Published in final edited form as:

Stat Theory Relat Fields. 2018 ; 2(1): 80–88. doi:10.1080/24754269.2018.1466096.

Deep advantage learning for optimal dynamic treatment regime

Shuhan Liang¹, Wenbin Lu¹, and Rui Song¹

¹Department of Statistics, North Carolina State University, Raleigh, NC 27695, USA

Abstract

Recently deep learning has successfully achieved state-of-the-art performance on many difficult tasks. Deep neural network outperforms many existing popular methods in the field of reinforcement learning. It can also identify important covariates automatically. Parameter sharing of convolutional neural network (CNN) greatly reduces the amount of parameters in the neural network, which allows for high scalability. However few research has been done on deep advantage learning (A-learning). In this paper, we present a deep A-learning approach to estimate optimal dynamic treatment regime. A-learning models the advantage function, which is of direct relevance to the goal. We use an inverse probability weighting (IPW) method to estimate the difference between potential outcomes, which does not require to make any model assumption on the baseline mean function. We implemented different architectures of deep CNN and convexified convolutional neural networks (CCNN). The proposed deep A-learning methods are applied to a data from the STAR*D trial and are shown to have better performance compared with the penalized least square estimator using a linear decision rule.

Keywords

Advantage Learning; Convexified Convolutional Neural Networks; Convolutional Neural Networks; Dynamic Treatment Regime; Inverse Probability Weighting

1 Introduction

Optimal treatment regime aims to tailor medical treatment by taking into account patients' heterogeneity. Compared to the traditional "one-size-fits-all" approach, optimal treatment regime can individualize treatment and get optimal output for each patient. Different treatments include differences in treatment type and dosage level variation. For some diseases, treatment adjustment is required through the entire treatment process and multiple treatment selections are needed. Dynamic treatment regime (DTR) aims to select a sequence of treatments at multiple time points for each patient based on patient's characteristics. By following these rules, the best (maximal) response over the entire population can be achieved. One difficulty in DTR estimation is for each patient at each decision point, we only observe the response of one treatment option. The potential outcomes of other treatments are missing. Many approaches have been proposed to solve this problem such as Q-learning (Watkins [1989], Watkins and Dayan [1992]) and A-learning (Murphy [2003]).

In this paper, we propose a new method to estimate optimal dynamic treatment regime using deep A-learning.

Deep learning has been widely used in many fields such as game playing (Mnih et al. [2013]), finance (Ding et al. [2015]), robotics (Lenz et al. [2015]), control and operations research (Mnih et al. [2015]), and language processing (Collobert and Weston [2008]). There are many successful examples of implementing DNN to solve challenging problems. Google Deepmind's AlphaGo, a program using deep neural networks to play the Go game, won 99.8% of the games against other computer programs and won all 5 games against the European champion (Silver et al. [2016]). NVIDIA have implemented deep learning technique to achieve self-driving car (Bojarski et al. [2016]). Using CNN and end-to-end learning, goals like simultaneous localization and mapping and movement planning can be achieved. Besides implementations, theoretical results have been obtained. Pinkus [1999] and Hornik et al. [1989] discussed theoretical results of multilayer feedforward perceptron (MLP) approximation. Choromanska et al. [2015] have shown that as long as the size of neural network is large enough, the performances of any local minimum and global minimum are very similar on testing datasets.

Convolutional neural networks (CNN) has its own advantages due to parameter sharing and local connectivity. The parameters of each filter are shared across patches. It greatly reduces the number of parameters. LeCun et al. [1998], first successfully implemented CNN in handwritten digit recognition and lowercase words recognition. Since then many work have been done on CNN. In ImageNet LSVRC-2012 contest, Krizhevsky et al. [2012] proposed a deep CNN with dropout and GPU implementation. It successfully classifies more than 1,000,000 images into more than 1,000 categories (LeCun et al. [2015]). Its top-5 test error rate is 11% lower than second-place solution. Zhang et al. [2016] used low rank matrix to represent the filter of CNN in the reproducing kernel Hilbert space (RKHS). They further proposed CCNN by relaxing the rank constraint to nuclear norm constraint. The authors proved that under binary classification case, its generalization error has oracle inequality. The author also compared several variants of CCNN on image classification.

In this paper we combine deep convolutional neural network with A-learning. The value functions estimated by CCNN and CNN are compared with an A-learning based penalized least square estimator. The rest of the paper is organized as follows: In Section 2 we briefly discuss popular DTR techniques in literature. We formulate the A-learning based CCNN and CNN in Section 3. The detailed CCNN and CNN algorithms are included. Deep A-learning is extended to DTR estimation using backward induction. We apply the proposed methods to a data from the STAR*D clinical trial in Section 4. Network architecture and model training details are discussed. Section 5 gives conclusion and lists of avenues for future research.

2 Literature Review

2.1 Notation and assumption

Denote the predictor vector available at the j -th time point by X_j , the treatment at the j -th time point as A_j , $j = 1, 2, \dots, K$, the final observed outcome as Y . The bar notation represents a sequence of past information, e.g., $\bar{A}_j = \{A_1, A_2, \dots, A_j\}$. Let d_j denote treatment regime at

j-th time point and the asterisk notation represents optimal decision rules. There are several common assumptions needed in estimating optimal dynamic treatment regime, for example, see Basu [1980], Robins [1997] and Schulte et al. [2014]. To be specific, we need

1. No unmeasured confounders assumption:

$$A_j \perp Y_j^*(a) \left| \left\{ X_1, A_1, X_2, A_2, \dots, X_j \right\}, a \in \psi_j(\bar{X}_j, \bar{A}_{j-1}) \right.$$

Here $Y_j^*(a)$ denotes the potential outcome given treatment a is received.

