Study of the article
Neural Thompson Sampling

Etienne Levecque & Rony Abecidan

April 8, 2022

Introduction

We propose here to study the article **"Neural Thompson Sampling"** written by Weitong Zhang, Dongruo Zhou, Lihong Li and Quanquan Gu, published in October 2020 [1]. In this paper the authors proposed a new Thompson sampling strategy based on deep learning methods for solving the contextual bandit problem where the arms are changing at every round and the mean reward is seen as any bounded function h. All our experiments are available in the repository : `https://github.com/RonyAbecidan/NeuralTS`.

1 Neural Tangent Kernel

1.1 Motivation and intuition

Before going into the core of the paper, we thought that it could be interesting to talk about the Neural Tangent Kernel theory which is omnipresent in the paper but not well introduced. So we will try to give some taste about it in this part.

The main idea of this theory is that the output of an artificial neural network (also called **ANN**) can be approximated with the help of a feature map[1] when it is big in terms of parameters and initialized with a particular distribution : The LeCun initialization. This approximation can be seen as a linear approximation of the ANN with respect to its parameters. To give some intuition about it, let's think about an ANN which optimizes a loss by gradient descent. Supposing that it is large enough[2], if we slightly change every parameters, this could results in a big change of the output. Hence the gradient descent is fostered to update the parameters very slowly in order to avoid this situation. The more parameters there is and the slower the update will be. In the end, these very small updates can be seen as linear updates w.r.t. every parameters along the current gradient.

This idea of linearity with respect to the vector of parameters is in fact translated with the Taylor's theorem on the output of the ANN. Using the same notations as the ones in the paper, we note $f(\mathbf{x}, \theta)$ the output of the ANN with $\mathbf{x}$ belonging to the input space I and θ the vector of learnable parameters. Let θ_t be the vector of parameters at step t of the gradient descent and let θ_* be the optimal solution for the gradient descent problem. Then the Taylor's Theorem precisely tell us that if the width of the Artificial Neural Network is big enough, then:

$$\forall x \in I, \forall t \geq 0, \quad f(x, \theta_*) \approx f(x, \theta_t) + \nabla_\theta f(x, \theta_t)^T(\theta_* - \theta_t)$$

To prove now that this formulation is linear we need to show that $\nabla_\theta f(x, \theta_t)$ is a constant with respect to the parameter θ.

1.2 Accuracy of the approximation

We could ask us the following question: How much does this approximation makes sense ? Let's do a simple proof in the case of a single hidden layer as illustrated in Figure 1. This proof derives from [2].

Here, there is one hidden layer and hence, we only have two vectors of parameters which are W_1 and W_L. We will also denote $W^{(i)}$ the parameters concerning neurons $i \in [1, m]$ and remark that in comparison with the paper, here $f(x, \theta) = f_L$. We will see later why the authors have decided to multiply the output by $\sqrt{m}$ for their strategy.

For this proof, we actually need that ReLU is twice differentiable but it is not the case. Nevertheless, we could approximate the ReLU activation with a smoother one like SoftPlus and the result will be the same. We will note for instance σ this smooth approximation of the ReLU in what follows.

[1] that gives birth to a kernel, the famous neural tangent kernel

[2] we will see later what this means in terms of dimensions

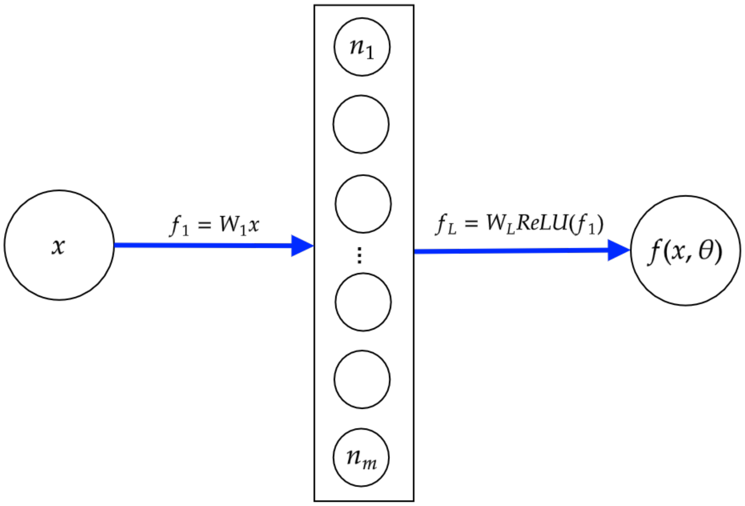

Figure 1: Simple MLP architecture considered here

$$\forall x \in I, \quad f(x,\theta) = \sum_{i=1}^{m} W_L^{(i)} \sigma(W_1^{(i)} x)$$

If we take a look at the gradient of f,

$\forall x \in I$:

$$\frac{\partial f}{\partial W_1^{(i)}}(x,\theta) = W_L^{(i)} \sigma'(W_1^{(i)} x) x \tag{1}$$

$$\frac{\partial f}{\partial W_L^{(i)}}(x,\theta) = \sigma(W_1^{(i)} x) \tag{2}$$

With (1) and (2) we can deduce:

$$\|\nabla_\theta f\| = \sqrt{\left(\frac{\partial f}{\partial W_1^{(1)}}\right)^2 + ... + \left(\frac{\partial f}{\partial W_1^{(m)}}\right)^2 + \left(\frac{\partial f}{\partial W_L^{(1)}}\right)^2 + ... + \left(\frac{\partial f}{\partial W_L^{(m)}}\right)^2} \tag{3}$$

Now if we fix $x \in I$, then x is totally constant. Concerning $\sigma(W_1^{(i)} x)$ and $\sigma'(W_1^{(i)} x)$, since we assume that all the parameters of the ANN are initialized like in the paper, it means they are drawn from the LeCun initialization: a normal i.i.d of zero mean and variance proportional to $\frac{1}{m}$. So they are random variables with constant variance as the width increase. And so we only have the variance of $W_L^{(i)}$ which varies with the width. To be more precise, they are varying in $\mathcal{O}(\frac{1}{m})$.

