Q-LDA: Uncovering Latent Patterns in Text-based Sequential Decision Processes

Authored by:

Li Deng
Lihong Li
Chong Wang
Lin Xiao
Jianshu Chen
Ji He

Abstract

In sequential decision making, it is often important and useful for end users to understand the underlying patterns or causes that lead to the corresponding decisions. However, typical deep reinforcement learning algorithms seldom provide such information due to their black-box nature. In this paper, we present a probabilistic model, Q-LDA, to uncover latent patterns in text-based sequential decision processes. The model can be understood as a variant of latent topic models that are tailored to maximize total rewards; we further draw an interesting connection between an approximate maximum-likelihood estimation of Q-LDA and the celebrated Q-learning algorithm. We demonstrate in the text-game domain that our proposed method not only provides a viable mechanism to uncover latent patterns in decision processes, but also obtains state-of-the-art rewards in these games.

1 Paper Body

Reinforcement learning [21] plays an important role in solving sequential decision making problems, and has seen considerable successes in many applications [16, 18, 20]. With these methods, however, it is often difficult to understand or examine the underlying patterns or causes that lead to the sequence of decisions. Being more interpretable to end users can provide more insights to the problem itself and be potentially useful for downstream applications based on these results [5]. To investigate new approaches to uncovering underlying patterns of a text-based sequential decision process, we use text games (also known as interactive fictions) [11, 19] as the experimental domain. Specifically, we focus on choice-based and hypertext-based games studied in the literature [11],

where both the action space and the state space are characterized in natural languages. At each time step, the decision maker (i.e., agent) observes one text document (i.e., observation text) that describes the current observation of the game environment, and several text documents (i.e., action texts) that characterize different possible actions that can be taken. Based on the history of these observations, the agent selects one of the provided actions and the game transits to a new state with an immediate reward. This game continues until the agent reaches a final state and receives a terminal reward. In this paper, we present a probabilistic model called Q-LDA that is tailored to maximize total rewards in a decision process. Specially, observation texts and action texts are characterized by two separate topic models, which are variants of latent Dirichlet allocation (LDA) [4]. In each topic model, topic proportions are chained over time to model the dependencies for actions or states. And ?

these proportions are partially responsible for generating the immediate/terminal rewards. We also show an interesting connection between the maximum-likelihood parameter estimation of the model and the Q-learning algorithm [22, 18]. We empirically demonstrate that our proposed method not only provides a viable mechanism to uncover latent patterns in decision processes, but also obtains state-of-the-art performance in these text games. Contribution. The main contribution of this paper is to seamlessly integrate topic modeling with Q-learning to uncover the latent patterns and interpretable causes in text-based sequential decisionmaking processes. Contemporary deep reinforcement learning models and algorithms can seldom provide such information due to their black-box nature. To the best of our knowledge, there is no prior work that can achieve this and learn the topic model in an end-to-end fashion to maximize the long-term reward. Related work. Q-LDA uses variants of LDA to capture observation and action texts in text-based decision processes. In this model, the dependence of immediate reward on the topic proportions is similar to supervised topic models [3], and the chaining of topic proportions over time to model long-term dependencies on previous actions and observations is similar to dynamic topic models [6]. The novelty in our approach is that the model is estimated in a way that aims to maximize long-term reward, thus producing near-optimal policies; hence it can also be viewed as a topic-model-based reinforcement-learning algorithm. Furthermore, we show an interesting connection to the DQN variant of Q-learning [18]. The text-game setup used in our experiment is most similar to previous work [11] in that both observations and actions are described by natural languages, leading to challenges in both representation and learning. The main difference from that previous work is that those authors treat observation-texts as Markovian states. In contrast, our model is more general, capturing both partial observability and long-term dependence on observations that are common in many text-based decision processes such as dialogues. Finally, the choice of reward function in Q-LDA share similarity with that in Gaussian pro-

2

cess temporal difference methods [9]. Organization. Section 2 describes the details of our probabilistic model, and draws a connection to the Q-learning algorithm. Section 3 presents an end-to-end learning algorithm that is based on mirror descent back-propagation. Section 4 demonstrates the empirical performance of our model, and we conclude with discussions and future work in Section 5.

