Scalable Lifelong Learning with Active Task Selection

Paul Ruvolo and **Eric Eaton**
Bryn Mawr College
Computer Science Department
101 North Merion Avenue, Bryn Mawr, PA 19010
{pruvolo,eeaton}@cs.brynmawr.edu

Abstract

The recently developed Efficient Lifelong Learning Algorithm (ELLA) acquires knowledge incrementally over a sequence of tasks, learning a repository of latent model components that are sparsely shared between models. ELLA shows strong performance in comparison to other multi-task learning algorithms, achieving nearly identical performance to batch multi-task learning methods while learning tasks sequentially in three orders of magnitude (over 1,000x) less time. In this paper, we evaluate several curriculum selection methods that allow ELLA to actively select the next task for learning in order to maximize performance on future learning tasks. Through experiments with three real and one synthetic data set, we demonstrate that active curriculum selection allows an agent to learn up to 50% more efficiently than when the agent has no control over the task order.

Introduction

Current multi-task learning algorithms improve performance by sharing learned knowledge between multiple task models. With few exceptions, multi-task learning has focused primarily on learning in a batch framework, where all tasks are given at once. However, versatile learning agents need to be able to learn tasks consecutively, continually building upon their knowledge over a lifetime of experience.

In this *lifelong learning* framework, tasks arrive sequentially and the learning agent's goal is to maximize its performance across all tasks. We recently developed the Efficient Lifelong Learning Algorithm (ELLA) to solve this online multi-task learning problem (Ruvolo and Eaton 2013). ELLA learns and maintains a repository of latent model components that are shared between task models. Given a new learning task, ELLA transfers knowledge from these latent model components when training the new task model, and then refines the components with knowledge learned from the new task. This refinement process allows newly acquired knowledge to improve existing model components, thereby improving previously learned task models that depend on these components. The computational efficiency of ELLA is supported by rigorous theoretical guarantees on performance and convergence. Empirically, we have shown that ELLA yields nearly identical performance to batch multi-task learning while learning all tasks in three orders of magnitude (over 1,000x) less time.

In this paper, we explore the use of *active curriculum selection* to improve ELLA's scalability for learning over a long sequence of tasks. Specifically, we focus on a setting in which a lifelong learner can choose the next task to learn from a pool of candidate tasks in order to maximize performance. In its original evaluation, ELLA had no control over the order in which tasks were presented. We show that active curriculum selection enables ELLA to achieve large gains in learning efficiency compared to when the task order is outside the agent's control, thereby reducing the number of tasks required to achieve a particular level of performance and improving scalability.

After surveying related work, we describe ELLA in detail and summarize its empirical performance. Then, we formalize the active curriculum selection problem, describe several mechanisms for active task selection, and finally compare these mechanisms in lifelong learning scenarios.

Related Work

The model framework used in ELLA is closely related to a number of current multi-task learning (MTL) algorithms (Rai and Daumé III 2010; Zhang, Ghahramani, and Yang 2008; Kumar and Daumé III 2012) that represent each individual task's model as a linear combination of a common shared basis $\mathbf{L}$. In most cases, the basis is learned in tandem with all task models in an expensive batch training process. Adding even a single task may require re-optimization of all models, making these batch MTL approaches unsuitable for lifelong learning. Saha et al. (2011) proposed an algorithm for online multi-task classification, OMTL, that is based on perceptron learning and supports incremental learning. However, OMTL exhibits worse performance than ELLA, both in terms of accuracy and speed.

Task selection is closely related to both active learning and (passive) curriculum learning (Bengio et al. 2009) methods, which order training instances in an attempt to maximize learning performance. Active learning has been evaluated in a number of MTL algorithms (Saha et al. 2010; Zhang 2010; Saha et al. 2011), and has been shown to generally reduce the amount of training data necessary to achieve

a particular level of performance. This paper builds on this earlier work by extending these ideas to lifelong learning, and introducing an efficient and highly effective mechanism for selecting the next task to learn: the diversity heuristic. In contrast to these active MTL methods, which focus primarily on optimizing model performance, we focus on learning the basis $\mathbf{L}$ efficiently, which will provide maximal benefit to learning future tasks.

The Lifelong Learning Problem

This paper uses the following notational conventions: matrices are denoted by bold uppercase letters, vectors are denoted by bold lowercase letters, scalars are denoted by lowercase letters in normal typeface, and sets are denoted using script typeface (e.g., $\mathcal{A}$). Quantities related to a particular task are denoted using parenthetical superscripts (e.g., matrix $\mathbf{A}^{(t)}$ and vector $\mathbf{v}^{(t)}$ are each related to task t). We use the shorthand $\mathbf{A}^{(1:t)}$ to refer to the list of variables $\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(t)}$.