By squaring all the terms inside the square root, the variances become means. On one hand, we have the terms from the partial derivative with respect to $W_1^{(i)}$, m terms that are in $\mathcal{O}(\frac{1}{m})$. So those goes toward 0 as m tend to infinity. In the other hand, we have the terms from the partial derivative with respect to $W_L^{(i)}$ with mean $\mathcal{O}(1)$. With those observations, we have from (3):

$$\|\nabla_\theta f(x,\theta)\| \approx \sqrt{m}\sqrt{\frac{1}{m}\left(\sigma^2(W_1^{(1)} x) + ... + \sigma^2(W_1^{(m)} x)\right)} \tag{4}$$

Knowing that all parameters are drawn i.i.d., we have that every $\sigma^2(W_1^{(i)} x)$ are i.i.d. So the term under the square root is a empirical mean which approach the true mean as $m \to \infty$ because of the law of large numbers. This true average is a constant so finally we have proven that:

$$\|\nabla_\theta f(x,\theta)\| \sim \mathcal{O}(\sqrt{m}) \tag{5}$$

Using the same type of reasoning, we could prove that, as the width increase:

$$\|\nabla_\theta^2 f\| \sim \mathcal{O}(1) \tag{6}$$

Why are we interesting in those values? Because what we want to show is, that the parameter θ have to almost no change so that f is giving the good output and that this change in θ has no impact on the gradient of f. Since we are using a gradient descent, if the learning rate is rather small, then we have :

$$\text{Net change in } f(x,\theta) \leq \|f(x,\theta_0) - h(x)\| \tag{7}$$

where $h(x)$ is the true output value.

We have on one hand:

$$\text{The distance } d \text{ moved in } \theta \text{ space } \approx \frac{\text{Net change in } f(x,\theta)}{\text{Rate of change of } \nabla_\theta f(x,\theta_0)} = \frac{\|f(x,\theta_0)-h(x)\|}{\|\nabla_\theta f(x,\theta_0)\|}$$

And in the other hand,

$$\text{Change in the model Jacobian} \approx \text{distance } d \times \text{rate of change of the Jacobian} = d\|\nabla^2_\theta f(x,\theta_0)\|$$

Now we can infer that,

$$\kappa(\theta_0) = \text{relative change in model Jacobian} \approx \frac{\text{Change in model Jacobian}}{\text{Norm of the Jacobian}} = \frac{d\|\nabla^2_\theta f(x,\theta_0)\|}{\|\nabla_\theta f(x,\theta_0)\|} = \|f(x,\theta_0)-h(x)\|\frac{\|\nabla^2_\theta f(x,\theta_0)\|}{\|\nabla_\theta f(x,\theta_0)\|^2}$$

Finally with the symmetric initialization used in the paper, we have $f(x,\theta_0)=0$ so that, as the width increase, we always have (7)$\sim \mathcal{O}(1)$. And using (5) & (6), we have that $\kappa(\theta_0) \sim \mathcal{O}(\frac{1}{m})$. To interpret this result we can say that the amount of change required to give a correct output for $f(x,\theta)$ have a very small impact on the gradient of f with respect to θ. In other words, during training, the gradient of f is almost constant with respect to θ. Moreover, the wider the network, the better the approximation.

In particular, thanks to the NTK theory, the approximation using the Taylor's Theorem should be linear with respect to θ.

1.3 Going further

The paper addresses the case of multi-layers ANN which is a bit more challenging but the idea stays the same. They need this tool to track the approximation error during the regret analysis. If you need more details about the NTK theory, we advise you to read the interesting blog post `https://rajatvd.github.io/NTK/`. Moreover, the paper scale the output of the ANN by $\sqrt{m}$. This scale has certainly multiple reasons such as a notation reason. But one of the impact of this scale is that the time where we can approximate the ANN by its Kernel map is reached faster. We can see that throw the expression of κ. If you add a factor before f in both terms, we will obtain the following:

$$\kappa(\theta_0) = \|f(x,\theta_0)-h(x)\|\frac{\|\nabla^2_\theta \sqrt{m} f(x,\theta_0)\|}{\|\nabla_\theta \sqrt{m} f(x,\theta_0)\|^2} = \frac{1}{\sqrt{m}}\|f(x,\theta_0)-h(x)\|\frac{\|\nabla^2_\theta f(x,\theta_0)\|}{\|\nabla_\theta f(x,\theta_0)\|^2} \sim \mathcal{O}\left(\frac{1}{m\sqrt{m}}\right)$$

2 A solution for non-linear rewards : Neural Thompson Sampling

As we already know, Thompson Sampling could reveal to be a very efficient algorithm for solving the contextual multi-armed bandit in the linear case. Now, it is interesting to see to what extent it's possible to extend this algorithm in the non-linear case.

2.1 Scenario and framework

In the paper studied, the authors worked on the famous contextual bandit problem. In this problem, we consider that we have an agent which has to make sequential choices based on some **contexts** (also called **arms**) from its environment at each round. Each choice leads to a certain **reward** linked to the chosen context vector and, the final goal of the agent is to find this link so that it can maximize its total reward. To be more specific, the authors decided to broach a rather general version of this problem considering that the context vectors at each round are changing and the link between the contexts and rewards is made thanks to some bounded mapping. However, the number of contexts per round stays constant as we usually see in the litterature.

