set of labels $\mathcal{Y}^{(t)}$ (typically $\mathcal{Y}^{(t)} = \{-1, +1\}$ for classification tasks and $\mathcal{Y}^{(t)} = \mathbb{R}$ for regression tasks). Each task t has n_t training instances $\mathbf{X}^{(t)} \in \mathbb{R}^{d \times n_t}$ with corresponding labels $\mathbf{y}^{(t)} \in {\mathcal{Y}^{(t)}}^{n_t}$ given by $\hat{f}^{(t)}$. We assume that *a priori* the learner does not know the total number of tasks $T_{\max}$, the distribution of these tasks, or their order.

Each time step, the agent receives a batch of labeled training data for some task t, either a new task or a previously learned task. Let T denote the number of tasks encountered so far. After receiving each batch of data, the agent may be asked to make predictions on instances of any previous task. Its goal is to construct task models $f^{(1)}, \ldots, f^{(T)}$ where each $f^{(t)} : \mathbb{R}^d \mapsto \mathcal{Y}^{(t)}$ such that: (1) each $f^{(t)}$ will approximate $\hat{f}^{(t)}$ to enable the accurate prediction of labels for new instances, (2) each $f^{(t)}$ can be rapidly updated as the agent encounters additional training data for known tasks, and (3) new $f^{(t)}$'s can be added efficiently as the agent encounters new tasks. We assume that the total numbers of tasks $T_{\max}$ and data instances $\sum_{t=1}^{T_{\max}} n_t$ will be large, and so a lifelong learning algorithm must have a computational complexity to update the models that scales favorably with both quantities.

3.2. Task Structure Model for ELLA

ELLA takes a parametric approach to lifelong learning in which the prediction function $f^{(t)}(\mathbf{x}) = f(\mathbf{x}; \boldsymbol{\theta}^{(t)})$ for each task t is governed by the task-specific parameter vector $\boldsymbol{\theta}^{(t)} \in \mathbb{R}^d$. To model the relationships between tasks, we assume that the parameter vectors can be represented using a linear combination of shared latent model components from a knowledge repository. Many recent MTL methods employ this same technique of using a shared basis as a means to transfer knowledge between learning problems (see Section 2).

Our model of latent task structure is based on the GO-MTL model proposed by Kumar & Daumé III (2012). ELLA maintains a library of k latent model components $\mathbf{L} \in \mathbb{R}^{d \times k}$ shared between tasks. Each task parameter vector $\boldsymbol{\theta}^{(t)}$ can be represented as a linear combination of the columns of $\mathbf{L}$ according to the weight vector $\mathbf{s}^{(t)} \in \mathbb{R}^k$ (i.e., $\boldsymbol{\theta}^{(t)} = \mathbf{L}\mathbf{s}^{(t)}$). We encourage the $\mathbf{s}^{(t)}$'s to be sparse (i.e., use few latent components) in order to ensure that each learned model component captures a maximal reusable chunk of knowledge.

Given the labeled training data for each task, we optimize the models to minimize the predictive loss over all tasks while encouraging the models to share structure. This problem is realized by the objective function:

$$e_T(\mathbf{L}) = \frac{1}{T}\sum_{t=1}^{T} \min_{\mathbf{s}^{(t)}} \left\{ \frac{1}{n_t}\sum_{i=1}^{n_t} \mathcal{L}\left(f\left(\mathbf{x}_i^{(t)}; \mathbf{L}\mathbf{s}^{(t)}\right), y_i^{(t)}\right) + \mu \|\mathbf{s}^{(t)}\|_1 \right\} + \lambda \|\mathbf{L}\|_{\mathsf{F}}^2 \ , \tag{1}$$

where $(\mathbf{x}_i^{(t)}, y_i^{(t)})$ is the ith labeled training instance for task t, $\mathcal{L}$ is a known loss function, and the L_1 norm of $\mathbf{s}^{(t)}$ is used as a convex approximation to the true vector sparsity. This is similar to the model used in GO-MTL, with the modification that we average the model losses on the training data across tasks (giving rise to the $\frac{1}{T}$ term). This modification is crucial for obtaining the convergence guarantees in Section 3.6.

Since Equation 1 is not jointly convex in $\mathbf{L}$ and the $\mathbf{s}^{(t)}$'s, our goal will be to develop a procedure to arrive at a local optimum of the objective function. A common approach for computing a local optimum for objective functions of this type is to alternately perform two convex optimization steps: one in which $\mathbf{L}$ is optimized while holding the $\mathbf{s}^{(t)}$'s fixed, and another in which the $\mathbf{s}^{(t)}$'s are optimized while holding $\mathbf{L}$ fixed. These two steps are then repeated until convergence (this is the approach employed for model optimization in GO-MTL). Next, we discuss two reasons why this approach is inefficient and thus inapplicable to lifelong learning with many tasks and data instances.

