Homework 6 Solution

Problem 1:

1.1 (a)

A deterministic policy $\pi : S \to A$ is determined uniquely by its value for all the states $s \in S$; since $\pi(s_i) \in A = \{a_1, a_2, a_3, a_4\}$ for $i = 1, 2, 3, 4$, i.e. each of the 4 states have 4 possible actions, in total $4^4 = 256$ deterministic policies are possible.

1.1 (b)

V*(1) =4.11, V*(2) = 3.88, V*(3) = 3.61, V*(4) = 3.61

π*(1) = 2, π*(2) = 3, π*(3) = 2, π*(4) = 2

Yes, there appears to be a unique optimal deterministic policy. After running value iterations till the value function converged to within a threshold of $1e-5$ (in terms of max-norm), I obtained the optimal policy by the greedy choice $\pi^*(s) \in \arg\max_a R(s,a) + \gamma V^*(T(s, \pi(s)))$, and in doing so saw that all the greedy choices are unique, i.e. for each s there's a unique action that maximizes $R(s,a) + \gamma V^*(T(s, \pi(s)))$.

1.1 (c)

We show that an optimal stochastic policy must necessarily be deterministic in this case.

Given a stochastic policy π, let π_s^a denote the probability that action a should be taken in state s. Then the value function for state s is defined as the expected discounted reward starting in s and following π:

$$V^\pi(s) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r(t)] \tag{1}$$

where $r(t)$ is a (random) award at time t, and the expectation is taken with respect to the probability distribution over state sequences determined by s and π. We can rewrite the above by conditioning on the first (random) action a_0, by law of iterated expectation:

$$V^\pi(s) = \mathbb{E}[\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r(t)|a_0]] \tag{2}$$

$$= \sum_{a \in A} \pi_s^a \{\mathbb{E}[r(0)|a_0 = a] + \mathbb{E}[\sum_{t=1}^{\infty} \gamma^t r(t)|a_0 = a]\} \tag{3}$$

$$= \sum_{a \in A} \pi_s^a \{R(s,a) + \gamma \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r(t)|s_0 = T(s,a)]\} \tag{4}$$

$$= \sum_{a \in A} \pi_s^a \{R(s,a) + \gamma V^\pi(T(s,a))\} \tag{5}$$

where we used the linearity of expectation, the fact that $\mathbb{E}[r(0)|a_0 = a]$ is simply a constant (reward for starting in state s and taking action a), and the definition of V^π.

For any state s, an optimal stochastic policy π^* must be greedy in the sense that

$$V^{\pi^*}(s) = \max_{\pi_s} \sum_{a \in A} \pi_s^a \{R(s,a) + \gamma V^{\pi^*}(T(s,a))\}$$

where π_s is a probability vector of size $|A|$; the optimality of π^* follows from the same argument as in the deterministic case. Note the above is a linear programming problem in π_s over a simplex constraint ($\forall a \in A, \pi_s^a > 0, \sum_a \pi_s^a = 1$), therefore the maximum must occur at an extreme point π_s such that its component corresponding to $\max_{\pi_s} \sum_{a \in A} \pi_s^a \{R(s,a) + \gamma V^{\pi^*}(T(s,a))$ is one, while other components are 0, i.e. $\exists a, s.t. \pi_s^a = 1, \pi_s^{b \neq a} = 0$, in other words π_s is deterministic and places all probability mass on the best action in state s. This holds for every $s \in S$, and finding the optimal stochastic policy reduces to finding the optimal deterministic policy, so the above greedy criteria reduces to Bellman equations

$$V^{\pi^*}(s) = \max_{a \in A} \{R(s,a) + \gamma V^{\pi^*}(T(s,a))\}, \forall s \in S$$

In general, the optimal policy may not be unique, as there can be multiple greedy choices giving rise to the same optimal value function; in this problem there appears to be a unique optimal policy (see answer in 1.2 (b)).

1.1 (d)

When change λ from 0.8 to 0.01,

We get a new optimal value function V^* =[1.01,1.0051,0.51,0.51]

but the the optimal policy's result will not change. But the training progress will converge much quicker.

(ONLY for this case, in general, it is not necessary.)

1.2

Well, given an MDP, the Bellman equations always define the optimal value function V, such that it assigns value to each state that is no worse than any policy, i.e. $\forall \pi, \forall s \in S, V^*(s) \geq V^\pi(s)$. Then run one step of policy iteration to find the policy $\pi_3 = \pi^*$ that is greedy with respect to V^*, and we certainly have $\forall s \in S, V^{\pi_3}(s) \geq V^{\pi_2}(s)$ and $\forall s \in S, V^{\pi_3}(s) \geq V^{\pi_1}(s)$, for any π_2 and π_1.

