Reinforcement learning using MNIST dataset

Manpreet Singh
Masters of Applied Computer Science
B00853930
mn407919@dal.ca

Abstract

Reinforcement learning(RL) is an area of machine learning concerned with how software agents ought to take actions in an environment to maximize the notion of cumulative reward. In this paper, I have discussed implementing and evaluating a reinforcement learner that takes MNIST handwritten digits as input and goes either up or down by calculating Q-values and policy values, then it chooses the best action for that input. It is a model-free RL task as a reward in this environment is r=1 for state 0, and r=2 for state 9. There is no reward given in the states between, and the rewards are not provided to the learner in advance. In this task two models were created, first RL model which uses a pre-trained CNN Model to predict MNIST images and calculate Q-values. The second RL model does not use the pretrained Model to predict images and images are directly passed to the model to predict Q-values. I tested the performance of the RL network with frozen and unfrozen layers of the pre-trained network. Then I tested the performance of the RL network without using the pre-trained network. The results of both models are discussed and compared in this paper.

1.Intoduction

Reinforcement learning (RL) algorithm enables an agent to learn an optimal behaviour when letting it interact with some unknown environment and learn from its obtained rewards. The RL agent uses a policy to control its behaviour, where the policy is a mapping from obtained inputs to actions. The aim of this project is to build an RL Network on MNIST dataset with and without using a pretrained base CNN Model for predicting images on MNIST data. First I created a CNN Model which was trained to classify the MNIST handwritten digit images. Then I created RL Model and used pretrained CNN Model as base model to predict the images and then calculated Q-values and policy values which give the best action for each state. Second RL model was built from scratch without using any pretrained base model. All the layers of this model were trained during reinforcement learning. Later, I tested both the models on test dataset, to compare the predicted Q-values and policy values.

2. Dataset

The MNIST is a well-known dataset that is commonly used for training various image processing systems. It contains images with 28 X 28 pixels in gray scale, each coped with a label from 0 to 9, indicating the corresponding digit. Dataset contains 70,000 labelled images which is divided into training set of 60,000 images and 10,000 test images. I have picked 60,000 images for training CNN Model and 1000 images for training RL Model. I picked 400 images for testing RL Model with pretrained CNN Model and without using pretrained Model.I have picked less number of images as the training was very low and required more powerful GPU on such a large dataset

3. Baseline Model

To predict the MNIST data images, first a Convolutional Neural Network Model with five layers was created. For the CNN Model, first layer is a single convolutional layer with a small filter size (3,3) and a modest number of filters (32) followed by a max pooling layer. The filter maps are flattened to provide features to the classifier. As the output layer should be of 10 nodes to predict the number between 0 and 9 which requires the use of a softmax activation function. Between the feature extractor and the output layer, a dense layer of 100 nodes was added to interpret the features, which uses ReLU activation function. Then a conservative configuration for the stochastic gradient descent optimizer with a learning rate of 0.01 and a momentum of 0.9 was used. The accuracy of baseline model is 99 percent to predict MNIST dataset images.

4.RL Model using Pretrained CNN Model

After creating the CNN model, I used this model to predict MNIST dataset images. The output from the base model is one hot encoding of the image which is fed into the sixth layer of 20 nodes with ReLU activation function. The last layer is the output layer which has two nodes and uses linear activation. This layer gives the Q values for the input image. The RL model is run for 1000 trails where each trail is an image from the training set. Then we check if the image is the terminal state (0 or 9). If the image belongs to a terminal state, the loop is broken, and the next image is picked. If the image does not belong to terminal state, then using the RL model Q values are calculated and the next action is computed. Then the image and action are passed to the transition function which returns the next image. If this state is terminal then rewards are computed, else based on action new image is picked. This way the model is trained. So once the model is trained policy is calculated for test data based on Q values. Two experiments were done for RL Model using the pretrained model:
(i) all layers of the pretrained model were frozen
(ii) first three layers were frozen and last two were unfrozen.

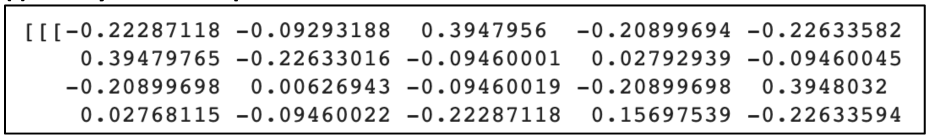
5. RL Model without using Pretrained CNN Model

