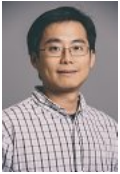
Neural Contextual Bandits with UCB-based Exploration

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

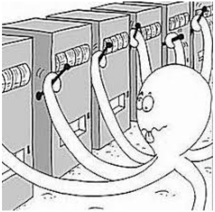
Outline

- 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
  - $\widetilde{O}(\sqrt{T})$ 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)

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

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)

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

- 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)

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

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

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

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

Background – contextual linear bandit

$$r_{t,a_t} = \langle \boldsymbol{\theta}^*, \mathbf{x}_{t,a_t} \rangle + \xi_t,\ \xi_t \sim \nu\text{-sub-Gaussian}$$

Background – contextual linear bandit

$$r_{t,a_t} = \langle \boldsymbol{\theta}^*, \mathbf{x}_{t,a_t} \rangle + \xi_t,\ \xi_t \sim \nu\text{-sub-Gaussian}$$

- Build confidence set for $\boldsymbol{\theta}^*$ and use *optimism in the face of uncertainty* (OFU) principle

Background – contextual linear bandit

$$r_{t,a_t} = \langle \boldsymbol{\theta}^*, \mathbf{x}_{t,a_t} \rangle + \xi_t,\ \xi_t \sim \nu\text{-sub-Gaussian}$$

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

Background – general reward function

$$r_{t,a_t} = h(\mathbf{x}_{t,a_t}) + \xi_t,\ 0 \le h(\mathbf{x}) \le 1,\ \xi_t \sim \nu\text{-sub-Gaussian}$$

Background – general reward function

$$r_{t,a_t} = h(\mathbf{x}_{t,a_t}) + \xi_t,\ 0 \leq h(\mathbf{x}) \leq 1,\ \xi_t \sim \nu\text{-sub-Gaussian}$$

- Including many popular contextual bandit problems
  - Linear bandit
    - $h(\mathbf{x}) = \langle \boldsymbol{\theta}, \mathbf{x} \rangle$, where $\|\boldsymbol{\theta}\|_2 \leq 1$, $\|\mathbf{x}\|_2 \leq 1$
  - Generalized linear bandit
    - $h(\mathbf{x}) = g(\langle \boldsymbol{\theta}, \mathbf{x} \rangle)$, where $\|\boldsymbol{\theta}\|_2 \leq 1$, $\|\mathbf{x}\|_2 \leq 1$, $|\nabla g| \leq 1$

Background – general reward function

$$r_{t,a_t} = h(\mathbf{x}_{t,a_t}) + \xi_t,\ 0 \leq h(\mathbf{x}) \leq 1,\ \xi_t \sim \nu\text{-sub-Gaussian}$$

- Including many popular contextual bandit problems
  - Linear bandit
    - $h(\mathbf{x}) = \langle \boldsymbol{\theta}, \mathbf{x} \rangle$, where $\|\boldsymbol{\theta}\|_2 \leq 1$, $\|\mathbf{x}\|_2 \leq 1$
  - Generalized linear bandit
    - $h(\mathbf{x}) = g(\langle \boldsymbol{\theta}, \mathbf{x} \rangle)$, where $\|\boldsymbol{\theta}\|_2 \leq 1$, $\|\mathbf{x}\|_2 \leq 1$, $|\nabla g| \leq 1$

We do not know what h is...