2

A Probabilistic Model for Text-based Sequential Decision Processes

In this section, we first describe text games as an example of sequential decision processes. Then, we describe our probabilistic model, and relate it to a variant of Q-learning. 2.1

Sequential decision making in text games

Text games are an episodic task that proceeds in discrete time steps $t \in \{1, \dots, T\}$, where the length T may vary across different episodes. At time step t, the agent receives a text document of N words $S \in$ describing the current observation of the environment: $w_t^S \triangleq \{w_{t,n}\}_{n=1}^N$. We call these words observation text. The agent also receives A_t text documents, each of which describes a possible a action that the agent can take. We denote them by $w_t^a \triangleq \{w_{t,n}\}_{n=1}^N$ with $a \in \{1, \dots, A_t\}$, where A_t is the number of feasible actions and it could vary over time. We call these texts action texts. After the agent takes one of the provided actions, the environment transits to time $t + 1$ with a new state and an immediate reward r_t; both dynamics and reward generation may be stochastic and unknown. The new S a state then reveals a new observation text w_{t+1} and several action texts w_{t+1} for $a \in \{1, \dots, A_{t+1}\}$. The transition continues until the end of the game at step T when the agent receives a terminal reward r_T. The reward r_T depends on the ending of the story in the text game: a good ending leads to a large positive reward, while bad endings negative rewards. The goal of the agent is to maximize its cumulative reward by acting optimally in the environment. S A a At step t, given all observation texts $w_{1:t}$, all action texts $w_{1:t}$, $\{w_{1:t} : \vdots a\}$, previous actions S A $a_{1:t-1}$ and rewards $r_{1:t-1}$, the agent is to find a policy, $\pi(a_t | w_{1:t}, w_{1:t}, a_{1:t-1}, r_{1:t-1})$, a conditional 2

For notation simplicity, we assume all texts have the same length N.

2

A

a $w_{t,n}$

A

a $z_{t,n}$

N A

a $w_{t+1,n}$

A

a $z_{t+1,n}$

—A_t —

—A_{t+1} —

π_t^a

A π_{t+1}

a ?t+1
at
rt+1
at+1
?tS
?tS
S ?t+1
S ?t+1
S
S zt,n
S
S zt+1,n
S
S wt,n
S
S wt+1,n
?tA
?
rt
N
N
?
N

Figure 1: Graphical model representation for the studied sequential decision process. The bottom section shows the observation topic models, which share the same topics in S , but the topic distributions ?tS changes with time t. The top section shows the action topic models, sharing the same action topics in A , but with time varying topic distribution ?ta for each a 2 At . The middle section shows the dependence of variables between consecutive time steps. There are no plates for the observation text (bottom part of the figure) because there is only one observation text document at each time step. We follow the standard notation for graphical models by using shaded circles as observables. Since the topic distributions ?tS and ?ta and the Dirichlet parameters ?tS and ?tA (except ?1S and ?1A) are not observable, we need to use their MAP estimate to make end-to-end learning feasible; see Section 3 for details. The figure characterizes the general case where rewards appear at each time step, while in our experiments the (non-zero) rewards only appear at the end of the games. PT probability of selecting action at , that maximizes the expected long-term reward E{ ? =t ? t r? }, where 2 (0, 1) is a discount factor. In this paper, for simplicity of exposition, we focus on problems where the reward is nonzero only in the final step T . While our algorithm can be generalized to the general case (with greater complexity), this special case is an important case of RL (e.g., [20]). As a S A result, the policy is independent of r1:t 1 and its form is simplified to ?(at —w1:t , w1:t , a1:t 1). The problem setup is similar to previous work [11] in that both observations and actions are described by natural languages. For actions described by natural languages, the action space is inherently discrete and

4

large due to the exponential complexity with respect to sentence length. This is different from most reinforcement learning problems where the action spaces are either small or continuous. Here, we take a probabilistic modeling approach to this challenge: the observed variables—observation texts, action texts, selected actions, and rewards—are assumed to be generated from a probabilistic latent variable model. By examining these latent variables, we aim to uncover the underlying patterns that lead to the sequence of the decisions. We then show how the model is related to Q-learning, so that estimation of the model leads to reward maximization. 2.2

The Q-LDA model

The graphical representation of our model, Q-LDA, is depicted in Figure 1. It has two instances of topic models, one for observation texts and the other for action texts. The basic idea is to chain the topic proportions (?s in the figure) in a way such that they can influence the topic proportions in the future, thus capturing long-term effects of actions. Details of the generative models are as follows. For the observation topic model, we use the columns of S ? Dir(S)3 to denote the topics for the observation texts. For the action topic model, we use the columns of A ? Dir(A) to denote the topics for the action texts. We assume these topics do not change over time. Given the initial topic proportion Dirichlet parameters??1S and ?1A for observation and action texts respectively?the Q-LDA proceeds sequentially from t = 1 to T as follows (see Figure 1 for all latent variables). 3

S is a word-by-topic matrix. Each column is drawn from a Dirichlet distribution with hyper-parameter representing the word-emission probabilities of the corresponding topic. A is similarly defined.

