Neural Contextual Bandits with UCB-based Exploration

Dongruo Zhou [1] Lihong Li [2] Quanquan Gu [1]



[1]Department of Computer Science, UCLA

[2]Google Research

Outline

- Background
  - Contextual bandit problem
  - Deep neural networks

Outline

- Background
  - Contextual bandit problem
  - Deep neural networks
- Algorithm – NeuralUCB
  - Use a neural network to learn the reward
  - Use neural network's gradient to explore
  - Upper confidence bound strategy

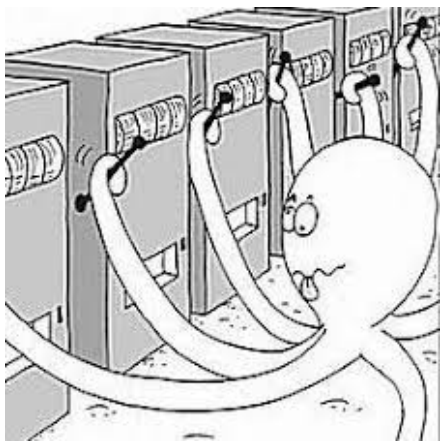
Outline

- Background
  - Contextual bandit problem
  - Deep neural networks
- Algorithm – NeuralUCB
  - Use a neural network to learn the reward
  - Use neural network's gradient to explore
  - Upper confidence bound strategy
- Main theory
  - Neural tangent kernel matrix and effective dimension
  - $\tilde{O}(\sqrt{T})$ regret

Background – decision-making problems

Decision-making problems are everywhere!

- As a gambler in a casino, find a slot machine, you will...
  - Limited budget, maximize the payoff !
  - Which arm to pull?
- As a movie recommender, you need to...
  - Recommend movies based on users' interests, maximize users' purchase rate
  - Which movie to recommend?



(a) Slot machine

(b) Movie recommendation

Background – contextual bandit

K-armed contextual bandit problem: movie recommendation

Background – contextual bandit

K-armed contextual bandit problem: movie recommendation
At round t,

- Agent observes K d-dimensional contextual vectors (user's movie purchase history)

$$\{x_{t,a} \in \mathbb{R}^d \mid a \in [K]\}$$

Background – contextual bandit

K-armed contextual bandit problem: movie recommendation
At round t,

- Agent observes K d-dimensional contextual vectors (user's movie purchase history)

$$\{x_{t;a} \in \mathbb{R}^d \mid a \in [K]\}$$

- Agent selects an action a_t and receives a reward $r_{t;a_t}$ (recommends some movie and user choose to purchase or not)

Background – contextual bandit

K-armed contextual bandit problem: movie recommendation
At round t,

- Agent observes K d-dimensional contextual vectors (user's movie purchase history)

$$\{\mathbf{x}_{t;a} \in \mathrm{R}^d \mid a \in [K]\}$$

- Agent selects an action a_t and receives a reward $r_{t;a_t}$ (recommends some movie and user choose to purchase or not)
- The goal is to minimize the following pesudo regret

$$R_T = \mathrm{E}\left[\sum_{t=1}^{T} (r_{t;a_t^*} - r_{t;a_t})\right]$$

where $a_t = \mathrm{argmax}_{a \in [K]} \mathrm{E}[r_{t;a}]$ is the optimal action at round t

Background – contextual linear bandit

$$r_{t;a_t} = h \quad ; x_{t;a_t} i + \quad _t, \quad _t \qquad \text{-sub-Gaussian}$$

Background – contextual linear bandit

$$r_{t;a_t} = h \quad ; x_{t;a_t} i + \quad _t, \quad _t \qquad \text{-sub-Gaussian}$$

- Build confidence set for and use *optimism in the face of uncertainty* (OFU) principle

Background – contextual linear bandit

$$r_{t;a_t} = \langle \quad ; x_{t;a_t} \rangle + \quad _t, \quad _t \qquad \text{-sub-Gaussian}$$

- Build confidence set for and use *optimism in the face of uncertainty* (OFU) principle
- Leads to $\tilde{\mathcal{O}}(d\sqrt{T})$ regret (Abbasi-Yadkori et al. 2011)
- Strongly depends on linear structure!

