Deep Multi Armed Bandit

Alex Nowak
Mathematiques Vision Apprentissage
ENS Cachan
alexnowakvila@gmail.com

Paul Bertin
Mathematiques Vision Apprentissage
ENS Cachan
paul.f.bertin@gmail.com

Abstract

This work is done for the course *Reinforcement Learning* taught by ALESSANDRO LAZARIC in the master *Mathématiques Vision Apprentissage* at ENS Cachan. We investigate several neural approaches to tackle the Contextual Multi Armed Bandit problem and deal with the Exploration-Exploitation dilemma.

1 Introduction

Recently, neural networks have been used in *Reinforcement Learning* (RL) for approximating agent policies and/or value functions, leading to impressive results. Indeed, neural networks allow the agent to model complex relations among the data as they are universal function approximators, but they come with a lot of challenging concerns.

In particular, neural networks struggle to model the uncertainty of the predictions, a key feature to perform effective exploration. This is the reason why most of the algorithms in RL that use neural networks explore the space using a naive ϵ-greedy strategy, with some notable exceptions (Bootstrapped DQN (1)). Being able to model the uncertainty allows the agent to coordinate his exploration over time, a strategy which has been in the heart of most usual techniques such as the *Upper Confidence Bound* algorithm (2).

Thompson sampling is a classical technique (3) to explore according to the posterior distribution of the parameters. As the actual posterior is intractable, one will be concerned with efficiently approximating the posterior, a problem which is ubiquitous in Machine Learning.

To simplify the setup, we will restrict ourselves to the contextual bandit problem where the agent is given a context at each step. Usually this context is rather low dimensional, but effective methods to deal with high dimension would open the way to a whole new range of problems.

We will explore several neural based techniques and test them on a few benchmark problems as intended in (4).

2 State of the Art

The contextual bandit problem works as follows : at each time step $t = 1, ...n$, the agent is given a context $X_t \in \mathbb{R}^d$ which contains some information about the decision the agent should make. Then the agent chooses an action a_t among k available actions, and receives a reward $r_t = r_t(a_t, X_t)$. The goal of the algorithm is to maximize the cumulative reward $r = \sum_{i=1}^{n} r_t$. We will also use cumulative regret $\mathcal{R} = \mathbb{E}[r^\star - r]$ where $r^\star$ is the cumulative reward of the optimal policy.

2.1 Neural ϵ-greedy

The first and intuitive baseline we will be using is the neural ϵ-greedy strategy.

It is a simple method which consists in predicting the reward of the different actions given a context with a neural network, and act ϵ-greedily according to that prediction. The network corresponding to the chosen action is trained with regular backpropagation.

2.2 Thompson sampling

As mentioned above, having a precise strategy to direct exploration is the key to many improvements, and the main strategy we use is *Thompson* sampling.

This strategy relies on modelling the distribution of the parameters of the model given the data observed up to the current time. This distribution models the uncertainty of the weights, hence, the procedure consists in sampling from it and act greedily.

We rely on the intuitive belief that this will lead to a near optimal cumulative reward. This is only one of many possible approaches. For example, other strategies have been explored, such as *Information directed sampling* (5).

Therefore, if we assume that sampling from posterior leads to accurate exploration, one should study the effect of posterior approximations on the performance of the algorithm. Indeed, computing the exact posterior will be intractable most of the time, and making approximations will be inevitable.

Let us present the algorithm. We consider the prior distribution π_0 over the space of model parameters θ which completely define a model instance.

Algorithm 1 Thompson sampling

Input : Prior distribution over models, $\pi_0 : \theta \in \Theta \to [0, 1]$.
for time $t = 0, ...N$ **do**
  Observe context $X_t \in \mathbb{R}^d$.
  Sample model $\theta_t \sim \pi_t$.
  Compute a_t = BestAction(X_t, θ_t).
  Select action a_t and observe reward r_t.
  Update posterior distribution π_{t+1} with (X_t, a_t, r_t).
end for

Note here that in Algorithm 1 there are two sources of error; the exploration strategy, namely, *Thompson Sampling*, and the approximation to the true posterior.

The methods described below all involve Thompson sampling to explore efficiently given some posterior distribution.

2.3 Linear Methods

Linear Methods are among the simplest methods available. If we assume that the reward is generated according to $Y = X^T\beta + \epsilon$ where $\epsilon \sim \mathcal{N}(0, \sigma^2\Sigma_t)$, the idea is simply to perform a Bayesian linear regression over the data and act greedily at each step according to a *Thompson* sampled model.

