LEARNING ACTIVE LEARNING IN THE BATCH-MODE SETUP WITH ENSEMBLES OF ACTIVE LEARNING AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Supervised learning models perform best when trained on a lot of data, but annotating training data is very costly in some domains. Active learning aims to choose only the most informative subset of unlabelled samples for annotation, thus saving annotation cost. Several heuristics for choosing this subset have been developed, which use fixed policies for their choice. They are easily understandable and applied. However, there is no heuristic performing optimal in all settings. This led to the development of agents learning the best selection policy from data. They formulate active learning as a Markov decision process and apply reinforcement learning (RL) methods to it. Their advantage is that they are able to use many features and adapt to the specific task.

Our paper proposes a new approach combining these advantages of learning active learning and heuristics: We propose to learn active learning using a parameterized ensemble of agents, where the parameters are learned using Monte Carlo policy search. As this approach can incorporate any active learning agent into its ensemble, it allows to increase the performance of every active learning agent by learning how to combine it with others.

1 INTRODUCTION

Supervised machine learning systems perform best when trained on a large amount of training data. Obtaining this data by labelling can cause huge time and cost efforts in some domains. Active learning in the selective scenario overcomes this bottleneck by selecting a subset of all unlabelled samples to be labelled such that the model trained on them learns as much as possible and achieves a high accuracy (Cohn et al., 1994).

Heuristic active learning agents choose the samples to be labelled using a fixed policy. They have a known and predictable behaviour and their policy is easy to understand (Settles et al., 2008). However, they have two main disadvantages: First, they rarely combine different features. Second, it was found that the best heuristic highly depends on the dataset and supervised learning model used (Lowell et al., 2018).

More recently, these shortcomings have been addressed by learning active learning directly from data (Konyushkova et al., 2017; 2018; Bachman et al., 2017; Fang et al., 2017; Liu et al., 2018b;a). The authors formulate active learning as a Markov decision process and apply reinforcement learning (RL) methods like Q-Learning and imitation learning to it. While this approach promises to overcome the advantages of heuristics it introduces new problems: There is the credit assignment problem (Minsky, 1961), the training is computationally very costly (Amodei et al.), and many results are not significant and hard to reproduce (Henderson et al., 2017).

Learning active learning with RL in the batch-mode setting has only received little attention in literature. One of the reasons is that choosing a batch of samples instead of a single one makes the action space exponentially bigger and thus finding the action maximizing a value function can not be done by iterating over all actions anymore. Furthermore, it makes it harder to attribute the reward to a specific parameter of the policy.

Our paper addresses these shortcomings by proposing an active learning agent being a weighted ensemble of other agents. The weights are learned using Monte Carlo policy search and black-box optimization. As several different agents can be used as part of the ensemble, many different features can be included. By learning the weights of each agent, the ensemble can adapt to the dataset, model and optimization metric. Nonetheless, the learning is very robust and global, as black-box optimization over a small number of parameters is much easier than reinforcement learning. Furthermore, the policy is easily understandable and interpretable. The ensemble should not be seen as an alternative approach to current approaches, but rather as an extension of them allowing to learn how to combine several approaches to further increase their performance.

We evaluate our approach using active learning tasks from different domains and with random forests, CNNs and LSTMs as classifiers. The experiments show that the ensemble consistently performs at least as good as the best agent it includes and sometimes even outperforms it by a significant margin.

Our main contributions are:

- We propose an approach to learning active learning using an ensemble of heuristics. It combines the advantages of heuristics and approaches learning active learning. The experiments show that the theoretical advantages also translate into a high empirical performance.
- We show with our experiments that it is very important to train the ensemble on a similar task it is evaluated on. This is contrary to assumptions in earlier literature that a learning agent trained on a synthetic dataset works well in completely different domains.

2 Related work

2.1 Active learning heuristics

Heuristic frameworks are active learning frameworks relying on engineered, fixed policies. Their performance depends highly on the active learning task, with no heuristic being able to outperform the others in all cases. There are three core ideas which kind of samples should be chosen to be labelled. They can be clearly distinguished, as they rely on disjoint sets of features:

Informativeness sampling: One group of heuristics prefers to choose informative samples, which can be expected to change the supervised learning model a lot when added to the labelled set. Their features for a sample can be calculated only given the current supervised learning model and the sample itself. The most popular heuristic in this group is uncertainty sampling (Lewis & Gale, 1994; Scheffer et al., 2001; Shannon, 1948), while many others exist. **Diversity sampling:** This approach chooses samples which are dissimilar to already labelled samples. Its features are similarity metrics from a sample to the current labelled set. It is nearly always combined with uncertainty based sampling. **Representative sampling:** This approach, also called density-based sampling, chooses samples which are representative for the whole dataset. In particular that means that outliers should not be chosen. Like diversity sampling, it is usually combined with uncertainty sampling.