3

S,

1. Draw observation text wtS as follows, (a) Draw observation topic proportions ?tS ? Dir(?tS). (b) Draw all words for the observation text wtS ? LDA(wtS —?tS , S), where LDA(?) denotes the standard LDA generative process given its topic proportion ?tS and topics S S S [4]. The latent variable zt,n indicates the topic for the word wt,n . a 2. For a = 1, ..., At , draw action text wt as follows, (a) Draw action topic proportions ?ta ? Dir(?tA). (b) Draw all words for the a-th action text using wta ? LDA(wta —?ta , A), where the latent a a variable zt,n indicates the topic for the word wt,n . S A 3. Draw the action: at ? ?b (at —w1:t , w1:t , a1:t 1), where ?b is an exploration policy for data collection. It could be chosen in different ways, as discussed in the experiment Section 4. After model learning is finished, a greedy policy may be used instead (c.f., Section 3). 4. The immediate reward rt is generated according to a Gaussian distribution with mean function ?r (?tS , ?tat , U) and variance r2 : rt ? N ?r (?tS , ?tat , U),

2 r

.

(1)

Here, we defer the definitions of ?r (?tS , ?tat , U) and its parameter U to the next section, where we draw a connection between likelihood-based learning

and Q-learning. 5. Compute the topic proportions Dirichlet parameters for the next time step t + 1 as WSS ?tS + WSA ?tat + ?1S ,

S ?t+1 =

A ?t+1 =

WAS ?tS +WAA ?tat +?1A ,

(2)

where (x) , max{x, ?} with ? being a small positive number (e.g., 10 6), at is the action selected by the agent at time t, and {WSS , WSA , WAS , WAA } are the model parameters to t be learned. Note that, besides ?tS , the only topic proportions from {?ta }A a=1 that will influence S A ?t+1 and ?t+1 is ?tat , i.e., the one corresponding to the chosen action at . Furthermore, since S A ?tS and ?tat are generated according to Dir(?tS) and Dir(?tA), respectively, ?t+1 and ?t+1 at S are (implicitly) chained over time via ?t and ?t (c.f. Figure 1). This generative process defines a joint distribution p(?) among all random variables depicted in Figure 1. Running this generative process?step 1 to 5 above for T steps until the game ends? produces one episode of the game. Now suppose we already have M episodes. In this paper, we choose to directly learn the conditional distribution of the rewards given other observations. By learning the model in a discriminative manner [2, 7, 12, 15, 23], we hope to make better predictions of the rewards for different actions, from which the agent could obtain the best policy for taking actions. This can be obtained by applying Bayes rule to the joint distribution defined by the generative process. Let ? denote all model parameters: ? = { S , A , U, WSS , WSA , WAS , WAA }. We have the following loss function ( ) M X S A min ln p(?) ln p r1:Ti —w1:Ti , w1:Ti , a1:Ti , ? , (3) ?

i=1

where p(?) denotes a prior distribution of the model parameters (e.g., Dirichlet parameters over S and A), and Ti denotes the length of the i-th episode. Let KS and KA denote the number of topics for the observation texts and action texts, and let VS and VA denote the vocabulary sizes for the observation texts and action texts, respectively. Then, the total number of learnable parameters for Q-LDA is: VS ? KS + VA ? KA + KA ? KS + (KS + KA)2 .

