Precision Medicine Workshop: Applications of Deep Learning and Inverse Reinforcement Learning to Precision Medicine

Michael R. Kosorok, Nikki L. B. Freeman and Owen E. Leete

Department of Biostatistics
Gillings School of Global Public Health
University of North Carolina at Chapel Hill

October 7, 2020

Outline

Topics Covered

- Chapter 1: Deep Learning Overview
- Chapter 2: Applications of Deep Learning to Precision Medicine
- Chapter 3: Multiple Utilities and Inverse Reinforcement Learning

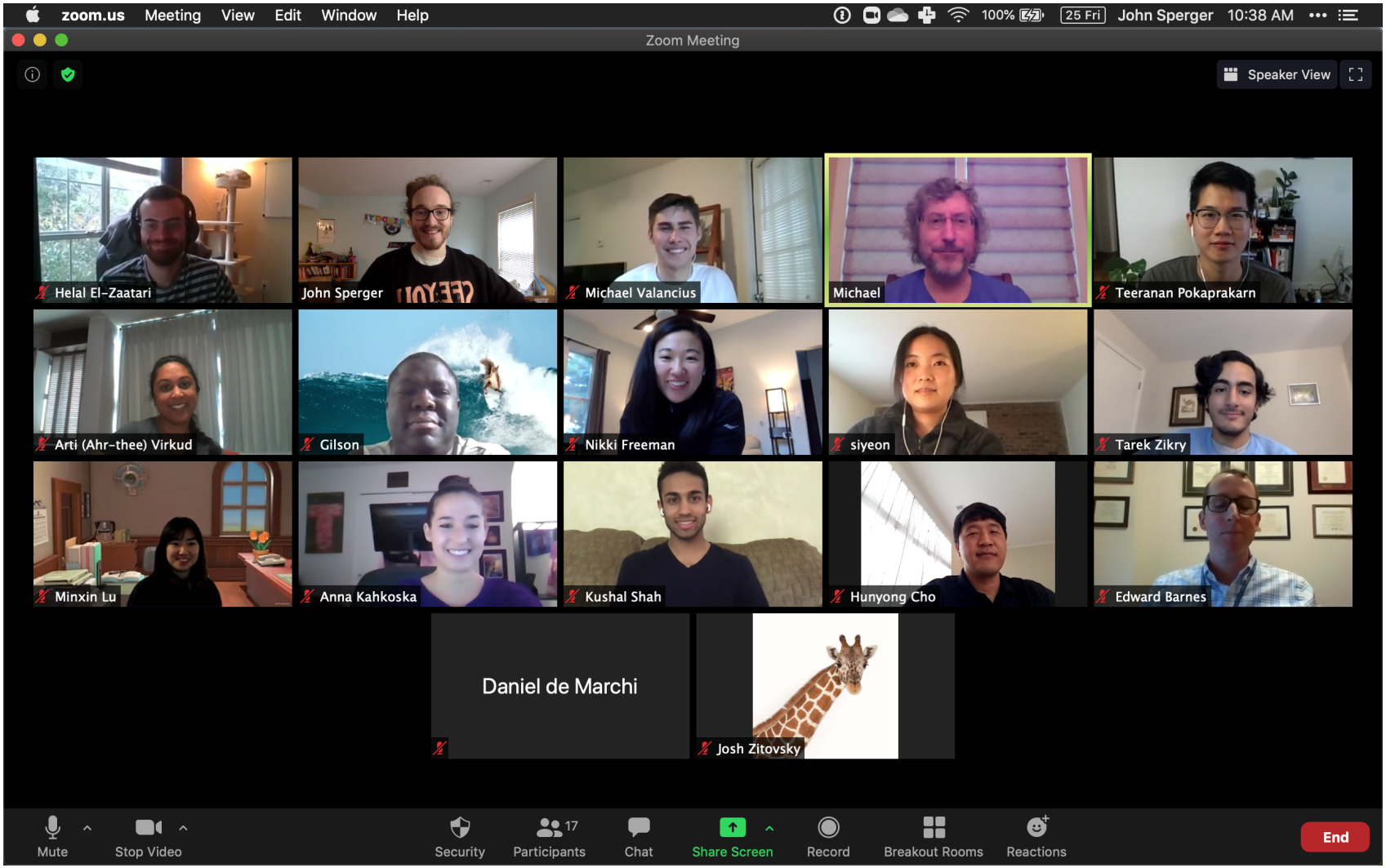
General References

- LeCun Y, Bengio Y, and Hinton G (2015). Deep learning. *Nature* 521:436-444.
- Kosorok MR and Laber EB (2019). Precision medicine. *Annual Reviews of Statistics and Its Application* 6:263-286.
- UNC Biostatistics 740 (Precision Medicine) course materials: `mkosorok.web.unc.edu/teaching-and-mentoring/`

Chapter References

- Buttler EL, Laber EB, Davis SM, and Kosorok MR (2018). Incorporating patient preferences into estimation of optimal dynamic treatment rules. *Biometrics* 74:18-26.
- Goudet O, Kalainathan D, Callou P, Guyon I, Lopez-Paz D, and Sebag M (2018). Causal generative neural networks. `arXiv:1711.08936v2 [stat.ML]`
- Liang M, Ye T, and Fu H (2018). Estimating individualized optimal combination therapies through outcome weighted deep learning algorithms. *Statistics in Medicine* 37:3869-3886.
- Luckett DJ, Laber EB, and Kosorok MR (2019+). Estimation and optimization of composite outcomes. `arXiv:1711.10581v4 [stat.ML]`
- Wang L, Laber EB, and Witkiewitz K (2018). Sufficient Markov decision processes with alternating deep neural networks. `arXiv:1704.07531v2 [stat.ME]`

Kosorok Precision Health and Artificial Intelligence Research Lab

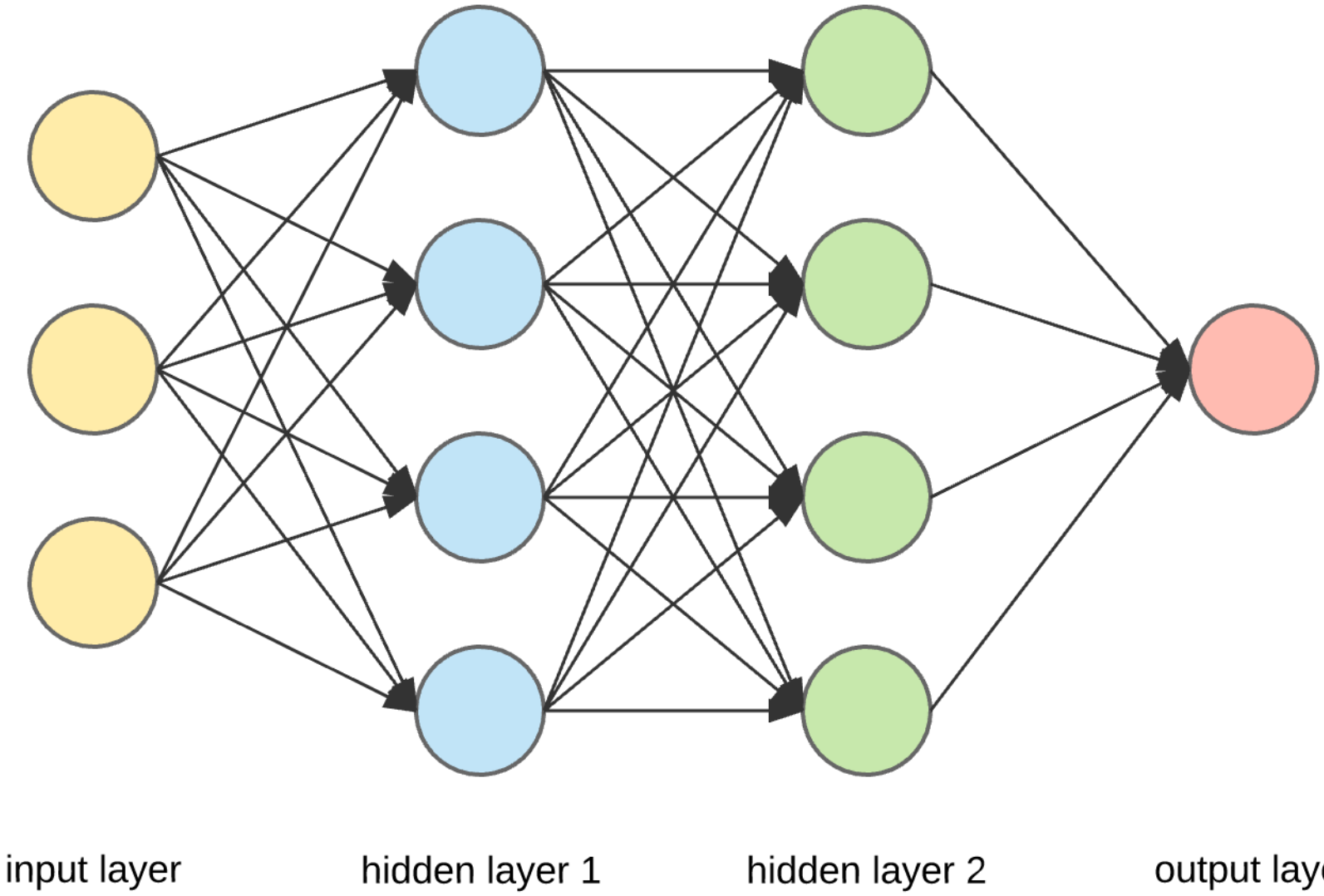


Outline

Artificial Neural Networks

- Artificial neural networks (ANN) are machine learning methods inspired by how neurons work in the brain
- ANNs are based on a collection of connected units or nodes called artificial neurons
- ANNs are mathematical functions of varying complexity that map a set of input values to output values
- ANNs are flexible models that can be used with many different types of input and output values
- By connecting the artificial neurons in different ways ANNs have been adapted to a wide variety of tasks

Artificial Neural Networks



Deep Learning

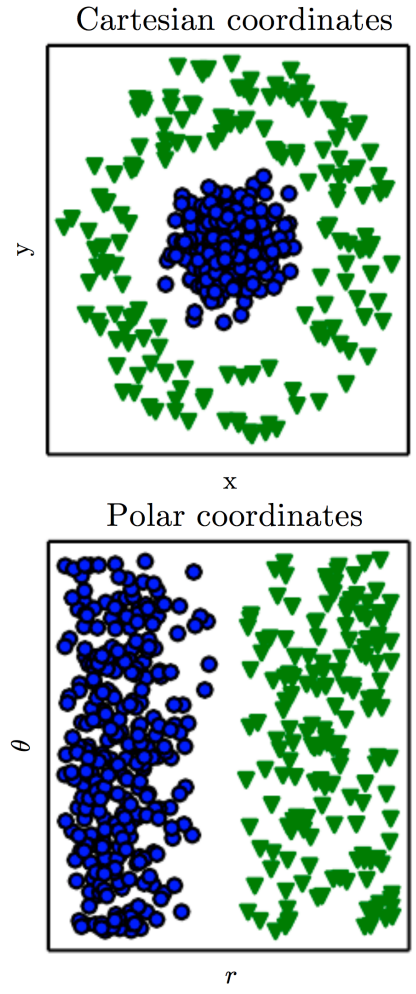
- Deep learning is a class of methods based on artificial neural networks
- The "deep" in deep learning refers to the number of hidden layers in an ANN
- A larger number of hidden layers allows deep neural networks to produce extremely intricate functions of its inputs
- Deep learning models can be simultaneously sensitive to minute details, but insensitive to large irrelevant changes

Feature Engineering

- Pattern-recognition and machine-learning systems have historically relied on carefully engineered features to extract useful representations from the raw data
- Engineered features are common in many applications
  - Example: BMI = (weight in kg)/(height in m)^2
- In 2013, Andrew Ng said:
  Coming up with features is difficult, time-consuming, requires expert knowledge. "Applied machine learning" is basically feature engineering.
- Deep learning essentially automates the feature engineering process

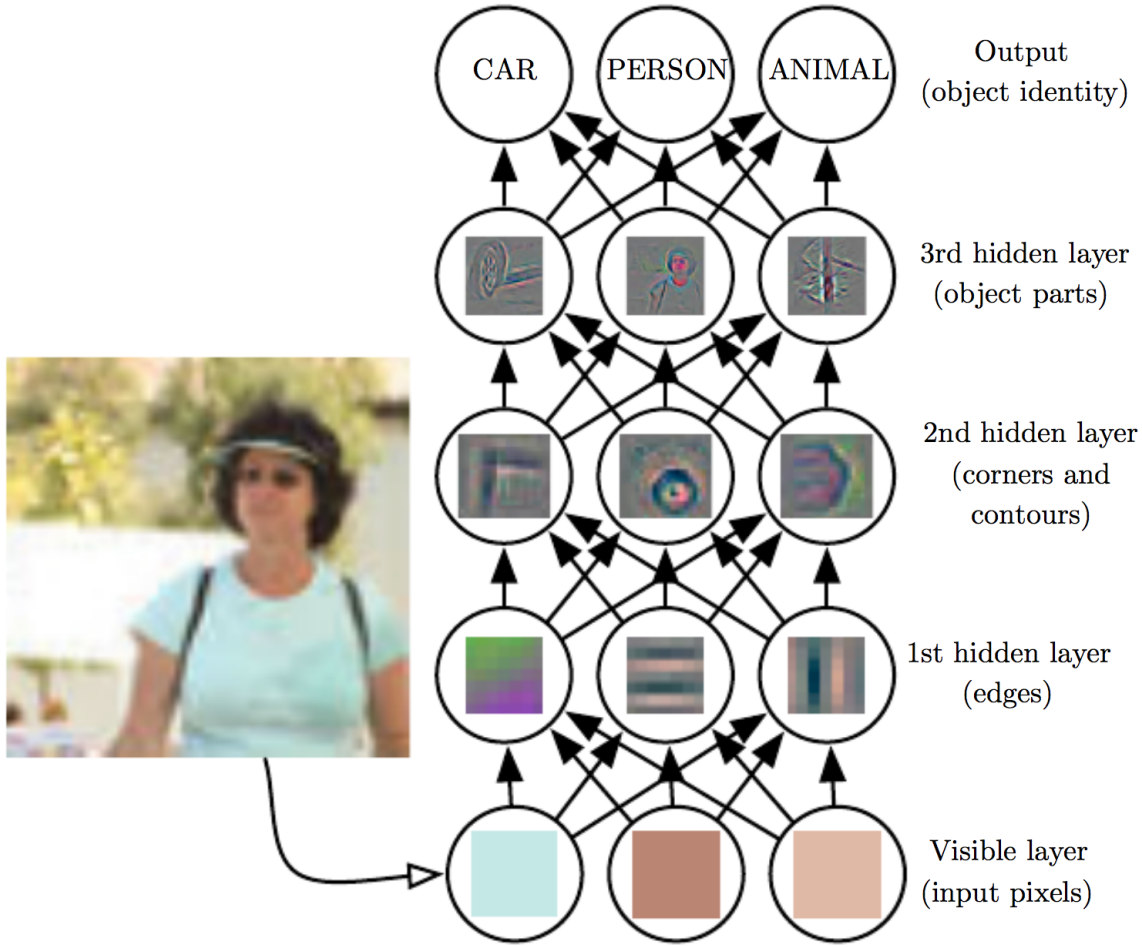
Representation learning

- Representation learning is a set of methods that allow ML algorithms to automatically discover representations of the data that make detection and classification easier
- Deep learning methods develop multiple levels of representation by compositing several simple non-linear transformations