The combination of features or heuristics from these three disjoint groups is used by many researchers to increase the performance of active learning agents (Wang et al., 2017; Sener & Savarese, 2017; Zhu et al., 2009). There are many ways to combine heuristics, e.g. it is possible to first choose a subset of candidate points using one heuristic and then choose the final points to be labelled using the other heuristic, or one could directly define a heuristic combining two existing ones. While our approach with a learned ensemble is the first known approach to learn the best combination of heuristics, there exist many ideas in current literature, which could improve the performance of the proposed ensemble even further:

Wang et al. (2017) have added a diversity constraint to uncertainty sampling, and use a Quadratic Programming approach to enforce this constraint. Sener & Savarese (2017) have described active learning as a core-set-selection problem. Thus, they try to choose samples which are dissimilar both to each other and the labelled set and representative of the unlabelled set. These two approaches share that they propose advanced non-greedy methods for choosing a batch of samples. A greedy choice is only a $1 - \frac{1}{e}$-algorithm, as already pointed out by Kirsch et al. (2019). Adapting our

approach such that the samples are chosen non-greedily using the method by Wang et al. or Sener and Savarese is thus a promising future research direction.

Zhu et al. (2009) have proposed a *sampling by uncertainty and density* approach choosing the samples whose product of their uncertainty and density metric is highest. This idea can be used to extend the simple linear combination of heuristics we proposed: The product of the utility values of two heuristics or any other combination of them can easily be included as the utility of an additional heuristic of the ensemble.

Ash et al. (2019) have proposed a batch-mode AL agent choosing samples such that their gradients are diverse and used the k-means++ algorithm to choose the batch to be labelled. The diversity of gradients could be added as an additional feature for the ensemble we propose.

Active learning agents may perform better if they use different strategies in the beginning than in the end of the active learning episode. Tang & Huang (2019) have shown that it helps if the active learning agent chooses easier samples in the beginning and harder samples only in the end. This idea could also be used to extend our approach, by letting it learn a schedule of policies instead of having a constant policy.

2.2 Learning active learning

As the MDP of active learning is unknown for new tasks and datasets, methods to find the optimal policy given this MDP must rely on data. Thus, dynamic programming is not feasible, instead reinforcement learning algorithms are the suitable method for this problem. In the sequential active learning setting, the action space is a categorical one-dimensional space allowing to efficiently calculate the action maximizing a function given the action. This makes the common reinforcement learning framework of Q-Learning suitable. It fits a regression function Q(observation, action) to the training data, which tries to predict the reward given the observation for each action. Its policy $\pi(observation)$ is simply choosing the action maximizing the Q-Value given this observation. This approach is used by Konyushkova et al. (2017; 2018); Bachman et al. (2017); Fang et al. (2017). Another approach is used by Liu et al. (2018b;a): They use imitation learning and train the agent to choose the action maximizing the reward directly without estimating the reward. Their policy to imitate is a so-called algorithmic expert: It is a policy already knowing the true labels of the samples to choose and is thus able to calculate the improvement of the model if a sample is added to the labelled set. This equals evaluating the reward function without actually performing the step. They have applied this approach not only in the sequential, but also in the batch-mode active learning setting: Their agent sequentially chooses the samples to add to the batch. Our approach is also based on this idea of sequential batch-filling active learning.

The error decay can also be assumed to only depend on the number of samples in a cluster (Chang et al., 2020). They predicted the error decay using a parameterized policy depending only on this single value. Because this policy is mostly a parameterized heuristic, it only needs a small number of samples to be trained on a dataset.

Learning batch-mode active learning has also partly been covered by Ravi & Larochelle (2018): They used a Q-learning algorithm to train a regression model to predict the improvement of the accuracy if a sample is added to the labelled set. Then they multiply this quality metric with a diversity metric to gain a final expert metric. The diversity metric takes the similarity of a sample to other unlabelled samples into account. The policy is to choose the samples having the highest expert metric. While the quality metric is a learned metric, it is a design choice to maximize the product of the quality metric and the diversity metric. Thus, this approach can be seen as a mixture of an engineered heuristic and a learning agent. Their idea to use not only a combination of heuristics, but in particular a combination of a learned active learning agent and a heuristic can also be used to improve the ensemble we propose: Any learned active learning agent can be added to the set of agents making up the ensemble.

