Multitask and Transfer Learning via Mixtures of Shared Sequence Generators

Eric Crawford, Joelle Pineau
Reasoning and Learning Lab, McGill University

Abstract

Multitask learning explores techniques for learning in situations wherein one is confronted with several tasks simultaneously. In such cases, it is often possible to improve performance on all tasks by sharing information, and having the learning of each task be influenced by data from all tasks. This is particularly true when there are many tasks each with a small amount of data. Thus far, the majority of work in this field has targeted the classification and regression settings, while comparatively little work has looked at multitask learning for sequential data. In this article we work towards filling this void. In particular, we identify a common form of relatedness between sequence learning tasks, wherein the sequence-generating system underlying each task is a probabilistic mixture of several more elementary sequence-generating systems, with the elementary systems shared between tasks. We then present a Bayesian algorithm capable of exploiting this form of task relatedness. We report experiments on both synthetic and real data, showing that our algorithm successfully shares data between tasks, achieving improved performance compared to all baseline algorithms. The algorithm is shown to be capable of handling sequences composed of either discrete or continuous observations, and to be capable of accommodating different assumptions with regard to the form of the elementary sequence generators.

Multitask learning seeks to improve performance on multiple related learning tasks by learning the tasks simultaneously and sharing information between them. A central notion for machine learning, though especially for multitask learning, is that of *inductive bias*. Inductive bias captures the collection of assumptions that are made, either explicitly or implicitly, about the nature of the world when applying a learning algorithm to obtain a hypothesis from training data. Importantly, inductive bias is the reason that generalization is possible at all; without making such assumptions (and having them be true), it is impossible for data generated from a system to be informative about what kind of data might be generated from that system in the future.

A common example of an inductive bias in regression settings is that the function to be learned is smooth. This assumption gives us power, since when we are trying to learn a function from a collection of data, it allows us to rule out those functions which would not be smooth on the training data set. If we throw this assumption away, then we obtain no generalization power from the training data, since the values of the function at input points other than those seen during training are completely unconstrained by the training data points.

In multitask learning, inductive bias plays a role at two separate levels. First, it has the role that is plays in all machine learning, influencing the manner in which training data from each task affects our selection of a hypothesis for that task. Second, it plays a role at a higher level of abstraction, influencing the way in which data from each task affects our selection of hypotheses for *other* tasks.

Coming up with new mechanisms for sharing data between tasks, then, generally consists of coming up with new forms of inductive bias, at this second level of abstraction, that will be commonly encountered in real data.

One mechanism for sharing information between tasks that has been successfully applied to both classification and regression problems consists of assuming that the true hypothesis for each task can be expressed as a sparse linear combination of a shared collection of basis vectors. We can think of the shared basis vectors as a common "dictionary" of vectors, and so algorithms that make use of this approach are often called dictionary learning algorithms. One instance of this idea is Grouping and Overlap Multitask Learning (GO-MTL) (Kumar and Daume III, 2012), which can be applied when learning collections of linear hypotheses, and yields improved performance when there are groups of similar tasks or tasks whose hypothese "overlap". They provide an objective function and an accompanying optimization algorithm which effectively iden-

tifies groups of common tasks and, for each group, uses data from all tasks in the group to improve the learning of the hypothesis that the group shares. See also Zhang et al. (2008) which aims to provide a more general formulation of this overall idea. TODO: This paper actually covers a framework wherein the PARAMETERS are linear combinations of a basis. Are we making the same assumption? Not quite. If we transferred that over to the sequence learning case, we would get something more in line with the Simplicial mixtures of markov chains.

An alternative to regression or classification is *sequence learning*. In single task sequence learning, we are given, as training data, a collection of sequences of observations generated by some system. The learning task is to identify a function which, given the beginning of a sequence generated from the same system, is capable of predicting the next observation or the remainder of the sequence. In multitask sequence learning, we are given training sequences generated by several different systems, and at test time, given the start of a sequence as well as an indication of which system it was generated from, predict the remainder of the sequence. If the systems are related in some way, then we can hope to use data from all the systems to improve our overall performance.

The current work extends the dictionary learning idea to the sequence learning setting, providing a novel mechanism for sharing information between sequence learning tasks. In particular, we assume that the system generating sequences for each task is actually a mixture of a smaller dictionary of elementary systems. This turns out to be equivalent to assuming that each task's probability distribution over sequences is a convex combination of a dictionary of basis probability distributions, further echoing the dictionary learning assumptions. We propose a Bayesian algorithm for exploiting such collections of tasks which is closely related to Latent Dirichlet Allocation (a technique for extracting topics from collections of natural language documents).