Background – general reward function

$r_{t;a_t} = h(x_{t;a_t}) + {}_t$, 0 $h(x)$ 1, $_t$ -sub-Gaussian

Background – general reward function

$$r_{t;a_t} = h(\mathrm{x}_{t;a_t}) + \ _t, \ 0 \quad h(\mathrm{x}) \quad 1, \quad _t \quad \text{-sub-Gaussian}$$

- Including many popular contextual bandit problems
  - Linear bandit
    - $h(\mathrm{x}) = h \ ; \mathrm{x}i$, where $k \ k_2 \quad 1, k\mathrm{x}k_2 \quad 1$
  - Generalized linear bandit
    - $h(\mathrm{x}) = g(h \ ; \mathrm{x}i)$, where $k \ k_2 \quad 1, k\mathrm{x}k_2 \quad 1, j \ r \ gj \quad 1$

Background – general reward function

$$r_{t;a_t} = h(\mathrm{x}_{t;a_t}) + \ _t, \ 0 \quad h(\mathrm{x}) \quad 1, \quad _t \quad \text{-sub-Gaussian}$$

- Including many popular contextual bandit problems
  - Linear bandit
    - $h(\mathrm{x}) = h \ ; \mathrm{x}i$, where $k \ k_2 \quad 1, k\mathrm{x}k_2 \quad 1$
  - Generalized linear bandit
    - $h(\mathrm{x}) = g(h \ ; \mathrm{x}i)$, where $k \ k_2 \quad 1, k\mathrm{x}k_2 \quad 1, j \ r \ gj \quad 1$

We do not know what h is...

Background – general reward function

$$r_{t;a_t} = h(\mathbf{x}_{t;a_t}) + \quad _t,\ 0 \quad h(\mathbf{x}) \quad 1, \quad _t \quad \text{-sub-Gaussian}$$

- Including many popular contextual bandit problems
  - Linear bandit
    - $h(\mathbf{x}) = h \ ; \mathbf{x}i$, where $k \ k_2 \quad 1, k\mathbf{x}k_2 \quad 1$
  - Generalized linear bandit
    - $h(\mathbf{x}) = g(h \ ; \mathbf{x}i)$, where $k \ k_2 \quad 1, k\mathbf{x}k_2 \quad 1, j \ r \ gj \quad 1$

We do not know what h is...

Use some universal function approximator, such as neural networks!

Background – neural network

Fully connected neural networks:

$$f(x; \quad) = \overline{m}\, W_L \quad W_{L\ 1} \qquad (W_1 x)$$

Background – neural network

Fully connected neural networks:

$$f(\mathrm{x};\ ) = {}^{p}\overline{m}\mathrm{W}_L \quad \mathrm{W}_{L\ 1} \qquad (\mathrm{W}_1\mathrm{x})$$

- $(x) = \max\{x; 0\}$ is the ReLU activation function

Background – neural network

Fully connected neural networks:

$$f(\mathrm{x};\ ) = \sqrt[p]{m}\,\mathrm{W}_L \quad \mathrm{W}_{L\ 1} \qquad (\mathrm{W}_1\mathrm{x})$$

- $(x) = \max\{x; 0\}$ is the ReLU activation function
- W_i is the weight matrix
  - $\mathrm{W}_1 \in \mathrm{R}^{m\ d}$
  - $\mathrm{W}_i \in \mathrm{R}^{m\ m}; 2 \quad i \quad L \quad 1$
  - $\mathrm{W}_L \in \mathrm{R}^{m\ 1}$

Background – neural network

Fully connected neural networks:

$$f(\mathrm{x};\ ) = \frac{p}{m}\mathrm{W}_L \quad \mathrm{W}_{L\ 1} \qquad (\mathrm{W}_1\mathrm{x})$$

- $(x) = \max\{x; 0\}$ is the ReLU activation function
- $= [\mathrm{vec}(\mathrm{W}_1)^\top; \ldots; \mathrm{vec}(\mathrm{W}_L)^\top]^\top \in \mathrm{R}^p$, $p = m + md + m^2(L \quad 1)$

