LEARNING ALGORITHMS FOR ACTIVE LEARNING

Philip Bachman*, **Alessandro Sordoni***, **Adam Trischler**
Maluuba Research
first.last@maluuba.com

ABSTRACT

We present a model that learns active learning algorithms via metalearning. For each metatask, our model jointly learns: a data representation, an item selection heuristic, and a one-shot classifier. Our model uses the item selection heuristic to construct a labeled support set for the one-shot classifier. Using metatasks based on the Omniglot and MovieLens datasets, we show that our model performs well in synthetic and practical settings.

1 INTRODUCTION

In active learning, a model selects the items for which it will observe labels, to maximize prediction performance and minimize labeling cost. Active learning is motivated by the observation that a model may perform better while training on less labeled data if it can choose the data on which it trains (Cohn et al., 1996). Previous work has proposed various heuristics for selecting items to label, such as choosing the item whose label the model is most uncertain about, or the item whose label is expected to maximally reduce the model's uncertainty about labels for other items (Settles, 2010; Gilad-Bachrach et al., 2005; Houlsby et al., 2011).

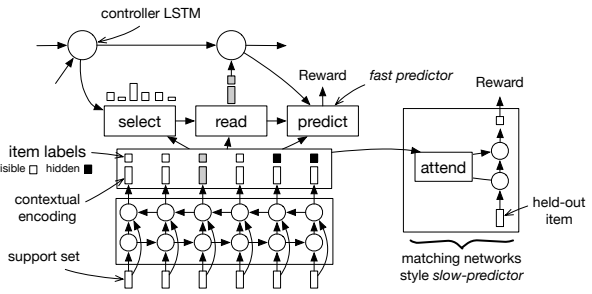
We propose moving away from engineered selection heuristics towards *learning* active learning algorithms end-to-end via *metalearning*. Our model interacts with labeled items for many related tasks in order to learn an active learning strategy for the task at hand. In recommendation systems, for example, ratings data for existing users can inform a strategy that efficiently elicits preferences for new users who lack prior rating data, thus *bootstrapping* the system quickly out of the cold-start setting (Golbandi et al., 2010; 2011; Sun et al., 2013; Kawale et al., 2015). A learned strategy could outperform task-agnostic heuristics by sharing experience across related tasks. In particular, the model's (i) data representation, (ii) strategy for selecting items to label, and (iii) prediction function could all co-adapt. Moving from pipelines of independently-engineered components to end-to-end learning has lead to rapid improvements in, e.g., computer vision, speech recognition, and machine translation (Krizhevsky et al., 2012; He et al., 2016; Hannun et al., 2014; Wu et al., 2016).

We build on the Matching Networks (MN) introduced by Vinyals et al. (2016). We extend the MN's one-shot learning ability to settings where labels are not available *a priori*. We cast active learning as a sequential decision problem: at each step the model requests the label for a particular item, then adds this item to a labeled support set, which is used for MN-style prediction. We train our model end-to-end via reinforcement learning and backpropagation. We expedite the training process by allowing our model to observe and mimic a strong selection policy with oracle knowledge of the labels. We demonstrate empirically that our model learns effective active learning algorithms for both image classification and bootstrapping a movie recommendation system from a cold-start.

2 TASK AND MODEL DESCRIPTION

We summarize our model's architecture and objectives in Figure 1, and Equation 2. Succinctly, our model iteratively picks items to label from a pool of unlabeled items, thereby constructing the labeled support set used by a Matching Network to classify test items. We train the underlying data representation, iterative item selection function, and MN end-to-end via metalearning.

Our model refines its behaviour over many *training episodes*, in order to maximize performance during *test episodes* not encountered in training. In each episode, our model interacts with a support set $S \equiv \{(x, y)\}$ comprising items x for which the model can request labels y, and a similarly-defined



Algorithm 1 End-to-end active learning loop.

