NQ-Net: Deep Non-crossing Quantile Learning

Hongtu Zhu[1]

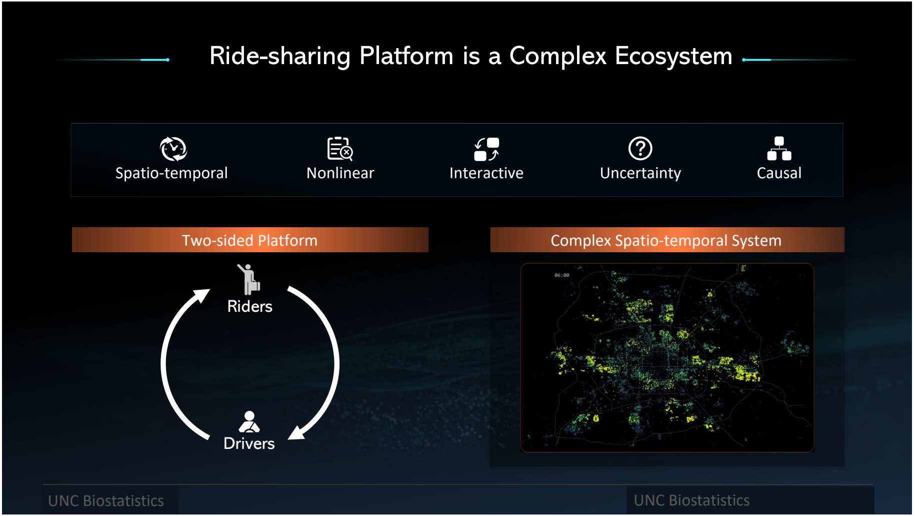
[1]The University of North Carolina at Chapel Hill

My collaborators: Guohao Shen (PolyU), Shikai Luo (Bytedance), and Chengchun Shi (LSE)

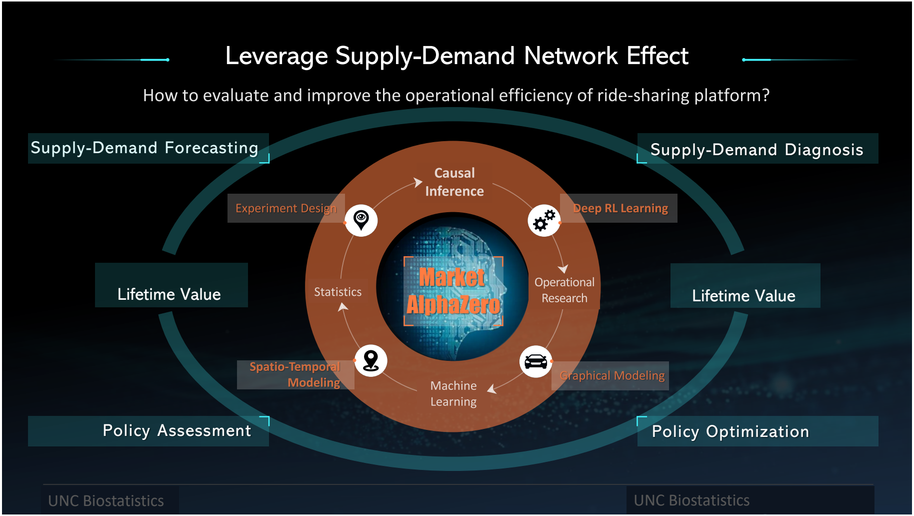
Table of Contents

Table of Contents

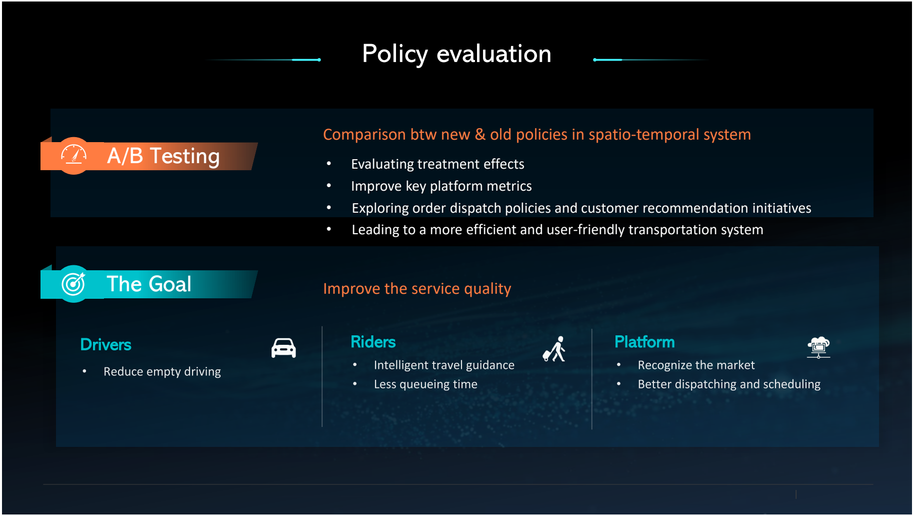
Ride-sharing Platform



Market Alphazero in Two-sided Marketplace



Experimental Design in Two-sided Marketplace



Trustworthy Machine Learning & Quantile Regression

Enhancing Robustness

- Models variability beyond the mean for a fuller data picture.
- Improves reliability against outliers and skewed distributions.

Improving Interpretability

- Reveals variable relationships across the distribution.
- Enhances model transparency and trust with detailed insights.

Promoting Fairness

- Mitigates disparities across subgroups at different quantiles.
- Identifies and corrects biases for equitable outcomes.

Quantifying Uncertainty

- Facilitates prediction interval estimation, measuring uncertainty.
- Supports informed decision-making with accountable models.

An introduction example

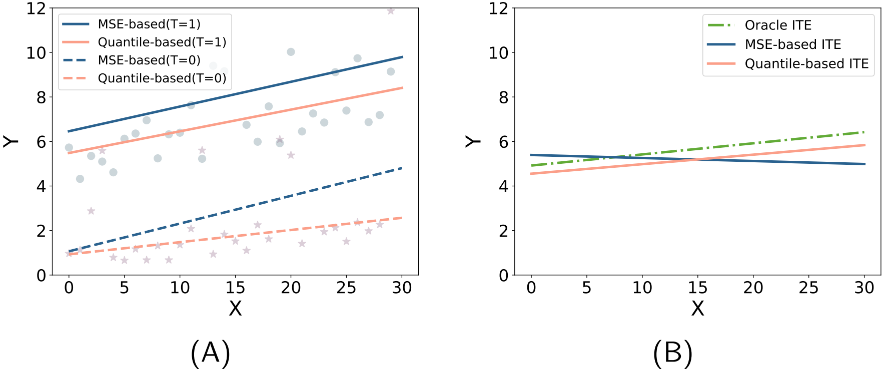


Figure: A toy simulation example to visualize the disadvantage of the conditional average treatment effect (CATE) with heavy-tailed outcomes. Panel A plots the data distribution for treatments 0 and 1 with circles and stars. The blue and orange lines are the conditional mean and median estimators. Panel B displays the corresponding CATE. The green dashed line depicts the Median treatment effect values.

Table of Contents

Problem formulation

- Let $(X, Y) \sim P_{X,Y}$, QR concerns the τth conditional quantile

$$Q_Y^\tau(x) = F_{Y|X=x}^{-1}(\tau), \qquad \text{for } \tau \in (0,1).$$

Problem formulation

- Let $(X, Y) \sim P_{X,Y}$, QR concerns the τth conditional quantile

$$Q_Y^\tau(x) = F^{-1}_{Y|X=x}(\tau), \qquad \text{for } \tau \in (0,1).$$

- Given $\tau \in (0,1)$, the $Q_Y^\tau(x)$ can be consistently estimated by

$$\arg\min_{f \in \mathcal{F}} \mathbb{E}_{X,Y}[\rho_\tau(Y - f(X))],$$

where $\rho_\tau(a) = a[\tau - 1(a < 0)]$ is the check loss and $\mathcal{F}$ is a class of neural networks.

Problem formulation

- Let $(X, Y) \sim P_{X,Y}$, QR concerns the τth conditional quantile

$$Q_Y^\tau(x) = F_{Y|X=x}^{-1}(\tau), \qquad \text{for } \tau \in (0,1).$$

- Given $\tau \in (0,1)$, the $Q_Y^\tau(x)$ can be consistently estimated by

$$\arg\min_{f \in \mathcal{F}} \mathbb{E}_{X,Y}[\rho_\tau(Y - f(X))],$$

where $\rho_\tau(a) = a[\tau - 1(a < 0)]$ is the check loss and $\mathcal{F}$ is a class of neural networks.

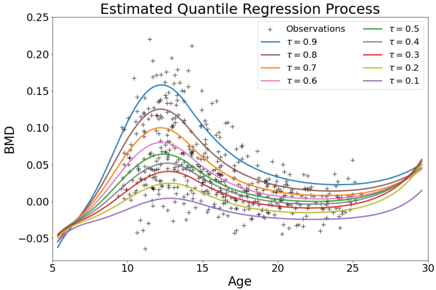
- Objective of distributional learning: $Q_Y^{\tau_1}(x), \dots, Q_Y^{\tau_K}(x)$ at K levels:

$$\arg\min_{f \in \mathcal{F}} L(f) = \arg\min_{f \in \mathcal{F}} \sum_{k=1}^{K} \frac{1}{K} \mathbb{E}_{X,Y}[\rho_{\tau_k}(Y - f_k(X))]. \qquad (1)$$

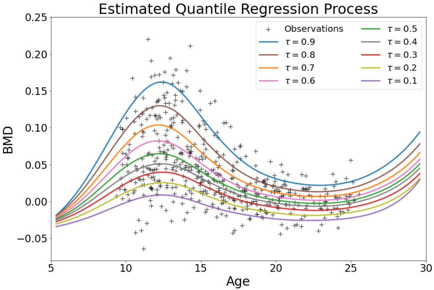
Crossing-quantile Problems

- The learned quantile curves $\hat{f}_1(x), \ldots, \hat{f}_K(x)$ have crossing-quantile problems even when x is one-dimensional.
- $\hat{f}_1(x) \leq \hat{f}_2(x) \leq \cdots \leq \hat{f}_K(x)$ does not hold.