Source: Goodfellow et al, 2016

Representation learning



ANN Origins — Perceptrons

- In 1958 Frank Rosenblatt described a binary classifier called the perceptron algorithm
- Given a d-dimensional vector of covariates x_i, the class of the observation is predicted according to the function

$$f(\mathrm{x}) = \begin{cases} 1 & \text{if } \sum_{i=1}^{d} w_i x_i + b > 0, \\ 0 & \text{otherwise} \end{cases}$$

  where w is a vector of real-valued weights
- Perceptrons are an early form of linear classification
- ANNs are sometimes referred to as multi-layer perceptrons

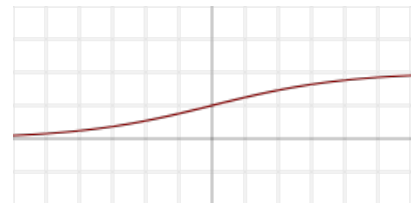
Activation Functions

- Each layer in an ANN is composed of a linear combination of the node values from the previous layer
- Applying a non-linear activation function to the linear combinations allows successive layers to learn increasingly complex features
- While selecting a model, it is common to test many different activation functions and find that several perform comparably
- There are some situations where the choice of activation functions can greatly impact the performance of ANNs

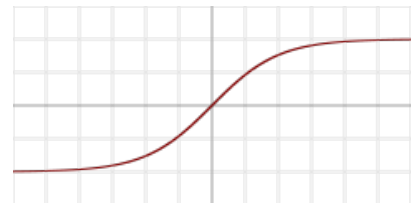
Activation Functions

- Several activation functions have been published, but it is likely that most remain unpublished
- Some of the most common activation functions are:

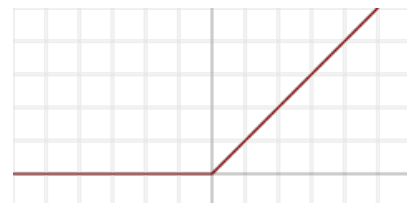
Logistic $$g(x) = \frac{1}{1 + e^{-x}}$$



TanH $$g(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$



ReLU $$g(x) = \begin{cases} 0 & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases}$$



Architecture Design

- A key design consideration for neural networks is determining the architecture
- Architecture refers to the overall structure of the network
  - How many layers
  - How many units in each layer
  - How should these units be connected to each other
  - Which activation functions to use
- Many ANNs use a chain based architecture
  - The first layer is given by

$$\mathrm{h}^{(1)} = g^{(1)}\left(\mathrm{W}^{(1)T}\mathrm{x} + \mathrm{b}^{(1)}\right)$$

  - Subsequent layers are given by

$$\mathrm{h}^{(j)} = g^{(j)}\left(\mathrm{W}^{(j)T}\mathrm{h}^{(j-1)} + \mathrm{b}^{(1)}\right)$$

Output Units

- ANNs can be used for a variety of different learning tasks by changing the output units
- Let h be the features from the final hidden layer
- Linear Units for Continuous Output Distributions
  - The output units produces a vector $\hat{y} = W^T h + b$
  - Linear output layers are often used to produce the mean of a conditional Gaussian distribution:

$$p(y \mid x) = \mathcal{N}(y; \hat{y}, I)$$

- Sigmoid Units for Bernoulli Output Distributions

$$\hat{y} = \frac{\exp\{w^T h + b\}}{1 + \exp\{w^T h + b\}}$$

Output Units, cont.

- Softmax Units for Multinoulli Output Distributions
  - A linear layer predicts unnormalized log (relative) probabilities

$$\mathrm{z} = \mathrm{W}^T \mathrm{h} + \mathrm{b}$$

    where $z_i = \log P(y = i \mid \mathrm{x})$
  - The softmax function can normalize z to obtain the desired $\hat{y}$

$$\mathrm{softmax}(\mathrm{z}) = \frac{\exp\{z_i\}}{\sum_j \exp\{z_j\}}$$

- There are many other output units that can return images, sound, video, etc.

Training via Backpropagation

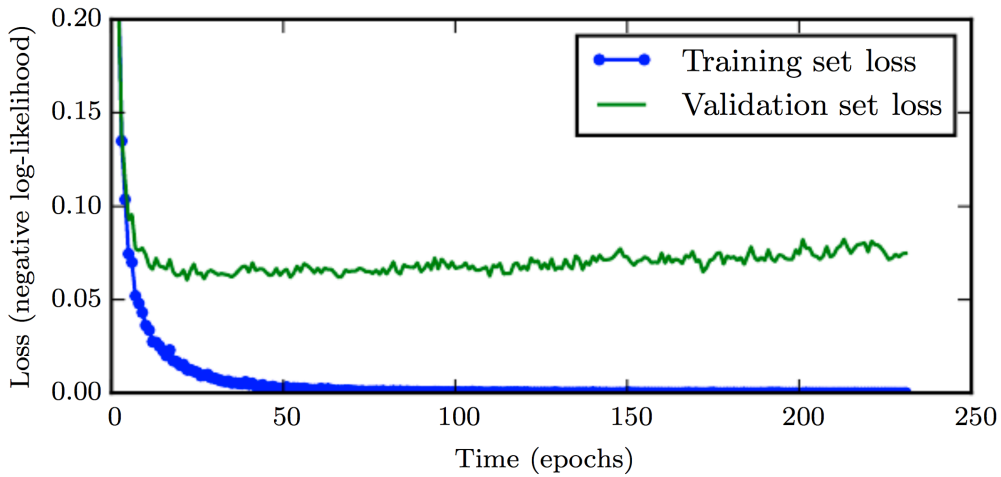
- Multi-layer architectures can be trained by gradient descent
- If the nodes are relatively smooth functions of the inputs, the gradients can be calculated using the backpropagation procedure
- For a given loss function we can determine how the weights in the final layer need to change to lower the loss
- Repeated application of the chain rule allows us to determine how weights in previous layers need to change
- Some activation functions are not differentiable at all points (e.g. ReLU), but they can still be used with gradient-based learning algorithms at all input points.

Regularization

- DL models typically have a large number of parameters, sometimes more parameters than training examples
- Regularization methods are required to prevent overfitting
- L^1 and L^2 norms can be applied to the weights for each node, but this is uncommon in DL
- Ensembles of neural networks with different model configurations are known to reduce overfitting
  - It is impractical to have an ensemble of multiple large neural networks
  - A single model can be used to simulate having a large number of different network architectures by randomly dropping out nodes during training
  - Dropout is a computationally efficient and remarkably effective method to approximate an ensemble approach

Regularization

- One of the most common regularization methods used for ANNs is early stopping
- The training error almost always decreases, but validation error tends to increases with excessive training
- A model with small validation error can be found buy stopping the training process early

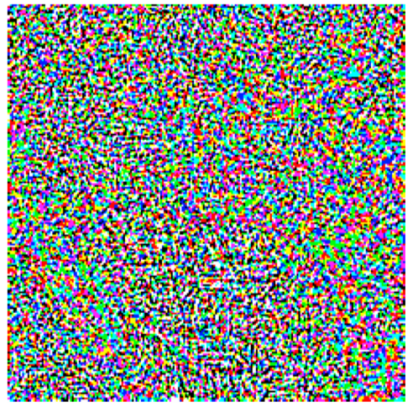


Adversarial Examples

- Adversarial examples are samples of input data which are designed/selected to cause a machine learning classifier to misclassify it
- Adversarial examples can be used while training to make a DL model more robust
  - Samples with noise added can make the predictions less sensitive to small differences
  - Exposing a model to samples known to lie close to the decision boundary can improve performance
- Adversarial examples have important implications for the safety of certain applications (e.g. self driving cars)

Adversarial examples

$+ .007 \times$



$=$

$\boldsymbol{x}$

"panda"
57.7% confidence

$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$

"nematode"
8.2% confidence

$\boldsymbol{x} +$
$\epsilon \text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"gibbon"
99.3 % confidence

- By adding a imperceptible amount of noise, the classification of the image can be changed

Adversarial examples

These examples are likely close to the decision boundary

Convolutional Neural Networks

- Convolutional Neural Networks (CNNs) are designed to process data that come in the form of multiple arrays
- CNNs are used in many applications such as: image and video recognition, recommender systems, image classification, medical image analysis, and natural language processing
- The few layers of a typical CNN is composed of two types of layers
  - Convolutional layers
  - Pooling layers

Convolution

- A convolution is an operation on two functions of a real-valued argument
- Convolutions are used to look at localized areas of an array

$$s(t) = \int x(a)w(t-a)\,da$$

- The convolution operation is typically denoted with an asterisk

$$s(t) = (x * w)(t)$$

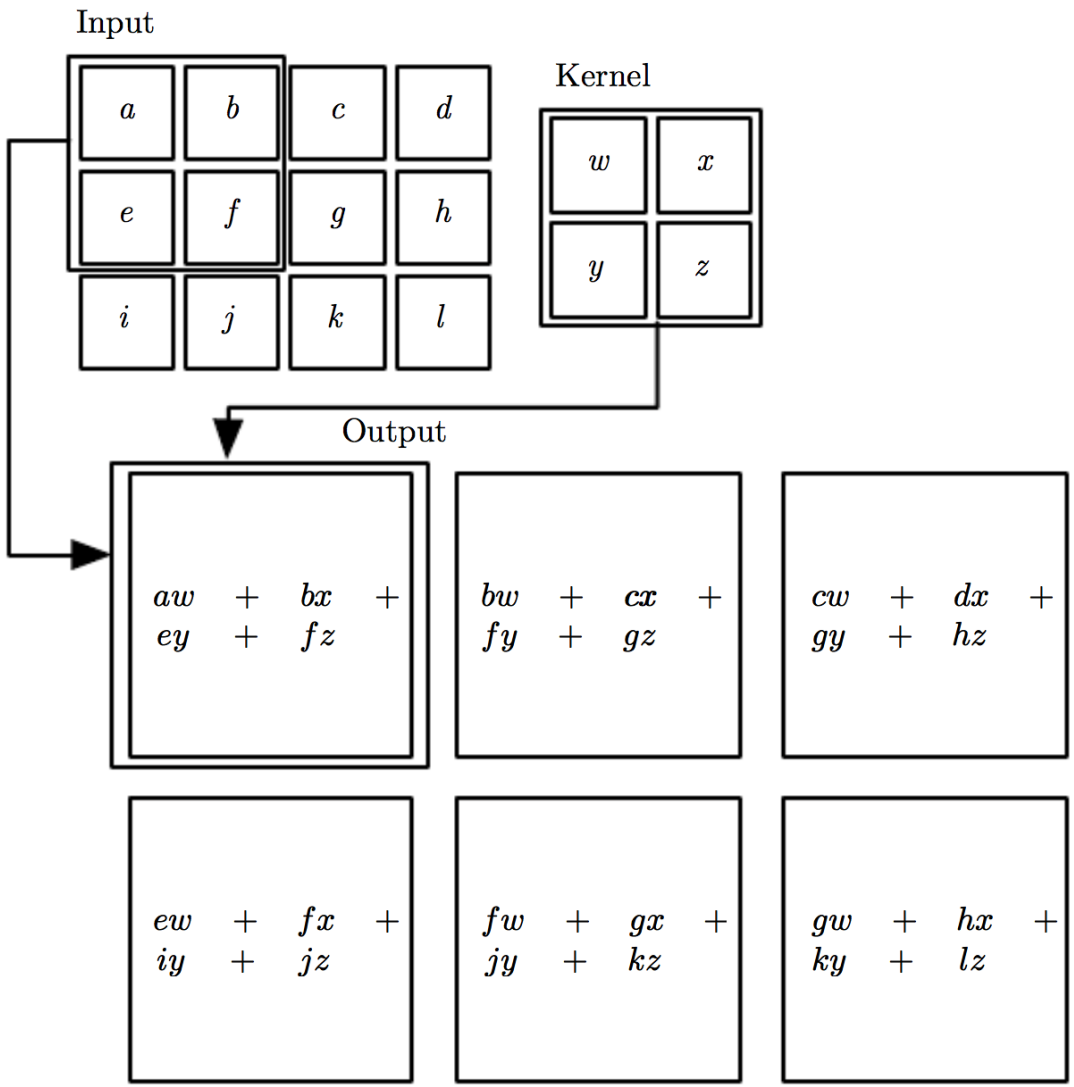
Convolution

- Convolutions are often used over more than one axis at a time
- For a d-dimensional input, convolutions can be calculated with a d-dimensional kernel K
- For an $m \times n$ image as input, we can write the convolution as

$$S(i,j) = (X * K)(i,j) = \sum_m \sum_n X(m,n) K(i-m, j-n)$$

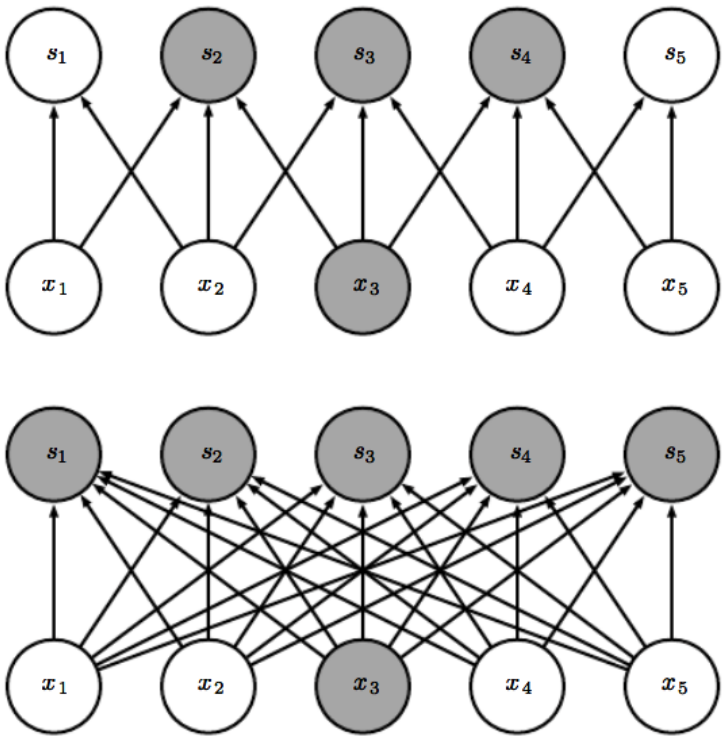
- Discrete convolution can be viewed as multiplication by a matrix, where the matrix has several entries constrained to be equal

Convolution Layer



Source: Goodfellow et al, 2016

Local Connectivity



Source: Goodfellow et al, 2016

- Unlike other ANNs, CNNs have layers that are not fully connected
- Convolutional layers have local connections
- For example, an input image might have thousands or millions of pixels, but meaningful features usually occupy only tens or hundreds of pixels

