Advanced Topics in Machine Learning - Assignment 7

Christoffer Thrysøe - dfv107

October 30, 2017

1 Label-efficient prediction and label-efficient bandits

1.1

We have that Z_t is a Bernoulli random variable with bias ε, therefore the probability is given by:

$$P\{Z_t = 1\} = \varepsilon$$
$$P\{Z_t = 0\} = 1 - \varepsilon$$

The expected value Z_t is given as followed:

$$\mathbb{E}[Z_t] = P\{Z_t = 1\} \cdot 1 + P\{Z_t = 0\} \cdot 0 = \varepsilon \cdot 1 + (1-\varepsilon) \cdot 0 = \varepsilon \tag{1}$$

Thus the expected value of Z_t is it's bias ε. For the algorithm, we draw Z_t at each iteration. We wish to express how many times we can expect to have $Z_t = 1$ over the T iterations. To quantify this, we sum the expected value, which we just derived:

$$\mathbb{E}\left[\sum_{t=1}^{T} Z_t\right] = \sum_{t=1}^{T} \mathbb{E}[Z_t] = \sum_{t=1}^{T} \varepsilon = \varepsilon T \tag{2}$$

where the second term follows from the linearity of expectation, which allows us to move the expected value into the sum. Thus concluding that the expected number of times the algorithm requests to observe the labels is εT.

1.2

For the algorithm, we are in adversarial full information setting. That is, when we request to observe the columns, we are allowed to observe $l_t^1 \dots, l_t^K$. We wish to show that the expected regret of the algorithm satisfies the following bound on the expected regret:

$$\mathbb{E}[R_T] \leq \sqrt{2\frac{1}{\varepsilon} T \ln K} \tag{3}$$

If we have that $\varepsilon = 1$, we recover the exact same algorithm as hedge, I will therefore use a similar approach to bounding the expected regret. The initial analysis is similar to the hedge, therefore I will start by lemma

5.2 from Yevgeny's lecture notes and work from there. The lemma states:

$$\sum_{t=1}^{T}\sum_{a=1}^{K} p_t(a) l_t^a - \min_a L_T(a) \leq \frac{\ln K}{\eta} + \frac{\eta}{2}\sum_{t=1}^{T}\sum_{a=1}^{K} p_t(a)(l_t^a)^2 \tag{4}$$

First we note that the left hand side of (4) is the expected regret. We only observe and record loss if $Z_t = 1$, which we from the previous assignment know is expected to happen εT times, thus we only need to sum εT times instead of T times. We also note that since our loss is scaled with ε, that is from the algorithm. when we update our loss, we take l_t^a/ε and since $l_t^a \in [0,1]$, we know that for our algorithm, the added loss will be $l_t^a \in [0, \frac{1}{\varepsilon}]$ Thus we can write the following:

$$\begin{aligned}
\frac{\ln K}{\eta} + \frac{\eta}{2}\sum_{t=1}^{\varepsilon T}\sum_{a=1}^{K} p_t(a)(l_t^a)^2 &\leq \frac{\ln K}{\eta} + \frac{\eta}{2}\sum_{t=1}^{\varepsilon T}\sum_{a=1}^{K}\left(\frac{1}{\varepsilon}\right)^2 p_t(a) && (5)\\
&\leq \frac{\ln K}{\eta} + \frac{\eta}{2}\sum_{t=1}^{\varepsilon T}\frac{1}{\varepsilon^2} && (6)\\
&= \frac{\ln K}{\eta} + \frac{\eta}{2}\frac{T}{\varepsilon} && (7)\\
&= \frac{\ln K}{\eta} + \frac{\eta T}{2\varepsilon} && (8)
\end{aligned}$$

Taking the derivative with respect to η we get:

$$\frac{\partial}{\partial \eta}\left[\frac{\ln K}{\eta} + \frac{\eta T}{2\varepsilon}\right] = -\frac{\ln k}{\eta^2} + \frac{T}{2\varepsilon} \tag{9}$$

Setting the derivative to zero:

$$\begin{aligned}
0 &= -\frac{\ln k}{\eta^2} + \frac{T}{2\varepsilon} \Rightarrow && (10)\\
\eta &= \sqrt{\frac{2\varepsilon \ln K}{T}} && (11)
\end{aligned}$$