We employ a lifelong learning framework in which the agent faces a series of supervised learning tasks $\mathcal{Z}^{(1)}, \mathcal{Z}^{(2)}, \ldots, \mathcal{Z}^{(T_{\max})}$. Each learning task $\mathcal{Z}^{(t)} = \left(\hat{f}^{(t)}, \mathbf{X}^{(t)}, \mathbf{y}^{(t)}\right)$ is specified by a (hidden) function $\hat{f}^{(t)} : \mathcal{X}^{(t)} \mapsto \mathcal{Y}^{(t)}$ 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). To learn $\hat{f}^{(t)}$, the agent is given 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 $\hat{f}^{(t)}$. For brevity, we use $(\mathbf{x}_i^{(t)}, y_i^{(t)})$ to represent the ith labeled training instance for task t. The agent does not know *a priori* the total number of tasks $T_{\max}$, their order, or their distribution.

Each time step, the agent receives a batch of labeled training data for some task t, either a new task or as additional data for a previous task. Let T denote the number of tasks the agent has encountered so far. At any time, the agent may be asked to make predictions on data from 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 each $f^{(t)}$ will approximate $\hat{f}^{(t)}$ to enable the accurate labeling of new data. Additionally, the agent must be able to rapidly update each model $f^{(t)}$ in response to new training data for a previously learned task, and efficiently add $f^{(t)}$'s to model new tasks. Figure 1 illustrates the lifelong learning process.

Efficient Lifelong Learning Algorithm

This section provides an overview of the Efficient Lifelong Learning Algorithm (ELLA) (Ruvolo and Eaton 2013) that forms the foundation for our proposed enhancements. For further details on ELLA, see the original paper.

ELLA learns and maintains a repository of k latent model components $\mathbf{L} \in \mathbb{R}^{d \times k}$, which forms a basis for all task models and serves as the mechanism for knowledge transfer between tasks. For each task t, ELLA learns a model $f^{(t)}(\mathbf{x}) = f(\mathbf{x}; \boldsymbol{\theta}^{(t)})$ that is parameterized by a d-dimensional task-specific vector $\boldsymbol{\theta}^{(t)} = \mathbf{L}\mathbf{s}^{(t)}$ represented as a sparse linear combination of the columns of $\mathbf{L}$ using the weight vector $\mathbf{s}^{(t)} \in \mathbb{R}^k$. The specific form of $f^{(t)}(\mathbf{x})$ is dependent on the base learning algorithm, as described later.

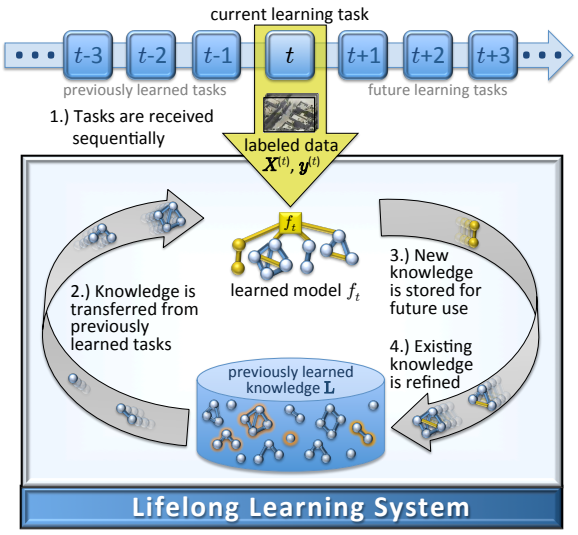

Figure 1: An illustration of the lifelong learning process.

Given the labeled training data for each task, ELLA optimizes the predictive loss of each model while encouraging shared structure through $\mathbf{L}$ by minimizing:

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

where $\mathcal{L}$ is a known loss function for fitting the base learning algorithm (e.g., squared loss, etc.). The form of this optimization problem is closely related to a number of batch MTL algorithms, such as GO-MTL (Kumar and Daumé III 2012). Equation 1 is not jointly convex in $\mathbf{L}$ and the $\mathbf{s}^{(t)}$'s, and so most batch MTL methods (e.g., GO-MTL) yield a local optimum using an alternating optimization procedure. This alternating optimization procedure over all tasks is expensive, preventing its use in a lifelong learning setting.

To enable its efficient solution, ELLA approximates Equation 1 using two simplifications:

1. To eliminate the explicit dependence on all of the previous training data through the inner summation, ELLA approximates 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 $\boldsymbol{\theta}^{(t)} = \arg\min_{\boldsymbol{\theta}} \frac{1}{n_t}\sum_{i=1}^{n_t} \mathcal{L}\left(f\left(\mathbf{x}_i^{(t)}; \boldsymbol{\theta}\right), y_i^{(t)}\right)$ is an optimal predictor learned on only the training data on task t. Crucially, this step removes the optimization's dependence on the number of data instances $n_1 \ldots n_T$ in each task.
2. To eliminate the need to recompute each of the $\mathbf{s}^{(t)}$'s in response to any changes in $\mathbf{L}$, ELLA computes each $\mathbf{s}^{(t)}$ only when training on task t, and does not update them when training on other tasks. While computationally ef-