Parameter Sharing

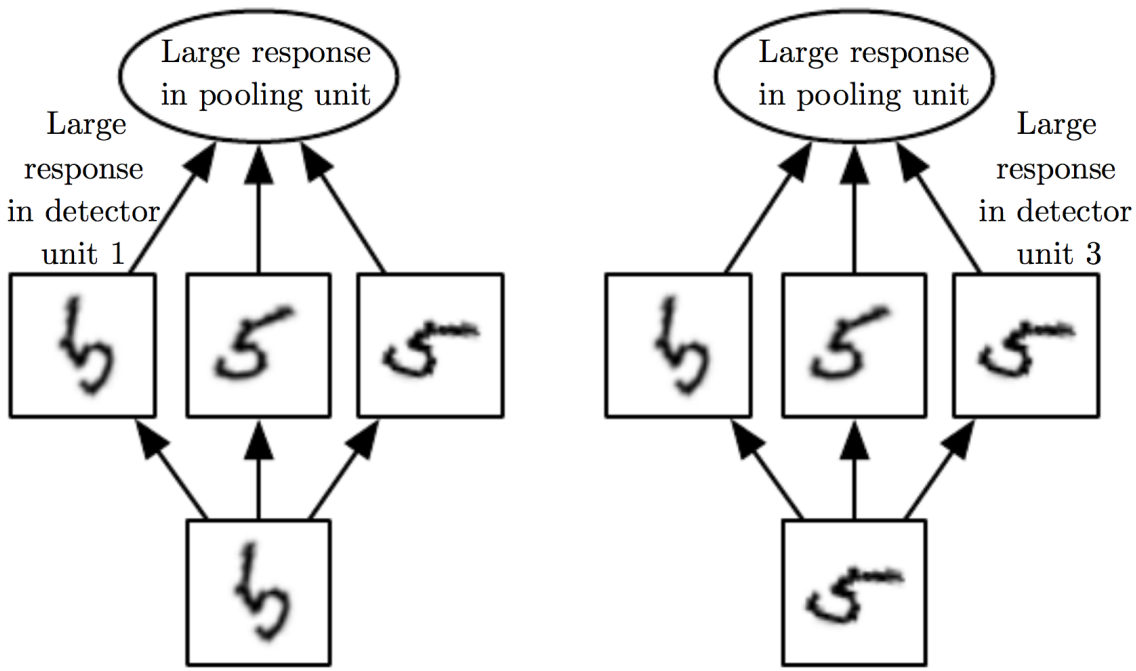
- In a convolutional neural net, each member of the kernel is used at every position of the input
- The parameter sharing used by the convolution operation means that rather than learning a separate set of parameters for every location, we learn only one set
- Parameter sharing causes a layer to have a property called equivariance to translation
  - Features can be identified regardless of where they occur in an image
- Both local connectivity and parameter sharing can greatly reduce the number of parameters needed compared to a similarly sized traditional neural network

Pooling

- A pooling function replaces the output of the net at a certain location with a summary statistic of the nearby outputs
  - Example: Max pooling operation reports the maximum output within a rectangular neighborhood
- Pooling over spatial regions can help to make the representation approximately invariant to small translations of the input
- The feature generation process can learn which transformations to become invariant to by pooling over the outputs of a range of parameterized convolutions

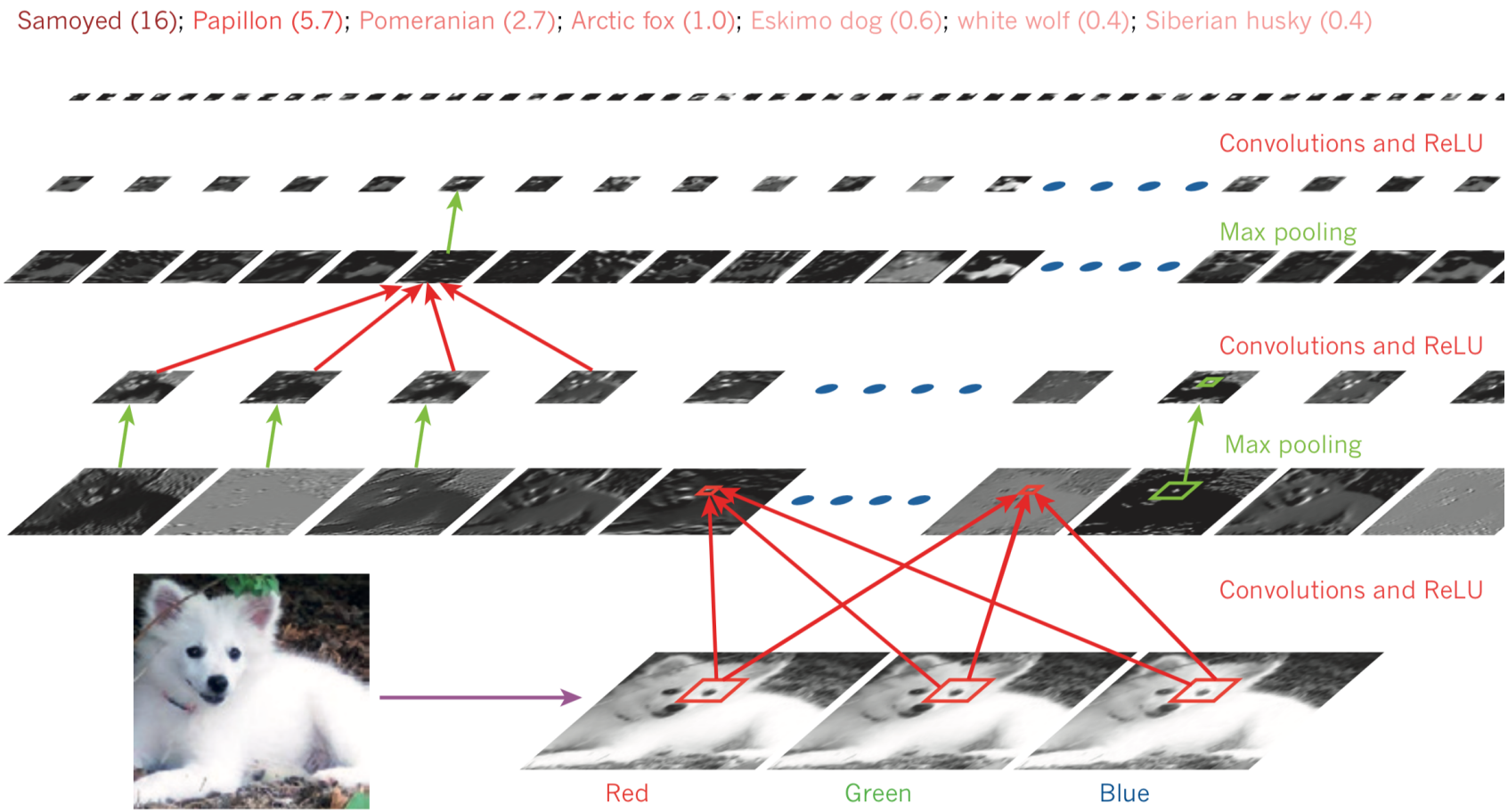
Pooling

- Example: All three filters are intended to detect a hand written 5
- Each filter attempts to match a slightly different orientation of the 5



Source: Goodfellow et al, 2016

Example of CNN Architecture



Recurrent Neural Networks

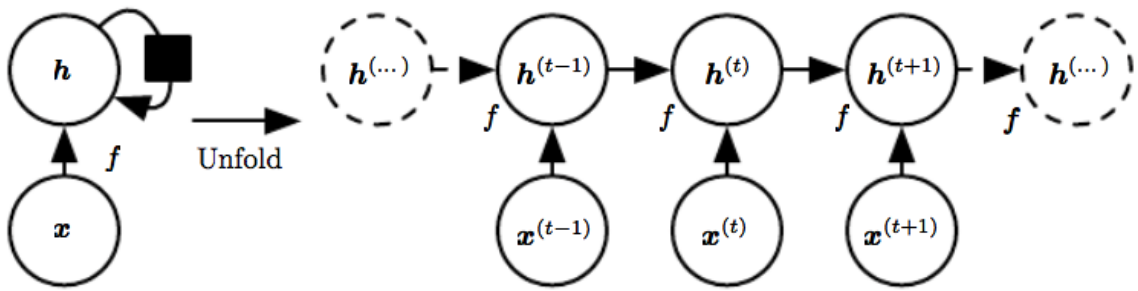
- Recurrent neural networks (RNNs) are a family of neural networks for processing sequential data
- RNNs process an input sequence one element at a time, maintaining in their hidden units a 'state vector' that contains information about the history of the sequence
- Most RNNs can process sequences of variable length, and can scale to much longer sequences than would be practical for networks without sequence-based specialization
  - Both of these qualities are largely due to parameter sharing

Unfolding Computational Graphs

- A computational graph is a way to formalize the structure of a set of computations
- Consider a dynamical system where the state at time t is $\mathrm{h}^{(t)}$. The system depends on a function f, parameters θ, and is driven by an external signal $\mathrm{x}^{(t)}$

$$\begin{aligned}\mathrm{h}^{(t+1)} &= f(\mathrm{h}^{(t)}, x^{(t)}; \theta) \\ &= f(f(\ldots f(\mathrm{h}^{(1)}, x^{(1)}; \theta), \ldots, x^{(t-1)}; \theta), x^{(t)}; \theta)\end{aligned}$$

- This system can be represented using the graphical model



Source: Goodfellow et al, 2016

Unfolding Computational Graphs

- RNNs can be described as a computational graph that has a recurrent structure
- A recurrent computational graph can be unfolded to a computational graph with a repetitive structure
- Complex models can be succinctly represented with a recurrent graph
- The unfolded graph provides an explicit description of which computations to perform

Recurrent Neural Networks

- RNNs learn a single shared model and apply the same set of computations at each time step
- A shared model allows generalization to sequence lengths that did not appear in the training set, and needs far fewer training examples than would be required without parameter sharing
- RNNs can output a result at each time step (stock market predictions) or read an entire sequence before outputting a result (meaning of a sentence)

Bidirectional RNNs

- RNNs need not have a causal structure. In many applications we want to output a prediction that may depend on the whole input sequence
- For example, in natural language processing, the meaning of a word might require the context of nearby words in both directions
- Bidirectional RNNs are composed of two RNNs: one that moves forward through time from the start of the sequence, and another that moves backward through time from the end of the sequence

The Challenge of Long-Term Dependencies

- Long-Term dependencies are difficult to model because gradients propagated over many stages tend to either vanish or explode
- There have been attempts to avoid the problem by staying in a region of the parameter space where the gradients do not vanish or explode
- Unfortunately, in order to store memories in a way that is robust to small perturbations, the RNN must enter a region of parameter space where gradients vanish
- Even if the parameters are such that the recurrent network is stable, long-term interactions have exponentially smaller weights compared to short-term interactions

Skip Connections and Leaky Units

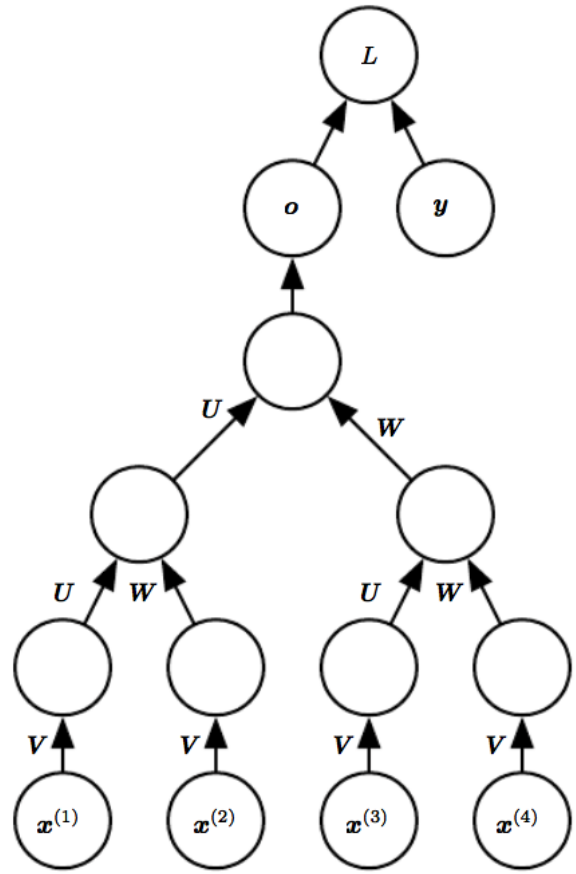
- Skip connections obtain coarse time scales by adding direct connections from variables in the distant past to variables in the present
  - In ordinary recurrent networks, a recurrent connection goes from a unit at time t to a unit at time $t+1$, but longer connections are possible $(t+d)$
  - For τ time steps, gradients now diminish exponentially as a function of τ/d rather than τ
- Leaky Units have linear self-connections that "remember" past values
  - Leaky units accumulate a running average $\mu^{(t)}$ of some value $v^{(t)}$ by applying the update $\mu^{(t)} = \alpha\mu^{(t-1)} + (1-\alpha)v^{(t)}$
  - When α is near one, the leaky unit remembers information about the past for a long time, and when α is near zero, information about the past is rapidly discarded

Long Short-Term Memory Nodes

- Leaky units use self-connections to accumulate information, but there is no mechanism to "forget" old information even when it would be beneficial to do so
- Long Short-Term Memory units have several "gates" to control how the unit behaves at each time step
  - Input gate: Controls when the node gets updated
  - Forget gate: Controls how long information is retained
  - output gate: Controls when the node has an output value
- Each gate has parameters controlling its behavior allowing the model to learn when each behavior is beneficial

Recursive Neural Networks

- Recursive neural networks are a generalization of recurrent networks, with a computational graph which is structured as a tree
- For a sequence of the same length, the number of compositions of nonlinear operations is smaller for recursive neural networks than RNNs which might help deal with long-term dependencies



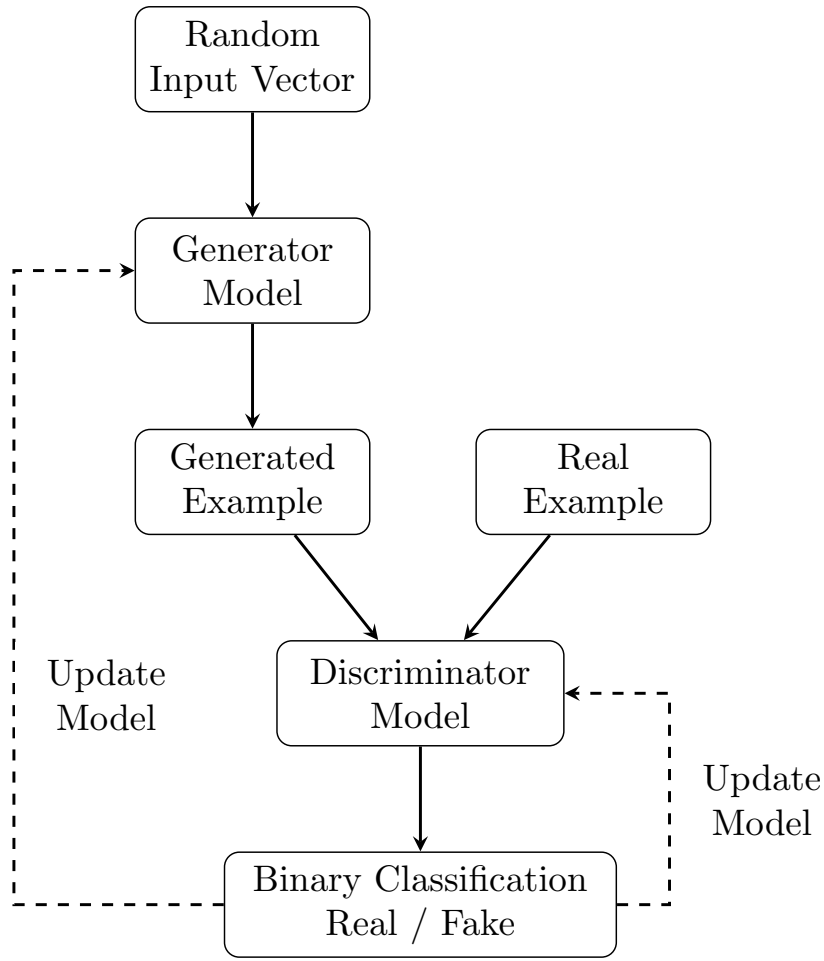
Source: Goodfellow et al, 2016

Generative Modeling

- Generative modeling is an unsupervised learning task
- A generative model is used to generate new examples that could have been drawn from the original data distribution
- Generative adversarial networks (GANs) are a way of training a generative model by framing it as a supervised learning problem with two sub-models
  - A generative network which learns to map from a latent space to a data distribution of interest
  - A discriminative network which distinguishes candidates produced by the generator from the true data distribution

