Deep Learning Approximation for Stochastic Control Problems

Paper Review and Application

Alice Harting, SF2525

Stochastic control problem

Stochastic control problems seek to minimize a cost (or maximize a reward) obtained as some state variable X_s travels through space in time $[t, T]$. The path of X_s is dependent on some control α_s. The problem is framed in terms of a value function $u(x, t)$ defined by the minimal accumulated running cost $h(X_s, \alpha_s)$ over the time interval and terminal cost $g(X_T)$.

$$\begin{aligned} u(x,t) &= \inf_{\alpha(s):[t,T]\to\mathcal{A}} \int_t^T h(X_s,\alpha_s)ds + g(X_T) \\ X'(s) &= f(X_s,\alpha_s),\ s\in[t,T] \\ X(t) &= x \end{aligned} \tag{1}$$

Traditionally, this problem has been solved by Pontryagin principle and Hamilton-Jacobi equation.

Paper approach

In the paper *Deep Learning Approximation for Stochastic Control Problems*, authors Jiequn Han and Weinan E propose a deep learning approach to solve high-dimensional stochastic control problems. This mitigates the "curse of dimensionality", i.e. the problem that data quickly becomes sparse in high dimensions.

Rather than dealing with the value function, the authors solve for the optimal controls directly by setting up a neural network with the total cost as the loss function. The network consists of a set of subnetworks - each of which corresponds to a time step - in which the optimal controls are approximated. The subnetworks are stacked to reflect the model dynamics and the total cost is calculated at the end.

Mathematical formulation

Given a probability space $(\Omega, \mathcal{F}, \mathbb{P})$ and a finite time horizon T, the state $s_t \in S_t$ (the set of potential states) evolves as

$$\begin{aligned} s_{t+1} &= s_t + b_t(s_t, a_t) + \zeta_{t+1} \\ s_0 &= s \end{aligned} \tag{2}$$

for $t = 0, 1, ..., T$ where $b_t(s_t, a_t)$ is the deterministic drift, a_t is the control and ζ_{t+1} are independent $\mathcal{F}_{t+1}$-measurable random variables containing all the noise.

We may impose constraints on the control:

$$\begin{aligned} g_i(s_t, a_t) &= 0,\ i = 1, ..., I \\ h_j(s_t, a_t) &\geq 0,\ j = 1, ..., J. \end{aligned} \tag{3}$$

Thus, the set of eligible controls is

$$\mathcal{A}_t = \{a_t(s_t) : \mathbb{R}^m \to \mathbb{R}^n \mid \text{constraints in eq. 3 satisfied } \forall s_t \in S_t\}. \tag{4}$$

At each time step, we accumulate the cost $c_t(s_t, a_t)$. Without constraints, our north star is

$$\min_{\alpha_t\in\mathcal{A}_t \text{ for } t=0,...,T-1} \mathbb{E}\Big(\sum_{t=0}^{T-1} c_t(s_t, a_t)\Big) + c_T(s_T). \tag{5}$$

To achieve eq. 5, we fit the parameters $\boldsymbol{\theta}_t$ of the neural network such that $a_t(s_t) \approx a_t(s_t|\boldsymbol{\theta}_t)$.

With constraints, we add to eq. 5

$$\sum_{i=1}^{I} \lambda_i P_{eq}(g_i(s_t, a_t)) + \sum_{j=1}^{J} \sigma_j P_{ineq}(h_j(s_t, a_t)) \tag{6}$$

which penalizes violations of eq. 3.

In practice [a], we use

$$\begin{aligned} P_{eq}(x) &= x^2 \\ P_{ineq}(x) &= (\min\{0, x\})^2. \end{aligned} \tag{7}$$

Figure 1 shows the network architecture with N hidden layers (each denoted h_t^i at t) in each subnetwork. For simplicity, we denote the cumulative cost at time $t \leq T$ with C_t.

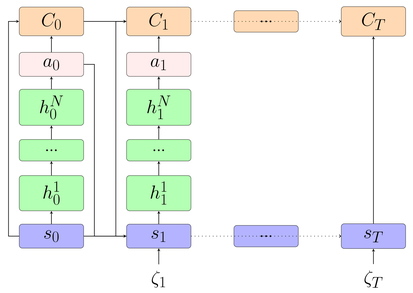


Figure 1. Network architecture

Application

SL's commuting problem

The application is inspired by an example in the paper. We consider the flow of commuters in a constrained system of public transport between 6AM and 11AM as illustrated in figure 2.