$S = \{(x, y)\}$, $S_0^u = \{(x, \cdot)\}$, $S_t^k = \emptyset$, $E = \{(\hat{x}, \hat{y})\}$, s_0
 # encode support set
$X_s \leftarrow \texttt{ENCODE}(S_t^u)$
for $t = 1 \ldots T$ **do**
  # select next instance
  $j \leftarrow \texttt{SELECT}(S_t^u, S_t^k, s_{t-1})$
  # read labeled instance and update controller
  $(x_j, y_j) \leftarrow \texttt{READ}(S, j)$
  $s_t \leftarrow \texttt{UPDATE}(x_j, y_j, s_{t-1})$
  # update known / unknown set
  $S_t^k \leftarrow S_{t-1}^k \cup \{(x_j, y_j)\}$
  $S_t^u \leftarrow S_{t-1}^u \setminus \{(x_j, y_j)\}$
  # perform fast prediction
  $L_t^S \leftarrow \texttt{FAST-PRED}(S, S_t^u, S_t^k, s_t)$
end
 # perform slow prediction
$L_T^E \leftarrow \texttt{SLOW-PRED}(E, S_T^u, S_T^k, s_T)$

Figure 1: Our model's architecture and the main active learning loop. Support set items are processed with a context-sensitive encoder (a bidirectional LSTM). The selection module places a distribution over which unlabelled item to label next using a combination of controller-item similarity features and item-item similarity features. The read module extracts the item selected for labelling, and transforms it for input to the controller, which updates its state. Fast predictions are made within the support set based on sharpened item-item similarity features. Slow predictions are made for the evaluation set using a MN-style function which accounts for known/unknown labels, and conditions on the controller. We train end-to-end with Reinforcement Learning.

evaluation set $E \equiv \{(\hat{x}, \hat{y})\}$. Let $S_t^u \equiv \{(x, \cdot)\}$ denote the set of items in the support set whose labels are still unknown after t label queries, and let $S_t^k \equiv \{(x, y)\}$ denote the complementary set of items whose labels are known. Let S_t denote the joint set of labeled and unlabeled items after t label queries. Let s_t denote the *control state* of our model after viewing t labels, and let $R(E, S_t, s_t)$ denote the reward won by our model when predicting labels for the evaluation set based on the information it has received after t label queries.

At each step t of active learning, the model requests the label for an item $x \in S_{t-1}^u$. The resulting pair (x, y) is used to update s_{t-1} to s_t, and to update the labeled/unlabeled support sets to S_t^k/S_t^u. We define prediction reward as log-likelihood of predictions $p(\hat{y}|\hat{x}, s_t, S_t)$ on the evaluation set:

$$R(E, S_t, s_t) \equiv \sum_{(\hat{x}, \hat{y}) \in E} \log p(\hat{y}|\hat{x}, s_t, S_t), \tag{1}$$

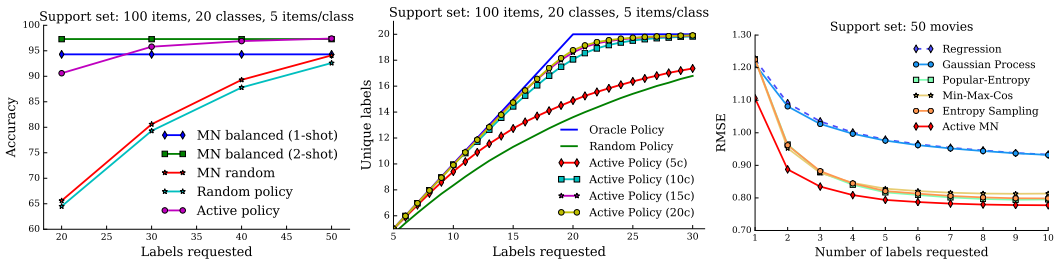
During training, our model optimizes the following objective:

$$\underset{\theta}{\text{maximize}} \; \mathop{\mathbb{E}}_{(S,E)\sim\mathcal{D}} \left[\mathop{\mathbb{E}}_{\pi(S,T)} \left[\sum_{t=1}^{T} \tilde{R}(S_t^u, S_t, s_t) + R(E, S_T, s_T) \right] \right], \tag{2}$$

in which T is the number of label queries to perform, (S, E) is an episode sampled from distribution $\mathcal{D}$, and $\pi(S, T)$ indicates unrolling the active learning policy π for T steps on support set S. Unrolling π produces the intermediate states $\{(S_1, s_1), ..., (S_T, s_T)\}$. $\tilde{R}(S_t^u, S_t, s_t)$ is a prediction reward for unlabeled items in the support set. We assume all labels are available during training. We compute $\tilde{R}(S_t^u, S_t, s_t)$ using a *fast prediction* module, and compute $R(E, S_T, s_T)$ using a *slow prediction* module. The fast prediction module applies attention using adaptively-sharpened cosine similarities among items $x_i, x_j \in S$, while the slow prediction module is (roughly) a MN. Attention sharpening for the fast predictions at step t conditions on s_t and S_t. Though the model computes only the slow predictions at test time, training relies on the fast predictions for computational reasons.

To optimize Equation 2, our model repeatedly samples an episode (S, E), then unrolls π for T steps of active learning, and maximizes the reward $\tilde{R}(E, S_t^u, s_t)$ at each step. Alternately, our model could maximize only the reward $R(E, S_T, s_T)$ at the final step. We maximize reward at each step in order to promote *anytime* behaviour – i.e. the model should perform as well as possible after each label query. Anytime behaviour is desirable in many practical settings, e.g. recommendation.

We optimize the model parameters using a combination of backpropagation and policy gradients. For a clear review of optimization techniques for general stochastic computation graphs, see Schulman et al. (2016). We expedite the training process by constructing our model's selection policy as a mixture of its learned active policy π and an "oracle" policy, which selects items to label in proportion to their current (exponentiated) prediction error. We anneal this mixture as training progresses until it contains only the learned policy. Our model thus learns (heuristic) algorithms by (noisy) demonstration. We stabilize the advantage estimates used in GAE by relying on *improvements* in prediction rewards, which effectively treats the previous step's reward as a baseline.



(a) (b) (c)

Figure 2: Results for our model on Omniglot (a, b), and MovieLens (c). (a) shows how prediction accuracy improves with the number of labels requested in a challenging 20-way setting. (b) shows the number of unique labels with respect to the number of requested labels, in a 20-way setting, for models trained on problems with different numbers of classes. The algorithms learned by our model generalize to different number of classes. (c) compares our active learning policy with several baselines on a cold-start recommendation task.

3 RESULTS

We run our first experiments on the Omniglot dataset (Lake et al., 2015), following the one-shot learning setup of Vinyals et al. (2016). We test on 20-way classification: each episode has a support set S with 5 items from each of 20 classes, and an evaluation set with 1 item per class. Character classes in training and testing are disjoint. We train our model to query 50 labels (of 100 in S). We first train the model to query 20 labels (i.e., 1-shot setting), and then fine-tune the model for 50 queries. While fine-tuning, we add an auxiliary reward which encourages a class-balanced selection policy. In the k-shot setting, we evaluate accuracy after $20 \times k$ items. We compare three baselines: our model with a random selection policy, a MN whose support set is a labeled random subsample of S (MN-random, a lower bound on performance), and a MN whose support set is a class-balanced subsample from S (MN-balanced, a near optimal policy). In Fig. 2(a), our model falls short of the class-balanced baseline in the one-shot setting, but comes closer in the two-shot setting. In all cases our model significantly outperforms the random selection baselines. In Fig. 2(b), we examine the generalization performance of our model when testing on problems with a different number of classes than were present in training. We train active learning policies for 5-way, 10-way, 15-way, and 20-way classification. We test all policies on 20-way classification, and measure how well they approximate the (near) optimal policy which collects a class-balanced set of labels. Aside from the 5-way-trained policy, our model's algorithms generalize well and outperform the random policy baseline.