Multitask Sequence Learning

Multitask sequence learning can be formalized as follows. Assume we are given T sequence prediction tasks. The training data for each task consists of a collection of N sequences. From this training data we wish to select a collection of hypotheses (or a single "meta"-hypothesis) $\{H_t\}_{t=1}^{T}$ which will provide the most accurate predictions on a test dataset containing data from each task.

Mixtures of Shared Sequence Generators

When transplanting LDA to the sequence learning setting, one obtains a model which we call **SequentialLDA**. * We have a set of elementary sequence generators, $G^{(i)}(x_1 \dots x_n)$, where $G^{(i)}()$ is a probability distribution over strings. Each task is a convex combination (or mixture) $\theta^{(t)}$ of these elementary generators. To generate a sequence for task t, we first select an elementary generator according to $\theta^{(t)}$ (selecting $G^{(i)}$ with probability $\theta^{(t)}(i)$, and generate the entire sequence according to $G^{(i)}$. The elementary sequence generators do not necessarily need to be simple, though learning will progress more quickly if they are.

This flexible framework can be used to model a variety of situations. One possibility is viewing each task as a sensor, and each of the elementary generators as an entity with a unique behaviour profile. In this sitation, each generated sequence can be regarded as an encounter between an entity and a sensor, with each sensor encountering the entities with a different frequency captured by $\theta^{(i)}$. Another possible viewpoint is that the tasks are behaving agents and the generators are stereotyped behaviours, with each agent exhibiting a different combination of the basic behaviours.

In any case, this model is intuitively amenable to multitask learning because we can hope to leverage data from all tasks to learn about the elementary generators, while simultaneously learning which combination of elementary generators each task exhibits. In the next section we outline a Bayesian algorithm which capitalizes on this intuition.

Markov Chain Topics Modelling topic-conditional distributions using sequence models requires new notation. Let $\Sigma = [S]$ be our alphabet, where $S \in \mathbb{N}$ is the number of symbols. Denote the set of finite-length strings by $\Sigma^* = \bigcup_{k=0}^{\infty} \Sigma^k$. There are many ways of viewing Markov Chains, but for our purposes we will say that a Markov Chain is a tuple $\langle \pi, T \rangle$ where $\pi \in \mathbb{R}^{|\Sigma|}$ is a vector such that $\pi(i)$ gives the probability that the first symbol is i, $T \in \mathbb{R}^{|\Sigma| \times |\Sigma|}$ is a matrix such that $T_{i,j} = T(j|i)$ gives the probability of emitting symbol j given we have just emitted symbol i and $h \in \mathbb{R}^{|\Sigma|}$ is a vector such that $h(i)$ gives the probabilty of halting given that we have just emitted symbol i. Also, all parameters are between 0 and 1, and $\sum_{j=1}^{|\Sigma|} T(j|i) + h(i) = 1$. A Markov Chain defines a distribution over strings of arbitrary length (elements of Σ^*) with a probability mass function given by $P(x) = \pi(x_1) \prod_{i=1}^{n-1} T(x_{j+1}|x_j) h(x_n)$, assuming $x = x_1 x_2 \dots x_n \in \Sigma^*$. Alternatively, a Markov Chain can be seen to specify a distribution over strings of a fixed length L (elements of Σ^L) by setting $h(i) = 0 \; \forall i$, and taking $P(x) = \pi(x_1) \prod_{i=1}^{L-1} T(x_{j+1}|x_j)$, corresponding to artificially halting the generation process after L symbols, though we will not make use of this formulation in the current work.

A Markov Chain is a relatively compact way of specifying a distribution over strings of arbitrary length, requiring just $(|\Sigma| + 1)(|\Sigma| - 1)$ parameters (since each

of the $|\Sigma| + 1$ multinomial distributions that make up a Markov Chain requires $|\Sigma| - 1$ parameters). The price paid for this representational efficiency is of course reduced expressivity; the space of string distributions that can be described via a Markov Chain is a small subset of the space of all possible string distributions.

Hidden Markov Model Topics

Kalman Filter

Recurrent Neural Network

Relation to Latent Dirichlet Allocation

TODO: Redo this section taking into account its new position. We first briefly introduce Latent Dirichlet Allocation, which is the basis for our model. LDA was originally introduced as a means of inferring latent topics in a collection of documents. Each topic is a multinomial distribution over all possible words, and each document is taken to be a convex combination of the topics. Given a fixed set of parameters consisting of a concentration parameter $\alpha \in \mathbb{R}$ and a topic matrix $\beta \in \mathbb{R}^{K \times W}$ where the k-th row $\beta_{k,:}$ is a topic specifying a multinomial distribution over words, LDA assumes that the i-th document in the corpus is generated as follows. First, select topic mixture weights $\theta_i \sim \text{Dirichlet}(\alpha)$. Then for each word location $j \in [N]$, choose a topic for that location $z_{ij} \sim \text{Multinomial}(\theta_i)$, and generate a word the chosen topic $w_{ij} \sim \text{Multinomial}(\beta_{z_{ij},:})$. The graphical model for this generative process is depicted in Fig. **??**.

