Causality in Machine Learning: Class projects

Robert Ness

This course focuses on the application of causal inference theory in generative machine learning. The course project requires you to implement a cutting-edge causal machine learning workflow.

You are implementing your project in groups of 2-3 students. The final deliverable is a Github repository that includes notebooks with code and description, and a presentation of your results to the class.

Please choose one of the following projects.

Deep causal variational inference

This project is ideal for students interested in deep learning.

In this project, you will train a supervised variational autoencoder using Deepmind's dSprites dataset.

dSprites is a dataset of sprites, which are 2D shapes procedurally generated from 5 ground truth independent "factors." These factors are color, shape, scale, rotation, x and y positions of a sprite.

All possible combinations of these variables are present exactly once, generating N = 737280 total images.

Factors and their values:

- Shape: 3 values {square, ellipse, heart}
- Scale: 6 values linearly spaced in (0.5, 1)
- Orientation: 40 values in (0, 2π)
- Position X: 32 values in (0, 1)
- Position Y: 32 values in (0, 1)

There is a sixth factor for color, but it is white for every image in this dataset.

The purpose of this dataset was to evaluate the ability of disentanglement methods. In these methods, you treat these factors as latent and then try to "disentangle" them in the latent representation.

However, in this project, you will not treat these factors as latent, and include them as labels in the model training. Further, you will invent a causal story that relates these factors and the images in a DAG such as in the following figure:

You would implement this causal story as a structural causal model of the form:

The image variable will be a 64 x 64 array. The noise term for the image variable will be the traditional Gaussian random variable. The structural assignment g for the image variable will be the decoder.

Deliverables

- Create a DAG and an SCM articulate a causal story relating shape, orientation, scale, X, Y, and the data.
- Resample the dataset so you get a new dataset with an empirical distribution that is faithful to the DAG and is entailed by the SCM (instructor will provide guidance).
- Using Pyro, implement the causal VAE. The instructor will provide you with a notebook that implements a primitive version of the model, including code for setting up data loaders, reshaping of tensors, and the learning algorithm. The instructor will provide you with guidance on how to incrementally expand this model.
- Use the trained model to answer some counterfactual queries, for example, "given this image of a heart with this orientation, position, and scale, what would it have looked like if it were a square?"