2.3 Ensembles

In supervised learning, combinations of several different models can be combined to form an ensemble. They combine the output of these models to calculate the final output. The combination itself might be learned by another supervised learning algorithm using the predictions of the other

models as input features. This approach is called stacking and usually performs better than the best method being part of the ensemble (Wolpert, 1992). Learning the weights of an ensemble of agents can also be seen as a form of stacking, which further validates our claim that an ensemble of active learning agents performs better than the single best agent being part of it.

While ensembles of supervised learning models are widely used, there has been very little research on ensembles of agents. Nonetheless, it was found that ensembles of reinforcement learning algorithms using majority voting and Boltzmann multiplication perform significantly better than single algorithms (Wiering & Van Hasselt, 2008).

Gao et al. (2018) have proposed to learn AL by learning to choose one of three heuristics at each step of the active learning process. Different to our approach, the heuristics they used and their experiments are restricted to learning network representations. Furthermore, their approach is only applied in the sequential AL setting.

3 Framework for learning batch-mode active learning

Our framework for learning batch-mode active learning is split into three parts: First, batch-mode active learning is formalized as a Markov decision process. Second, an ensemble of several different active learning is defined. Third, the parameters of this ensemble are learned using Monte Carlo policy and its objective function and optimizer are defined.

3.1 Markov decision process for sequential batch-filling active learning

The goal of every active learning agent is to choose the next batch of samples to be labelled based on features of the current active learning process such that the supervised learning model learns most. The MDP for pool-based active learning is described by the 4-tuple (S, A, P, R):

The **state space** S consists of the labelled and unlabelled set of samples and the model trained on the labelled set. As this space is extremely huge and can include millions of variables, it is very hard for an agent to learn from this state. Thus, we define an observation space, which includes features of the state which might be relevant to the active learning agent. Common metrics in this space are the predictions of a sample and metrics computed out of it, the uncertainty of the prediction and similarities or distances of unlabelled samples to each other and labelled samples. Most active learning heuristics define a single metric to rely on and do not combine them.

The **action space** A is the categorical choice of one sample or a batch of samples in the unlabelled set. As the unlabelled set becomes smaller each step, the action space is variable.

Following the frameworks for learning sequential active learning (Konyushkova et al., 2017; 2018; Bachman et al., 2017; Fang et al., 2017) we are using reward shaping (Ng & Jordan, 2003) and set the reward to be the improvement of the model accuracy after a full batch is labelled and the supervised learning model is re-trained on the new labelled set. This makes the MDP a minimal extension of existing MDPs to batch-mode AL and facilitates comparison with them.

Instead of defining the **probability** $P(s'|s, a)$ of a new state given the current state and action, we define a transition function directly mapping the current state and action to the new state. In the case of active learning this step includes the annotation of a chosen sample or batch of samples, the re-training of the model with it, the change of both the labelled and unlabelled dataset, and the increase of the currently spent annotation budget.

Choosing a batch of samples to be annotated next is a very hard task: If one wants to choose *b* samples out of a pool of *n* unlabelled samples, there are $\binom{n}{k}$ options. Thus, we set the definition of a step of one MDP for batch-mode active learning to be the choice of one sample which is added to the batch. The corresponding pseudocode for this sequential batch-filling MDP is shown in Algorithm 1. In each step, the agent chooses the next sample to be added to the batch based on the current observation. As the observation includes similarity measures of unlabelled samples to samples in the batch, the agent can choose samples which are not too close to samples already in the batch. While the unlabelled set becomes smaller every step as one sample leaves it and enters the batch, the labelled set is only updated if the batch is full (i.e. the if-condition in the pseudocode is fulfilled): Then the labelled set is extended by all samples in the batch, thus all samples in the batch are

annotated. As the labelled set was updated, the supervised learning model can be retrained on it, changing the observation significantly.