During training, α and β are unknown; we are given a corpus of T documents and our goal is to learn α and β for the corpus and θ_i for each document. This can be achieved by a number of algorithms; later in the paper we outline the original variational expectation-maximization algorithm. The learned model can then be used for *document completion* where we predict other likely words in the documents we have already seen, *document evaluation* where we evaulate the likelihood of previously unseen documents (the model should assign high likelihood to documents that are part of the corpus but were held out of the training data for testing purposes) and *document classification*, where a document's distribution over topics θ_i is taken as a summary of the information contained therein and used as a description of the document for various classification tasks. As suggested by this last task, LDA has been called "multinomial PCA" by some authors, highlighting the fact that the topics a document contains can be taken to be a low-dimensional representation of the information that it contains.

The central insight of the current paper is that the LDA model can be used for multitask learning in many different learning settings by making an analogy between documents and tasks, and changing the family of

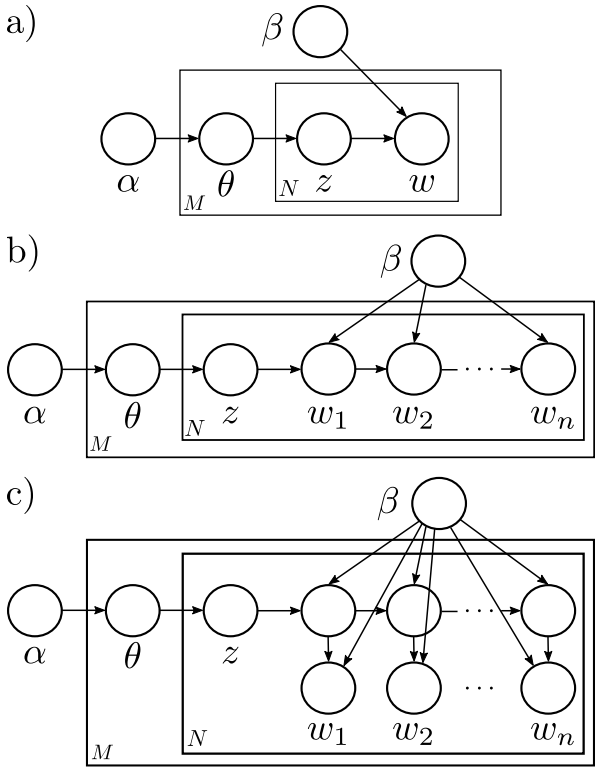

Figure 1: Graphical Models for a) LDA, b) LDA with Markov Topics, and c) LDA with HMM Topics.

distributions used as topics, replacing the multinomials used in vanilla LDA to another family that is more suitable for the application of interest. In the current work, we are interested in multitask *sequence* learning, and consequently we take the topics to be sequence models (e.g. markov chains and hidden markov models).

There are at least two benefits to making this switch: 1) it becomes possible to use the model for making predictions *within* a word/sequence (i.e. predicting the next character in a word/sequence) which is not readily possible when using a multinomial distribution over words/sequences, and 2) if the sequence generation model realizes a set of distributions over words that is smaller than the space of all possible distributions, then this can provide a form of regularization which may reduce the sample complexity of learning the model.

If one were alternatively interested in real-valued data, then the topic distributions could be Gaussians, and each document/task would consequently be modelled as a Gaussian mixture model, with each mixture component being shared by multiple documents/task.