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$(\mathbf{L})$
  $\mathbf{s}^{(t)} \leftarrow \arg\min_{\mathbf{s}^{(t)}} \ell(\mathbf{L}_m, \mathbf{s}^{(t)}, \boldsymbol{\theta}^{(t)}, \mathbf{D}^{(t)})$
  $\mathbf{A} \leftarrow \mathbf{A} + \left(\mathbf{s}^{(t)}\mathbf{s}^{(t)\top}\right) \otimes \mathbf{D}^{(t)}$
  $\mathbf{b} \leftarrow \mathbf{b} + \text{vec}\left(\mathbf{s}^{(t)\top} \otimes \left(\boldsymbol{\theta}^{(t)\top}\mathbf{D}^{(t)}\right)\right)$
  $\mathbf{L} \leftarrow \text{mat}\left(\left(\frac{1}{T}\mathbf{A} + \lambda\mathbf{I}_{k\times d, k\times d}\right)^{-1}\frac{1}{T}\mathbf{b}\right)$
end while

ficient, this choice forces all subsequent effects on previously learned models from future learning to occur only through changes in $\mathbf{L}$ (instead of altering $\mathbf{s}^{(t)}$), but empirically does not significantly affect performance. As the number of tasks grows large, the performance penalty for this simplification becomes vanishingly small (Ruvolo and Eaton 2013).

These simplifications yield the following efficient update equations that approximate the result of Equation 1:

$$\mathbf{s}^{(t)} \leftarrow \arg\min_{\mathbf{s}^{(t)}} \ell(\mathbf{L}_m, \mathbf{s}^{(t)}, \boldsymbol{\theta}^{(t)}, \mathbf{D}^{(t)}) \tag{2}$$

$$\mathbf{L}_{m+1} \leftarrow \arg\min_{\mathbf{L}} \hat{g}_m(\mathbf{L}) \tag{3}$$

$$\hat{g}_m(\mathbf{L}) = \lambda\|\mathbf{L}\|_{\text{F}}^2 + \frac{1}{T}\sum_{t=1}^{T} \ell\left(\mathbf{L}, \mathbf{s}^{(t)}, \boldsymbol{\theta}^{(t)}, \mathbf{D}^{(t)}\right) \tag{4}$$

$$\text{where } \ell\left(\mathbf{L}, \mathbf{s}, \boldsymbol{\theta}, \mathbf{D}\right) = \mu\left\|\mathbf{s}\right\|_1 + \left\|\boldsymbol{\theta} - \mathbf{L}\mathbf{s}\right\|_{\mathbf{D}}^2 \ , \tag{5}$$

$\mathbf{L}_m$ refers to the value of the latent model components at the start of the mth iteration, $\mathbf{D}^{(t)}$ is the Hessian of the loss function $\mathcal{L}$ evaluated at $\boldsymbol{\theta}^{(t)}$ multiplied by $1/2$, and t is assumed to correspond to the specific task for which ELLA just received training data.

Given a new task t, ELLA first computes an optimal model $\boldsymbol{\theta}^{(t)}$ using only the data from task t. Once $\boldsymbol{\theta}^{(t)}$ has been computed, we compute $\mathbf{D}^{(t)}$. The form of this step depends on the base learning algorithm. For example, in the regression setting, where $\mathbf{y}^{(t)} \in \mathbb{R}^{n_t}$, $f(\mathbf{x}; \boldsymbol{\theta}) = \boldsymbol{\theta}^\top\mathbf{x}$, and $\mathcal{L}$ is the squared-loss function, the optimal single-task model $\boldsymbol{\theta}^{(t)} = \left(\mathbf{X}^{(t)}\mathbf{X}^{(t)\top}\right)^{-1}\mathbf{X}^{(t)}\mathbf{y}^{(t)}$ and $\mathbf{D}^{(t)} = \frac{1}{2n_t}\mathbf{X}^{(t)}\mathbf{X}^{(t)\top}$. Using logistic regression for classification, where $\mathbf{y}^{(t)} \in \{-1, +1\}^{n_t}$, $f(\mathbf{x}; \boldsymbol{\theta}) = 1/\left(1+e^{-\boldsymbol{\theta}^\top\mathbf{x}}\right)$ and $\mathcal{L}$ is the log-loss function, ELLA first optimizes $\boldsymbol{\theta}^{(t)}$ by solving the standard single-task logistic regression problem, and then computes $\mathbf{D}^{(t)}$ as:

$$\mathbf{D}^{(t)} = \frac{1}{2n_t}\sum_{i=1}^{n_t} f(\mathbf{x}_i^{(t)}, \boldsymbol{\theta}^{(t)})\left(1 - f(\mathbf{x}_i^{(t)}, \boldsymbol{\theta}^{(t)})\right)\mathbf{x}_i^{(t)}\mathbf{x}_i^{(t)\top} \ .$$