Background – general reward function

$$r_{t,a_t} = h(\mathbf{x}_{t,a_t}) + \xi_t,\ 0 \le h(\mathbf{x}) \le 1,\ \xi_t \sim \nu\text{-sub-Gaussian}$$

- Including many popular contextual bandit problems
  - Linear bandit
    - $h(\mathbf{x}) = \langle \boldsymbol{\theta}, \mathbf{x} \rangle$, where $\|\boldsymbol{\theta}\|_2 \le 1$, $\|\mathbf{x}\|_2 \le 1$
  - Generalized linear bandit
    - $h(\mathbf{x}) = g(\langle \boldsymbol{\theta}, \mathbf{x} \rangle)$, where $\|\boldsymbol{\theta}\|_2 \le 1$, $\|\mathbf{x}\|_2 \le 1$, $|\nabla g| \le 1$

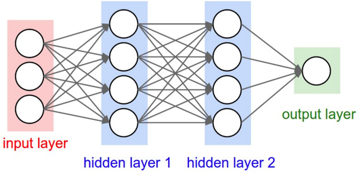
We do not know what h is...

Use some universal function approximator, such as neural networks!

Background – neural network

Fully connected neural networks:

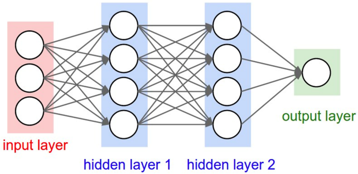
$$f(\mathbf{x};\boldsymbol{\theta}) = \sqrt{m}\mathbf{W}_L\sigma\Big(\mathbf{W}_{L-1}\sigma\big(\cdots\sigma(\mathbf{W}_1\mathbf{x})\big)\Big)$$



Background – neural network

Fully connected neural networks:

$$f(\mathbf{x};\boldsymbol{\theta}) = \sqrt{m}\mathbf{W}_L\sigma\Big(\mathbf{W}_{L-1}\sigma\big(\cdots\sigma(\mathbf{W}_1\mathbf{x})\big)\Big)$$

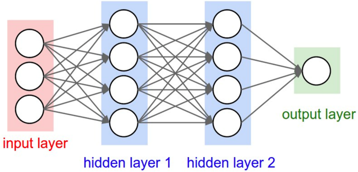


- $\sigma(x) = \max\{x, 0\}$ is the ReLU activation function

Background – neural network

Fully connected neural networks:

$$f(\mathbf{x};\boldsymbol{\theta}) = \sqrt{m}\mathbf{W}_L\sigma\Big(\mathbf{W}_{L-1}\sigma\big(\cdots\sigma(\mathbf{W}_1\mathbf{x})\big)\Big)$$

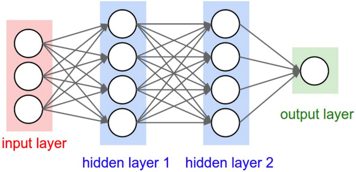


- $\sigma(x) = \max\{x, 0\}$ is the ReLU activation function
- $\mathbf{W}_i$ is the weight matrix
  - $\mathbf{W}_1 \in \mathbb{R}^{m\times d}$
  - $\mathbf{W}_i \in \mathbb{R}^{m\times m}, 2 \le i \le L-1$
  - $\mathbf{W}_L \in \mathbb{R}^{m\times 1}$

Background – neural network

Fully connected neural networks:

$$f(\mathbf{x};\boldsymbol{\theta}) = \sqrt{m}\mathbf{W}_L\sigma\Big(\mathbf{W}_{L-1}\sigma\big(\cdots\sigma(\mathbf{W}_1\mathbf{x})\big)\Big)$$

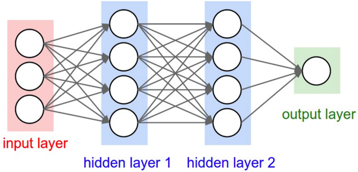


- $\sigma(x) = \max\{x, 0\}$ is the ReLU activation function
- $\boldsymbol{\theta} = [\mathsf{vec}(\mathbf{W}_1)^\top, \ldots, \mathsf{vec}(\mathbf{W}_L)^\top]^\top \in \mathbb{R}^p$, $p = m + md + m^2(L-1)$

Background – neural network

Fully connected neural networks:

$$f(\mathbf{x};\boldsymbol{\theta}) = \sqrt{m}\mathbf{W}_L\sigma\Big(\mathbf{W}_{L-1}\sigma\big(\cdots\sigma(\mathbf{W}_1\mathbf{x})\big)\Big)$$