We note that a good model learned through Eq. (3) may predict the values of rewards well, but might not imply the best policy for the game. Next, we show by defining the appropriate mean function for the rewards, ?r (?tS , ?tat , U), we can achieve both. This closely resembles Q-learning [21, 22], allowing us to effectively learn the policy in an iterative fashion. 2.3

From Q-LDA to Q-learning

Before relating Q-LDA to Q-learning, we first give a brief introduction to the latter. Q-learning [22, 18] is a reinforcement learning algorithm for finding an optimal policy in a Markov decision process (MDP) described by (S, A, P, r,), where S is a state space, A is an action space, and 2 (0, 1) is a discount factor. Furthermore, P defines a transition probability p(s0 —s, a) for going to the next 4

state s0 2 S from the current state s 2 S after taking action a 2 A, and r(s, a) is the immediate reward corresponding to this transition. A policy ?(a—s) in

an MDP is defined to be the probability of taking action a at state s. Let st and at be the state and action at time t, and let rt = r(st , at) be the immediate reward at time t. An optimal policy is the one that maximizes the expected long-term P+1 reward E{ t=1 t 1 rt }. Q-learning seeks to find the optimal policy by estimating the Q-function, Q(s, a), defined as the expected long-term discounted reward for taking action a at state s and then following an optimal policy thereafter. It satisfies the Bellman equation [21] Q(s, a) = E{r(s, a) + ? max Q(s0 , b)—s, a} , (4) b

and directly gives the optimal action for any state s: arg maxa Q(s, a). Q-learning solves for Q(s, a) iteratively based on observed state transitions. The basic Q-learning [22] requires storing and updating the values of Q(s, a) for all state?action pairs in S ? A, which is not practical when S and A are large. This is especially true in our text games, where they can be exponentially large. Hence, Q(s, a) is usually approximated by a parametric function Q? (s, a) (e.g., neural networks [18]), in which case the model parameter ? is updated by: ? ? + ? ? r? Q? ? (dt Q? (st , at)) , (5)

where dt , rt + ? maxa0 Q?0 (st+1 , a0) if st nonterminal and dt , rt otherwise, and ?0 denotes a delayed version of the model parameter updated periodically [18]. The update rule (5) may be understood as applying stochastic gradient descent (SGD) to a regression loss function J(?) , E[dt Q? (s, a)]2 . Thus, dt is the target, computed from rt and Q?0 , for the prediction Q? (st , at). We are now ready to define the mean reward function ?r in Q-LDA. First, we model the Q-function by Q(?tS , ?ta) = (?ta)T U ?tS , where U is the same parameter as the one in (1).4 This is different from typical deep RL approaches, where black-box models like neural networks are used. In order to connect our probabilistic model to Q-learning, we define the mean reward function as follows, ? ? S b ?r (?tS , ?tat , U) = Q(?tS , ?tat) ? E max Q(?t+1 , ?t+1)—?tS , ?tat (6) b

?tat

Note that ?r remains as a function of and since the second term in the above expression is a conditional expectation given ?tS and ?tat . The definition of the mean reward function in Eq. (6) has a strong relationship with the Bellman equation (4) in Q-learning; it relates the long-term reward Q(?tS , ?tat) to the mean immediate reward ?r in the same manner as the Bellman equation (4). To see this, we move the second term on the right-hand side of (6) to the left, and make the identification that ?r corresponds to E{r(s, a)} since both of them represent the mean immediate reward. The resulting equation share a same form as the Bellman equation (4). With the mean function ?r defined above, we show in Appendix B that the loss function (3) can be approximated by the one below using the maximum a posteriori (MAP) estimate of ?tS and ?tat (denoted as ??tS and ??tat , respectively): Ti M X n i2 o X 1 h dt Q(??tS , ??tat) (7) min ln p(S — S) ln p(A — A) + 2 ? 2 r i=1 t=1 ?tS

S b where dt = rt + maxb Q(??t+1 , ??t+1) for t ¡ Ti and dt = rt for t = Ti . Observe that the first two terms in (7) are regularization terms coming from the Dirichlet prior over S and A , and the third term shares a similar form as the cost J(?) in Q-learning; it can also be interpreted as a regression problem

for estimating the Q-function, where the target dt is constructed in a similar manner as Q-learning. Therefore, optimizing the discriminative objective (3) leads to a variant of Q-learning. After learning is finished, we can obtain the greedy policy by taking the action that maximizes the Q-function estimate in any given state.

We also note that we have used the MAP estimates of ?tS and ?tat due to the intractable marginalization of the latent variables [14]. Other more advanced approximation techniques, such as Markov Chain Monte Carlo (MCMC) [1] and variational inference [13] can also be used, and we leave these explorations as future work.

3

End-to-end Learning by Mirror Descent Back Propagation

4 The intuition of choosing Q(?, ?) to be this form is that we want ?tS to be aligned with ?ta of the correct action (large Q-value), and to be misaligned with the ?ta of the wrong actions (small Q-value). The introduction of U allows the number and the meaning of topics for the observations and actions to be different.