To compute $\mathbf{s}^{(t)}$, ELLA first re-initializes (either randomly or to one of the $\boldsymbol{\theta}^{(t)}$'s) any columns of $\mathbf{L}$ that are all-zero (due to their being unused in any model), and then computes $\mathbf{s}^{(t)}$ using the current basis $\mathbf{L}_m$ by solving Equation 2. Once task t has been modeled, ELLA updates $\mathbf{L}$ to incorporate new knowledge by nulling the gradient of Equation 4 and solving for $\mathbf{L}$ as $\mathbf{A}^{-1}\mathbf{b}$, where:

$$\mathbf{A} = \lambda\mathbf{I}_{d\times k, d\times k} + \frac{1}{T}\sum_{t=1}^{T}\left(\mathbf{s}^{(t)}\mathbf{s}^{(t)\top}\right) \otimes \mathbf{D}^{(t)} \tag{6}$$

$$\mathbf{b} = \frac{1}{T}\sum_{t=1}^{T}\text{vec}\left(\mathbf{s}^{(t)\top} \otimes \left(\boldsymbol{\theta}^{(t)\top}\mathbf{D}^{(t)}\right)\right) \ . \tag{7}$$

The matrix $\mathbf{A}$ is built up incrementally as new tasks arrive for efficiency. The complete ELLA is given as Algorithm 1.

Ruvolo and Eaton (2013) show that ELLA can be viewed as a more general case of online dictionary learning for sparse coding, and provide a variety of theoretical guarantees on ELLA's performance and convergence. These guarantees are supported by empirical results showing that ELLA achieves nearly identical performance to batch multi-task optimization of Equation 1 (98.9–99.7% accuracy of batch GO-MTL on real data; 97.7% on synthetic data) while learning all tasks in three orders of magnitude (1,350x–5,026x) less time. Since ELLA can efficiently learn new tasks, it can acquire a single new model in four to five orders of magnitude (38,400x–502,600x) less time than it takes to re-optimize Equation 1. In comparison, the most recently developed competing algorithm, online MTL (Saha et al. 2011), only works on classification problems and achieves 82.2%–97.6% accuracy of batch GO-MTL with a speedup of only 22x–948x to learn all tasks.

Active Curriculum Selection

In the formulation of lifelong learning depicted in Figure 1, the learning agent has no control over which task is presented next in sequence. However, in situations where a lifelong learning agent can choose the task order, the agent can actively select the next task to learn in order to maximize performance over all tasks.

We formalize the problem of active curriculum selection in the following manner. Consider a learning agent that has access to training data from a pool of unlearned tasks $\mathbf{X}^{(T+1)}, \ldots, \mathbf{X}^{(T_{\text{pool}})}$, where $T + 1 \leq T_{\text{pool}} \leq T_{\max}$ and only a subset of the training instances for each of these tasks is labeled. Let $\left(\hat{\mathbf{X}}^{(t)}, \hat{\mathbf{y}}^{(t)}\right)$ denote this subset of initially labeled data for task t. The agent must select the index of the

next task to learn $t_{\text{next}} \in \{T+1, \ldots, T_{\text{pool}}\}$.[1] The value of T_{pool} could be a fixed value or it could be set dynamically during learning (for example, we could set $T_{\text{pool}} = T + z$ so that the agent has knowledge of the next z upcoming tasks). Once the agent decides to learn a particular task, all of the training labels $\mathbf{y}^{(t_{\text{next}})}$ are revealed to the agent, allowing it to learn the new task model.

What strategy should an agent employ in selecting t_{next}? Ideally, the agent should choose to learn a task that maximizes its learning performance across future tasks that it is likely to encounter. We can evaluate the quality of a task selection criterion by its ability to learn a knowledge repository $\mathbf{L}$ that can efficiently and accurately model new tasks. The more effective a task selection criterion is, the less tasks it will need to learn in order to achieve a specified performance threshold on future tasks.

We consider three potential approaches for choosing the next task to learn:

- choose t_{next} to maximize the information gain on $\mathbf{L}$,
- choose t_{next} as the task with the worst performance given the current $\mathbf{L}$, encouraging $\mathbf{L}$ to support diverse tasks, and
- choose t_{next} randomly (this corresponds to the original lifelong learning setting in which the curriculum order is not under the learner's control).

Next, we describe these approaches for active curriculum selection, and then, in the Evaluation section, compare their performance in a lifelong learning setting. We also evaluated a fourth approach that chooses to learn the task with the best performance given the current $\mathbf{L}$, but found that this method performed significantly worse than choosing t_{next} randomly in all cases and so do not consider it further.

Information Maximization Heuristic

Here, we develop a heuristic for choosing the task to learn next that maximizes our information (or equivalently minimizes our uncertainty) about the "true" latent model component repository $\mathbf{L}$. The intuition behind this choice is that the less ambiguity we have about the value of $\mathbf{L}$, the better our latent component repository will be for efficiently and accurately learning new tasks. Therefore, if our ultimate goal is to learn a good component repository using the fewest number of learned tasks, we should choose the task to learn at each iteration that maximizes our expected information gain about the true value of $\mathbf{L}$.

