AA 598: Decision-making and Control for Safe Interactive Autonomy

Homework 1— (Recommended) Due date: Friday October 18th
Starter code: `https://github.com/UW-CTRL/AA598-aut24`

Goal. To become familiar with basic supervised learning and generative modeling concepts and their uses in predicting multimodal outcomes. Additionally, to become familiar with basic PyTorch utilities.
Dataset. In this homework, we will work with some very simple synthetically generated 1D data, `wave_data` and `multimodal_data`. Fig. 1 shows a sample from each dataset. Each dataset consists of *trajectory history* (shown in blue), and a *trajectory future* (not blue). For `wave_data`, there is only one possible future. For `multimodal_data`, there are multiple possible futures. While the `multimodal_data` is very simple, the format is representative of human interaction data where the history could be observation history of the environment, and the future could be the future states/controls of agents in the environment.
Note. Please keep your responses brief. Long-winded responses will result in zero points.

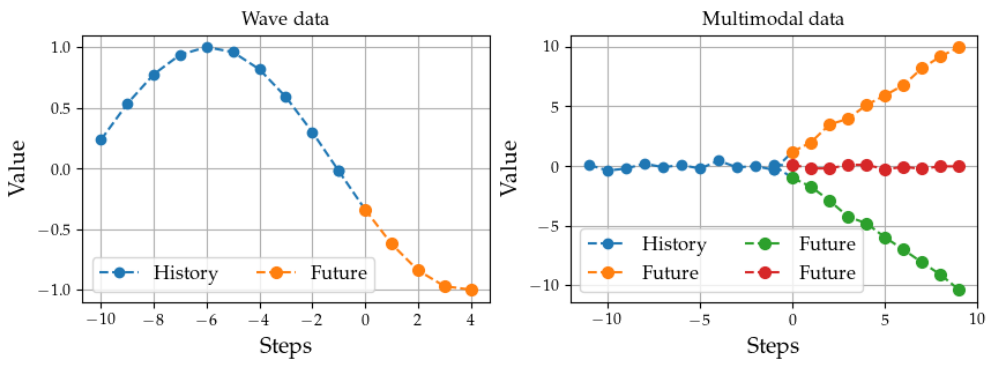

Fig. 1: Example of trajectory data that you are working with. Left: `wave_data`. Right: `multimoda_data`.

1 Basic regression problem

In this problem, you will train a neural network to perform a simple regression problem. Given a trajectory history x, we want to predict a future trajectory y. We want to learn a function $f_\theta(\cdot)$ parameterized by θ such that $y \approx f_\theta(x)$. First we try on

1.1 Multilayer perception (MLP)

We can just learn an MLP that treats the entire trajectory history as a concatenated vector and outputs the future trajectory, again, as a concatenated vector.

(a) Implement a simple MLP ($\approx$ 2 layers) and train it on the `wave_data`. Describe your MLP architecture. Report the test loss and include some plots showing the prediction on the test set. *Briefly* justify your choice of hyperparameters.

(b) *Briefly* describe some potential benefits and limitations of using MLPs as the core architecture for human behavior prediction.

1.2 Long-short term memory (LSTM)

It is common to use recurrent neural networks (RNN) to process time-series data. The idea is to process the inputs sequentially—this makes a lot of sense if future inputs depend only on previous inputs. LSTMs are a particular type of RNN [1].

(a) Implement a simple prediction model that uses an LSTM to encode the trajectory history and another LSTM to decode the output. Describe your LSTM architecture. Report the test loss and include some plots showing the prediction on the test set. *Briefly* justify your choice of hyperparameters.

(b) *Briefly* describe some potential benefits and limitations of using RNNs/LSTMs as the core architecture for human behavior prediction?

(c) What happens if you choose a prediction horizon that is longer than the training data? Include a plot of your prediction with an extended prediction horizon.

2 Regression on multimodal outcomes

Let's try the same type of models you have built but on data where for similar trajectory histories, the future trajectories could have multiple outcomes. This is what we refer to as *multimodality*.

(a) Train another MLP and LSTM prediction model using the `multimodal_data`. Report the test loss and include some plots showing the prediction on the test set.

(b) Are these models successful in prediction the future trajectories well? Are the predictions what you would expect? *Briefly* justify your answer.

Problem 3: Conditional Variational Autoencoder (CVAE)

Rather than performing regression (i.e., learn a function approximation such that $y \approx f_\theta(x)$), we instead can learn a conditional distribution such that $y \sim p_\theta(y \mid x)$ where the distribution is parameterized by θ. Specifically, we want to learn p_θ such that it matches the data distribution that generated the dataset. We say that we want to learn a *generative model*, a model that can generate new samples that is similar to the training data.