The first inefficiency arises due to the explicit dependence of Equation 1 on *all* of the previous training data (through the inner summation). We remove this inefficiency by approximating Equation 1 using the second-order Taylor expansion of $\frac{1}{n_t}\sum_{i=1}^{n_t} \mathcal{L}\left(f\left(\mathbf{x}_i^{(t)}; \boldsymbol{\theta}\right), y_i^{(t)}\right)$ around $\boldsymbol{\theta} = \boldsymbol{\theta}^{(t)}$, where $\boldsymbol{\theta}^{(t)} = \arg\min_{\theta} \frac{1}{n_t}\sum_{i=1}^{n_t} \mathcal{L}\left(f\left(\mathbf{x}_i^{(t)}; \boldsymbol{\theta}, y_i^{(t)}\right)\right)$ (that is, $\boldsymbol{\theta}^{(t)}$ is an optimal predictor learned on only the training data for task t). Plugging the second-order Taylor expansion into Equation 1 yields:

$$g_T(\mathbf{L}) = \frac{1}{T}\sum_{t=1}^{T} \min_{\mathbf{s}^{(t)}} \left\{ \frac{1}{n_t} \|\boldsymbol{\theta}^{(t)} - \mathbf{L}\mathbf{s}^{(t)}\|_{\mathbf{D}^{(t)}}^2 + \mu \|\mathbf{s}^{(t)}\|_1 \right\} + \lambda \|\mathbf{L}\|_{\mathsf{F}}^2 \tag{2}$$

where

$$\mathbf{D}^{(t)} = \frac{1}{2}\nabla^2_{\boldsymbol{\theta},\boldsymbol{\theta}} \frac{1}{n_t}\sum_{i=1}^{n_t} \mathcal{L}\left(f\left(\mathbf{x}_i^{(t)}; \boldsymbol{\theta}\right), y_i^{(t)}\right) \Bigg|_{\boldsymbol{\theta}=\boldsymbol{\theta}^{(t)}}$$

$$\boldsymbol{\theta}^{(t)} = \arg\min_{\boldsymbol{\theta}} \frac{1}{n_t}\sum_{i=1}^{n_t} \mathcal{L}\left(f\left(\mathbf{x}_i^{(t)}; \boldsymbol{\theta}\right), y_i^{(t)}\right) \ ,$$

and $\|\mathbf{v}\|_{\mathbf{A}}^2 = \mathbf{v}^\top \mathbf{A}\mathbf{v}$. In Equation 2, we have suppressed the constant term of the Taylor expansion (since it does not affect the minimum) and there is no linear term (since by construction $\boldsymbol{\theta}^{(t)}$ is a minimizer). Crucially, we have removed the dependence of the optimization on the number of data instances $n_1 \ldots n_T$ in each task. The approximation is exact in an important special case: when the model is linear and the loss function is squared loss (see Section 3.4.1).

The second inefficiency of Equation 1 is that in order to evaluate a single candidate $\mathbf{L}$, an optimization problem must be solved to recompute the value of each of the $\mathbf{s}^{(t)}$'s (which will become increasingly expensive as the number of tasks learned T increases). To overcome this problem, we modify the formulation in Equation 2 to remove the minimization over $\mathbf{s}^{(t)}$. We accomplish this by computing each of the $\mathbf{s}^{(t)}$'s when the training data for task t is last encountered, and not updating them when training on other tasks. At first glance this might seem to prevent the ability for previously learned tasks to benefit from training on later tasks (which we call *reverse transfer*); however, these tasks can benefit by subsequent modifications to $\mathbf{L}$. Later in Section 3.6, we show that this choice to update $\mathbf{s}^{(t)}$ only when we encounter training data for the respective task does not significantly affect the quality of model fit to the data as the number of tasks grows large. Using the previously computed values of $\mathbf{s}^{(t)}$ gives rise to the following optimization procedure (where we use the notation $\mathbf{L}_m$ to refer to the value of the latent components at the start of the mth iteration, and t is assumed to correspond to the particular

Algorithm 1 ELLA (k, d, λ, μ)

```
  $T \leftarrow 0$,   $\mathbf{A} \leftarrow \mathbf{zeros}_{k\times d,k\times d}$,
  $\mathbf{b} \leftarrow \mathbf{zeros}_{k\times d,1}$,   $\mathbf{L} \leftarrow \mathbf{zeros}_{d,k}$
  while isMoreTrainingDataAvailable() do
    $(\mathbf{X}_{\text{new}}, \mathbf{y}_{\text{new}}, t) \leftarrow$ getNextTrainingData()
    if isNewTask($t$) then
      $T \leftarrow T + 1$
      $\mathbf{X}^{(t)} \leftarrow \mathbf{X}_{\text{new}}$,   $\mathbf{y}^{(t)} \leftarrow \mathbf{y}_{\text{new}}$
    else
      $\mathbf{A} \leftarrow \mathbf{A} - \left(\mathbf{s}^{(t)}\mathbf{s}^{(t)\top}\right) \otimes \mathbf{D}^{(t)}$
      $\mathbf{b} \leftarrow \mathbf{b} - \text{vec}\left(\mathbf{s}^{(t)\top} \otimes \left(\boldsymbol{\theta}^{(t)\top}\mathbf{D}^{(t)}\right)\right)$
      $\mathbf{X}^{(t)} \leftarrow \left[\mathbf{X}^{(t)}\ \mathbf{X}_{\text{new}}\right]$,   $\mathbf{y}^{(t)} \leftarrow \left[\mathbf{y}^{(t)}; \mathbf{y}_{\text{new}}\right]$
    end if
    $\left(\boldsymbol{\theta}^{(t)}, \mathbf{D}^{(t)}\right) \leftarrow$ singleTaskLearner($\mathbf{X}^{(t)}, \mathbf{y}^{(t)}$)
    $\mathbf{L} \leftarrow$ reinitializeAllZeroColumns($\mathbf{L}$)
    $\mathbf{s}^{(t)} \leftarrow$ Equation 3
    $\mathbf{A} \leftarrow \mathbf{A} + \left(\mathbf{s}^{(t)}\mathbf{s}^{(t)\top}\right) \otimes \mathbf{D}^{(t)}$
    $\mathbf{b} \leftarrow \mathbf{b} + \text{vec}\left(\mathbf{s}^{(t)\top} \otimes \left(\boldsymbol{\theta}^{(t)\top}\mathbf{D}^{(t)}\right)\right)$
    $\mathbf{L} \leftarrow \text{mat}\left(\left(\frac{1}{T}\mathbf{A} + \lambda\mathbf{I}_{k\times d,k\times d}\right)^{-1}\frac{1}{T}\mathbf{b}\right)$
  end while
```