Formally, let's call K the number of possible arms per round and T the number of rounds (also called the **horizon time**). In that case, we imagine a situation where the reward $r_{t,k}$ obtained at time t, pulling the k-th arm verify :

$$\boxed{\forall_{1\leq k\leq K} \;\; \forall_{1\leq t\leq T} \;\; r_{t,k} = h\left(\mathbf{x}_{t,k}\right) + \xi_{t,k}, \quad \text{with} \quad \left|h\left(\mathbf{x}_{t,k}\right)\right| \leq 1}$$

where **h** could be any bounded function and $\xi_{t,k}$ is a noise meeting the condition :

$$\boxed{\forall\lambda \in \mathbb{R} \;\; \mathbb{E}\left[\exp\left(\lambda\xi_{t,k}\right) \mid \xi_{1:t-1,k}, \mathbf{x}_{1:t,k}\right] \leq \exp\left(\lambda^2 R^2\right)}$$

This condition is met whenever $|\xi_{t,k}| \leq R$ almost surely[3]

Our scenario is resumed in Figure 2 :

[3] see Remark 1 in Appendix A.1 of Filippi et al. (2010) [3]

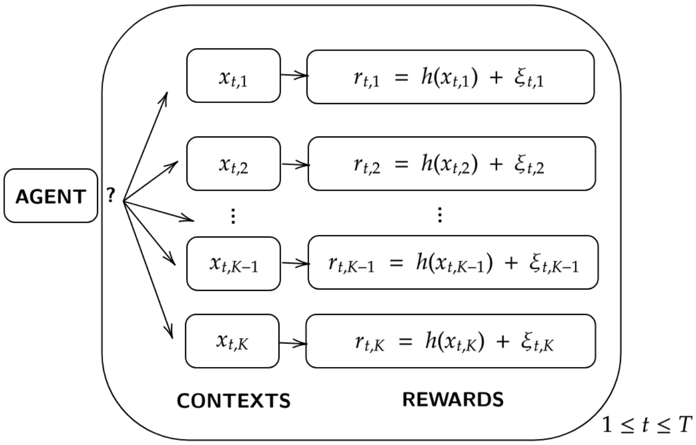
AGENT
?
$x_{t,1}$
$r_{t,1} = h(x_{t,1}) + \xi_{t,1}$
$x_{t,2}$
$r_{t,2} = h(x_{t,2}) + \xi_{t,2}$
$\vdots$
$\vdots$
$x_{t,K-1}$
$r_{t,K-1} = h(x_{t,K-1}) + \xi_{t,K-1}$
$x_{t,K}$
$r_{t,K} = h(x_{t,K}) + \xi_{t,K}$
CONTEXTS
REWARDS
$1 \leq t \leq T$

Figure 2: Illustration of the scenario proposed in the paper

As we said before, the final aim of the agent is to maximize its total reward. When we design a strategy for the agent, an other way to see this goal could be to minimize what we called the **pseudo-regret** which is simply the difference between the mean reward of the best agent always selecting the optimal arm at each round and the mean reward of the agent following our strategy. Formally, it is defined by the relation :

$$\boxed{\mathsf{R}_T = \mathbb{E}\left[\sum_{t=1}^{T}\left(r_{t,a_t^*} - r_{t,a_t}\right)\right]}$$

where a_t^* represents the best arm to pull at round t.

This quantity is really interesting for comparing the performance of different strategies.

2.2 From LinearTS to NeuralTS

2.2.1 LinearTS : Framework and results

We are already familiar with the case where h is linear w.r.t. $x_{t,k}$. In that particular situation we have :

$$\forall_{1\leq k\leq K} \quad \forall_{1\leq t\leq T} \quad r_{t,k} = \theta_*^T x_{t,k} + \xi_{t,k}, \quad \text{with } |\theta_*^T x_{t,k}| \leq 1 \text{ where } \theta_* \text{ is a parameter to find.}$$

One way to solve this problem could be the Linear Thompson Sampling algorithm which ensures a regret bound in $O(\sqrt{T}\log T)$ for σ^2-sub-Gaussian noise. We recall the frame of this algorithm below :

- First, we propose a prior distribution for θ_* : $\theta_* \sim \mathcal{N}(0, \kappa^2 I_d)$ and we consider a gaussian noise $\xi_{t,k} \sim \mathcal{N}(0, \sigma^2)$.
- Then, if we are at time t=0, we sample θ^* from its prior, else at round t+1, we deduce the posterior $p(\theta_* \mid (x_{\tau,k_\tau}, r_{\tau,k_\tau})_{\tau\leq t}) \sim \mathcal{N}\left(\hat{\theta}_t^\lambda, \sigma^2\left(B_t^\lambda\right)^{-1}\right)$
  with :

  $B_t^\lambda = \lambda \mathbf{I}_d + \sum_{\tau=1}^{t} x_{\tau,k_\tau} x_{\tau,k_\tau}^\top$ is the **design matrix**, also called the **covariance matrix**
  $\hat{\theta}_t^\lambda = (B_t^\lambda)^{-1}\left(\sum_{\tau=1}^{t} r_{\tau,k_\tau} x_{\tau,k_\tau}\right)$ is the θ which minimizes the loss function

  $$\mathcal{L}(\theta) = \frac{1}{2}\sum_{\tau=1}^{t}(r_{\tau,k_\tau} - \theta^T x_{\tau,k_\tau})^2 + \frac{1}{2}\lambda||\theta||^2 \text{ with } \lambda = \frac{\sigma^2}{\kappa^2}.$$

  Knowing the posterior of θ_*, we can sample one point $\tilde{\theta}_t$ from this posterior.
- Then, we can choose naturally to pull the arm k which maximizes the estimate of the mean reward $\tilde{\theta}_t^{\,T} x_{t,k}$

The most important point of the algorithm rely on the ability to deduce the posterior distribution of θ^*. Since we are going to use this result later so that we can justify a formula used in the paper, we propose now to prove its expression.

Proof : Let $t \in \mathbb{R}$

$$\begin{aligned} p(\theta_* \mid (x_{\tau,k_\tau}, r_{\tau,k_\tau})_{\tau \leq t}) \;&\alpha\; p(\theta_*)p((x_{\tau,k_\tau}, r_{\tau,k_\tau})_{\tau \leq t}|\theta_*) \\ &\alpha\; \prod_{\tau \leq t} \underbrace{p(r_{\tau,k_\tau} \mid \theta_*, x_{\tau,k_\tau})}_{\mathcal{N}(\theta_*^T x_{\tau,k}, \sigma^2)} \;\; \underbrace{p(\theta_*)}_{\mathcal{N}(0,\kappa^2 I_d)} \qquad (8) \\ &\alpha\; \exp(\sum_{\tau \leq t} \underbrace{\frac{1}{2\sigma^2}(\theta_*^T x_{\tau,k} - r_{\tau,k`_\tau})^2 + \frac{\lambda}{2\sigma^2}(\theta_*^T\theta_*))}_{\frac{1}{\sigma^2}\mathcal{L}(\theta_*)} \qquad (9) \end{aligned}$$

Now, by (1) we know that the posterior of θ_* given the history at time t+1 is a product of gaussians involving θ_* and thus, it is also a gaussian. Then, we know that the mean of this gaussian is the θ which maximizes its density function, i.e. the θ that maximizes $\mathcal{L}(\theta)$ according to (2) and this is precisely $\hat{\theta}_t^\lambda$ by definition.