The second derivative of (8) is $\dfrac{2\ln k}{\eta^3}$ and since η is positive, we have that the optimum in (11) is an extremal point. Setting this optimal value for η in (8) we get the following:

$$\begin{aligned}
\frac{\ln K}{\eta} + \frac{\eta}{2\varepsilon}T &= \frac{\ln K}{\sqrt{\frac{2\varepsilon \ln K}{T}}} + \frac{T\sqrt{\frac{2\varepsilon \ln K}{T}}}{2\varepsilon} && (12)\\
&= \sqrt{2\ln K\frac{T}{\varepsilon}} = \sqrt{2\frac{1}{\varepsilon}T\ln K} && (13)
\end{aligned}$$

From (4) we now have that

$$\mathbb{E}[R_T] \leq \sqrt{2\frac{1}{\varepsilon}T\ln K} \tag{14}$$

which is what we wanted to prove.

1.3

We wish to show that with high probability the exact number of observed columns is not significantly larger than εT. That is we wish to provide a bound on the probability that the sum of our independent random variable Z_t deviates from its expected value εT. For this we can use Hoeffding's inequality, which is listed in Corollary 2.4 in Yevgeny's lecture notes:

$$P\left\{\frac{1}{T}\sum_{i=1}^{T} Z_t - \mathbb{E}[Z_t] \geq \epsilon\right\} \leq e^{-2T\epsilon^2} \tag{15}$$

If we take $\epsilon = \sqrt{\frac{\ln \frac{1}{\delta}}{2T}}$ we get:

$$P\left\{\frac{1}{T}\sum_{i=1}^{T} Z_t - \mathbb{E}[Z_t] \geq \sqrt{\frac{\ln \frac{1}{\delta}}{2T}}\right\} \leq \delta \tag{16}$$

To get a high probability bound, we re-write (16):

$$P\left\{\frac{1}{T}\sum_{i=1}^{T} Z_t - \mathbb{E}[Z_t] \leq \sqrt{\frac{\ln \frac{1}{\delta}}{2T}}\right\} \geq 1-\delta \tag{17}$$

$$P\left\{\frac{1}{T}\sum_{i=1}^{T} Z_t \leq \frac{1}{T}\sum_{i=1}^{T}\mathbb{E}[Z_t] + \sqrt{\frac{\ln \frac{1}{\delta}}{2T}}\right\} \geq 1-\delta \tag{18}$$

Thus, with high probability $1-\delta$, we have that exact number of observed columns is not significantly higher than εT. If it was significantly higher, then the first inequality of (18), would not hold.

1.4

1.5

2 Policy evaluation

We wish to prove that the iterative policy evaluation converges to the true value function, that is: $\forall s \in S$: $\lim_{k\to\infty} V_k(s) = V^\pi(s)$, where the update rule is given by:

$$\forall s \in S : V_{k+1}(s) \leftarrow \sum_a \pi(s,a) \sum_{s'} P^a_{ss'}[R^a_{ss'} + \gamma V_k(s')] \tag{19}$$

and $0 < \gamma < 1$. First we note that for the converged value function we have:

$$V^\pi(s) = \sum_a \pi(s,a) \sum_{s'} P^a_{ss'}[R^a_{ss'} + \gamma V^\pi(s')] \tag{20}$$

We note, by definition that:

$$\begin{aligned} V_{k+1} &= \sum_a \pi(s,a) \sum_{s'} P^a_{ss'}[R^a_{ss'} + \gamma V_k(s')] && (21)\\ &\leq \sum_a \pi(s,a) \sum_{s'} P^a_{ss'}[R^a_{ss'} + \gamma V^\pi(s')] && (22)\\ &= V^\pi(s) && (23)\end{aligned}$$

and we define:

$$\Delta_k = \max_s |V_k(s) - V^\pi(s)| \tag{24}$$

rewriting (24), we define:

$$V_k(s) \geq V^\pi(s) - \Delta_k \tag{25}$$

Now we can write:

$$\begin{aligned}
V_{k+1} &= \sum_a \pi(s,a) \sum_{s'} P^a_{ss'}[R^a_{ss'} + \gamma V_k(s')] && (26)\\
&\geq \sum \pi(s,a) \sum_{s'} P^a_{ss'}[R^a_{ss'} + \gamma(V^\pi(s') - \Delta_k)] && (27)\\
&= \sum \pi(s,a) \sum_{s'} P^a_{ss'}[R^a_{ss'} + \gamma(V^\pi(s')] - \sum_a \pi(s,a) \sum_{s'} P^a_{ss'} \gamma \Delta_k && (28)
\end{aligned}$$