5

Algorithm 1 The training algorithm by mirror descent back propagation 1: Input: D (number of experience replays), J (number of SGD updates), and learning rate. 2: Randomly initialize the model parameters. 3: for m = 1, . . . , D do 4: Interact with the environment using a behavior policy ?bm (at —xS1:t , xA 1:t , a1:t 1) to collect M S A M episodes of data {w1:T , w , a , r } and add them to D. 1:Ti 1:Ti i=1 1:Ti i 5: for j = 1, . . . , J do 6: Randomly sample an episode from D. 7: For the sampled episode, compute ??tS , ??ta and Q(??tS , ??ta) with a = 1, . . . , At and t = 1, . . . , Ti according to Algorithm 2. 8: For the sampled episode, compute the stochastic gradients of (7) with respect to ? using back propagation through the computational graph defined in Algorithm 2. 9: Update {U, WSS , WSA , WAS , WAA } by stochastic gradient descent and update { S , A } using stochastic mirror descent. 10: end for 11: end for Algorithm 2 The recursive MAP inference for one episode a 1: Input: ?1S , ?1A , L, , xS t , {xt : a = 1, . . . , At } and at , for all t = 1, . . . , Ti . S S 2: Initialization: ? ? 1 = ?1 and ? ? 1A = ?1A 3: for t = 1, . . . , Ti do ? h i? S ? ?S 1 1 ?S T xt t 4: Compute ??tS by repeating ??tS ? exp + for L times with initialS S ??S C t ??tS t ization ??tS / 1, where C is a normalization factor. ? h i? a ? ?A 1 1 ?a T xt t 5: Compute ??ta for each a = 1, . . . , At by repeating ??ta ? exp + A A ??a C t ??ta t a ? for L times with initialization ?t / 1, where C is a normalization factor. S A 6: Compute ? ? t+1 and ? ? t+1 from ??tS and ??tat according to (11). 7: Compute the Q-values: Q(??tS , ??ta) = (??ta)T U ??tS for a = 1, . . . , At . 8: end for In this section, we develop an end-to-end learning algorithm for Q-LDA, by minimizing the loss function given in (7). As shown in the previous section, solving (7) leads to a variant of Q-learning, thus our algorithm could be viewed as a reinforcement-learning algorithm for the proposed model. We consider learning our model with experience replay [17], a widely used technique in recent stateof-the-art systems [18]. Specifically, the learning process consists

of multiple stages, and at each stage, the agent interacts with the environment using a fixed exploration policy ?b (at —xS1:t , xA 1:t , a1:t 1) to S A M collect M episodes of data {w1:T , w , a , r } and saves them into a replay memory D. 1:T 1:T i i i=1 1:Ti i (We will discuss the choice of ?b in section 4.) Under the assumption of the generative model Q-LDA, our objective is to update our estimates of the model parameters in ? using D; the updating process may take several randomized passes over the data in D. A stage of such learning process is called one replay. Once a replay is done, we let the agent use a new behavior policy ?b0 to collect more episodes, add them to D, and continue to update ? from the augmented D. This process repeats for multiple stages, and the model parameters learned from the previous stage will be used as the initialization for the next stage. Therefore, we can focus on learning at a single stage, which was formulated in Section 2 as one of solving the optimization problem (7). Note that the objective (7) is a function of the MAP estimates of ?tS and ?tat . Therefore, we start with a recursion for computing ??tS and ??tat and then introduce our learning algorithm for ?. 3.1

Recursive MAP inference by mirror descent

The MAP estimates, ??tS and ??ta , for the topic proportions ?tS and ?ta are defined as S A (??tS , ??ta) = arg max p(?tS , ?ta —w1:t , w1:t , a1:t ?tS ,?ta

6

1)

(8)

Solving for the exact solution is, however, intractable. We instead develop an approximate algorithm that recursively estimate ??tS and ??ta . To develop the algorithm, we rely on the following result, whose proof is deferred to Appendix A. Proposition 1. The MAP estimates in (8) could be approximated by recursively solving the problems: ? ? ??tS = arg max ln p(xSt —?tS , S) + ln p ?tS —? ? tS (9) ?tS

? ??ta = arg max ln p(xat —?ta , a ?t

A)

+ ln p ?ta —? ? tA

?