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

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 efficient 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 $\widetilde{O}(\sqrt{T})$
- Matches regret bound for linear setting (Abbasi-Yadkori et al. 2011)

NeuralUCB – initialization

- Special initialization on $\boldsymbol{\theta}_0$
  - For $1 \leq l \leq L-1$,

$$\mathbf{W}_l = \begin{pmatrix} \mathbf{W} & \mathbf{0} \\ \mathbf{0} & \mathbf{W} \end{pmatrix}, \mathbf{W}_{\{i,j\}} \sim N(0, 4/m)$$

  - For L, $\mathbf{W} = (\mathbf{w}^\top, -\mathbf{w}^\top)$, $\mathbf{w}_{\{i\}} \sim N(0, 2/m)$

NeuralUCB – initialization

- Special initialization on $\boldsymbol{\theta}_0$
  - For $1 \leq l \leq L-1$,

$$\mathbf{W}_l = \begin{pmatrix} \mathbf{W} & \mathbf{0} \\ \mathbf{0} & \mathbf{W} \end{pmatrix}, \mathbf{W}_{\{i,j\}} \sim N(0, 4/m)$$

  - For L, $\mathbf{W} = (\mathbf{w}^\top, -\mathbf{w}^\top)$, $\mathbf{w}_{\{i\}} \sim N(0, 2/m)$
- Normalization on $\{\mathbf{x}^i\}$: for any $1 \leq i \leq TK$, $\|\mathbf{x}^i\|_2 = 1$ and $[\mathbf{x}^i]_j = [\mathbf{x}^i]_{j+d/2}$
  - For any unit vector $\mathbf{x}$, construct $\mathbf{x}' = (\mathbf{x}; \mathbf{x})/\sqrt{2}$

NeuralUCB – initialization

- Special initialization on $\boldsymbol{\theta}_0$
  - For $1 \leq l \leq L-1$,

$$\mathbf{W}_l = \begin{pmatrix} \mathbf{W} & \mathbf{0} \\ \mathbf{0} & \mathbf{W} \end{pmatrix}, \mathbf{W}_{\{i,j\}} \sim N(0, 4/m)$$

  - For L, $\mathbf{W} = (\mathbf{w}^\top, -\mathbf{w}^\top)$, $\mathbf{w}_{\{i\}} \sim N(0, 2/m)$
- Normalization on $\{\mathbf{x}^i\}$: for any $1 \leq i \leq TK$, $\|\mathbf{x}^i\|_2 = 1$ and $[\mathbf{x}^i]_j = [\mathbf{x}^i]_{j+d/2}$
  - For any unit vector $\mathbf{x}$, construct $\mathbf{x}' = (\mathbf{x}; \mathbf{x})/\sqrt{2}$

Guarantee that $f(\mathbf{x}^i; \boldsymbol{\theta}_0) = 0$!

NeuralUCB – upper confidence bounds

At round t, NeuralUCB will...

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

NeuralUCB – upper confidence bounds

At round t, NeuralUCB will...

- Observe $\{\mathbf{x}_{t,a}\}_{a=1}^{K}$
- Compute upper confidence bound for each arm a, which is

$$U_{t,a} = \underbrace{f(\mathbf{x}_{t,a}; \boldsymbol{\theta}_{t-1})}_{\text{mean}} + \gamma_{t-1} \underbrace{\sqrt{\mathbf{g}(\mathbf{x}_{t,a}; \boldsymbol{\theta}_{t-1})^\top \mathbf{Z}_{t-1}^{-1} \mathbf{g}(\mathbf{x}_{t,a}; \boldsymbol{\theta}_{t-1})/m}}_{\text{variance}}$$

NeuralUCB – upper confidence bounds

At round t, NeuralUCB will...