While there are many different ways to create generative model (we discussed in class), in this homework, we will look into conditional variational autoencoders due to its simplicity and also its use of a *latent space* to capture salient information.

(a) Read (Multimodal Deep Generative Models for Trajectory Prediction: A Conditional Variational Autoencoder Approach [2], in particular Section III.A. Prove Equation (2) from the paper.

Ultimately, to train a CVAE, we seek to minimize the following objective:

$$\mathcal{L}(x,y;\theta,\varphi,\phi) = \mathbb{E}_{(x,y)\sim p_{\mathcal{D}}(x,y)}\left[\underbrace{-\mathbb{E}_{q_\varphi(z\mid x,y)}\left[\log \overbrace{p_\phi(y \mid x,z)}^{\text{Decoder}}\right]}_{\text{Negative log-likelihood}} + \underbrace{\mathcal{D}_{\text{KL}}\left[\overbrace{q_\varphi(z \mid x,y)}^{\text{Importance weight}} \;||\; \overbrace{p_\theta(z \mid x)}^{\text{Encoder}}\right]}_{\text{KL divergence}}\right] \tag{1}$$

Describing each term in the loss:

(a) $p_\theta(z \mid x)$: The *encoder* that takes input x and outputs a distribution over z (or parameters of the distribution). The latent space can either be *continuous* or *discrete*. We explore both in this homework.

(b) $p_\phi(y \mid x,z)$: The *decoder* that takes input x and latent variable z, and outputs a distribution over y (or parameters of the distribution). In this homework, we let $p_\phi(y \mid x,z) = \mathcal{N}(\mu_\phi(x,z), \Sigma_\phi(x,z))$ where μ_ϕ and Σ_ϕ are neural networks. Note: for simplicity, we let $\Sigma_\phi(x,z) = \text{diag}(\log f_\phi(x,z))$ as this ensures that Σ_ϕ is positive definite.

(c) $q_\varphi(z \mid x,y)$: The importance weight which encodes what are likely values for z given x and y. This importance weight is used to ensure that we are learning a meaningful $p_\theta(z \mid x)$. Like $p_\theta(z \mid x)$, $q_\varphi(z \mid x,y)$ can be either *continuous* or *discrete*.

(d) The negative log-likelihood term measures how likely the data is under the distribution $p_\phi(y \mid x,z)$ when $z \sim q_\varphi(z \mid x,y)$. We take the mean over samples from $q_\varphi(z \mid x,y)$ where (x,y) are samples from the training dataset.

(e) The KL divergence measures how similar distribution p is to q. Intuitively, the KL divergence term encourages $p_\theta(z \mid x)$ and $q_\varphi(z \mid x,y)$ to be similar so that $p_\theta(z \mid x)$ can generate z that would likely generate y.

Take a look at the starter code. It includes code (some for you to fill out) for constructing a CVAE, training one, and visualizing the results. We will be using MLP for the encoder and decoder, but this can easily be swapped for different neural architectures.

2.1 Continuous latent space

In the continuous latent space case, $p_\theta(z \mid x)$ and $q_\varphi(z \mid x, y)$ are multivariate normal distributions.

(a) Complete the `ContinuousCVAE` class. Each function should be less than 5 lines long.

(b) Train the continuous CVAE model. Try training several times and test out different hyperparameters to get a sense of how the results change. *Briefly* summarize your results and insights. Include a plot of the prediction.

2.2 Discrete latent space

In the discrete latent space case, $p_\theta(z \mid x)$ and $q_\varphi(z \mid x, y)$ are one-hot categorical distributions, and a Gumbel-Softmax [3, 4] is used to parameterize the latent space to make it differentiable. A full implementation is provided.

(a) Take a look at the `DiscreteCVAE` class. Given `latent_dim` and `num_categories`, what shape is the latent vector? How many possible values can it take on? Give an example of what values a latent vector could take on.

(b) Train the discrete CVAE model. Try training several times and test out different hyperparameters and latent space sizes to get a sense of how the results change. *Briefly* summarize your results and insights. Include a plot of the prediction.

(c) Comparing your results with the continuous and discrete CVAE. What are some benefits/drawbacks from using each?

References

[1] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, 1997.

[2] B. Ivanovic, K. Leung, E. Schmerling, and M. Pavone, "Multimodal deep generative models for trajectory prediction: A conditional variational autoencoder approach," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 295–302, 2021.

[3] E. Jang, S. Gu, and B. Poole, "Categorial reparameterization with gumbel-softmax," in *Int. Conf. on Learning Representations*, 2017.

[4] C. J. Maddison, A. Mnih, and Y. W. Teh, "The concrete distribution: A continuous relaxation of discrete random variables," in *Int. Conf. on Learning Representations*, 2017.