Prediction

Algorithm

Modelling Topics as Sequence Generators

Results

Discussion and Related Work

Notation

- $[n] = \{1, 2, \ldots, n\}$ for $n \in \mathbb{N}$
- Δ^{k-1}: the $k-1$ dimensional probability simplex. Contains all discrete distributions over k options.
- T: number of documents/tasks
- N: number of words/sequences in each document
- K: number of topics
- V: number of unique words
- $\theta_i \in \Delta^{K-1}$: a discrete distribution over topics for the i-th document
- $\theta = \{\theta_1, \ldots, \theta_M\}$
- $\beta_k \in \Delta^{V-1}$: a discrete distribution over words for the k-th topic
- $\beta = \{\beta_1, \ldots, \beta_K\}$
- $z_{ij} \in [K]$: topic of the j-th token in the i-th document
- $Z_i = \{z_{i1}, \ldots, z_{iN}\}$: topic variables for the i-th document
- $Z = \{Z_1, \ldots, Z_M\}$: topic variables for the entire corpus
- $w_{ij} \in [V]$: word type of the j-th token in the i-th document
- $W_i = \{w_{i1}, \ldots, w_{iN}\}$: word types for the i-th document
- $W = \{W_1, \ldots, W_M\}$: word types for the entire corpus
- $\alpha \in \mathbb{R}_+$ or $\mathbb{R}_+^K$: parameter for the Dirichlet distribution over topic distributions

Learning Algorithm

Variational Expectation-Maximization for LDA

In this section we review the original variational expectation-maximization algorithm for estimating LDA parameters α and β and latent variables θ and Z. A number of improvements on this basic algorithm have been presented, including Gibbs Sampling, Collapsed Gibbs Sampling, etc., some of which could applied to our algorithm with minor modifications.

Assume we have been given a corpus of T documents, call this W. Our goal is to learn values for parameters α and β which maximize the log-likelihood of the observed corpus. The other unobserved variables, Z and θ, are treated as latent variables for which we will also obtain estimates. We can formulate the learning problem as one of maximizing the log-likelihood of the observed documents:

$$\max_{\alpha,\beta} \log P(W|\alpha, \beta) \tag{1}$$

Because of the latent variables Z and θ, this cannot be optimized directly. Instead, we derive a lower-bound on the log likelihood and maximize that. Let Q be an arbitrary distribution over the latent variables. Then it can be shown that:

$$\log P(W|\alpha, \beta) \tag{2}$$

$$= E_{Z,\theta\sim Q}\left[\log \frac{P(Z, \theta, W|\alpha, \beta)}{Q(Z, \theta)}\right] - \tag{3}$$

$$E_{Z,\theta\sim Q}\left[\log \frac{P(Z, \theta|\alpha, \beta)}{Q(Z, \theta)}\right] \tag{4}$$

$$= \mathcal{L}(\alpha, \beta, Q) + D_{KL}(Q(\cdot, \cdot)||P(\cdot, \cdot|\alpha, \beta)) \tag{5}$$

Since the KL-divergence is non-negative, $\mathcal{L}(\alpha, \beta, Q)$ is a lower bound on the log-likelihood; thus it is often called a *variational lower bound*. In EM, $\mathcal{L}$ is maximized by co-ordinate ascent. In particular, on the E-step $\mathcal{L}$ is maximized with respect to Q, while in the M-step $\mathcal{L}$ is maximized with respect to the parameters α and β.

E-step To perform the E-step, we observe that $\log P(W|\alpha, \beta)$ does not depend on Q, so $\mathcal{L}$ is maximized with respect to Q when the KL-divergence term is minimized, which is achieved when $Q(Z, \theta) = P(Z, \theta|W, \alpha, \beta)$. If $P(Z, \theta|W, \alpha, \beta)$ is tractable, then the E-step just consists of making this assignment. However, in case of LDA, $P(Z, \theta|W, \alpha, \beta)$ is not tractable. Instead, we perform a variational approximation; we select a tractable, parameterized family of distributions, and then perform free-form optimization with respect to those parameters to obtain the parameter setting that minimizes the KL-divergence term. This approximation is what makes it a *variational* EM algorithm.

We select a family of distributions in which the $\theta_i \perp\!\!\!\perp z_{i,j} \forall j \in [N]$ and $z_{ij} \perp\!\!\!\perp z_{i\ell} \forall j \neq \ell$. That is, within a given document, we are explicitly ignoring dependencies both between the topic distribution θ_i and topic variables z_{ik}, and between pairs of topic variables. The

distribution takes the form:

$$Q(Z, \theta|\gamma, \phi) \tag{6}$$

$$= \prod_{i=1}^{M} Q(Z_i, \theta_i) \tag{7}$$

$$= \prod_{i=1}^{M} \left(Q(\theta_i|\gamma_i) \prod_{j=1}^{N} Q(z_{ij}|\phi_{ij}) \right) \tag{8}$$

$\gamma_i \in \mathbb{R}_+^K$ is a set of variational parameters picking out a Dirichlet distribution for θ_i. Likewise, $\phi_{ij} \in \Delta^{K-1}$ is a set of variational parameters picking out a Multinomial distribution for z_{ij}. The E-step then consists of solving the following optimization problem for the i-th document:

$$(\gamma_i^*, \phi_i^*) = \arg\min_{\gamma_i, \phi_i} D_{KL} \left(Q(\cdot|\gamma_i, \phi_i) || P(\cdot|\alpha, \beta) \right) \tag{9}$$