- Observe $\{\mathbf{x}_{t,a}\}_{a=1}^{K}$
- Compute upper confidence bound for each arm a, which is

$$U_{t,a} = \underbrace{f(\mathbf{x}_{t,a}; \boldsymbol{\theta}_{t-1})}_{\text{mean}} + \gamma_{t-1} \underbrace{\sqrt{\mathbf{g}(\mathbf{x}_{t,a}; \boldsymbol{\theta}_{t-1})^\top \mathbf{Z}_{t-1}^{-1} \mathbf{g}(\mathbf{x}_{t,a}; \boldsymbol{\theta}_{t-1})/m}}_{\text{variance}}$$

Compared with LinUCB (Li et al. 2010)

$$U_{t,a} = \underbrace{\langle \mathbf{x}_{t,a}, \boldsymbol{\theta}_{t-1} \rangle}_{\text{mean}} + \gamma_{t-1} \underbrace{\sqrt{\mathbf{x}_{t,a}^\top \mathbf{Z}_{t-1}^{-1} \mathbf{x}_{t,a}}}_{\text{variance}}$$

NeuralUCB – upper confidence bounds

At round t, NeuralUCB will...

- Observe $\{\mathbf{x}_{t,a}\}_{a=1}^{K}$
- Compute upper confidence bound for each arm a, which is

$$U_{t,a} = \underbrace{f(\mathbf{x}_{t,a}; \boldsymbol{\theta}_{t-1})}_{\text{mean}} + \gamma_{t-1} \underbrace{\sqrt{\mathbf{g}(\mathbf{x}_{t,a}; \boldsymbol{\theta}_{t-1})^\top \mathbf{Z}_{t-1}^{-1} \mathbf{g}(\mathbf{x}_{t,a}; \boldsymbol{\theta}_{t-1})/m}}_{\text{variance}}$$

Compared with LinUCB (Li et al. 2010)

$$U_{t,a} = \underbrace{\langle \mathbf{x}_{t,a}, \boldsymbol{\theta}_{t-1} \rangle}_{\text{mean}} + \gamma_{t-1} \underbrace{\sqrt{\mathbf{x}_{t,a}^\top \mathbf{Z}_{t-1}^{-1} \mathbf{x}_{t,a}}}_{\text{variance}}$$

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

NeuralUCB – update parameter

After receiving reward, NeuralUCB will...

- Update $\mathbf{Z}_t$

$$\mathbf{Z}_t = \mathbf{Z}_{t-1} + \mathbf{g}(\mathbf{x}_{t,a_t}; \boldsymbol{\theta}_{t-1})\mathbf{g}(\mathbf{x}_{t,a_t}; \boldsymbol{\theta}_{t-1})^\top / m$$

NeuralUCB – update parameter

After receiving reward, NeuralUCB will...

- Update $\mathbf{Z}_t$

$$\mathbf{Z}_t = \mathbf{Z}_{t-1} + \mathbf{g}(\mathbf{x}_{t,a_t}; \boldsymbol{\theta}_{t-1})\mathbf{g}(\mathbf{x}_{t,a_t}; \boldsymbol{\theta}_{t-1})^\top / m$$

- Update $\boldsymbol{\theta}_t$ using gradient descent
  - Denote loss function $\mathcal{L}(\boldsymbol{\theta})$ as

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{i=1}^{t} (f(\mathbf{x}_{i,a_i}; \boldsymbol{\theta}) - r_{i,a_i})^2/2 + m\lambda \|\boldsymbol{\theta} - \boldsymbol{\theta}^{(0)}\|_2^2/2$$

NeuralUCB – update parameter

After receiving reward, NeuralUCB will...