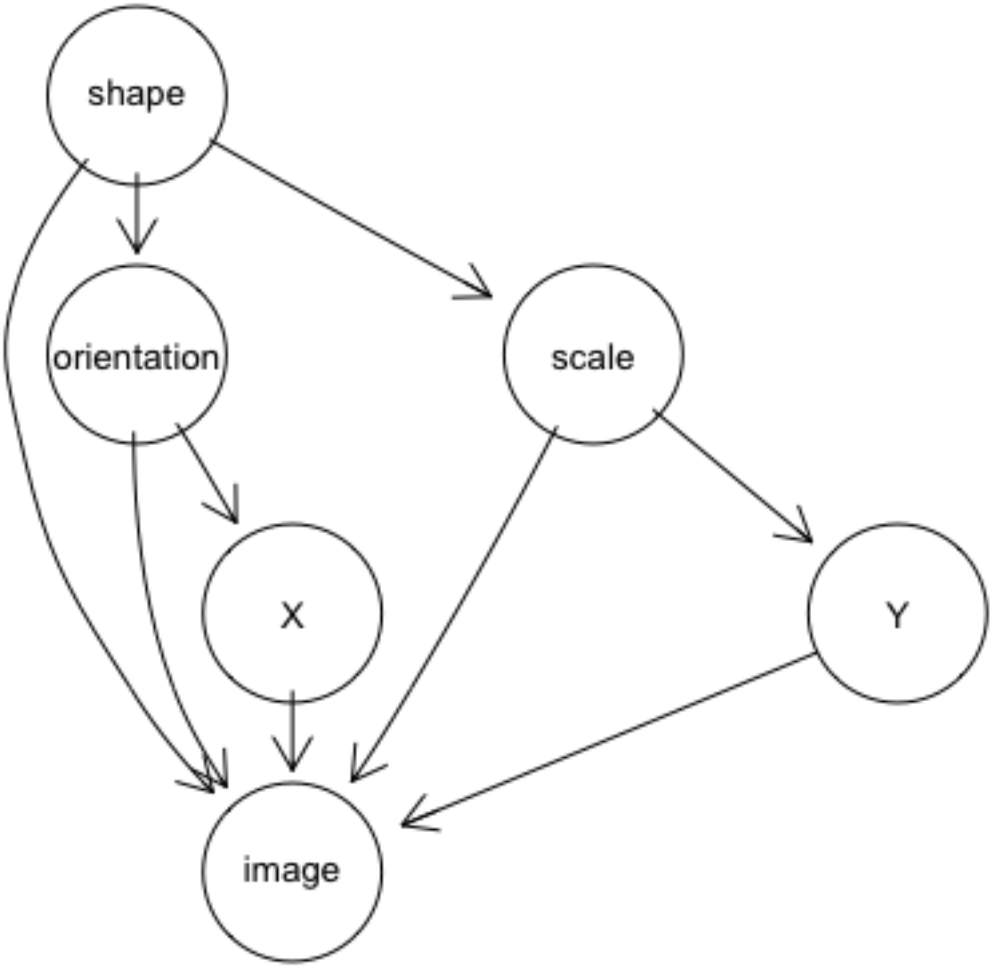

Figure 1: vae_dag

$$
\begin{aligned}
\text{shape} &= f_{\text{shape}}(N_{\text{shape}}) \\
\text{orientation} &= f_{\text{orientation}}(\text{shape}, N_{\text{orientation}}) \\
\text{scale} &= f_{\text{scale}}(\text{shape}, N_{\text{scale}}) \\
\text{X} &= f_{\text{X}}(\text{orientation}, N_{\text{X}}) \\
\text{Y} &= f_{\text{Y}}(\text{scale}, N_{\text{Y}}) \\
\text{image} &= g(\text{orientation}, \text{scale}, \text{X}, \text{Y}, \text{image}, N_{\text{image}})
\end{aligned}
$$

Figure 2: vae_scm

Counterfactual policy evaluation with Open AI gym

This project is ideal for students who are interested in reinforcement learning.

In this project, you will take an environment from OpenAI gym and demonstrate a counterfactual evaluation use case.

Deliverables

- Select an environment with a discrete state and action space (e.g. the frozen lake models)
- Reimplement the transition function in this environment as a structural causal model. Use the technique described in class and in the reference below.
- Create two policies for this environment. You can train them using a standard RL algorithm. Alternatively, you can make the policy up, though it should still be a sensible policy; for example in the frozen lake environment, you could make up a policy that always tries to take a step that minimizes the distance to the goal but still tries to avoid holes.
- Pretend this policy was in production, and generate logs.
- With the second policy, use counterfactual evaluation with the structural causal model to evaluate how well the second policy would have performed had it been in production. Compare the counterfactual expectation of reward, to the expected reward of the in-production model.

Counterfactual Off-Policy Evaluation with Gumbel-Max Structural Causal Models

Causal modeling of a dynamic system

This project is ideal for students who are interested in doing applied causal machine learning in these fields, such as economics, game theory, computational ecology, and systems biology.

In these fields, it is standard to use mathematic models of how the behavior of a system evolves in time (e.g., ordinary differential equations) to model dynamic systems. However, modelers are often interested in how an intervention causes a system to behave at equilibrium – where the system dynamics have stabilized. For example, in economics, there are differential equations that define the rate of change of price given supply and demand. However, an economist is generally interested predicting the effect of an intervention on supply (e.g., through a tariff) will affect the price of a product at equilibrium, they are less concerned with predicting exactly how the intervention will cause the price to fluctuate before reaching equilibrium.

Another basic example is predator-prey modeling. Imagine we want to predict how a rabbit population control policy will affect local ecology. Suppose that in this ecology, the only predator of rabbits is foxes, and we are concerned that rabbit-hunting might adversely affect the fox population.

To model this system, we could model the rate of change in the rabbit population and the rate of change in the fox population. Our causal reasoning would emerge in how we specified these rates of change. For example, we could say that rabbit birth rate depends on the current population of rabbits; in causal terms, this means that the presence of rabbits causes the birth of more rabbits. Similarly, we could say that the rabbit death rate is proportional to the current population of rabbits times the current population of foxes (i.e., the number of potential rabbit-fox interactions); in causal terms, this says that the presence of rabbits and foxes causes the death of rabbits.