Now, we can deduce the variance of this distribution Σ computing the quadratic term in θ_* in (2).

$$\Sigma^{-1} = (\sigma^2)^{-1} \underbrace{\sum_{\tau \leq t}(x_{\tau,k_\tau} x_{\tau,k_\tau}^T + \lambda I_d)}_{\mathbf{B_t^\lambda}} \text{ and so } \Sigma = \sigma^2(\mathbf{B_t^\lambda})^{-1} \;\square$$

We can also compute the posterior distribution of the scalar estimated reward $r_{\tilde{t,k}} = \tilde{\theta_t}^T x_{t,k}$.

Given the shape of the distribution of $\tilde{\theta_t}$ and the linear expression relying it with the reward, we can deduce that the posterior of the reward follows also a normal distribution. Now, we just have to compute its mean and variance.

$$\mathbb{E}(r_{\tilde{t,k}} \mid (x_{\tau,k})_{\tau \leq t}) = (\hat{\theta}_t^\lambda)^T x_{t,k} := f(x_{t,k}, \hat{\theta}_t^\lambda)$$

$$\begin{aligned} \mathbb{V}(r_{\tilde{t,k}} \mid (x_{\tau,k \tau \leq t}) &= \mathbb{V}(\tilde{\theta_t}^T x_{\tau,k})) \\ &= \mathbb{V}(x_{\tau,k}^T \tilde{\theta_t}) \\ &= x_{\tau,k}^T \mathbb{V}(\tilde{\theta_t}) x_{\tau,k} \\ &= x_{\tau,k}^T \sigma^2 (B_t^\lambda)^{-1} x_{\tau,k} \\ &= \kappa^2 \lambda x_{\tau,k}^T (B_t^\lambda)^{-1} x_{\tau,k} \end{aligned}$$

If now the mean reward function is not linear, we could expect from the LinTS to have a linear regret. However, it can perform surprisingly well in some cases as shown in **our illustrative notebook**.

However, this behaviour is not constant and it could be better to have a strategy that can fit with a non linear regret. Let's see how the authors have extended the TS strategy in the case where h can be any bounded function.

2.2.2 NeuralTS

As suggested by its name, the NeuralTS algorithm tries to exploit the ability of a neural network to approximate a non-linear and bounded function. Here, this function corresponds to the mean reward of our problem $h(x_{t,k})$. In order to approximate this reward, the authors designed a multi-layer perceptron with L layers using the ReLU activation between each of them. On top of that, they rescaled the output of the final layer in order to be in a proper background for applying the theory behind the NTK. At the end, the output of the neural network is a function $f(x, \theta)$ where θ contains all the learnable parameters $W_1, ..., W_L$ and, we seek for a θ_* such that $f(x, \theta_*)$ is the best approximation possible for $h(x)$. The structure of the neural network proposed is presented in Figure 3.

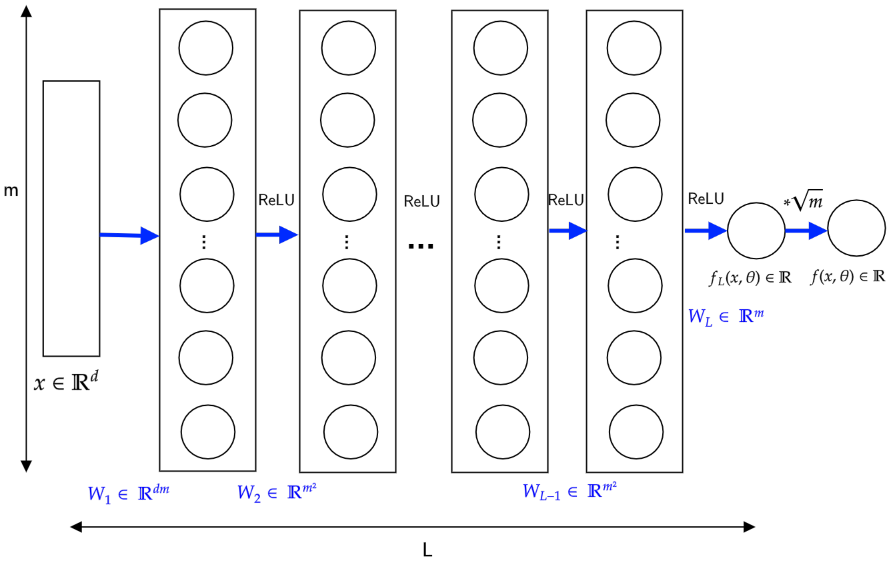
m
$x \in \mathbb{R}^d$
ReLU
ReLU
...
ReLU
ReLU
$*\sqrt{m}$
$f_L(x,\theta) \in \mathbb{R}$
$f(x,\theta) \in \mathbb{R}$
$W_L \in \mathbb{R}^m$
$W_1 \in \mathbb{R}^{dm}$
$W_2 \in \mathbb{R}^{m^2}$
$W_{L-1} \in \mathbb{R}^{m^2}$
L

Figure 3: MLP architecture estimating the expectation of the reward

The authors claimed that θ lives in $\mathbb{R}^p$ with $p = dm + m^2(L-2) + m$. This remark is actually true when there is no bias for each layer and the authors didn't mention it. In the following scheme, we specified the dimensions of the weight vectors considering also that there are no biases in order to be in the same setting as the authors. We are a bit surprised by this choice since the bias is really helpful for a regression task[4]. We have illustrated that point in **our illustrative notebook**.

The remaining question here is : How are we going to adapt the Thompson sampling algorithm in such situation ? They didn't really explain this adaptation and it was in fact difficult for us to understand directly how they derive the formulas they used for updating the estimated mean reward posterior. That's why we propose for the reader some clarifications in what follows.

