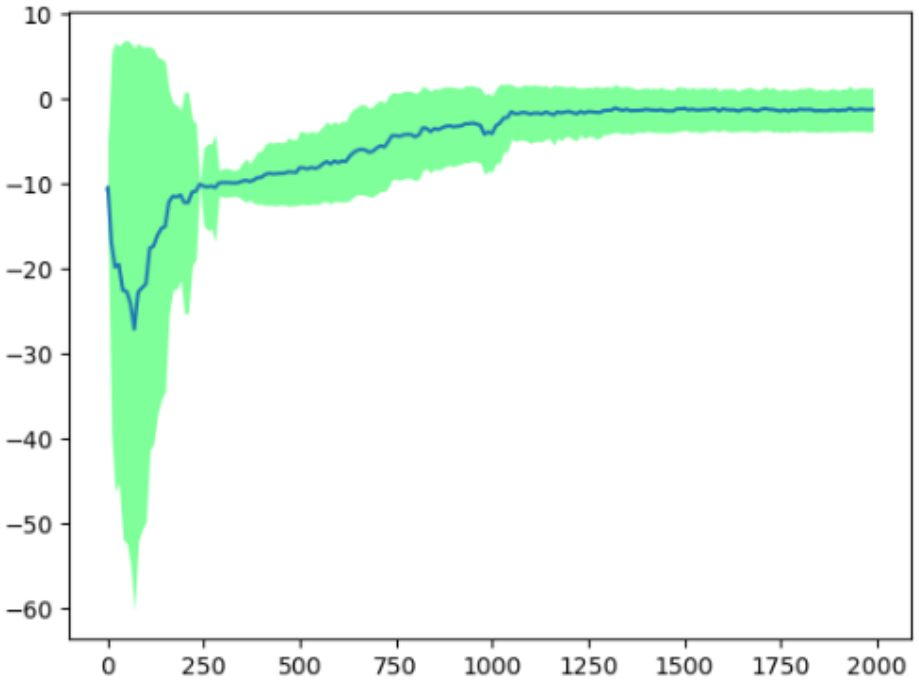
1.

It gets stable after around 1000 episodes.



2.

The network is trained with one hot encoded state action pairs. Given a state, the output is probability of all the actions where argmax gives the best action.

```
model = tf.keras.models.Sequential([
    tf.keras.layers.Input(shape=(num_states)),
    tf.keras.layers.Dense(64, activation="relu"),
    tf.keras.layers.Dense(64, activation="relu"),
    tf.keras.layers.Dense(num_actions, activation='softmax')
])

model.compile(optimizer='adam',
              loss=tf.keras.losses.categorical_crossentropy)

model.summary()
```

```
Model: "sequential_13"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_40 (Dense)            (None, 64)                32064

 dense_41 (Dense)            (None, 64)                4160

 dense_42 (Dense)            (None, 6)                 390

=================================================================
Total params: 36,614
Trainable params: 36,614
Non-trainable params: 0
_________________________________________________________________
```

Results:

In [198]:
```
e = 100
q = False
print("Average reward with NN", e ,"episodes =" ,eval_nn(q, e))
q = True
print("Average reward with qtable", e ,"episodes =" ,eval_nn(q, e))
```

```
Average reward with NN 100 episodes = 7.52
Average reward with qtable 100 episodes = 8.26
```