At every time step, r_t new commuters arrive at the bus station. SL now has the option to send them with a bus (a_t^{rb}), send them to wait (a_t^{rw}) or send them off with a taxi a_t^{rt}. Buses can take no more than γ travellers per time increment. To keep a person waiting costs ϕ per time increment. The taxi cost per person and the number of new commuters vary with time, k_t and r_t. The expected curves are seen in figure 3a and 3b to which we add random (normal) noise.

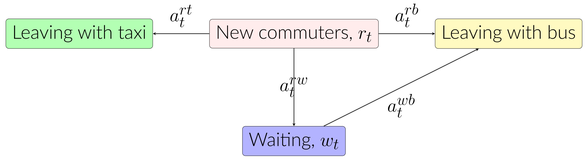


Figure 2. SL's commuting problem

To summarize, we have the state evolution

$$w_{t+1} = w_t + a_t^{rw} - a_t^{wb} \tag{8}$$

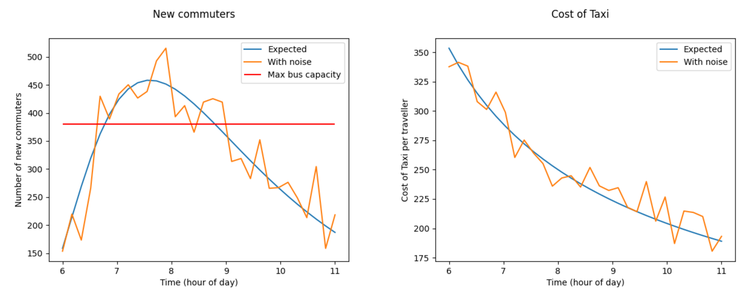
with constraints

$$\begin{aligned} r_t &= a_t^{rt} + a_t^{rw} + a_t^{rb} \\ a_t^{rb} + a_t^{wb} &\leq \gamma \\ a_t^{wb} &\leq w_t \end{aligned} \tag{9}$$

and cost

$$c_t(w_t, a_t) = \phi w_t + k_t a_t^{rt}. \tag{10}$$

[a] The paper suggests $\min\{0, x^2\}$ but I suspect this is a misprint



(a) New commuters, r_t (b) Cost of taxi per traveller, k_t

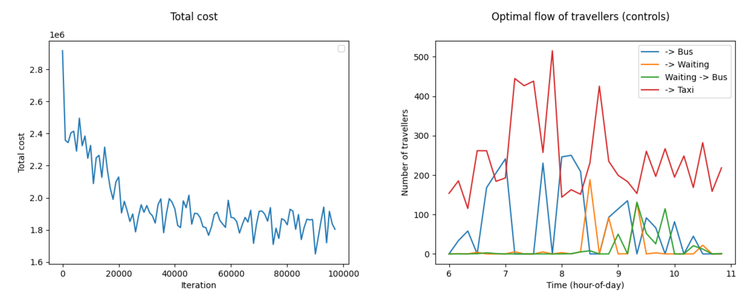
Figure 3. Input data

Implementation

The model is trained with $N = 10^5$ randomly generated samples of $\mathbf{r}$ and $\mathbf{k}$. We use 10 hidden layers per subnetwork, each with 10 nodes. Parameters are fitted with the Adam optimizer, which is a variation of SGD. The total commuting population is 10^4 and travels between 6AM and 11AM in the morning. Time is discretized into 10-minute increment. We set $w_0 = 0, \gamma = 380, \phi = 10$ and penalty coefficients $\sigma_1, \sigma_2 = 10^9$. We replace the equality constraint by an inequality constraint by solving for one control while requiring positivity instead of fitting it. All activation functions are sigmoid except the last one in each subnetwork which is a ReLU to ensure positive flows.

Results

As shown in figure 4a, the cost (i.e. the loss) converges. Increasing the penalty coefficients doesn't change the cost limit, hence the constraints are respected. This is also seen in figure 4b showing the optimal control for test data. It's interesting to see that w_t (waiting) is populated only towards the end. However, it's strange to see that the capacity γ isn't met before a_t^{rt} (taxi) is populated.



(a) Total cost (b) Optimal controls

Figure 4. Neural network output

References

[1] Jiequn Han and Weinan E.
Deep learning approximation for stochastic control problems.
CoRR, abs/1611.07422, 2016.

aharting@kth.se github/aharting