In this model, the input an MNIST dataset image is fed into the model without pre training. First layer of this model is a single convolutional layer with a small filter size (3,3) and a modest number of filters (32) followed by a max pooling layer. The filter maps are flattened to provide features to the classifier. Next layer consists of 10 nodes to represent the number between 0 and 9 which requires the use of a softmax activation function. Between the feature extractor and the softmax layer, a dense layer of 100 nodes was added to interpret the features, which uses RelU activation function. Then a conservative configuration for the stochastic gradient descent optimizer with a learning rate of 0.01 and a momentum of 0.9 was used. The sixth layer consists of 20 nodes with ReLU activation function. The last layer is the output layer which has two nodes and uses linear activation. This layer gives the Q values for the input image. The RL model is run for 1000 trials where each trial is an image from the training set. Then we check if the image is the terminal state (0 or 9). If the image belongs to a terminal state, the loop is broken, and the next image is picked. If the image does not belong to terminal state, then using the RL model Q values are calculated and the next action is computed. Then the image and action are passed to the transition function which returns the next image. If this state is terminal then rewards are computed, else based on action new image is picked. This way the model is trained. So once the model is trained policy is calculated for test data based on Q values.

6. Results

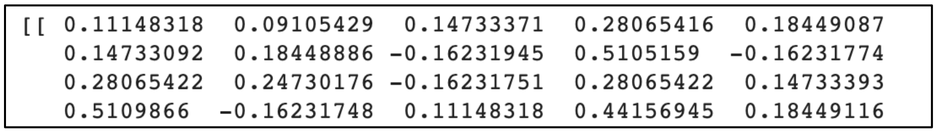
RL Model using Pretrained CNN Model - Q1 and Q2 values are different for different images(states). The policy which is predicted contains both actions (1,-1) i.e is to move up or down based on Q-values.

(i) All layers of the pretrained model were frozen –



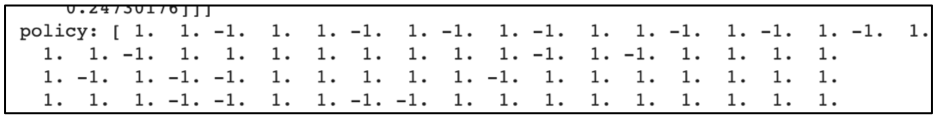
```
[[[-0.22287118 -0.09293188  0.3947956  -0.20899694 -0.22633582
    0.39479765 -0.22633016 -0.09460001  0.02792939 -0.09460045
   -0.20899698  0.00626943 -0.09460019 -0.20899698  0.3948032
    0.02768115 -0.09460022 -0.22287118  0.15697539 -0.22633594
```

Figure 1 Q1- values when all layers are frozen



```
[[ 0.11148318  0.09105429  0.14733371  0.28065416  0.18449087
   0.14733092  0.18448886 -0.16231945  0.5105159  -0.16231774
   0.28065422  0.24730176 -0.16231751  0.28065422  0.14733393
   0.5109866  -0.16231748  0.11148318  0.44156945  0.18449116
```

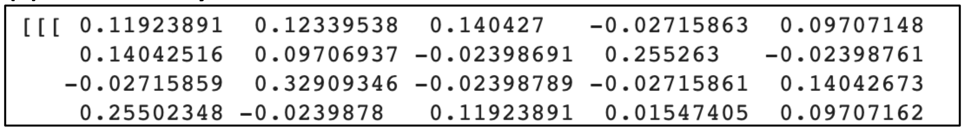
Figure 2 Q2-values when all layers are frozen



```
policy: [ 1.  1. -1.  1.  1. -1.  1. -1.  1. -1.  1.  1. -1.  1. -1.  1. -1.  1.
  1.  1. -1.  1.  1.  1.  1.  1.  1.  1.  1. -1.  1. -1.  1.  1.  1.  1.
  1. -1.  1. -1. -1.  1.  1.  1.  1.  1. -1.  1.  1.  1.  1.  1.  1.  1.
  1.  1.  1. -1. -1.  1.  1. -1. -1.  1.  1.  1.  1.  1.  1.  1.  1.  1.
```

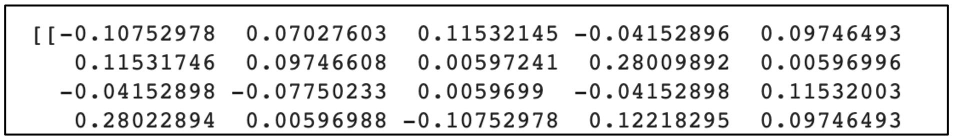
Figure 3 Policy values when all layers are frozen

(ii) first three layers were frozen and last two were unfrozen.