Quantile Crossing



Quantile estimations with CROSSING.

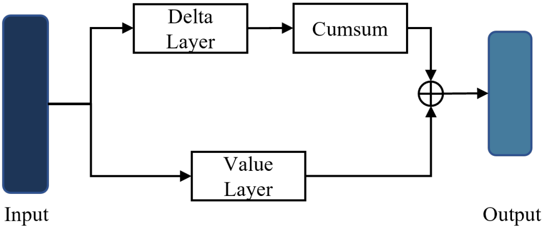


Quantile estimations with NO CROSSING.

Figure: An example of quantile crossing problem in bone mineral density (BMD) data set. Estimated quantile curves at $\tau = 0.1, 0.2, \ldots, 0.9$ and the observations are depicted.

Non-Crossing Quantile Layer

Non-crossing Quantile Network with **Delta Layer** and **Value Layer**.



Non-Crossing Quantile Network

Figure: The delta layer $d(\cdot;\theta_\delta)$ produce non-crossing zero-mean quantile vector. And the value layer $v(\cdot;\theta_v)$ predicts the mean of quantiles. Adding them together would finally produce the quantile predictions $NQ(x) = v(x;\theta_v) \oplus d(x;\theta_\delta)$.

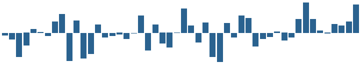
Non-Crossing Quantile Estimation

We use the right figure to show how to formulate non-crossing estimation of quantiles.

Non-Crossing Quantile Estimation

We use the right figure to show how to formulate non-crossing estimation of quantiles.

- Output of a base deep neural network.

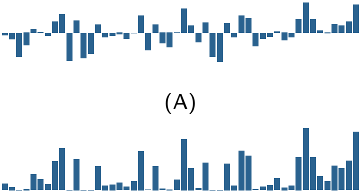


(A)

Non-Crossing Quantile Estimation

We use the right figure to show how to formulate non-crossing estimation of quantiles.

- Output of a base deep neural network.
- Apply the activation function $\sigma(x) = ELU(x) + 1$ to create non-negative outputs.

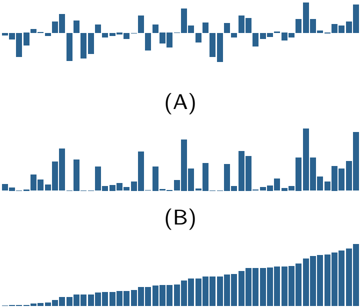


(A) (B)

Non-Crossing Quantile Estimation

We use the right figure to show how to formulate non-crossing estimation of quantiles.

- Output of a base deep neural network.
- Apply the activation function $\sigma(x) = ELU(x) + 1$ to create non-negative outputs.
- Apply the cumsum function to generate non-crossing quantiles.

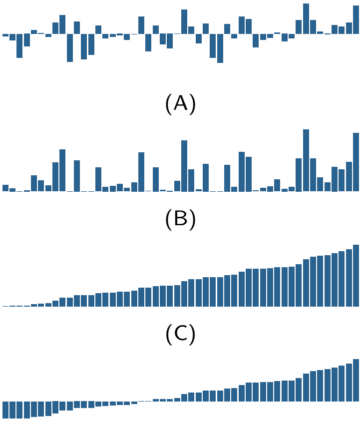


(A) (B) (C)

Non-Crossing Quantile Estimation

We use the right figure to show how to formulate non-crossing estimation of quantiles.

- Output of a base deep neural network.
- Apply the activation function $\sigma(x) = ELU(x) + 1$ to create non-negative outputs.
- Apply the cumsum function to generate non-crossing quantiles.
- Center the outputs.



(A)

(B)

(C)

(D)

NQ neural networks

- NQ net $f(x) = v(x) \oplus (ELU + 1)(d(x)) \in \mathbb{R}^K$ with $\mathcal{D}$ hidden layers

$$\begin{pmatrix} v(x) \\ d(x) \end{pmatrix} = \mathcal{L}_{\mathcal{D}} \circ \sigma \circ \mathcal{L}_{\mathcal{D}-1} \circ \sigma \circ \cdots \circ \sigma \circ \mathcal{L}_1 \circ \sigma \circ \mathcal{L}_0(x), x \in \mathbb{R}^{d_0}.$$

NQ neural networks

- NQ net $f(x) = v(x) \oplus (ELU + 1)(d(x)) \in \mathbb{R}^K$ with $\mathcal{D}$ hidden layers

$$\begin{pmatrix} v(x) \\ d(x) \end{pmatrix} = \mathcal{L}_{\mathcal{D}} \circ \sigma \circ \mathcal{L}_{\mathcal{D}-1} \circ \sigma \circ \cdots \circ \sigma \circ \mathcal{L}_1 \circ \sigma \circ \mathcal{L}_0(x), x \in \mathbb{R}^{d_0}.$$

  1. $\mathcal{L}_i(x) = W_i x + b_i$ is the i-th linear transformation with $x \in \mathbb{R}^{p_i}$ where $W_i \in \mathbb{R}^{p_{i+1} \times p_i}$ is the weight matrix and $b_i \in \mathbb{R}^{p_{i+1}}$ is the bias vector.

NQ neural networks

- NQ net $f(x) = v(x) \oplus (ELU + 1)(d(x)) \in \mathbb{R}^K$ with $\mathcal{D}$ hidden layers

$$\begin{pmatrix} v(x) \\ d(x) \end{pmatrix} = \mathcal{L}_{\mathcal{D}} \circ \sigma \circ \mathcal{L}_{\mathcal{D}-1} \circ \sigma \circ \cdots \circ \sigma \circ \mathcal{L}_1 \circ \sigma \circ \mathcal{L}_0(x), x \in \mathbb{R}^{d_0}.$$

1. $\mathcal{L}_i(x) = W_i x + b_i$ is the i-th linear transformation with $x \in \mathbb{R}^{p_i}$ where $W_i \in \mathbb{R}^{p_{i+1} \times p_i}$ is the weight matrix and $b_i \in \mathbb{R}^{p_{i+1}}$ is the bias vector.
2. $\sigma = \max\{x, 0\}$ is the rectified linear unit (ReLU) activation function

NQ neural networks

- NQ net $f(x) = v(x) \oplus (ELU + 1)(d(x)) \in \mathbb{R}^K$ with $\mathcal{D}$ hidden layers

$$\begin{pmatrix} v(x) \\ d(x) \end{pmatrix} = \mathcal{L}_{\mathcal{D}} \circ \sigma \circ \mathcal{L}_{\mathcal{D}-1} \circ \sigma \circ \cdots \circ \sigma \circ \mathcal{L}_1 \circ \sigma \circ \mathcal{L}_0(x), x \in \mathbb{R}^{d_0}.$$

1. $\mathcal{L}_i(x) = W_i x + b_i$ is the i-th linear transformation with $x \in \mathbb{R}^{p_i}$ where $W_i \in \mathbb{R}^{p_{i+1} \times p_i}$ is the weight matrix and $b_i \in \mathbb{R}^{p_{i+1}}$ is the bias vector.
2. $\sigma = \max\{x, 0\}$ is the rectified linear unit (ReLU) activation function

- Class of NQ networks $\mathcal{F} = \{f \text{ over all possible choice of } \{(W_i, b_i)\}_{i=0}^{\mathcal{D}}, \text{and } \|f\|_\infty \leq \mathcal{B}, \|\frac{\partial}{\partial \tau} f\|_\infty \leq \mathcal{B}'\}.$

NQ neural networks

- NQ net $f(x) = v(x) \oplus (ELU + 1)(d(x)) \in \mathbb{R}^K$ with $\mathcal{D}$ hidden layers

$$\begin{pmatrix} v(x) \\ d(x) \end{pmatrix} = \mathcal{L}_{\mathcal{D}} \circ \sigma \circ \mathcal{L}_{\mathcal{D}-1} \circ \sigma \circ \cdots \circ \sigma \circ \mathcal{L}_1 \circ \sigma \circ \mathcal{L}_0(x), x \in \mathbb{R}^{d_0}.$$

  1. $\mathcal{L}_i(x) = W_i x + b_i$ is the i-th linear transformation with $x \in \mathbb{R}^{p_i}$ where $W_i \in \mathbb{R}^{p_{i+1} \times p_i}$ is the weight matrix and $b_i \in \mathbb{R}^{p_{i+1}}$ is the bias vector.
  2. $\sigma = \max\{x, 0\}$ is the rectified linear unit (ReLU) activation function

- Class of NQ networks $\mathcal{F} = \{f \text{ over all possible choice of } \{(\mathrm{W_i}, \mathrm{b_i})\}_{\mathrm{i}=0}^{\mathcal{D}}, \text{and } \|\mathrm{f}\|_\infty \leq \mathcal{B}, \|\frac{\partial}{\partial \tau}\mathrm{f}\|_\infty \leq \mathcal{B}'\}.$

  1. Depth $\mathcal{D}$, width $\mathcal{W} = \max\{p_1, ..., p_{\mathcal{D}}\}$

NQ neural networks

- NQ net $f(x) = v(x) \oplus (ELU + 1)(d(x)) \in \mathbb{R}^K$ with $\mathcal{D}$ hidden layers

$$\begin{pmatrix} v(x) \\ d(x) \end{pmatrix} = \mathcal{L}_{\mathcal{D}} \circ \sigma \circ \mathcal{L}_{\mathcal{D}-1} \circ \sigma \circ \cdots \circ \sigma \circ \mathcal{L}_1 \circ \sigma \circ \mathcal{L}_0(x), x \in \mathbb{R}^{d_0}.$$

  1. $\mathcal{L}_i(x) = W_i x + b_i$ is the i-th linear transformation with $x \in \mathbb{R}^{p_i}$ where $W_i \in \mathbb{R}^{p_{i+1} \times p_i}$ is the weight matrix and $b_i \in \mathbb{R}^{p_{i+1}}$ is the bias vector.
  2. $\sigma = \max\{x, 0\}$ is the rectified linear unit (ReLU) activation function