,

a 2 {1, . . . , At } ,

(10)

where xSt and xat are the bag-of-words vectors for the observation text wtS and the a-th action text wta , respectively. To compute ? ? tS and ? ? tA , we begin with ? ? 1S = ?1S and ? ? 1A = ?1A and update their values for the next t + 1 time step according to ? ? ? ? S A ? ? t+1 = WSS ??tS +WSA ??tat +?1S , ? ? t+1 = WAS ??tS +WAA ??tat +?1A (11) Note from (9)?(10) that, for given ??tS and ??ta , the solution of ?tS and ?ta now becomes At +1 decoupled sub-problems, each of which has the same form as the MAP inference problem of Chen et al. [8]. Therefore, we solve each sub-problem in (9)?(10) using their mirror descent inference algorithm, and then use (11) to compute the Dirichlet parameters at the next time step. The overall MAP

9

inference procedure is summarized in Algorithm 2. We further remark that, after obtaining $\hat{\theta}_t^S$ and $\hat{\theta}_t^a$, the Q-value for the t step is readily estimated by: $? ? S A E Q(?tS , ?ta)—w1:t , w1:t , a1:t 1 ? Q(??tS , ??ta), a 2 \{1, . . . , At \} , (12)$

where we approximate the conditional expectation using the MAP estimates. After learning is finished, the agent may extract a greedy policy for any state s by taking the action arg maxa Q(??S , ??a). It is known that if the learned Q-function is closed to the true Q-function, such a greedy policy is near-optimal [21]. 3.2

End-to-end learning by backpropagation

The training loss (7) for each learning stage has the form of a finite sum over M episodes. Each term inside the summation depends on ??tS and ??tat , which in turn depend on all the model parameters in ? via the computational graph defined by Algorithm 2 (see Appendix E for a diagram of the graph). Therefore, we can learn the model parameters in ? by sampling an episode in the data, computing the corresponding stochastic gradient in (7) by back-propagation on the computational graph given in Algorithm 2, and updating ? by stochastic gradient/mirror descent. More details are found in Algorithm 1, and Appendix E.4 gives the gradient formulas.

4

Experiments

In this section, we use two text games from [11] to evaluate our proposed model and demonstrate the idea of interpreting the decision making processes: (i) ?Saving John? and (ii) ?Machine of Death? (see Appendix C for a brief introduction of the two games).5 The action spaces of both games are defined by natural languages and the feasible actions change over time, which is a setting that Q-LDA is designed for. We choose to use the same experiment setup as [11] in order to have a fair comparison with their results. For example, at each m-th experience-replay learning (see Algorithm 1), we use the softmax action selection rule [21, pp.30?31] as the exploration policy to collect data (see Appendix E.3 for more details). We collect M = 200 episodes of data (about 3K time steps in ?Saving John? and 16K in ?Machine of Death?) at each of D = 20 experience replays, which amounts to a total of 4, 000 episodes. At each experience replay, we update the model with 10 epochs before the next replay. Appendix E provides additional experimental details. We first evaluate the performance of the proposed Q-LDA model by the long-term rewards it receives when applied to the two text games. Similar to [11], we repeat our experiments for five times with different random initializations. Table 1 summarize the means and standard deviations of the rewards 5

The simulators are obtained from https://github.com/jvking/text-games

7

Table 1: The average rewards (higher is better) and standard deviations of different models on the two tasks. For DRRN and MA-DQN, the number of topics becomes the number of hidden units per layer. Tasks Saving John Machine of Death

topics 20 50 100 20 50 100

Q-LDA 18.8 (0.3) 18.6 (0.6) 19.1 (0.6) 19.9 (0.8) 18.7 (2.1) 17.5 (2.4)
DRRN (1-layer) 17.1 (0.6) 18.3 (0.2) 18.2 (0.2) 7.2 (1.5) 8.4 (1.3) 8.7 (0.9)
DRRN (2-layer) 18.4 (0.1) 18.5 (0.3) 18.7 (0.4) 9.2 (2.1) 10.7 (2.7) 11.2 (0.6)
MA-DQN (2-layer) 4.9 (3.2) 9.0 (3.2) 7.1 (3.1) 2.8 (0.9) 4.3 (0.9) 5.2 (1.2)