Blei et al. (2003) derive a co-ordinate descent algorithm for performing this optimization, whose updates take the form:

$$\phi_{ijk} \propto \beta_{k,w_{ij}} \exp\{E_q[\log(\theta_{ik})]\} \tag{10}$$

$$= \beta_{k,w_{ij}} \exp\{\Psi(\gamma_{ik}) - \Psi(\sum_{\ell=1}^{K} \gamma_{i\ell})\} \tag{11}$$

$$\gamma_{ik} = \alpha_k + \sum_{j=1}^{N} \phi_{ijk} \tag{12}$$

where $\Psi(\cdot)$ is the Digamma function. Each update corresponds to completely minimizing the KL-divergence w.r.t. one of the variables. For a single E-step, these updates are repeated until convergence.

M-step Recall that the M-step consists of maximizing $\mathcal{L}(\alpha, \beta, Q)$ with respect to α and β. For β, the updates take the form:

$$\beta_{kw} \propto \sum_{i=1}^{M} \sum_{j=1}^{N} \phi_{ijk} \mathbb{1}_{w_{ij}=w} \tag{13}$$

Basically, for each topic, the probability of each word is a normalized version of the sum of all the ϕ values for that topic-word combination. This comes from computing an M-projection of the distribution obtained by normalized the sum-of-phi's onto the space of multinomial distributions (i.e. the space of all discrete distributions; therefore no projection is really done at all). $L(\alpha, \beta, Q)$ is optimized w.r.t α using a fast Newton-Raphson method.

E-STEP(W, β, α)

for $i = 1$ **to** M
  $\gamma_{ik}^0 = \alpha_k + N/K$ for all $j \in [K]$
  $\phi_{ijk}^0 = 1/K$ for all $j \in [N], k \in [K]$
  $t = 0$
  while not converged
    for $j = 1$ **to** N
      for $k = 1$ **to** K
        $\phi_{ijk}^{(t+1)} = \beta_{k,w_{ij}} \exp\{\Psi(\gamma_{ik}^{(t)})\}$
      Normalize $\phi_{ij\cdot}$ to sum to 1
    for $k = 1$ **to** K
      $\gamma_{ik}^{(t+1)} = \alpha_k + \sum_{j=1}^{N} \phi_{jk}^{(t+1)}$
    $t = t + 1$
return (γ, ϕ)

M-STEP(W, β, α)

for $k = 1$ **to** K
  for $w = 1$ **to** M
    for
return (γ, ϕ)

LATENT-DIRICHLET-ALLOCATION(W, α, β)

$t = 0$
while not converged
  $(\gamma, \phi) =$ E-STEP(W, α, β)
  $(\alpha, \beta) =$ M-STEP(W, γ, ϕ)
  $t = t + 1$

Topics as Sequence Generators

To modify the LDA learning algorithm to work with different families of topic-conditional distributions, we need only modify the M-step. For vanilla LDA, the M-step update described above arises from needing to solve the following optimization problem:

$$\max_{\beta_{k,:}} \sum_{w=1}^{V} \left(\sum_{i=1}^{M} \sum_{j=1}^{N} \phi_{ijk} \mathbb{1}_{w_{ij}=w} \right) \log P_{\beta_k}(w) \tag{14}$$

$$= \max_{\beta_{k,:}} \sum_{w=1}^{V} P_k(w) \log P_{\beta_k}(w) \tag{15}$$

where $P_{\beta_k}(w)$ is the probability of generating word w from topic k parameterized by β_k, and $P_k(w) \propto \sum_{i=1}^{M} \sum_{j=1}^{N} \phi_{ijk} \mathbb{1}_{w_{ij}=w}$. Since we have been working with multinomial topics thus far, $P_{\beta_k}(w) = \beta_{kw}$, and the solution to (15) is $\beta_{kw} = P_k(w)$. The form of optimization given in (15) is even more useful, however, because it shows us how to perform the M-step for topic-conditional distributions other than the multinomial.

Some sequence models, such as n-th order Markov Models, provide a closed form solution to (15). More expressive models that contain a notion of hidden state, such as Hidden Markov Models, can use Expectation-Maximization to obtain a local maximum of a lower

bound of the objective. For other models, the objective can be maximized by stochastic gradient descent.

References

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

Kumar, A. and Daume III, H. (2012). Learning task grouping and overlap in multi-task learning. *arXiv preprint arXiv:1206.6417*.

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