Generative Adversarial Networks

- The generator model "learns" the data distribution by competing with the discriminator model
- Both the generator and discriminator models are updated to improve their performance
- Training continues until the discriminator is consistently "fooled" 50% of the time



GAN Progress

- GANs have made considerable progress in recent years
- Image generators can fool both discriminator networks and human observers, which misclassify up to 40 percent of generated images

GAN Applications

- GANs are useful for their ability to represent high-dimensional probability distributions
- There are many potential applications of GANs
  - Generation of images, video, etc.
  - Data augmentation
  - Missing Data imputation
  - Semi-supervised learning
  - Reinforcement learning
- If carefully constructed, GANs can be used to learn more about the underlying data distributions

Motivation

- The gold standard for discovering causal relationships is experiments
- Experiments can be prohibitively expensive, unethical, or impossible, so there is a need for observational causal discovery
- Causal generative neural networks (CGNNs) learn functional causal models by fitting a generative neural networks that minimizes the maximum mean discrepancy
- Using deep neural networks allows CGNNs to learn more complex causal relationships than other approaches

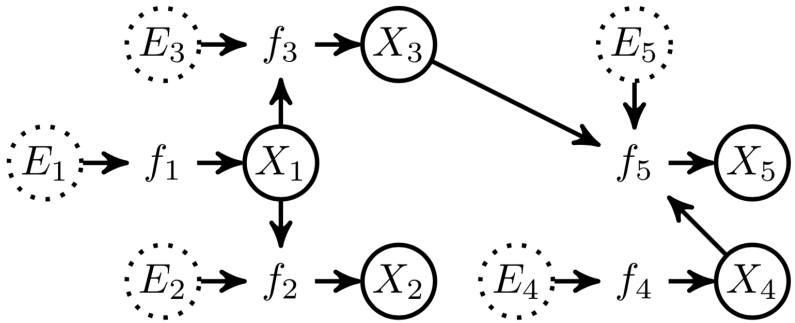
Functional Causal Models

- A functional causal model (FCM) on a vector of random variables $\mathbf{X} = (X_1, X_2, \ldots, X_d)$ is a triplet $C = (\mathcal{G}, f, \mathcal{E})$, where:
  - $\mathcal{G}$ is a graph
  - f characterizes the relationships between X's
  - $\mathcal{E}$ is an error distribution
- FCMs can be represented by a set of equations

$$X_i \leftarrow f_i(X_{Pa(i,\mathcal{G})}, E_i), \; E_i \sim \mathcal{E}, \; \text{for } i = 1, \ldots, d$$

  where $X_{Pa(i;\mathcal{G})}$ are the "parents" of X_i in graph $\mathcal{G}$
- For notational simplicity X_i interchangeably denotes an observed variable and a node in the graph $\mathcal{G}$

Functional Causal Models



$$\begin{cases} E_i & \sim \mathcal{E} \\ X_1 & = f_1(E_1) \\ X_2 & = f_2(X_1, E_2) \\ X_3 & = f_3(X_1, E_3) \\ X_4 & = f_4(E_4) \\ X_5 & = f_5(X_3, X_4, E_5) \end{cases}$$

Source: Goudet et al., 2018

- FCMs can be represented as a directed acyclic graph (DAG) as in the example above
- There exists a direct causal relation from X_j to X_i iff there exists a directed edge X_j to X_i in $\mathcal{G}$

Causal Generative Neural Networks

- Let $\mathrm{X} = (X_1, \ldots, X_d)$ denote a set of continuous random variables with joint distribution P
- If the joint density function associated with P is continuous and strictly positive on a compact subset of $\mathbb{R}^d$ and zero elsewhere, it can be shown that there is a CGNN that approximates P with arbitrary accuracy
- Rather than use a discriminator model to evaluate the generator, CGNNs train the generator to minimize the maximum mean discrepancy (MMD) between the real and generated data

Maximum Mean Discrepancy

- MMD measures whether two distributions are the same
- Let $\mathcal{F}$ be a class of functions $f : X \to \mathbb{R}$ and let p, q be distributions

$$\mathrm{MMD}(\mathcal{F}, p, q) = \sup_{f \in \mathcal{F}} \left(\mathrm{E}_{x \sim p}[f(x)] - \mathrm{E}_{y \sim q}[f(y)] \right)$$

- For samples $X \sim p$ of size m and $Y \sim q$ of size n then the estimate of the MMD is

$$\widehat{\mathrm{MMD}}(\mathcal{F}, X, Y) = \sup_{f \in \mathcal{F}} \left(\frac{1}{m} \sum_{i=1}^{m} f(X_i) - \frac{1}{n} \sum_{i=1}^{n} f(Y_i) \right)$$

- Under certain conditions $\mathrm{MMD}(\mathcal{F}, p, q) = 0$ iff $p = q$

Scoring Metric

- The maximum over $\mathcal{F}$ is made tractable by assuming that $\mathcal{F}$ is the unit ball of a RKHS with kernel k
- For an estimated distribution $\widehat{P}$ we want to know if it is close to the true distribution P
- The estimated MMD between the n-sample observational data D, and an n-sample $\widehat{D}$ from $\widehat{P}$ is

$$\widehat{\mathrm{MMD}}_k(\mathcal{D}, \widehat{\mathcal{D}}) = \frac{1}{n^2}\sum_{i,j=1}^{n} k(x_i, x_j) + \frac{1}{n^2}\sum_{i,j=1}^{n} k(\hat{x}_i, \hat{x}_j) - \frac{2}{n^2}\sum_{i,j=1}^{n} k(x_i, \hat{x}_j)$$

- The estimated FCM $\widehat{C}$ is trained by maximizing

$$S(\widehat{\mathcal{G}}, \mathcal{D}) = -\widehat{\mathrm{MMD}}_k(\mathcal{D}, \widehat{\mathcal{D}}) - \lambda|\widehat{\mathcal{G}}|$$

Searching Causal Graphs

- An exhaustive explorations of all DAGs with d variables using brute force search is infeasible for moderate d
- To solve this issue the authors assume that the skeleton of the graph $\mathcal{G}$ is obtainable from domain knowledge
- The CGNN follows a greedy procedure to find $\mathcal{G}$ and f_i:
  - Orient each $X_i - X_j$ as $X_i \to X_j$ or $X_j \to X_i$ by selecting the 2-variable CGNN with the best score
  - Follow paths from a random set of nodes until all nodes are reached and no cycles are present
  - For a number of iterations, reverse the edge that leads to the maximum improvement of the score $S(\mathcal{G}, \mathcal{D})$ over a d-variable CGNN, without creating a cycle
  - At the end of this process, we evaluate a confidence score for any edge $X_i \to X_j$ as

$$V_{X_i \to X_j} = S(\mathcal{G}, \mathcal{D}) - S(\mathcal{G} - \{X_i \to X_j\}, \mathcal{D})$$

Dealing with Hidden Confounders

- The search method relies on the no unmeasured confounders assumption
- If this assumption is violated, we know that each edge $X_i - X_j$ in the skeleton is due to one out of three possibilities
  - $X_i \to X_j$
  - $X_i \leftarrow X_j$
  - $X_i \leftarrow E_{i,j} \to X_j$ for some unobserved variable $E_{i,j}$
- The search method can be modified to allow for confounders as follows:
  - Each equation in the FCM is extended to:

$$X_i \leftarrow f_i(X_{Pa(i,\mathcal{G})}, E_{i,Ne(i,\mathcal{S})}, E_i)$$

    where $Ne(i, \mathcal{S})$ is the set of indicies of variables adjacent to X_i in the skeleton
  - In this case, regularization by $\lambda|\widehat{\mathcal{G}}|$ promotes simple graphs

Discovering v-structures

- Consider the random variables (A, B, C) with skeleton $A - B - C$, four causal structures are possible
  - $A \rightarrow B \rightarrow C$
  - $A \leftarrow B \leftarrow C$
  - $A \leftarrow B \rightarrow C$
  - $A \rightarrow B \leftarrow C$
- All four structures are Markov equivalent, and therefore indistinguishable from each other using statistics alone
- Previous methods have had difficulty identifying the correct structure
- CGNNs can accurately discriminate between the v-structures using the MMD criteria

Conclusion

- CGNNs are a new framework to learn functional causal models from observational data
- CGNNs combine the power of deep learning and the interpretability of causal models
- CGNNs are better able to identify the causal structure of relationships compared to other methods
- There is still a need to characterize the sufficient identifiability conditions for this approach

Outline

Introduction

- Sequential decision problems arise in many application areas
  - Autonomous Vehicles
  - Finance
  - Logistics
  - Robotics
  - Healthcare
- Markov decision processes (MDPs) are the primary mathematical model for sequential decision problems
- Almost any decision process can be made into an MDP
- Coercing a decision process into the MDP framework can lead to high-dimensional system information that is difficult to model

Motivation

- High-dimensional, infinite horizon MDPs can often be represented by low dimensional approximations
- Option 1: Create a finite discretization of the space and treat the process as a finite MDP
  - Can result in a significant loss of information
  - Can be difficult to apply when the system state information is continuous and high-dimensional
- Option 2: Construct a low-dimensional summary of the underlying system states
  - No guarantee low-dimensional summary contains salient features needed for making good decisions
- Can we find a good low dimensional approximation?

Setup and Notation

- The observed data are

$$\{(S_i^1, A_i^1, U_i^1, S_i^2, \ldots, A_i^T, U_i^T, S_i^{T+1})\}$$

  - $T \in \mathbb{N}$ - Observation time
  - $S^t \in \mathbb{R}^{p_t}$ - Summary of information until time t
  - $A^t \in \mathcal{A} = \{1, \ldots, K\}$ - Decision made at time t
  - $U^t = U^t(S^t, A^t, S^{t+1})$ - Quantifies momentary "goodness" of current action / state transition

- We assume that U is bounded such that $\sup_t |U^t| \leq M$
- The observed data has a time horizon T, but the method should work for any time horizon

Markov and Homogeneous

- We assume that the process is Markov and homogeneous in that

$$P\left(\mathsf{S}^{t+1} \in \mathcal{G}^{t+1} \middle| A^t, \mathsf{S}^t, \ldots, A^1, \mathsf{S}^1\right) = P\left(\mathsf{S}^{t+1} \in \mathcal{G}^{t+1} \middle| A^t, \mathsf{S}^t\right),$$

  where $\mathcal{G}^{t+1} \subseteq \mathrm{dom}\mathsf{S}^{t+1}$, and the probability measure does not depend on t
- These conditions may not be satisfied without some modification
  - For any process $(\mathsf{S}^1, A^1, \mathsf{S}^2 \ldots)$ we can define $\widetilde{\mathsf{S}} = (\mathsf{S}^t, A^{t-1}, \ldots, \mathsf{S}^{t-m_t})$ where m_t is chosen such that the Markov property holds
  - Augmenting the state with a variable for time, i.e. defining the state at time t to be $(\widetilde{\mathsf{S}}^t, t)$, can ensure homogeneity

Decision Strategy

- The decision strategy $\pi : \mathcal{S} \mapsto \mathcal{A}$, makes decision $\pi(\mathsf{s}^t)$ when presented with $\mathsf{S}^t = \mathsf{s}^t$ at time t
- Let $\bar{\mathsf{a}}^t = (a^1, a^2, \ldots, a^t)$ and $\bar{\mathsf{s}}^t = (\mathsf{s}^1, \mathsf{s}^2, \ldots, \mathsf{s}^t)$ be the action and state histories at time t
- $\mathsf{S}^{*t}(\bar{\mathsf{a}}^{t-1})$ is the potential state under trajectory $\bar{\mathsf{a}}^{t-1}$
- The potential state under decision strategy π is

$$\mathsf{S}^{*t}(\pi) = \sum_{\bar{\mathsf{a}}^{t-1}} \mathsf{S}^{*t}(\bar{\mathsf{a}}^{t-1}) \prod_{v=1}^{t-1} 1_{\pi\{\mathsf{S}^{*v}(\bar{\mathsf{a}}^{v-1})\}=a^v}$$

Value Function

- The potential utility under decision strategy π is

$$U^{*t}(\pi) = U\left[\mathsf{S}^{*t}(\pi), \pi(\mathsf{S}^{*t}(\pi)), \mathsf{S}^{*(t+1)}(\pi)\right]$$

- The discounted mean utility under π is

$$V(\pi) = \mathbb{E}\left\{\sum_{t\geq 1} \gamma^{t-1} U^{*t}(\pi)\right\}$$

- For a class of decision strategies Π, the optimal decision strategy $\pi^{\text{opt}} \in \Pi$ satisfies $V(\pi^{\text{opt}}) \geq V(\pi)\ \forall\ \pi \in \Pi$

Sufficient Markov Decision Processes

- It can be difficult to construct a high-quality estimator of π^{opt} when the states S^t are high-dimensional
- For any map $\phi : \mathcal{S} \mapsto \mathbb{R}^q$, define $\mathsf{S}_\phi^t = \phi(\mathsf{S}_t)$
- ϕ induces a sufficient MDP for π^{opt} if $(\overline{\mathsf{A}}^t, \overline{\mathsf{S}}_\phi^{t+1}, \overline{\mathsf{U}}^t)$ contains all relevant information about π^{opt}
- Given a policy $\pi_\phi : \text{dom} S_\phi^t \mapsto \mathcal{A}$, the potential utility under π_ϕ is

$$U_\phi^{*t}(\pi_\phi) = \sum_{\overline{\mathsf{a}}^t} U\left[\mathsf{S}^{*t}(\overline{\mathsf{a}}^{t-1}), a^t, \mathsf{S}^{*(t+1)}(\overline{\mathsf{a}}^t)\right] \prod_{v=1}^{t} 1_{\pi_\phi\{\mathsf{S}_\phi^{*v}(\overline{\mathsf{a}}^{v-1})\}=a^v}$$

