Research Presentation

Active Learning with V-Learning

Thorben Werner

March 31, 2022

Information Systems and Machine Learning Lab (ISMLL)
Institute for Computer Science
University of Hildesheim

Methodology

Pool-Based Active Learning

$\mathcal{L} \in \mathbb{R}^{\lambda \times m}$ Labeled Set

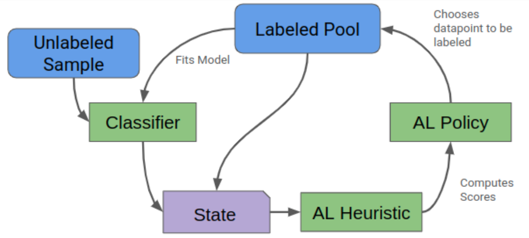
$\mathcal{U} \in \mathbb{R}^{\mu \times m}$ Unlabeled Set

$\phi_\theta := \mathbb{R}^m \rightarrow \mathbb{R}^c$ Classifier

$\mathcal{K} \in \mathbb{R}^{k \times m}$ Unlabeled Sample

$\psi := \mathbb{R}^{k \times m} \rightarrow \mathbb{R}^k$ Active Learning Heuristic

$\pi_\psi := \mathit{argmax}\ \psi(\mathcal{S})$ Active Learning Policy



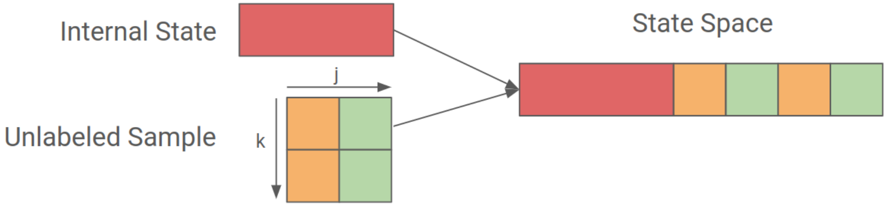
Active Learning Cycle

Active Learning with Q-Learning

State Space:

1. Internal state of the environment: content of labeled pool $\mathcal{L}$, state of the classifier θ, remaining budget, current F1-Score, etc. $\rightarrow \mathbb{R}^{i}$
2. Information about the unlabeled sample $\mathcal{K}$: Output of the classifier $\phi_\theta(k_t)$, the datapoints themselves, other metrics, etc. $\rightarrow \mathbb{R}^{k \times j}$

Results in a flattened state space of $\mathcal{S} \in \mathbb{R}^{i+k \times j}$



Active Learning with Q-Learning

$\mathcal{S} \in \mathbb{R}^{i+k\times j}$ State Space

$\mathcal{A} \in [0, \ldots, k]$ Action Space

$\mathcal{R} := \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ Reward Function

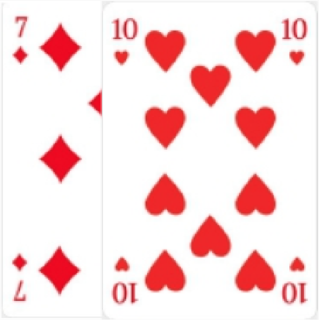
$\tau := \{\mathcal{S}, \mathcal{A}, \mathcal{S}, \mathcal{R}, \mathbb{R}\}$ Transition

Problems:

- Fixed Sample size
- Expensive Transitions
- Same actions in different places

Example: Pick a Card

- Consider a cardgame where the agent needs to collect cards with high value
- Each card is worth a certain number of points, and the goal is to collect the maximum amount of points
- At each iteration the agent is presented two cards to choose from
- Q-Learning is not the correct choice of agent here



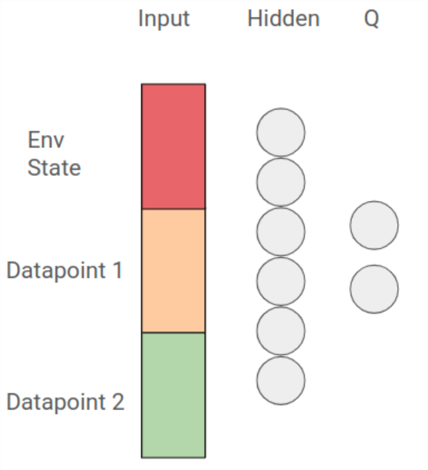
Active Learning with Q-Learning

Sample Size $k = 2$

The same datapoint can appear in multiple places in the input

Both output nodes essentially learn the same function since the datapoints are sampled randomly and independently.

A permutation invariant network can fix the problem and create a ranking of actions



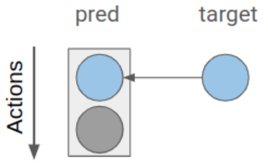
The Bellman Target

Q-Learning Target: What is the expected discounted improvement in F1-Score under the current policy when we choose a given datapoint?