One of the main idea in this article consists in sampling directly from the posterior distribution of the estimated mean of the scalar reward $r_{t,k}$ rather than sampling the posterior distribution of θ_*. This choice is clever since it enables to save time and hence make the algorithm scalable when we work with a complex neural network with a large width m. Indeed, instead of sampling a large parameter which lives in $\mathbb{R}^p$ with $p = dm + m^2(L-2) + m$, we simply sample a reward in $\mathbb{R}$. Moreover, to estimate the parameter θ, it is proposed to learn it with a gradient descent applied with the cost function :

$$\boxed{L(\boldsymbol{\theta}) = \frac{1}{2}\sum_{i=1}^{t}\left[f\left(\mathbf{x}_{i,a_i};\boldsymbol{\theta}\right) - r_{i,a_i}\right]^2 + \frac{1}{2}m\lambda\left\|\boldsymbol{\theta} - \boldsymbol{\theta}_0\right\|_2^2}$$

This cost function fosters θ to be as optimal as possible while staying not too far from the initialization θ_0. A new question arises now : How are we going to design an estimate for this mean reward ?

In the context of the NTK theory, we can derive from our neural network a feature map :

$$\boxed{\Phi(x,\theta) = \nabla_\theta f_L(x,\theta) = \frac{\nabla_\theta f(x,\theta)}{\sqrt{m}} = \frac{g(x,\theta)}{\sqrt{m}} \text{ with } g = \nabla_\theta f \text{ [5]}}$$

and, when m is enough large, with the first-order Taylor approximation of our output f, we can write for $t \leq T$:

$$\boxed{f(x,\theta_*) \approx f(x,\theta_t) + g(x,\theta_t)^T(\theta_* - \theta_t)}$$

Hence, at time $t+1$, for the k-th arm, we can use as our estimate for the mean reward distribution

$$r_{t\tilde{+}1,k} = f(x_{t,k},\theta_t) + \underbrace{g(x_{t,k},\theta_t)^T}_{\sqrt{m}\Phi(x_{t,k},\theta_t)^T}(\theta_* - \theta_t) = \Theta_*^T\Phi(x_{t,k},\theta_t)$$

with $\Theta_*^T = (\sqrt{m}\theta_*^T, \textbf{bias}^T)$[6]and adding a coefficient equals to 1 to $\Phi(x_{t,k},\theta_t)$

So, at the end, we end up with a linear estimate w.r.t.Θ_*.

[4] However, there is no need to have a bias in the output layer since we can deduce it and remove it

[5] g has been used in the paper without being defined. We have first guessed its expression and then confirmed it with the authors.

Now using the LeCun initialization for our neural network, $\theta_0 \sim \mathcal{N}(0, \frac{1}{m})$, we can make clear analogies with the LinearTS algorithm with $\mathcal{N}(0, \frac{1}{m})$ as the prior distribution of θ_*.

Linear Thompson Sampling	Neural Thompson Sampling
$\theta_* \sim N(0, \kappa^2 I_d)$	$\theta_* \sim N\left(0, \frac{1}{m} I_d\right)$
$B_t^\lambda = \lambda I_d + \sum x_{t,k_t} {x_{t,k_t}}^T$	$B_t^\lambda = \lambda I_d + \sum \Phi(x_{t,k_t}, \theta_t)\Phi\left({x_{t,k_t}}^T, \theta_t\right)$
$\theta_* \vert (x_{\tau,k_\tau},\ r_{\tau,k_\tau})_{1\leq\tau\leq t} \sim N(\theta_t, \lambda\kappa^2 (B_t^\lambda)^{-1})$	$\Theta_* \vert (x_{\tau,k_\tau},\ r_{\tau,k_\tau})_{1\leq\tau\leq t} \sim N(\Theta_t, \lambda (B_t^\lambda)^{-1})$
$\tilde{r}_{t+1,k} = {\theta_*}^T x_{t,k}$	$\tilde{r}_{t+1,k} = {\Theta_*}^T \Phi(x_{t,k}, \theta_t)$
$\tilde{r}_{t+1,k} \sim N\left(f(x_{t,k}, \theta_t),\ \lambda\kappa^2 {x_{t,k}}^T (B_t^\lambda)^{-1} x_{t,k}\right)$	$\tilde{r}_{t+1,k} \sim N\left(f(x_{t,k}, \theta_t), \lambda\Phi\left({x_{t,k}}^T, \theta_t\right)(B_t^\lambda)^{-1}\Phi(x_{t,k}, \theta_t)\right)$

Figure 4: Analogies between LinTS and NeuralTS at initialization and round t+1

We can give more details justifying these analogies.

Concerning the first one, we can simply consider that the authors chose $\kappa^2 = \frac{1}{m}$. However, they didn't explicitly give this information. They just proposed to initialize with $\theta_0 \sim \mathcal{N}(0, \frac{1}{m})$. Behind this choice, we made the link with the initialization of the LinearTS algorithm which lies on the prior of θ^*. Moreover, assuming that $\mathcal{N}(0, \frac{1}{m})$ was the prior of θ_* in the paper enables to retrieve the formulas given by the authors so we have the feeling that it was a good guess.

For the design matrix, there are two surprising things. At first, contrarily to the linear case, in the NeuralTS each components of the covariance sum depends on θ_t. We can then wonder the validity of this expression since θ_t changes the transformation at each round t. Actually this expression is fine and we can see it simply noticing that it is like having at time t an offline dataset made of features $(X_{t,k})_{1\leq t\leq\tau, 1\leq k\leq K} = (\Phi(x_{t,k}))_{1\leq t\leq\tau, 1\leq k\leq K}$. Moreover, if we look carefully the loss function given previously : $\mathcal{L}(\theta) = \frac{1}{2}\sum_{\tau=1}^{t} ||r_{\tau,k_\tau} - \theta^T x_{\tau,k_\tau}||^2 + \frac{1}{2} m\lambda||\theta||^2$ we see that the regularization parameter is λm and not just λ, so how do the authors end up with the expression presented above ?
In fact, they divided by m the real covariance matrix in order to make appears the feature map $\Phi = \frac{g}{\sqrt{m}}$. Due to that choice, they had to multiply by m the expression of the variance of $r_{t\tilde{+}1,k}$ and doing so, they end up again with the feature map Φ in this expression of this variance. Unfortunately, the authors didn't explained that trick and we understood it later doing the computations.