We provide the update formula which admits a computationally efficient online version. The posterior of parameters at time t is given by $\pi_t(\beta, \sigma^2) = \pi_t(\beta \mid \sigma^2)\pi_t(\sigma^2)$ where we assume that $\sigma^2 \sim IG(a_t, b_t)$ and $\beta \mid \sigma^2 \sim \mathcal{N}(\mu_t, \sigma^2\Sigma_t)$ an Inverse Gamma and Normal distribution respectively. After observing X, Y, their parameters are given by :

$$\Sigma_t = (X^TX + \Lambda_0)^{-1} \qquad \mu_t = \Sigma_t(\Lambda_0\mu_0 + X^TY)$$
$$a_t = a_0 + t/2 \qquad b_t = b_0 + \frac{1}{2}(Y^TY + \mu_0^T\Sigma_0\mu_0 - \mu_t^T\Sigma_t^{-1}\mu_t)$$

Prior hyperparameters are set to $\mu_0 = 0$, $\Lambda_0 = \lambda Id$ and $a_0 = b_0 = \alpha > 1$.

In practice, computing correlations between all parameters can be computationally intractable, and diagonal approximations for Σ_t are very common.

2.4 Neural Linear

If the dimension of the context becomes very high, Linear Methods can become computationally intractable. On top of that, they are unable to give account of complex relationships among parameters. To tackle this issue, one can learn a low dimensional representation of the context, which contains the relevant information and has linearized the relations among parameters. This is precisely what neural networks are good at, thus the idea of neural linear methods emerges naturally.

In Neural Linear Methods, which are introduced in (4), the bayesian linear regression is performed on a low dimensional representation of the context $\phi(X)$ where ϕ is typically a feed forward neural network. The predicted value of each action a_i is given by $v_i = \phi(X)^T \beta_i$.

One of the limitations of this approach is that it does not estimate the uncertainty all along the pipeline. Even if the uncertainty is estimated in the bayesian linear regression process, the forward pass does not provide any confidence estimate.

The bayesian linear regression and the network can be updated at different time scales, for example one can perform a backward pass in the network every twenty regression updates.

2.5 Bootstrap

Another way to simulate Thompson sampling is to train in parallel q different models and train them on slightly different datasets (6). At each step, one of the models is selected at random and the agent acts greedily according to this model.

The drawback of this approach is that it can be very computationally expensive to train q models in parallel.

More precisely, each model is trained on a dataset $\mathcal{D}_i$. Each new sample of the data is appended to each $\mathcal{D}_i$ independently with probability $p \in [0, 1]$. Note that even when $p = 1$, we have some variability as each network is initialized at random.

2.6 Bayesian by backpropagation

One of the key ideas to avoid training many models in parallel is to directly encode the uncertainty in the weights $\mathbf{w}$ of a neural network. The idea here is to compute a variational approximation to the Bayesian posterior distribution of the weights. Variational methods find the parameters θ of a distribution on the weights $q(\mathbf{w}|\theta)$ that minimizes the Kullback-Leibler (KL) divergence with the true Bayesian posterior on the weights

$$\theta^* = \arg\min_\theta \mathrm{KL}(q(\mathbf{w}|\theta)\|p(\mathbf{w}|D)) = \arg\min_\theta \mathrm{KL}(q(\mathbf{w}|\theta)\|p(\mathbf{w})) - \mathbb{E}_{q(\mathbf{w}|\theta)} \log p(D|\mathbf{w}) \quad (1)$$

The first term in the loss will act as a regularization term by forcing the posterior of the weights not to be far away from the prior. The second term, which is the usual log-likelihood term, will try to fit the complexity of the data into the model. If the weights distribution is parametrized as $\mathbf{w} = t(\theta, \epsilon)$ where $\epsilon \sim q(\epsilon)$ and $t(\theta, \epsilon)$ is a deterministic function, then in (7) they show that the gradients of (1) can be estimated by sampling from $q(\epsilon)$.

The posterior distribution is approximated using a Mean-Field approximation, i.e, that $q(\mathbf{w}|\theta)$ factorizes on the weights through an independence assumption. As we will see, this assumption is very strong.