$\psi_j(\bar{X}_j, \bar{A}_{j-1})$ is all possible treatments given medical and treatment history. This assumes that X_j contains sufficient information thus all predictors that interact with treatment have been observed. No unmeasured confounders assumption holds for sequentially randomized experiments.

2. Positivity assumption: the treatment sequences following dynamic treatment regime can occur. Positivity assumption can be summarized as:

$$P \left[\prod_{j=1}^K \Pr(A_j \in \psi_j(\bar{X}_j, \bar{A}_{j-1}) \mid \bar{X}_j, \bar{A}_{j-1}) > 0 \right] = 1.$$

3. Stable unit treatment assumption (SUTVA): It assumes that the outcome of a patient is only influenced by the treatment(s) he or she receives. There is no interference between subjects. It also assumes that for each treatment there is one unique version. SUTVA can be summarized as:

$$Y_j = \sum_{a \in \psi_j(\bar{X}_j, \bar{A}_{j-1})} Y_j^*(a) I(A_j = a).$$

When the above assumptions hold, the optimal DTR can be estimated based on observed data. Next we will introduce two popular DTR estimation techniques.

2.2 Q-learning

Watkins [1989] proposed Q-learning. It uses incremental dynamic programming to learn optimal action. Q-function reflects the expected outcome if at the k-th time point treatment a_k is received and at any later time points the optimal treatments are received. Value function represents the expected outcome if at the k-th and any later time points the optimal treatments are received. It can be estimated by solving estimating equations. The optimal decision rule d^* can be estimated as follows:

$$d_k^*(\bar{X}_k, \bar{A}_{k-1}) = \underset{A_k: p_k(A_k \mid \bar{X}_k, \bar{A}_{k-1}) > 0}{\operatorname{argmax}} Q_k(\bar{X}_k, \bar{A}_k), \text{ for } k = K, K-1, \dots, 1.$$

Multi-stage treatment regime estimation is based on backward induction proposed by Cowell et al. [2006]. One drawback of Q-learning is that it not consistent if Q-function is

misspecified. In the next section, we will review A-learning, which is less sensitive to model misspecification.

2.3 Advantage learning

Murphy [2003] proposed Advantage learning. A-learning explicitly model the contrast function/regret function. Regret function is the difference in potential outcome between actually received and optimal treatment. Under the A-learning framework, optimal decision rule can be derived directly. From now on, we only consider the case where binary treatment choices are available. The two treatments are denoted as 0 and 1 respectively. Contrast function C , and optimal treatment d^* are defined as follows:

$$C_k(\bar{X}_k, \bar{A}_{k-1}) = Q_k(\bar{X}_k, \bar{A}_{k-1}, A_k = 1) - Q_k(\bar{X}_k, \bar{A}_{k-1}, A_k = 0),$$

$$d_k^*(\bar{X}_k, \bar{A}_{k-1}) = I\{C_k(\bar{X}_k, \bar{A}_{k-1}) > 0\}.$$

The contrast function can be estimated using g-estimation proposed by Moodie et al. [2007]

A-learning has the double-robustness property which makes it suffer less from model misspecification. A-learning makes it possible to build a complex model for baseline function and an easy-to-interpret model of contrast function. It reduces in the influence of model misspecification and generates a relative simple decision rule.

2.4 DTR estimation with variable selection

One way to improve decision accuracy of dynamic treatment regime is via variable selection. Qian and Murphy [2011] extended Q-learning using l_1 -penalized least square (PLS). However it still suffers problems that rises from Q-learning. The estimator derived from the two-step procedure may not be consistent if the conditional mean model is misspecified. Lu et al. [2013] considered model selection for estimating optimal treatment regime via penalized least square. Shi et al. [2016] extended Lu's method to cases where the propensity score is unknown. They studied the theoretical properties of the proposed estimator given the number of covariates is of the non-polynomial (NP) order of the sample size. Shi et al. [2017] studied penalizing A-learning estimation equations for dynamic treatment regime. Besides Q- and A- learning framework, Zhao et al. [2012] proposed outcome weighted learning (OWL). The optimal decision rule is derived by maximizing the value function estimator. Song et al. [2015] proposed penalized outcome weighted learning (POWL) which adds a variable selection module to OWL. Penalty functions include lasso (Tibshirani [1996]) and SCAD (Fan and Li [2001]).

3 Method

3.1 Inverse probability weighted estimator

We start with one stage optimal treatment regime estimation and extend it to multi-stage in later section. Zhang et al. [2012] proposed the inverse probability weighted estimator (IPWE). The IPWE estimator of $EY(d^*)$ is:

$$C_{\eta, i} = A_i d^*(X_i, \eta) + (1 - A_i) [1 - d^*(X_i, \eta)],$$

$$IPWE(\eta) = \frac{1}{n} \sum_{i=1}^n \frac{C_{\eta, i} Y_i}{\pi(X_i)^{A_i} (1 - \pi(X_i))^{1 - A_i}}.$$

η is the parameter in decision function d^* . $\pi(X_i)$ is the known propensity score of patient i receives treatment 1. Optimal treatment regime is estimated by maximizing IPWE, which is equivalent to estimating the contrast function:

$$\hat{C}_{IPWE}(X_i) = \frac{A_i}{\pi(X_i)} Y_i - \frac{1 - A_i}{1 - \pi(X_i)} Y_i.$$

We now show that given X_i , $\hat{C}_{IPWE}(X_i)$ is an unbiased estimator of contrast function.