Algorithm 1 Pseudocode of the step function of a batch-mode pool-based active learning MDP

```
function stepFunction(sampleToLabel)
    D_unlabelled ← D_unlabelled \ {sampleToLabel}
    batch ← batch ∪ {sampleToLabel}
    if |batch| == batchSize then
        D_labelled ← D_labelled ∪ batch
        batch ← {}
        newAccuracy ← supervisedLearningModel.trainOn(D_labelled)
        reward ← newAccuracy − oldAccuracy
        oldAccuracy ← newAccuracy
    else
        reward ← 0
    end if
    observation ← getObservation(D_unlabelled, D_labelled, batch)
    epochFinished ← |D_labelled| + |batch| >= annotationBudget
    return observation, reward, epochFinished
end function
```

This definition of sequential batch-filling active learning combines the advantages of sequential and batch-mode active learning. The action per step is still one categorical choice like for sequential active learning, making it much easier for a learning agent to learn which sample to choose next. As the labelling is only performed for a full batch, the practical advantages of batch-mode learning are still preserved. However, there is a theoretical disadvantage: Compared to true sequential learning, the supervised learning model is not trained on the samples just chosen, and thus the features depending on the supervised learning model are not updated. Thus, it can be expected that a sequential batch-filling agent performs slightly worse than a true sequential agent, with the difference monotonically increasing with the batch size.

One shortcoming of this definition of the MDP is that the reward is only non-zero if a full batch was labelled. This makes reinforcement learning approaches like Q-learning difficult to apply, as it is hard to attribute the reward to the choice of a specific sample, i.e. a single action. This could be solved by re-training the supervised learning model after each sample added to the batch to calculate a fictive reward during the training of the RL approach. After this, the model has to be set back to the previous state to keep the state and observation independent of this step. Such an approach would make it possible to attribute each reward to exactly one sample, however, it has the disadvantage of needing —batch— times more re-trainings of the supervised learning model, thus increasing the computational complexity by the batch size. Hence, Monte Carlo policy search is more effective than approaches like Q-Learning.

A general challenge of learning active learning is that each computation of a reward needs the re-training of a supervised learning model, which is computationally very expensive for large models and complex problems. This property makes sample-efficiency an important criterion for choosing a suitable framework to learn active learning. Consequently, Q-Learning and Monte Carlo policy gradient are unsuited methods. Model-based RL is also unsuited, because predicting the next state is difficult for two reasons: First, the state space has multiple features for each of many samples and is thus very large. Second, predicting the next state essentially equals predicting the outcome of a neural network which is not easier than training the neural network itself.

3.2 Weighted ensemble of active learning agents

We propose an active learning policy that is represented by an ensemble of different active learning heuristics. The basic idea is to have several different active learning agents which all assign a measure for the utility of choosing a certain data point to be labelled next and each choose the data point with the highest utility. We evaluate this approach using an ensemble of five agents:

The **random sampling** agent samples the utility of each sample independently from the same Gaussian distribution. The **uncertainty sampling** agent defines the utility of a sample as the entropy of

its prediction vector. The **diversity sampling** agent chooses samples which are dissimilar to samples already chosen for labelling. We defined its utility measure to be the minimum euclidean distance of a sample to a sample already in the labelled or batch set. The **representative sampling** agent chooses samples which are representative of the whole data distribution. In particular it aims not to choose outliers. We defined its utility as -1 * the 5%-percentile of the euclidean distance of a sample to all other samples. The **uncertainty diversity sampling** agent defines its utility as the product of the utilities of the uncertainty and diversity sampling agent (Zhu et al., 2009). The ensemble agent chooses the sample having the highest overall utility defined as the weighted sum of the utility measure of each heuristic. The corresponding function is given in Equation 1.

$$\text{utility(ensemble)} \leftarrow \sum_{\text{i}} \beta_{\text{i}} * \text{utility(agent}_{\text{i}}) \tag{1}$$

The beta-parameters could be chosen such that the agent performs exactly like one single heuristic by setting one β to a high value and the others to zero. Thus, assuming this agent learned the best β-parameters for a task, its worst-case performance is the one of the best heuristic on this task, but it can also perform significantly better.

We do not claim that an ensemble made of these four simple heuristics outperforms state-of-the-art batch-mode active learning agents. We do claim, however, that the ensemble performs at least as good as all agents it contains, but can perform significantly better. Thus, any state-of-the-art active learning agent can be added to the set of agents making up the ensemble, which allows the ensemble to further improve the performance of this agent by combining it with others.