Background – neural network

Fully connected neural networks:

$$f(\mathbf{x};\ ) = \frac{p}{m}\mathbf{W}_L \quad \mathbf{W}_{L\ 1} \qquad (\mathbf{W}_1\mathbf{x})$$

- $(x) = \max\{x; 0\}$ is the ReLU activation function
- $= [\mathrm{vec}(\mathbf{W}_1)^\top; \ldots; \mathrm{vec}(\mathbf{W}_L)^\top]^\top \in \mathbb{R}^p$, $p = m + md + m^2(L\ \ 1)$
- Gradient of the neural network $\mathbf{g}(\mathbf{x};\ ) = r_\theta f(\mathbf{x};\ ) \in \mathbb{R}^p$

Question

- Neural network-based contextual bandit algorithms (Riquelme et al. 2018; Zahavy and Mannor 2019)
- No theoretical guarantee

Question

- Neural network-based contextual bandit algorithms (Riquelme et al. 2018; Zahavy and Mannor 2019)
- No theoretical guarantee

Can we design provably efficient neural network-based algorithm to learn the general reward function?

Question

- Neural network-based contextual bandit algorithms (Riquelme et al. 2018; Zahavy and Mannor 2019)
- No theoretical guarantee

Can we design provably e cient neural network-based algorithm to learn the general reward function?

Yes! NeuralUCB

- Neural network to model reward function, UCB strategy to explore
- Theoretical guarantee on regret $\tilde{\mathcal{O}}(\sqrt{T})$
- Matches regret bound for linear setting (Abbasi-Yadkori et al. 2011)

NeuralUCB { initialization

- Special initialization on $_0$
  - For $1 \quad l \quad L \quad 1$,

$$W_l = \begin{matrix} W & 0 \\ 0 & W \end{matrix} ; W_{f\,i;j\,g} \quad N(0; 4=m)$$

  - For L, $W = (w^{>}; \quad w^{>})$, $w_{f\,i\,g} \quad N(0; 2=m)$

NeuralUCB { initialization

- Special initialization on $_0$
  - For $1 \quad l \quad L \quad 1$,

$$W_l = \begin{matrix} W & 0 \\ 0 & W \end{matrix} ; W_{f\,i;j\,g} \quad N(0; 4=m)$$

  - For L, $W = (w^{>}; \quad w^{>})$, $w_{f\,i\,g} \quad N(0; 2=m)$
- Normalization onf $x^i g$: for any $1 \quad i \quad TK$, $kx^i k_2 = 1$ and $[x^i]_j = [x^i]_{j + d=2}$
  - For any unit vector$x$, construct $x^0 = (x; x)=\sqrt{2}$

NeuralUCB { initialization

- Special initialization on $_0$
  - For $1 \quad l \quad L \quad 1$,

$$W_l = \begin{matrix} W & 0 \\ 0 & W \end{matrix} ; W_{f\,i;j\,g} \quad N(0; 4=m)$$

  - For L, $W = (w^{>}; \quad w^{>})$, $w_{f\,i\,g} \quad N(0; 2=m)$
- Normalization onf $x^i g$: for any $1 \quad i \quad TK$, $kx^i k_2 = 1$ and $[x^i]_j = [x^i]_{j + d=2}$
  - For any unit vector$x$, construct $x^0 = (x; x)=\sqrt{2}$

Guarantee thatf $(x^i; \quad _0) = 0$!

NeuralUCB { upper con dence bounds

At round t, NeuralUCB will...

- Observe $\{x_{t,a}\}_{a=1}^{K}$

NeuralUCB { upper con dence bounds

At round t, NeuralUCB will...

- Observe $\{x_{t;a}\}_{a=1}^{K}$
- Compute upper con dence bound for each arm a, which is

$$U_{t;a} = \underbrace{f(x_{t;a}; \quad _{t\ 1})}_{\text{mean}} + \quad _{t\ 1} \underbrace{\sqrt{\frac{g(x_{t;a}; \quad _{t\ 1})^{>} Z_{t}^{\ 1}{}_{1} g(x_{t;a}; \quad _{t\ 1})}{m}}}_{\text{variance}}$$

NeuralUCB { upper con dence bounds

At round t, NeuralUCB will...