where the above follows from (25) and taking Δ_k outside. We note that $\sum_{s'} P^a_{ss'} = 1$ and therefore we can get the following:

$$\begin{aligned}
\sum_a \pi(s,a) \sum_{s'} P^a_{ss'}[R^a_{ss'} + \gamma(V^\pi(s')] - \sum_a \pi(s,a) \sum_{s'} P^a_{ss'} \gamma \Delta_k &= V^\pi(s) - \sum_a \pi(s,a) \gamma \Delta_k && (29)\\
&\geq V^\pi(s) - \gamma \Delta_k && (30)
\end{aligned}$$

where the last line holds because $\pi(s,a) \leq 1$. Thus we have that:

$$\sum \pi(s,a) \sum_{s'} P^a_{ss'}[R^a_{ss'} + \gamma(V^\pi(s') - \Delta_k)] \geq V^\pi(s) - \gamma \Delta_k \Leftrightarrow V^\pi(s) - \Delta_k \geq V^\pi(s) - \gamma \Delta_k \tag{31}$$

which implies that:

$$\Delta_{k+1} \leq \gamma \Delta_k \tag{32}$$

Because of the above, we can now state the following:

$$|V_{k+1}(s) - V^\pi(s)| \leq \gamma |V_k(s) - V^\pi(s)| \leq \cdots \leq \gamma^{k+1} |V_0(s) - V^\pi(s)| \tag{33}$$

with $0 < \gamma < 1$. Because the difference converges to zero, $V_k(s)$ converges towards $V^\pi(s)$, thus concluding that:

$$\lim_{k \to \infty} V_k(s) = V^\pi(s) \tag{34}$$

which is what we wanted to prove.

$$\begin{aligned}
|V_{k+1}(s) - V^\pi(s)| &= \left| \sum_a \pi(s,a) \sum_{s'} P^a_{ss'}[R^a_{ss'} + \gamma V_k(s')] - \sum_a \pi(s,a) \sum_{s'} P^a_{ss'}[R^a_{ss'} + \gamma V^\pi(s')] \right| && (35)\\
&= \sum_a \pi(s,a) \sum_{s}^{\prime} (|P^a_{ss'} R^a_{ss'} + \gamma P^a_{ss'} V_k(s') - P_{ss'} R^a_{ss'} - \gamma P^a_{ss'} V^\pi(s'))| && (36)\\
&= \sum_a \pi(s,a) \sum_{s'} \gamma P^a_{ss'} (|V_k(s') - V^\pi(s')|) && (37)\\
&\leq \sum_a \pi(s,a) \sum_{s'} \gamma (|V_k(s') - V^\pi(s')|) && (38)\\
&= \sum_a \pi(s,a) \sum_{s'} \gamma (\sum_a \pi(s,a) \sum_{s'} \gamma P^a_{ss'} (V_{k-1}(s') - V^\pi(s')) && (39)
\end{aligned}$$

3 Reinforcement learning

For this assignment, we have been given a floor plan, which represents an apartment with $3 \times 4 = 12$ rooms. Each room has at least one door, which the agent can use to enter the adjacent room (if connected by a door), and some rooms may contain a penalty when entered. Each move done by the agent will inflict a penalty -1. The agent has four possible movements at each state: {up,down,right,left}. If the agent chooses an action, where there is no door to connect to the next room, the agent will remain in the same room, viewed as the agent bumping into the door-less wall. When bumping into a wall, the movement penalty of -1 is still enforced. In the apartment, there is a single room, which is a terminal state. When entered, the episode ends and the agent does not move further. The problem was modelled using an `Apartment` object, which keeps track of each room in the apartment and is used to fetch a given room and set the initial rewards and so on. An object for `Room` is also used, which is a model for each room i.e. state in the apartment. Each room contains information such as room number, the reward of the room (which in our case is only negative), the action policy of the room $\pi(s,a)$, which is the probability of performing action a when being in state s, and a mapping of each action to a new state. For example for room 0 (the upper left room) the "`up`" action maps to going into a wall (which i denote -1 to identify), as there is no door going up. The "`right`" action maps to room 1 (which is to it's right, connected by a door).

