Applying Q-learning to Gridworld

In this lab, you will get hands-on experience with Reinforcement Learning in a classic problem domain, Gridworld. The sample project is implemented in Java, and consists of 4 class (.java) files:

1. **ExpRunner.java** – this class contains the main method and is used to initialise experiment instances and for data logging.
2. **Utilities.java** – this class contains static helper methods for features such as mixed radix conversion and formatting of experimental results.
3. **Environment.java** – this is a sample implementation of a Gridworld environment.
4. **Agent.java** – this is a sample implementation of a Q-learning agent using a tabular Q value store and epsilon-greedy action selection.

Setup instructions:

1. Download and install the latest Java Development Kit (JDK) for your machine (e.g. Windows x64)
2. Download and install the Eclipse Java IDE (the sample code is set up with an Eclipse project, use a different Java IDE if you prefer)
3. Start Eclipse and create a workspace (folder to store your projects)
4. In Eclipse, click on File > Import… and type "Existing" into the search bar, then select "Existing Projects into Workspace" and click Next
5. Select the Gridworld_Qlearning.zip archive, and click finish. The project should appear in the workspace
6. Build path issues (if they appear) may be resolved by right clicking on the project name in the Package Explorer, clicking on properties, then on the libraries tab, and clicking on Add Library… > JRE System Library > Workspace default

Exercise:

The application is set up to conduct multiple statistical runs of a Gridworld experiment with a single call to the `main()` method. The number of runs conducted in any run of the application is controlled by the `numRuns` variable at the top of ExpRunner.java. When run, the application will output experimental results in a timestamped folder (located in the output directory of the Eclipse project). The outputs files contain the final Q values table (QTables.txt) at the end of each statistical run, along with the number of steps to goal for each run (steptToGoal.csv). The stepsToGoal.csv file may be opened or imported and edited in a spreadsheet application (e.g. Microsoft Excel, OpenOffice Calc, etc.).

In this exercise you will run the experiments a number of times while varying the environment and agent parameters, and plot a graph of the results that you obtain using a spreadsheet application (e.g. Microsoft Excel). You will plot **Episode number** on the horizontal axis and the **average number of steps to reach the goal** on the vertical axis.

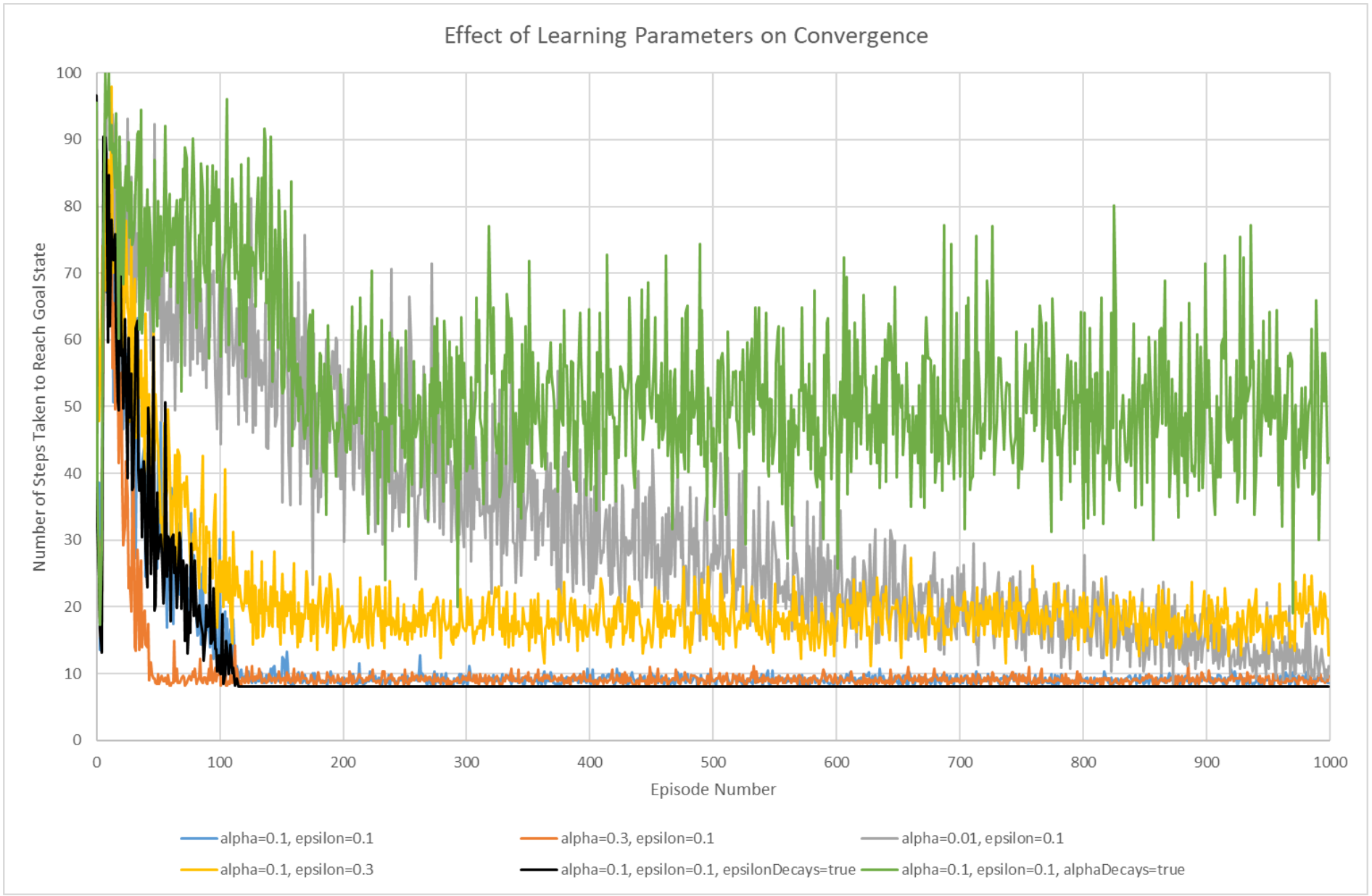
First create a new spreadsheet, and for each combination of settings below, conduct 10 statistical runs using the code sample provided, copy the results into a new worksheet in your spreadsheet and plot the average of the 10 runs on your results graph. You can use Excel's built in import from .csv functionality to import the results for each parameter setting.

A sample graph is provided overleaf (note that you can achieve smoother graphs by increasing the number of statistical runs conducted).

For each of the settings below, write a short note into your spreadsheet explaining the performance observed.

Settings to plot (assume that alphaDecays and epsilonDecays are set to false unless otherwise stated):

1. Default settings provided, alpha = 0.1, epsilon=0.1
2. alpha=0.3, epsilon=0.1
3. alpha=0.01, epsilon=0.1
4. alpha = 0.1, epsilon=0.3
5. alpha = 0.1, epsilon=0.1, epsilonDecays=true
6. alpha = 0.1, epsilon=0.1, alphaDecays=true


Effect of Learning Parameters on Convergence
Number of Steps Taken to Reach Goal State
Episode Number
0
10
20
30
40
50
60
70
80
90
100
0
100
200
300
400
500
600
700
800
900
1000
alpha=0.1, epsilon=0.1
alpha=0.3, epsilon=0.1
alpha=0.01, epsilon=0.1
alpha=0.1, epsilon=0.3
alpha=0.1, epsilon=0.1, epsilonDecays=true
alpha=0.1, epsilon=0.1, alphaDecays=true