- Observe $\{x_{t;a}\}_{a=1}^{K}$
- Compute upper con dence bound for each arm a, which is

$$U_{t;a} = \underbrace{f(x_{t;a}; \quad _{t\ 1})}_{\text{mean}} + \quad _{t\ 1} \underbrace{\sqrt{g(x_{t;a}; \quad _{t\ 1})^{>} Z_{t}^{\ 1}{}_{1} g(x_{t;a}; \quad _{t\ 1})/m}}_{\text{variance}}$$

Compared with LinUCB (Li et al. 2010)

$$U_{t;a} = \underbrace{x_{t;a}; \quad _{t\ 1}}_{\text{mean}} + \quad _{t\ 1} \underbrace{\sqrt{x_{t;a}^{>} Z_{t}^{\ 1}{}_{1} x_{t;a}}}_{\text{variance}}$$

NeuralUCB { upper con dence bounds

At round t, NeuralUCB will...

- Observe $\{x_{t;a}\}_{a=1}^{K}$
- Compute upper con dence bound for each arm a, which is

$$U_{t;a} = \underbrace{f(x_{t;a}; \ _{t\ 1})}_{\text{mean}} + \ _{t\ 1} \underbrace{\sqrt{g(x_{t;a}; \ _{t\ 1})^{>} Z_{t}^{\ 1}{}_{1} g(x_{t;a}; \ _{t\ 1})=m}}_{\text{variance}}$$

Compared with LinUCB (Li et al. 2010)

$$U_{t;a} = \underbrace{x_{t;a}; \ _{t\ 1}}_{\text{mean}} + \ _{t\ 1} \underbrace{\sqrt{x_{t;a}^{>} Z_{t}^{\ 1}{}_{1} x_{t;a}}}_{\text{variance}}$$

- Select $a_t = \operatorname{argmax}_{a2[K]} U_{t;a}$, play a_t and observe reward $r_{t;a_t}$

NeuralUCB { update parameter

After receiving reward, NeuralUCB will...

- Update Z_t

$$Z_t = Z_{t\ \ 1} + g(x_{t;a_t};\ \ _{t\ \ 1})g(x_{t;a_t};\ \ _{t\ \ 1})^{>} = m$$

NeuralUCB { update parameter

After receiving reward, NeuralUCB will...

- Update Z_t

$$Z_t = Z_{t\ 1} + g(x_{t;a_t};\ _{t\ 1})g(x_{t;a_t};\ _{t\ 1})^{>} = m$$

- Update $_t$ using gradient descent
  - Denote loss function $L(\ )$ as

$$L(\ ) = \sum_{i=1}^{t} (f(x_{i;a_i};\ )\quad r_{i;a_i})^2 = 2 + m\ \ k \qquad {}^{(0)}\ k_2^2 = 2$$

NeuralUCB { update parameter

After receiving reward, NeuralUCB will...

- Update Z_t

$$Z_t = Z_{t-1} + g(x_{t;a_t}; \theta_{t-1}) g(x_{t;a_t}; \theta_{t-1})^\top / m$$

- Update θ_t using gradient descent
  - Denote loss function $L(\theta)$ as

$$L(\theta) = \sum_{i=1}^{t} (f(x_{i;a_i}; \theta) - r_{i;a_i})^2 / 2 + m \lambda \theta^{(0)} \theta / 2$$

  - Run J step gradient descent on $L(\theta)$ starting from θ_0, take θ_t as the last iterate

$$\theta^{(0)} = \theta_0; \quad \theta^{(j+1)} = \theta^{(j)} - \eta \nabla L(\theta^{(j)}); \quad \theta_t = \theta^{(J)}$$

NeuralUCB { con dence radius

After update neural network function, NeuralUCB will compute $_t$, which is ...