Specifically,

$$\begin{aligned} & E \left[\frac{A_i}{\pi(X_i)} Y_i - \frac{1 - A_i}{1 - \pi(X_i)} Y_i \middle| X_i \right] \\ &= E \left[\frac{Y_i [1 - \pi(X_i)]}{\pi(X_i) [1 - \pi(X_i)]} \middle| A_i = 1, X_i \right] \pi(X_i) \\ &+ E \left[\frac{Y_i [1 - \pi(X_i)]}{\pi(X_i) [1 - \pi(X_i)]} \middle| A_i = 0, X_i \right] [1 - \pi(X_i)] \\ &= E[Y_i | A_i = 1, X_i] - E[Y_i | A_i = 0, X_i]. \end{aligned}$$

$\hat{C}_{IPWE}$ is the adjusted observed outcome base on propensity score. It does not posit any parametric assumptions on contrast function. A-learning requires model specification of baseline function. The IPWE does not make any assumptions on those nuisance parameters either. Therefore it suffers less from model misspecification issues. Next we propose a class of algorithms which integrate IPWE with convolutional neural network.

3.2 Deep convolutional neural network for advantage learning

3.2.1 Convolutional neural network—LeCun et al. [1998] proposed convolutional neural network. A CNN usually consists of convolutional layers, pooling layers and fully-connected layers:

1. convolutional layer: The convolutional layer takes in multi-dimensional features. In the convolutional layer, weighted summation over each region is calculated and a non-linear transformation is operated on top of the summation. Weight matrices (filters) are shared across regions so that the number of parameters is reduced. As a result, CNN is easier to train compared to fully-connected neural network with similar number of neurons. The hyper-parameters in a convolutional layer include number of filters, patch size, and stride size. The patch size is usually 3×3 or 5×5 . The stride is usually 1. Together with appropriate zero-padding, convolutional layer can preserve the width and height of input. Common activation functions include Rectified Linear Units (ReLUs): $f(x) = \max(x, 0)$, polynomial functions and hyperbolic tangent function. Krizhevsky et al. [2012] showed that ReLU can reduce the training time than other common activation functions.
2. pooling layer: Pooling layer is usually placed between convolutional layers. Local pooling can summarize information within a neighborhood. It reduces the dimension of feature space and avoid overfitting. The pooling operation is operated on non-overlapping or small-proportion-overlapping regions, which controls the correlation between hidden neurons. Two common pooling operations are maximum pooling and average pooling. The pooling layer has two parameters: pooling patch size and pooling stride size. Smaller pooling size can keep enough information and it is observed that for maximum pooling, the best pooling stride are 2 and the best patch size are 2 or 3. (Karpathy [2017]).
3. fully-connected layer: The fully connected layer maps reshaped previous output to the final output of neural network. Fully-connected layer guarantees that all neurons are connected to the output. Convolutional layer and fully-connected layer are mutually convertible. For any convolutional layer, the corresponding fully-connected layer has sparse weight matrices. Their non-zero blocks share similar patterns.

Backpropagation is a technique used for parameter update in DNN. It uses relationship of gradient between parameters of consecutive layers to train a NN. Backpropagation is implemented in many deep learning software libraries. Recent research has shown that multilayer stack architecture enhances both the abilities of capturing discriminating and ignoring irrelevant aspects. Compared to its shallow counterpart, deep NN is more capable of automatic feature representation and capturing complicated relationships.

3.2.2 A-learning based Convolutional neural network—In this section, we proposed a new approach, which integrates convolutional neural network with IPWE of contrast function. The input is all available information of each patient and the output is an estimate of IPWE of contrast function $\frac{Y_i[A_i - \pi(X_i)]}{\pi(X_i)[1 - \pi(X_i)]}$. We use least square loss to measure the prediction performance of CNN. The details of CNN is covered in Algorithm 1.

Algorithm 1:

m-layer advantage learning based CNN

Input: $(X_i, A_i, y_i)_{i=1}^n$ **Output:** predictor $\{H_m(X_i)\}_{i=1}^n$ and optimal treatment regime

$$\{I\{H_m(X_i) > 0\}\}_{i=1}^n$$

1 (1) $H_0(X_i) = X_i, i = 1, \dots, n;$ 2 (2) **for** $j \leftarrow 1$ **to** m **do**3 **if** $j < m$ **then**4 Apply the convolution filter f_j^{conv} and the pooling filter f_j^{pool} on $H_{j-1}(X_i), i = 1, \dots, n$ to get $H_j(X_i), i = 1, \dots, n;$ 5 **end**6 **else**7 Apply the fully connected layer f_m on $H_{m-1}(X_i), i = 1, \dots, n$
 to get $H_m(X_i);$ 8 **end**9 **end**10 (3) Use backpropagation to estimate all parameters in (f_j^{conv}, f_j^{pool}) , $j = 1, 2, \dots, m-1$ and f_m ;

$$\hat{A} = \operatorname{argmin}_n \frac{1}{n} \left\{ \frac{Y_i[A_i - \pi(X_i)]}{\pi(X_i)[1 - \pi(X_i)]} - H_m(X_i) \right\}^2.$$