2.7 Hamiltonian Monte Carlo

In order to estimate efficiently Expectations to compute posterior distributions in large networks, it is mandatory to use efficient sampling methods. Here we present a very good alternative to the regular MCMC. In the case of *Bayesian by backpropagation*, we made the Mean-Field approximation, so we do not need to sample from a complicated distribution. Nevertheless if one decides to relax the independence assumption, efficient sampling will be a tool of primary importance.

Hamiltonian Monte Carlo (HMC) (8) is a sampling method with a high acceptance rate enabling more efficient exploration. It can be used instead of the regular *Monte Carlo Markov Chain* (MCMC) method. The key idea is to simulate a Hamiltonian dynamical system : a particle is given some velocity and moves in the potential field associated to the energy of the model.

Formally, HMC samples from a distribution $\pi(w, v) \propto p(w \mid X)e^{-\frac{1}{2}v^T M^{-1}v}$ where the second term corresponds to the *kinetic* energy.

One of the limitations of the HMC technique is that it needs to compute the gradient of π (which gives the direction where the particle goes) at each step. One can use a noisy gradient estimate based on a subset of the data.

3 Experiments

We performed several experiments to test the performances of the different models presented above. Find the reproducible code in `https://github.com/alexnowakvila/deep_MAB`.

3.1 Models

3.1.1 ε-greedy

We used a single fully-connected layer as network with output dimension 100. We took $\varepsilon = 0.1$.

3.1.2 Linear

For the linear method, we initialize hyperparameters as follows : $\mu_0 = 0, \Lambda_0 = Id, a_0 = b_0 = 1.5$. We consider the exact linear posterior, i.e, we do not assume a diagonal covariance matrix.

3.1.3 Neural Linear

We used the same hyperparameters as mentioned above for the Bayesian linear regression, and always performed the regression in dimension 100, regardless of the dimension of the context. We used three different networks :

- Single fully-connected layer of output dimension 100
- Two fully-connected layers with 100 hidden units and output dimension 100. We used ReLU non-linearity.
- One 5×5 convolution layer (5 filters, stride = 3) followed by ReLU non-linearity followed by a fully-connected layer with output dimension 100.

The last network was only used on the MNIST dataset as convolutions are relevant for images, in which case information can be extracted from local structures.

3.1.4 Bayesian by backpropagation

We use a centered gaussian prior $q(\varepsilon) \sim \mathcal{N}(0, \sigma^2)$ and a gaussian posterior parametrized by its mean and variance in the following way $\mathbf{w} = \mu + \log(1 + \exp(\rho)) \circ \varepsilon$. Observe here that the number of parameters to learn is only the double of the number of weights of the neural network. We set a prior variance of $\sigma^2 = \exp(-3)$. The specific form of the neural network used will be specified in the corresponding experiment. The input of the network is the context and an action encoded as a one hot vector of dimension k (number of actions) $\phi_{\mathbf{w}}(X, a)$.

3.2 Datasets

We will test the models in three different datasets which are described here below.

3.2.1 Mushroom

The Mushroom Dataset (9) contains 22 attributes per mushroom, and two classes: poisonous and safe. As in (7), we create a bandit problem where the agent must decide whether to eat or not a given mushroom. Eating a safe mushroom provides reward +5. Eating a poisonous mushroom delivers reward +5 with probability $1/2$ and reward -35 otherwise. If the agent does not eat a mushroom, then the reward is 0. We set $n = 50000$. See Figure 1 for the results obtained.