3.3 Choice of parameters of ensemble

The linear combination of active learning agents is a policy parameterized by the β-parameters. We decided to learn which parameters are good using Monte Carlo policy search. The objective is to maximize the performance of the supervised learning model after a complete episode. The objective function is defined in Eqn. 2 and Algorithm 2 and calculates the performance of one choice of β. Taking the mean of several active learning runs reduces the noise of the objective function. It is very hard to attribute the performance at the end of an episode to the choice of a specific action in a specific state or to the choice of β. Thus we do not solve the optimization problem using policy gradients, but treat it as a black-box function or hyperparameter optimization problem instead. Methods like Bayesian optimization, random search, grid search, tree parzen estimators, or population-based training could be used to find the optimal solution. Among them, we chose tree parzen estimators (Bergstra et al., 2013) as they are very sample-efficient and search for the global optimum.

$$\begin{aligned} &objectiveFunction(\beta) : R^n \rightarrow R \\ &with\ n\ =\ number\ of\ heuristics\ making\ up\ the\ ensemble \end{aligned} \tag{2}$$

Algorithm 2 Pseudocode for the objective function of the ensemble agent

```
Require: trainingTask, activeLearningEnvironment, EnsembleAgent
  function objectiveFunction(β)
      agent ← EnsembleAgent(β)
      for i ← 1 to numberEpisodes do
          results ← activeLearningEnvironment.run(trainingTask, agent)
          performances[i] ← results.getFinalPerformance()
      end for
      meanPerformance ← mean(performances)
      return meanPerformance
  end function
```

4 EXPERIMENTS AND RESULTS

4.1 SETTINGS

We compare our ensemble of five heuristics with each of these five heuristics and an ensemble not using the learned weights, but setting all weighs to the same value. We set the annotation budget such that the performance differences between the agents are clearly visible and set the batch size such that between 3 and 5 iterations could be performed. The starting set size was set to be proportional to the number of classes.

For training the β-parameters, we fixed the β-parameter for random sampling at 1 and chose a log-normal distribution with a mean of 1 and a standard deviation of 2 as prior for the other parameters.

The later parts of this section show the performance of the active learning agent on the tasks as plots of the classification accuracy over the number of labelled samples. The plots for the different evaluation tasks all share the same structure: The dark shaded area around the curves of each agent report the 95% confidence intervals, the light shaded areas are the standard deviations. A suffix to the agent names in the form of '_batchSize' denotes the batch size the agent used. There is no such suffix for the random, diversity and representative sampling agent, as their features are independent of the batch.

4.2 EXPERIMENTS ON UCI DATASETS

We used 11 datasets from the UCI machine learning repository (Dua & Graff, 2017). The datasets 2-breast cancer, 3-diabetes, 4-flare solar, 5-heart, 6-german, 7-mushrooms, 8-waveform, 9-wdbc were used for training and 0-adult, 1-australian, and 10-spam for evaluation, similar to Konyushkova et al. (2018). The objective function was set to be the harmonic mean of the final accuracies the ensemble reached on each of the 7 training tasks. This allows training on multiple datasets easily and thus improves generalization. The results for the 1-australian evaluation task is shown in Figure 2, the results for the other evaluation tasks are quite similar.

4.3 EXPERIMENTS ON CHECKERBOARD DATASETS

The datasets checkerboard 2x2 and rotated checkerboard 2x2 were used as training tasks to train the learning agents, the checkerboard 4x4 task was used as evaluation task. All checkerboard tasks are XOR-like binary classification tasks with two-dimensional input data. We used a random forest as a classifier and the results are shown in Fig. 2.

4.4 EXPERIMENTS ON (FASHION-) MNIST TASKS

The image classification tasks use the MNIST (LeCun et al., 2010) and fashion-MNIST (Xiao et al., 2017) dataset and a three-layer convolutional neural network implemented in keras (Chollet et al., 2015a) as classifier. Because Munjal et al. (2020) have shown that performance differences between different active learning agents may vanish when more regularization techniques are used, we tried to maximize the performance of the random baseline using following techniques: random data augmentation, dropout, L1- and L2 regularization and hyperparameter optimization using tree parzen estimators (Bergstra et al., 2013). We did not use more advanced networks like the ResNet or VGG16 because they are computationally much more expensive and tend to overfit their huge number of parameters on the very small training set size. Instead of calculating the distance metrics using the raw images, we used tSNE embeddings (Maaten & Hinton, 2008). We trained the ensemble agent on the MNIST task and evaluated it on the fashion-MNIST task, the results are shown in Fig. 2.