3.3 Deep convexified convolutional neural network for advantage learning

3.3.1 Convexified convolutional neural network—Zhang et al. [2016] proposed convexified convolutional neural network (CCNN). If the activation function of CNN is smooth enough, the filter can be represented using RKHS. Some good choices of kernel functions include Gaussian kernel and inverse polynomial kernel. The parameter sharing properties of convolutional neural network results in low rank constraint, which can be relaxed to nuclear norm constraint. The network can be learned using convex optimization techniques. Compared to regular convolutional neural network, CCNN is computationally efficient and has ideal theoretical properties.

Denote $X_i \in \mathbf{R}^{d_0}$ as input, y as output $crop_1(X), \dots, crop_p(X)$ are P functions that create patches of size d_1 from input, $\mathcal{X}$ is the $n \times p$ observation matrix with training sample X_i^T as

its i -th row. $\{\omega_j \in \mathbf{R}^{d_1}\}_{j=1}^r$ are weight vectors where r is number of filters. $\beta_{r \times P}$ is the filter-patch weight matrix. $g(X)$ is the output of neural network:

$$g(X_i) = \sum_{j=1}^r \sum_{p=1}^P \beta_{j,p} \sigma(\text{crop}_p(X_i)^T \omega_j). \quad (1)$$

Under proper choices of activation function, there exists φ s.t.

$$\sigma(\langle \omega_j, z \rangle) = \langle \bar{\omega}_j, \varphi(z) \rangle$$

holds where κ is kernel function with feature map φ . $\langle \cdot \rangle$ stands for inner product. Since the parameters are estimated using only the training dataset, without loss of generality, we assume $\bar{\omega}_j \in \text{span}(\text{crop}_p(X_i))$. Denote the linear coefficients as γ_j . $Q(\mathcal{X}) \in \mathbf{R}^{nP \times s}$ is the factorization of kernel matrix of pairwise patches from training dataset, i.e.

$K = Q(\mathcal{X})Q(\mathcal{X})^T$. Here s is dimension of random feature approximation, which is explained in details in the next session. $Q(X_i) \in \mathbf{R}^{P \times s}$ is the sub-matrix of $Q(\mathcal{X})$ corresponding to X_i . It can be shown that $\langle \bar{\omega}_j, \varphi(z) \rangle = \langle Q^* v(z), Q(\mathcal{X})^T \gamma_j \rangle$, where Q^* is the pseudo-inverse of Q , and $v(z)$ is a vector with each position as $\kappa(z, \text{crop}_p(X_i))$. Denote

$$Z(x)_{P \times nP} = \begin{pmatrix} Q^* V(\text{crop}_1(x)) \\ Q^* V(\text{crop}_2(x)) \\ \dots \\ Q^* V(\text{crop}_P(x)) \end{pmatrix}, \text{ then we have:}$$

$$g(X_i) = \sum_{j=1}^r \beta_j^T Z(X_i) Q(\mathcal{X})^T \gamma_j = \text{tr} \left(Z(X_i) \left(\sum_{j=1}^r Q(\mathcal{X})^T \gamma_j \beta_j^T \right) \right) = \text{tr}(Z(X_i) B).$$

CCNN can be summarized as:

$$\left\{ g^B : \max_{j \in [r]} \|\omega_j\|_2 \leq C_\sigma(B_1) \right\} \text{ and } \max_{j \in [r]} \|\beta_j\|_2 \leq B_2 \text{ and } \text{rank}(B) = r. \quad (2)$$

where $C_\sigma(B_1)$ is a constant. To relax the non-convex constraints in (2), we consider the following class with the nuclear norm constraint:

$$\{g^B : \|B\|_* \leq C_\sigma(B_1) B_2 r\}$$

Since the optimization problem is transferred to a convex version, it is easier to compute and the resulting estimator has better theoretical properties.

3.3.2 A-learning based convexified convolutional neural network—We proposed a new approach which integrates CCNN with advantage learning. The contrast function is estimated by CCNN. For multi-layer CCNN, the low rank output of previous layer is fed to the next layer as input. If the number of channel is greater than 1, the processed patches are concatenated into one vector. This multi-channel extension technique can keep the scale of transformed data the same regardless of number of channels. Denote number of layers as m , nuclear norm regularization parameters as R . The algorithm is summarized in Algorithm 2.¹

$Q(\mathcal{X})$ can be calculated using Random Fourier Transformation proposed by Rahimi and Recht [2008]: $f_{RFT} : R^{s_0} \rightarrow R^s$

$$f_{RFT}(X_i) = \begin{pmatrix} \sqrt{\frac{2}{s}} \cos(\omega_1^T X_i + b_1) \\ \vdots \\ \sqrt{\frac{2}{s}} \cos(\omega_s^T X_i + b_s) \end{pmatrix}$$

We choose Gaussian kernel with parameter λ then $\omega_1, \dots, \omega_s$ are i.i.d. samples from $N(0, 2_{s_0} I)$ and $b_1, \dots, b_s$ are i.i.d. samples from Uniform $[0, 2\pi]$. Before training all weights and biases are randomly initialized. During the training process, we use least-square loss function to measure the difference between IPWE and output of neural network. The optimization with constraints in step 3 are achieved by projected gradient descent proposed by Duchi et al. [2008]. Parameters are updated using stochastic gradient descent followed by a projection onto the nuclear norm ball.