1.3

To introduce second-order dependency into the MDP framework, we can simply redefine the set of environment states S to be all tuples of "atomic" states, i.e. a "superstate" of the two consecutive states, for a total of $|S|^2$ superstates. This allows the MDP to still satisfy the Markov property: given the most recent superstate (two states), the future superstate is independent of the past (the future superstate will share the present state with the past superstate). So the transition function can be redefined to be $T : (S \times S) \times A \to (S \times S)$, for example $T((s, s'), a) = (s', s'')$. Then the transition function, reward function, and policy can be similarly redefined to operate on"superstates", and only minor modification to the MDP framework is needed. For example, the Bellman equations become

$$V^{\pi^*}(s, s') = \max_{a \in A} \{R((s, s'), a) + \gamma V^{\pi^*}(T((s, s')a))\}, \forall (s, s') \in S \times S$$

while all the algorithms remain the same.

Problem 2:

2.1

Poisson data log-likelihood:

$$L(\lambda) = \log p(x^{(1)\ldots(m)}|\lambda) = \sum_{i=1}^{m} \log p_\lambda(x^{(i)}) \tag{6}$$

$$= \log \lambda \sum_{i=1}^{m} x^{(i)} - m\lambda - \sum_{i=1}^{m} \log x^{(i)}! \tag{7}$$

Setting derivative to zero:

$$\frac{\partial L}{\partial \lambda} = \frac{\sum_{i=1}^{m} x^{(i)}}{\lambda} - m = 0$$

so

$$\lambda_{MLE} = \frac{\sum_{i=1}^{m} x^{(i)}}{m}$$

is simply the sample mean.

The second derivative $-\frac{\sum_{i=1}^{m} x^{(i)}}{\lambda} < 0$ shows that L is concave, so λ_{MLE} indeed maximizes L.

2.2

Lemma 2.1. *For a Poisson random variable X with parameter λ, its moment generating function is*

$$M_X(t) = \mathbb{E}[\exp(tX)] = \exp[\lambda(\exp(t) - 1)]$$

The above is a standard statistical result. See `https://www.statlect.com/probability-distributions/Poisson-distribution` for an example proof.

Lemma 2.2. *Let $X_1, ..., X_m$ be i.i.d samples; define sample mean $\bar{X} = \sum_i X_i/m$. Then*

$$M_{\bar{X}}(t) = \mathbb{E}[\exp(t\bar{X})] = [M_X(\frac{t}{m})]^m$$

Proof.

$$\begin{aligned} M_{\bar{X}}(t) &= \mathbb{E}[\exp(t\bar{X})] && (8)\\ &= \mathbb{E}[\exp(\frac{t}{m}\sum_i X_i)] && (9)\\ &= \prod_i \mathbb{E}[\exp(\frac{t}{m}X_i)] && (10)\\ &= [M_X(\frac{t}{m})]^m && (11) \end{aligned}$$

where we used the fact that $X_1, ..., X_m$ are i.i.d. □

Combining the above lemmas, we have the MGF of the sample mean of m i.i.d. Poisson samples $X_i \sim Poisson(\lambda), i = 1, ..., m$ is

$$M_{\bar{X}}(t) = \mathbb{E}[\exp(t\bar{X})] = \exp[\lambda(\exp(\frac{t}{m}) - 1)]^m = \exp[m\lambda(\exp(\frac{t}{m}) - 1)] \quad (12)$$

Substituting $\bar{X}$ into the given inequality yields, for any $t > 0$,

$$\mathbb{P}(\bar{X} \geq a) \leq \frac{\mathbb{E}[\exp(t\bar{X})]}{\exp(ta)} = \exp[m\lambda(\exp(\frac{t}{m}) - 1) - ta] \quad (13)$$

Since we want the upper bound as tight as possible, we minimize the right-hand-side w.r.t to t; setting derivative to 0 gives:

$$\begin{aligned} &\frac{\partial}{\partial t}\exp[m\lambda(\exp(\frac{t}{m}) - 1) - ta] && (14)\\ &= \frac{\partial}{\partial t}\exp[m\lambda(\exp(\frac{t}{m}) - 1) - ta](\lambda(\exp(\frac{t}{m}) - a) = 0 && (15) \end{aligned}$$

Thus $t^* = m\log\frac{a}{\lambda} > 0$; this achieves the global minimum of the upper bound (13), which is convex in t (its second derivative is the second moment of $\bar{X}$ which is always non-negative). Plugging t^* into the bound gives:

$$\mathbb{P}(\bar{X} \geq a) \leq \frac{\mathbb{E}[\exp(t\bar{X})]}{\exp(ta)} = \exp[m(a - \lambda - a\log(\frac{a}{\lambda}))] \quad (16)$$

In our case, $\lambda_{MLE} = \bar{X}$, and we'd like $\mathbb{P}(\bar{X} \leq \lambda + \epsilon) \geq 1 - \exp(-5)$, which is equivalent to $\mathbb{P}(\bar{X} \geq \lambda + \epsilon) \leq \exp(-5)$. This requires setting $a = \lambda + \epsilon$ in the above, and

$$\mathbb{P}(\bar{X} \geq \lambda + \epsilon) \leq \exp[m(\epsilon - (\lambda + \epsilon)\log(1 + \frac{\epsilon}{\lambda}))] \leq \exp(-5) \quad (17)$$