Suppose we are trying to infer the value of a random vector $\mathbf{h}$ that cannot be directly observed. We assume that we are able to observe the values of random vectors that in turn allow us to infer the value of $\mathbf{h}$. Let $\mathcal{X}_j$ be a set of random vectors that the learner can choose to observe at iteration j. Here, we use a myopic information maximization (InfoMax) approach in which the learner chooses to observe the random vector from the set $\mathcal{X}_j$ that gives the most expected information about $\mathbf{h}$:

$$\begin{aligned} t_{\text{next}} &= \arg\max_{\mathbf{x} \in \mathcal{X}_j} I(\mathbf{h}; \mathbf{x} | \mathcal{I}_j) \\ &= \arg\max_{\mathbf{x} \in \mathcal{X}_j} H\left[\mathbf{h} | \mathcal{I}_j\right] - H\left[\mathbf{h} | \mathbf{x}, \mathcal{I}_j\right] \\ &= \arg\min_{\mathbf{x} \in \mathcal{X}_j} H\left[\mathbf{h} | \mathbf{x}, \mathcal{I}_j\right] \\ &= \arg\min_{\mathbf{x} \in \mathcal{X}_j} \int p(\mathbf{x} = \mathbf{v} | \mathcal{I}_j) H\left[\mathbf{h} | \mathbf{x} = \mathbf{v}, \mathcal{I}_j\right] d\mathbf{v} \ , \end{aligned}$$

where I is the mutual information operator, H is the differential entropy operator, $\mathcal{I}_j$ represents all of the information available at iteration j, $\mathbf{x}$ is a random vector, and $\mathbf{x} = \mathbf{v}$ is the event that the random vector $\mathbf{x}$ takes on value $\mathbf{v}$.

To apply the InfoMax heuristic to the problem of active curriculum selection, we choose t_{next} such that when we observe $\boldsymbol{\theta}^{(t_{\text{next}})}$ and $\mathbf{D}^{(t_{\text{next}})}$, we have the highest expected gain in information about the latent task matrix $\mathbf{L}$. However, the original ELLA is not formulated probabilistically and thus we have no way to compute the differential entropy of $\mathbf{L}$ given the previously learned tasks. To overcome this limitation of the original model, we use the LaPlace approximation to construct a Gaussian distribution for $\mathbf{L}$ about a mode of Equation 4. Specifically, given $\mathbf{L}^\star$, which is the column-wise vectorized version of a mode of Equation 1, we apply LaPlace's approximation to derive a Gaussian approximation to the column-wise vectorized version of $\mathbf{L}$ that has mean $\mathbf{L}^\star$ and covariance $\boldsymbol{\Sigma}^\star = \frac{1}{2}\left(\lambda \mathbf{I} + \frac{1}{T}\sum_{t=1}^{T} \frac{1}{n_t}\left(\mathbf{s}^{(t)}\mathbf{s}^{(t)\top}\right) \otimes \mathbf{D}^{(t)}\right)^{-1}$. Computing the expected conditional entropy after observing the tuple $\left(\boldsymbol{\theta}^{(t)}, \mathbf{D}^{(t)}\right)$ under the LaPlace approximation yields:

$$\begin{aligned} &\iint p\left(\boldsymbol{\theta}^{(t)} = \mathbf{u}, \mathbf{D}^{(t)} = \mathbf{V} | \mathbf{D}^{(1:T)}, \boldsymbol{\theta}^{(1:T)}\right) \qquad (8) \\ &\quad \times H\left[\mathbf{L} | \boldsymbol{\theta}^{(t)} = \mathbf{u}, \mathbf{D}^{(t)} = \mathbf{V}, \mathbf{D}^{(1:T)}, \boldsymbol{\theta}^{(1:T)}\right] d\mathbf{u}\, d\mathbf{V} \ , \end{aligned}$$

where

$$\begin{aligned} &H\left[\mathbf{L} | \boldsymbol{\theta}^{(t)} = \mathbf{u}, \mathbf{D}^{(t)} = \mathbf{V}, \mathbf{D}^{(1:T)}, \boldsymbol{\theta}^{(1:T)}\right] \qquad (9) \\ &\quad = -\frac{1}{2}\log_2 \det\left(\Sigma_j^{-1} + \left(\mathbf{s}^{(t)}_{\boldsymbol{\theta}^{(t)}, \mathbf{D}^{(t)}} \mathbf{s}^{(t)\top}_{\boldsymbol{\theta}^{(t)}, \mathbf{D}^{(t)}}\right) \otimes \mathbf{D}^{(t)}\right) + \text{c} \ , \end{aligned}$$

$$\mathbf{s}^{(t)\top}_{\boldsymbol{\theta}^{(t)}, \mathbf{D}^{(t)}} = \arg\min_{\mathbf{s}} \ell\left(\mathbf{L}_j, \mathbf{s}, \boldsymbol{\theta}^{(t)}, \mathbf{D}^{(t)}\right) \ , \qquad (10)$$

and c is a constant. Unfortunately, computing the double integral over all possible values of $\boldsymbol{\theta}^{(t)}$ and $\mathbf{D}^{(t)}$ is intractable in the general case. Here, we use the portion of labeled training data for training task t to estimate the values of $\boldsymbol{\theta}^{(t)}$ and $\mathbf{D}^{(t)}$ as $\boldsymbol{\theta}^{(t)\star}$ and $\mathbf{D}^{(t)\star}$. Next, we replace the integral over all possible values of $\boldsymbol{\theta}^{(t)}$ and $\mathbf{D}^{(t)}$ with $H\left[\mathbf{L} | \boldsymbol{\theta}^{(t)\star}, \mathbf{D}^{(t)\star}, \mathbf{D}^{(1:T)}, \boldsymbol{\theta}^{(1:T)}\right]$. Once we apply this approximation, we can select

$$t_{\text{next}} = \arg\min_{t \in \{T+1, \ldots, T_{\text{pool}}\}} H\left[\mathbf{L} | \boldsymbol{\theta}^{(t)\star}, \mathbf{D}^{(t)\star}, \boldsymbol{\theta}^{(1:T)}, \mathbf{D}^{(1:T)}\right] \cdot \qquad (11)$$