3.4 Multi-stage CCNN and CNN

This algorithm can be extended to dynamic treatment regime using backward induction. The framework is identical for CNN and CCNN: When estimating the decision rule for one stage, the outcome is adjusted as if during any later stages the optimal treatments have been received. The potential outcome is shifted based on contrast function estimation at each stage. Algorithm 3 is the procedure of multi-stage optimal treatment regime estimation with K decision points.

Algorithm 2:

m -layer advantage learning based CCNN

Input : $(X_i, A_i, y_i)_{i=1}^n$

Output: predictor $\{t_r(Z(H_{m-1}(X_i))\hat{B})\}_{i=1}^n$ and optimal treatment

¹The codes used in our paper are adapted from the source codes of Zhang et al. [2016] for CCNN.

$$\text{regime} \left\{ \left\{ t_r(Z(H_{m-1}(X_i))\hat{B}) > 0 \right\} \right\}_i^n = 1$$

1 $H_0(\mathcal{X}) = (\mathcal{X});$

2 **for** $j \leftarrow 1$ **to** m **do**

3 (1) Generate $P(H_{j-1}(\mathcal{X}))$, patches of $H_{j-1}(\mathcal{X})$.

4 (2) Use the random feature approximation to get $Q(H_{j-1}(\mathcal{X}))$,

which corresponds to the decomposition of kernel matrix of

$$P(H_{j-1}(\mathcal{X})).$$

5 (3) Use batch gradient descent to iteratively update $\hat{B}$:

$$\hat{B} = \underset{\|B\|_* \leq R^n}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^n \left\{ \frac{Y_i[A_i - \pi(X_i)]}{\pi(X_i)[1 - \pi(X_i)]} - \operatorname{tr}(Z(H_{j-1}(X_i))B) \right\}^2. \quad (3)$$

(4) Use singular value decomposition $B = U\Lambda V^T$ to get low

rank approximation of j -th layer output $H_j(X_i)$. Here $\hat{U}$ is the

$\text{_rst } r$ columns of U ,

$$H_j(X_i) = \hat{U}^T(Z(H_{j-1}(X_i)))^T.$$

6 **end**

Algorithm 3:

K-stage advantage learning based CCNN/CNN

Input : $(\bar{A}_{ti}, \bar{X}_{ti}, y_i)_{i=1}^n$

Output: DTR $d_k^*(\bar{A}_{(k-1)i}, \bar{X}_{ki})$, $i = 1, \dots, n$, $k = 1, \dots, K$

1 $V_{(k+1)i} = y_i$

2 **for** $k \leftarrow K$ **to** 1 **do**

3 (1) Train CNN following Algorithm 1 or CCNN following

Algorithm 2, inputs are $(\bar{A}_{ki}, \bar{X}_{ki}, V_{(k+1)i})_{i=1}^n$. Denote the

return as $f_k(\bar{A}_{(k-1)i}, \bar{X}_{ki})$, the optimal treatment regime at

stage k is:

$$d_k^*(\bar{A}_{(k-1)i}, \bar{X}_{ki}) = I\{f_k(\bar{A}_{(k-1)i}, \bar{X}_{ki}) > 0\}.$$

(2) Update value function:

$$4 \quad V_k(\bar{A}_{(k-1)i}, \bar{X}_{ki}) \leftarrow V_{k+1}(\bar{A}_{ki}, \bar{X}_{(k+1)i}) + f_k(\bar{A}_{(k-1)i}, \bar{X}_{ki}) * (d_k^*(\bar{A}_{(k-1)i}, \bar{X}_{ki}) - A_{ki}).$$

5 **end**

4 Simulation

We ran simulation studies to compare advantage learning based CCNN with existing popular methods. Our comparison is based on two stages situation. For both training and testing datasets, covariates X_1 and X_2 , randomized treatment A_1 and A_2 are generated using STAR*D data (Details of the dataset is covered in the next section). This guarantees we simulate data that is close to true distribution. The response variable Y is generated as follows:

$$y = A_1 A_2 + A_2 \sin(\beta_2^T [X_1 : X_2]) + A_1 \sin(\beta_1^T X_1) + \epsilon,$$

where random error ϵ is generated independently from normal distribution with mean zero and standard deviation 0.1. $\bar{X}_2$ is obtained by stacking new information at each stage according to chronological order, i.e. $[X_1 : X_2]$. We considered four scenarios with fixed coefficients generated from different distribution combinations:

$$\text{Case 1: } \beta_{1i} \sim N(0, 1) \forall i = 1, 2, \dots, \dim(\beta_1^T), \beta_{2j} \sim N(0, 1) \forall j = 1, 2, \dots, \dim(\beta_2^T);$$

$$\text{Case 2: } \beta_{1i} \sim U(0, 1) \forall i = 1, 2, \dots, \dim(\beta_1^T), \beta_{2j} \sim N(0, 1) \forall j = 1, 2, \dots, \dim(\beta_2^T);$$

$$\text{Case 3: } \beta_{1i} \sim N(0, 1) \forall i = 1, 2, \dots, \dim(\beta_1^T), \beta_{2j} \sim U[0, 1] \forall j = 1, 2, \dots, \dim(\beta_2^T);$$

$$\text{Case 4: } \beta_{1i} \sim U(0, 1) \forall i = 1, 2, \dots, \dim(\beta_1^T), \beta_{2j} \sim U[0, 1] \forall j = 1, 2, \dots, \dim(\beta_2^T);$$

Here $\dim(X_1)$ and $\dim([X_1 : X_2])$ are dimensions of β_1^T and β_2^T respectively and each dimension follows independent and identical distribution. U stands for uniform distribution.