on the two games. We include the results of Deep Reinforcement Relevance Network (DRRN) proposed in [11] with different hidden layers. In [11], there are several variants of DQN (deep Q-networks) baselines, among which MA-DQN (max-action DQN) is the best performing one. We therefore only include the results of MA-DQN. Table 1 shows that Q-LDA outperforms all other approaches on both tasks, especially ?Machine of Death?, where Q-LDA even beats the DRRN models by a large margin. The gain of Q-LDA on ?Saving John? is smaller, as both Q-LDA and DRRN are approaching the upper bound of the reward, which is 20. ?Machine of Death? was believed to be a more difficult task due to its stochastic nature and larger state and action spaces [11], where the upper bound on the reward is 30. (See Tables 4?5 for the definition of the rewards for different story endings.) Therefore, Q-LDA gets much closer to the upper bound than any other method, although there may still be room for improvement. Finally, our experiments follow the standard online RL setup: after a model is updated based on the data observed so far, it is tested on newly generated episodes. Therefore, the numbers reported in Table 1 are not evaluated on the training dataset, so they truthfully reflect the actual average reward of the learned models. We now proceed to demonstrate the analysis of the latent pattern of the decision making process using one example episode of ?Machine of Death?. In this episode, the game starts with the player wandering in a shopping mall, after the peak hour ended. The player approaches a machine that prints a death card after inserting a coin. The death card hints on how the player will die in future. In one of the story development, the player?s death is related to a man called Bon Jovi. The player is so scared that he tries to combat with a cardboard standee of Bon Jovi. He reveals his concern to a friend named Rachel, and with her help he finally overcomes his fear and maintains his friendship. This episode reaches a good ending and receives the highest possible reward of 30 in this game. In Figure 2, we show the evolution of the topic proportions for the four most active topics (shown in Table 2)6 for both the observation texts and the selected actions? texts. We note from Figure 2 that the most dominant observation topic and action topic at beginning of the episode are ?wander at mall? and ?action at mall?, respectively, which is not surprising since the episode starts at a mall scenario. The topics related to ?mall? quickly dies off after the player starts the death machine. Afterwards, the most salient observation topic becomes ?meet Bon Jovi? and then ?combat? ($t = 8$). This is because after the activation of death machine, the story enters a scenario where the player tries to combat with a cardboard standee. Towards the end of the episode, the observation topic ?converse w/rachel? and the topic ?kitchen & chat? corresponding to the selected action reach their peaks and then decay right before the end of the story, where the action topic ?relieve? climbs up to its peak. This is consistent with the story ending, where the player chooses to overcome his fear after chatting with Rachel. In Appendix D, we show

the observation and the action texts in the above stages of the story. Finally, another interesting observation is about the matrix U . Since the Q-function value is computed from [??ta]T U ??tS , the (i, j)-th element of the matrix U measures the positive/negative correlation between the i-th action topic and the j-th observation topic. In Figure 2(c), we show the value of the learned matrix U for the four observation topics and the four action topics in Table 2. Interestingly, the largest value (39.5) of U is the (1, 2)-th element, meaning that the action topic ?relieve? and the state topic ?converse w/rachel? has strong positive contribution to a high long-term reward, which is what happens at the end of the story. 6 In practice, we observe that some topics are never or rarely activated during the learning process. This is especially true when the number of topics becomes large (e.g., 100). Therefore, we only show the most active topics. This might also explain why the performance improvement is marginal when the number of topics grows.

8

Table 2: The four most active topics for the observation texts and the action texts, respectively. Observation Topics 1: combat 2: converse w/ rachel 3: meet Bon Jovi 4: wander at mall Action Topics 1: relieve 2: kitchen & chat 3: operate the machine 4: action at mall 1 0.8

minutes, lights, firearm, shoulders, whiff, red, suddenly, huge, rendition rachel, tonight, grabs, bar, towards, happy, believing, said, moonlight small, jovi, bon, door, next, dog, insists, room, wrapped, standees ended, catcher, shopping, peak, wrapped, hanging, attention, door leave, get, gotta, go, hands, away, maybe, stay, ability, turn, easy, rachel wait, tea, look, brisk, classics, oysters, kitchen, turn, chair, moment coin, insert, west, cloth, desk, apply, dollars, saying, hands, touch, tell alarm, machine, east, ignore, take, shot, oysters, win, gaze, bestowed 1

Observation Topic 1 Observation Topic 2 Observation Topic 3 Observation Topic 4

0.8
0.6
0.6
0.4
0.4
0.2
0.2
0
5
10
(a) Observation topic
15
?tS
0
Action Topic 1 Action Topic 2 Action Topic 3 Action Topic 4
2
1.2 622.1 4 2.5 5.3

5

10

(b) Selected action topic

39.5 12.4 4.8 8.4

20.7 1.4 4.1 13.3

3 12.2 0.27 1.9 5 4.1

15

?tat

(c) Learned values of matrix U

Figure 2: The evolution of the most active topics in ?Machine of Death.?