4.5 RESULTS ON QUESTION ANSWERING TASKS

The bAbI dataset (Weston et al., 2015) is a collection of question answering datasets of which two different challenges are used: one with a single supporting fact and one with two supporting facts. The classifier used is using a long short term memory and taken from a keras example, which follows Sukhbaatar et al. (2015). Like for the image classification tasks, the hyperparameters were optimized

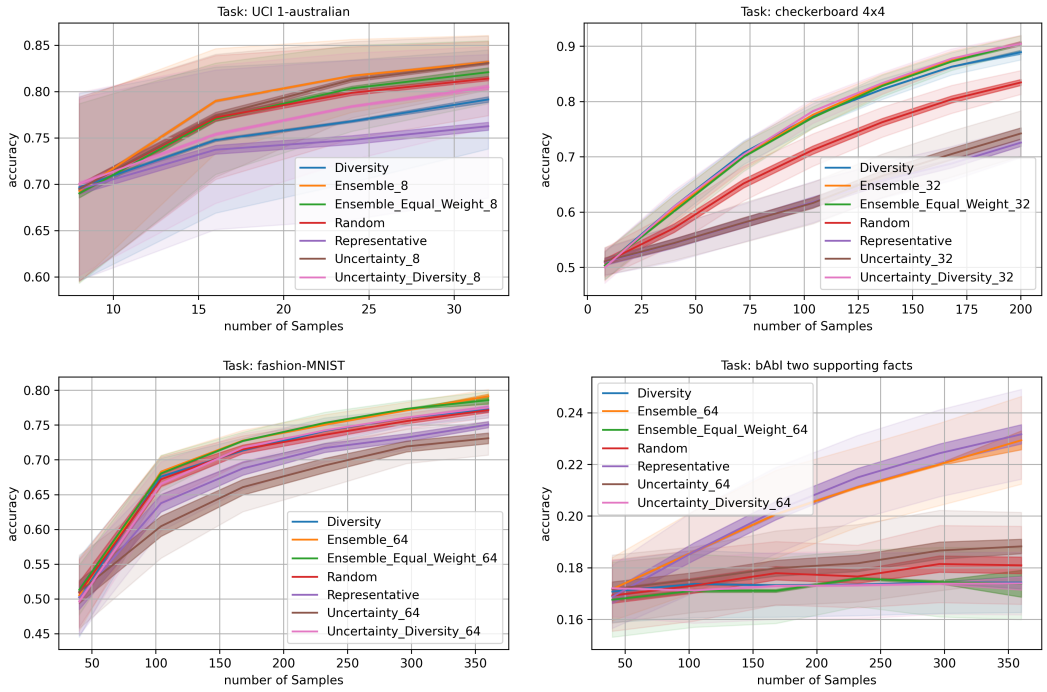

Figure 1: Results on evaluation tasks. The ensemble performs at least as good as the best heuristic, but sometimes significantly better.

to minimize the loss on a small random subset. We used the bAbI task with a single supporting fact as a training task and the one with two supporting facts as an evaluation task. For calculating the distances, we used word2vec embeddings (Mikolov et al., 2015). The results are shown in Fig. 2.

4.6 Interpretation of learned parameters

The ensemble learns the weights of the agents being part of it. These parameters can be easily interpreted and thus allow us to understand the ensemble's policy better. Furthermore they might be used to find out which kind of samples could be additionally generated, if possible. The weights learned for the four different tasks or tasks groups are given in Table 4.6.

	random	**uncertainty**	**diversity**	**representative**	**uncertainty diversity**
UCI	1.00	1.15	0.03	0.89	0.74
checkerboard	1.00	0.93	1.91	0.00	1.62
MNIST	1.00	0.50	0.40	1.80	1.70
bAbI	1.00	0.00	0.00	6643.90	21.50

Table 1: Comparison of beta-parameters learned on the different training task combinations. The greener, the higher the parameter within a row.

For the UCI datasets, diversity sampling is very unimportant, which is indicated by the weight of diversity sampling being nearly zero. Thus, it can be assumed that the samples in the UCI datasets are already quite dissimilar to each other. The ensemble trained on the checkerboard datasets has a weight of representative sampling being zero, which can be well explained by the fact that it does not contain any outliers. Both MNIST and bAbI have much more samples in the unlabelled set, explaining why representative sampling is quite important for them. The weights differ a lot between the different training tasks, indicating the importance of training the parameters of the ensemble on a similar dataset it is later applied on. a

5 CONCLUSION

In this paper, we proposed a novel approach for active learning using an ensemble of different active learning heuristics or trained agents. By learning the weights of a linear combination of different agents in the ensemble, this approach can be easily adapted to different datasets and supervised learning models. To train the ensemble we formalized batch-mode active learning as a Markov decision process. Experiments in different domains and using different supervised learning models have shown the effectiveness of our approach:

As expected, the worst-case performance of the ensemble is the one of the best heuristic, while it outperformed the best one significantly on some datasets. Furthermore, it was shown that it is very important to adapt the active learning agent to the dataset it is applied on.