- Class of NQ networks $\mathcal{F} = \{f \text{ over all possible choice of } \{(\mathrm{W_i}, \mathrm{b_i})\}_{\mathrm{i}=0}^{\mathcal{D}}, \text{and } \|\mathrm{f}\|_\infty \leq \mathcal{B}, \|\frac{\partial}{\partial \tau}\mathrm{f}\|_\infty \leq \mathcal{B}'\}.$
  1. Depth $\mathcal{D}$, width $\mathcal{W} = \max\{p_1, ..., p_{\mathcal{D}}\}$
  2. Size $\mathcal{S} = \sum_{i=0}^{\mathcal{D}} \{p_{i+1} \times (p_i + 1)\}$

NQ neural networks

- NQ net $f(x) = v(x) \oplus (ELU + 1)(d(x)) \in \mathbb{R}^K$ with $\mathcal{D}$ hidden layers

$$\begin{pmatrix} v(x) \\ d(x) \end{pmatrix} = \mathcal{L}_{\mathcal{D}} \circ \sigma \circ \mathcal{L}_{\mathcal{D}-1} \circ \sigma \circ \cdots \circ \sigma \circ \mathcal{L}_1 \circ \sigma \circ \mathcal{L}_0(x), x \in \mathbb{R}^{d_0}.$$

  1. $\mathcal{L}_i(x) = W_i x + b_i$ is the i-th linear transformation with $x \in \mathbb{R}^{p_i}$ where $W_i \in \mathbb{R}^{p_{i+1} \times p_i}$ is the weight matrix and $b_i \in \mathbb{R}^{p_{i+1}}$ is the bias vector.
  2. $\sigma = \max\{x, 0\}$ is the rectified linear unit (ReLU) activation function

- Class of NQ networks $\mathcal{F} = \{f \text{ over all possible choice of } \{(\mathrm{W_i}, \mathrm{b_i})\}_{\mathrm{i}=0}^{\mathcal{D}}, \text{and } \|\mathrm{f}\|_\infty \le \mathcal{B}, \|\frac{\partial}{\partial \tau}\mathrm{f}\|_\infty \le \mathcal{B}'\}.$
  1. Depth $\mathcal{D}$, width $\mathcal{W} = \max\{p_1, ..., p_{\mathcal{D}}\}$
  2. Size $\mathcal{S} = \sum_{i=0}^{\mathcal{D}} \{p_{i+1} \times (p_i + 1)\}$
  3. Number of neurons $\mathcal{U} = \sum_{i=1}^{\mathcal{D}} p_i$

Learning Guarantee

Theorem (Non-asymptotic upper bounds)

Suppose the ground truth Q^Y are β-Hölder smooth. For any integers $U, M \in \mathbb{N}^+$, let the class of networks $\mathcal{F}$ uniformly bounded by $\mathcal{B}$, has width $\mathcal{W} = 38(K+1)(\lfloor\beta\rfloor+1)^2 d_0^{\lfloor\beta\rfloor+1} U \log_2(8U)$ and depth $\mathcal{D} = 21(\lfloor\beta\rfloor+1)^2 d_0^{\lfloor\beta\rfloor+1} M \log_2(8M)$. Then for any $\delta > 0$, with prob. at least $1-\delta$

$$\mathcal{R}(\hat{f}_N) := \mathcal{L}(\hat{f}_N) - \mathcal{L}(Q_Y) \leq \frac{2\sqrt{2}(K+2)\mathcal{B}}{\sqrt{N}}\left(C\sqrt{K\mathcal{S}\mathcal{D}\log(\mathcal{S})\log(N)} + \sqrt{\log(1/\delta)}\right)$$
$$+ 18(K+2)\mathcal{B}(\lfloor\beta\rfloor+1)^2 d_0^{\lfloor\beta\rfloor+(\beta\vee 1)/2}(UM)^{-2\beta/d_0} + (K+2)\exp(-\mathcal{B})$$

for $N \geq c \cdot \mathcal{D}\mathcal{S}\log(\mathcal{S})$ where $C, c > 0$ are universal constants, and d_0 is the input dimension of the target quantile functions Q_Y and also neural networks in $\mathcal{F}$.

Learning Guarantee

Theorem (Non-asymptotic upper bounds)

Suppose the ground truth Q^Y are β-Hölder smooth. For any integers $U, M \in \mathbb{N}^+$, let the class of networks $\mathcal{F}$ uniformly bounded by $\mathcal{B}$, has width $\mathcal{W} = 38(K+1)(\lfloor\beta\rfloor+1)^2 d_0^{\lfloor\beta\rfloor+1} U \log_2(8U)$ and depth $\mathcal{D} = 21(\lfloor\beta\rfloor+1)^2 d_0^{\lfloor\beta\rfloor+1} M \log_2(8M)$. Then for any $\delta > 0$, with prob. at least $1-\delta$

$$\mathcal{R}(\hat{f}_N) := \mathcal{L}(\hat{f}_N) - \mathcal{L}(Q_Y) \leq \frac{2\sqrt{2}(K+2)\mathcal{B}}{\sqrt{N}}\left(C\sqrt{K\mathcal{S}\mathcal{D}\log(\mathcal{S})\log(N)} + \sqrt{\log(1/\delta)}\right)$$
$$+ 18(K+2)\mathcal{B}(\lfloor\beta\rfloor+1)^2 d_0^{\lfloor\beta\rfloor+(\beta\vee 1)/2}(UM)^{-2\beta/d_0} + (K+2)\exp(-\mathcal{B})$$

for $N \geq c \cdot \mathcal{D}\mathcal{S}\log(\mathcal{S})$ where $C, c > 0$ are universal constants, and d_0 is the input dimension of the target quantile functions Q_Y and also neural networks in $\mathcal{F}$.

- Stochastic error(variance) increasing in network size, decreasing in sample size N.

Learning Guarantee

Theorem (Non-asymptotic upper bounds)

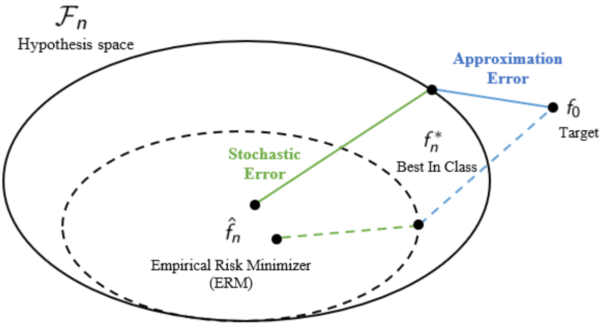
Suppose the ground truth Q^Y *are* β*-Hölder smooth. For any integers* $U, M \in \mathbb{N}^+$*, let the class of networks* $\mathcal{F}$ *uniformly bounded by* $\mathcal{B}$*, has width* $\mathcal{W} = 38(K+1)(\lfloor \beta \rfloor + 1)^2 d_0^{\lfloor \beta \rfloor + 1} U \log_2(8U)$ *and depth* $\mathcal{D} = 21(\lfloor \beta \rfloor + 1)^2 d_0^{\lfloor \beta \rfloor + 1} M \log_2(8M)$*. Then for any* $\delta > 0$*, with prob. at least* $1 - \delta$

$$\mathcal{R}(\hat{f}_N) := \mathcal{L}(\hat{f}_N) - \mathcal{L}(Q_Y) \leq \frac{2\sqrt{2}(K+2)\mathcal{B}}{\sqrt{N}} \left(C\sqrt{K\mathcal{S}\mathcal{D}\log(\mathcal{S})\log(N)} + \sqrt{\log(1/\delta)} \right)$$
$$+ 18(K+2)\mathcal{B}(\lfloor \beta \rfloor + 1)^2 d_0^{\lfloor \beta \rfloor + (\beta \vee 1)/2} (UM)^{-2\beta/d_0} + (K+2)\exp(-\mathcal{B})$$

for $N \geq c \cdot \mathcal{D}\mathcal{S}\log(\mathcal{S})$ *where* $C, c > 0$ *are universal constants, and* d_0 *is the input dimension of the target quantile functions* Q_Y *and also neural networks in* $\mathcal{F}$*.*

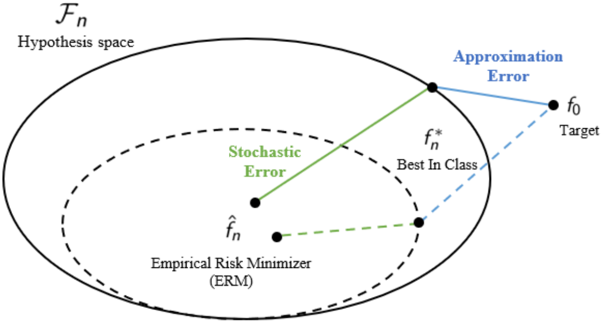
- Stochastic error(variance) increasing in network size, decreasing in sample size N.
- Approximation error(bias) decreasing in network size, smoothness β of target Q^Y.

Bias and Variance Trade-off



- Stochastic error(variance) increasing in network size, decreasing in sample size N.

Bias and Variance Trade-off



- Stochastic error(variance) increasing in network size, decreasing in sample size N.
- Approximation error(bias) decreasing in network size, smoothness β of target Q^Y.

Learning Guarantee

Theorem (Non-asymptotic upper bounds)

Suppose the ground truth Q^Y are β-Hölder smooth. For any integers $U, M \in \mathbb{N}^+$, let the class of networks $\mathcal{F}$ uniformly bounded by $\mathcal{B}$, has width $\mathcal{W} = 38(K+1)(\lfloor\beta\rfloor+1)^2 d_0^{\lfloor\beta\rfloor+1} U \log_2(8U)$ and depth $\mathcal{D} = 21(\lfloor\beta\rfloor+1)^2 d_0^{\lfloor\beta\rfloor+1} M \log_2(8M)$. Then for any $\delta > 0$, with prob. at least $1-\delta$

$$\mathcal{R}(\hat{f}_N) := \mathcal{L}(\hat{f}_N) - \mathcal{L}(Q_Y) \leq \frac{2\sqrt{2}(K+2)\mathcal{B}}{\sqrt{N}}\left(C\sqrt{K\mathcal{S}\mathcal{D}\log(\mathcal{S})\log(N)} + \sqrt{\log(1/\delta)}\right)$$
$$+ 18(K+2)\mathcal{B}(\lfloor\beta\rfloor+1)^2 d_0^{\lfloor\beta\rfloor+(\beta\vee 1)/2}(UM)^{-2\beta/d_0} + (K+2)\exp(-\mathcal{B})$$

for $N \geq c \cdot \mathcal{D}\mathcal{S}\log(\mathcal{S})$ where $C, c > 0$ are universal constants, and d_0 is the input dimension of the target quantile functions Q_Y and also neural networks in $\mathcal{F}$.