Sufficient Markov Decision Processes

- Let $\Pi \subseteq \mathcal{A}^{\mathcal{S}}$ denote a class of decision strategies defined on $\mathcal{S}$ and $\Pi_\phi \subseteq \mathcal{A}_\phi^{\mathcal{S}}$ a class of decision strategies defined on $\mathcal{S}_\phi = \text{dom}\mathsf{S}_\phi^t \subseteq \mathbb{R}^q$
- The pair (ϕ, Π_ϕ) induces a sufficient MDP for π^{opt} within Π if the following conditions hold for all $t \in \mathbb{N}$:
  - The process $(\overline{\mathsf{A}}^t, \overline{\mathsf{S}}_\phi^{t+1}, \overline{\mathsf{U}}^t)$ is Markov and homogeneous
  - There exists $\pi^{\text{opt}} \in \text{argmax}_{\pi \in \Pi} V(\pi)$ which can be written as $\pi^{\text{opt}} = \pi_\phi^{\text{opt}} \circ \phi$, where

$$\pi_\phi^{\text{opt}} \in \underset{\pi_\phi \in \Pi_\phi}{\text{argmax}}\, \mathbb{E}\left\{ \sum_{t \geq 1} \gamma^{t-1} U_\phi^{*t}(\pi_\phi) \right\}$$

- It suffices to store only the process $\{(\overline{\mathsf{S}}_{\phi,i}^{t+1}, \overline{\mathsf{A}}_i^t, \overline{\mathsf{U}}_i^t)\}_{i=1}^n$

Conditional Independence

- Verifiable conditions for checking that (ϕ, Π_ϕ) induces a sufficient MDP require a few assumptions
  - Consistency: $\mathsf{S}^t = \mathsf{S}^{*t}(\overline{\mathsf{A}}^{t-1})$
  - Positivity: $P\left(A^t = a^t \middle| \overline{\mathsf{S}}^t = \overline{\mathsf{s}}^t, \overline{\mathsf{A}}^{t-1} = \overline{\mathsf{a}}^{t-1}\right) > 0$
  - Sequential ignorability: $\{\mathsf{S}^{*t}(\overline{\mathsf{a}}^{t-1})\}_{t\geq 1} \perp A^t | \overline{\mathsf{S}}^t, \overline{\mathsf{A}}^{t-1}$
- Define $\mathsf{Y}^{t+1} = \{U^t, (\mathsf{S}^{t+1})^T\}^T$ for all $t \in \mathbb{N}$
- Let $(\mathsf{S}^1, A^1, U^1, \mathsf{S}^2, \dots)$ be an MDP that satisfies the above assumptions. Suppose that there exists $\phi : \mathcal{S} \mapsto \mathbb{R}^q$ such that

$$\mathsf{Y}^{t+1} \perp \mathsf{S}^t | \mathsf{S}^t_\phi, A^t$$

then, (ϕ, Π_ϕ) induces a sufficient MDP for π^{opt} within Π

Conditional Independence

- Let $(S^1, A^1, U^1, S^2, \dots)$ be an MDP that satisfies the previous assumptions. Suppose that there exists $\phi : \mathcal{S} \mapsto \mathbb{R}^q$ such that at least one of the following conditions hold:

  (i) $\{Y^{t+1} - \mathbb{E}(Y^{t+1}|S^t_\phi, A^t)\} \perp S^t|A^t$

  (ii) $\{S^t - \mathbb{E}(S^t|S^t_\phi)\} \perp (Y^{t+1}, S^t_\phi)|A^t$

  then $Y^{t+1} \perp S^t|S^t_\phi, A^t$

- This result can be used to verify the conditional independence condition using Brownian distance covariance

Variable Screening

- Conditional independence may be too strong of an assumption in the presence of certain noise variables
- For example, let $\{\mathrm{B}^t\}_{t\geq 1}$ denote a homogeneous Markov process independent of $(\mathrm{S}^1, A^1, U^1, \mathrm{S}^2, \dots)$
- Y^{t+1} need not be conditionally independent of $\{(\mathrm{S}^t)^T, (\mathrm{B}^t)^T\}^T$ given S^t, but π^{opt} does not depend on $\{\mathrm{B}^t\}_{t\geq 1}$
- For any map $\phi : \mathcal{S} \mapsto \mathbb{R}^q$, let $\mathrm{Y}_\phi^{t+1} = \{U^t, (\mathrm{S}_\phi^{t+1})^T\}^T$
- If $\mathrm{Y}_\phi^{t+1} \perp \mathrm{S}^t | \mathrm{S}_\phi^t, A^t$ then, (ϕ, Π_ϕ) induces a sufficient MDP for π^{opt} within Π

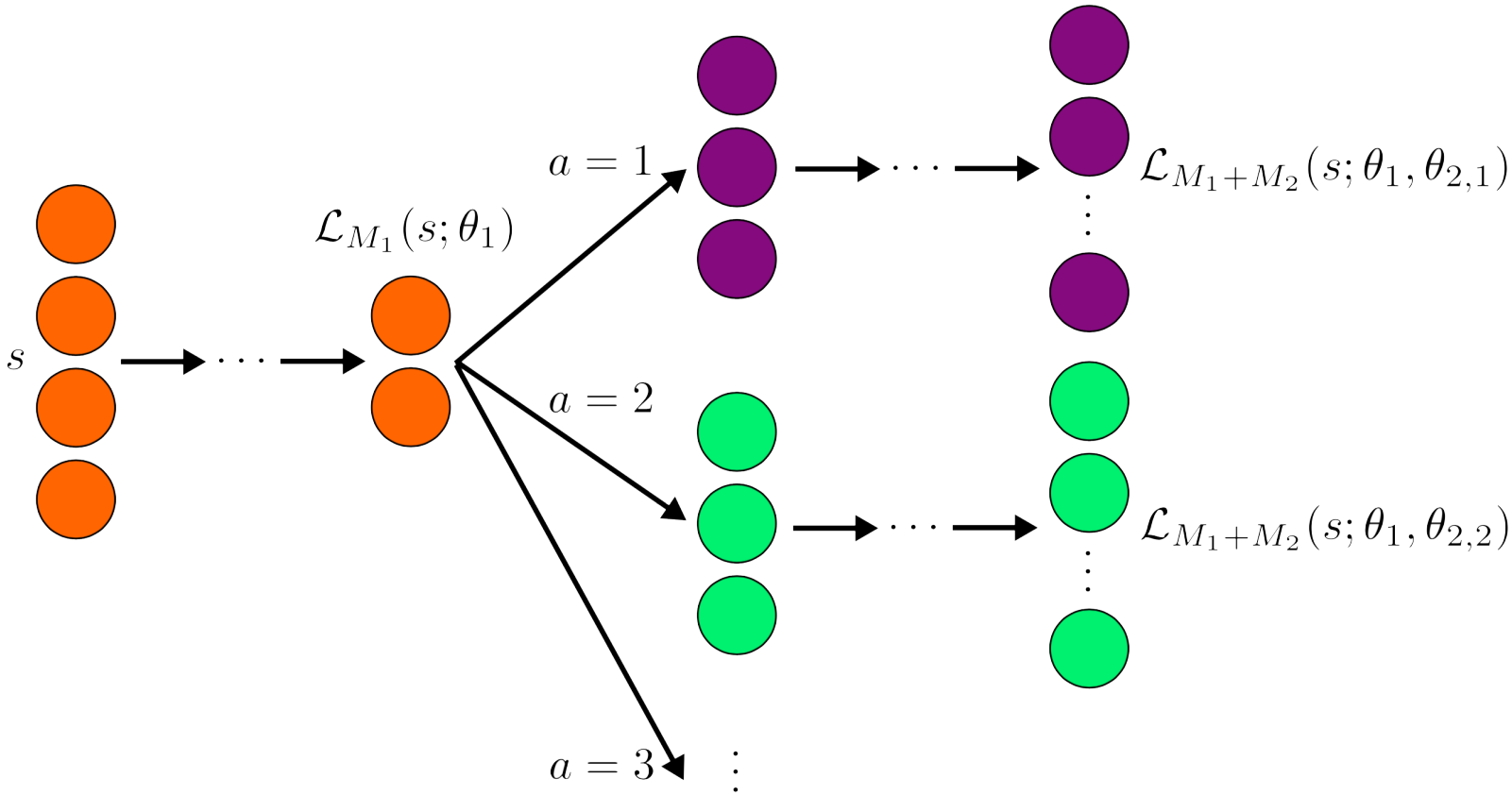
Alternating Deep Neural Networks

- Let $\mathcal{S} = \mathbb{R}^p$ and consider summary functions $\phi : \mathbb{R}^p \mapsto \mathbb{R}^q$ that are representable as multi-layer neural networks
- To construct a data-driven summary function ϕ, we require a model for the regression of Y^{t+1} on S_ϕ^t and A^t
  - This predictive model can also be representable as a multi-layer neural network
- The model can be visualized as two connected multi-layer neural networks
  - One that composes the feature map ϕ
  - One that models the regression of Y^{t+1} on S_ϕ^t and A^t

Alternating Deep Neural Networks

- The First M_1 layers of of the NN determine the feature map $\phi \equiv \mathcal{L}_{M_1}(\mathsf{s}, \theta_1)$
- The following M_2 layers fit the regression model $\mathcal{L}_{M_2}(\mathsf{s}, \theta_1, \theta_2)$
  - Separate regression models are fit for each action a
- There are several tuning parameters including the number of layers, width of the layers, and the dimension of the feature map $\phi : \mathcal{S} \mapsto \mathbb{R}^q$
- The dimension q is chosen to be the lowest dimension for which the Brownian distance covariance test of independence fails to reject at a pre-specified error level

Alternating Deep Neural Networks



Conclusions

- Choosing a parsimonious representation of a decision process that fits the MDP model is non-trivial
- The deep neural networks can model complex and nonlinear structure in the high-dimensional state space
- The method proposed above can effectively reduce the dimension of the state space without adversely impacting the performance
- Possible future work includes
  - Online estimation of the feature map
  - States with complex data structures (e.g. images and text)

Introduction

- Many conditions have multiple treatments available
- Combination therapies are becoming increasingly common
- Multiclass individualized treatment rule (ITR) estimation methods are not scalable with large numbers of treatment options
- Existing algorithms may not be adequate for finding ITRs with combination therapies

Motivation

- Patients with type 2 diabetes often receive multiple medications because a single treatment may be insufficient to effectively control the blood glucose level
- Medications exerting their effects through different mechanisms can increase effectiveness of the therapy
  - Sulfonylurea (SU) increases insulin release from β-cells in the pancreas
  - DDP4 increases incretin level, which inhibits glucagon release
- Effects may be additive or weakly interacting because DDP4 and SU function through different biological pathways

Multiclass vs Multilabel

- Combination therapies often allow patients to be assigned to any combination of K possible treatments
- Multiclass classification problems would have 2^K classes
  - Dimension increases exponentially with the number of treatments
  - Proposed strategies to simplify computation have many drawbacks
- Multilabel classification considers K binary treatment choices (yes/no for each drug)
  - Reduces computational cost of estimation
  - Requires treatment effects to be additive or small interactions

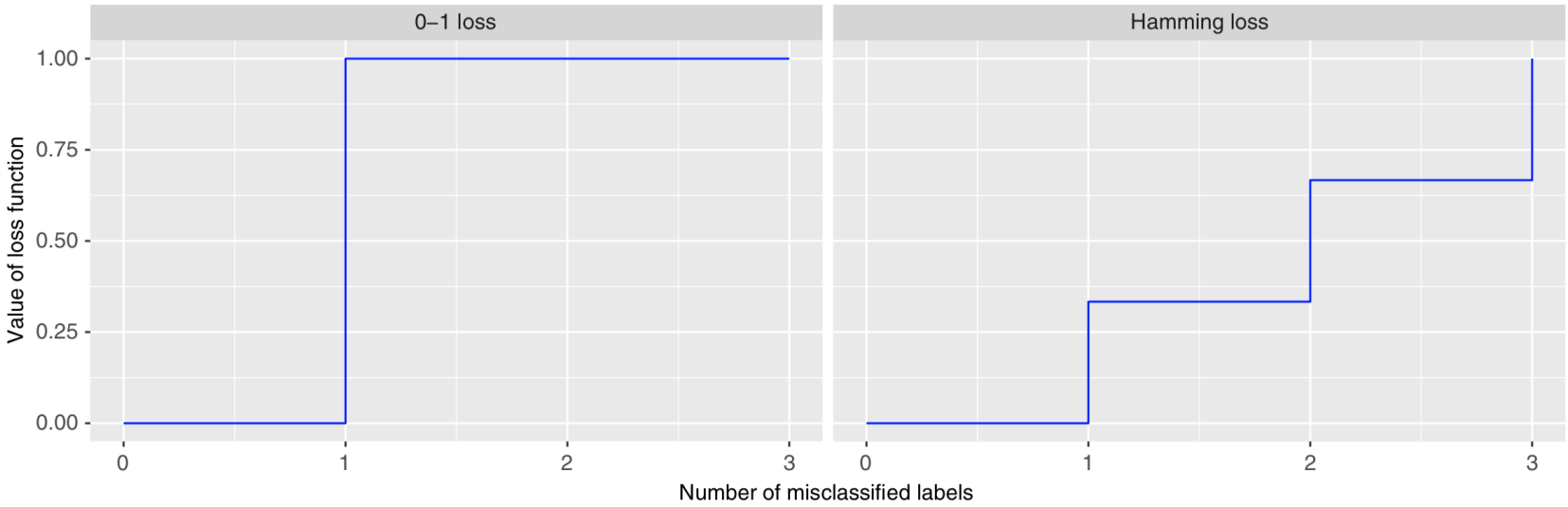
Hamming Loss

- Let $A = (A_1, \ldots, A_K) \in \mathcal{A}$ represent a vector of length K, where $\mathcal{A} = \{-1, 1\}^K$
- Accordingly, the decision rule is $D(X) = (D_1(X), \ldots, D_K(X)) \in \{-1, 1\}^K$
- A commonly used loss function for multilabel classification is the Hamming loss

$$\frac{1}{K} \sum_{k=1}^{K} I\{A_k \neq D_k(X)\}$$

- The Hamming loss quantifies the proportion of mislabeled treatments

Hamming Loss



- Like the 0-1 loss, the Hamming loss is discontinuous and difficult to optimize
- To aid computation we will use a convex surrogate loss

OWL with Multilabel Classification

- Consider the following loss function for a given decision rule D(X):

$$L(D) = \frac{1}{n}\sum_{i=1}^{n}\frac{R_i}{\pi_{A_i}}\frac{1}{K}\sum_{k=1}^{K} I\{A_k \neq D_k(X)\}$$

  where $\pi_{A_i} = Pr(A = a|X)$
- For multiclass classification, this loss reduces to the outcome weighted 0-1 loss
- In observational studies π_{A_i} can be difficult to estimate when K is large. Possible approaches:
  - $P(A = a|X) = P(A = a)$
  - $P(A = a|X) = \prod_{k=1}^{K} P(A_k = a_k|X)$

Decision Rule with Deep Learning

- Any suitable classifier can fit into the proposed framework
- We consider neural networks because of their advantages in shared subspace and scalable computation
- Let $\widehat{D}(X) = (\widehat{D}_1(X), \ldots, \widehat{D}_K(X))$ be the estimated decision rule
- Using the hinge loss, we can formulate the optimization problem as

$$\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \frac{R_i}{\pi_{A_i}} \frac{1}{K} \sum_{k=1}^{K} [1 - A_{ik} \widehat{D}_k(X_i)]_+$$

where θ represents all parameters in the NN

Overfitting

- Due to strong representative power of DNN, overfitting can be an issue
- The authors recommend a regularization strategy with penalization by solving the following problem

$$\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \frac{R_i}{\pi_{A_i}} \frac{1}{K} \sum_{k=1}^{K} [1 - A_{ik} \widehat{D}_k(X_i)]_+ + \lambda f(\theta)$$

