Quick Start Guide To Programming Gym

You are free to investigate more about the gym, but for the purposes of this project you don't need to know too much about the architecture etc. You will have to understand the syntax of how to generate q-values etc. Unfortunately, the gym overview articles (documentation) seem to have been deleted from their website. We are in the process of creating our own and will put them here. We will go over some details to get you started during the section on Monday.
You can check out this website for more information about the gym.

A basic Python, Numpy, Pytorch introduction

Python data types:

Please check out the link.
'Dictionary' is a {key: value ...} structure, given the 'key', you can access the value by dict[key].

```
# enumerate the keys of the dictionary
for i, k in enumerate(example_dict):
    print(i, k, example_dict[k])

# enumerate through both keys and values this is the way
for i, (k, v) in enumerate(example_dict.items()):
    print(i, k, v)
```

One useful feature for debugging is string formattin.

```
>>> print('We are the {} who say "{}!"'.format('knights', 'Ni'))
We are the knights who say "Ni!"
```

Numpy structures:

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.
You can read the link to get familiar with basic numpy operations.

The feature that makes numpy so powerful is **broadcasting.** The term broadcasting describes how numpy treats arrays with different shapes during arithmetic operations. Here is a brief introduction to it.

Sometimes you need to add a newaxis to a numpy array or squeeze the array. You can try:

```
# example is an existing array
# add a new axis
example[:, np.newaxis]
```

```
# resize the array
example.resize((2, 1))
# squeeze the array
np.squeeze(example)
```

Pytorch tensor types:
A torch.Tensor is a multi-dimensional matrix containing elements of a single data type.
The tensor types in pytorch are defined here. Note that the data type of tensors in pytorch could not be the same as the data type of numpy arrays.

Pytorch numpy bridge:

```
# tensor to numpy
a = torch.ones(5)
b = a.numpy()
# numpy to tensor
a = np.ones(5)
b = torch.from_numpy(a)
```

Cuda semantics:
To use high performance Gpus, please check out the documents.
Remember, a tensor on cpu is not the same type as a tensor on gpu.
When training or testing a model, you have to move both the model and the data to the same gpu device.

```
model = NeuralNetwork().to(device)
inputs, labels = data[0].to(device), data[1].to(device)
```

You can move it back to cpu by

```
model.cpu()
```

Saving & Loading models:
One recommended way to save & load models is

```
torch.save(model.state_dict(), PATH)

model = TheModelClass(*args, **kwargs)
model.load_state_dict(torch.load(PATH))
```

When training or testing (evaluating) the model, you have to set the model into training/evaluating modes respectively.

```
model.train()
or
model.eval()
```

Before testing, remember to set the model into “no gradient mode”

```
with torch.no_grad():
    for data in testloader:
        ……
```

You can also remove some tensor from the gradient computation graph by ‘detach’ which stops the gradients from flowing through the tensor.

Finally, here is a simple deep-q-learning tutorial with pytorch.

Troubleshooting with graphical displays:

Normally you will SSH to the machine but the gym environment will give you an error because it can’t write to the display.

The easiest solution is to install an X-server on your machine and then ssh -x to the machine. On my macbook Air M1 I installed xQuartz and then ssh -x and it worked fine.

```
# on the top of dqn.py
import matplotlib
matplotlib.use('Agg')
```