Learning how to Active Learn: A Deep Reinforcement Learning Approach

Meng Fang, Yuan Li, Trevor Cohn

The University of Melbourne

Presenter: Jialin Song

April 05, 2018

Overview

Introduction: Active Learning

1. Annotation:

Introduction: Active Learning

1. Annotation:
   - select a subset of data to annotate from a large unlabelled dataset (adding labels)

Introduction: Active Learning

1. Annotation:
   - select a subset of data to annotate from a large unlabelled dataset (adding labels)
   - then we can train a supervised learning model ϕ (classifier)

Introduction: Active Learning

1. Annotation:
   - select a subset of data to annotate from a large unlabelled dataset (adding labels)
   - then we can train a supervised learning model ϕ (classifier)
   - we hope to maximize the accuracy of the classification model

Introduction: Active Learning

1. Annotation:
   - select a subset of data to annotate from a large unlabelled dataset (adding labels)
   - then we can train a supervised learning model ϕ (classifier)
   - we hope to maximize the accuracy of the classification model
2. Active learning:

Introduction: Active Learning

1. Annotation:
   - select a subset of data to annotate from a large unlabelled dataset (adding labels)
   - then we can train a supervised learning model ϕ (classifier)
   - we hope to maximize the accuracy of the classification model
2. Active learning:
   - there is high cost annotating every sentence

Introduction: Active Learning

1. Annotation:
   - select a subset of data to annotate from a large unlabelled dataset (adding labels)
   - then we can train a supervised learning model ϕ (classifier)
   - we hope to maximize the accuracy of the classification model
2. Active learning:
   - there is high cost annotating every sentence
   - how to select raw data to add labels in order to maximize the accuracy of the classification model

Introduction: Active Learning

1. Annotation:
   - select a subset of data to annotate from a large unlabelled dataset (adding labels)
   - then we can train a supervised learning model ϕ (classifier)
   - we hope to maximize the accuracy of the classification model
2. Active learning:
   - there is high cost annotating every sentence
   - how to select raw data to add labels in order to maximize the accuracy of the classification model
   - active learning becomes a sequential decision: as each sentence arrives, annotate it or not (our action)

Introduction: MDP

1. Markov Decision Process (MDP):

Introduction: MDP

1. Markov Decision Process (MDP):
   - a framework to model a **sequential** decision process

Introduction: MDP

1. Markov Decision Process (MDP):
   - a framework to model a **sequential** decision process
   - in each decision stage, agent observes state variables (s) and take a action (a) to maximize its current payoff

Introduction: MDP

1. Markov Decision Process (MDP):
   - a framework to model a **sequential** decision process
   - in each decision stage, agent observes state variables (s) and take a action (a) to maximize its current payoff
   - after taking the action, a reward associated with the action and state ($r(s, a)$) is generated and current state transits to next state

Introduction: MDP

1. Markov Decision Process (MDP):
   - a framework to model a **sequential** decision process
   - in each decision stage, agent observes state variables (s) and take a action (a) to maximize its current payoff
   - after taking the action, a reward associated with the action and state ($r(s, a)$) is generated and current state transits to next state
   - agent aims maximizing the **expected value** of rewards over all stages

Introduction: Bellman Equation

1. The dynamics of MDP can be modeled in Bellman equations

Introduction: Bellman Equation

1. The dynamics of MDP can be modeled in Bellman equations
   - Bellman equation 1: value function

$$J(s) = \max_a \left\{ \bar{r}(s,a) + \alpha \sum_{s'} P_{ss'}(a) J(s') \right\}$$

$$a_s^* = \operatorname{argmax} J(s)$$

Introduction: Bellman Equation

1. The dynamics of MDP can be modeled in Bellman equations
   - Bellman equation 1: value function

$$J(s) = \max_a \left\{ \bar{r}(s,a) + \alpha \sum_{s'} P_{ss'}(a) J(s') \right\}$$

