(c) Next, implement the EM algorithm for Gaussian mixtures. Write **three** functions: `log-likelihood`, `gm_e_step`, and `gm_m_step` as given in the lecture. Identify the correct arguments, and the order to run them. Initialize the algorithm with means as in (1.1), covariances with $\hat{\mathbf{\Sigma}}_1 = \hat{\mathbf{\Sigma}}_2 = I$, and $\hat{\pi}_1 = \hat{\pi}_2$.
Run the algorithm until convergence and show the resulting cluster assignments on a scatter plot either using different color codes or shape or both. Also plot the log-likelihood vs. the number of iterations. Report your misclassification error.

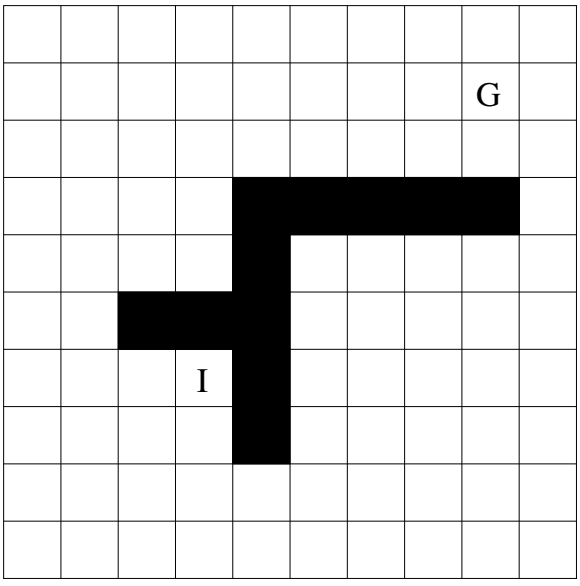
(d) Comment on the results:

(a) Compare the performance of k-Means and EM based on the resulting cluster assignments.

(b) Compare the performance of k-Means and EM based on their convergence rate. What is the bottleneck for which method?

(c) Experiment with 5 different data realizations (generate new data), run your algorithms, and summarize your findings. Does the algorithm performance depend on different realizations of data?

**2. Reinforcement Learning, 65 pts.** In this portion of the assignment, you will write software to implement a simulated environment and build a reinforcement learning agent that discovers the optimal (shortest) path to a goal. The agent's environment will look like:




Each cell is a state in the environment. The cell labeled "I" is the initial state of the agent. The cell labeled "G" is the goal state of the agent. The black cells are barriers—states that are inaccessible to the agent. At each time step, the agent may move one cell to the left, right, up, or down. The environment does not wraparound. Thus, if the agent is in the lower left corner and tries to move down, it will remain in the same cell. Likewise, if the agent is in the initial state and tries to move up (into a barrier), it will remain in the same cell.

2.1. *Implementation (20 points).* You should implement a Q learning algorithm that selects moves for the agent. The algorithm should perform exploration by choosing the action with the maximum Q value 90% of the time, and choosing one of the four actions at random the remaining 10% of the time. We should "break-ties" when the Q-values are zero for all the actions (happens initially) by essentially choosing uniformly from the action. So now you have two conditions to act randomly: for $\epsilon$ amount of the time, or if the Q values are all zero.

The simulation consist of a series of trials, each of which runs until the agent reaches the goal state, or until it reaches a maximum number of steps, which you can set to 100. The reward at the goal is 10, but at every other state is 0. You can set the parameter $\gamma$ to 0.9.

2.2. *Experiments.*

1. Basic Q learning experiments (10 points).
   (a) Run your algorithm several times on the given environment.
   (b) Run your algorithm by passing in a list of 2 goal locations: (1,8) and (5,6). Note: we are using 0-indexing, where (0,0) is top left corner. Report on the results.
2. Experiment with the exploration strategy, in the original environment (20 points).
   (a) Try different $\epsilon$ values in $\epsilon$-greedy exploration: We asked you to use a rate of $\epsilon$=0.1, but try also 0.5 and 0.01. Graph the results (for the 3 $\epsilon$-values) and discuss the costs and benefits of higher and lower exploration rates.
   (b) Try exploring with policy derived from the **softmax of Q-values** described in the Q learning lecture. Use the values $\beta \in \{1, 3, 6\}$ for your experiment, keeping $\beta$ fixed throughout the training.
   (c) Instead of fixing the $\beta = \beta_0$ to the initial value, we will increase the value of $\beta$ as the number of episodes $t$ increase:

   $$\beta(t) = \beta_0 e^{kt} \tag{2.1}$$

   That is, the $\beta$ value is fixed for a particular episode. Run the training again for different values of $k \in \{0.05, 0.1, 0.25, 0.5\}$, keeping $\beta_0 = 1.0$. Compare the results obtained with this approach to those obtained with a static $\beta$ value.
3. Stochastic environments (15 points).
   (a) Make the environment stochastic (uncertain), such that the agent only has say a 95% chance of moving in the chosen direction, and a 5% chance of moving in a random direction.
   (b) Change the learning rule to handle the non-determinism, and experiment with different values of the probability that the environment performs a random action $p_{rand} \in \{0.05, 0.1, 0.25, 0.5\}$ in this new rule. How does performance vary as the environment becomes more stochastic?

   Use the same parameters as in the first part, except change the alpha to be less than 1, e.g., $\alpha = 0.5$. Report your results.

2.3. *Write-up.* Hand in a brief summary of your experiments. For each sub-section, this summary should include a one paragraph overview of the problem and your implementation. It should include a graph showing number of steps required to reach the goal as a function of learning trials (one trial is one run of the agent through the environment until it reaches the goal or maximum number of steps). You should also make a figure showing the policy of your agent for the first 2.1.1 section. The policy can be summarized by making an array of cells corresponding to the states of the environment, and indicating the direction (up, down, left,right) that the agent is most likely to move if it is in that state.

**3. Course Evaluation, 5 pts.** You get the final 5 points on the assignment since you submitted your course evaluation.