The most natural extension of our ensemble is to make it up with more and more advanced active learning agents. Additionally, we assume following future research directions to be very promising: First, different agent weights could be learned for different time points of the episode, allowing to change the policy with increasing accuracy. Second, the weights could be learned on a small subset of the dataset it is later applied on. This allows its application on tasks where no similar training task is available or where the ensemble generalizes badly between the training and application task.

REFERENCES

Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. arxiv 2016. *arXiv preprint arXiv:1606.06565*.

Jordan T Ash, Chicheng Zhang, Akshay Krishnamurthy, John Langford, and Alekh Agarwal. Deep batch active learning by diverse, uncertain gradient lower bounds. *arXiv preprint arXiv:1906.03671*, 2019.

Philip Bachman, Alessandro Sordoni, and Adam Trischler. Learning algorithms for active learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 301–310. JMLR. org, 2017.

James Bergstra, Daniel Yamins, and David Daniel Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. 2013.

Haw-Shiuan Chang, Shankar Vembu, Sunil Mohan, Rheeya Uppaal, and Andrew McCallum. Using error decay prediction to overcome practical issues of deep active learning for named entity recognition. *Machine Learning*, pp. 1–30, 2020.

François Chollet et al. Keras. `https://keras.io`, 2015a.

François Chollet et al. python implementation of a memory network trained on the babi dataset, 2015b. URL `https://github.com/keras-team/keras/blob/c55ce21fdd4f55709269140e068cc840a7c7ec48/examples/babi_memnn.py`.

David Cohn, Les Atlas, and Richard Ladner. Improving generalization with active learning. *Machine learning*, 15(2):201–221, 1994.

Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

Meng Fang, Yuan Li, and Trevor Cohn. Learning how to active learn: A deep reinforcement learning approach. *arXiv preprint arXiv:1708.02383*, 2017.

Li Gao, Hong Yang, Chuan Zhou, Jia Wu, Shirui Pan, and Yue Hu. Active discriminative network representation learning. In *IJCAI International Joint Conference on Artificial Intelligence*, 2018.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. *arXiv preprint arXiv:1709.06560*, 2017.

Andreas Kirsch, Joost van Amersfoort, and Yarin Gal. Batchbald: Efficient and diverse batch acquisition for deep bayesian active learning. In *Advances in Neural Information Processing Systems*, pp. 7024–7035, 2019.

Ksenia Konyushkova, Raphael Sznitman, and Pascal Fua. Learning active learning from data. In *Advances in Neural Information Processing Systems*, pp. 4225–4235, 2017.

Ksenia Konyushkova, Raphael Sznitman, and Pascal Fua. Discovering general-purpose active learning strategies. *arXiv preprint arXiv:1810.04114*, 2018.

Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, 2, 2010.

David D Lewis and William A Gale. A sequential algorithm for training text classifiers. In *SIGIR'94*, pp. 3–12. Springer, 1994.

Ming Liu, Wray Buntine, and Gholamreza Haffari. Learning to actively learn neural machine translation. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pp. 334–344, Brussels, Belgium, October 2018a. Association for Computational Linguistics. doi: 10.18653/v1/K18-1033. URL `https://www.aclweb.org/anthology/K18-1033`.

Ming Liu, Wray Buntine, and Gholamreza Haffari. Learning how to actively learn: A deep imitation learning approach. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1874–1883, Melbourne, Australia, July 2018b. Association for Computational Linguistics. doi: 10.18653/v1/P18-1174. URL `https://www.aclweb.org/anthology/P18-1174`.

David Lowell, Zachary C. Lipton, and Byron C. Wallace. How transferable are the datasets collected by active learners? *CoRR*, abs/1807.04801, 2018. URL `http://arxiv.org/abs/1807.04801`.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

Tomas Mikolov, Kai Chen, Gregory S Corrado, and Jeffrey A Dean. Computing numeric representations of words in a high-dimensional space, May 19 2015. US Patent 9,037,464.

Marvin Minsky. Steps toward artificial intelligence. *Proceedings of the IRE*, 49(1):8–30, 1961.

Prateek Munjal, Nasir Hayat, Munawar Hayat, Jamshid Sourati, and Shadab Khan. Towards robust and reproducible active learning using neural networks. *arXiv*, pp. arXiv–2002, 2020.