task for which we just received training data):

$$\mathbf{s}^{(t)} \leftarrow \arg\min_{\mathbf{s}^{(t)}} \ell(\mathbf{L}_m, \mathbf{s}^{(t)}, \boldsymbol{\theta}^{(t)}, \mathbf{D}^{(t)}) \quad (3)$$

$$\mathbf{L}_{m+1} \leftarrow \arg\min_{\mathbf{L}} \hat{g}_m(\mathbf{L}) \quad (4)$$

$$\hat{g}_m(\mathbf{L}) = \lambda\|\mathbf{L}\|_{\mathsf{F}}^2 + \frac{1}{T}\sum_{t=1}^{T} \ell\left(\mathbf{L}, \mathbf{s}^{(t)}, \boldsymbol{\theta}^{(t)}, \mathbf{D}^{(t)}\right) \quad (5)$$

where

$$\ell\left(\mathbf{L}, \mathbf{s}, \boldsymbol{\theta}, \mathbf{D}\right) = \mu\left\|\mathbf{s}\right\|_1 + \|\boldsymbol{\theta} - \mathbf{L}\mathbf{s}\|_{\mathbf{D}}^2 \quad . \quad (6)$$

Next, we present the specific steps needed to perform the updates in the preceding equations.

3.3. Model Update for ELLA

Suppose that at the mth iteration we receive training data for task t. We must perform two steps to update our model: compute $\mathbf{s}^{(t)}$ and update $\mathbf{L}$. In order to compute $\mathbf{s}^{(t)}$, we first compute an optimal model $\boldsymbol{\theta}^{(t)}$ using only the data from task t. The details of this step will depend on the form of the model and loss function under consideration, and thus here we treat it as a black box. If the training data for a particular task arrive interleaved with other tasks and not in a single batch, it may be important to use an online single-task learning algorithm to achieve maximum scalability.

Once $\boldsymbol{\theta}^{(t)}$ has been computed, we next compute $\mathbf{D}^{(t)}$ (which is model-dependent) and re-initialize (either randomly or to one of the $\boldsymbol{\theta}^{(t)}$'s) any columns of $\mathbf{L}$ that are all-zero (which will occur if a particular latent component is currently unused). We then compute $\mathbf{s}^{(t)}$ using the current basis $\mathbf{L}_m$ by solving an L_1-regularized regression problem—an instance of the Lasso.

To update $\mathbf{L}$, we null the gradient of Equation 5 and solve for $\mathbf{L}$. This procedure yields the updated column-wise vectorization of $\mathbf{L}$ as $\mathbf{A}^{-1}\mathbf{b}$, where:

$$\mathbf{A} = \lambda\mathbf{I}_{d\times k,d\times k} + \frac{1}{T}\sum_{t=1}^{T}\left(\mathbf{s}^{(t)}\mathbf{s}^{(t)\top}\right) \otimes \mathbf{D}^{(t)} \quad (7)$$

$$\mathbf{b} = \frac{1}{T}\sum_{t=1}^{T} \text{vec}\left(\mathbf{s}^{(t)\top} \otimes \left(\boldsymbol{\theta}^{(t)\top}\mathbf{D}^{(t)}\right)\right) \quad . \quad (8)$$

To avoid having to sum over all tasks to compute $\mathbf{A}$ and $\mathbf{b}$ at each step, we construct $\mathbf{A}$ incrementally as new tasks arrive (see Algorithm 1 for details).

Computational Complexity: Each update begins by running a single-task learner to compute $\boldsymbol{\theta}^{(t)}$ and $\mathbf{D}^{(t)}$; we assume that this step has complexity $O(\xi(d, n_t))$. Next, to update $\mathbf{s}^{(t)}$ requires solving an instance of the Lasso, which has complexity $O(nd\min(n, d))$, where d is the dimensionality and n is the number of data instances. Equation 3 can be seen as an instance of the Lasso in k dimensions with d data instances, for a total complexity of $O(dk^2)$. However, to formulate the Lasso problem requires computing the eigendecomposition of $\mathbf{D}^{(t)}$, which takes $O(d^3)$, and multiplying the matrix square root of $\mathbf{D}^{(t)}$ by $\mathbf{L}$, which takes $O(kd^2)$. Therefore, updating $\mathbf{s}^{(t)}$ takes time $O(d^3 + kd^2 + dk^2)$. A straightforward algorithm for updating $\mathbf{L}$ involves inverting a $(d \times k)$-by-$(d \times k)$ matrix, which has complexity $O(d^3k^3)$. However, we can exploit the fact that the updates to $\mathbf{A}$ are low-rank to derive a more efficient algorithm with complexity $O(d^3k^2)$ based on a recursive method (Yu, 1991) for updating the eigendecomposition of $\mathbf{A}$ (see Online Appendix). Therefore, using this more advanced approach, the overall complexity of each ELLA update is $O(k^2d^3 + \xi(d, n_t))$.

3.4. Base Learning Algorithms

Next, we show how two popular single-task learning algorithms can be used as the base learner for ELLA.