We compare the results of l_1 penalized least squares (l_1 -pls). Lasso penalty is added to least square loss (4) to avoid overfitting. The objective function (5) is optimized using scikit-learn module in python (Pedregosa et al. [2011]). Shrinkage parameter λ is tuned based on grid search.

$$L(\beta) = \frac{1}{n} \sum_{i=1}^n \left\{ \frac{Y_i [A_i - \pi(X_i)]}{\pi(X_i) [1 - \pi(X_i)]} - \beta^T X_i \right\}^2, \quad (4)$$

$$\min_{\beta} L(\beta) + \lambda \sum_{j=1}^{p+1} \left| \beta_j \right|. \quad (5)$$

For both methods, we assume the propensity score is a constant and estimate it by sample mean. we ran simulation using 50 Monte Carlo data sets and reported the mean value function on testing dataset based on different estimated decision rules. Figure 1 summarized simulation results.

Compared to I_1 -pls, CCNN has better performance in terms of overall potential outcome: Value function based on CCNN is larger than that of I_1 -pls: in case 1, the value ratio for CCNN and I_1 -pls is 1.34. We also notice that the difference between two methods in stage two is less than that of stage one. There are several reasons: the optimal decision rule at stage one is more intricate than that of stage two. Therefore neural network outperforms I_1 -pls in approximation of this highly non-linear function. Optimal decision rule at stage two can be written as:

$$d_2^*(X_2, A_1) = I(A_1 + \sin(\beta_2^T [X_1; X_2]) > 0).$$

While optimal decision rule at stage one is more complicated. Another possible explanation may be optimal decision rule at stage one involves fewer covariates, resulting in less noise and more signal. Nevertheless, when the true decision function can be approximated well enough by linear regression, it is possible that I_1 -pls may achieve better results.

5 Application to STAR*D Study

In this section we demonstrate the performance of the proposed deep A-learning methods using the STAR*D clinical trial dataset. We compared mean population outcome following the estimated DTR using the deep A-learning and penalized least square estimator based on a linear decision rule.

5.1 Dataset

Sequenced Treatment Alternatives to Relieve Depression (STAR*D) is the largest and longest clinical trial to compare the effectiveness of treatments for major depressive disorder. It has four levels. Each level lasts for 12 weeks. The severity of depression is measured by Quick Inventory of Depressive Symptomatology (QIDS) score. Participants without adequate clinical response at the end of each level would continue to the next stage. For level 1 all participants received citalopram. For each level of 2 – 4, patients received one randomized treatment. Covariates are collected from enrollment, IVR call, ROA interviews, clinic visit and other events (such as suicide, non-serious adverse event and protocol deviation). See Fava et al. [2003] for design and measurement details of STAR*D study.

5.2 Processing

In general there are two types of options: switch to a different medication or adding on to their existing medication. Since all patients received the same treatment at level 1 and the number of patients who entered level 4 is too small, we only focus on the 299 patients that has complete information at level 2 and level 3. Table 1 is the list of treatment switch and treatment augmentation options at the two levels. We take the negative level 3 16-item QIDS (QIDS-C16) as response variable Y . The propensity score is assumed to be constant and is estimated using sample mean. At each level, we remove a few covariates with small variance and reshape the covariate vector to a square matrix. The inputs of CNN/CCNN for level 2 is a 17×17 matrix and a 19×19 matrix for level 3.

Local normalization and Zero Components Analysis (ZCA-whitening) are incorporated for the input data. ZCA-whitening proposed by Krizhevsky and Hinton [2009] is a popular technique for pre-processing of CNN. It can preserve local properties. ZCA-whitening can remove correlation between nearby covariates. ZCA is summarized in (6) where Σ is the covariance matrix, and ϵ is the regularization parameter to avoid numerically unstable situations.

$$\Sigma = P \begin{pmatrix} s_1 & & & \\ & s_2 & & \\ & & \dots & \\ & & & s_r \end{pmatrix} P^T, \quad (6)$$

$$W = P \begin{pmatrix} \frac{1}{\sqrt{s_1 + \epsilon}} & & & \\ & \frac{1}{\sqrt{s_2 + \epsilon}} & & \\ & & \dots & \\ & & & \frac{1}{\sqrt{s_r + \epsilon}} \end{pmatrix} P^T,$$

$$X_{transform} = XW$$

5.3 Architecture

The architecture of 2-layer CNN is as follows: the convolutional layer is of size 3×3 for stage 2 and 5×5 for stage 3 since number of covariates at level 3 is larger. The average pooling is implemented in the pooling layer with pooling patch = 2 and pooling stride = 2. We do not use any padding techniques in order to control overfitting. Then a fully-connected layer maps all neurons to a scalar. For m-layer CNN, it has m-1 convolutional layers followed by one fully-connected layer. The number of filters is tuned by IPWE.