Concerning the posterior of Θ_*, it comes from the same computation than we did before for LinearTS. However, here we can't have explicitly the optimal θ that minimizes $\mathcal{L}(\theta)$ at round t since we are solving a non linear problem. Hence, θ_t which is obtained doing a gradient descent is seen as the best approximation of the mean of θ_* we can have at round t. Hence, we can choose naturally that the posterior of θ^* is $\mathcal{N}(\theta_t, \kappa^2\lambda(B_t^\lambda)^{-1})$ with $\kappa^2 = \frac{1}{m}$. In that case, our estimate for the mean reward is unbiased and the posterior of Θ^* becomes naturally [7] $\mathcal{N}(\Theta_t = \sqrt{m}\theta_t, m * \frac{1}{m}\lambda(B_t^\lambda)^{-1})$. At last, for the posterior distribution of the expected reward estimator, it looks exactly like we have previously done with the LinTS:

[6] bias=$f(x_{t,k}, \theta_t) - g(x_{t,k}, \theta_t)^T\theta_t$

[7] We don't count the bias term which is a deterministic value

$$\mathbb{E}(\tilde{r_{t+1,k}} \mid (x_{\tau,k})_{\tau\leq t}) = f(x_{t,k},\theta_t) + \sqrt{m}\Phi(x_{t,k})\underbrace{(\mathbb{E}(\theta_*)-\theta_t)}_{0} = f(x_{t,k},\theta_t)$$

$$\begin{aligned}
\mathbb{V}(\tilde{r_{t+1,k}} \mid (x_{\tau,k\tau\leq t}) &= \mathbb{V}(\sqrt{m}\Phi(x_{t,k})^T\theta_*) \\
&= m\Phi(x^T_{\tau,k})\mathbb{V}(\tilde{\theta_*})\Phi(x_{\tau,k}) \\
&= m \qquad \underbrace{(\lambda m)}_{\text{regularization term}} \quad \underbrace{\frac{1}{m}}_{\kappa^2} \qquad\qquad \underbrace{m^{-1}}_{\text{multiplication by m for the true covariance matrix}} \qquad\qquad \Phi(x_{\tau,k})^T(B_t^\lambda)^{-1}\Phi(x_{\tau,k}) \\
&= \lambda\Phi(x_{\tau,k})^T(B_t^\lambda)^{-1}\Phi(x_{\tau,k})
\end{aligned}$$

Now, we have all the tools to understand the proposed algorithm :

```
Input: Number of rounds $T$, exploration variance $\nu$, network width $m$, regularization parameter $\lambda$.
Set $\mathbf{B}_0^\lambda = \lambda\mathbf{I}$
Initialize $\theta_0 = (\mathrm{vec}(\mathbf{W}_1);\cdots;\mathrm{vec}(\mathbf{W}_L)) \in \mathbb{R}^p$, where :
for each $1 \leq l \leq L-1, \mathbf{W}_l = (\mathbf{W},\mathbf{0};\mathbf{0},\mathbf{W})$ each entry of $\mathbf{W}$ is generated independently from $\mathcal{N}(0,4/m)$
and $\mathbf{W}_L = (\mathbf{w}^\top, -\mathbf{w}^\top)$ , each entry of $\mathbf{w}$ is generated independently from $\mathcal{N}(0,2/m)$
for $t = 1,\cdots,T$ do
    for $k = 1,\cdots,K$ do
        $\sigma_{t,k}^2 = \lambda\mathbf{g}^\top(\mathbf{x}_{t,k};\boldsymbol{\theta}_{t-1})(\mathbf{B}_{t-1}^\lambda)^{-1}\mathbf{g}(\mathbf{x}_{t,k};\boldsymbol{\theta}_{t-1})/m$
        Sample estimated rewards $\tilde{r}_{t,k} \sim \mathcal{N}\left(f(\mathbf{x}_{t,k};\boldsymbol{\theta}_{t-1}), \nu^2\sigma_{t,k}^2\right)$
    Pull arm $k_t$ and receive reward $r_{t,k_t}$, where $k_t = \mathrm{argmax}_k\, \widetilde{r}_{t,k}$
    Set $\boldsymbol{\theta}_t$ to be the output of gradient descent w.r.t. the loss $\mathcal{L}(\theta)$ and the learning rate $\eta$.
    $\mathbf{B}_t^\lambda = \mathbf{B}_{t-1}^\lambda + \mathbf{g}(\mathbf{x}_{t,a_t};\boldsymbol{\theta}_t)\mathbf{g}(\mathbf{x}_{t,a_t};\boldsymbol{\theta}_t)^\top/m$
```

Algorithm 1: NeuralTS algorithm

As you see, we can easily understand how they derive this algorithm based on our knowledge on the LinearTS algorithm and our previous remarks. We can notice the symmetry of the coefficients and the fact that the non-zero weights are generated via a centered normal law with a variance proportional to $\frac{1}{m}$. Then, we can also notice the introduction of an inflation ν to the variance of the reward posterior. This choice is consistent with the litterature around Thompson Sampling for influencing the degree of exploration.

At last, presented like that, it seems that we don't use the **Sherman-Morrison** formula for computing $(\mathbf{B}_{t-1}^\lambda)^{-1}$ which should enable to have a much more faster computation. In practice we tested the algorithm using the Sherman-Morrison formula and, we noticed that the computation for the design matrix diverged numerically rather quickly. Hence, we couldn't used it in such circumstances. Moreover, due to time constraints, the authors stated that they relaxed this computation approximating the inverse of the design matrix by the inverse of the matrix made of the diagonal elements.