- Stochastic error(variance) increasing in network size, decreasing in samplesize N.
- Approximation error(bias) decreasing in network size, smoothness β of target Q^Y.
- Let $U = 1$, $M = N^{d_0/[2(d_0+2\beta)]}$ and $\mathcal{B} = \log(N)$, then $\mathcal{R}(\hat{f}_N) = O_p((\log N)^4 N^{-\beta/(2\beta+d_0)})$.

Learning Guarantee

Theorem (Non-asymptotic upper bounds)

Suppose the ground truth Q^Y *are* β*-Hölder smooth. For any integers* $U, M \in \mathbb{N}^+$*, let the class of networks* $\mathcal{F}$ *uniformly bounded by* $\mathcal{B}$*, has width* $\mathcal{W} = 38(K+1)(\lfloor\beta\rfloor+1)^2 d_0^{\lfloor\beta\rfloor+1} U \log_2(8U)$ *and depth* $\mathcal{D} = 21(\lfloor\beta\rfloor+1)^2 d_0^{\lfloor\beta\rfloor+1} M \log_2(8M)$*. Then for any* $\delta > 0$*, with prob. at least* $1-\delta$

$$\mathcal{R}(\hat{f}_N) := \mathcal{L}(\hat{f}_N) - \mathcal{L}(Q_Y) \leq \frac{2\sqrt{2}(K+2)\mathcal{B}}{\sqrt{N}}\left(C\sqrt{K\mathcal{S}\mathcal{D}\log(\mathcal{S})\log(N)} + \sqrt{\log(1/\delta)}\right)$$
$$+ 18(K+2)\mathcal{B}(\lfloor\beta\rfloor+1)^2 d_0^{\lfloor\beta\rfloor+(\beta\vee 1)/2}(UM)^{-2\beta/d_0} + (K+2)\exp(-\mathcal{B})$$

for $N \geq c \cdot \mathcal{D}\mathcal{S}\log(\mathcal{S})$ *where* $C, c > 0$ *are universal constants, and* d_0 *is the input dimension of the target quantile functions* Q_Y *and also neural networks in* $\mathcal{F}$*.*

- Stochastic error(variance) increasing in network size, decreasing in samplesize N.
- Approximation error(bias) decreasing in network size, smoothness β of target Q^Y.
- Let $U = 1$, $M = N^{d_0/[2(d_0+2\beta)]}$ and $\mathcal{B} = \log(N)$, then $\mathcal{R}(\hat{f}_N) = O_p((\log N)^4 N^{-\beta/(2\beta+d_0)})$.
- ptsize Self-calibration: $\sum_{k=1}^K \mathbb{E}|f_{\tau_k}(X) - Q_Y^{\tau_k}(X)|^2 \leq c \cdot \mathcal{R}(f)$. under proper condition.

Learning Guarantee with low-dim data

Assumption

The predictor X is supported on $\mathcal{M}_\rho$, a ρ-neighborhood of $\mathcal{M} \subset [0,1]^{d_0}$, where $\mathcal{M}$ is a compact $d_{\mathcal{M}}$-dimensional Riemannian sub-manifold and

$$\mathcal{M}_\rho = \{x \in [0,1]^{d_0} : \inf\{\|x - y\|_2 : y \in \mathcal{M}\} \leq \rho\}, \ \rho \in (0,1).$$

Learning Guarantee with low-dim data

Assumption

The predictor X is supported on $\mathcal{M}_\rho$, a ρ-neighborhood of $\mathcal{M} \subset [0,1]^{d_0}$, where $\mathcal{M}$ is a compact $d_{\mathcal{M}}$-dimensional Riemannian sub-manifold and

$$\mathcal{M}_\rho = \{x \in [0,1]^{d_0} : \inf\{\|x-y\|_2 : y \in \mathcal{M}\} \le \rho\}, \ \rho \in (0,1).$$

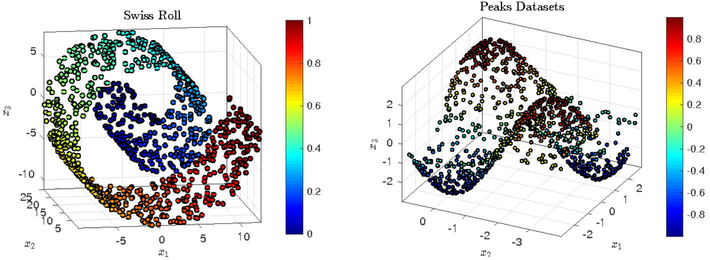


Figure: An example of data with low-dimensional support.

Learning Guarantee with low-dim data

Theorem (Non-asymptotic upper bounds with low-dim data)

Suppose the ground truth Q^Y *are* β*-Hölder smooth. For any integers* $U, M \in \mathbb{N}^+$*, let class of networks* $\mathcal{F}$ *uniformly bounded by* $\mathcal{B}$*, has width* $\mathcal{W} = 38(K+1)(\lfloor\beta\rfloor+1)^2(d_0^*)^{\lfloor\beta\rfloor+1} U \log_2(8U)$ *and depth* $\mathcal{D} = 21(\lfloor\beta\rfloor+1)^2(d_0^*)^{\lfloor\beta\rfloor+1} M \log_2(8M)$*. Then for any* $\delta > 0$*, with prob. at least* $1-\delta$

$$\mathcal{R}(\hat{f}_N) := \mathcal{L}(\hat{f}_N) - \mathcal{L}(Q_Y) \le \frac{2\sqrt{2}(K+2)\mathcal{B}}{\sqrt{N}}\left(C\sqrt{K\mathcal{S}\mathcal{D}\log(\mathcal{S})\log(N)} + \sqrt{\log(1/\delta)}\right)$$
$$+ 18(K+2)\mathcal{B}(\lfloor\beta\rfloor+1)^2(d_0^*)^{\lfloor\beta\rfloor+(\beta\vee 1)/2}(UM)^{-2\beta/(d_0^*)} + (K+2)\exp(-\mathcal{B})$$

for $d_0^* = O(d_{\mathcal{M}}\log(d_0/\delta)/\delta^2)$ *is an integer satisfying* $d_{\mathcal{M}} \le d_0^* < d_0$ *for any given* $\delta \in (0,1)$ *and* $\rho \le C_2(UM)^{-2\beta/d_0^*}(\beta+1)^2 d_0^{1/2}(d_0^*)^{3\beta/2}(\sqrt{d_0/d_0^*}+1-\delta)^{-1}(1-\delta)^{1-\beta}$.

- d_0^* is effective instead of d_0 where $d_0^* \le d_0$.

Learning Guarantee with low-dim data

Theorem (Non-asymptotic upper bounds with low-dim data)

Suppose the ground truth Q^Y *are* β*-Hölder smooth. For any integers* $U, M \in \mathbb{N}^+$*, let class of networks* $\mathcal{F}$ *uniformly bounded by* $\mathcal{B}$*, has width* $\mathcal{W} = 38(K+1)(\lfloor\beta\rfloor+1)^2(d_0^*)^{\lfloor\beta\rfloor+1}U\log_2(8U)$ *and depth* $\mathcal{D} = 21(\lfloor\beta\rfloor+1)^2(d_0^*)^{\lfloor\beta\rfloor+1}M\log_2(8M)$*. Then for any* $\delta > 0$*, with prob. at least* $1-\delta$

$$\mathcal{R}(\hat{f}_N) := \mathcal{L}(\hat{f}_N) - \mathcal{L}(Q_Y) \leq \frac{2\sqrt{2}(K+2)\mathcal{B}}{\sqrt{N}}\left(C\sqrt{K\mathcal{S}\mathcal{D}\log(\mathcal{S})\log(N)} + \sqrt{\log(1/\delta)}\right) + 18(K+2)\mathcal{B}(\lfloor\beta\rfloor+1)^2(d_0^*)^{\lfloor\beta\rfloor+(\beta\vee 1)/2}(UM)^{-2\beta/(d_0^*)} + (K+2)\exp(-\mathcal{B})$$

for $d_0^* = O(d_{\mathcal{M}}\log(d_0/\delta)/\delta^2)$ *is an integer satisfying* $d_{\mathcal{M}} \leq d_0^* < d_0$ *for any given* $\delta \in (0,1)$ *and* $\rho \leq C_2(UM)^{-2\beta/d_0^*}(\beta+1)^2 d_0^{1/2}(d_0^*)^{3\beta/2}(\sqrt{d_0/d_0^*}+1-\delta)^{-1}(1-\delta)^{1-\beta}$.

- d_0^* is effective instead of d_0 where $d_0^* \leq d_0$.
- Let $U = 1$, $M = N^{d_0^*/[2(d_0^*+2\beta)]}$ and $\mathcal{B} = \log(N)$, then $\mathcal{R}(\hat{f}_N) = O_p((\log N)^4 N^{-\beta/(2\beta+d_0^*)})$.

Table of Contents

Application to Conditional Average Treatment Effect

There are different types of UI design for the same APP. How to personalize the UI for each user based on their preference.