3.1

We wish to implement a Markov decision process (MPD) to compute the value function V^{rand}. *rand* means that the policy is uniform. Therefore for each possible action, in each state, the policy, i.e. the probability of performing an action given a state, will be uniform: $\pi(s_i, a_j) = 0.25$ because at each state the agent has four possible actions. For this task I have chosen to implement the iterative policy evaluation algorithm, which is shown in algorithm 1.

```
Algorithm 1: Dynamic programming
Input: policy $\pi$ to be evaluated, arbitrary $V$, threshold $\theta > 0$
1 repeat
2     $\Delta \leftarrow 0$
3     foreach $s \in S$ do
4         $v \leftarrow V(s)$
5         $V(s) \leftarrow \sum_a \pi(s,a) \sum_{s'} P^a_{ss'} [R^a_{ss'} + \gamma V(s')]$
6         $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
7 until $\Delta < \theta$
Output: $V \approx V^\pi$
```

$P^a_{ss'}$ is the probability of going to state s' when being in state s and performing action a. For this given problem, the probability will be 1 or 0, because the transitions are deterministic e.g. if we are in room 0 and choose the action "`right`", we will with probability 1 go to room 1, thus for the calculation on line 5 in algorithm 1, we only need to sum up rewards for each possible actions. The reward $R^a_{ss'}$ is also deterministic, where we enforce a penalty of -1 for each movement and entering certain rooms give a greater penalty. However if the agent is in one of the extra penalized rooms and bump into a wall, this penalty is not enforced again, only the penalty of -1 for making a move. The terminal state was implemented such that no penalty was made when moving inside, i.e. bumping into a wall when being in the terminal state, and the agent could not leave the state. The algorithm was run with $\gamma = 1.0$, which is a setting we can chose since we have a zero reward absorbing state (terminal state), which is reachable from all states. The algorithm also takes an initial value function V with all it's entries set to zero. The stopping criteria was set to $\theta = 1 \times 10^{-13}$. The returned value function V^{rand}, after running the algorithm was:

$$V^{rand} = \begin{bmatrix} -224 & -218 & -218 \\ -226 & -203 & -214 \\ -214 & -184 & -201 \\ -228 & -94 & 0 \end{bmatrix} \tag{40}$$

where the entries in the matrix corresponds to the given room, i.e the value for room 0 is -224. As evident from (39), the worst starting place is room 9, which is because we must go through room 6, which has a penalty of -10. The lowest penalty (not including the terminal state) is room 10, which is adjacent to the terminal state.

3.2

For this assignment we wish to compute the optimal value function V^*, that is at each state we want to find the optimal action, which is the action that gets us with least penalty to the terminal state. To solve this, I have implemented the policy iteration algorithm, shown in algorithm 2.

Algorithm 2: Policy iteration

```
Input: policy $\pi$ and value function $V$, threshold $\theta > 0$
repeat
    repeat // policy evaluation
        $\Delta \leftarrow 0$
        foreach $s \in S$ do
            $v \leftarrow V(s)$
            $V(s) \leftarrow \sum_a \pi(s,a) \sum_{s'} P^a_{ss'} [R^a_{ss'} + \gamma V(s')]$
            $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
    until $\Delta < \theta$
    $f_{\text{stable}} \leftarrow$ True
    foreach $s \in S$ do // policy improvement
        $a \leftarrow \pi(s)$
        $\pi(s) \leftarrow \text{argmax}_a \sum_{s'} P^a_{ss'} [R^a_{ss'} + \gamma V(s')]$
        if $a \neq \pi(s)$ then $f_{\text{stable}} \leftarrow$ False
until $f_{\text{stable}} =$ True
Output: $\pi \approx \pi^*, V \approx V^\pi$
```

The algorithm consists of two steps, the first step is evaluating the value function V^π based on the given policy π, in which we can use the algorithm presented in question 3.1. The next step is checking to see if we can improve this policy. If it can be improved, we start over and compute the value function again with the new improved policy π. If we can't further optimize the policy, we terminate with the given policy π^* and value function V^π. For the algorithm I started out with the uniform probability for each action modelled as a list $[0.25, 0.25, 0.25, 0.25]$. As evident from line 12 in algorithm 2, the algorithm chooses a new single action which optimizes the reward, for example if the was action "`up`", it is represented as $[1, 0, 0, 0]$, which is because π can map a state and an action to a probability or it can map a state to a given action, this way of representing the policy allows me to do both. Running the algorithm gives an optimal policy for each state and a value function, first I will show the policy and then the Value function. Figure 1 shows the apartment and the numbering of each room. The arrows represent the action of the optimal policy, returned from algorithm 2. Looking at the arrows, they all make sense with regards to getting to the Terminal state with as little penalty as possible. For example if we take room 3: The shortest path may be to move down instead of up, but because a penalty of 10 is enforced by moving down it moves up resulting in a total penalty of 11 instead of 14.