In the case where the distribution over $\mathbf{L}$ is Gaussian, the information maximization criterion above is equivalent to the

[1] Note that after the index is chosen, $T \leftarrow T + 1$ and the tasks are reordered such that the remaining unlearned tasks in the pool are given as $\mathcal{Z}^{(T+1)}, \ldots, \mathcal{Z}^{(T_{\text{pool}})}$.

D-optimality criterion from the Bayesian Experimental Design literature (Chaloner and Verdinelli 1995). In addition to this criterion, we tested another commonly used criterion from Bayesian Experimental Design called A-optimality (Chaloner and Verdinelli 1995) which corresponds to selecting the task that minimizes:

$$\text{trace}\left(\left(\Sigma_j^{-1}+\left(\mathbf{s}^{(t)}_{\boldsymbol{\theta}^{(t)},\mathbf{D}^{(t)}}\mathbf{s}^{(t)\top}_{\boldsymbol{\theta}^{(t)},\mathbf{D}^{(t)}}\right)\otimes\mathbf{D}^{(t)}\right)^{-1}\right) \quad . \quad (12)$$

This equation corresponds to minimizing the average variance of the distribution over the entries of $\mathbf{L}$.

Diversity Heuristic (Worst Performing Task)

Next, we propose a heuristic that encourages $\mathbf{L}$ to serve as an effective basis for a wide variety of tasks, enabling ELLA to be able to rapidly learn any new task. To encourage diversity, ELLA should select the next task as the one that the current basis $\mathbf{L}$ is doing the *worst* job solving, relative to how well that particular task could be solved using a single task learning model:

$$t_{\text{next}} = \underset{t\in\{T+1,\ldots,T_{\text{pool}}\}}{\arg\max}\ \min_{\mathbf{s}} \ell\left(\mathbf{L}_j,\mathbf{s},\boldsymbol{\theta}^{(t)\star},\mathbf{D}^{(t)\star}\right) \quad , \quad (13)$$

where $\boldsymbol{\theta}^{(t)\star}$ and $\mathbf{D}^{(t)\star}$ are computed based on the subset of labeled data for task t, as in the previous section. Effectively, Equation 13 focuses on selecting tasks that are poorly encoded with the current basis (i.e. it is unlike anything that the learner has seen so far), thereby encouraging diversity.

Diversity++ Heuristic

Part of our inspiration for developing the diversity heuristic was the work by Arthur and Vassilvitskii (2007) on choosing points to initialize the k-means algorithm. This algorithm, k-means++, has been shown to perform very well in a wide variety of applications (Arthur and Vassilvitskii 2007). The k-means++ algorithm begins by selecting a random data instance to serve as a cluster centroid. At each stage of the algorithm, a new data instance is added as a cluster center with probability proportional to the square of the distance between the candidate instance and the closest currently selected center. This selection rule encourages instances to be selected as centers that are poorly represented by the current set of centers. We view this strategy as analogous to the *diversity* heuristic in the sense that we measure squared distance based on the error of the current basis on coding the training data for a candidate task. The other key difference between the two approaches is that the *diversity* heuristic is deterministic (i.e. it always selects the worst fit task) whereas k-means++ is stochastic (i.e. it selects instances proportional to the squared distance). In order to test if this detail crucially affects performance, we tested an additional active task selection method called the *diversity++* heuristic in which the probability of selecting a particular task for learning is:

$$p(t_{\text{next}}=t) = \frac{\left(\min_{\mathbf{s}} \ell\left(\mathbf{L}_j,\mathbf{s},\boldsymbol{\theta}^{(t)\star},\mathbf{D}^{(t)\star}\right)\right)^2}{\sum_{t'}\left(\min_{\mathbf{s}} \ell\left(\mathbf{L}_j,\mathbf{s},\boldsymbol{\theta}^{(t')\star},\mathbf{D}^{(t')\star}\right)\right)^2} \quad . \quad (14)$$

The *diversity++* heuristic is a softer and stochastic version of the *diversity* heuristic; in fact, the diversity heuristic is equivalent to $\arg_t \max p(t_{\text{next}}=t)$.

Evaluation

We evaluated both *InfoMax* heuristics (*A-optimality* and *D-optimality*), the *diversity* heuristic, and the *diversity++* heuristic against *random* task selection. Evaluations were performed on one synthetic and three real learning problems.

Data Sets

We tested each method of task selection on four data sets: (1) a set of synthetic regression tasks, (2) student examination score prediction, (3) land mine detection from radar imaging, and (4) identification of three different facial movements from photographs of a subject's face.