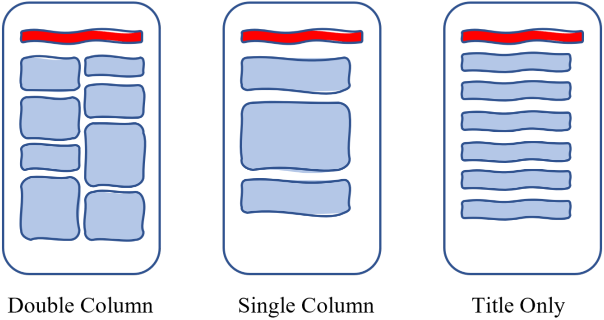


Figure: An example of uplift modeling

Application to Conditional Average Treatment Effect

Problem definition

Given observed features x, we want to estimate conditional average treatment effect (CATE), $\tau_t(x) = E[Y^*(t) - Y^*(0)|X = x]$, under different treatment t, where $Y^*(t)$ is the potential outcome under treatment t.

Application to Conditional Average Treatment Effect

Problem definition

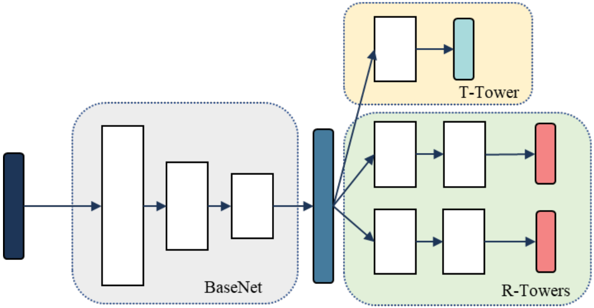
Given observed features x, we want to estimate conditional average treatment effect (CATE), $\tau_t(x) = E[Y^*(t) - Y^*(0)|X = x]$, under different treatment t, where $Y^*(t)$ is the potential outcome under treatment t.

Assumption of CATE estimation

(A1) $Y = Y^*(T)$.
(A2) T is independent of $(Y^*(0), Y^*(1), \ldots, Y^*(M-1))$ given X.
(A3) $\pi_0(t|x) := P(T = t|X = x) > 0$ for $\forall x, t$.

Baselines

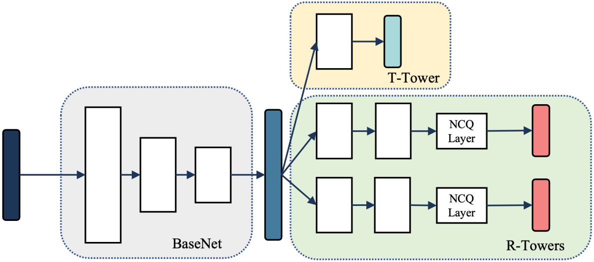
Usually, baselines such as TARNET or DragonNet use a share-bottom architecture to learn response of each treatment with MSE loss function.



Share-bottom Arch

Model illustration: DNet

Based on NQ-network, one can implement a DNet.



DNet with R-Tower being our NQ network

- A *BaseNet* $b(\cdot) = b(\cdot; \theta_b)$ that learns a shared representation for all treatments.
- A *R-Tower* associated with each individual treatment t, represented by $R(\cdot, t; \theta_r)$ with the last layer being our proposed non-crossing quantile network.
- A *T-Tower*, a simple softmax layer that estimates the propensity vector, $\pi(x; \theta_\pi) = \{P(T = t | X = x, \theta_\pi)\}_{t=0}^{M-1}$.

Model training: DNet

- For the *R-Tower*'s, we consider quantile Huber loss or check loss ℓ_{γ_k}:

$$\ell_q(R(b(x), t; \theta_r), y) = \frac{1}{K}\sum_{k=1}^{K} \ell_{\gamma_k}(y - q_{\gamma_k}(b(x), t)),$$

where $q_{\gamma_k}(b(x), t)$ is the kth qauntile output of $R(b(x), t; \theta_r)$ under treatment t.

Model training: DNet

- For the *R-Tower*'s, we consider quantile Huber loss or check loss ℓ_{γ_k}:

$$\ell_q(R(b(x), t; \theta_r), y) = \frac{1}{K}\sum_{k=1}^{K} \ell_{\gamma_k}(y - q_{\gamma_k}(b(x), t)),$$

where $q_{\gamma_k}(b(x), t)$ is the kth qauntile output of $R(b(x), t; \theta_r)$ under treatment t.

- For the *T-Tower*'s, we consider the cross entropy loss

$$\ell_{ce}(\pi(b(x); \theta_\pi), t) = \frac{1}{M}\sum_{k=0}^{M-1} t^{(k)} \log(\pi(b(x), ; \theta_\pi)^{(k)}), \quad (2)$$

where $\mathbf{t} = (t^{(0)}, t^{(1)}, \ldots, t^{(M-1)})^T$ is the one-hot vector of treatment, and $\pi(b(x); \theta_\pi) = (\pi(b(x); \theta_\pi)^{(0)}, \pi(b(x); \theta_\pi)^{(1)}, \ldots, \pi(b(x); \theta_\pi)^{(M-1)})^T$ is the predicted score.

Model training: DNet

- For the *R-Tower*'s, we consider quantile Huber loss or check loss ℓ_{γ_k}:

$$\ell_q(R(b(x), t; \theta_r), y) = \frac{1}{K}\sum_{k=1}^{K} \ell_{\gamma_k}(y - q_{\gamma_k}(b(x), t)),$$

where $q_{\gamma_k}(b(x), t)$ is the kth qauntile output of $R(b(x), t; \theta_r)$ under treatment t.

- For the *T-Tower*'s, we consider the cross entropy loss

$$\ell_{ce}(\pi(b(x); \theta_\pi), t) = \frac{1}{M}\sum_{k=0}^{M-1} t^{(k)} \log(\pi(b(x), ; \theta_\pi)^{(k)}), \tag{2}$$

where $\mathbf{t} = (t^{(0)}, t^{(1)}, \dots, t^{(M-1)})^T$ is the one-hot vector of treatment, and $\pi(b(x); \theta_\pi) = (\pi(b(x); \theta_\pi)^{(0)}, \pi(b(x); \theta_\pi)^{(1)}, \dots, \pi(b(x); \theta_\pi)^{(M-1)})^T$ is the predicted score.

- The final loss of DNet for on sample $\{(x_i, t_i, y_i)\}_{i=1}^{N}$ is given by

$$\mathcal{L}_N(b, R, \pi) = \frac{1}{N}\sum_{i=1}^{N} \ell_q(R(b(x_i), t_i; \theta_r), y_i) + \omega \ell_{ce}(\pi(b(x_i); \theta_\pi), t_i),$$

where ω is a weight parameter that balances the two loss components.

Learning Guarantee: Assumption

- Define the target function of the *BaseNet*, *R-Tower* and the *T-Tower* to be b_0, R_0 and π_0 respectively, which satisfy

$$(b_0, R_0, \pi_0) = \arg\min_{(b,R,\pi)} \mathcal{L}(b, R, \pi).$$

Learning Guarantee: Assumption

- Define the target function of the *BaseNet*, *R-Tower* and the *T-Tower* to be b_0, R_0 and π_0 respectively, which satisfy

$$(b_0, R_0, \pi_0) = \arg\min_{(b,R,\pi)} \mathcal{L}(b, R, \pi).$$

- Let $\hat{b}_N$,$\hat{R}_N$ and $\hat{\pi}_N$ denote the empirical risk minimizer of the empirical loss, i.e.,

$$(\hat{b}_N, \hat{R}_N, \hat{\pi}_N) = \arg\min_{b\in\mathcal{F}_b, R\in\mathcal{F}_R, \pi\in\mathcal{F}_\pi} \mathcal{L}_N(b, R, \pi).$$

Learning Guarantee: Assumption

- Define the target function of the *BaseNet*, *R-Tower* and the *T-Tower* to be b_0, R_0 and π_0 respectively, which satisfy

$$(b_0, R_0, \pi_0) = \arg\min_{(b,R,\pi)} \mathcal{L}(b, R, \pi).$$

- Let $\hat{b}_N$,$\hat{R}_N$ and $\hat{\pi}_N$ denote the empirical risk minimizer of the empirical loss, i.e.,

$$(\hat{b}_N, \hat{R}_N, \hat{\pi}_N) = \arg\min_{b\in\mathcal{F}_b, R\in\mathcal{F}_R, \pi\in\mathcal{F}_\pi} \mathcal{L}_N(b, R, \pi).$$

Assumption

(C1) : *The domain of the input of* b_0 *is* $\mathcal{X} = [0, 1]^d$. *The probability distribution of* X *is absolutely continuous w.r.t the Lebesgue measure.*

(C2) : *The target* b_0 *is* β_b*-Hölder smooth with constant* B_b.

(C3) : *The target* R_0 *is* β_R*-Hölder smooth with constant* B_R.

(C4) : *The target* π_0 *is* β_π*-Hölder smooth with constant* B_π.