- Under the overparameterized setting $m(\quad 1)$,

$$_t = O\Bigg(\underbrace{\sqrt{S + \log\frac{\det Z_t}{\det \quad I}}}_{\text{con dence radius}} + \underbrace{(\quad + tL)(1 \quad m \quad)^{J=2}\sqrt{\frac{}{t=}}}_{\text{function approximation error}}\Bigg)$$

NeuralUCB { con dence radius

After update neural network function, NeuralUCB will compute $_t$, which is ...

- Under the overparameterized setting (m 1),

$$_t = O\Big(\underbrace{\sqrt{S} + \sqrt{\log \frac{\det Z_t}{\det \ I}}}_{\text{con dence radius}} + \underbrace{(\ + tL)(1 \quad m \ )^{J=2}\sqrt{\frac{}{t=}}}_{\text{function approximation error}} \Big)$$

Compared with LinUCB,

$$_t = O\Big(\sqrt{S} + \sqrt{\log \frac{\det Z_t}{\det \ I}} \Big)$$

no function approximation error part!

Main theory { assumptions

Assumption

There exists $_0 > 0$ such that $H \quad _0 I$, where H is the neural tangent kernel matrix (Jacot et al. 2018; Cao and Gu 2019) on contexts $f x^i g_{i=1}^{TK}$.

Main theory { assumptions

Assumption

There exists $_0 > 0$ such that $H \quad _0 I$, where H is the neural tangent kernel matrix (Jacot et al. 2018; Cao and Gu 2019) on contexts $f x^i g_{i=1}^{TK}$.

- Satis ed if no two contexts $\inf x^i g_{i=1}^{TK}$ are parallel.

Main theory { assumptions

Assumption

There exists $_0 > 0$ such that $H \quad _0 I$, where H is the neural tangent kernel matrix (Jacot et al. 2018; Cao and Gu 2019) on contexts $f x^i g_{i=1}^{TK}$.

- Satis ed if no two contexts $\inf f x^i g_{i=1}^{TK}$ are parallel.

De nition

The e ective dimension d^e of the neural tangent kernel matrix on contexts $f x^i g_{i=1}^{TK}$ is de ned as $d^e = \log \det(I + H =) = \log(1 + TK =)$.

Main theory { assumptions

Assumption

There exists $_0 > 0$ such that $H \quad _0 I$, where H is the neural tangent kernel matrix (Jacot et al. 2018; Cao and Gu 2019) on contexts $f x^i g_{i=1}^{TK}$.

- Satis ed if no two contexts $\inf f x^i g_{i=1}^{TK}$ are parallel.

De nition

The e ective dimension d^e of the neural tangent kernel matrix on contexts $f x^i g_{i=1}^{TK}$ is de ned as $d^e = \log \det(I + H =) = \log(1 + TK =)$.

- Notion adapted from Valko et al. (2013) and Yang and Wang (2019)

Main theory { assumptions

Assumption

There exists $_0 > 0$ such that $H \quad _0 I$, where H is the neural tangent kernel matrix (Jacot et al. 2018; Cao and Gu 2019) on contexts $f x^i g_{i=1}^{TK}$.

- Satis ed if no two contexts $\inf f x^i g_{i=1}^{TK}$ are parallel.

De nition

The e ective dimension d^e of the neural tangent kernel matrix on contexts $f x^i g_{i=1}^{TK}$ is de ned as $d^e = \log \det(I + H =) = \log(1 + TK =)$.

- Notion adapted from Valko et al. (2013) and Yang and Wang (2019)
- $d^e \quad \log T$ in several special cases (Valko et al. 2013)