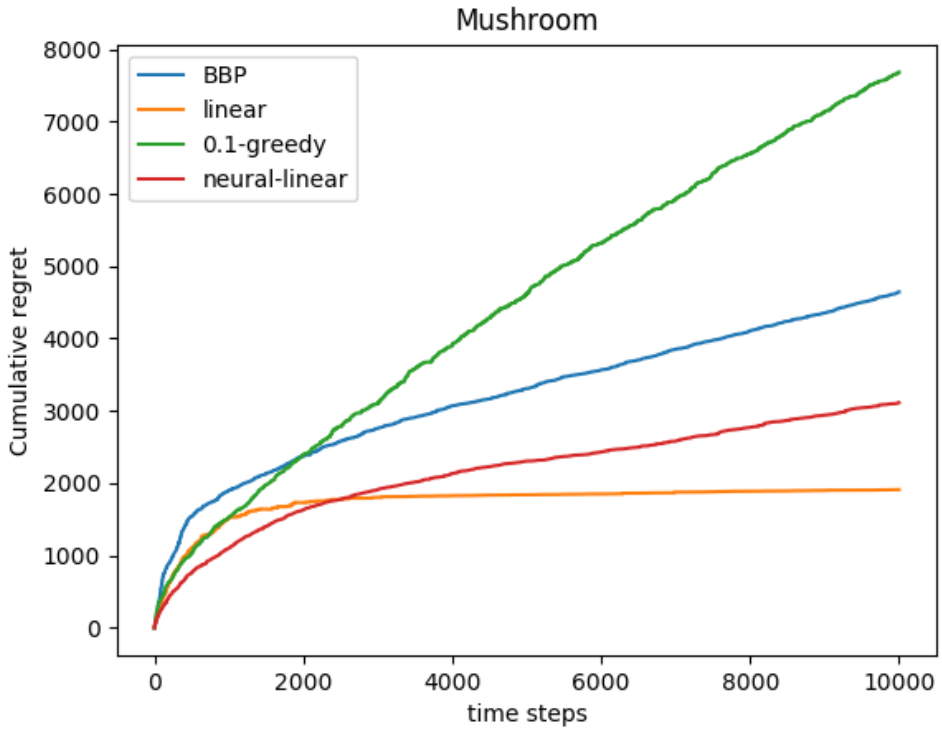

Figure 1: Regret over 10k steps averaged over 4 experiments for the different strategies on the *Mushroom*-dataset. As expected, the ε-greedy strategy performs worst. For the other strategies, we recognize a first phase of intensive exploration followed by an exploitation phase during which the regret increases slower. *Bayesian by backpropagation* and Neural linear methods seem to have a similar performance, although Neural linear is slightly better. The best strategy on this dataset happens to be the linear one as the relations among inputs are quite simple for this dataset. However, we have observed than some of the neural linear experiments perform as good as the linear model.

3.2.2 Statlog

The Shuttle Statlog Dataset (10) provides the value of d = 9 indicators during a space shuttle flight, and the goal is to predict the state of the radiator subsystem of the shuttle. There are k = 7 possible states, and if the agent selects the right state, then reward 1 is generated. Otherwise, the agent obtains no reward (r = 0). The most interesting aspect of the dataset is that one action is the optimal one in 80% of the cases, and some algorithms may commit to this action instead of further exploring. In this case, n = 43500. See Figure 2 for the results obtained.

3.2.3 MNIST

The MNIST dataset (11) consists of black and white 28×28 images of handwritten digits from 0 to 9. We cast it as a bandit problem by considering $k = 10$ arms, giving a reward of 1 if the right arm is pulled and 0 otherwise. Find a comparison between two different neural network architectures on the MNIST dataset under the neural-linear mode in Figure 4. The Bayesian by Backprop model and ε-greedy model did not converge during the 10k iterations, and the Linear Model was extremely slow due to the high dimensionality of the input data ($28 \times 28 = 784$).