The Bellman target does not rank actions, but tries to estimate their independent, global value under the current policy

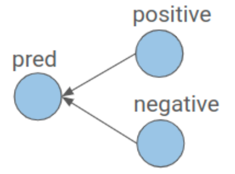
Reinforcement Learning

$$\mathcal{L}_Q := \hat{\pi}(s_t, a_t) - r_t + \gamma \max_a(\pi(s_{t+1}))$$



Ranking (Triplet Loss)

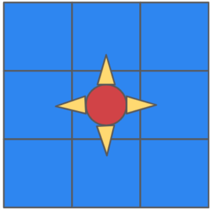
$$\mathcal{L}_{triplet} := max(0, m + d(z_a, z_p) - d(z_a, z_n))$$



V-Learning vs Q-Learning

Q-Learning:

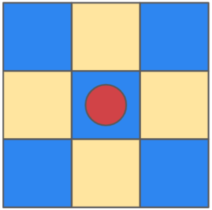
$Q(s_t) = \mathbb{R}^4$



$$Q(s_t) = \begin{bmatrix} V(s_{t+1}(\text{up}) \\ V(s_{t+1}(\text{down}) \\ V(s_{t+1}(\text{left}) \\ V(s_{t+1}(\text{right}) \end{bmatrix}$$

V-Learning:

$V(s_{t+1}(a)) = \mathbb{R} \quad \text{with} \quad a = [1, \ldots, 4]$



$$Q(s_t, a) = V(s_{t+1}(a))$$

Adaptations

- We use the batch dimension ϕ for representing the sample size k
- No action space needed

$\mathcal{S} \in \mathbb{R}^{i+j}$ State Space

$\mathcal{R} := \mathcal{S} \rightarrow \mathbb{R}$ Reward Function

$V_\theta := \mathcal{S} \rightarrow \mathbb{R}$ Agent Network

Example:

$$
\begin{aligned}
\text{State is generated} &: s \in \mathbb{R}^{\phi \times (i+j)} \\
\text{Agents makes prediction} &: v = V_\theta(s) \\
\textbf{Policy selects a point} &: a = \operatorname*{argmax}_{\phi} v_\phi \\
\text{Action is applied} &: s' = \text{env}(a) \\
\text{Reward is observed} &: r = \mathcal{R}(s') \\
\text{Transition is stored} &: \tau = \{s_a, r, \bar{s}', \text{done}\} \\
\textbf{with} &: \bar{s}' = \frac{1}{\phi} \sum_{p=0}^{\phi} s'_p
\end{aligned}
$$

Storing Transitions in Memory

Q-Learning:

$$O(\tau) = s \times a \times s' \times r \times \mathbb{R}$$
$$O(\tau) = k^2 \times \sigma^2 + 3$$

V-Learning:

$$O(\tau) = s_a \times a \times \bar{s}' \times r \times \mathbb{R}$$
$$O(\tau) = 2 \times \sigma^2 + 3$$

Current Status

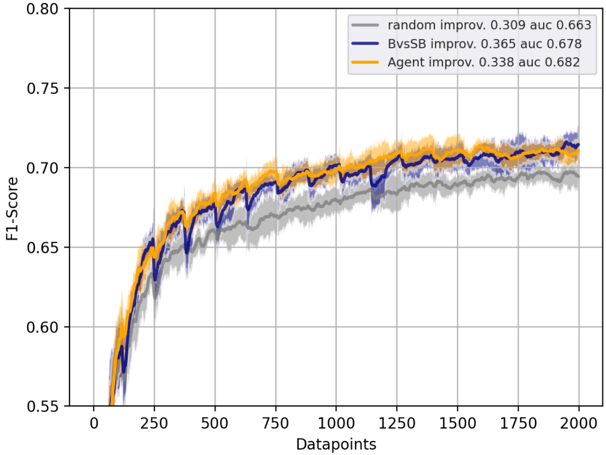
Agent Performance

V-Learning

CIFAR-10 (Custom Embedding)

Samplesize: 20

Interactions: 2.4M



Problems

- Extremely hard to train a sequence of 2000 interactions
- Seemingly sensible state spaces collapse the performance
- Agent is overly sensitive

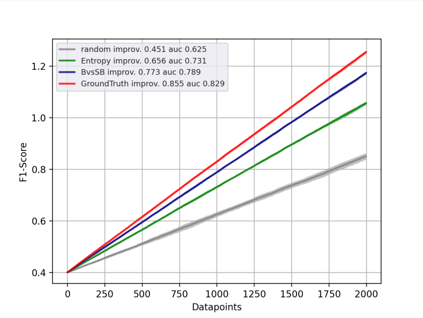
Mock Environment

To test different agents and perform a quasi-gridsearch in feasible time I created a mock environment

Internal state is a fictional F1-Score

Each datapoint has a random ”quality” value that governs its mock Entropy and BvsSB scores

A choosen sample add its ”quality” value to the internal F1-Score



Mock Environment (Formulas)

Internal state

$$\theta = 0 \qquad //\text{current F1 score}$$
$$\mu = \text{Uniform}(0, 0.9)^k \qquad //\text{current qualities}$$

State

$$\text{BvsSB} = \mu + \mathcal{N}(0, 0.2)$$
$$\text{entropy} = 2 + (\mu - 0.6) * 2 + \mathcal{N}(0, 1)$$
$$s = [\theta, \text{entropy}, \text{BvsSB}]$$

Step

$$\theta = \theta + \mu_a + \mathcal{N}(0, 0.5)$$
$$\mu = \text{Uniform}(0, 0.9)^k$$

Grid Search

Hand-picked combinations of ”Budget”, ”Interactions”, ”LR”, etc.

3-fold cross validation

Measuring regret with respect to BvsSB baseline

Preleminary results:

- N-Steps have seemingly no benefit
- Budget is more important than noise in the environment