$$a_s^* = \operatorname{argmax} J(s)$$

   - Bellman equation 2 (more common!): **Q-function**

$$Q(s,a) = \bar{r}(s,a) + \alpha \sum_{s'} P_{ss'}(a) \max_u Q(s',u)$$

$$a_s^* = \operatorname{argmax} Q(s,a)$$

Introduction: Bellman Equation

1. The dynamics of MDP can be modeled in Bellman equations
   - Bellman equation 1: value function

$$J(s) = \max_a \left\{ \bar{r}(s,a) + \alpha \sum_{s'} P_{ss'}(a) J(s') \right\}$$

$$a_s^* = \operatorname{argmax} J(s)$$

   - Bellman equation 2 (more common!): **Q-function**

$$Q(s,a) = \bar{r}(s,a) + \alpha \sum_{s'} P_{ss'}(a) \max_u Q(s',u)$$

$$a_s^* = \operatorname{argmax} Q(s,a)$$

   - where $\bar{r}(s,a)$ is the expected reward, $P_{ss'}(a)$ is the transition probability from state s to s', α is the discount of reward

Q-Learning

1. If $P_{ss'}(a)$ is known, then solve the Bellmen equations (VI/PI) to get the optimal policy. There is no need to 'learn' !!!

Q-Learning

1. If $P_{ss'}(a)$ is known, then solve the Bellmen equations (VI/PI) to get the optimal policy. There is no need to 'learn' !!!

2. If $P_{ss'}(a)$ is not known, then how to compute Q-function becomes a learning problem

Q-Learning

1. If $P_{ss'}(a)$ is known, then solve the Bellmen equations (VI/PI) to get the optimal policy. There is no need to 'learn' !!!
2. If $P_{ss'}(a)$ is not known, then how to compute Q-function becomes a learning problem
3. Q-learning:

Q-Learning

1. If $P_{ss'}(a)$ is known, then solve the Bellmen equations (VI/PI) to get the optimal policy. There is no need to 'learn' !!!

2. If $P_{ss'}(a)$ is not known, then how to compute Q-function becomes a learning problem

3. Q-learning:

   - $Q_{t+1}(s_t, a_t) = (1 - \epsilon_t) Q_t(s_t, a_t) + \epsilon_t \big(\bar{r}(s_t, a_t) + \alpha \max_u Q_t(s_{t+1}, u)\big)$

Q-Learning

1. If $P_{ss'}(a)$ is known, then solve the Bellmen equations (VI/PI) to get the optimal policy. There is no need to 'learn' !!!

2. If $P_{ss'}(a)$ is not known, then how to compute Q-function becomes a learning problem

3. Q-learning:

   - $Q_{t+1}(s_t, a_t) = (1 - \epsilon_t) Q_t(s_t, a_t) + \epsilon_t \big(\bar{r}(s_t, a_t) + \alpha \max_u Q_t(s_{t+1}, u)\big)$
   - where t is iteration and ϵ_t is the learning rate

Q-Learning

1. If $P_{ss'}(a)$ is known, then solve the Bellmen equations (VI/PI) to get the optimal policy. There is no need to 'learn' !!!

2. If $P_{ss'}(a)$ is not known, then how to compute Q-function becomes a learning problem

3. Q-learning:

   - $Q_{t+1}(s_t, a_t) = (1 - \epsilon_t) Q_t(s_t, a_t) + \epsilon_t \big(\bar{r}(s_t, a_t) + \alpha \max_u Q_t(s_{t+1}, u)\big)$
   - where t is iteration and ϵ_t is the learning rate
   - In practice, above is useless: $|S| \times |A|$ is huge

Deep Q-Learning

1. Deep Q-learning:

Deep Q-Learning

1. Deep Q-learning:
   - use the output of a DNN parametrized by θ, i.e., $f_\theta(s, u)$ to approximate $Q(s, a)$:

Deep Q-Learning

1. Deep Q-learning:
   - use the output of a DNN parametrized by θ, i.e., $f_\theta(s, u)$ to approximate $Q(s, a)$:
   - input: state s, action a, reward $r(s, a)$, state transition s'

Deep Q-Learning

1. Deep Q-learning:
   - use the output of a DNN parametrized by θ, i.e., $f_\theta(s, u)$ to approximate $Q(s, a)$:
   - input: state s, action a, reward $r(s, a)$, state transition s'
   - output: approximation of Q-function: $f_\theta(s, u)$