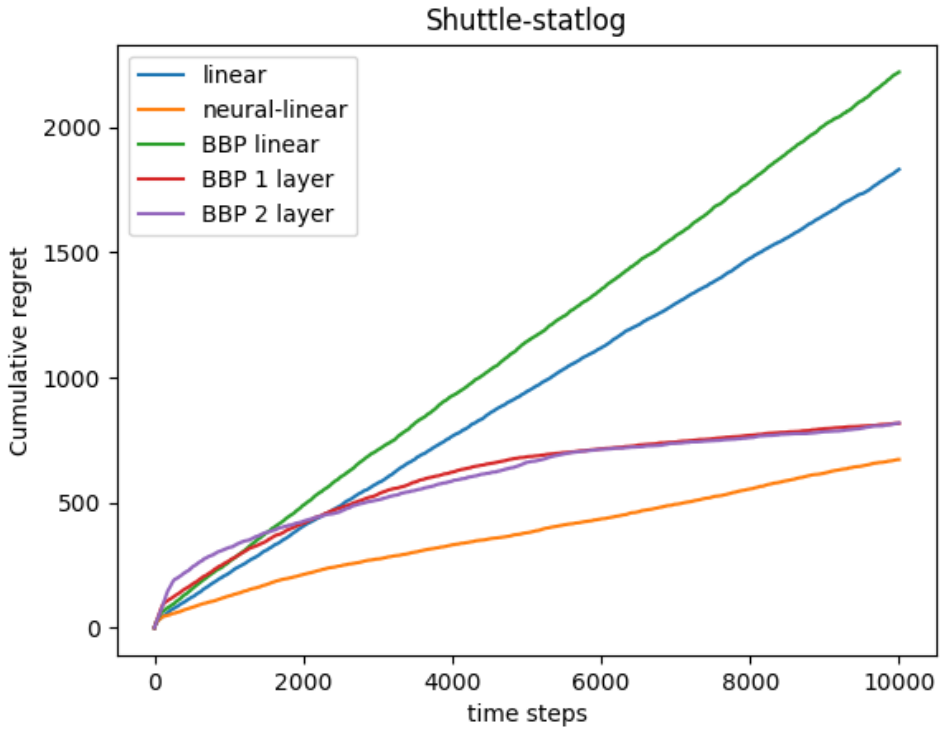

Figure 2: Regret over 10k steps averaged over 4 experiments for the different strategies on the Shuttle-statlog dataset. The dimensionality of the data in the Shuttle-statlog dataset is very small compared to the others ($d = 9$), however the experiments reveal that there are complicated relations between the inputs. Here, the linear strategy performs poorly. We tested *Bayesian by backpropagation* with different number of hidden layers. The linear BBP (where the network is simply a fully-connected layer) is in fact equivalent to the linear method, but it does an approximation instead of computing the exact posterior. Indeed the results are coherent with this fact. *Bayesian by backpropagation* with 1 and 2 hidden layers both perform very well and are asymptotically better than the neural linear strategy

4 Conclusion

We tested several *Thompson*-sampling based methods on three datasets, yielding very promising results. As expected, those methods always perform better than the naive ϵ-greedy strategy.

The easiest task was the *Mushroom*-dataset on which the basic linear method performs best; see Figure 1. Nevertheless, when there happens to be more complex relations between inputs, the representation power of the basic linear method reaches its limits. Therefore, on the *Shuttle-Statlog* dataset, the basic linear method is outperformed by the neural ones. It seems that the best method is *Bayesian by backpropagation* as it performs best asymptotically; see Figure 2.

Even if the Bayesian neural methods manage to model uncertainty, they still suffer some limitations. In the case of *Bayesian by backpropagation* we see that the independence assumption between the network weights is very strong. Indeed, in a multilayer network, the weights are strongly inter-dependent, thus, sampling independently from each weight as the Mean-Field approximation does can lead to very poor performance. In particular, for deep networks, the error of the distribution associated to each weight geometrically increases with the depth of the architecture. Future research would be to investigate this issue by breaking the weight independence while maintaining the tractability of the approximated posterior.

References

[1] I. Osband, C. Blundell, A. Pritzel, and B. Van Roy, "Deep exploration via bootstrapped dqn," in *Advances in Neural Information Processing Systems*, pp. 4026–4034, 2016.

[2] T. L. Lai and H. Robbins, "Asymptotically efficient adaptive allocation rules," *Advances in Applied Mathematics*, 1985.

[3] W. R. Thompson, "On the likelihood that one unknown probability exceeds another in view of the evidence of two samples," *Biometrika*, 1933.

[4] Anonymous, "Deep bayesian bandits showdown: An empirical comparison of bayesian deep networks for thompson sampling," *International Conference on Learning Representations*, 2018.

[5] D. Russo and B. V. Roy, "Learning to optimize via information-directed sampling," *Advances in Neural Information Processing Systems*, 2014.

[6] X. Lu and B. Van Roy, "Ensemble sampling," *arXiv preprint arXiv:1705.07347*, 2017.

[7] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural networks," *arXiv preprint arXiv:1505.05424*, 2015.

[8] T. Chen, E. B. Fox, and C. Guestrin, "Stochastic gradient hamiltonian monte carlo," 2014.

[9] J. Schlimmer, "Mushroom records drawn from the audubon society field guide to north american mushrooms," *GH Lincoff (Pres), New York*, 1981.

[10] A. Asuncion and D. Newman, "Uci machine learning repository," 2007.

[11] Y. LeCun, "The mnist database of handwritten digits," *http://yann. lecun. com/exdb/mnist/*, 1998.