The architecture of CCNN is very similar to CNN except that pooling is taken place before convolution operation. This can reduce the number of parameters in the neural network. The number of dimension of random feature approximation, number of channels, scale parameter of Gaussian kernel and nuclear norm constraints are tuned by cross-validation. Both CNN and CCNN are trained using mini-batch gradient descent. The learning rate of CNN and CCNN are $2e-3$ and $2e-4$, respectively. CNN is implemented using tensorflow (Abadi et al. [2015]).

5.4 Results

We compare the results based on neural network with I_1 -pls. Shrinkage parameter λ is tuned based on BIC. To measure the performance of estimated DTR, we split the data into 3 parts: 60% of data set to be training data, 20% as validation and 20% as testing. Since parameters are randomly initialized, the accuracy of DTR varies each time, we trained each architecture 100 times using the training dataset and choose the best network based on the maximum value function estimation $\hat{V}_{IPW}$

$$\hat{V}_{IPW} = \frac{1}{n} \sum_{i=1}^n \frac{y_i I\{A_{1,i} = d_1^*(X_{1i}), A_{2,i} = d_2^*(\bar{X}_{2i}, A_{1i})\}}{(1 - \pi_1(X_{1i}))^{1 - A_{1,i}} \pi_1(X_{1i})^{A_{1,i}} (1 - \pi_2(\bar{X}_{2i}, A_{1i}))^{1 - A_{2,i}} \pi_2(\bar{X}_{2i}, A_{1i})^{A_{2,i}}}$$

on the validation dataset. We report the value function estimation on the testing dataset.

Table 2 summarized results of different architectures. Based on the results, we have the following observations: Both CNN and CCNN can learn a dynamic treatment regime that outperforms I_1 -pls. They use intricate functions to learn a decision rule. It is much accurate than penalized least square estimator. For fixed number of layers, CCNN has better performance than CNN. The performance of both architectures improve as the depth of neural network increases. We also notice that processing has a great influence on decision accuracy. For example, without zca-whitening, both CNN and CCNN would have smaller estimated value function.

6 Conclusion

In this paper, we propose a deep A-learning that integrates CNN and CCNN with A-learning for optimal treatment regime estimation. Here our contrast function is estimated based on IPW estimator. The new methods have the following advantages: It uses intricate functions to learn decision rules, which, allows for model complexity. The parameter sharing

mechanism makes it computational efficient and controls overfitting. This method can be used in situations where medical images are available. The results indicate that the deep A-learning is competitive and outperforms penalized least square estimator. Although deep neural network has shown its competency in intricate scenarios, linear approximation based Q-learning or A-learning may still lead to a more accurate and computational-efficient solution when the true decision rule is simple.

In the future, more sophisticated algorithms such as RMSProp and Adagrad can be implemented to better estimate parameters in the neural network. Adaptive methods for hyperparameter specification is also worth studying. Recently complicated architectures which consist of multiple subnetworks have demonstrated themselves to be very powerful. Another interesting future work is to implement those architectures to see whether better performance can be achieved. Last but not least, the performance of neural network would improve if training sample size increases. In computer vision, techniques like rotation and random cropping are widely used for data augmentation. It may be interesting to see whether those techniques can help further improve the performance of optimal DTR estimation.

References

- Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, Corrado GS, Davis A, Dean J, Devin M, Ghemawat S, Goodfellow I, Harp A, Irving G, Isard M, Jia Y, Jozefowicz R, Kaiser L, Kudlur M, Levenberg J, Mané D, Monga R, Moore S, Murray D, Olah C, Schuster M, Shlens J, Steiner B, Sutskever I, Talwar K, Tucker P, Vanhoucke V, Vasudevan V, Viégas F, Vinyals O, Warden P, Wattenberg M, Wicke M, Yu Y, and Zheng X. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015 URL <http://tensorflow.org/>. Software available from tensorflow.org.
- Basu D. Randomization analysis of experimental data: the fisher randomization test. *Journal of the American Statistical Association*, 75(371): 575–582, 1980.
- Bojarski M, Del Testa D, Dworakowski D, Firner B, Flepp B, Goyal P, Jackel LD, Monfort M, Muller U, Zhang J, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.
- Choromanska A, Hena M, Mathieu M, Arous GB, and LeCun Y. The loss surfaces of multilayer networks. In *AISTATS*, volume 38, pages 192–204, 2015.
- Collobert R and Weston J. A unified architecture for natural language processing: Deep neural networks with multitask learning In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, 2008.
- Cowell RG, Dawid P, Lauritzen SL, and Spiegelhalter DJ. Probabilistic networks and expert systems: Exact computational methods for Bayesian networks. Springer Science & Business Media, 2006.
- Ding X, Zhang Y, Liu T, and Duan J. Deep learning for event-driven stock prediction. In *IJCAI*, pages 2327–2333, 2015.
- Duchi J, Shalev-Shwartz S, Singer Y, and Chandra T. Efficient projections onto the l_1 -ball for learning in high dimensions In *Proceedings of the 25th international conference on Machine learning*, pages 272–279. ACM, 2008.
- Fan J and Li R. Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American statistical Association*, 96 (456):1348–1360, 2001.
- Fava M, Rush J, Trivedi MH, Nierenberg A, Thase M, Sackeim F, and Quitkin H, Wisniewski S, Lavori P, Rosenbaum J, et al. Background and rationale for the sequenced treatment alternatives to relieve depression (star d) study. *Psychiatric Clinics of North America*, 26(2): 457–494, 2003. [PubMed: 12778843]
- Hornik K, Stinchcombe M, and White H. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.
- Karpathy A. Lecture notes in cs231n: Convolutional neural networks for visual recognition., Spring 2017.