The challenge is using a causal Bayesian network (CBN) or a structural causal model (SCM) to capture the equilibrium of such a system. Imagine our rabbit population control policy kept the rabbit population fixed at a specific value "x" (e.g., we took away a rabbit each time one were born, and added a rabbit each time one died). We would model this intervention by changing the dynamic model so that the rabbit population was constant at "x". At the same time, we apply the intervention do(R = x) to the equilibrium CBN or SCM. Ideally, we would the CBN or SCM in such a way that when the dynamic model reached equilibrium, the

number of foxes would be the same as the predicted number given by the CBN/SCM under intervention. It turns out that specifying a CBN or SCM that can do this is non-trivial.

Deliverables

1. Use Pyro to implement the Gillespie algorithm on a simple three molecule biochemical model. The Gillespie algorithm is a probabilistic way of modeling dynamic interactions between molecules. The instructor will provide the model and algorithm pseudocode.
2. Use Pyro code to model the same system as a structural causal model. The instructor will provide the model and the pseudocode.
3. Use the `do` operator to compare interventions on both models, and analyze the conditions under which the model interventions match and the conditions under which the model interventions differ.

Schema network proof-of-concept

This project is suitable for those interested in applying causal modeling and causal discovery to reinforcement learning.

Schema networks are a model-base reinforcement learning approach proposed by Vicarious AI.

In this task you will implement a basic proof-of-concept of the object-oriented Markov decision processes described in section 3.2 in the reference paper. The object here is too implement the basic modeling abstractions. You are not expected to provide an implementation sophisticated enough to model the Atari games described in the reference.

Deliverables

- Provide a well-organized and well-documented repository of schema network extractions.
- Transition function must be represented as Pyro programs
- Students will be judged on Python code-quality and quality of modeling abstractions.
- Provide a notebook illustrating a run of a toy schema network model as a proof-of-concept

Deconfounding film predictions

This project is ideal for people who want experience applying standard probabilistic modeling to a practical causal inference question.

In this paper, you will implement a method described in two papers by Yixin Wang et al.; The Blessings of Multiple Causes and The Deconfounded Recommender:A Causal Inference Approach to Recommendation. In these papers the authors propose a deconfounding technique using a class of models called probabilistic factor models. The core of the approach in each paper is the same, though this description focuses on the problem of predicting box office revenue, described in the first paper. The second paper describes a recommendation system application, and is a good choice because it has a clearer standard of evaluation.

When you read these papers, you will see that they are premised on the potential outcomes framework, which is different from the approach we've taken in class. Below I provide some guidance for how to implement things our way.

In the box-office prediction problem, your goal is to predict revenue given which actors will be in a film. A naive approach would be just to train a predictive model, such as a neural net or linear regression, mapping an actor vector to a revenue outcome. However, if the goal is to use this model to choose a set of actors for a film, then this is an intervention problem – instead of `condition({"actor": "Brad Pitt"}`, you `do({"actor": "Brad Pitt"})` when you make this decision.

Your goal therefore is to find $E(Y|do(A_j = 1)) - E(Y|do(A_j = 0))$ for a given actor (or for multiple actors). However, there are unobserved confounders. For example Samantha Bond played Ms. Moneypenny in several James Bond films, all of which made large amounts of money. It would be silly however to suggest that she has a large causal effect on box office revenue, and that she should be added to a new film. We need a model that deconfounds things like whether or not a film is a Bond film.

The paper proposes the following technique to deconfounding this prediction. Let Y be revenue and $A_1...A_m$ be binary variables for the presence or absence of m actors in the database.

1. Fit a probabilistic model M that assumes that the As have a common latent cause Z.
2. Verify this is a good model using a posterior predictive check (you won't have to do this).
3. Augment the data by using the model to estimate $\hat{Z} = E_M(Z|A_1, ..., A_J)$, a vector of predictive values for Z.
4. Estimate the intervention distribution by adjusting for $\hat{Z}$: $P(Y|do(A_j = 1)) = \sum_{\hat{Z}=\hat{z}} P(Y|do(A_j = 1), \hat{Z} = \hat{z})P(\hat{Z} = \hat{z})$.
5. Predict causal effects in terms of box office revenue using $E(Y|do(A_j = 1)) - E(Y|do(A_j = 0))$.

How do we model this?

Data

Use the TMDB 5000 Movie Dataset dataset, as in the paper, with log revenue as the prediction target. There are multiple variables you can use as causes, but you should just stick to actors. I suggest starting by taking a small set of actors, and subsetting the data to films where at least one of these actors is present, then gradually expanding the set of actors and the data subset. Note that in this approach there should not be any edges between actor-causes.