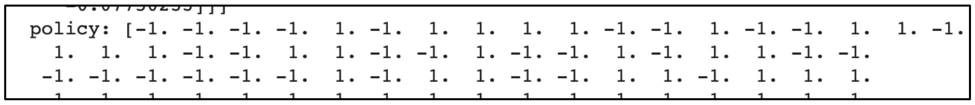
```
[[[ 0.11923891  0.12339538  0.140427   -0.02715863  0.09707148
    0.14042516  0.09706937 -0.02398691  0.255263   -0.02398761
   -0.02715859  0.32909346 -0.02398789 -0.02715861  0.14042673
    0.25502348 -0.0239878   0.11923891  0.01547405  0.09707162
```

Figure 4 Q1- values when three layers are frozen and two are unfrozen



```
[[-0.10752978  0.07027603  0.11532145 -0.04152896  0.09746493
   0.11531746  0.09746608  0.00597241  0.28009892  0.00596996
  -0.04152898 -0.07750233  0.0059699  -0.04152898  0.11532003
   0.28022894  0.00596988 -0.10752978  0.12218295  0.09746493
```

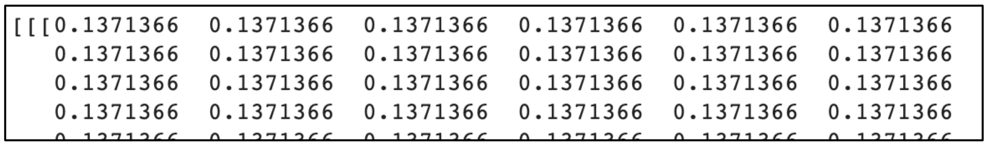
Figure 5 Q2- values when three layers are frozen and two are unfrozen



```
policy: [-1. -1. -1. -1.  1. -1.  1.  1.  1.  1. -1. -1.  1. -1. -1.  1.  1. -1.
  1.  1.  1. -1. -1.  1.  1. -1. -1.  1. -1. -1.  1. -1.  1.  1. -1. -1.
 -1. -1. -1. -1. -1. -1.  1. -1.  1.  1. -1. -1.  1.  1. -1.  1.  1.  1.
```

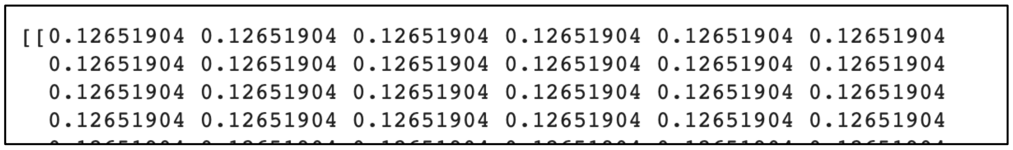
Figure 6 Policy values when three layers are frozen and two are unfrozen

RL Model using Pretrained CNN Model - Q1 and Q2 values are same. The policy which is predicted has only -1 value. This is because we are not using pretraining model to predict images so the model predicts each image wrong and as same image. As in the below result when I checked the images it predicted all images as 6.



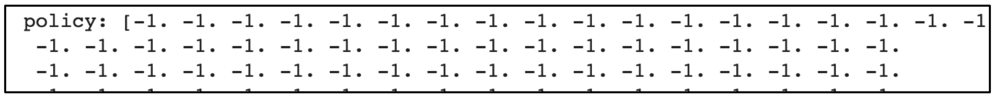
```
[[[0.1371366  0.1371366  0.1371366  0.1371366  0.1371366  0.1371366
   0.1371366  0.1371366  0.1371366  0.1371366  0.1371366  0.1371366
   0.1371366  0.1371366  0.1371366  0.1371366  0.1371366  0.1371366
   0.1371366  0.1371366  0.1371366  0.1371366  0.1371366  0.1371366
```

Figure 7 Q1- values when pretrained model is not used



```
[[0.12651904 0.12651904 0.12651904 0.12651904 0.12651904 0.12651904
  0.12651904 0.12651904 0.12651904 0.12651904 0.12651904 0.12651904
  0.12651904 0.12651904 0.12651904 0.12651904 0.12651904 0.12651904
  0.12651904 0.12651904 0.12651904 0.12651904 0.12651904 0.12651904
```

Figure 8 Q2- values when pretrained model is not used



```
policy: [-1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1
 -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1.
 -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1.
```

Figure 9 Policy- values when pretrained model is not used

6. Conclusion

From the above experiments it can be concluded that RL Model using a pretrained model to predict MNIST dataset images gives better results as when we don't use pretrained model to predict images and feed into networks it predicts the images wrong and gives same Q values for all images.

7. References

[1] [Online]. Available: https://machinelearningmastery.com/how-to-develop-a-convolutional-neural-network-from-scratch-for-mnist-handwritten-digit-classification/. [Accessed 4 12 2020].

[2] G. M. B. Abdul Mueed Hafiz* 1, "Reinforcement Learning Based Handwritten Digit Recognition with Two-State Q-Learning," 2020.

[3] Wiering MA, Hasselt Hv, Pietersma A-D, Schomaker L Reinforcement Learning Algorithms for solving Classification Problems.

[4] T. P. Trappenberg, Fundamentals of machine learning. Oxford, United Kingdom: Oxford University Press, 2020.