3.4.1. Linear Regression

In this setting $\mathbf{y}^{(t)} \in \mathbb{R}^{n_t}$, $f(\mathbf{x}; \boldsymbol{\theta}) = \boldsymbol{\theta}^\top\mathbf{x}$, and $\mathcal{L}$ is the squared-loss function. To apply ELLA, we compute the optimal single-task model $\boldsymbol{\theta}^{(t)}$, which is available in closed form as $\boldsymbol{\theta}^{(t)} = \left(\mathbf{X}^{(t)}\mathbf{X}^{(t)\top}\right)^{-1}\mathbf{X}^{(t)}\mathbf{y}^{(t)}$ (assuming that $\mathbf{X}^{(t)}\mathbf{X}^{(t)\top}$ is full-rank). $\mathbf{D}^{(t)}$ is also available in closed form as $\mathbf{D}^{(t)} = \frac{1}{2n_t}\mathbf{X}^{(t)}\mathbf{X}^{(t)\top}$. Given $\boldsymbol{\theta}^{(t)}$ and $\mathbf{D}^{(t)}$, we simply follow Algorithm 1 to fill in the model-independent details.

3.4.2. Logistic Regression

In this setting $\mathbf{y}^{(t)} \in \{-1,+1\}^{n_t}$, $f(\mathbf{x};\boldsymbol{\theta}) = 1/\left(1+e^{-\boldsymbol{\theta}^\top \mathbf{x}}\right)$, and $\mathcal{L}$ is the log-loss function. To apply ELLA, we first use a single-task learner for logistic regression (of which there are many free and robust implementations) to compute the value of $\boldsymbol{\theta}^{(t)}$. $\mathbf{D}^{(t)}$ is then given as:

$$\begin{aligned}\mathbf{D}^{(t)} &= \frac{1}{2n_t}\sum_{i=1}^{n_t}\sigma_i^{(t)}(1-\sigma_i^{(t)})\mathbf{x}_i^{(t)}\mathbf{x}_i^{(t)\top} \\ \sigma_i^{(t)} &= \frac{1}{1+e^{-\boldsymbol{\theta}^{(t)\top}\mathbf{x}_i^{(t)}}} \ .\end{aligned}$$

Given these formulas for $\boldsymbol{\theta}^{(t)}$ and $\mathbf{D}^{(t)}$, we follow Algorithm 1 to fill in the model-independent details.

3.5. Connection to Dictionary Learning for Sparse Coding

ELLA is closely connected to the problem of learning a dictionary online for sparse coding a set of input vectors. In fact, this problem is a special case of ELLA in which the $\boldsymbol{\theta}^{(t)}$'s are given as input instead of learned from training data and the $\mathbf{D}^{(t)}$'s are equal to the identity matrix. These simplifications yield the following objective function:

$$\begin{aligned}\beta(\mathbf{L}) &= \lambda\|\mathbf{L}\|_{\mathsf{F}}^2 + \frac{1}{T}\sum_{t=1}^{T}\|\boldsymbol{\theta}^{(t)} - \mathbf{L}\mathbf{s}^{(t)}\|_2^2 \qquad (9) \\ \mathbf{s}^{(t)} &= \arg\min_{\mathbf{s}}\left\{\mu\|\mathbf{s}\|_1 + \|\boldsymbol{\theta}^{(t)} - \mathbf{L}_t\mathbf{s}\|_2^2\right\} \ .\end{aligned}$$

Equation 9 is identical to the equation used for efficient online dictionary learning by Mairal et al. (2009) with the one difference that we use a soft constraint on the magnitude of the entries of $\mathbf{L}$ (L_2 regularization), whereas Mairal et al. use a hard length constraint on each column of $\mathbf{L}$.

3.6. Convergence Guarantees

Here, we provide proof sketches for three results; complete proofs are available in the Online Appendix. For simplicity of exposition, our analysis is performed in the setting where ELLA receives training data for a new task at each iteration. Therefore, the number of tasks learned T is always equal to the iteration number m. Extending our analysis to the more general case outlined in Algorithm 1 is straightforward. Our convergence proof is closely modeled on the analysis by Mairal et al. (2009).

We define the expected cost of a particular $\mathbf{L}$ as

$$g(\mathbf{L}) = \mathbb{E}_{\mathbf{D}^{(t)},\boldsymbol{\theta}^{(t)}}\left[\min_{\mathbf{s}}\ell(\mathbf{L},\mathbf{s},\boldsymbol{\theta}^{(t)},\mathbf{D}^{(t)})\right] \ ,$$

where we use a subscript on the expectation operator to denote which part of the expression is a random variable. The expected cost represents how well a particular set of latent components can be used to represent a randomly selected task given that the knowledge repository $\mathbf{L}$ is not modified.

We show three results on the convergence of ELLA, given respectively as Propositions 1–3:

1. The latent model component matrix, $\mathbf{L}_T$, becomes increasingly stable as T increases.
2. The value of the surrogate cost function, $\hat{g}_T(\mathbf{L}_T)$, and the value of the true empirical cost function, $g_T(\mathbf{L}_T)$, converge almost surely (a.s.) as $T \to \infty$.
3. $\mathbf{L}_T$ converges asymptotically to a stationary point of the expected loss g.

These results are based on the following assumptions:

A. The tuples $\left(\mathbf{D}^{(t)},\boldsymbol{\theta}^{(t)}\right)$ are drawn *i.i.d.* from a distribution with compact support.