2.3 Regret guarantees

Under the following assumptions :

- $\forall t \in [T], \forall k \in [K], \|\mathbf{x}_{t,k}\|_2 = 1$ and $[\mathbf{x}_{t,k}]_j = [\mathbf{x}_{t,k}]_{j+d/2}$.
- $\nu = B + R\sqrt{\widetilde{d}\log(1+TK/\lambda)+2+2\log(1/\delta)}$ where $\widetilde{d}$ is a metric linked to the underlying dimension of the set of observed contexts, $B = \max\left\{1/(22\mathrm{e}\sqrt{\pi}), \sqrt{2\mathbf{h}^\top\mathbf{H}^{-1}\mathbf{h}}\right\}$, $h = \left(h\left(\mathbf{x}^1\right), \ldots, h\left(\mathbf{x}^{TK}\right)\right)^\top$,$\mathbf{H}$ is the neural tangent kernel matrix on the context set, and R is the sub-Gaussian parameter.
- $\eta = C_1(m\lambda + mLT)^{-1}$ and $J = (1+LT/\lambda)\left(C_2 + \log\left(T^3L\lambda^{-1}\log(1/\delta)\right)\right)/C_1$ for some positive constant C_1 and C_2
- $m \geq C\max\left\{\sqrt{\lambda}L^{-3/2}\left[\log\left(TKL^2/\delta\right)\right]^{3/2}, T^6K^6L^6\log(TKL/\delta)\max\left\{\lambda_0^{-4}, 1\right\}\right\}$

The authors have shown that, with probability at least $1-\delta$, the regret of Algorithm 1 verify

$$R_T \leq C_2\left(1+c_T\right)\nu\sqrt{2\lambda L(\widetilde{d}\log(1+TK)+1)T} + \left(4 + C_3\left(1+c_T\right)\nu L\right)\sqrt{2\log(3/\delta)T} + 5$$

where C_2, C_3 are constants, and $c_T = \sqrt{4\log T + 2\log K}$.

It's a pretty heavy result but the main information here lies on the regret bound they obtained which is in $O(\sqrt{T\log(T)})$, like the state of the art algorithms solving this kind of problem.

3 About the demonstration of the regret analysis

The demonstration of the regret analysis in the case of Thompson Sampling applied to linear contextual bandit is already not so easy. But this time the authors needed to go deeper into the model to track the approximation error of the Artificial Neural Network. This part of the demonstration is certainly the more important concerning this paper.

This is achieved by using the theory of the Neural Tangent Kernel (NTK) see in the first section. The main idea is to achieve a particular size of the network (in terms of width m) such that the theory of NTK is true with probability $1-\delta$ for δ fixed by the user.

In the regret analysis of the paper, they used a particular event called $\mathcal{E}_t^\mu$ that should be interpreted as the event of the neural network outputting a good approximation of the non-linear function h. Then every other results are conditioned on this event with a condition on the width of the network.

4 Comments about their experiments

In order to illustrate the potential of their strategy in real situation, they tested it with some classification problems. It could seem surprising since this is not the kind of problem for which we could think for their algorithm. However, they adapted the problems so that it could match with their scenario. Their idea consisted in transforming each observation x of a dataset into K context vectors x_k where K is the number of labels. Concretely, they transformed x into $\mathbf{x}_1 = (\mathbf{x};\mathbf{0};\cdots;\mathbf{0}), \mathbf{x}_2 = (\mathbf{0};\mathbf{x};\cdots;\mathbf{0}), \cdots, \mathbf{x}_k = (\mathbf{0};\mathbf{0};\cdots;\mathbf{x})$. Then, if we call y_x the label associated to x, the reward obtained pulling x_k is simply equal to 1 if $k = y_x$ and 0 else. So, basically they considered a very simple situation with no noise implied.

Concerning their implementation, they absolutely not used the assumptions introduced before which ensured the great regret bound. For instance, the context vectors don't satisfy the symmetry condition defined above. Furthermore, they chose a rather little m, $L = 1$ and have tuned their hyperparameters with a grid-search enabling them to present the best results possible for their algorithm. We can also note that the propositions for these hyperparemeters were rather arbitrary and not linked to the theory. However, if we think about it, the condition on m for instance is really difficult to ensure for large horizon time. Indeed, if we follow the condition on m seen previously, we should have a m greater than T^6 and hence, **with $T = 100$, a very small horizon time, m should be greater than** 10^{12} ! Obviously, this is too high and using such value for m leads to an intractable computation. In reality, even small values of m such as 1000 imply a long training time as we will explain later. That's why, for their experiments, the authors finally restricted themselves to $m = 100$. Despite of these gaps between their theoretical assumptions and their implementation, they still succeed in outperforming the state of the art strategies for different classification tasks and this is remarkable.

They also tested their strategy in the case where the rewards are delayed and showed empirically that it is performing better than the NeuralUCB baseline which was proposed also by themselves in an other paper. Now, it's time to test if their strategy is reproducible and leads to good results in practice.

5 Let's make some Experiments !

We chose to test the NeuralTS strategy simulating situation where h is non-linear. Since we observed in practice that this algorithm was very slow, we restrict ourself to the following settings, $m = 20$,$L = 3$,$\lambda = 1$, $d = 10$, $K = 10$, $N_{runs} = 20$, $T = 500$. Moreover, we approximated the inverse of the design matrix as the inverse of the diagonal elements just like the authors. At last, we also wanted to see if adding a bias to the linear layers could help the network to learn. Hence, we implemented a variation of the NeuralTS using biases to see if it was a good idea. All our implementations are available in the github repository : `https://github.com/RonyAbecidan/NeuralTS`. For our simulations, we used $h_1(x) = \frac{1}{d}\sum_{i=1}^{d}\sin(x_i)$ and $h_2(x) = \frac{1}{d}\sum_{i=1}^{d}\sin(\dfrac{1}{x_i})$ and $h_3(x) = \frac{1}{d}\sum_{i=1}^{d}|x_i|$ with $\xi \sim \mathcal{N}(0, \sigma^2 I_d)$ and $\sigma = 0.001$. Furthermore, we considered other strategies as baselines such as FTL,UCB,LinTS and LinUCB ajusted with the theory so that they could provide good regret bounds.

