InClass Prediction Competition

COL-774, Spring-2019

Assignment of Machine Learning course (COL-774) at IIT Delhi, Spring 2019.

14 teams · 4 days to go

Overview

This is Assignment 4 for the Machine Learning course (COL-774) at IIT Delhi (Semester II, 2018-19). The goal of this assignment is to make you familiar with various machine learning algorithms on a cutting edge research problem.

There are two parts of this question: Part 1) Where you have to implement a set of fixed learning algorithms and Part 2) which is a competition where you have to try out algorithms of your choice - independent of whether they have been covered in the class or not. This assignment should provide an opportunity for you to learn new techniques/tools on your own as well as try the ones that you have already seen in the class.

Problem Description:

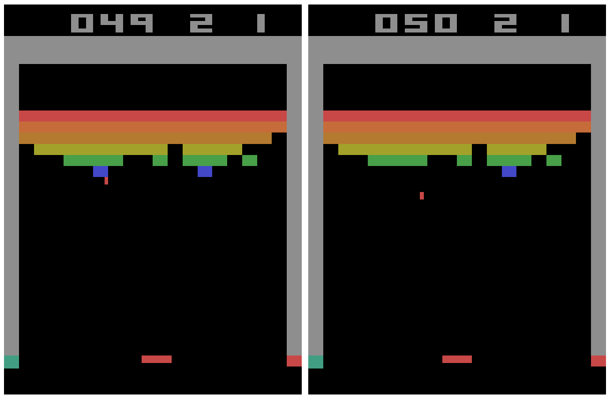
We will first start with some terminology,

1. **Game of Breakout:** You must have played the game of Breakout in your childhood in which you have layers of bricks at the top and there is a paddle at the bottom that you control. A ball bounces between the bricks and the paddle. Each time the ball hits a brick you get a positive score and goal of the player is to score as many points as possible by breaking the bricks (hence the name "Breakout"). See this video for a quick visualization of the game.
2. **Episode:** An episode consists of a sequence of frames of a gameplay. An episode ends when your paddle misses the ball.
3. **Reward:** Each time we move from one frame to another, we receive a reward corresponding to the transition. There are two types of rewards: +1 reward: You hit a brick, 0 reward: When you do not hit a brick

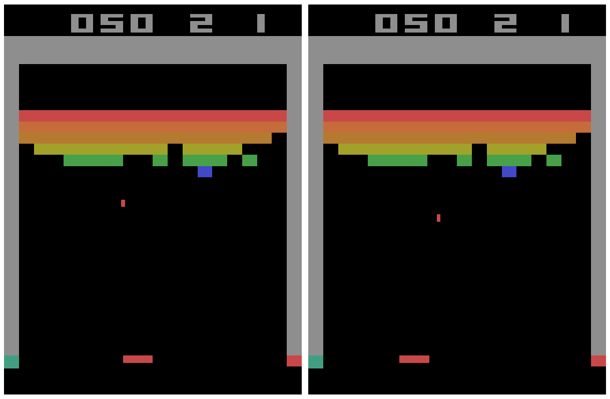
hit a brick. 0 reward: When you do not hit a brick.

Note that this is different from the score that you accumulate which is displayed on the top of the game. For each hit, you may get a different score (depending on the type of brick), but in this question, we are only interested in predicting whether the ball is going to hit a brick or not.

For example, you will get +1 reward in below transition:



For example, you will get 0 reward in below transition:



4. **Dataset:** You are given a dataset consisting of a set of 500 episodes, such that, each episode consists of a sequence of frames and with each frame transition there is an associated reward. Note that, because each reward corresponds to a transition of one frame to another so the number of rewards will be one less than the number of frames in the sequence. See Data tab for details on the format of dataset.
5. **The task:** You are playing this game on a very old television that sometimes does not display some frame. To be precise, in every 7 frames it randomly misses (exactly) 2 frames. Because of this, you are facing difficulty in playing the game. You think that if somehow you can predict the future reward from these noisy sequence of frames then you will be able to play better. Being a student of Machine Learning you decide to train an agent on above dataset, such that, at time step t, when given a subset of 5 out of past 7 frames, it predicts the reward at time t+3. Here, the reward at time step t refers to the reward obtained when transiting from t-1 to t.
6. **At test time:** Let $f_{t-6}, f_{t-5}, \ldots, f_t$ represent 7 frames at time step t and let $r_{t-6}, r_{t-5}, \ldots, r_t$ denote corresponding rewards. Then, at testing time you will

be given any 4 random frames (still in sequence) out of

$f_{t-6}, f_{t-5}, f_{t-4}, f_{t-3}, f_{t-2}, f_{t-1}$ and frame f_t will always be given to you. Your task will be to predict the reward at time step t+1. For example, in above case we may drop frame f_{t-5} and f_{t-3} then your task will be to predict the value of r_{t+1} given $f_{t-6}, f_{t-4}, f_{t-2}, f_{t-1}, f_t$. Assume that each such test frame sequence is randomly sampled from a set of episodes, where the episodes are generated by playing the game and have the same distribution as the train set of episodes.

7. **Evaluation Metric:** As the number of 0 rewards is more than the number of +1 rewards so predicting all 0s will give very high prediction accuracy but in reality it is a very bad model. Hence we will use f-measure as our evaluation criteria. You can read about f-measure here (the metric is referred to as F1 score).