Deep Q-Learning

1. Deep Q-learning:
   - use the output of a DNN parametrized by θ, i.e., $f_\theta(s, u)$ to approximate $Q(s, a)$:
   - input: state s, action a, reward $r(s, a)$, state transition s'
   - output: approximation of Q-function: $f_\theta(s, u)$
   - the loss function minimization

$$\min_\theta \left\{ \frac{1}{2} \left(f_{\theta_t}(s_t, a_t) - \bar{r}(s_t, a_t) - \alpha \max_u f_{\theta_t}(s_{t+1}, u) \right)^2 \right\}$$

Model Active Learning as MDP

1. sentence x_i from an unlabelled dataset arrives one by one

Model Active Learning as MDP

1. sentence x_i from an unlabelled dataset arrives one by one
2. for each arriving sentence, agent decides whether to annotate it or not (binary action)

Model Active Learning as MDP

1. sentence x_i from an unlabelled dataset arrives one by one
2. for each arriving sentence, agent decides whether to annotate it or not (binary action)
3. if agent annotates it, then the annotated set (labelled set) gets expanded, and the classifier ϕ is re-trained and updated

Model Active Learning as MDP

1. sentence x_i from an unlabelled dataset arrives one by one
2. for each arriving sentence, agent decides whether to annotate it or not (binary action)
3. if agent annotates it, then the annotated set (labelled set) gets expanded, and the classifier ϕ is re-trained and updated
4. evaluate the updated classier on a separate independent dataset, get the test accuracy (reward)

Model Active Learning as MDP

1. sentence x_i from an unlabelled dataset arrives one by one
2. for each arriving sentence, agent decides whether to annotate it or not (binary action)
3. if agent annotates it, then the annotated set (labelled set) gets expanded, and the classifier ϕ is re-trained and updated
4. evaluate the updated classier on a separate independent dataset, get the test accuracy (reward)
5. next sentence arrives

Model Active Learning as MDP

1. Sate (s): comprised of 2 parts:

Model Active Learning as MDP

1. Sate (s): comprised of 2 parts:
   - input setence: x_i (encoded using a CNN, h_c)

Model Active Learning as MDP

1. Sate (s): comprised of 2 parts:
   - input setence: x_i (encoded using a CNN, h_c)
   - trained classification model ϕ (encoded using a CNN, h_e)

Model Active Learning as MDP

1. Sate (s): comprised of 2 parts:
   - input setence: x_i (encoded using a CNN, h_c)
   - trained classification model ϕ (encoded using a CNN, h_e)
2. Action (a):

Model Active Learning as MDP

1. Sate (s): comprised of 2 parts:
   - input setence: x_i (encoded using a CNN, h_c)
   - trained classification model ϕ (encoded using a CNN, h_e)
2. Action (a):
   - $a_i = 1$: annotate x_i

Model Active Learning as MDP

1. Sate (s): comprised of 2 parts:
   - input setence: x_i (encoded using a CNN, h_c)
   - trained classification model ϕ (encoded using a CNN, h_e)
2. Action (a):
   - $a_i = 1$: annotate x_i
   - $a_i = 0$: not annotate x_i

Model Active Learning as MDP

1. Sate (s): comprised of 2 parts:
   - input setence: x_i (encoded using a CNN, h_c)
   - trained classification model ϕ (encoded using a CNN, h_e)
2. Action (a):
   - $a_i = 1$: annotate x_i
   - $a_i = 0$: not annotate x_i
3. Reward (r):

Model Active Learning as MDP

1. Sate (s): comprised of 2 parts:
   - input setence: x_i (encoded using a CNN, h_c)
   - trained classification model ϕ (encoded using a CNN, h_e)
2. Action (a):
   - $a_i = 1$: annotate x_i
   - $a_i = 0$: not annotate x_i
3. Reward (r):
   - evaluate the classification model on a held-out set after the action a is taken and get the test accuracy