We run additional experiments using the MovieLens-20M dataset,[1] which contains real movie ratings from real users. Each episode samples a user u, then 50 movies rated by u to include in the support set and 10 movies rated by u for the evaluation set. We minimize mean-squared rating prediction error. We pretrained movie embeddings for all models using a standard matrix factorization approach on the training set. We compare against several baselines in Figure 2(c) (results are averaged over 3 different random seeds). The Regression baseline performs regularized linear regression on movies from the support set whose ratings have been observed up to time t, where labels are observed in random order. The Gaussian Process baseline selects the next movie to label in proportion to the variance of the predictive posterior distribution over its rating. This gives an idea of the impact of using MN one-shot capabilities rather than standard regression techniques. The Popular-Entropy, Min-Max-Cos, Entropy Sampling baselines train our model end-to-end, but using a fixed selection policy. This gives an idea of the importance of learning the selection policy. Popular-Entropy (Elahi et al., 2016) aims to collect ratings for movies that are popular and have been rated differently by different users. Min-Max-Cos selects the unrated movie which has minimum maximum cosine similarity to any of the rated movies. Roughly, this selects the unrated movie which differs most from the rated movies. Entropy Sampling selects movies in proportion to rating prediction entropy. Our model outperforms the baselines in terms of RMSE, particularly after requesting only a few ratings. After 10 ratings, our model achieves an improvement of 2.5% in RMSE against the best baseline. Our model can be interpreted as learning a compact parametric representation of a decision tree for bootstrapping the movie recommendation system from a cold-start setting, as proposed by Sun et al. (2013).

[1] Available at `http://grouplens.org/datasets/movielens/`

REFERENCES

David A Cohn, Zoubin Ghahramani, and Michael I Jordan. Active learning with statistical models. *Journal of artificial intelligence research*, 4(1):129–145, 1996.

Mehdi Elahi, Francesco Ricci, and Neil Rubens. A survey of active learning in collaborative filtering recommender systems. *Computer Science Review*, 20:29–50, 2016.

Ran Gilad-Bachrach, Amir Navot, and Naftali Tishby. Query by committee made real. In *Proceedings of the 18th International Conference on Neural Information Processing Systems*, pp. 443–450. MIT Press, 2005.

Nadav Golbandi, Yehuda Koren, and Ronny Lempel. On bootstrapping recommender systems. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, CIKM '10, 2010.

Nadav Golbandi, Yehuda Koren, and Ronny Lempel. Adaptive bootstrapping of recommender systems using decision trees. In *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining*, WSDM '11, 2011.

Awni Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, and Andrew Y. Ng. Deep speech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567v2*, 2014.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Neil Houlsby, Ferenc Huszár, Zoubin Ghahramani, and Máté Lengyel. Bayesian active learning for classification and preference learning. *arXiv preprint arXiv:1112.5745*, 2011.

Jaya Kawale, Hung H Bui, Branislav Kveton, Long Tran-Thanh, and Sanjay Chawla. Efficient thompson sampling for online matrix-factorization recommendation. In *Advances in Neural Information Processing Systems (NIPS)*. 2015.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2012.

Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

John Schulman, Nicolas Heess, Theophane Weber, and Pieter Abbeel. Gradient estimation using stochastic computation graphs. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

Burr Settles. Active learning literature survey. *University of Wisconsin, Madison*, 52(55-66):11, 2010.

Mingxuan Sun, Fuxin Li, Joonseok Lee, Ke Zhou, Guy Lebanon, and Hongyuan Zha. Learning multiple-question decision trees for cold-start recommendation. In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*, WSDM '13, 2013.

Oriol Vinyals, Charles Blundell, Tim Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 3630–3638, 2016.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.