B. For all $\mathbf{L}$, $\mathbf{D}^{(t)}$, and $\boldsymbol{\theta}^{(t)}$, the smallest eigenvalue of $\mathbf{L}_\gamma^\top\mathbf{D}^{(t)}\mathbf{L}_\gamma$ is at least κ (with $\kappa > 0$), where γ is the set of non-zero indices of the vector $\mathbf{s}^{(t)} = \arg\min_{\mathbf{s}}\|\boldsymbol{\theta}^{(t)} - \mathbf{L}\mathbf{s}\|_{\mathbf{D}^{(t)}}^2$. The non-zero elements of the unique minimizing $\mathbf{s}^{(t)}$ are given by: ${\mathbf{s}^{(t)}}_\gamma = \left(\mathbf{L}_\gamma^\top\mathbf{D}^{(t)}\mathbf{L}_\gamma\right)^{-1}\left(\mathbf{L}_\gamma^\top\mathbf{D}^{(t)}\boldsymbol{\theta}^{(t)} - \mu\boldsymbol{\epsilon}_\gamma\right)$, where the vector $\boldsymbol{\epsilon}_\gamma$ contains the signs of the non-zero entries of $\mathbf{s}^{(t)}$.

Proposition 1: $\mathbf{L}_{T+1} - \mathbf{L}_T = O\left(\frac{1}{T}\right)$.

Proof sketch: First, we show that the L_2 regularization term bounds the maximum magnitude of each entry of $\mathbf{L}$, and that the L_1 regularization term bounds the maximum magnitude of each entry of $\mathbf{s}^{(\mathbf{t})}$. Next, we show that $\hat{g}_T - \hat{g}_{T-1}$ is Lipschitz with constant $O\left(\frac{1}{T}\right)$. This result and the facts that $\mathbf{L}_{T-1}$ minimizes $\hat{g}_{T-1}$ and the L_2 regularization term ensures that the minimum eigenvalue of the Hessian of $\hat{g}_{T-1}$ is lower bounded by 2λ allow us to complete the proof. ■

Before stating our next proposition, we define:

$$\alpha(\mathbf{L},\boldsymbol{\theta}^{(t)},\mathbf{D}^{(t)}) = \arg\min_{\mathbf{s}}\ell(\mathbf{L},\mathbf{s},\boldsymbol{\theta}^{(t)},\mathbf{D}^{(t)}) \ , \qquad (10)$$

and introduce the following lemma:

Lemma 1:

A. $\min_{\mathbf{s}}\ell(\mathbf{L},\mathbf{s},\boldsymbol{\theta}^{(t)},\mathbf{D}^{(t)})$ is continuously differentiable in $\mathbf{L}$ with $\nabla_{\mathbf{L}}\min_{\mathbf{s}}\ell(\mathbf{L},\mathbf{s},\boldsymbol{\theta}^{(t)},\mathbf{D}^{(t)}) = -2\mathbf{D}^{(t)}\left(\boldsymbol{\theta}^{(t)} - \mathbf{L}\alpha(\mathbf{L},\boldsymbol{\theta}^{(t)},\mathbf{D}^{(t)})\right)\alpha(\mathbf{L},\boldsymbol{\theta}^{(t)},\mathbf{D}^{(t)})^\top$.

B. g is continuously differentiable with $\nabla g(\mathbf{L}) = 2\lambda\mathbf{I} + \mathbb{E}_{\boldsymbol{\theta}^{(t)},\mathbf{D}^{(t)}}\left[\nabla_{\mathbf{L}}\min_{\mathbf{s}}\ell(\mathbf{L},\mathbf{s},\boldsymbol{\theta}^{(t)},\mathbf{D}^{(t)})\right]$.

C. $\nabla_{\mathbf{L}}g(\mathbf{L})$ is Lipschitz on the space of latent model components $\mathbf{L}$.

Proof sketch: Part (A) can be easily shown using the fact that α is continuous and by applying a corollary of Theorem 4.1 as stated by Bonnans & Shapiro (1998) (originally shown by Danskin (1967)). Part (B) follows directly since the tuple $\left(\mathbf{D}^{(t)}, \boldsymbol{\theta}^{(t)}\right)$ is drawn from a distribution with compact support. Part (C) of the lemma crucially relies on Assumption (B), which ensures that the optimal sparse coding solution is unique. This fact, in combination with some properties that the optimal sparse coding solution must satisfy, allows us to prove that α is Lipschitz, which implies that $\nabla_{\mathbf{L}} g$ is Lipschitz due to the form of the gradient established in Parts (A) and (B). ■

Proposition 2:

A. $\hat{g}_T(\mathbf{L}_T)$ converges a.s.
B. $\hat{g}_T(\mathbf{L}_T) - g_T(\mathbf{L}_T)$ converges a.s. to 0.
C. $\hat{g}_T(\mathbf{L}_T) - g(\mathbf{L}_T)$ converges a.s. to 0.
D. $g(\mathbf{L}_T)$ converges a.s.

Proof sketch: First, we show that the sum of the positive variations of the stochastic process $u_T = \hat{g}_T(\mathbf{L}_T)$ are bounded by invoking a corollary of the Donsker theorem ((Van der Vaart, 2000) Chapter 19.2, lemma 19.36, ex. 19.7). Given this result, we apply a theorem from (Fisk, 1965) to show that u_t is a quasi-martingale that converges almost surely. The fact that u_t is a quasi-martingale along with a simple theorem of positive sequences allows us to prove part (B) of the proposition. The final two parts (C & D) can be shown due to the equivalence of g and g_T as $T \to \infty$. ■

Proposition 3: The distance between $\mathbf{L}_T$ and the set of g's stationary points converges a.s. to 0 as $T \to \infty$.