An Value Iteration Q-learning Algorithm

Algorithm 1 Learn an active learning policy

Input: data $\mathcal{D}$, budget $\mathcal{B}$

Output: π

for episode = $1, 2, \ldots, N$ **do** — stopping criteria
  $\mathcal{D}_l \leftarrow \emptyset$ and shuffle $\mathcal{D}$
  $\phi \leftarrow$ Random
  for $i \in \{0, 1, 2, \ldots, |\mathcal{D}|\}$ **do** — sentence input
    Construct the state s_i using $\mathbf{x}_i$
    The agent makes a decision according to $a_i = \arg\max Q^{\pi}(s_i, a)$
    if $a_i = 1$ **then** — action: annoate
      Obtain the annotation $\mathbf{y}_i$
      $\mathcal{D}_l \leftarrow \mathcal{D}_l + (\mathbf{x}_i, \mathbf{y}_i)$ — annotated training set expanded
      Update model ϕ based on $\mathcal{D}_l$
    end if — train and update classifier
    Receive a reward r_i using held-out set
    if $|\mathcal{D}_l| = \mathcal{B}$ **then** — test classifier on a separate set
      Store $(s_i, a_i, r_i, \text{Terminate})$ in $\mathcal{M}$
      Break
    end if
    Construct the new state s_{i+1} — transition to 2nd sentence
    Store transition (s_i, a_i, r_i, s_{i+1}) in $\mathcal{M}$
    Sample random minibatch of transitions $\{(s_j, a_j, r_j, s_{j+1})\}$ from $\mathcal{M}$, and perform gradient descent step on $\mathcal{L}(\theta)$ — update NN (Q) using SGD
    Update policy π with θ — argmax of updated Q function (NN output)
  end for
end for
return the latest policy π — \pi_n-\pi_{n+1} < \epsilon

Important Step 1

for $i \in \{0, 1, 2, \ldots, |\mathcal{D}|\}$ **do**

Construct the state s_i using $\mathbf{x}_i$

The agent makes a decision according to $a_i = \arg\max Q^{\pi}(s_i, a)$

if $a_i = 1$ **then**

Obtain the annotation $\mathbf{y}_i$

$\mathcal{D}_l \leftarrow \mathcal{D}_l + (\mathbf{x}_i, \mathbf{y}_i)$

Update model ϕ based on $\mathcal{D}_l$

end if

Receive a reward r_i using held-out set

Important Step 2

Construct the new state s_{i+1}
Store transition (s_i, a_i, r_i, s_{i+1}) in $\mathcal{M}$
Sample random minibatch of transitions $\{(s_j, a_j, r_j, s_{j+1})\}$ from $\mathcal{M}$, and perform gradient descent step on $\mathcal{L}(\theta)$
Update policy π with θ
end for

Important Step 3

for episode $= 1, 2, \ldots, N$ **do**

1. Remarks on the Q-learning algorithm:

Important Step 3

for episode $= 1, 2, \ldots, N$ **do**

1. Remarks on the Q-learning algorithm:
   - input: unlabelled dataset D

Important Step 3

for episode $= 1, 2, \ldots, N$ **do**

1. Remarks on the Q-learning algorithm:
   - input: unlabelled dataset D
   - output: a series of actions (a_i): policy π

Relaxation 1: Transfer Policy

1. train annotation policy π in source language (e.g., English) and transfer it to low-source target language

Relaxation 1: Transfer Policy

1. train annotation policy π in source language (e.g., English) and transfer it to low-source target language

Algorithm 2 Active learning by policy transfer

Input: unlabelled data $\mathcal{D}$, budget $\mathcal{B}$, policy π

Output: $\mathcal{D}_l$

target low-source language

well trained policy on one language

$\mathcal{D}_l \leftarrow \emptyset$
$\phi \leftarrow$ Random
for $|\mathcal{D}_l| \neq \mathcal{B}$ and $\mathcal{D}$ not empty **do**
  Randomly sample $\mathbf{x}_i$ from the data pool $\mathcal{D}$ and construct the state s_i
  The agent chooses an action a_i according to $a_i = \arg\max Q^{\pi}(s_i, a)$
  if $a_i = 1$ **then**
    Obtain the annotation $\mathbf{y}_i$
    $\mathcal{D}_l \leftarrow \mathcal{D}_l + (\mathbf{x}_i, \mathbf{y}_i)$
    Update model ϕ based on $\mathcal{D}_l$
  end if
  $\mathcal{D} \leftarrow \mathcal{D} \backslash \mathbf{x}_i$ exclude the annotated sentence from raw dataset
  Receive a reward r_i using held-out set
  Update policy π
end for only one episode to mimic the source scarcity
return $\mathcal{D}_l$