- Krizhevsky A and Hinton G. Learning multiple layers of features from tiny images. MSc thesis, 2009.
- Krizhevsky A, Sutskever I, and Hinton GE. Imagenet classification with deep convolutional neural networks In *Advances in neural information processing systems*, pages 1097–1105, 2012.
- LeCun Y, Bottou L, Bengio Y, and Hinton G. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- LeCun Y, Bengio Y, and Hinton G. Deep learning. *Nature*, 521(7553): 436–444, 2015. [PubMed: 26017442]
- Lenz I, Lee H, and Saxena A. Deep learning for detecting robotic grasps. *The International Journal of Robotics Research*, 34(4–5):705–724, 2015.
- Lu W, Zhang H, and Zeng D. Variable selection for optimal treatment decision. *Statistical methods in medical research*, 22(5):493–504, 2013. [PubMed: 22116341]
- Mnih V, Kavukcuoglu K, Silver D, Graves A, Antonoglou I, Wierstra D, and Riedmiller M. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.
- Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, Graves A, Riedmiller M, Fidjeland AK, Ostrovski G, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529–533, 2015. [PubMed: 25719670]
- Moodie EE, Richardson TS, and Stephens DA. Demystifying optimal dynamic treatment regimes. *Biometrics*, 63(2):447–455, 2007. [PubMed: 17688497]
- Murphy SA. Optimal dynamic treatment regimes. *Journal of the Royal Statistical Society: Series B*, 65(2):331–355, 2003.
- Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, and Duchesnay E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- Pinkus A. Approximation theory of the mlp model in neural networks. *Acta Numerica*, 8:143–195, 1999.
- Qian M and Murphy S. Performance guarantees for individualized treatment rules. *Annals of statistics*, 39(2):1180–1210, 2011. [PubMed: 21666835]
- Rahimi A and Recht B. Random features for large-scale kernel machines In *Advances in Neural Information Processing Systems 20*, pages 1–8. Curran Associates, 2008.
- Robins J. Causal inference from complex longitudinal data In *Latent variable modeling and applications to causality*. Springer, 1997.
- Schulte PJ, Tsiatis AA, Laber EB, and Davidian M. Q-and a-learning methods for estimating optimal dynamic treatment regimes. *Statistical science*, 29(4):640–661, 2014. [PubMed: 25620840]
- Shi C, Song R, Lu W, et al. Robust learning for optimal treatment decision with np-dimensionality. *Electronic Journal of Statistics*, 10(2):2894–2921, 2016. [PubMed: 28781717]
- Shi C, Fan A, Song R, and Lu W. High-dimensional a-learning for optimal dynamic treatment regimes. *Annals of Statistics*, 4(1):59–68, 2017.
- Silver D, Huang A, Maddison CJ, Guez A, Sifre L, Van Den Driessche G, Schrittwieser J, Antonoglou I, Panneershelvam V, Lanctot M, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016. [PubMed: 26819042]
- Song R, Kosorok M, Zeng D, Zhao Y, Laber E, and Yuan M. On sparse representation for optimal individualized treatment selection with penalized outcome weighted learning. *Stat*, 4(1):59–68, 2015. [PubMed: 25883393]
- Tibshirani R. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B*, 58(1):267–288, 1996.
- Watkins CJ and Dayan P. Q-learning. *Machine learning*, 8(3–4):279–292, 1992.
- Watkins CJCH. Learning from delayed rewards PhD thesis, University of Cambridge England, 1989.
- Zhang B, Tsiatis AA, Davidian M, Zhang M, and Laber E. Estimating optimal treatment regimes from a classification perspective. *Stat*, 1(1): 103–114, 2012. [PubMed: 23645940]
- Zhang Y, Liang P, and Wainwright MJ. Convexified convolutional neural networks. *arXiv preprint arXiv:1609.01000*, 2016.

Zhao Y, Zeng D, Rush J, and Kosorok M. Estimating individualized treatment rules using outcome weighted learning. *Journal of the American Statistical Association*, 107(499):1106–1118, 2012. [PubMed: 23630406]

Author Manuscript

Author Manuscript

Author Manuscript

Author Manuscript

Figure 1:
Value function given patients received estimated optimal treatments for both stages

Table 1:

Lists of *STAR*D* treatment options at level 2 and level 3

Type	Level	Treatment
Switch	2	Bupropion SR , Sertraline , Venlafaxine XR
Switch	3	Citalopram plus Bupropion SR, Citalopram plus Buspirone
Augmentation	2	Nortriptyline, Mirtazapine
Augmentation	3	Lithium augmentation, Triiodothyronine augmentation

Table 2:

Evaluation results for estimated values on STAR*D dataset

Method	Number of layers	Value function estimation
CCNN	2	-7.22
CCNN	3	-4.57
CNN	2	-7.58
CNN	3	-5.92
CNN	4	-4.16
l_1 -pls	-	-8.38