- Update $\mathbf{Z}_t$

$$\mathbf{Z}_t = \mathbf{Z}_{t-1} + \mathbf{g}(\mathbf{x}_{t,a_t}; \boldsymbol{\theta}_{t-1})\mathbf{g}(\mathbf{x}_{t,a_t}; \boldsymbol{\theta}_{t-1})^\top / m$$

- Update $\boldsymbol{\theta}_t$ using gradient descent
  - Denote loss function $\mathcal{L}(\boldsymbol{\theta})$ as

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{i=1}^{t} (f(\mathbf{x}_{i,a_i}; \boldsymbol{\theta}) - r_{i,a_i})^2/2 + m\lambda \|\boldsymbol{\theta} - \boldsymbol{\theta}^{(0)}\|_2^2/2$$

  - Run J step gradient descent on $\mathcal{L}(\boldsymbol{\theta})$ starting from $\boldsymbol{\theta}_0$, take $\boldsymbol{\theta}_t$ as the last iterate

$$\boldsymbol{\theta}^{(0)} = \boldsymbol{\theta}_0,\ \boldsymbol{\theta}^{(j+1)} = \boldsymbol{\theta}^{(j)} - \eta \nabla \mathcal{L}(\boldsymbol{\theta}^{(j)}),\ \boldsymbol{\theta}_t = \boldsymbol{\theta}^{(J)}$$

NeuralUCB – confidence radius

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

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

$$\gamma_t = O\left(\underbrace{\sqrt{\lambda} S + \nu \sqrt{\log \frac{\det \mathbf{Z}_t}{\delta \det \lambda \mathbf{I}}}}_{\text{confidence radius}} + \underbrace{(\lambda + tL)(1 - \eta m \lambda)^{J/2} \sqrt{t/\lambda}}_{\text{function approximation error}} \right)$$

NeuralUCB – confidence radius

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

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

$$\gamma_t = O\Bigg(\underbrace{\sqrt{\lambda}S + \nu\sqrt{\log\frac{\det \mathbf{Z}_t}{\delta \det \lambda \mathbf{I}}}}_{\text{confidence radius}} + \underbrace{(\lambda + tL)(1 - \eta m \lambda)^{J/2}\sqrt{t/\lambda}}_{\text{function approximation error}}\Bigg)$$

> Compared with LinUCB,
>
> $$\gamma_t = O\Bigg(\sqrt{\lambda}S + \nu\sqrt{\log\frac{\det \mathbf{Z}_t}{\delta \det \lambda \mathbf{I}}}\Bigg)$$

no function approximation error part!

Main theory – assumptions

Assumption

There exists $\lambda_0 > 0$ such that $\mathbf{H} \succeq \lambda_0 \mathbf{I}$, where $\mathbf{H}$ is the neural tangent kernel matrix (Jacot et al. 2018; Cao and Gu 2019) on contexts $\{\mathbf{x}^i\}_{i=1}^{TK}$.

Main theory – assumptions

Assumption

There exists $\lambda_0 > 0$ such that $\mathbf{H} \succeq \lambda_0 \mathbf{I}$, where $\mathbf{H}$ is the neural tangent kernel matrix (Jacot et al. 2018; Cao and Gu 2019) on contexts $\{\mathbf{x}^i\}_{i=1}^{TK}$.

- Satisfied if *no* two contexts in $\{\mathbf{x}^i\}_{i=1}^{TK}$ are parallel.

Main theory – assumptions

Assumption

There exists $\lambda_0 > 0$ such that $\mathbf{H} \succeq \lambda_0 \mathbf{I}$, where $\mathbf{H}$ is the neural tangent kernel matrix (Jacot et al. 2018; Cao and Gu 2019) on contexts $\{\mathbf{x}^i\}_{i=1}^{TK}$.

- Satisfied if *no* two contexts in $\{\mathbf{x}^i\}_{i=1}^{TK}$ are parallel.

Definition

The effective dimension $\widetilde{d}$ of the neural tangent kernel matrix on contexts $\{\mathbf{x}^i\}_{i=1}^{TK}$ is defined as $\widetilde{d} = \log\det(\mathbf{I} + \mathbf{H}/\lambda)/\log(1 + TK/\lambda)$.

