# Wet Exercise 1

For any question, use the course forum to ask questions.

# Introduction

In this exercise we will implement several neural architectures on two simple environments with two basic approaches: Deep Q-Networks (DQN) and Policy Gradient Method (PG). The goal of the exercise to introduce the current state of the art frameworks and methods for solving typical problems in RL.

This exercise can be submitted in pairs.

# The Environments

Luckily, today we have many RL environments to test our algorithms on. By far, the most popular framework that provides good environments for testing new algorithms is Gym by OpenAI. For the sake of simplicity, we will focus on the "Taxi" Domain for applying DQN and Acrobot for applying PG.

# Setup Your Computation Environment

## Step 1 - Install Python

The exercise is in Python. Install python on your machine. Note, it is better to do the exercise on Mac or Linux Environments, but Windows can work as well. We will use Python 3.6 or above.

There are several ways to install Python. From my experience, it is best to use Conda or Anaconda packages. They come with many useful functionalities.

## Step 2 - Install IDE

I recommend to use PyCharm. There is a free version called "community edition". You do not need to pay for that. The exercise can be also be solved using Jupyter notebooks. It comes by default with Anaconda package.

## Step 3 - Install Basic Gym

Installing basic Gym.

## Step 4 - Install PyTorch

The framework that we will use as a NN solver in this exercise is PyTorch. Install it as well. Follow the these tutorial of Pytorch in order to understand how to use it:

1. Using PyTorch from examples.
2. Solving RL problem with PyTorch.

# The Tasks

## (1) Solving the Taxi environment.

Please answer all the questions in the following task requirements.

1. Read the description of the taxi environment in here.
2. Build a fully connected network according to the following specifications:
    - It has a two layers: input layer and output layer where both of them are fully connected.
    - The state space will be encoded as one-hot.
    - The architecture should follow the **DQN Architecture**. What are the state space and action space of this environment?
    - Note, the size of the middle neuron layer is not specified. You should try and test to see what are the best dimensions for the internal layer in this problem.
    - What is the trade-off when choosing the inner layer size?
    - In a one-hot state encoding as an input, what is the dimension of the input?
    - The input encoding should be fully one-hot encoding.
3. Train the network to act optimally. Describe the optimizer, and other interesting parameters that you chose in your training of the network. In this section, do not use any special regularizations.
    - Plot the accumulated reward in an episode vs. no. of episode.
    - You may smooth the results using a sliding window. Plot the smoothed result on the same graph.
    - Save to disk the optimal network and write a script to load the network and show the demonstrate the results where the learning is halted
    - Average the results over 10 runs, and compute the variance of the learning..
4. Repeat the training with
    - dropout on both layers

- L1 regularization on both layers
  Analyze the results and support your arguments with graphs.

5. Suggest **another input encoding**. Compare your suggested encoding with one-hot-encoding w.r.t. the training time, complexity, accuracy, memory imprint, etc. Provide graphs to prove your arguments.
6. **Learning curves and a comparison between different optimizers.** In this section, choose 3 optimizers from here. Give a brief explanation of each of them in text, and compare them empirically. Provide a detailed explanation what is best to use in the case of the Taxi problem.
7. **Solving this task using Policy Gradient (PG)**.
   - Choose a PG method as you like (Vanilla PG, PPO, TRPO, etc.). Suggest a two layers architecture and solve the problem accordingly.
   - Train the network and show the training graph as in previous sections.
   - Again, save the optimal network and provide a script to load the network and demonstrate the network behavior.
   - In your report describe your architecture, how did you solve it, how did you avoid the algorithm from getting stuck, and what influenced your algorithm convergence speed.

## (2) Solving a Control Task from Visual Input

1. In this section we will solve the control problem Acrobot.
2. Read the tutorial of PyTorch and RL, Note specifically, how to use the rendering visual output and not the default observation that Gym provides.
3. Suggest a convolutional NN that solves the Acrobot task. When suggesting an architecture, describe your considerations when choosing the layers, their sizes, initialization, etc.
4. What preprocessing did you used? Describe why you've chosen these preprocessing and how it helped to solve the task.
5. Have you used any regularization/robustness methods? If so, describe why have you chosen them to be that way.
6. Train the network. Similar to previous sections, plot a learning curve together with smoothed version.
7. Save the trained network and provide a script to load the network and demonstrate the optimal policy.
8. In addition to the optimal policy demonstration, generate a 10 seconds movie that demonstrate the optimal policy.

# What to Submit?

1. Push the code into a git repository in github. When we'll check your code we will clone your github code to our local computer and run it as if it was written on our computer.

2. Put each of the tasks above in a different python package (each instance of run).
3. Also, push the trained networks into the repository.
4. For each trained network, provide a script to load the trained network and running it alive on our computer.
5. Pay attention for a good documentation of the code.
6. Add a readme.md file that explains what are in the files that you provide..
7. The final report (one file) should be a pdf document with all the graphs, explanations, that were specified above. Even if not said explicitly, we would like to hear your insights, analyses, and seeing that you understand what you are doing, so don't save on them!

**GOOD LUCK!!!**