Relaxation 2: Transfer Model and Policy

1. train a classification model ϕ and annotation policy π in source language (e.g., English) and transfer both to low-source target language

Relaxation 2: Transfer Model and Policy

1. train a classification model ϕ and annotation policy π in source language (e.g., English) and transfer both to low-source target language
2. this relaxation is more like a test and implementation procedure

Algorithm 3 Active learning by policy *and* model transfer, for 'cold-start' scenario

Input: unlabelled data $\mathcal{D}$, budget $\mathcal{B}$, policy π, model ϕ

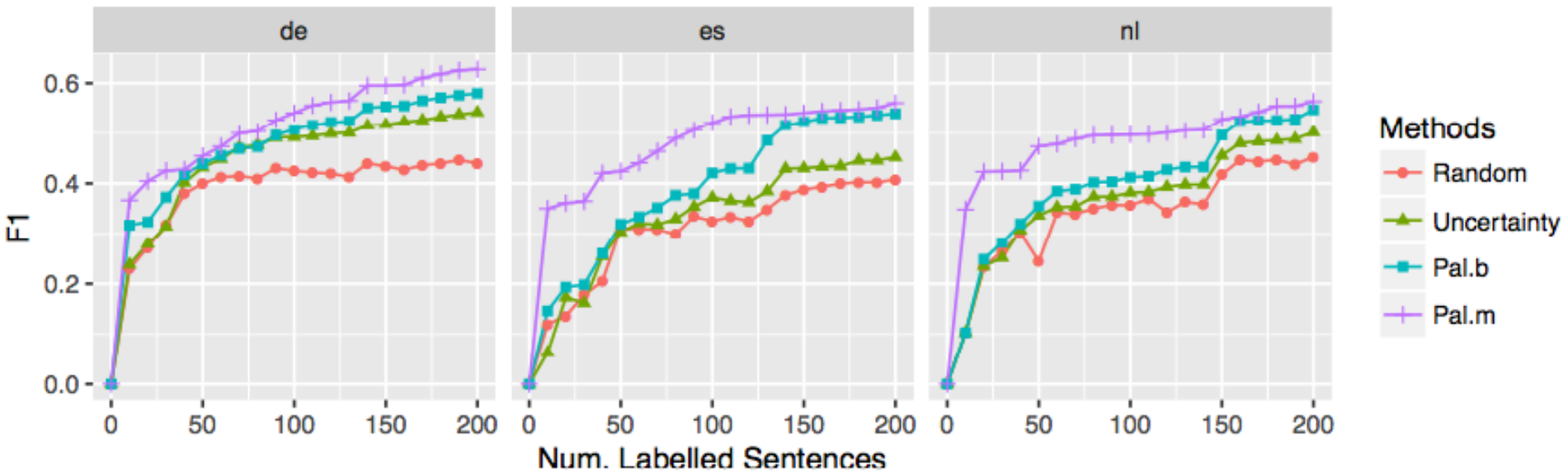
target dataset; trained policy; trained classification model

Output: $\mathcal{D}_l$

1: $\mathcal{D}_l \leftarrow \emptyset$
2: **for** $|\mathcal{D}_l| \neq \mathcal{B}$ and $\mathcal{D}$ not empty **do**
3: Randomly sample $\mathbf{x}_i$ from the data pool $\mathcal{D}$ and construct the state s_i
4: The agent chooses an action a_i according to $a_i = \arg\max Q^{\pi}(s_i, a)$
5: **if** $a_i = 1$ **then**
6: $\mathcal{D}_l \leftarrow \mathcal{D}_l + (\mathbf{x}_i, -)$ annotate based on \phi
7: **end if**
8: $\mathcal{D} \leftarrow \mathcal{D} \backslash \mathbf{x}_i$
9: **end for**

Numerical Experiments

A couple of numerical experiments show that the newly proposed active learning approach by deep Q-learning works better than some existing active learning methods such as uncertainty sampling and random sampling.



Thank You!

.....Question?