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 using 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., only half the latent components were used to construct each $\boldsymbol{\theta}^{(t)}$). Each element of the input training data $\mathbf{X}^{(t)}$ was generated from a standard normal distribution. The training labels for each task 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.

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 and 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 exam year and a bias term as additional features, giving each data instance $d = 27$ features total.

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 10 input features (plus a bias term) are automatically extracted from radar data; see (Xue et al. 2007) for more details. The data set consists 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.

Evaluation Procedure

For each task, we created random $50\%/50\%$ splits between training and hold-out test data. Additionally, we divided the set of tasks into two approximately equally sized subsets: one set of *training tasks* that serves as the pool for active learning and is used to learn $\mathbf{L}$, and one set of *evaluation tasks* on which we measure the performance of the learned $\mathbf{L}$. Therefore, $T_{\text{pool}} \approx \frac{1}{2}T_{\max}$ in these experiments. The agent never has access to the evaluation tasks, and so must select the curriculum order from the training tasks alone. For each training task, 25% of its training data was revealed as the initial sample of labeled data (recall that each task selection heuristic requires a small amount of labeled data to measure the utility of learning a particular task). Each experiment was repeated 1,000 times to smooth out variability due to the factors discussed above. The evaluation of a particular task selection method on each trial is as follows:

1. Begin with no tasks learned; $\mathbf{L}$ is initialized randomly.
2. Select the next training task to learn using some method.
3. Learn the new task using Equations 2–4 to arrive at a new set of latent components $\mathbf{L}$.
4. Compute models for each evaluation task using Equation 2 for the current $\mathbf{L}$ (no modification of $\mathbf{L}$ is allowed).
5. Record the performance of the evaluation task models on the hold-out test data, yielding an estimate of how well the current $\mathbf{L}$ captures knowledge for novel tasks.
6. Go to step 2 while additional training tasks remain.

The values of the parameters k and λ for ELLA were selected using a grid-search over the values $k \in \{1 \dots 10\}$ and $\lambda \in \{e^{-5}, e^{-2}, e^{1}, e^{4}\}$ to maximize performance on the evaluation tasks averaged over each of the task selection methods. The value of μ was set to 1.

We are primarily concerned with the level of performance achieved by each task selection method as a function of the number of tasks learned. Specifically, we compute the following metrics:

1. **Performance**: the average performance level (across tasks) achieved by a particular heuristic as a function of the number of tasks learned. Performance is measured using area under the ROC for classification tasks (which was chosen due to the heavily biased class distribution for the data sets used in the experiments), and negative root mean-squared error (-rMSE) for regression tasks.
2. **% Less Tasks Required**: this number quantifies how many less tasks (expressed as a percentage) are required for the active task selection method to achieve and maintain a particular performance level compared to the number of tasks required for random task selection to achieve and maintain that same performance level. A score of 0 indicates that a particular active task selection method is no more efficient than random selection, a negative score indicates that it is *less* efficient, and a positive score indicates that it is *more* efficient. If a particular run did not achieve a specific level of performance, then the number of tasks required to achieve that level of performance was set to one plus the total number of training tasks.

Results

Figure 2 depicts our results, showing that selecting tasks actively is almost always more efficient than random selection. Most encouragingly, the diversity heuristic achieves large gains in efficiency over random selection on *all* data sets. In some cases, the diversity heuristic achieves equivalent performance to random selection using approximately one-half the number of tasks—a large gain in efficiency.

The InfoMax methods, A-optimality and D-optimality, also show strong gains in efficiency over random selection. The results for these methods are not as consistent as those for the diversity heuristic, since they can lead to inefficient task selection in some cases on the two classification data sets. However, D-optimality achieves the best efficiency of all approaches on the London Schools data set.

Recall that the diversity++ heuristic is a stochastic version of the diversity heuristic. Consequently, it does not always select the task with the worst performance to learn next, as does the diversity heuristic. The results show that the diversity++ heuristic was dominated in three of the four data sets by the diversity heuristic, affirming the benefits of selecting the task with the worst performance to learn next. However, both diversity++ and D-optimality perform better than the diversity heuristic on the London Schools data set. In contrast to the other data sets, the London Schools data is sparse (containing $\sim$70% zero entries). Further investigation is needed to determine whether particular characteristics of the data set, such as the degree of sparsity of its input features, are important for determining when each heuristic should be used.

Conclusion

We have extended ELLA by considering the setting of *active curriculum selection* in which a learner can select which task to learn next in order to maximize performance across future learning tasks. One of our proposed mechanisms for active curriculum selection, the *InfoMax* heuristic, did not achieve superior performance to random selection on one of four data sets, and it exceeded its performance on three. More work is needed to pinpoint the precise cause of these mixed results for this heuristic. In contrast, our proposed *diversity* heuristic is efficient to compute and performs well across all data sets investigated, achieving significant reductions in the number of tasks (up to 50%) required to obtained a particular performance level when compared to random selection.

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 on this paper.