5

Conclusion

We proposed a probabilistic model, Q-LDA, to uncover latent patterns in text-based sequential decision processes. The model can be viewed as a latent topic model, which chains the topic proportions over time. Interestingly, by modeling the mean function of the immediate reward in a special way, we showed that discriminative learning of Q-LDA using its likelihood is closely related to Q-learning. Thus, our approach could also be viewed as a Q-learning variant for sequential topic models. We evaluate Q-LDA on two text-game tasks, demonstrating state-of-the-art rewards in these games. Furthermore, we showed our method provides a viable approach to finding interesting latent patterns in such decision processes. Acknowledgments The authors would like to thank all the anonymous reviewers for their constructive feedback.

2 References

[1] Christophe Andrieu, Nando De Freitas, Arnaud Doucet, and Michael I Jordan. An introduction to MCMC for machine learning. Machine learning, 50(1):5?43, 2003. [2] C. M. Bishop and J. Lasserre. Generative or discriminative? getting the best of both worlds. Bayesian Statistics, 8:3?24, 2007. [3] D. M. Blei and J. D. Mcauliffe. Supervised topic models. In Proc. NIPS, pages 121?128, 2007. [4] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet allocation. JMLR, 3:993?1022, 2003. [5] David M Blei. Probabilistic topic models. Communications of the ACM, 55(4):77?84, 2012. [6] David M Blei and John D Lafferty. Dynamic topic models. In Proceedings of the 23rd international conference on Machine learning, pages 113?120. ACM, 2006. 9

[7] G. Bouchard and B. Triggs. The tradeoff between generative and discriminative classifiers. In Proc. COMPSTAT, pages 721?728, 2004. [8] Jianshu Chen, Ji He, Yelong Shen, Lin Xiao, Xiaodong He, Jianfeng Gao, Xinying Song, and Li Deng. End-to-end learning of lda by mirror-descent back propagation over a deep architecture. In Proc. NIPS, pages 1765?1773, 2015. [9] Yaakov Engel, Shie Mannor, and Ron Meir. Reinforcement learning with Gaussian processes. In Proceedings of the Twenty-Second International Conference on Machine Learning (ICML-05), pages 201?208, 2005. [10] Matthew Hausknecht and Peter Stone. Deep recurrent Q-learning for partially observable MDPs. In

Proc. AAAI-SDMIA, November 2015. [11] Ji He, Jianshu Chen, Xiaodong He, Jianfeng Gao, Lihong Li, Li Deng, and Mari Ostendorf. Deep reinforcement learning with a natural language action space. In Proc. ACL, 2016. [12] A. Holub and P. Perona. A discriminative framework for modelling object classes. In Proc. IEEE CVPR, volume 1, pages 664?671, 2005. [13] Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. An introduction to variational methods for graphical models. In Learning in graphical models, pages 105?161. Springer, 1998. [14] Michael Irwin Jordan. Learning in graphical models, volume 89. Springer Science & Business Media, 1998. [15] S. Kapadia. Discriminative Training of Hidden Markov Models. PhD thesis, University of Cambridge, 1998. [16] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. Journal of Machine Learning Research, 17(1):1334?1373, 2016. [17] Long-Ji Lin. Reinforcement learning for robots using neural networks. Technical report, Technical report, DTIC Document, 1993. [18] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. Nature, 518:529?533, 2015. [19] Karthik Narasimhan, Tejas Kulkarni, and Regina Barzilay. Language understanding for textbased games using deep reinforcement learning. In Proc. EMNLP, 2015. [20] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. Nature, 529:484?489, 2016. [21] Richard S Sutton and Andrew G Barto. Reinforcement learning: An introduction. MIT press Cambridge, 1998. [22] Christopher Watkins and Peter Dayan. Q-learning. Machine learning, 8(3-4):279?292, 1992. [23] Oksana Yakhnenko, Adrian Silvescu, and Vasant Honavar. Discriminatively trained Markov model for sequence classification. In Proc. IEEE ICDM, 2005.

10