- Let L be the number of layers and W_ℓ be the weight matrix from the ℓ^{th} layer
  - Ridge penalty: $f(\theta) = \sum_{\ell=1}^{L} \|W_\ell\|_F^2$ (Frobenius norm)
  - Lasso penalty: $f(\theta) = \sum_{\ell=1}^{L} \|W_\ell\|_1$
- Other methods to prevent overfitting can also be used

Fisher Consistency of Hamming loss

- Define the risk of the outcome weighted 0-1 loss as

$$\mathcal{R}(D) = \mathbb{E}\left[\frac{R}{\pi_A} I\{A \neq D(X)\}\right]$$

- And the risk of the outcome weighted Hamming loss as

$$\mathcal{R}_H(D) = \mathbb{E}\left[\frac{R}{\pi_A}\frac{1}{K}\sum_{k=1}^{K} I\{A_k \neq D_k(X)\}\right]$$

- Any decision rule $\widetilde{D}$ such that $\mathcal{R}_H(\widetilde{D}) = \inf_D\{\mathcal{R}_H(D)\}$ also satisfies $\mathcal{R}(\widetilde{D}) = \inf_D\{\mathcal{R}(D)\}$
- Since the outcome weighted 0-1 loss is fisher consistent, so is the outcome weighted Hamming loss

Multilabel Consistency of the Surrogate Loss

- Define the risk of the surrogate outcome weighted Hamming loss as

$$\Phi_H(D) = \mathbb{E}\left[\frac{R}{\pi_A}\frac{1}{K}\sum_{k=1}^{K}\phi\left(A_k D_k(X)\right)\right]$$

  where ϕ is a predefined convex function

- The surrogate loss is consistent with outcome weighted Hamming loss if ϕ is one of the following

  1. Exponential: $\phi(x) = e^{-x}$
  2. Hinge: $\phi(x) = (1-x)_+$
  3. Least squares: $\phi(x) = (1-x)^2$
  4. Logisticregression: $\phi(x) = \ln(1+e^{-x})$

Conclusions

- Using multilabel classification can make finding ITRs for combination therapies tractable
- The proposed Hamming loss does not account for strong interactions among treatments which may limit its applicability
- A generalization of the Hamming loss can handle interactions, but the theoretical properties are unknown
- Possible extensions include multiple decision points and estimating contrasts of treatment effects in combination therapies

Outline

Shared decision making

- So far we have focused on DTRs that tailor treatments to individual patient characteristics.
- Patient characteristics may include clinical information as well as patient preferences.
- Recall that an optimal DTR maximizes the mean of a pre-specified clinical outcome if it is applied to all patients in a population of interest.
- While this definition of optimality is mathematically convenient, it does not directly allow for shared decision making where patient preferences are integrated into the decision process.

Why including patient preferences into DTR construction is important

- Including patient preferences in treatment selection in a mathematically rigorous way is important.
- First, it facilitates 'patient-centered care' in which patients play a key role in decision making and the evaluation of their own outcomes.
- Second, it offers a principled means for matching patient preferences to an optimal treatment based on potentially complex outcome profiles.

Patient preference elicitation

- Eliciting patient preferences is not necessarily straightforward.
- For example, it would be convenient if patients could choose parameters indexing a composite outcome. However, without specialized training for the patients, this is not feasible.
- An alternative approach is to administer a questionnaire populated with items that are accessible to a patient in a domain context yet are informative about preferences in the outcome space.
- Butler, Laber, Davis, and Kosorok (2018) incorporate this latter approach to derive a preference-sensitive optimal DTR for each patient. We will explore their proposed methodology in detail.

Setup and notation

- Observed data $\{\mathrm{W}_i, \mathrm{X}_i, A_i, Y_i, Z_i\}_{i=1}^n$ comprises n independent and identically distributed tuples $(\mathrm{W}, \mathrm{X}, A, Y, Z)$ where
  - $\mathrm{W} \in \{0,1\}^p$ denotes answers to items in a preference questionnaire,
  - $\mathrm{X} \in \mathbb{R}^m$ denotes pre-treatment patient covariates,
  - $A \in \{-1, 1\}$ denotes the assigned treatment,
  - $Y, Z \in \mathbb{R}$ denote outcomes of interest, coded so that higher values are better.
- A DTR is denoted by $\pi : \mathrm{dom}\ \mathrm{W} \times \mathrm{dom}\ \mathrm{X} \to \mathrm{dom}\ A$.

Optimal DTR

To define an optimal DTR,

- Assume that each individual in the pouplation has a latent preference, $H \in \mathbb{R}$, that indexes a utility function $U(y, z; h)$.

- Assume that the utility function induces an ordering on dom $Y \times$ dom Z so that a patient with preference $H = h$ would prefer outcomes (y, z) to (y', z') if $U(y, z; h) \geq U(y', z'; h)$.

Optimal DTR (cont.)

- Let $Y^*(a)$ and $Z^*(a)$ denote the potential outcomes under treatments $a \in \{-1, 1\}$ so that $U\{Y^*(a), Z^*(a)\}$ is the potential utility function under treatment a.
- Define the potential utility as

$$V_U(\pi) = \mathbb{E}\left[\sum_{a\in\{-1,1\}} U\{Y^*(a), Z^*(a); H\} I(\pi(\mathsf{W}, \mathsf{X}) = a)\right]$$

  where the expectation is taken with respect to the joint distribution of $\{\mathsf{X}, \mathsf{W}, H, A, Y^*(0), Y^*(1), Z^*(0), Z^*(1)\}$.
- The optimal DTR, π_U^{opt} satisfies

$$V_U(\pi_U^{\text{opt}}) \geq V_U(\pi) \quad \text{for all } \pi.$$

Form of the utility

- Assume the utility has the form

$$U(y, z; h) = \Phi(h)y + \{1 - \Phi(h)\}z$$

  where $\Phi(\cdot)$ denotes the cumulative distribution function for a standard normal random variable.

- Interpretation: A patient with preference h cares $\Phi(h)/(1 - \Phi(h))$ more about Y than Z.

- Linear utility is a common assumption in multiobjective optimization, however the assumption of a constant gain in utility per unit increase in outcome may not be reasonable in some contexts.

The optimal DTR for any rational utility can be expressed as the optimal DTR for some linear utility

- A rational utility will always prefer a treatment that is better on both outcomes to one that is worse on both outcomes.
- Butler et al. (2018) prove that the DTR for any rational utility may be expressed as the optimal DTR for some linear utility function.
- To state this formally, define the following

$$
\begin{aligned}
R_Z(\mathsf{w},\mathsf{x}) &= \{Z^*(1)|\mathsf{W}=\mathsf{w},\mathsf{X}=\mathsf{x}\} - \{Z^*(-1)|\mathsf{W}=\mathsf{w},\mathsf{X}=\mathsf{x}\} \\
R_Y(\mathsf{w},\mathsf{x}) &= \{Y^*(1)|\mathsf{W}=\mathsf{w},\mathsf{X}=\mathsf{x}\} - \{Y^*(-1)|\mathsf{W}=\mathsf{w},\mathsf{X}=\mathsf{x}\}, \\
R_U(\mathsf{w},\mathsf{x}) &= \{U\{Y^*(1),Z^*(1);H\}|\mathsf{W}=\mathsf{w},\mathsf{X}=\mathsf{x}\} \\
&\quad - \{U\{Y^*(-1),Z^*(-1);H\}|\mathsf{W}=\mathsf{w},\mathsf{X}=\mathsf{x}\}.
\end{aligned}
$$

- Note, it can be shown that $\pi_U^{\text{opt}}(\mathsf{w},\mathsf{x}) = \text{sign}\{R_U(\mathsf{w},\mathsf{x})\}$

The optimal DTR for any rational utility is the optimal DTR for some linear utility (formally)

Lemma (2.1)

Assume that $\max\{R_U(\mathrm{w},\mathrm{x})R_Z(\mathrm{w},\mathrm{x}), R_U(\mathrm{w},\mathrm{x})R_Y(\mathrm{w},\mathrm{x})\} > 0$ *for all* x, w. *Then, there exists a real-valued random variable,* H', *such that: (i)* $H' \perp\!\!\!\perp A, \{Z^*(a), Y^*(a) : a \in \{-1,1\}\}|\mathrm{X},\mathrm{W}$; *and (ii) the DTR*

$$\pi^{opt}_{CVX}(\mathrm{x},\mathrm{w}) = \underset{a\in\{-1,1\}}{\operatorname{argmax}}\, E[\Phi(H')Y^*(a) + \{1-\Phi(H')\}Z^*(a)|\mathrm{X}=\mathrm{x},\mathrm{W}=\mathrm{w}]$$

satisfies $V_U(\pi^{opt}_{CVX}) = V_U(\pi^{opt}_{U})$.

Identification

- The optimal DTR is defined in terms of potential outcomes. To identify the model, we assume
  - C1 Causal consistency, $(Y, Z) = \{Y^*(A), Z^*(A)\}$,
  - C2 Positivity, there exists $\epsilon > 0$ so that $P(A = a|\mathrm{X}, \mathrm{W}) \geq \epsilon$,
  - C3 Ignorability, $[\{Y^*(a), Z^*(a)\} : a \in \{-1, 1\}] \perp\!\!\!\perp A|\mathrm{X}, \mathrm{W}$, and
  - C4 $(A, Y, Z) \perp\!\!\!\perp H|\mathrm{X}, \mathrm{W}$.

- C1, C2, and C3 are standard assumptions.

- C4 holds if the assumptions of Lemma 2.1 hold.

- C4 can be weakened to $A \perp\!\!\!\perp H|\mathrm{X}, \mathrm{W}$ at the expense of postulating a model for the conditional mean of $\Phi(H)Y$ and $\Phi(H)Z$ given $(\mathrm{X}, \mathrm{W}, Z)$.

Estimation strategy

- Under C1, C2, and C3, it can be shown that

$$\pi^{\text{opt}}(\mathsf{x},\mathsf{w}) = \underset{a\in\{-1,1\}}{\text{argmax}}\, \mathbb{E}[\Phi(H)Y + \{1-\Phi(H)\}Z|\mathsf{X}=\mathsf{x},\mathsf{W}=\mathsf{w},A=a]$$

  which under C4 can be written as

$$\pi^{\text{opt}}(\mathsf{x},\mathsf{w}) = \underset{a\in\{-1,1\}}{\text{argmax}}\, E[\Phi(H)|\mathsf{X}=\mathsf{x},\mathsf{W}=\mathsf{w}]\mathbb{E}(Y|\mathsf{X}=\mathsf{x},\mathsf{W}=\mathsf{w},A=a)$$
$$+[1-E\{\Phi(H)|\mathsf{X}=\mathsf{x},\mathsf{W}=\mathsf{w}\}]\mathbb{E}(Z|\mathsf{X}=\mathsf{x},\mathsf{W}=\mathsf{w},A=a)].$$

- This suggests a strategy for estimating π^{opt}:
  - Construct estimators for
    $Q_Y(\mathsf{x},\mathsf{w},a) = \mathbb{E}(Y|\mathsf{X}=\mathsf{x},\mathsf{W}=\mathsf{w},A=a)$ and
    $Q_Z(\mathsf{x},\mathsf{w},a) = \mathbb{E}(Z|\mathsf{X}=\mathsf{x},\mathsf{W}=\mathsf{w},A=a)$
  - Postulate a latent preference model linking the unobservable preference H with covariates X and preference questionnaire items W and use this model to estimate
    $\mu_H(\mathsf{x},\mathsf{w}) = \mathbb{E}\{\Phi(H)|\mathsf{X}=\mathsf{x},\mathsf{W}=\mathsf{w}\}$.
  - Plug in estimators to estimate π^{opt}.

Estimation of the Q functions

- To estimate Q_Y and Q_Z, Butler et al. (2018) propose linear working models of the form

$$Q_Y(\mathsf{x}, \mathsf{w}, a; \psi_Y) = \mathsf{x}_{Y,0}^\intercal \psi_{Y,0} + \mathsf{w}_{Y,1}^\intercal \psi_{Y,1} + a\mathsf{x}_{Y,1}^\intercal \psi_{Y,2} + a\mathsf{w}_{Y,1}^\intercal \psi_{Y,3}$$
$$Q_Z(\mathsf{x}, \mathsf{w}, a; \psi_Z) = \mathsf{x}_{Z,0}^\intercal \psi_{Z,0} + \mathsf{w}_{Z,0}^\intercal \psi_{Z,1} + a\mathsf{x}_{Z,1}^\intercal \psi_{Z,2} + a\mathsf{w}_{Z,1}^\intercal \psi_{Z,3},$$

  where $\mathsf{x}_{\ell,j}$ and $\mathsf{w}_{\ell,j}$ for $\ell = Y, Z$ and $j = 0, 1$ are known feature vectors from x and w and ψ_W and ψ_Y are unknown parameter vectors.

- Let $\hat{\psi}_{Y,n} = \text{argmin}_{\psi_Y} \mathbb{P}_n\{Y - Q_Y(\mathsf{X}, \mathsf{W}, A; \psi_Y)\}^2$ and $\hat{\psi}_{Z,n} = \text{argmin}_{\psi_Z} \mathbb{P}_n\{Z - Q_Z(\mathsf{X}, \mathsf{W}, A; \psi_Z)\}^2$.

- Construct estimators $Q_Y(\mathsf{x}, \mathsf{w}, a; \hat{\psi}_Y)$ and $Q_Z(\mathsf{x}, \mathsf{w}, a; \hat{\psi}_Z)$ of $Q_Y(\mathsf{x}, \mathsf{w}, a)$ and $Q_Z(\mathsf{x}, \mathsf{w}, a)$, respectively.

Specification of the latent preference model

- Assume that $H \perp\!\!\!\perp X|W$ for ease of exposition. (Note that X could be included in the latent patient preference model.)

- Assume that the latent patient preferences are connected to items on the questionnaire through a Rasch model of the form

$$\text{logit}\{P(W_j = 1|H = h)\} = \beta_{0,j} + \beta_{1,j}h, \quad j = 1, \ldots, p$$

  which is indexed by $\beta = (\beta_{0,1}, \beta_{1,1}, \ldots, \beta_{0,p}, \beta_{1,p})$.

- Let β^* denote the true parameter value. The EM algorithm can be used to construct an estimator $\hat{\beta}_n$ of β^*.

Estimation of μ_H

- Given an estimator $\hat{\beta}_n$ and a postulated marginal distribution, p_h for the latent preferences, the conditional distribution of H given $\mathrm{W} = \mathrm{w}$ is proportional to $p(\mathrm{w}|h)p_h(h)$.
- This conditional distribution can be approximated using Metropolis Hastings.
- A computationally less burdensome approach is to apply a method of moments type estimator:
  - Let $\hat{h}(\mathrm{w})$ denote the solution to $\sum_{j=1}^{p} \hat{\beta}_{n,1,j} \mathrm{expit}(\hat{\beta}_{n,j,0} + \hat{\beta}_{n,1,j} h) = \sum_{j=1}^{p} \hat{\beta}_{n,1,j} w_j$.
- Subsequently let $\hat{\mu}_{H,n}(\mathrm{x}, \mathrm{w}) = \Phi(\hat{h}_n(\mathrm{w}))$ denote our estimator of $\mu(\mathrm{w}, \mathrm{x})$.