Main theory { regret bound

Theorem

Let $h = [h(x^i)]_{i=1}^{TK} \in R^{TK}$. Set $J = \frac{e}{p}(TL=)$, $= ((mTL + m)^{1})$ and $S = 2\sqrt{h^{>} H^{1} h}$. Under the overparameterized setting $m($ 1), with probability at least 1 ,

$$R_T = \mathcal{O}\left(\frac{p}{e} \frac{q}{dT} \max\{d, S^2\}\right) :$$

Main theory – regret bound

Theorem

Let $\mathbf{h} = [h(\mathbf{x}^i)]_{i=1}^{TK} \in \mathbb{R}^{TK}$. *Set* $J = \sqrt{e(TL =)}$, $= ((mTL + m\ )^{-1})$ *and* $S = 2\sqrt{\mathbf{h}^\top \mathbf{H}^{-1}\mathbf{h}}$. *Under the overparameterized setting* ($m \quad 1$), *with probability at least* $1 \quad$,

$$R_T = \mathcal{O}\left(\sqrt{\frac{\log \frac{1}{\delta}}{T}}\max\{d, S^2\}\right) :$$

- h belongs to the RKHS space H spanned by $\mathbf{H}\ )\ S \quad \|h\|_H$

Main theory – regret bound

Theorem

Let $\mathbf{h} = [h(\mathbf{x}^i)]_{i=1}^{TK} \in \mathbb{R}^{TK}$. *Set* $J = \sqrt{e(TL =)}$, $= ((mTL + m) \ ^{1})$ *and* $S = 2\sqrt{\mathbf{h}^{\top}\mathbf{H} \ ^{1}\mathbf{h}}$. *Under the overparameterized setting* ($m \quad 1$), *with probability at least* $1 \quad$,

$$R_T = \mathcal{O}\left(\sqrt{\frac{}{dT}}\sqrt{\max\{d, S^2\}}\right) :$$

- h belongs to the RKHS space H spanned by $\mathbf{H} \ ) \ S \quad \|h\|_H$
- R_T does not depend on p, the dimension of the dynamic feature mapping $\mathbf{g}(\mathbf{x}; \ )$

Main theory – regret bound

Theorem

Let $\mathbf{h} = [h(\mathbf{x}^i)]_{i=1}^{TK} \in \mathbb{R}^{TK}$. *Set* $J = \sqrt{\mathbf{h}^\top \mathbf{H}^{-1}\mathbf{h}}$, $= ((mTL + m)^{-1})$ *and* $S = 2\sqrt{\mathbf{h}^\top \mathbf{H}^{-1}\mathbf{h}}$. *Under the overparameterized setting* ($m \geq 1$), *with probability at least* $1 - \delta$,

$$R_T = \tilde{\mathcal{O}}\left(\frac{\sqrt{d\tilde{d}T}}{\max\{d\tilde{d}, S^2\}}\right).$$

- h belongs to the RKHS space H spanned by $\mathbf{H}$ ($\sqrt{\mathbf{h}^\top \mathbf{H}^{-1}\mathbf{h}} \leq S / \|h\|_H$)
- R_T does not depend on p, the dimension of the dynamic feature mapping $\mathbf{g}(\mathbf{x}; \theta)$
- Recover the regret for linear contextual bandit $\tilde{\mathcal{O}}(d\sqrt{T})$ (Abbasi-Yadkori et al. 2011)

Takeaway message

- NeuralUCB uses neural network $f(\mathbf{x}; \quad_t)$ to predict, gradient $\mathbf{g}(\mathbf{x}; \quad_t)$ to explore

Takeaway message

- NeuralUCB uses neural network $f(\mathbf{x}; \ _t)$ to predict, gradient $\mathbf{g}(\mathbf{x}; \ _t)$ to explore
- NeuralUCB achieves $\tilde{\mathcal{O}}(\sqrt{T})$ regret, matches result for linear setting

Takeaway message

- NeuralUCB uses neural network $f(\mathbf{x}; \ _t)$ to predict, gradient $\mathbf{g}(\mathbf{x}; \ _t)$ to explore
- NeuralUCB achieves $\tilde{\mathcal{O}}(\sqrt{T})$ regret, matches result for linear setting
- NeuralUCB also works well empirically

Takeaway message

- NeuralUCB uses neural network $f(\mathbf{x}; \theta_t)$ to predict, gradient $g(\mathbf{x}; \theta_t)$ to explore
- NeuralUCB achieves $\tilde{O}(\sqrt{T})$ regret, matches result for linear setting
- NeuralUCB also works well empirically

Thank you!