Solving for m gives the desired sample complexity bound

$$m \geq \frac{5}{(\lambda + \epsilon)\log(1 + \frac{\epsilon}{\lambda}) - \epsilon} \quad (18)$$

2.3

Given a prior distribution $p(\lambda)$ and data likelihood $p(D|\lambda)$, where $D = \{x^{(1)\cdots(m)}\}$ is an i.i.d sample from $p(x|\lambda)$, the MAP estimate of λ is

$$\begin{aligned}
\lambda_{MAP} &= \arg\max_{\lambda} p(\lambda|D) && (19)\\
&= \arg\max_{\lambda} \log p(\lambda|D) && (20)\\
&= \arg\max_{\lambda} \log p(\lambda, D) && (21)\\
&= \arg\max_{\lambda} \log p(\lambda) + \log p(D|\lambda) && (22)
\end{aligned}$$

where we used the fact that log is monotonic and Bayes rule. We're given

$$p(\lambda) = \frac{1}{5}\max\{1 - \lambda/10, 0\}$$

, and therefore

$$\log p(\lambda) = \begin{cases} \log\{\frac{1}{5}(1-\lambda/10)\} & 0 < \lambda < 10 \\ -\infty & \text{otherwise} \end{cases}$$

From question 1 we have

$$\begin{aligned}
\log p(D|\lambda) &= \sum_{i=1}^{m} \log p_\lambda(x^{(i)}) && (23)\\
&= \log\lambda \sum_{i=1}^{m} x^{(i)} - m\lambda - \sum_{i=1}^{m} \log x^{(i)}! && (24)\\
&= \log\lambda s - m\lambda - \sum_{i=1}^{m} \log x^{(i)}! && (25)
\end{aligned}$$

where we adopt the shorthand

$$s \triangleq \sum_{i=1}^{m} x^{(i)}$$

Substituting in (22) gives

$$\begin{aligned}
\lambda_{MAP} &= \arg\max_{0<\lambda<10} \log p(\lambda) + \log p(D|\lambda) && (26)\\
&= \arg\max_{0<\lambda<10} f(\lambda) && (27)
\end{aligned}$$

where we only keep terms involving λ:

$$f(\lambda) = \log(1 - \lambda/10) + \log\lambda s - m\lambda$$

Setting derivative to zero

$$f'(\lambda) = \frac{1}{\lambda - 10} + \frac{s}{\lambda} - m = 0$$

and solving for λ gives

$$\lambda^* = \frac{1 + s + 10m - \sqrt{(1+s+10m)^2 - 40ms}}{2m}$$

(the other root is always greater than 10 and discarded). It's clear that $f'(\lambda)$ is decreasing over $0 < \lambda < 10$, so $\lambda^* = \lambda_{MAP}$ is indeed the MAP estimate.

Problem 3:

The network structure for perceptrons can be formed as three layers:

The **1 layer(input layer)** consist of 10 neurons consisting of input boolean value.

The **2 layer(hidden layer)** consist of 3 neurons. Each accept all inputs as first layer's outputs. Setting w = [1, 1, 1, 1, 1, 1, 1, 1, 1, 1], b = 0.

Neuron (1): when the sum of four inputs equals 0, output 1, otherwise 0.

Neuron (2): when the sum of four inputs equals 4, output 1, otherwise 0.

Neuron (3): when the sum of four inputs equals 8, output 1, otherwise 0.

The **3 layer(output layer)** consist of 1 neuron accepting all input boolean value from second layer. Setting w = [1, 1, 1], b = 0, when the sum of three inputs equals 1, output 1 representing divided by 4, otherwise 0 representing not divided by 4.

For relu neural network,
Define loss maximized loss function:

$$\mathcal{L}(\hat{y}, y) = -\Big[(1-y)\log(1-\hat{y}) + y\log\hat{y}\Big]$$

W update rule,

$$W^{[\ell]} = W^{[\ell]} - \alpha\frac{\partial\mathcal{L}}{\partial W^{[\ell]}}$$

There are two layers need to update, differentiate seperately for both layer3 and layer2,

$$\frac{\partial\mathcal{L}}{\partial W^{[3]}} = (a^{[3]} - y)a^{[2]}$$

$$\frac{\partial\mathcal{L}}{\partial W^{[2]}} = \underbrace{\frac{\partial\mathcal{L}}{\partial a^{[3]}}}_{(a^{[3]}-y)}\underbrace{\frac{\partial a^{[3]}}{\partial z^{[3]}}\frac{\partial z^{[3]}}{\partial a^{[2]}}}_{W^{[3]}}\underbrace{\frac{\partial a^{[2]}}{\partial z^{[2]}}}_{g'(z^{[2]})}\underbrace{\frac{\partial z^{[2]}}{\partial W^{[2]}}}_{a^{[1]}} = (a^{[3]} - y)W^{[3]}g'(z^{[2]})a^{[1]}$$

When I increase the size of training set, the learned weights converge more accurately to my perceptron solution as vary the size of the raning set from 100 to 10000.