Estimation of π^{opt}

With estimates of μ_H, Q_Z, and Q_Y in hand, we can compute an estimate of the optimal DTR,

$$\hat{\pi}_n(\mathsf{x}, \mathsf{w}) = \underset{a \in \{-1,1\}}{\text{argmax}} [\hat{\mu}_{H,n}(\mathsf{x}, \mathsf{w}) \hat{Q}_{Y,n}(\mathsf{x}, \mathsf{w}, a) + \{1 - \hat{\mu}_{H,n}(\mathsf{x}, \mathsf{w})\} \hat{Q}_{Z,n}(\mathsf{x}, \mathsf{w}, a)].$$

Assumptions for the theoretical results

Let $h^*(\mathrm{w})$ denote the solution to $\sum_{j=1}^{p} \beta^*_{1,j} \mathrm{expit}(\beta^*_{j,0} + \beta^*_{j,1} h) = \sum_{j=1}^{p} \beta^*_{j,1} w_j$. Assume the following

(A1) The number of items satisfies $3 \leq p_n = o(e^n)$.

(A2) The estimator $\hat{h}_n(\mathrm{w})$ converges in probability to $h^*(\mathrm{w})$, pointwise for all w.

(A3) The estimators $\hat{Q}_{Y,n}(\mathrm{x}, \mathrm{w}, a)$ and $\hat{Q}_{Z,n}(\mathrm{x}, \mathrm{w}, a)$ converge in probability to $Q_Y(\mathrm{x}, \mathrm{w}, a)$ and $Q_Z(\mathrm{x}, \mathrm{w}, a)$ pointwise for each x, w, and a.

Theoretical results

The first theoretical result establishes consistency of the proposed estimator for the optimal DTR as the sample size diverges but the number of items remains fixed.

Theorem (Thm 2.2 in Butler et al. (2018))

Assume (A1) - (A3) and let the number of items, $p_n = p$, be fixed. Then $V_U(\pi^{opt}) - V_U(\hat{\pi}_n)$ converges to zero in probability as $n \to \infty$.

The second theoretical result says that if the number of items is allowed to diverge with the sample size then the estimated DTR performs as well as an oracle that knows each patient's individual preference.

Theorem (Thm 2.3 in Butler et al. (2018))

Assume (A1) - (A3) and suppose $p_n \to \infty$ as $n \to \infty$. Then $V_U(\hat{\pi}_n) - V_U(\pi^{oracle})$ converges to zero in probability as $n \to \infty$.

Case Study: CATIE

- The Clinical Antipsychotic Trials of Intervention Effectiveness (CATIE) schizophrenia trial was designed to compare new antipsychotic drugs to conventional ones in a randomized, controlled, double-blind, multi-phase trial.

- The trial targeted patients already being treated for schizophrenia but who might benefit from a medicinal change.

- Patients received antipsychotic treatments and were offered psychosocial treatment with their families.

First phase of CATIE

- The first phase of CATIE is suited for application of the method proposed in Butler et al. (2018).
- Patients were randomized to one of 5 medications,
  - 4 were atypical antipsychotics, and
  - 1 was a conventional antipsychotic (perphenazine).
- For ease of exposition, we will dichotomize the treatments into atypical antipsychotics and perphenazine.
- At baseline, patients answered a 10 question, binary response assessment, which can be used to measure patient preferences across two outcomes
  - Efficacy using the Positive and Negative Syndromes Scale (PANSS), and
  - Side effect burden measured as the sum of side effect and adverse event indicators.

Patient preference questions

The patient preference information was collected using a 10 question Drug Attitude Inventory assessment. One question was excluded from analysis.

Patient preference questions

Question	Coding
"For me, the good things about medication outweigh the bad"	Yes (1) No (0)
"I feel weird, like a 'zombie', on medication"	Yes (0) No (1)
"Medications make me feel more relaxed"	Yes (1) No (0)
"Medications make me feel tired and sluggish"	Yes (0) No (1)
"I take medications only when Im sick"	Yes (0) No (1)
"I feel more normal on medication"	Yes (1) No (0)
"It is unnatural for my mind and body to be controlled by medications"	Yes (0) No (1)
"My thoughts are clearer on medication"	Yes (1) No (0)
"By staying on medications I can prevent being sick"	Yes (1) No (0)

Figure 1: Table 1 from Butler et al. (2018)

Analysis

Tailoring covariates were selected based on clinical expertise and prior analyses.

Coefficients of the Q-Function for Y and Z: CATIE Case Study

	Q_Y	Q_Z
Intercept	−5.51	13.04
Treatment	9.97	−2.09
Male (ref: Female)	−5.91	0.19
Black (ref: Other)	3.08	−0.80
White (ref: Other)	4.02	−0.14
Age	−0.11	−0.01
Baseline Clinical Severity of Schizophrenia	3.29	−0.19
Age at First Antipsychotic Medication	0.24	0.02
Olanzapine (at baseline, pre-randomization)	−3.77	−0.08
Quetiapine (at baseline, pre-randomization)	−5.91	0.24
Risperidone (at baseline, pre-randomization)	−1.48	0.06
Ziprasidone (at baseline, pre-randomization)	−0.94	0.60
Perphenazine (at baseline, pre-randomization)	−3.01	1.17
Haldol (at baseline)	−3.72	0.38
Decanoate (at baseline)	−7.68	0.73
BMI	−0.04	−0.01
Diastolic Blood Pressure	0.05	0.01
Systolic Blood Pressure	−0.10	0.0008
Treatment*Male (ref: Female)	−5.05	−0.21
Treatment*Black (ref: Other)	−1.86	−0.42
Treatment*White (ref: Other)	−2.49	0.25
Treatment*Age	0.05	0.01
Treatment*Baseline Clinical Severity of Schizophrenia	0.21	−0.01
Treatment*Age at First Antipsychotic Medication	−0.04	0.02
Treatment*Olanzapine (at baseline, pre-randomization)	−1.07	0.26
Treatment*Quetiapine (at baseline, pre-randomization)	−5.22	−0.08
Treatment*Risperidone (at baseline, pre-randomization)	−2.78	0.22
Treatment*Ziprasidone (at baseline, pre-randomization)	3.97	0.62
Treatment*Perphenazine (at baseline, pre-randomization)	−1.04	0.03
Treatment*Haldol (at baseline, pre-randomization)	−1.38	0.20
Treatment*Decanoate (at baseline, pre-randomization)	−0.13	1.60
Treatment*BMI	0.02	0.05
Treatment*Diastolic Blood Pressure	−0.01	0.01
Treatment*Systolic Blood Pressure	−0.05	−0.02

Figure 2: Table 2 from Butler et al. (2018)

Analysis (cont.)

- By examining the coefficients for the Q-functions, we see that the main effect of treatment has an opposite sign in the two Q-functions as well as several interactions involving treatment.

- This suggests a trade-off between the two outcomes that must be made in choosing a treatment and that this trade off varies across patient characteristics.

Estimated optimal treatment allocation

While efficacy appears to favor atypical antipsychotics, side effect burden tends to favor perphenazine. The composite outcome occupies a middle ground between the two marginal outcomes.

Treatment recommendations

	Perphenazine	Atypical Antipsychotics
$\hat{\pi}_Y$: Efficacy	36%	64%
$\hat{\pi}_Z$: Side Effect Burden	67%	33%
$\hat{\pi}_n$: $\Phi(E)\,Y + \{1 - \Phi(E)\}Z$	41%	59%

Figure 3: Table 3 from Butler et al (2018)

Percent agreement in treatment recommendations

This table shows the fraction of overlap between the proposed estimated optimal DTR and the optimal DTR based only on efficacy or side effect burden.

Percent of agreement in treatment recommendations

	$\hat{\pi}_Y$: Efficacy	$\hat{\pi}_Z$: Side Effect Burden
Both recommend perphenazine	35%	36%
Only $\hat{\pi}_n$ recommends perphenazine	6%	5%
Only $\hat{\pi}_n$ recommends atypical antipsychotic	2%	31%
Both recommend atypical antipsychotic	57%	28%

Figure 4: Table 4 in Butler et al. (2018)

Discussion

- Butler et al. (2018) propose a strategy for balancing multiple, possibly competing outcomes when estimating a dynamic treatment regime.

- A few other methods for incorporating multiple outcomes into precision medicine have been proposed for various scenerios under various assumptions. However, the literature in this area is relatively small.

- An interesting extension is the multi-decision setting. This is a challenging extension as patient preferences may change over time in response to treatment received and interim outcome experiences.

Introduction

- Almost all methods for estimating individualized treatment rules have been designed to optimize a scalar outcome
- Clinical decision making often requires balancing trade-offs between multiple outcomes
- Examples:
  - Bipolar disorder treatments must manage both depression and mania
  - Antidepressants can prevent depressive episodes but may also induce manic episodes
- Utility functions can be used to summarize multiple outcomes as a single scalar

Setup and Notation

- We will assume two outcomes, but the method can be extended to more
- The available data are $(\mathsf{X}_i, A_i, Y_i, Z_i),\ i = 1, \ldots, n$
  - $\mathsf{X}_i \in \mathcal{X} \subseteq \mathbb{R}^p$ are patient covariates
  - $A \in \mathcal{A} = \{-1, 1\}$ is a binary treatment
  - Y and Z are two real-valued outcomes (higher is better)
- Let $Y^*(A)$ and $Z^*(A)$ be the potential outcomes under treatment a
- We will need the standard causal assumptions
  - Consistency: $Y = Y^*(A)$ and $Z = Z^*(A)$
  - Positivity: $\Pr(A = a|\mathsf{X} = \mathsf{x}) \geq c > 0$
  - Ignorability: $\{Y^*(-1), Y^*(1)\} \perp A \,|\, \mathsf{X}$

Optimal Treatments for Individual Outcomes

- Define $$Q_Y(\mathrm{x}, a) = \mathbb{E}(Y \mid \mathrm{X} = \mathrm{x}, A = a)$$ $$Q_Z(\mathrm{x}, a) = \mathbb{E}(Z \mid \mathrm{X} = \mathrm{x}, A = a)$$
- Under the preceding assumptions we have
  - $d_Y^{\mathrm{opt}}(\mathrm{x}) = \mathrm{argmax}_{a \in \mathcal{A}}\, Q_Y(\mathrm{x}, a)$
  - $d_Z^{\mathrm{opt}}(\mathrm{x}) = \mathrm{argmax}_{a \in \mathcal{A}}\, Q_Z(\mathrm{x}, a)$
- In general, $d_Y^{\mathrm{opt}}(\mathrm{x})$ need not equal $d_Z^{\mathrm{opt}}(\mathrm{x})$
- If both Y and Z are clinically relevant, neither d_Y^{opt} nor d_Z^{opt} may be acceptable

Utility Functions

- Define the composite outcome $U = u(Y, Z)$ for a utility function, u
  - $u : \mathbb{R}^2 \mapsto \mathbb{R}$ is the "goodness" of the outcome pair (y, z)
  - u may be unknown and possibly depend on the covariates
- Define $Q_U(\mathrm{x}, a) = \mathbb{E}(U \mid \mathrm{X} = \mathrm{x}, A = a)$
- The optimal regime with respect to U satisfies

$$\begin{aligned} d_U^{\text{opt}}(\mathrm{x}) &= \underset{a \in \mathcal{A}}{\text{argmax}}\, Q_U(\mathrm{x}, a) \\ &= \underset{a \in \mathcal{A}}{\text{argmax}}\, \mathbb{E}[u\{Y(a), Z(a)\} \mid \mathrm{x}] \end{aligned}$$

- Utility functions which are convex combinations of $Q_Y(\mathrm{x}, a)$ and $Q_Z(\mathrm{x}, a)$ are identifiable under the preceding assumptions

Inverse Reinforcement Learning

- We assume that clinicians act with the goal of optimizing each patient's utility
- Inverse reinforcement learning uses decisions made by an expert to construct a utility function
- We assume that the clinicians are approximately (i.e., imperfectly) assigning treatment according to $d^{\text{opt}}(\mathrm{x})$
  - There would be no need to estimate the optimal treatment policy if the clinician were always able to correctly identify the optimal treatment
- We assume that the probability of making the correct treatment decision depends on individual patient characteristics

$$\Pr\{A = d_U^{\text{opt}}(\mathrm{X}) \mid \mathrm{X} = \mathrm{x}\} = \text{expit}(\mathrm{x}^{\mathsf{T}}\beta)$$

where β is an unknown parameter

Fixed Utility

- We begin by assuming the utility function is constant across patients
- Let the utility function be $u(y, z; \omega) = \omega y + (1 - \omega)z$ for some $\omega \in [0, 1]$
- The optimal ITR for a broad class of utility functions is equivalent to the optimal ITR for a utility function of this form (Butler 2018, Lemma 1)
- Define $Q_\omega(\mathrm{x}, a) = \omega Q_Y(\mathrm{x}, a) + (1 - \omega) Q_z(\mathrm{x}, a)$ and $d_\omega^{\mathrm{opt}}(\mathrm{x}) = \mathrm{argmax}_{a \in \mathcal{A}} Q_\omega(\mathrm{x}, a)$

Fixed Utility, Cont.

