Runqiu Ye
Stanford CS299
Problem Set #4
08/07/2024

Problem Set #3: EM, Deep Learning, & Reinforcement Learning

Problem 1 Neural Networks: MNIST image classification
Implement a simple convolutional neural network to classify grayscale images of handwritten digits from the MNIST dataset. The starter code splits the set of 60000 training images and labels into a sets of 59600 examples as the training set and 400 examples for the dev set. To start, implement convolutional neural network and cross entropy loss, and train it with the provided dataset. The architecture is as follows:

(a) The first layer is a convolutional layer with 2 output channels with a convolution size of 4 by 4.

(b) The second layer is a max pooling layer of stride and width 5 by 5.

(c) The third layer is a ReLU activation layer.

(d) After the four layer, the data is flattened into a single dimension.

(e) The fifth layer is a single linear layer with output size 10 (the number of classes).

(f) The sixth layer is a softmax layer that computes the probabilities for each classes.

(g) Finally, we use a cross entropy loss as our loss function.

The cross entropy loss is

$$CE(y, \hat{y}) = -\sum_{k=1}^{K} y_k \log \hat{y}_k,$$

where $\hat{y} \in \mathbb{R}^K$ is the vector of softmax outputs from the model for the training example x, and $y \in \mathbb{R}^K$ is the ground-truth vector for the training example X such that $y = [0, \ldots, 0, 1, 0, \ldots, 0]^T$ contains a single 1 at the position of the correct classes.
We also use mini-batch gradient descent with a batch size of 16. Normally we would iterate over the data multiple times with multiple epochs, but for this assignment we only do 400 batches to save time.

(a) Implement functions within `p01_nn.py`.

(b) Implement a function that computes the full backward pass.

■

Problem 2 Off Policy Evaluatino And Causal Inference

Need methods for evaluating policies without actually implementing them. This task is off-policy evaluation or causal inference.

For this problem, consider MDPs with a single timestep. Consider universe of states S, actions A, a reward function $R(s, a)$ where s is a state and a is an action. We often only have a subset of a in our dataset. For eaxmple, each state s could represent a patient, each action a could represent which drug we prescribe to that patient and $R(s, a)$ be their lifespan after prescribing that drug.

A policy is defined as $\pi_i(s, a) = p(a \mid s, \pi_i)$ We are given observational dataset of $(s, a, R(s, a))$ tuples. Let $p(s)$ be the density for the distribution of the state s values wihin the dataset. Let $\pi_0(s, a) = p(a \mid s)$ within our observational data. π_0 corresponds to the baseline policy in observational data. Also given target policy $\pi_1(s, a)$ which gives the conditional probability $p(a \mid s)$ in our optimal policy. Our goal is to computer expected value of $R(s, a)$ in the same population as our observatinoal data, but with a policy of π_1 instead of π_0. In other words, we are trying to compute

$$\mathop{\mathbb{E}}_{\substack{s\sim p(s)\\ a\sim\pi_1(s,a)}}[R(s,a)] = \sum_{(s,a)} R(s,a)p(s,a) = \sum_{(s,a)} R(s,a)p(s)p(a \mid s) = \sum_{(s,a)} R(s,a)p(s)\pi_1(s,a).$$

Simplifying Assumptions: We will assume that each action has a non-zero probability in the observed policy $\pi_0(s, a)$. In other words, for all actions a and states s, $\pi_0(s, a) > 0$.

Regression: The simplest possible estimator is to directly use our learned MDP parameters to estimate our goal. This is called regression estimator. While training our MDP, we laern an estimator $\hat{R}(s, a)$ that estiamtes $R(s, a)$. We can now directly estimate

$$\mathop{\mathbb{E}}_{\substack{s\sim p(s)\\ a\sim\pi_1(s,a)}}[R(s,a)]$$

with

$$\mathop{\mathbb{E}}_{\substack{s\sim p(s)\\ a\sim\pi_1(s,a)}}[\hat{R}(s,a)].$$

If $\hat{R}(s, a) = R(s, a)$, then this estimator is trivially correct. We will now consider alternative approaches and explore why you might use one estimator over another.

(a) **Important Sampling:** Let $\hat{\pi}_0$ be an estimate of the true π_0. The *importance sampling estimator* uses that $\hat{\pi}_0$ and has the form

$$\mathop{\mathbb{E}}_{\substack{s\sim p(s)\\ a\sim\pi_0(s,a)}}\left[\frac{\pi_1(s,a)}{\hat{\pi}_0(s,a)}R(s,a)\right].$$

We now show that if $\hat{\pi}_0 = \pi_0$, then the importance sampling estimator is equal to

$$\mathop{\mathbb{E}}_{\substack{s\sim p(s)\\ a\sim\pi_1(s,a)}}[R(s,a)].$$

Proof. If $\hat{\pi}_0 = \pi_0$, then

$$\begin{aligned}\mathop{\mathbb{E}}_{\substack{s\sim p(s)\\ a\sim\pi_0(s,a)}}\left[\frac{\pi_1(s,a)}{\hat{\pi}_0(s,a)}R(s,a)\right] &= \sum_{(s,a)}\frac{\pi_1(s,a)}{\hat{\pi}_0(s,a)}R(s,a)p(s)\pi_0(s,a)\\ &= \sum_{(s,a)}\pi_1(s,a)R(s,a)p(s)\\ &= \mathop{\mathbb{E}}_{\substack{s\sim p(s)\\ a\sim\pi_1(s,a)}}[R(s,a)],\end{aligned}$$

as desired. □

Note that this estimator only requires us to model π_0 as we have the $R(s,a)$ values in the observational data.

■