Proof sketch: We employ the fact that both the surrogate $\hat{g}_T$ and the expected cost g each have gradients that are Lipschitz with constant independent of T. This fact, in combination with the fact that $\hat{g}_T$ and g converge a.s. as $T \to \infty$, completes the proof. ■

4. Evaluation

We evaluate ELLA against three other approaches: (1) GO-MTL (Kumar & Daumé III, 2012), a batch MTL algorithm, (2) a perceptron-based approach to online multi-task learning (OMTL) (Saha et al., 2011), and (3) independent single-task learning (STL). GO-MTL provides a reasonable upper-bound on the accuracy of the models learned by ELLA (since it is a batch algorithm that optimizes all task models simultaneously). We are chiefly interested in understanding the tradeoff in accuracy between models learned with ELLA and GO-MTL, and the computational cost of learning these models. The comparison to OMTL allows us to understand how the performance of ELLA compares with another approach designed to learn efficiently in the lifelong learning setting.

4.1. Data Sets

We tested each algorithm on four multi-task data sets: (1) synthetic regression tasks, (2) land mine detection from radar images, (3) identification of three different facial movements from photographs of a subject, and (4) predicting student exam scores. Data sets (2) and (4) are benchmark data sets for MTL. We introduce data set (3) as an MTL problem for the first time.

Synthetic Regression Tasks We created a set of $T_{\max} = 100$ random tasks with $d = 13$ features and $n_t = 100$ instances per task. The task parameter vectors $\boldsymbol{\theta}^{(t)}$ were generated as a linear combination of $k = 6$ randomly generated latent components in $\mathbb{R}^{12}$. The vectors $\mathbf{s}^{(t)}$ had a sparsity level of 0.5 (i.e., half the latent components were used to construct each $\boldsymbol{\theta}^{(t)}$). The training data $\mathbf{X}^{(t)}$ was generated from a standard normal distribution. The training labels were given as $\mathbf{y}^{(t)} = \mathbf{X}^{(t)\top} \boldsymbol{\theta}^{(t)} + \epsilon$, where each element of ϵ is independent univariate Gaussian noise. A bias term was added as the 13th feature prior to learning.

Land Mine Detection In the land mine data set (Xue et al., 2007), the goal is to detect whether or not a land mine is present in an area based on radar images. The input features are automatically extracted from radar data and consist of four-moment based features, three correlation-based features, one energy-ratio feature, one spatial variance feature, and a bias term; see (Xue et al., 2007) for more details. The data set consists of a total of 14,820 data instances divided into 29 different geographical regions. We treat each geographical region as a different task.

Facial Expression Recognition This data set is from a recent facial expression recognition challenge (Valstar et al., 2011). The goal is to detect the presence or absence of three different facial action units (#5: upper lid raiser, #10: upper lip raiser, and #12: lip corner pull) from an image of a subject's face. We chose this combination of action units to be a challenge, since two of the action units involve the lower face, suggesting a high potential for transfer, while the other is an upper face action unit, suggesting a low potential for transfer. Each task involves recognizing one of the three action units for one of seven subjects, yielding a total of 21 tasks, each with 450–999 images. To represent the images, we utilized a Gabor pyramid with a frequency bandwidth of 0.7 octaves, orientation bandwidth of 120 degrees, four orientations, 576 locations, and two spatial scales, yielding a total of 2,880 Gabor features for each image. We reduced the raw Gabor outputs to 100 dimensions using PCA, and added a bias term to produce the input features.

London School Data The London Schools data set consists of examination scores from 15,362 students in 139 schools from the Inner London Education Authority. We treat the data from each school as a separate task. The goal is to predict the examination score of each student. We use the same feature encoding as used by Kumar & Daumé III (2012), where four school-specific categorical variables along with three student-specific categorical variables are encoded as a collection of binary features. In addition, we use the examination year and a bias term as additional features, giving each data instance $d = 27$ features.

4.2. Evaluation Procedure

Each data set was evaluated using 10 randomly generated 50/50 splits of the data between a training and hold-out test set. The particular splits were standardized across algorithms. For the online algorithms (ELLA and OMTL), we evaluated 100 randomized presentation orders of the tasks.

The parameter values of k and λ for ELLA and GO-MTL were selected independently for each algorithm and data set using a gridsearch over values of k from 1 to either 10 or $\frac{1}{4}T_{\max}$ (whichever was smaller) and values of λ from the set $\{e^{-5}, e^{-2}, e^{1}, e^{4}\}$. For ELLA, we selected the parameter values based on the training data alone. For a given training/test split, we further subdivided each training set into 10 random 50/50 sub-training/sub-validation sets and then chose parameter values that maximized the average performance on each of these 10 sub-validation sets. For GO-MTL, OMTL, and STL, the particular parameter values were selected to maximize test performance on the hold-out data averaged across all 10 random splits. Note that this procedure of choosing parameter values to maximize test performance provides ELLA with a disadvantage relative to the other algorithms.

The two parameters (burn-in time and learning rate) for OMTL were optimized using a gridsearch. The burn-in time was optimized over the set $\{50, 100, 150, \ldots, 400\}$ and the learning rate was optimized over the set $\{e^{-30}, e^{-29}, \ldots, e^{0}\}$. We report results using the LogDet update rule for OMTL, and found that the results did not vary greatly when other rules were employed; see (Saha et al., 2011) for more details on OMTL. For STL, the ridge term for either logistic or linear regression was selected by performing a gridsearch over the set $\{e^{-5}, e^{-4}, \ldots, e^{5}\}$.