Learning Guarantee

Theorem (Non-asymptotic Upper bounds)

For any integers N_b, M_b, N_R, M_R and N_π, M_π, let widths and depths in $\mathcal{F}_b, \mathcal{F}_R, \mathcal{F}_\pi$ be
$\mathcal{W}_b = 38(\lfloor\beta_b\rfloor + 1)^2 d_1 d_0^{\lfloor\beta_b\rfloor+1} N_b \log_2(8N_b), \mathcal{D}_b = 21(\lfloor\beta_b\rfloor + 1)^2 d_0^{\lfloor\beta_b\rfloor+1} M_b \log_2(8M_b),$
$\mathcal{W}_R = 38(\lfloor\beta_R\rfloor + 1)^2 K d_1^{\lfloor\beta_R\rfloor+1} N_R \log_2(8N_R), \mathcal{D}_R = 21(\lfloor\beta_R\rfloor + 1)^2 d_1^{\lfloor\beta_R\rfloor+1} M_R \log_2(8M_R),$
$\mathcal{W}_\pi = 38(\lfloor\beta_\pi\rfloor + 1)^2 M d_1^{\lfloor\beta_\pi\rfloor+1} N_\pi \log_2(8N_\pi), \mathcal{D}_\pi = 21(\lfloor\beta_\pi\rfloor + 1)^2 d_1^{\lfloor\beta_\pi\rfloor+1} M_\pi \log_2(8M_\pi),$
then for any $\delta > 0$, with probability at least $1 - \delta$

$$
\begin{aligned}
\mathcal{R}(\hat{b}_N, \hat{R}_N, \hat{\pi}_N) &= \mathcal{L}(\hat{b}_N, \hat{R}_N, \hat{\pi}_N) - \mathcal{L}(b_0, R_0, \pi_0) \\
&\leq 6\mathcal{B}_R\{(\mathcal{S}_b + \mathcal{S}_R)(\mathcal{D}_b + \mathcal{D}_R)(d_0 + 1)\log(N\max\{\mathcal{W}_b, \mathcal{W}_R\})\}^{1/2} N^{-1/2} \\
&+ 6\omega(log(M) + 2B_\pi)\{(\mathcal{S}_b + \mathcal{S}_\pi)(\mathcal{D}_b + \mathcal{D}_\pi) d_0 \log(N\max\{\mathcal{W}_b, \mathcal{W}_\pi\})\}^{1/2} N^{-1/2} \\
&+ 6(\omega(log(M) + 2B_\pi) + B_R)\{\log(4\max\{M, K\}/\delta)\}^{1/2}(2N)^{-1/2} \\
&+ 18B_R(\lfloor\beta_R\rfloor + 1)^2 d_1^{\lfloor\beta_R\rfloor+1+(\beta_R\vee 1)/2}(N_R M_R)^{-2\beta_R/d_1} \\
&+ 18\omega B_\pi(\lfloor\beta_\pi\rfloor + 1)^2 d_1^{\lfloor\beta_\pi\rfloor+1+(\beta_\pi\vee 1)/2}(N_\pi M_\pi)^{-2\beta_\pi/d_1} \\
&+ 18(B_R + \omega B_\pi)B_b(\lfloor\beta_b\rfloor + 1)^2 d_0^{\lfloor\beta_b\rfloor+1+(\beta_b\vee 1)/2}(N_b M_b)^{-2\beta_b/d_0},
\end{aligned}
$$

where d_0 and d_1 is the dimension of the input and output respectively of neural networks in $\mathcal{F}_b$.

Learning Guarantee

Corollary

Suppose the conditions in previous Theorem hold and $\beta_b/d_0 < \min\{\beta_R/d_1, \beta_\pi/d_1\}$*. Let* $N_b = N_R = N_\pi = 1$*, and* $M_b = N^{d_0/[2(d_0+2\beta_b)]}$*,* $M_R = N^{d_1/[2(d_1+2\beta_R)]}$*,* $M_\pi = N^{d_1/[2(d_1+2\beta_\pi)]}$*. Then then for any* $\delta > 0$*, with probability at least* $1-\delta$*,*

$$\mathcal{R}(\hat{b}_N, \hat{R}_N, \hat{\pi}_N) \leq C_0[\mathcal{B}_R + \omega(log(M) + 2B_\pi)](\log N)^3 N^{-\beta_b/(2\beta_b+d_0)} + 6(\omega(log(M) + 2B_\pi) + B_R)\{\log(4\max\{M, K\}/\delta)\}^{1/2}(2N)^{-1/2},$$

where $C_0 > 0$ *is a constant depending only on* $\beta_b, \beta_R, \beta_\pi, d_0, d_1, M$ *and* K*. Simply*

$$\mathcal{R}(\hat{b}_N, \hat{R}_N, \hat{\pi}_N) = O_p((\log N)^3 N^{-\beta_b/(2\beta_b+d_0)}).$$

- d_0, d_1 are the dimension of the covariate and embedded features, $\beta_b, \beta_R, \beta_\pi$ are the smoothness of the targets b_0, R_0 and π_0.

Learning Guarantee

Corollary

Suppose the conditions in previous Theorem hold and $\beta_b/d_0 < \min\{\beta_R/d_1, \beta_\pi/d_1\}$*. Let* $N_b = N_R = N_\pi = 1$*, and* $M_b = N^{d_0/[2(d_0+2\beta_b)]}$*,* $M_R = N^{d_1/[2(d_1+2\beta_R)]}$*,* $M_\pi = N^{d_1/[2(d_1+2\beta_\pi)]}$*. Then then for any* $\delta > 0$*, with probability at least* $1-\delta$*,*

$$\mathcal{R}(\hat{b}_N, \hat{R}_N, \hat{\pi}_N) \leq C_0[\mathcal{B}_R + \omega(log(M) + 2B_\pi)](\log N)^3 N^{-\beta_b/(2\beta_b+d_0)} \\ + 6(\omega(log(M) + 2B_\pi) + B_R)\{\log(4\max\{M, K\}/\delta)\}^{1/2}(2N)^{-1/2},$$

where $C_0 > 0$ *is a constant depending only on* $\beta_b, \beta_R, \beta_\pi, d_0, d_1, M$ *and* K*. Simply*

$$\mathcal{R}(\hat{b}_N, \hat{R}_N, \hat{\pi}_N) = O_p((\log N)^3 N^{-\beta_b/(2\beta_b+d_0)}).$$

- d_0, d_1 are the dimension of the covariate and embedded features, $\beta_b, \beta_R, \beta_\pi$ are the smoothness of the targets b_0, R_0 and π_0.
- Assumed $\beta_b/d_0 < \min\{\beta_R/d_1, \beta_\pi/d_1\}$ as in practice d_0 is usually large and d_1 extracted features is relatively small.

Learning Guarantee

Corollary

Suppose the conditions in previous Theorem hold and $\beta_b/d_0 < \min\{\beta_R/d_1, \beta_\pi/d_1\}$. *Let* $N_b = N_R = N_\pi = 1$, *and* $M_b = N^{d_0/[2(d_0+2\beta_b)]}$, $M_R = N^{d_1/[2(d_1+2\beta_R)]}$, $M_\pi = N^{d_1/[2(d_1+2\beta_\pi)]}$. *Then then for any* $\delta > 0$, *with probability at least* $1-\delta$,

$$\mathcal{R}(\hat{b}_N, \hat{R}_N, \hat{\pi}_N) \le C_0[\mathcal{B}_R + \omega(\log(M) + 2B_\pi)](\log N)^3 N^{-\beta_b/(2\beta_b+d_0)} + 6(\omega(\log(M) + 2B_\pi) + B_R)\{\log(4\max\{M, K\}/\delta)\}^{1/2}(2N)^{-1/2},$$

where $C_0 > 0$ *is a constant depending only on* $\beta_b, \beta_R, \beta_\pi, d_0, d_1, M$ *and* K. *Simply*

$$\mathcal{R}(\hat{b}_N, \hat{R}_N, \hat{\pi}_N) = O_p((\log N)^3 N^{-\beta_b/(2\beta_b+d_0)}).$$

- d_0, d_1 are the dimension of the covariate and embedded features, $\beta_b, \beta_R, \beta_\pi$ are the smoothness of the targets b_0, R_0 and π_0.
- Assumed $\beta_b/d_0 < \min\{\beta_R/d_1, \beta_\pi/d_1\}$ as in practice d_0 is usually large and d_1 extracted features is relatively small.
- Generally, the rate is $O_p(N^{-\min\{\beta_b/(2\beta_b+d_0), \beta_R/(2\beta_R+d_1), \beta_\pi/(2\beta_\pi+d_1)\}})$ depends on ratios β_b/d_0, β_R/d_1, and β_π/d_1.

Implementation Variants

There are some variants of DNet implementations used to accommodate some real-world tasks.

- **Mono-DNet**
- We propose a monotonic DNet (Mono-DNet) by imposing the monotonic treatment constraint during the training phase.

- **ZI-DNet**
- Involving an auxiliary task for predicting whether the outcome is zero to predict response from a zero-inflated heavy-tailed distribution.

Semi-synthetic Datasets

	IHDP		ACIC	
	$\sqrt{\epsilon PEHE_{in}}$	$\sqrt{\epsilon PEHE_{out}}$	$\sqrt{\epsilon PEHE_{in}}$	$\sqrt{\epsilon PEHE_{out}}$
TARNET	0.88	0.95	4.35	4.69
CFR Wass	0.71	0.76	3.10	3.42
CFR MMD	0.73	0.77	3.08	3.38
DragonNet	0.68	0.77	4.04	4.35
DNet	**0.49±0.02**	**0.56±0.03**	**1.87± 0.18**	**2.34± 0.15**

Table: Performance summary of IHDP (Infant Health and Development Program) and ACIC (2019 Atlantic Causal Inference Conference competition. *in* stands for train and validation datasets while *out* stands for test set. PEHE denotes the Precision in Estimation of Heterogeneous Effect (PEHE) as the evaluation metric.

Real Data

To evaluate the effectiveness of the proposed DNet architecture in real-world scenarios, we conduct online randomized controlled experiments and collect two datasets from a leading technology company.

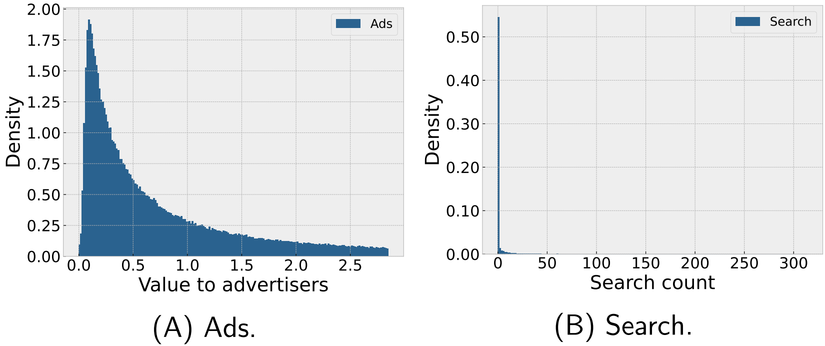


Figure: Histograms of outcomes in Ads/Search datasets. .

Real Data:DNet

	Ads	Search
TARNET	0.53 ± 0.03	1.12 ± 0.05
CFR Wass	0.48 ± 0.05	0.89 ± 0.04
CFR MMD	0.49 ± 0.03	0.87 ± 0.03
DragonNet	0.56 ± 0.03	1.13 ± 0.05
DNet	$\mathbf{0.59\pm0.02}$	$\mathbf{1.16\pm0.04}$

Table: Average AUUC of all treatments for Ads and Search datasets.