Main theory – assumptions

Assumption

There exists $\lambda_0 > 0$ such that $\mathbf{H} \succeq \lambda_0 \mathbf{I}$, where $\mathbf{H}$ is the neural tangent kernel matrix (Jacot et al. 2018; Cao and Gu 2019) on contexts $\{\mathbf{x}^i\}_{i=1}^{TK}$.

- Satisfied if *no* two contexts in $\{\mathbf{x}^i\}_{i=1}^{TK}$ are parallel.

Definition

The effective dimension $\widetilde{d}$ of the neural tangent kernel matrix on contexts $\{\mathbf{x}^i\}_{i=1}^{TK}$ is defined as $\widetilde{d} = \log\det(\mathbf{I} + \mathbf{H}/\lambda)/\log(1 + TK/\lambda)$.

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

Main theory – assumptions

Assumption

There exists $\lambda_0 > 0$ such that $\mathbf{H} \succeq \lambda_0 \mathbf{I}$, where $\mathbf{H}$ is the neural tangent kernel matrix (Jacot et al. 2018; Cao and Gu 2019) on contexts $\{\mathbf{x}^i\}_{i=1}^{TK}$.

- Satisfied if *no* two contexts in $\{\mathbf{x}^i\}_{i=1}^{TK}$ are parallel.

Definition

The effective dimension $\widetilde{d}$ of the neural tangent kernel matrix on contexts $\{\mathbf{x}^i\}_{i=1}^{TK}$ is defined as $\widetilde{d} = \log\det(\mathbf{I} + \mathbf{H}/\lambda)/\log(1 + TK/\lambda)$.

- Notion adapted from Valko et al. (2013) and Yang and Wang (2019)
- $\widetilde{d} \sim \log T$ in several special cases (Valko et al. 2013)

Main theory – regret bound

Theorem

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

$$R_T = \widetilde{O}\Big(\sqrt{\widetilde{d}T}\sqrt{\max\{\widetilde{d}, S^2\}}\Big).$$

Main theory – regret bound

Theorem

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

$$R_T = \widetilde{O}\Big(\sqrt{\widetilde{d}T}\sqrt{\max\{\widetilde{d}, S^2\}}\Big).$$

- h belongs to the RKHS space $\mathcal{H}$ spanned by $\mathbf{H} \Rightarrow S \leq \|h\|_{\mathcal{H}}$

Main theory – regret bound

Theorem

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

$$R_T = \widetilde{O}\Big(\sqrt{\widetilde{d}T}\sqrt{\max\{\widetilde{d}, S^2\}}\Big).$$

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

Main theory – regret bound

Theorem

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

$$R_T = \widetilde{O}\Big(\sqrt{\widetilde{d}T}\sqrt{\max\{\widetilde{d}, S^2\}}\Big).$$

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

Takeaway message

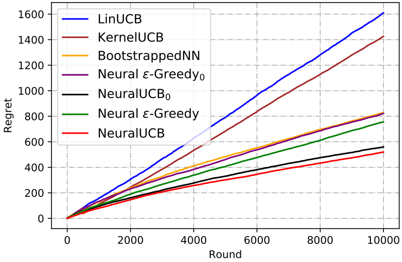
- NeuralUCB uses neural network $f(\mathbf{x}; \boldsymbol{\theta}_t)$ to predict, gradient $\mathbf{g}(\mathbf{x}; \boldsymbol{\theta}_t)$ to explore

Takeaway message

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

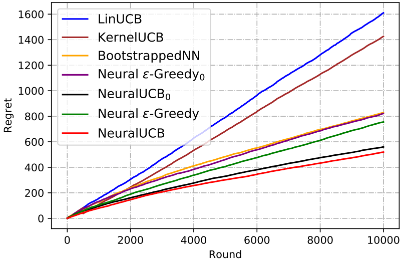
Takeaway message

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



Takeaway message

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



Thank you!