Assignment Details:

NOTE: For each of the parts below, you need to carefully think about what your input should be. You are given full episodes for training, but of course an entire episode can not be used as a single training example. Our task here is to predict reward at a given time step. How much prior frame information should be used? How much of it is available at test time? The best way to get a good generalisation accuracy is to have the train distribution match the test distribution. You also need to worry about noise that is present in the test samples. Use all these cues to determine (and possibly generate) the right training set for your problem. If you are using more than one frame for training, a standard way to give them as input is to stack them up in RGB channel. For example, stacking two frames of size $3x210x160$ will give an array of size $6x210x160$

Part 1: Fixed Algorithms (non-competitive) [50 points]. Deadline: <23 April 2019, 11:50 pm> (Buffer days allowed for this part only)

(A) PCA + SVM : [10 points] [Scikit-learn]: Apply Principal Component Analysis (PCA) on the given datasets (for images in first 50 episodes) using scikit-learn library and use the top 50 dimensions obtained using PCA. Next, use LIBSVM to learn a linear as well as Gaussian SVM classifier using the projected set of attributes. Use internal cross-validation to determine the C (and gamma) parameters. What are the train and test accuracies obtained? Comment on your observations. Here is a short introduction to PCA. Also check out the notes on the course website.

(B) Convolutional Neural Network 1 (CNN): [15 points] [Pytorch/TensorFlow/Keras] In this part, you will implement a convolutional neural network architecture. Preferably, you should use Pytorch as the language of implementation. Here is a short tutorial on how to use Pytorch. If you have familiarity with any other deep learning frameworks such as TensorFlow or Keras (or others), you can use these, but check with us first. Note that this part should

NOT be done using Scikit library since we would like you to work with a deep learning framework which will also be useful for next parts of the assignment.

Use the below architecture to train your model.

> {1 CNN layer with 32 kernels with filter size of (3,3) and stride of 2 > followed by MAX pooling (of filter size (2,2) and stride = 2)} > followed by {1 CNN layer with 64 kernels with filter size of (3,3) and > stride of 2 followed by MAX pooling (of filter size (2,2) and stride = > 2)} followed by {1 fully connected layer with 2048 units} followed by > {output layer}. For this part, use ReLU as the activation function in > the hidden layer. Use negative of the log-likelihood as your loss > function.

Next, vary the set of parameters (e.g., number of the kernels, units in the fully connected layer etc.) using internal cross-validation. Report the parameters for the best architecture obtained using your experimentation. Report your training and test accuracies for this architecture. Report any additional observations that you might have.

Note: We would like you stick to basic pre-processing including converting to grayscale. Resizing/fixed length cropping may also be ok (provided it does not result in loss of accuracy) but please mention this clearly In your report and the reasons for using the same. This is only for fixed parts. For competition, you can use whatever pre-processing you like (including complex feature extraction).

PART 2: Competition [50 points]. Deadline: <25 April 2019, 11:50 pm> (No buffer days allowed)

Try out algorithms of your choice to maximize the accuracy on the test data. Improve your model as much as you can. You are strongly encouraged to try out variations of the algorithms (deep learning or otherwise) described in the class as well as try new algorithms/models by looking up online resources. You are also free to use any standard publicly available libraries of your choice for this part. If you are using some software developed by others for special purposes, you should check with us first.

You will submit the labels for the test dataset on Kaggle and your performance (on classification accuracy) will be ranked on the leader-board. There are 2 leader-boards; a Public leader-board on which you can monitor your accuracy on 40% of the test dataset; a Private leader-board (which you cannot see) which will be used to determine the final rankings on the other 60% of the test data.

Note: The score obtained in this part will solely depend on the quality of your predictions on the test labels. You must submit the code for your best performing model (as well as include all the details in your report) so that we can replicate the results if required (see below).

Submission Guidelines:

Code: You need to submit separate code for each part on moodle. The deadline

Code: You need to submit separate code for each part on moodle. The deadline for each part is as mentioned above, i.e. Part 1 is due <> Part 1: You need to submit all your code that you implemented on moodle. Part 2: You should submit your best performing code (the one used to obtain your final test set labels). Make sure to include the details of any external libraries used in your report.

Report: <26 April 2019, 11:50 pm> You need to submit a single report for both the parts (details below). You can submit your report for both parts by the deadline above. No use of buffer days is allowed beyond this point.

Fixed Algorithms part: Your report should include details of your experimentation, observations, any graphs as well as accuracy as required by the question (similar to what you have done for earlier assignments). Competition Part: Your report should include all the details of your best performing model (including libraries used) so that we can replicate the results if required.

Demos: Please make sure that (a) the libraries are installed on your laptop/hpc, so that we can ask you to train them during the demo if needed; (b) have separate scripts for training the model and evaluating the model on the test sets so that we test each part independently.

Kernels

There are no kernels yet.

Be the first

0 discussion topics

There are no topics yet.

Start one

Launch
a day ago

Close
4 days

14	14	68
Teams	Competitors	Entries

Points This competition does not award standard ranking points

Tiers This competition does not count towards tiers

This is a Kaggle InClass competition provided free to academics.
Find out about hosting your own InClass competition »

Tags f_{beta} medium