Real Data:Mono-DNet

	T=1	T=2	T=3	T=4	Mean
DNet	0.53	0.58	0.68	0.58	0.59
Mono-DNet	**0.70**	**0.70**	**0.84**	**0.79**	**0.76**

Table: The Areas Under Uplift Curve (AUUC) of DNet and Monotonic-DNet models on value to advertiser in the ads dataset.

Real Data:ZI-DNet

	T=1	T=2	T=3	T=4	
DNet	0.84	1.02	0.96	1.05	
ZI-DNet	**0.90**	**1.12**	**1.04**	**1.11**	
	T=5	T=6	T=7	T=8	Mean
DNet	1.33	2.13	0.96	0.98	1.16
ZI-DNet	**1.52**	**2.26**	**1.13**	**0.96**	**1.26**

Table: AUUCs of DNet and ZI-DNet models on search counts in the search dataset.

Ablation Study

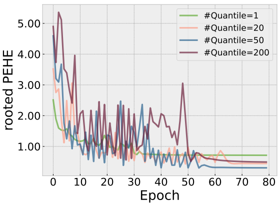


Figure: Validation PEHE versus training epochs.

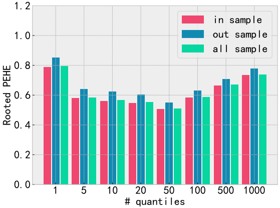


Figure: Rooted PEHE on IHDP dataset of models with different number of quantiles in NCQ Layer.

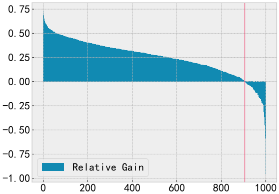


Figure: Relative differences of rooted PEHE on various tasks.

Online Deployment

- In the rewarded ads example, the optimal policy based on DNet architecture was able to achieve 2.8% significant increases in value to advertisers,
- In the search example, ZI-DNet was able to improve the number of search counts by more than 13%.
- Additionally, the DNet model has been adopted by the monetization department to improve user experience, resulting in a significant 0.1% increase in user activity.

Application to Distributional Reinforcement Learning

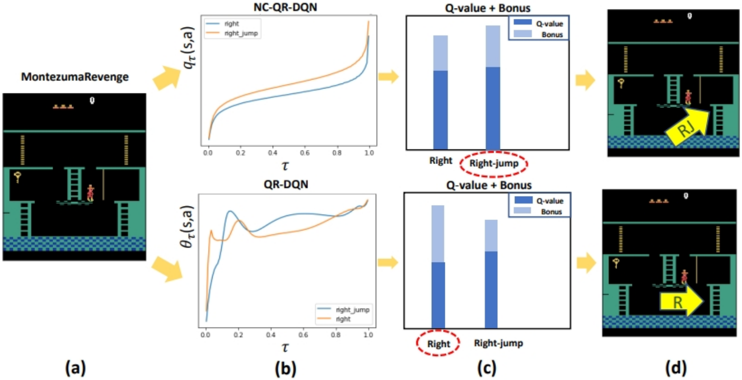


Figure: An Atari example to show how the crossing issue may affect the exploration efficiency.

Application to Distributional Reinforcement Learning

Given a Markov decision process (MDP) with $(\mathcal{X}, \mathcal{A}, R, P, \gamma)$,

- $\mathcal{X}$ and $\mathcal{A}$ are state and action spaces

Application to Distributional Reinforcement Learning

Given a Markov decision process (MDP) with $(\mathcal{X}, \mathcal{A}, R, P, \gamma)$,

- $\mathcal{X}$ and $\mathcal{A}$ are state and action spaces
- R is the r.v. reward function and $\gamma \in [0, 1)$ is the discount factor

Application to Distributional Reinforcement Learning

Given a Markov decision process (MDP) with $(\mathcal{X}, \mathcal{A}, R, P, \gamma)$,

- $\mathcal{X}$ and $\mathcal{A}$ are state and action spaces
- R is the r.v. reward function and $\gamma \in [0, 1)$ is the discount factor
- $P(x' \mid x, a)$ is the transition probability

Application to Distributional Reinforcement Learning

Given a Markov decision process (MDP) with $(\mathcal{X}, \mathcal{A}, R, P, \gamma)$,

- $\mathcal{X}$ and $\mathcal{A}$ are state and action spaces
- R is the r.v. reward function and $\gamma \in [0, 1)$ is the discount factor
- $P(x' \mid x, a)$ is the transition probability
- A policy $\pi(\cdot \mid x)$ maps each state $x \in \mathcal{X}$ to a distribution over $\mathcal{A}$.

Application to Distributional Reinforcement Learning

Given a Markov decision process (MDP) with $(\mathcal{X}, \mathcal{A}, R, P, \gamma)$,

- $\mathcal{X}$ and $\mathcal{A}$ are state and action spaces
- R is the r.v. reward function and $\gamma \in [0, 1)$ is the discount factor
- $P(x' \mid x, a)$ is the transition probability
- A policy $\pi(\cdot \mid x)$ maps each state $x \in \mathcal{X}$ to a distribution over $\mathcal{A}$.
- For a fixed π, the return is a r.v. of the sum of discounted rewards observed along one trajectory of states while following π.

$$Z^{\pi} = \sum_{t=0}^{\infty} \gamma^{t} R_{t}.$$

Application to Distributional Reinforcement Learning

Given a Markov decision process (MDP) with $(\mathcal{X}, \mathcal{A}, R, P, \gamma)$,

- $\mathcal{X}$ and $\mathcal{A}$ are state and action spaces
- R is the r.v. reward function and $\gamma \in [0, 1)$ is the discount factor
- $P(x' \mid x, a)$ is the transition probability
- A policy $\pi(\cdot \mid x)$ maps each state $x \in \mathcal{X}$ to a distribution over $\mathcal{A}$.
- For a fixed π, the return is a r.v. of the sum of discounted rewards observed along one trajectory of states while following π.

$$Z^{\pi} = \sum_{t=0}^{\infty} \gamma^{t} R_{t}.$$

Problem definition

We want to estimate the distribution of Z^{π} as well as get an optimal one Z^{π^*} in the sense that $\mathbb{E}Z^{\pi^*} \geq \mathbb{E}Z^{\pi}$ for any π.

Algorithm

Algorithm 1 Distributional RL with fitted NC Iteration

Require: MDP $(\mathcal{X}, \mathcal{A}, P, R, \gamma)$, sampling distribution σ, # samples N, # quantile levels K, # iterations M, NC networks $\mathcal{F}$, the initial estimator $Z^{(0)} = (Z_1^{(0)}, \dots, Z_K^{(0)})$.

for iteration $m = 0$ to $M - 1$ **do**

Sample i.i.d. observations $\{(x_i, a_i, r_i, x_i')\}_{i \in [N]}$.

Compute $(\mathcal{T} Z_k^{(m)})_i = r_i + \gamma Z_k^{(m)}(x', a')$ where $a' = \arg\max_{a \in \mathcal{A}} \sum_{k=1}^{K} Z_k^{(m)}(x', a)$

Update the estimation

$$Z^{(m+1)} \leftarrow \arg\min_{Z \in \mathcal{F}} \frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{K} \sum_{j=1}^{K} \rho_{\tau_k} \left((\mathcal{T} Z_j^{(m)})_i - Z_k(x_i, a_i) \right),$$

end for

Define policy π_M as the greedy policy with respect to $Q^{(M)}$.

Output: An estimator $Z^{(M)}$ of Z^* and the policy π_M

Learning Guarantee: Assumptions

- Modify NQ networks $\mathcal{F}_N$ for the value distribution estimation of distribution RL:

$$\mathcal{F}_N^{(RL)} = \{f : \mathcal{X} \times \mathcal{A} \to \mathbb{R} : f(\cdot, a) \in \mathcal{F}_N \text{ for any } a \in \mathcal{A}\}. \quad (3)$$

Learning Guarantee: Assumptions

- Modify NQ networks $\mathcal{F}_N$ for the value distribution estimation of distribution RL:

$$\mathcal{F}_N^{(RL)} = \{f : \mathcal{X} \times \mathcal{A} \to \mathbb{R} : f(\cdot, a) \in \mathcal{F}_N \text{ for any } a \in \mathcal{A}\}. \quad (3)$$

Assumption (Approximation efficiency characterization)

For any $f \in \mathcal{F}_N^{(RL)}$ and any $a, a' \in \mathcal{A}$, the function $R_\tau(\cdot, a) + \gamma f(\cdot, a')$ is β-Hölder smooth with constant B, where $R_\tau(x, a)$ denotes the τ-th conditional quantile of the reward given the state x and action a.

Learning Guarantee: Assumptions

- Modify NQ networks $\mathcal{F}_N$ for the value distribution estimation of distribution RL:

$$\mathcal{F}_N^{(RL)} = \{f : \mathcal{X} \times \mathcal{A} \to \mathbb{R} : f(\cdot, a) \in \mathcal{F}_N \text{ for any } a \in \mathcal{A}\}. \quad (3)$$

Assumption (Approximation efficiency characterization)

For any $f \in \mathcal{F}_N^{(RL)}$ and any $a, a' \in \mathcal{A}$, the function $R_\tau(\cdot, a) + \gamma f(\cdot, a')$ is β-Hölder smooth with constant B, where $R_\tau(x, a)$ denotes the τ-th conditional quantile of the reward given the state x and action a.

Assumption (Self-calibration)

There exist constants $C > 0$ and $c > 0$ such that for any $|\delta| \leq C$ and $m = 0, \ldots, M-1$,

$$|P_{\mathcal{T}Z^{(m)}|x,a}((\mathcal{T}Z^{(m)})_\tau(x+\delta, a)) - P_{\mathcal{T}Z^{(m)}|x,a}((\mathcal{T}Z^{(m)})_\tau(x))| \geq c|\delta|,$$

for all $\tau \in (0,1)$ and $x \in \mathcal{X}, a \in \mathcal{A}$ up to a negligible set, where $P_{\mathcal{T}Z^{(m)}|x,a}(\cdot)$ denotes the conditional distribution function of $\mathcal{T}Z^{(m)}$ given x and a and $(\mathcal{T}Z^{(m)})_\tau$ denotes the τ conditional quantile given x and a.

