MATH 5050: Extra Credit Homework (Due Monday, April 28)

In Problem 1, we implement the generative AI approach we learned in class, but on the smallest possible non-trivial state space: $\{0,1\}$. Before I state the questions, let me go through the theory. This will be a baby version of the approach that uses the Ornstein-Uhlenbeck process we saw in class.

Consider the state space $\{0,1\}$ and suppose the probabilities on this space are given by $\mathbb{P}(0) = 3/4$ and $\mathbb{P}(1) = 1/4$. Suppopse though that we do not know these numbers. Our goal is to estimate these probabilities. We do not want to do this by computing the relative frequencies of 0 and 1 in the given samples. Instead, we want to follow the noising and denoising procedure that generative AI uses.

For the forward Markov chain, we choose to use the transition matrix

$$P = \begin{bmatrix} 7/9 & 2/9 \\ 2/9 & 7/9 \end{bmatrix}. \tag{1}$$

Since all the entries are positive, the Markov chain is irreducible and has a unique invariant measure Φ_∞.

Let us denote by N the terminal time for the forward Markov chain. We will choose N so that by this time, the Markov chain is very close to the invariant measure. As we have seen in 5040, the probabilities $\Phi_n = (\mathbb{P}(X_n = 0), \mathbb{P}(X_n = 1))$ converge exponentially fast to the invariant measure. Therefore, N will not be a large number.

As we said above, we are pretending we do not know the initial probabilities $\Phi_0 = (3/4, 1/4)$. We want to learn the transition probabilities $\overleftarrow{P}_{n,u,v}$ of the reverse Markov chain. Things simplify a bit because the state space consists of the values 0 and 1. Namely, $\overleftarrow{P}_{n,u,v}$ is the probability that $X_{n-1} = v$, given that $X_n = u$. But since X_{n-1} takes only the values 0 or 1, it is a Bernoulli random variable. For a Bernoulli random variable, the probability it takes the value 1 is equal to the mean of the Bernoulli random variable. Therefore,

$$\overleftarrow{P}_{n,u,1} = \mathbb{E}[X_{n-1}|X_n = u].$$

Another simplification is that we only need to compute $\overleftarrow{P}_{n,u,1}$, because $\overleftarrow{P}_{n,u,0} = 1 - \overleftarrow{P}_{n,u,1}$. Let us, therefore, denote

$$s(n,u) = \overleftarrow{P}_{n,u,1} = \mathbb{E}[X_{n-1}|X_n = u].$$

By what was mentioned in the class about conditional expectations, s_n is the function that minimizes

$$\mathbb{E}\big[|X_{n-1} - s(n, X_n)|^2\big]. \tag{2}$$

This turns the task of estimating $\overleftarrow{P}_{n,u,v}$ into a regression problem.

For simplicity, we will use Machine Learning with just one layer. Also, when in the case of an SDE-based generative AI, time is continuous and, as we saw in clas, we have to sample the times at which we sample the forward stochastic process. Since here time is discrete and we are only looking at a small number of steps N, we will go ahead and estimate the transition probabilities $s(n,u)$ of the backward Markov chain for every $n \in \{1, \dots, N\}$.

We thus have N estimation problems, one for each $n \in \{1, \dots, N\}$. For each such n, we consider estimators of $s_\theta(u)$ of the form

$$s_\theta(u) = \phi(wu + b), \tag{3}$$

where, $\theta = (w, b)$ are the two parameters we need to estimate. The function ϕ is given by $\phi(x) = \min(1, \max(0, x))$. So $\phi(x) = x$ if x is between 0 and 1, $\phi(x) = 0$ if $x < 0$, and $\phi(x) = 1$ if $x > 1$. In other words, ϕ is almost like a ReLU function, but it also cuts things off when they go above 1. This is to make sure that $s_\theta(u)$ is a number between 0 and 1, since it is supposed to be estimating a probability.

Before describing the estimation procedure, let us describe where the data comes from.

Since this is an exercise to see how things work, we will generate the data ourselves. This is the only time when we will use the knowledge of the probabilities $\mathbb{P}(0) = 3/4$ and $\mathbb{P}(1) = 1/4$. Thus, we generate a number M of random samples of the initial state $X(0)$ according to these probabilities: 0 with probability 3/4 and 1 with probability 1/4. Let us denote these samples by $a_1, \dots, a_M$.