Andrew Y Ng and Michael I Jordan. *Shaping and policy search in reinforcement learning*. PhD thesis, University of California, Berkeley Berkeley, 2003.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Sachin Ravi and Hugo Larochelle. Meta-learning for batch mode active learning. 2018.

Tobias Scheffer, Christian Decomain, and Stefan Wrobel. Active hidden markov models for information extraction. In *International Symposium on Intelligent Data Analysis*, pp. 309–318. Springer, 2001.

Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. *arXiv preprint arXiv:1708.00489*, 2017.

Burr Settles, Mark Craven, and Lewis Friedland. Active learning with real annotation costs. In *Proceedings of the NIPS workshop on cost-sensitive learning*, pp. 1–10. Vancouver, CA, 2008.

Claude Elwood Shannon. A mathematical theory of communication. *Bell system technical journal*, 27(3):379–423, 1948.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. End-to-end memory networks. In *Advances in neural information processing systems*, pp. 2440–2448, 2015.

Ying-Peng Tang and Sheng-Jun Huang. Self-paced active learning: Query the right thing at the right time. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 5117–5124, 2019.

Gaoang Wang, Jenq-Neng Hwang, Craig Rose, and Farron Wallace. Uncertainty sampling based active learning with diversity constraint by sparse selection. In *2017 IEEE 19th International Workshop on Multimedia Signal Processing (MMSP)*, pp. 1–6. IEEE, 2017.

Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. Towards ai-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*, 2015.

Marco A Wiering and Hado Van Hasselt. Ensemble algorithms in reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 38(4):930–936, 2008.

David H Wolpert. Stacked generalization. *Neural networks*, 5(2):241–259, 1992.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Jingbo Zhu, Huizhen Wang, Benjamin K Tsou, and Matthew Ma. Active learning with sampling by uncertainty and density for data annotations. *IEEE Transactions on audio, speech, and language processing*, 18(6):1323–1331, 2009.

A Appendix

A.1 Details of classifiers

The classifier used for both the UCI and the checkerboard datasets is the same used by Konyushkova et al. (2017): A random forest implemented with scikit-learn (Pedregosa et al., 2011) and 50 estimators. The other hyperparameters were set to the default value. We chose this classifier for easier comparison with the work by Konyushkova et al. (2017).

The classifier used for the (fashion-)MNIST task is a CNN with two convolutional layers, one intermediate dense layer and a softmax output layer. Intermediate dropout and max-pooling layers were added. The exact structure and the setting of hyperparameters can be found in the additional material in `/supervised_learning_tasks/tasks_vision/task_Vision_CNN.py:108ff`. We used random data augmentation. We had also tried the ResNet16 architecture, but found it to have worse performance, probably because its large number of parameters was overfitting on the few hundred images it was trained on. Furthermore, it was computationally much more expensive. We chose a CNN with data augmentation as classifier as it is the state-of-the-art architecture for image classification.

The classifier used for the bAbI task is a memory network (Sukhbaatar et al., 2015). The code is largely based on a keras example (Chollet et al., 2015b). We chose this architecture as it provides state-of-the-art performance.

A.2 PERFORMANCE OF AL AGENTS WHEN TRAINED ON THE FULL DATASET

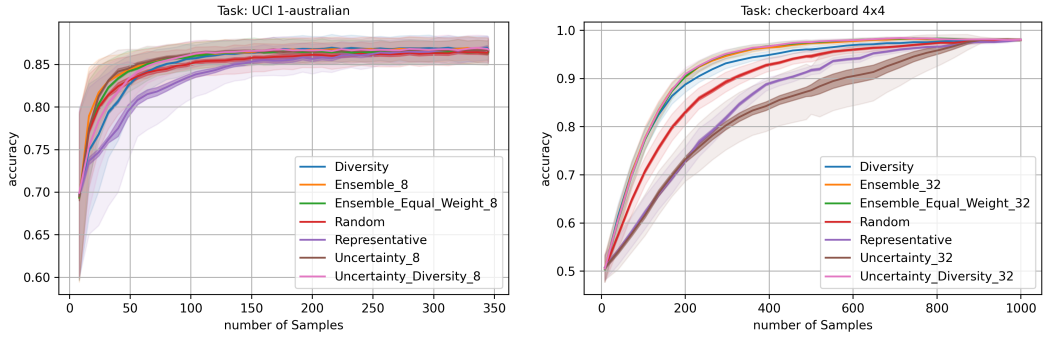

Figure 2: Results on UCI and checkerboard task with the annotation budget set to the full dataset. The worst agents need about 3 times more samples to reach the same accuracy as the best agents.