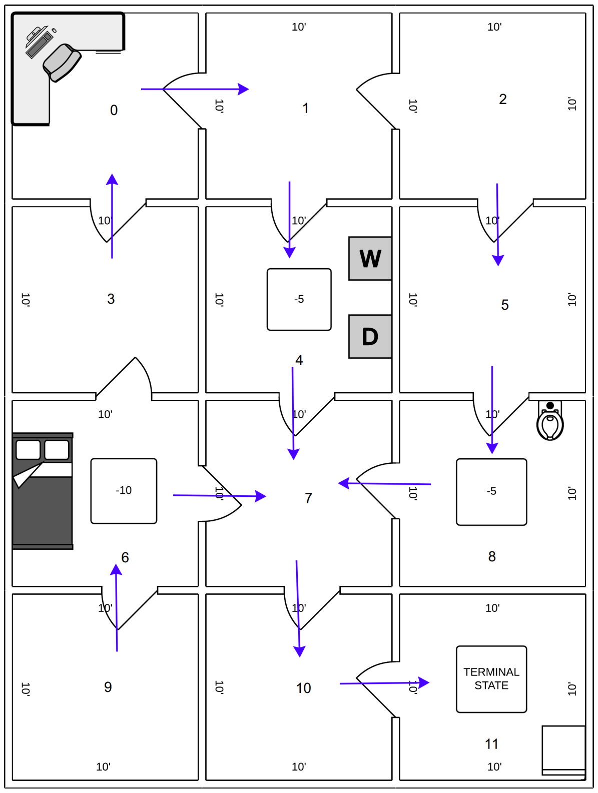

Figure 1: The returned optimal policy returned from algorithm 2. The arrows represent the optimal action for each state, e.g. the optimal action for room 5 is down.

The return value function V^* is given as followed:

$$V^* = \begin{bmatrix} -10 & -9 & -10 \\ -11 & -3 & -9 \\ -3 & -2 & -3 \\ -14 & -1 & 0 \end{bmatrix} \tag{41}$$

This value function can easily be verified by looking at figure 1, following the arrows from each state and counting the accumulated penalty, before reaching the terminal state. As with the random policy, the worst starting location is at room 9.

4 Bonus: Make Your Own Question

My proposed question is about support vector machines and the use of the dual representation of the optimization problem. I have chosen this question as I found it difficult to understand the different representations of the problem and the benefits of using these representations. This question will provide some "hands on" experience with the dual representation of the SVM optimization problem.

Question:

Use the dual representation of the SVM optimization problem to show that regardless of the feature space dimensionality of the data, we can determine a hyperplane using only two points, having different class labels:

$$x_1 \in C_1, (y_1 = +1) \tag{42}$$
$$x_2 \in C_2, (y_2 = -1) \tag{43}$$

That is we have:

$$y(x_1) = w^T x_1 + b = +1 \tag{44}$$
$$y(x_2) = w^T x_2 + b = -1 \tag{45}$$

The dual problem is defined as:

$$\text{Maximize}_\alpha \quad \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle \tag{46}$$
$$\text{subject to} \quad \sum_{i=1}^{N} \alpha_i y_i = 0 \tag{47}$$
$$\alpha_i \geq 0, i = 1, ..., N \tag{48}$$

With $N = 2$ we can reformulate the constraints as:

$$\alpha_1 - \alpha_2 = 0 \tag{49}$$
$$\alpha_1, \alpha_2 \geq 0 \tag{50}$$

The rest of the proof would consist of writing out the sums of the dual problem listed in (45) for the two points. Then taking the derivative with respect to α (for α_1, α_2), setting them to zero and getting both α. Once these have been expressed, they can be used to solve for the hyperplane using the following:

$$w = \sum_{i=1}^{N} \alpha_i y_i x_i \tag{51}$$

and solving for b:

$$b = -\frac{\max_{y_i=-1}(\langle w, x_i \rangle) + \min_{y_i=1}(\langle w, x_i \rangle)}{2} \tag{52}$$

where we know that the two data points are the support vectors.