Next, we use this data to estimate the transition probabilities $s(n, u)$. For each $n \in \{1, \dots, N\}$, we run through our initial samples a_i and, for each such initial state, we get a sample x_i of X_{n-1} and a sample y_i of X_n. Then we we use these samples $x_1, \dots, x_M$ and $y_1, \dots, y_M$ to estimate $s(n, u)$. At this point, we recall that s minimizes (2) and that we decided to use estimators of the form (3). Consequently, the task is to find w and b that minimize the loss function

$$L(w, b) = \sum_{i=1}^{M} |x_i - \phi(wy_i + b)|^2. \tag{4}$$

If we dentoe the parameters that achieve this minimization problem by w_n and b_n (they depend on n because x and y are samples of X_{n-1} and X_n), then

(5) we can use $\phi(w_n u + b_n)$ as our approximation of $s(n, u)$, for each $n \in \{1, \dots, N\}$.

Once we have the above learning step achieved, we can now generate as many new samples from $\{0, 1\}$ as we want, with the guarantee that the probability of 0 will be close to 3/4 and the probability of 1 will be close to 1/4. Here is the algorithm:

1. The initial state is sampled from the invariant measure distribution, i.e. 0 with probability $\Phi_\infty(0)$ and 1 with probability $\Phi_\infty(1)$.

2. Once the inital state is known, we run the Markov chain for N steps. This chain is not time-homogenous and so at step $n \in \{0, \dots, N-1\}$, we use the transition matrix

$$\begin{bmatrix} 1 - \phi(b_n) & \phi(b_n) \\ 1 - \phi(w_n + b_n) & \phi(w_n + b_n) \end{bmatrix}.$$

(Recall that $\phi(w_{N-n}u + b_{N-n})$ is our estimate of $s(N - n, u)$, which is the probability that $X_{N-n-1} = 1$, given that $X_{N-n} = u$. Thus, this is the probability the backward Markov chain goes from being at u at time n to being at 1 at time $n + 1$. One minus that is that probability it goes from u to 0. Also, note that u is 0 or 1. Thus, $w_n u + b_n$ is either b_n or $w_n + b_n$.)

Now you have all the necessary background and can proceed to working on Problem 1.

Problem 1. Consider the setting described above.

(a) Compute the invariant measure of the matrix P in (1).

(b) What is the limit of P^n as $n \to \infty$? (Answer this from your knowledge of what P^n means, NOT by plugging in a large value of n and computing P^n.) Denote this limit by A. Calculate $P^{10} - A$ and obswerve that all the entries are tiny. This tells us that already at 10 steps, the Markov chain is super close to the invariant measure. Thus, we will take the terminal time to be $N = 10$.

(c) In your code, define a function L as in (4). The inputs are vectors x and y and numbers w and b. The output is the right-hand side in (4).

Hint. In matlab, the code looks like this

```
function r=phi(z)
     r=min(1,max(0,z));
end
function r=s(w,b,u)
     r=phi(w*u+b);
end
function r=L(x,y,w,b)
     r=sum((x-s(w,b,y)).^2);
end
```

(d) Sample $1,000$ numbers from the set $\{0, 1\}$, with 0 having probability $3/4$ and 1 having probability $1/4$. Denote these samples by $a_1, \ldots, a_{1000}$.

Hint. In matlab, `binornd(1,p,1,M)` gives M independent Bernoulli(p) random variables.

(e) Next, run a `for` loop over n going from 1 to 10. For each n, do the following:

(e.1) Run a `for` loop over $i \in \{1, \ldots, 1000\}$ and for each such i, use a_i as the initial state and sample of the Markov chain at times $n-1$ and n. Denote these two states by x_i and y_i. To do this, you do not need to run the Markov chain for n steps. Instead, you just need to use the $(n-1)$-step transition matrix P^{n-1} to sample x_i and then the matrix P to sample y_i.

For example, if $n = 3$, then

$$P^{n-1} = P^2 = \begin{bmatrix} 53/81 & 28/81 \\ 28/81 & 53/81 \end{bmatrix}.$$

This means that if $a_i = 0$, then we look at the first row and we pick $x_i = 0$ with probability $53/81$ and $x_i = 1$ with probability $28/81$. Once we know what x_i is, we can sample y_i. For example if $x_i = 1$, then we look at the second row of P and pick $y_i = 0$ with probability $2/9$ and $y_i = 1$ with probability $7/9$.