Learning Guarantee

Theorem

Let $\{Z^{(m)}\}_{m=0}^{M}$ *be the iterates in Algorithm 1 using NQ networks* $\mathcal{F}_N^{(RL)}$*. Let the width and depth for networks be* $\mathcal{W} = 114(\lfloor\beta\rfloor+1)^2(K+1)(d_0)^{\lfloor\beta\rfloor+1}$ *and depth* $\mathcal{D} = 21(\lfloor\beta\rfloor+1)^2(d_0)^{\lfloor\beta\rfloor+1}N^{d_0/[2(d_0+2\beta)]}\log_2(8N^{d_0/[2(d_0+2\beta)]})$ *and bound* $\mathcal{B} = \log(N)$ *where* N *is the sample size. Denote* Z^{π_M} *by the action-value distribution w.r.t the greedy policy* π_M *from* $Z^{(M)}$*. Then*

$$\|\mathbb{E}Z^{\pi_M} - \mathbb{E}Z^*\|_{1,\mu} \leq \frac{2c\cdot c_{M,\sigma,\mu}(K+2)^3\gamma}{(1-\gamma)^2}|\mathcal{A}|(\log N)^4 N^{-\beta/(2\beta+d_0)} + \frac{4\gamma^{M+1}}{(1-\gamma)^2}R_{max}, \quad (4)$$

where $c_{\mu,\sigma} > 0$ *is a constant that only depends on the prob. dist.* μ *and sampling dist.* σ *and* $c > 0$ *is a universal constant.*

- Prediction error: the sum of estimation error and algorithmic error

Learning Guarantee

Theorem

Let $\{Z^{(m)}\}_{m=0}^{M}$ be the iterates in Algorithm 1 using NQ networks $\mathcal{F}_N^{(RL)}$. Let the width and depth for networks be $\mathcal{W} = 114(\lfloor\beta\rfloor + 1)^2(K+1)(d_0)^{\lfloor\beta\rfloor+1}$ and depth $\mathcal{D} = 21(\lfloor\beta\rfloor + 1)^2(d_0)^{\lfloor\beta\rfloor+1}N^{d_0/[2(d_0+2\beta)]}\log_2(8N^{d_0/[2(d_0+2\beta)]})$ and bound $\mathcal{B} = \log(N)$ where N is the sample size. Denote Z^{π_M} by the action-value distribution w.r.t the greedy policy π_M from $Z^{(M)}$. Then

$$\|\mathbb{E}Z^{\pi_M} - \mathbb{E}Z^*\|_{1,\mu} \leq \frac{2c \cdot c_{M,\sigma,\mu}(K+2)^3\gamma}{(1-\gamma)^2}|\mathcal{A}|(\log N)^4 N^{-\beta/(2\beta+d_0)} + \frac{4\gamma^{M+1}}{(1-\gamma)^2}R_{max}, \tag{4}$$

where $c_{\mu,\sigma} > 0$ is a constant that only depends on the prob. dist. μ and sampling dist. σ and $c > 0$ is a universal constant.

- Prediction error: the sum of estimation error and algorithmic error
- Algorithmic error converges to zero linearly in # iterations M. Estimation error dominates when iterations $M \geq C[\log|\mathcal{A}|^{-1} + (\beta/(2\beta + d_0))\log(N)]$

Learning Guarantee

Theorem

Let $\{Z^{(m)}\}_{m=0}^{M}$ be the iterates in Algorithm 1 using NQ networks $\mathcal{F}_N^{(RL)}$. Let the width and depth for networks be $\mathcal{W} = 114(\lfloor\beta\rfloor + 1)^2(K+1)(d_0)^{\lfloor\beta\rfloor+1}$ and depth $\mathcal{D} = 21(\lfloor\beta\rfloor + 1)^2(d_0)^{\lfloor\beta\rfloor+1}N^{d_0/[2(d_0+2\beta)]}\log_2(8N^{d_0/[2(d_0+2\beta)]})$ and bound $\mathcal{B} = \log(N)$ where N is the sample size. Denote Z^{π_M} by the action-value distribution w.r.t the greedy policy π_M from $Z^{(M)}$. Then

$$\|\mathbb{E}Z^{\pi_M} - \mathbb{E}Z^*\|_{1,\mu} \leq \frac{2c \cdot c_{M,\sigma,\mu}(K+2)^3\gamma}{(1-\gamma)^2}|\mathcal{A}|(\log N)^4 N^{-\beta/(2\beta+d_0)} + \frac{4\gamma^{M+1}}{(1-\gamma)^2}R_{max}, \qquad (4)$$

where $c_{\mu,\sigma} > 0$ is a constant that only depends on the prob. dist. μ and sampling dist. σ and $c > 0$ is a universal constant.

- Prediction error: the sum of estimation error and algorithmic error
- Algorithmic error converges to zero linearly in # iterations M. Estimation error dominates when iterations $M \geq C[\log|\mathcal{A}|^{-1} + (\beta/(2\beta+d_0))\log(N)]$

Learning Guarantee

Theorem

Let $\{Z^{(m)}\}_{m=0}^{M}$ be the iterates in Algorithm 1 using NQ networks $\mathcal{F}_N^{(RL)}$. Let the width and depth for networks be $\mathcal{W} = 114(\lfloor\beta\rfloor + 1)^2(K+1)(d_0)^{\lfloor\beta\rfloor+1}$ and depth $\mathcal{D} = 21(\lfloor\beta\rfloor+1)^2(d_0)^{\lfloor\beta\rfloor+1}N^{d_0/[2(d_0+2\beta)]}\log_2(8N^{d_0/[2(d_0+2\beta)]})$ and bound $\mathcal{B} = \log(N)$ where N is the sample size. Denote Z^{π_M} by the action-value distribution w.r.t the greedy policy π_M from $Z^{(M)}$. Then

$$\|\mathbb{E}Z^{\pi_M} - \mathbb{E}Z^*\|_{1,\mu} \le \frac{2c\cdot c_{M,\sigma,\mu}(K+2)^3\gamma}{(1-\gamma)^2}|\mathcal{A}|(\log N)^4 N^{-\beta/(2\beta+d_0)} + \frac{4\gamma^{M+1}}{(1-\gamma)^2}R_{max}, \qquad (4)$$

where $c_{\mu,\sigma} > 0$ is a constant that only depends on the prob. dist. μ and sampling dist. σ and $c > 0$ is a universal constant.

- Prediction error: the sum of estimation error and algorithmic error
- Algorithmic error converges to zero linearly in # iterations M. Estimation error dominates when iterations $M \ge C[\log|\mathcal{A}|^{-1} + (\beta/(2\beta+d_0))\log(N)]$
- Then prediction error has rate $|\mathcal{A}|N^{-\beta/(2\beta+d_0)}$, which is linearly in the cardinality $|\mathcal{A}|$

Application to Distributional Reinforcement Learning

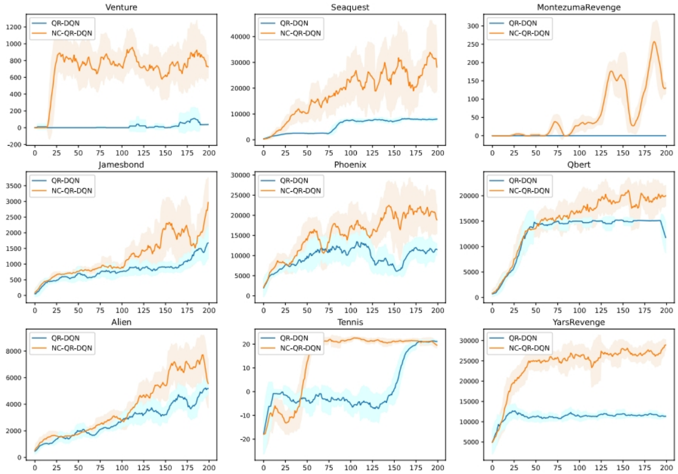


Figure: Performance comparison with QR-DQN. Each training curve is averaged by seeds.

Table of Contents

Conclusion

- Non-crossing Quantile regression network.
  - Delta layer with ELU activation for non-negative outputs
  - Learning guarantees, faster rate with low-dim structured data
- Applications to Conditional Average Treatment Effect
  1. Extension to DNet, a robust non-crossing NN architecture for quantile ITE estimation with heavy-tailed outcomes.
  2. Two variants of DNet that lead to improved AUUC scores in real-world applications.
- Applications to Distributional Reinforcement Learning
  1. Making use of global information to ensure the batch-based monotonicity of the learned quantile function based on NQ network.

Table of Contents

- Fan Zhou, Xiaocheng Tang, Chenfan Lu, Fan Zhang, Zhiwei Qin, Jieping Ye, and Hongtu Zhu "Multi-Objective Distributional Reinforcement Learning for Large-Scale Order Dispatching", *IEEE ICDM* 2021.
- Qin, Z., Zhu, H.T., and Jieping Ye. Reinforcement learning for ridesharing: an extended survey. *Transportation Research Part C: Emerging Technologies* 2022, 144, p. 103852.
- Wu, G., Song, G., Lv, X., Luo, S., Shi, C. and Zhu, H. DNet: Distributional network for distributional individualized treatment effects. *KDD* 2023.
- Shen, G., Luo, S., Shi, C., and Zhu, H. Deep Noncrossing Quantile Learning. In submission.
- Li, T., Shi, C., Lu, Z., Li, Y., and Zhu, H.T. Evaluating Dynamic Conditional Quantile Treatment Effects with Applications in Ridesharing. *Journal of American Statistical Association, AC & S*, 2024, in press.

Thank you!


STATISTICAL SCIENCE IN
ARTIFICIAL
INTELLIGENCE
JASA SPECIAL ISSUE
SUBMIT BY
DEC 31, 2024
Information:
www.reallygreatsite.com
Identification of Core AI Problem
Statistical Contributions to AI
Innovative Statistical Theory,
Method and Applications