Intuition

The intuition for how the technique works is that $\hat{Z}$ behaves as a propensity score, i.e. a statistic that renders the A_j independent of unobserved confounders. By construction, it renders all A_j conditionally independent of one another. If it accomplishes this, then conditioning on $\hat{Z}$ blocks all backdoor paths of dependence between the A_j's through latent confounders. In other words, $\hat{Z}$ is a proxy for unobserved latent confounders. It seems like magic, but the approach succeeds by relying on the multiplicity of actor-causes to even estimate $\hat{Z}$; in this respect it is an example of a statistic that is not identifiable with univariate data but is with multivariate data. It is not a free lunch, the trade-off is that you reduce confounder bias but causal effect estimates have more variance.

Model

We will use a generative modeling approach slightly different than that described in the paper, and more similar to the above variational autoencoder project described above. The forward model of the data should look like the following (very rough) pseudocode:

```
def model(A_vals, Y_vals):
    # gamma, alpha are hyperparameters
    theta = sample(p(·|gamma))
    for each i
        Zi = sample(p(·|alpha))
        for each j in J # J causes
            Aij = sample(p(·|Z[i], theta[j]), obs=A_vals[i, j])
        Yi ~ sample(p(·|Ai1 ,.., AiJ, Zi), obs=Y_vals[i])
```

Above `obs` does the same as `condition`. Note that in the line `Yi ~ sample(p(·|Ai1,..,AiJ, Zi), obs=Y_vals[i])`, once you have sampled the Z, this is just like standard supervised prediction. It is

the relationship between Z and A where the sophisticated modeling happens. The following guide function sketches the inference of Z given A. Note that we do not use Y to infer Z.

```
def guide(A_vals, Y_vals):
    # eta are hyperparameters you train using SVI
    for each i:
        Zi = q(·| A_vals[i, 1], ..., A_vals[i, J], eta)
```

Do not spend time trying to find a good model of the A-Z relationship. The reference paper found that probabilistic PCA (Edward example) Poisson factorization (Edward example) and deep exponential family (Pyro example) worked well. You should train the model using variational inference in Pyro. In addition to the linked examples above, look at the Pyro examples for inspiration, particularly the VAE, semi-supervised VAE, and LDA examples.

Deliverables

- Implement the predictive model M in Pyro. Provide a well documented repo and reproducible notebooks
- Estimate causal effects for each actor $E_M(Y|do(A_j = 1)) - E_M(Y|do(A_j = 0))$ both by adjusting for Z and by do-calculus. Compare the two outcomes and make sure they align.
- Implement a comparison predictive model M_2 and estimate the effects for each actor $E_{M_2}(Y|A_j = 1) - E_{M_2}(Y|A_j = 0)$. This should be a simple model with the same structural form as the `Yi = sample(p(·|Ai1 ,.., AiJ, Zi), obs=Y_vals[i])` line in the original model.
- Rank actors by the biggest difference between the two models. Sanity check your results against the original authors' results.
- BONUS: Add genre into your model, and answer questions where you have do(genre = scifi) or other.
- BONUS: Covert you model to an SCM using the method described in class, and answer a counterfactual question like, "How much more would movie X have made if they cast A instead of B"?

COMPAS Analysis

COMPAS is a commercial algorithm used to predict the risk of a criminal defendent's risk of becoming a recidivist – a term used to describe criminals who reoffend. ProPublica released an article and accompanying analysis that showed that black defendants were more likely than white defendants to be incorrectly judged to be at a higher risk of recidivism, and that white defendants were more likely than black defendants to be incorrectly flagged as low risk.

In this project you implement the counterfactual estimation methods discussed in the references: * Kusner, Matt J., et al. "Counterfactual fairness." Advances in Neural Information Processing Systems. 2017. * Zhang, Junzhe, and Elias Bareinboim. "Equality of opportunity in classification: A causal approach." Advances in Neural Information Processing Systems. 2018.

Deliverables

- Using the COMPAS data set, use Pyro to implement the model in Zhang et al.
- Compute the counterfactual fairness quantity described in Kusner et al.
- Estimate counterfactual direct error rate, the counterfactual indirect error rate, and counterfactual spurious error rate. You may use any method (do-calculus, estimation procedures described in paper). You may have to adjust the model for these quantities to be identifiable.
- Using algorithm 3 in Zhang et al to train a predictive algorithm in Pyro that removes direct discrimination.
- Compare the predictive performance and the counterfactual error rates of your algorithm to that of COMPAS

Causal bandits (WIP)

Causal reasoning in the context of bandit algorithms and general online learning systems.