5.1 First case : K fixed arms

In that situation, the authors didn't expect from the NeuralTS to behave as state of the art strategies and they were right. We have simulated that situation and we end up with the following regret curves (respective to each h defined above).

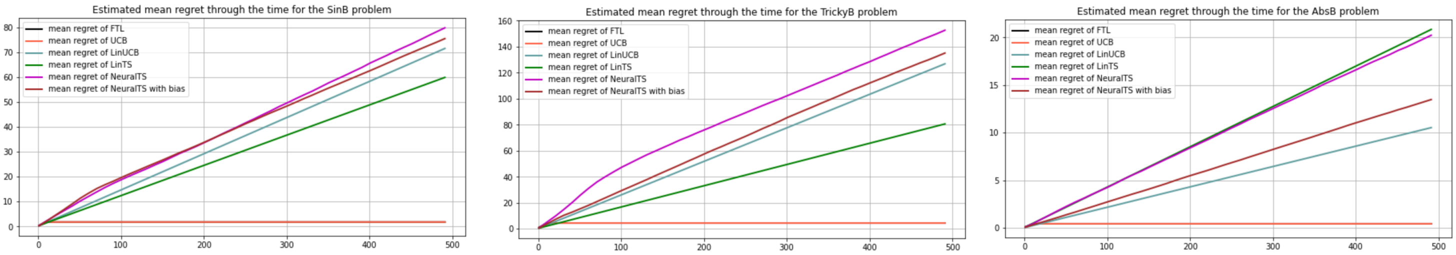

Figure 5: Results obtained with the case of K fixed arms over 20 runs

As we could observe, both NeuralTS strategies performed poorly in general even if they succeed in having a smaller regret bound than the LinTS for the last case. How could we explain that ?

We could potentially explain it by the fact that we are trying to predict h_1, h_2 and h_3 using only 10 points living in dimension $d = 10$. Because of that, the neural network does not succeed in capturing h_1 and h_2 because it doesn't have enough information to make a good guess. However, we observe that the strategy which uses the bias performed better than the original one proposed by the authors for the 3 cases. Hence that confirm in some sense our previous thoughts. Now, if we consider a situation where the arms are updated at each round, it could change the story.

5.2 Second case : K changing arms

Here, we are in the same scenario than the one imagined by the authors, i.e. , the K contexts are changing at every time. You can find below the regret curves found in that case.

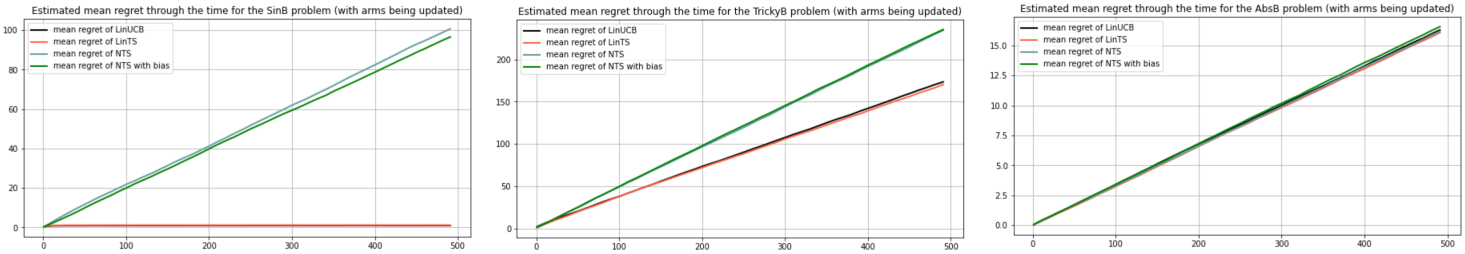

Figure 6: Results obtained with the case of K changing arms over 20 runs

For the first case, LinTS and LinUCB behaved impressively. In fact this is maybe not surprising since the mean reward function can be approximated by a linear function approximately around 0. Indeed, here the 10-dimensional contexts are normalized such that their norms are equal to 1 and so their coefficients are small. Given that situation, using many points, the two linear strategies should succeed in capturing very precisely the tangent hyperplane of the mean reward around 0. However, the NeuralTS strategies performed very poorly leading to linear regrets. Maybe the network we used was too complex for capturing the mean reward function which behaves almost as a linear function around 0.

For the second case, it is more mitigated. All the regrets were linear but still the LinTS and the LinUCB performed the best surprisingly and yet the reward function is really tricky and not smooth at all around 0. At last, in the third case, all the regrets were linear and very close.

We think we didn't succeed in exploiting the potential of the NeuralTS because of time constraints. Even if the strategy seems good, it is very slow to execute and we didn't have the time to do a grid search for the optimal hyperparameters λ,ν or testing different values of L and m. Hence, that could explain why our implementation of the NeuralTS is performing so poorly compared to strategies tuned correctly.

Conclusion

From one side, the authors proposed a new strategy enabling to solve the famous contextual bandit problem linking deep learning methods with the Thompson Sampling strategy. They showed both theoretically and empirically that this strategy can reveal itself to be among the best ones for solving this problem with a competitive regret bound. However, from an other side, it was a really difficult paper to read and grasp since it implicitly considers that the reader is an expert of the NTK theory which is not easy to follow and rather recent. Moreover, a lot of assumptions are hidden and we didn't succeed in entirely linking the LinTS strategy with the NeuralTS one before a while. Furthermore, our experiments made us understand that this strategy is really difficult to apply in real life for time-constrained situations since it is very slow. It even appeared that there is a non-negliglible time needed for tuning the hyperparameters of the network such that it could end up to a better solution than baseline methods. Hence, the results are mixed. Nevertheless, we congrats the authors for having such an interesting strategy because it's certainly not something that anyone could have found.

References

[1] Weitong Zhang et al. *Neural Thompson Sampling*. 2020. arXiv: 2010.00827 [cs.LG].

[2] Rajat Vadiraj Dwaraknath. *Understanding the Neural Tangent Kernel*. 2019.

[3] Sarah Filippi et al. "Parametric Bandits: The Generalized Linear Case". In: *Advances in Neural Information Processing Systems*. Ed. by J. Lafferty et al. Curran Associates, Inc., pp. 586–594.