- Let $\widehat{Q}_{Y,n}$ and $\widehat{Q}_{Z,n}$ be estimates based on regression models fit to the observed data
- For a fixed value of ω, let

$$\widehat{Q}_{\omega,n}(\mathsf{x}, a) = \omega \widehat{Q}_{Y,n}(\mathsf{x}, a) + (1-\omega)\widehat{Q}_{Z,n}(\mathsf{x}, a)$$

- Define $\widehat{d}_{\omega,n}(\mathsf{X}) = \text{argmax}_{a \in \mathcal{A}}\, \widehat{Q}_{\omega,n}(\mathsf{x}, a)$
- The joint distribution of (X, A, Y, Z) is

$$\begin{aligned} f(\mathsf{X}, A, Y, Z) &= f(Y, Z|\mathsf{X}, A) f(A|\mathsf{X}) f(\mathsf{X}) \\ &= f(Y, Z|\mathsf{X}, A) f(\mathsf{X}) \frac{\exp[\mathsf{X}^{\mathsf{T}}\beta 1\{A = d_{\omega}^{\text{opt}}(\mathsf{X})\}]}{1 + \exp(\mathsf{X}^{\mathsf{T}}\beta)} \end{aligned}$$

Pseudo-likelihood Estimation of Utility Functions

- Assuming that $f(Y, Z|\mathrm{X}, A)$ and $f(\mathrm{X})$ do not depend on ω or β, the likelihood for (ω, β) is

$$\mathcal{L}_n(\omega, \beta) \propto \prod_{i=1}^{n} \frac{\exp[\mathrm{X}^\mathsf{T}\beta 1\{A = d_\omega^{\mathrm{opt}}(\mathrm{X})\}]}{1 + \exp(\mathrm{X}^\mathsf{T}\beta)}$$

- Plugging in $\widehat{d}_{\omega,n}(\mathrm{X})$ for $d_\omega^{\mathrm{opt}}(\mathrm{X})$ yields a pseudo-likelihood
- If we let $\widehat{\omega}_n$ and $\widehat{\beta}_n$ denote the maximum pseudo-likelihood estimators, an estimator of the utility function is $\widehat{u}_n(y, z) = \widehat{u}_n(y, z; \widehat{\omega}_n) = \widehat{\omega}_n y + (1 - \widehat{\omega}_n)z$ and $\mathrm{expit}(\mathrm{X}^\mathsf{T}\widehat{\beta}_n)$ estimates the probability that a patient would be treated optimally

Algorithm

- The pseudo-likelihood is non-smooth in ω, so standard gradient-based optimization can't be used
- For a given value of ω, it is straightforward to compute the profile estimator $\widehat{\beta}_n(\omega)$
- Compute the profile pseudo-likelihood over a grid for ω and select the value yielding the largest pseudo-likelihood
- Finding $\widehat{\beta}_n(\omega)$ can be accomplished using logistic regression

Patient-specific Utility

- In some application domains outcome preferences can vary widely across patients
  - Schizophrenia
  - Pain management
  - etc.
- To accommodate this, we assume that the utility function takes the form $u(y, z; \mathsf{x}, \omega) = \omega(\mathsf{x})y + 1 - \omega(\mathsf{x})z$ where $\omega : \mathsf{X} \mapsto [0, 1]$ is a smooth function
  - e.g., Let $\omega(x; \theta) = \text{expit}(\mathsf{X}^\mathsf{T}\theta)$ where θ is an unknown parameter
- Define $Q_\theta(\mathsf{x}, a) = \omega(\mathsf{x}; \theta)Q_Y(\mathsf{x}, a) + (1 - \omega(\mathsf{x}; \theta))Q_Z(\mathsf{x}, a)$ and define $d_\theta^{\text{opt}}(\mathsf{x}) = \text{argmax}_{a \in \mathcal{A}}\, Q_\theta(\mathsf{x}, a)$

Patient-specific Utility

- For $\widehat{Q}_{Y,n}$, $\widehat{Q}_{Z,n}$ and a fixed value of θ, let

$$\widehat{Q}_{\theta,n}(x, a) = \omega(x; \theta)\widehat{Q}_{Y,n}(x, a) + (1 - \omega(x; \theta))\widehat{Q}_{Z,n}(x, a)$$

  and $\widehat{d}^{\text{opt}}_{\theta,n}(x) = \text{argmax}_{a \in \mathcal{A}}\, \widehat{Q}_{\theta,n}(x, a)$

- We can compute the estimators $(\widehat{\theta}_n, \widehat{\beta}_n)$ by maximizing the pseudo-likelihood

$$\mathcal{L}_n(\theta, \beta) \propto \prod_{i=1}^{n} \frac{\exp[X^T \beta 1\{A = \widehat{d}_{\theta,n}(X)\}]}{1 + \exp(X^T \beta)}$$

- An estimator for the utility function is $\widehat{u}_n(y, z; x) = \omega(x; \widehat{\theta}_n)y + (1 - \omega(x; \widehat{\theta}_n))z$
- An estimator for the optimal decision function is $\widehat{d}_{\widehat{\theta}_n,n}$

Algorithm

- As before, the pseudo-likelihood is non-smooth in θ
- It is again straightforward to compute the profile pseudo-likelihood estimator $\widehat{\beta}_n(\theta)$ for any $\theta \in \mathbb{R}^p$
- It is computationally infeasible to compute $\widehat{\beta}_n(\theta)$ over a grid for moderate p
- Instead we generate a random walk through the parameter space using the Metropolis algorithm (see next slide)
- After generating a chain $(\theta^1, \ldots, \theta^B)$, we select the θ^k that leads to the largest value of $\widetilde{L}_n(\theta^k)$ as the maximum pseudo-likelihood estimator

Algorithm, Cont.

Algorithm 2: Pseudo-likelihood estimation of patient-dependent utility function

Set a chain length, B, fix $\sigma^2 > 0$, and initialize θ^1 to a starting value in $\mathbb{R}^p$;

for $b = 2, \ldots, B$ **do**

  Generate e $\sim N(0, \sigma^2 I)$;

  Set $\widetilde{\theta}^{b+1} = \theta^b + \text{e}$;

  Compute $p = \min\{\widetilde{L}_n(\widetilde{\theta}^{b+1})/\widetilde{L}_n(\widetilde{\theta}^b), 1\}$;

  Generate $U \sim U(0,1)$; if $U \leq p$, set $\theta^{b+1} = \widetilde{\theta}^{b+1}$; otherwise, set $\theta^{b+1} = \theta^b$;

end

Theoretical Results

- We assume that $\Pr\{A = d^{\text{opt}}(\mathsf{x})|\mathsf{X} = \mathsf{x}\} = \text{expit}(\mathsf{x}^{\mathsf{T}}\beta_0)$
- The true utility is $u(y, z; \mathsf{x}, \theta_0) = \omega(\mathsf{X}; \theta_0)y + \{1 - \omega(\mathsf{X}; \theta_0)\}z$ where $\omega(\mathsf{X}; \theta)$ has bounded continuous derivative on compact sets
- $d^{\text{opt}}_{\theta_0}(\mathsf{X}) = d^{\text{opt}}_{\theta}(\mathsf{X})$ almost surely implies $\theta = \theta_0$
- The main theoretical results rely on a number of assumptions
  - A rate of convergence for the estimated Q-functions
    - Automatically satisfied if the Q-functions are estimated using linear or generalized linear models
  - Positive probability of patients with x values near the boundary between where each treatment is optimal

Asymptotic Inference

Theorem

Under regularity conditions, the pseudo-likelihood maximizers $\hat{\beta}_n$ and $\hat{\theta}_n$ satisfy

$$\sqrt{n}\begin{pmatrix} \hat{\theta}_n - \theta_0 \\ \hat{\beta}_n - \beta_0 \end{pmatrix} \rightsquigarrow \begin{pmatrix} U \\ I_0^{-1}\left[Z_A - k_0(Z_Y, Z_Z, U)\right] \end{pmatrix} = \begin{pmatrix} U \\ B \end{pmatrix},$$

where $U = \text{argmin}_u\, \beta_0^{\mathsf{T}} k_0(Z_Y, Z_Z, u)$, *and*

$$\begin{pmatrix} Z_Y \\ Z_Z \\ Z_A \end{pmatrix} \sim N(0, \Sigma_0).$$

A certain semiparametric bootstrap is also consistent in probability.

Asymptotic Inference

Main technical tools:

- The Argmax theorem
- The following for the bootstrap:

Theorem

- *Let H be compact with respect to a metric d and $\mathcal{F} \subset C[H]$ be compact with respect to $\|\cdot\|_H$*
- *For each $f \in \mathcal{F}$, let $u(f) = \operatorname{argmax}_{u \in H} f(u)$, where we arbitrarily choose a value if nonunique*
- *Suppose also that there exists an $\mathcal{F}_1 \subset \mathcal{F}$ such that each $f \in \mathcal{F}_1$ has a unique maximum*
- *Then*

$$\lim_{\delta \downarrow 0} \sup_{f \in \mathcal{F}_1} \sup_{g \in \mathcal{F}: \|f-g\|_H < \delta} d(u(f), u(g)) = 0$$

Parametric Bootstrap

Theorem

- *Assume* $\widehat{\Sigma}_n = \Sigma_0 + o_P(1)$
- *Let* $Z^* \sim N(0, I^{r\times r})$ *where* $r = p + q$, $\widetilde{Z}_n = \widehat{\Sigma}_n Z^* = (\widetilde{Z}_Y^\mathsf{T}, \widetilde{Z}_Z^\mathsf{T}, \widetilde{Z}_A^\mathsf{T})^\mathsf{T}$
- *Define* $\widetilde{U}_n = \text{argmin}_{u\in\mathbb{R}^d} \widehat{\beta}_n^\mathsf{T} \widetilde{k}_n(\widetilde{Z}_Y, \widetilde{Z}_Z, u)$ *and* $\widetilde{B}_n = I_n(\widehat{\beta}_n)^{-1}\{Z_A - \widetilde{k}_n(\widetilde{Z}_Y, \widetilde{Z}_Z, \widetilde{U}_n)\}$
- *Then*

$$\begin{pmatrix} \widetilde{U}_n \\ \widetilde{B}_n \end{pmatrix} \overset{P}{\underset{Z^*}{\rightsquigarrow}} \begin{pmatrix} U \\ B \end{pmatrix},$$

where U *and* B *are as defined on slide 125*

Simulation Studies

- $\mathsf{X} = (X_1, ..., X_5)^\mathsf{T} \sim N(0, \Sigma = 0.5^2 I)$
- $Y = A(4X_1 = 2X_2 + 2) + \epsilon_Y$
  $Z = A(2X_1 - 4X_2 - 2) + \epsilon_Z$
  where $\epsilon_Y \sim \epsilon_Z \sim N(0, 0.5^2)$
- Setting 1:
  - $\Pr\{A = d^{\text{opt}}(\mathsf{x}) \,|\, \mathsf{X} = \mathsf{x}\} = \rho$
- Setting 2:
  - $\Pr\{A = d^{\text{opt}}(\mathsf{X})\} = \text{expit}(0.5 + X_1)$
- Setting 3:
  - $\Pr\{A = d^{\text{opt}}(\mathsf{X})\} = \text{expit}(0.5 + X_1)$
  - $\omega(\mathsf{X}; \theta) = \text{expit}(1 - 0.5X_1)$

Simulation Results

- Value results for simulations where utility (ω) and probability of optimal treatment (ρ) are fixed

n	ω	ρ	Optimal	Estimated ω	Y only	Z only	Standard of care
100	0.25	0.60	1.90 (0.07)	1.75 (0.29)	0.39 (0.12)	1.77 (0.07)	0.39 (0.23)
		0.80	1.90 (0.07)	1.88 (0.07)	0.39 (0.12)	1.77 (0.07)	1.14 (0.21)
	0.75	0.60	1.89 (0.07)	1.69 (0.40)	1.76 (0.08)	0.39 (0.12)	0.40 (0.23)
		0.80	1.89 (0.07)	1.89 (0.07)	1.76 (0.08)	0.39 (0.12)	1.15 (0.21)
200	0.25	0.60	1.90 (0.07)	1.80 (0.25)	0.39 (0.11)	1.77 (0.07)	0.38 (0.17)
		0.80	1.90 (0.07)	1.89 (0.06)	0.39 (0.11)	1.77 (0.07)	1.15 (0.15)
	0.75	0.60	1.90 (0.07)	1.79 (0.26)	1.76 (0.07)	0.38 (0.11)	0.38 (0.17)
		0.80	1.90 (0.07)	1.89 (0.06)	1.76 (0.07)	0.38 (0.11)	1.16 (0.15)
300	0.25	0.60	1.90 (0.07)	1.86 (0.13)	0.37 (0.11)	1.76 (0.08)	0.38 (0.13)
		0.80	1.90 (0.07)	1.89 (0.07)	0.37 (0.11)	1.76 (0.08)	1.14 (0.12)
	0.75	0.60	1.90 (0.06)	1.84 (0.19)	1.76 (0.08)	0.39 (0.11)	0.39 (0.13)
		0.80	1.90 (0.06)	1.90 (0.07)	1.76 (0.08)	0.39 (0.11)	1.15 (0.12)
500	0.25	0.60	1.90 (0.06)	1.88 (0.08)	0.38 (0.11)	1.77 (0.07)	0.37 (0.11)
		0.80	1.90 (0.06)	1.90 (0.06)	0.38 (0.11)	1.77 (0.07)	1.13 (0.09)
	0.75	0.60	1.90 (0.07)	1.88 (0.08)	1.76 (0.08)	0.39 (0.11)	0.37 (0.10)
		0.80	1.90 (0.07)	1.90 (0.07)	1.76 (0.08)	0.39 (0.11)	1.13 (0.09)

Simulation Results

- Value results for simulations where utility (ω) is fixed and probability of optimal treatment is variable

n	ω	Optimal	Estimated ω	Y only	Z only	SoC
100	0.25	1.90 (0.06)	1.72 (0.41)	0.40 (0.11)	1.76 (0.07)	0.33 (0.24)
	0.75	1.90 (0.06)	1.76 (0.29)	1.76 (0.07)	0.38 (0.12)	0.58 (0.24)
200	0.25	1.90 (0.06)	1.84 (0.24)	0.38 (0.11)	1.75 (0.08)	0.32 (0.16)
	0.75	1.90 (0.06)	1.84 (0.16)	1.76 (0.07)	0.38 (0.11)	0.57 (0.16)
300	0.25	1.89 (0.07)	1.88 (0.14)	0.38 (0.11)	1.77 (0.07)	0.32 (0.14)
	0.75	1.90 (0.07)	1.87 (0.09)	1.76 (0.07)	0.39 (0.12)	0.56 (0.14)
500	0.25	1.90 (0.07)	1.90 (0.06)	0.38 (0.11)	1.77 (0.07)	0.33 (0.10)
	0.75	1.90 (0.07)	1.89 (0.08)	1.76 (0.07)	0.39 (0.11)	0.57 (0.10)

Simulation Results

- Value results for simulations where both utility and probability of optimal treatment are variable

n	Optimal	Estimated ω	Y only	Z only	Standard of care
100	1.74 (0.06)	1.53 (0.19)	1.59 (0.07)	0.44 (0.11)	0.51 (0.21)
200	1.73 (0.06)	1.61 (0.13)	1.59 (0.07)	0.44 (0.10)	0.51 (0.15)
300	1.74 (0.06)	1.64 (0.12)	1.59 (0.07)	0.44 (0.10)	0.50 (0.13)
500	1.74 (0.06)	1.68 (0.09)	1.59 (0.07)	0.43 (0.10)	0.50 (0.09)

Misspecified Model for the Utility Function

- Let the true underlying utility function be $u(y, z; bx, \theta) = \omega \mathrm{x}; \theta)y + \{1 - \omega(\mathrm{x}; \theta)\}z$
- Where $\omega(\mathrm{x}; \theta) = \mathrm{expit}(1 + x_1^2 + \mathrm{x}^{\mathsf{T}}\theta_0)$
- Consider a misspecified model fit to estimate the utility function containing only an intercept, X_1, X_2, X_3, and X_4
  - i.e., one important covariate and a squared term are omitted from the model for the utility function

n	Optimal	Correct	Misspecified	Standard of Care
100	1.86 (0.07)	1.61 (0.21)	1.64 (0.20)	0.59 (0.23)
200	1.85 (0.07)	1.68 (0.16)	1.69 (0.17)	0.57 (0.16)
300	1.86 (0.07)	1.72 (0.13)	1.74 (0.13)	0.57 (0.13)
500	1.86 (0.07)	1.77 (0.10)	1.76 (0.11)	0.58 (0.10)

Analysis of STEP-BD SCP Data

- Included an observational study with 1437 patients having bipolar disorder (Sachs et al, 2007, *NEJM*).
- Using the proposed method, we were able to estimate an improved decision rule which led to a 7% improvement (p-value < 0.0001).
- Both increased age and history of substance abuse were important factors leading to lower recommended use of antidepressants.
- If we selected the two outcomes to be depression and side effect burden, we obtain an improvement of 9% (p-value < 0.001).

Conclusions

- We can estimate patient utilities if we assume that clinicians make treatment decisions with the goal of maximizing each patient's utility
- Accounting for patient specific utilities can improve outcomes over standard of care
- Early results suggest the method is robust to utility model misspecificaion, but more research is needed
- The approach could be extended to multiple decision times, more than two outcomes, and more than two possible treatments
- A Bayesian approach could be developed to handle the non-smooth pseudo-likelihood