# Application of Reinforcement Learning to Solve Classification Problems

by

Tissa Senevirathne

## Introduction:

Reinforcement Learning (RL) is very effectively applied in solving episodic tasks. Games or driving a car through a street are examples of episodic tasks. In this paper we present how RL can be applied to solve binary classification problems. The methods presented here can be easily extended for multi-class classification problems.

We demonstrate the model using Breast Cancer dataset from University of California at Irvine. Breast Cancer dataset is a classic binary classification dataset with 2 classes, malignant and benign. This can be reduced to binary classification by making classification as *Malignant* and *not Malignant*. Label = 1 representing *Malignant* case and Label =0 representing the *not Malignant* case.

This paper first presents how a classification problem can be converted to an episodic task. Then we discuss applicable Q-learning equations and how they are modified to task in hand. Then we discuss how DQN (Deep Q Networks) can be applied to this scenario. Finally results of applying RL for classification of Breast cancer dataset is presented and compared against results from classical binary classification using Neural Networks.

## Converting Classification to Episodic Task:

Converting the classification task to an episodic task is the most important and the first step in the process.

In classification problems we have set of features and corresponding label(s) that represent the class (classes) that the datapoint belongs.

In an episodic task we have a state, action and reward and next state. Classification task is a single step game, that means there is no next state.

Consider classification exercise as we are playing a slot machine with two buttons. One button representing malignant and other button represents benign. Assume there is a display in front of the slot machine that represent a set of numbers (I,e, our feature set). Assume that this number is an accurate representation of the label. The task of the player is to study the number and pattern and deduce the correct label. When the correct button is pressed the player get a positive reward “+r” and when wrong button is pressed the player get a negative reward -r. Idea here is to discourage the player from making wrong moves.

To map the features into a finite state space, we map each feature to a finite set of bins. Each bin is independent from the other. Features can be considered as representation of the current state, without loss of generality.

Reward = +r

Correct Action=a

Action={a|a1}-> reward=0

Observation=s

Reward = -r

Wrong Action=a1

Action={a|a1}-> reward=0

Fig: Representation of binary classification as an episodic task

Since this is a single step episodic task, next state is a terminal state. Which yield zero reward. Above diagram depicts this scenario. With this approach we have converted a single step episode into multistep episode as needed by markov decision process.

Next step is to represent the above scenario in S,A,R,S,A or as a Q table.

## Q-Learning Equation:

General representation of Q values are as follows

From above discussion, we have deduced St+1 is a terminal state. As a result, our equation reduces to the following as future rewards of terminal state are zero.

The above can be re-written as below to represent error in learning

The error of the equation is , the task of learning is to minimize the error.

With the above equation in place, we can easily create a neural network that takes St as an input and output Q values against different actions (in our cases labels). Action with maximum Q value represent the action we should take or in this case the correct label.

## Application of DQN Networks:

Core of the learning is Deep Q Learning Network, which learns the Q value per action based on the observation.

As with typical DQN, we take each sample and deduce an action. We use exploit vs exploration to choose between predicted action vs. random actions. Then we store the state (observation), action and rewards in to a replay buffer.

From the replay buffer we select a mini batch of random samples. The neural network is trained on this sample over several iterations.

The above steps are the same as typical DQN learning process.

The python code implementing the above can be found at GitHub at link.

## Results

We use 4 Layer Densely connected Neural network with Relu as the activation function for intermediate layers. Final layer has 2 outputs (each representing each of the labels) and it has no activation. We use Adam optimizer and minimize mean squared error.

This leads to a training accuracy of 100% and validation accuracy of 97%

To compare the effectiveness of the approach we performed binary classification with a similar neural network with final output layer having sigmoid activation function and RMSprop as the optimizer and binary\_cross\_entropy as the loss function.

Classical classification provided training accuracy of 98% and validation accuracy of 95%

Training and Test size distributions were the same for both cases and the same data set was used. However, due to random nature of the train and test split , there is no guarantee exact sample set was used for both scenarios.

## Conclusions:

It proves that a classification task can be converted into an episodic task without loss of generality. Also, it proves that Reinforcement Learning algorithm provides slightly better accuracy over binary classification. However, further experimentation with larger and varying datasets are warranted. We suggest the work in this paper to be extended to multi-label classification as well as different datasets.