Each task was presented sequentially to ELLA and OMTL, following the lifelong learning framework (Section 3.1). ELLA learned each new task from a single batch of data that contained all training instances of that task. For OMTL, which learns one instance at a time, we performed five passes over the training data for each task. We also tried using more than five passes, but the OMTL model accuracy did not increase further. GO-MTL was considered to have converged when either its objective function value decreased by less than 10^{-3} or 2,000 iterations were executed.

We measured predictive performance on the classification problems using the area under the ROC curve (AUC). This particular performance metric was chosen since both classification data sets had highly biased class distributions, and therefore other metrics like misclassification rate would only be informative for specific applications with well-specified tradeoffs between true and false positives. For regression problems, the performance was evaluated using the negative root mean-squared error (-rMSE) metric (with -rMSE, higher numbers indicate better performance). Recall that OMTL does not support regression, and so we do not evaluate it on the regression tasks.

The computational cost of each algorithm was measured using wall-clock time on a Mac Pro computer with 8GB RAM and two 6-core 2.67GHz Intel Xeon processors. We report the running time for the batch GO-MTL algorithm, and the speedup that ELLA, STL, and OMTL obtain relative to the batch algorithm both for learning *all* tasks and for learning each consecutive new task. We optimized the implementations of all algorithms to ensure a fair comparison.

4.3. Results

For classification problems, ELLA achieves nearly identical performance to GO-MTL (Table 1) while registering speedups of at least 1,350 times for learning all tasks and 38,400 times for learning each new task (Table 2). In addition, OMTL, which is specifically designed for learning efficiently online, achieved significantly worse accuracy on land mine detection and moderately worse accuracy on facial expression recognition. While OMTL did run much faster than GO-MTL, its speed did not match ELLA's. STL was the fastest approach (ELLA is necessarily slower than STL since STL is used as a subroutine inside ELLA), but had lower accuracy than ELLA.

We find similar results for the regression problems, with ELLA achieving nearly identical accuracy to GO-MTL (within 1.1% for real data and 2.3% for synthetic data; see Table 1), while achieving dramatically shorter learning times when learning all tasks (minimum speedup of 2,721 times) and each new task (minimum speedup of 378,219 times). STL was again the fastest of all approaches, but had lower accuracy.

Table 1. The accuracy of ELLA, OMTL, and STL relative to batch multi-task learning (GO-MTL), showing that ELLA achieves nearly equal accuracy to batch MTL and better accuracy than OMTL. The N/A's indicate that OMTL does not handle regression problems. The standard deviation of a value is given after the $\pm$ symbol.

Dataset	**Problem Type**	**Batch MTL Accuracy**	**ELLA Relative Accuracy**	**OMTL Relative Accuracy**	**STL Relative Accuracy**
Land Mine	Classification	0.7802 ± 0.013 (AUC)	$99.73 \pm 0.7\%$	$82.2 \pm 3.0\%$	$97.97 \pm 1.5\%$
Facial Expr.	Classification	0.6577 ± 0.021 (AUC)	$99.37 \pm 3.1\%$	$97.58 \pm 3.8\%$	$97.34 \pm 3.9\%$
Syn. Data	Regression	-1.084 ± 0.006 (-rMSE)	$97.74 \pm 2.7\%$	N/A	$92.91 \pm 1.5\%$
London Sch.	Regression	-10.10 ± 0.066 (-rMSE)	$98.90 \pm 1.5\%$	N/A	$97.20 \pm 0.4\%$

Table 2. The running time of ELLA, OMTL, and STL as compared to batch multi-task learning (GO-MTL), showing that ELLA achieves three orders of magnitude speedup in learning all tasks, and four-to-five orders of magnitude speedup in learning each consecutive new task. The N/A's indicate that OMTL does not handle regression. Speedup was measured relative to the batch method using optimized implementations. The standard deviation of a value is given after the $\pm$.

Dataset	**Batch Runtime (seconds)**	**ELLA All Tasks (speedup)**	**ELLA New Task (speedup)**	**OMTL All Tasks (speedup)**	**OMTL New Task (speedup)**	**STL All Tasks (speedup)**	**STL New Task (speedup)**
Land Mine	231±6.2	1,350±58	39,150±1,682	22±0.88	638±25	3,342±409	96,918±11,861
Facial Expr.	2,200±92	1,828±100	38,400±2,100	948±65	19,900±1,360	8,511±1,107	178,719±23,239
Syn. Data	1,300±141	5,026±685	502,600±68,500	N/A	N/A	156,489±17,564	1.6E6±1.8E5
London Sch.	715±36	2,721±225	378,219±31,275	N/A	N/A	36,000±4,800	5.0E6±6.7E5

Recall that ELLA does not re-optimize the value of $\mathbf{s}^{(t)}$ unless it receives new training data for task t. Therefore, in each experiment, the value of $\mathbf{s}^{(t)}$ is set when the training data for that task is presented and never readjusted. Although the values of $\mathbf{s}^{(t)}$ are not updated, it is still possible that previously learned task models can benefit from training on subsequent tasks through modifications to $\mathbf{L}$.