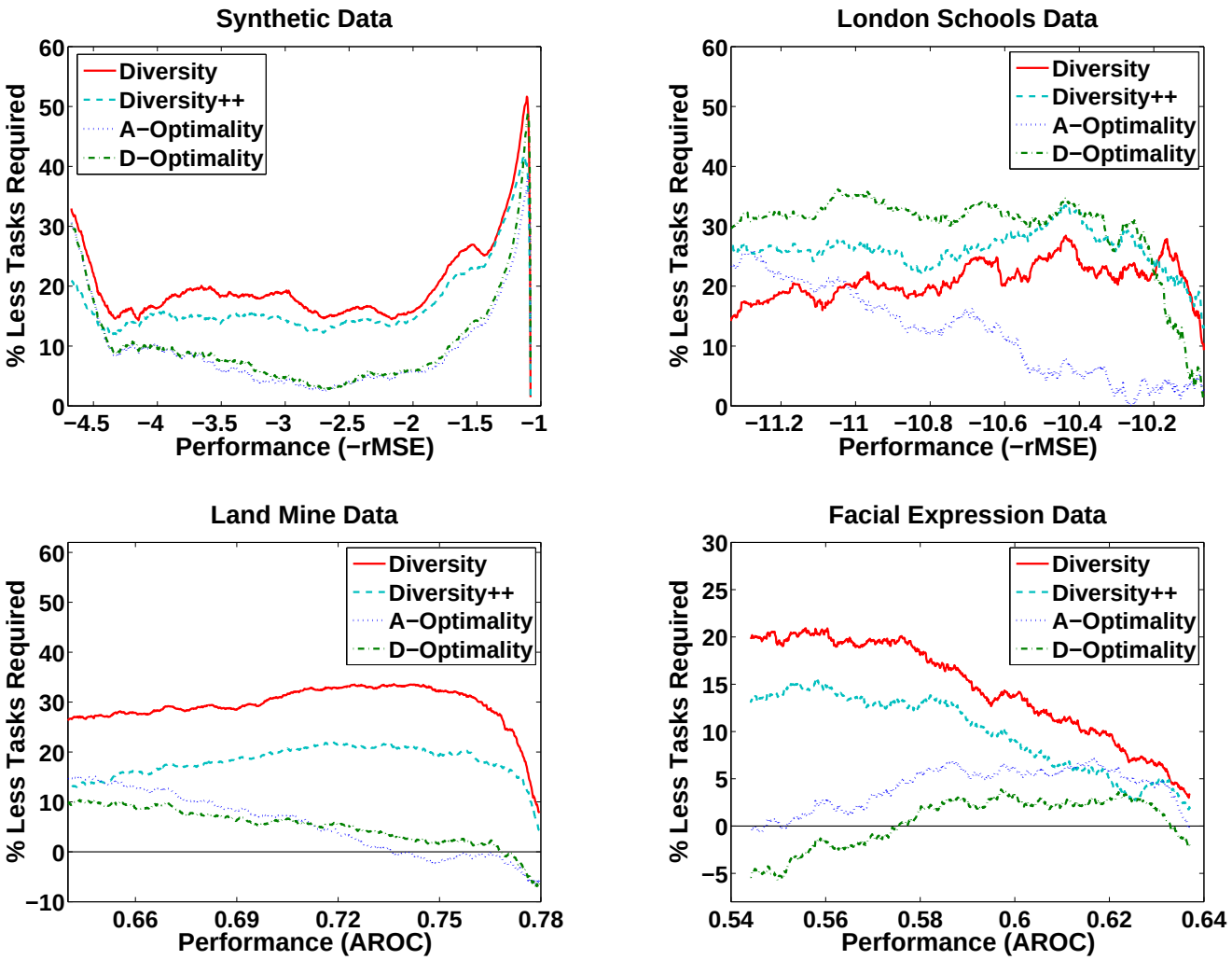

Figure 2: The results of active task selection on four different lifelong learning problems. Each plot shows the accuracy achieved by each method versus the relative efficiency (in number of tasks) as compared to random task selection. The efficiency is measured by the point at which the models' accuracy exceeds and never returns below the particular performance level.

References

Arthur, D., and Vassilvitskii, S. 2007. k-means++: The advantages of careful seeding. In *Proc. of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms*, 1027–1035. SIAM.

Bengio, Y.; Louradour, J.; Collobert, R.; and Weston, J. 2009. Curriculum learning. In *Proc. of the 26th International Conference on Machine Learning*, 41–48. ACM.

Chaloner, K., and Verdinelli, I. 1995. Bayesian experimental design: A review. *Statistical Science* 10(3):273–304.

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

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

Ruvolo, P., and Eaton, E. 2013. ELLA: An efficient lifelong learning algorithm. In *Proc. of the 30th International Conference on Machine Learning*.

Saha, A.; Rai, P.; Daumé III, H.; and Venkatasubramanian, S. 2010. Active online multitask learning. In *ICML 2010 Workshop on Budget Learning*.

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

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

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

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

Zhang, Y. 2010. Multi-task active learning with output constraints. In *Proc. of the 24th AAAI Conference on Artificial Intelligence*.