A Covariance of the β distribution

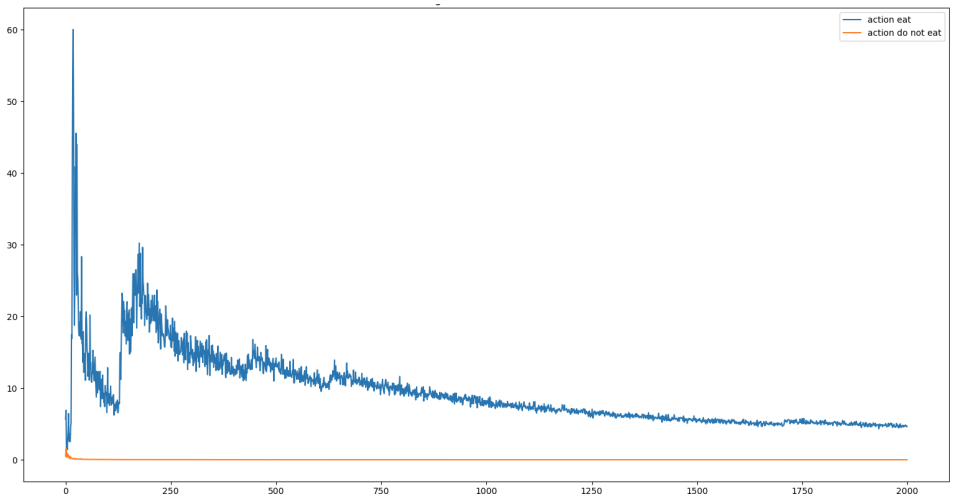

Figure 3: Greatest Eigenvalue of the covariance matrix of the beta distribution along training. Neural Linear method with single layer network on the *Mushroom* dataset

In the neural linear method, the β parameter we want to estimate follows a normal law $\mathcal{N}(\mu_t, \sigma^2\Sigma_t)$. Visualizing its variance is a good way to monitor the exploration strategy of the agent. In Figure 3, we plot the greatest eigenvalue of the covariance matrix of the law of β.

For the *eat*-action, this value explodes at the beginning, meaning that the agent explores a lot, and decreases with time, meaning that the agent exploits more and more its accumulated knowledge.

For the *do not eat*-action, the covariance matrix becomes very small right away, and the agent does not explore at all. This is due to the fact that the received reward is always 0 when taking action *do not eat*.

B Effect of convolutions on the MNIST dataset

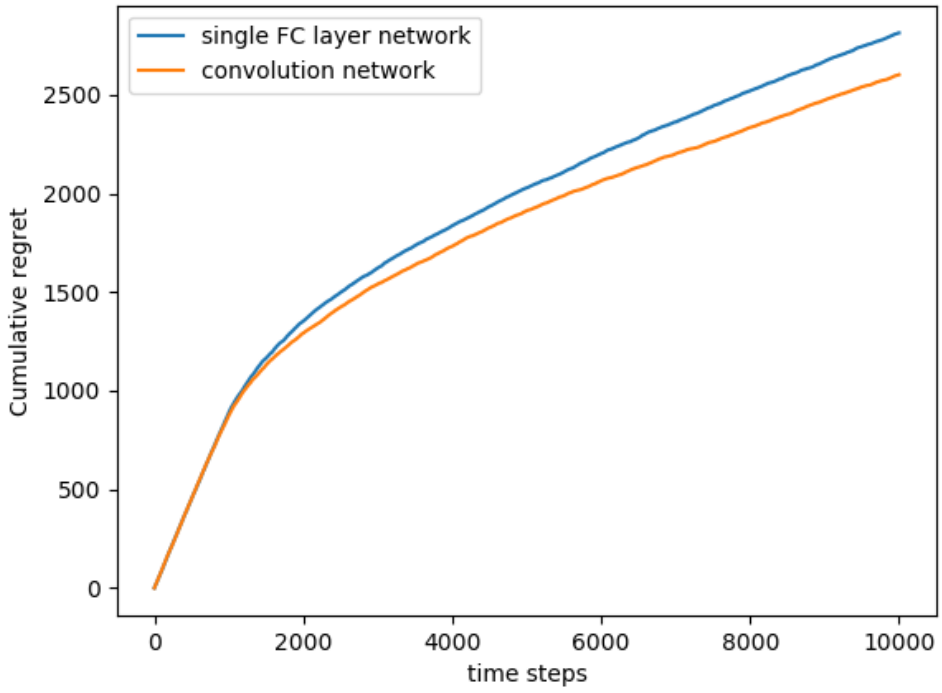

Figure 4: Cumulative regret on the MNIST dataset averaged over 3 experiments for the neural linear method with two different networks