To assess whether this phenomenon of reverse transfer occurred, we computed the change in accuracy from when a task was first learned until after *all* tasks had been learned. A positive change in accuracy for a task indicates that reverse transfer did occur. Figure 2 shows this change in accuracy as a function of position in the task sequence, revealing that reverse transfer occurred reliably in all data sets and that reverse transfer is most beneficial for tasks that were learned early (when the total amount of training data seen was low). Most importantly, these results show that, with few exceptions, subsequent learning did not reduce the performance of models that were learned early.

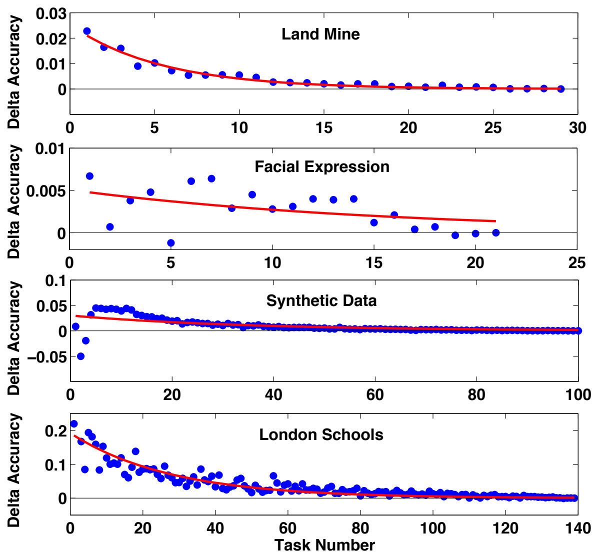


Figure 2. The change in accuracy from when a task is first learned until all tasks have been learned, as a function of position in the task sequence. Plotted lines are the best fitting exponential curve.

5. Conclusion

We have presented an efficient algorithm for lifelong learning (ELLA) that provides nearly identical accuracy to batch MTL, while requiring three orders of magnitude less runtime. Also, ELLA is more flexible, faster, and achieves better accuracy than a competing method for online MTL. We have shown that ELLA works well on synthetic data as well as three multi-task problems. Additionally, we discussed ELLA's connections to online dictionary learning for sparse coding, and presented theoretical guarantees that illuminate the reasons for ELLA's strong performance. Our future work will include extending ELLA to settings besides linear and logistic models and automatically adjusting the basis size k as it learns more tasks.

Acknowledgements

This research was supported by ONR grant N00014-11-1-0139. We thank Terran Lane, Diane Oyen, and the anonymous reviewers for their helpful feedback.

References

Bonnans, J.F. and Shapiro, A. Optimization problems with perturbations: A guided tour. *SIAM review*, 40 (2):228–264, 1998.

Carlson, A., Betteridge, J., Kisiel, B., Settles, B., Hruschka Jr., E.R., and Mitchell., T.M. Toward an architecture for never-ending language learning. In *Proceedings of the 24th Conference on Artificial Intelligence*. AAAI Press, 2010.

Danskin, J.M. *The theory of max-min and its application to weapons allocation problems*, volume 5 of Econometrics and Operations Research. Springer-Verlag, 1967.

Fisk, D.L. Quasi-martingales. *Transactions of the American Mathematical Society*, 120(3):369–389, 1965.

Kumar, A. and Daumé III, H. Learning task grouping and overlap in multi-task learning. In *Proceedings of the 29th International Conference on Machine Learning*, pp. 1383–1390. Omnipress, 2012.

Mairal, J., Bach, F., Ponce, J., and Sapiro, G. Online dictionary learning for sparse coding. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 689–696. ACM, 2009.

Rai, P. and Daumé III, H. Infinite predictor subspace models for multitask learning. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, pp. 613–620, 2010.

Ring, M.B. CHILD: A first step towards continual learning. *Machine Learning*, 28(1):77–104, 1997.

Saha, A., Rai, P., Daumé III, H., and Venkatasubramanian, S. Online learning of multiple tasks and their relationships. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, pp. 643–651, 2011.

Simm, J., Sugiyama, M., and Kato, T. Computationally efficient multi-task learning with least-squares probabilistic classifiers. *IPSJ Transactions on Computer Vision and Applications*, 3:1–8, 2011.

Sutton, R., Koop, A., and Silver, D. On the role of tracking in stationary environments. In *Proceedings of the 24th International Conference on Machine Learning*, pp. 871–878, Corvallis, OR, 2007. ACM.

Thrun, S. *Explanation-Based Neural Network Learning: A Lifelong Learning Approach.* Kluwer Academic Publishers, Boston, MA, 1996.

Thrun, S. and O'Sullivan, J. Discovering structure in multiple learning tasks: the TC algorithm. In *Proceedings of the 13th International Conference on Machine Learning*, pp. 489–497. Morgan Kaufmann, 1996.

Valstar, M.F., Jiang, B., Méhu, M., Pantic, M., and Scherer, K. The first facial expression recognition and analysis challenge. In *Proceedings of the IEEE International Conference on Automatic Face & Gesture Recognition*, pp. 921–926. IEEE, 2011.

Van der Vaart, A.W. *Asymptotic statistics*, volume 3. Cambridge University Press, 2000.

Xue, Y., Liao, X., Carin, L., and Krishnapuram, B. Multi-task learning for classification with Dirichlet process priors. *Journal of Machine Learning Research*, 8:35–63, 2007.

Yu, K.B. Recursive updating the eigenvalue decomposition of a covariance matrix. *IEEE Transactions on Signal Processing*, 39(5):1136–1145, 1991.

Zhang, J., Ghahramani, Z., and Yang, Y. Flexible latent variable models for multi-task learning. *Machine Learning*, 73(3):221–242, 2008.