Hint. The probability $x_i = 1$ is the entry in P^{n-1} on row number $a_i + 1$ and column 2. Then once we have sampled $x_i \in \{0, 1\}$, we know that the probability $y_i = 1$ is the entry in P on row $x_i + 1$ and column 2. Also, in matlab, `binornd(1,p)` returns a Bernoulli(p) random variable.

(e.2) Let x dentoe the vector $(x_1, \cdots, x_M)$ and y the vector $(y_1, \cdots, y_M)$. Find the parameters (w, b) that minimize L.

Hint. In matlab, you can use

```
[thmin Lmin] = fminsearch(L,[0 0]);
w(n)=thmin(1); b(n)=thmin(2);
```

Then matlab starts the search for the minimum with the values $w = b = 0$ and returns the minimizers w(n) and b(n).

(e.3) To ensure that the algorithm is not gettings stuck at a local minimum, as well as to see how compllicated L can be, plot L as a function of $w \in [-10, 10]$ and $b \in [-10, 10]$ and point out the point $(w(n), b(n), L(w(n), b(n))$.

Hint. In matlab, you can use

```
range=[-10 10 -10 10];
figure(n)
clf
fsurf(@(w,b)L(x,y,w,b),range)
xlabel('w'); ylabel('b');
hold on
plot3(w(n),b(n),Lmin,'ro','MarkerFaceColor','r')
```

You will get 10 plots, one for each n.

Now that we learned the transition probabilities of the backward Markov chain, we will generate new samples and test their distribution.

(f) Run a `for` loop on i going from 1 to 1000. For each i, do the following.

(f.1) Generate Y from the invariant measure.

Hint. In matlab, `binornd(1,p)` gives a Bernoulli(p) random variable.

(f.2) Run a `for` loop on n going from 0 to 9. For each n, use the existing value of Y as the state of backward Markov chain at time n to generatre the value of the Markoc chain at the next time $n + 1$. This will be the new value of Y.

Hint. If Y is the value of the backward chain at time n, then $\phi(w_{10-n}Y + b_{10-n})$ is the probability the chain is 1 at time $n+1$. Also, in matlab, `binornd(1,p)` gives a Bernoulli(p) random variable.

(f.3) After the `for` loop on n ends, Y is the new sample the AI algorithm generates. Record these samples. (E.g. as `output(i)=Y;`)

(g) Moment of truth: Calculate the mean of your output samples. (E.g. `sum(out)/1000`) Did it work? IT CLOSE TO 1/4?!

Let B be a standard Brownian motion, starting at $B(0) = 0$. We aim to construct a time-reversed version of this Brownian motion. So fix the terminal time to be 2 and let $Y(t) = B(2 - t)$ be the time-reversed process. So at $t = 0$, $Y(0)$ is a normal with mean 0 and variance 2 and at the end, $Y(2) = 0$. One way to generate samples of Y is to generate samples of B and then plot $B(2-t)$ as a function of $t \in [0, 2]$. But we would like to generate Y by first sampling its starting point $Y(0)$ (as a normal with mean 0 and variance 2), and then solving an SDE. That is, we do not want to have to use the forward process to generate the backward one. We want to be able to generate the backward process directly.

The forward Brownian motion satsfies the simple SDE $dB(t) = dB(t)$. In other words, it has a diffusion coefficient of 1 and no drift. In class, we saw that the time-reverse process must then satisfy the equation

$$dY(t) = \frac{\partial \log p}{\partial x}(2 - t, Y(t))\, dt + d\overline{B}(t).$$

Here, $\overline{B}$ is another standard Brownian motion and $p(t, x)$ is the pdf of the forward process $B(t)$. In words, Y has the same diffusion coefficient of 1, but the zero drift of the original process (the Brownian motion) acquires an additional drift given by the space-derivative of $\log p$, evaluated at the backward running time $2 - t$.

Problem 2. Consider the above setting.

(a) In this particular example, we know exactly what $p(t, x)$ is. Write down its formula and compute $\frac{\partial \log p}{\partial x}$. Plug it $2-t$ for time and $Y(t)$ for space and write down the SDE that $Y(t)$ satisfies.

(b) Sample $Y(0)$ as a normal random variable with mean 0 and variance 2 and then numerically solve the SDE you found in (a) up to time 2. (You will need to stop right before you reach time 2 because you will not be able to plug in $t = 2$ into the SDE. You will see why when you answer (a)!)

(c) Plot $Y(t)$ as a function of $t \in [0, 2)$. Does that look to you like the time-reversed path of a standard Brownian motion?