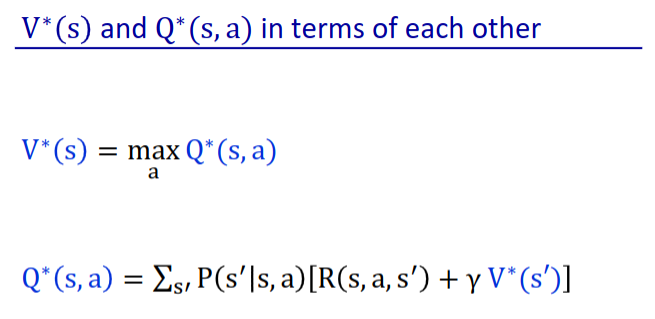
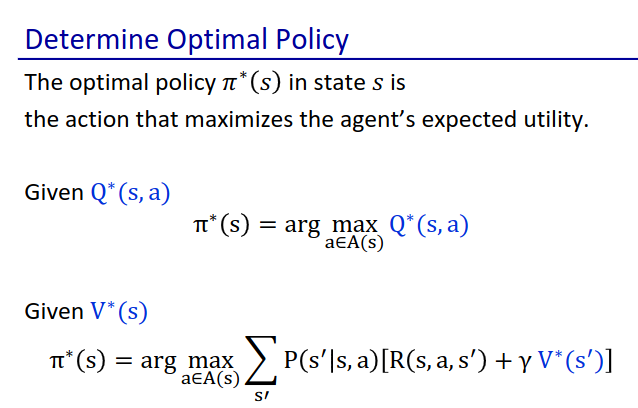
| **Value iteration**   * Method for iteratively learning the value function (V\*) of the model   + This can then be used to determine the optimal policy * **Process**  1. Start with arbitrary values for the 0th iteration V\*0(s)  * We can assign all spaces an initial value of 0  1. At the i’th iteration, compute Vi+1(s) as follows:  * This sets the value of the state to the largest value obtainable by taking some action at the state  1. Terminate if the changes are small enough  * Example on slide 33-37   **Q learning**   * Method for iteratively learning the optimal Q\* function of the model   + This can then be used to determine the optimal policy * **Process**   + Our model starts with some assumption about Q\*(s, a) and updates based upon taking the action and observing the reward   + **Update rule:**     - Q\*(s, a) Q\*(s, a) + Q\*(s’, a’) - Q\*(s, a)       * is the learning rate       * Q\*(s’, a’) is the observed value after taking the action       * Q\*(s, a) (on the right side of equation is the current estimate * **Advantages of Q learning:**   + Q learning does not need to know the probability of each transition, since it works based on observations     - With Q learning we can just let the agent explore the world     - Advantage over value iteration, which does need the probability of each transition * **Exploration-exploitation tradeoff**   + **Exploration**     - Try many different actions     - Most actions will be suboptimal, but some may allow us to learn more accurate Q\*(s, a) values   + **Exploitation**     - Stick to optimal options based on current Q\*(s, a) values     - If the current assumptions about Q\*(s, a) are inaccurate, actions may be suboptimal   + When doing Q-learning, we try to strike a balance between exploration and exploitation     - **The -greedy mechanism**       * Explore with probability of , and exploit with probability       * The higher the , the more likely we explore |
| --- |

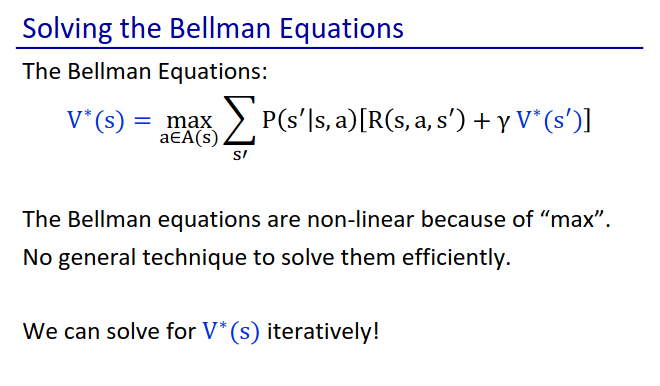
Recap:

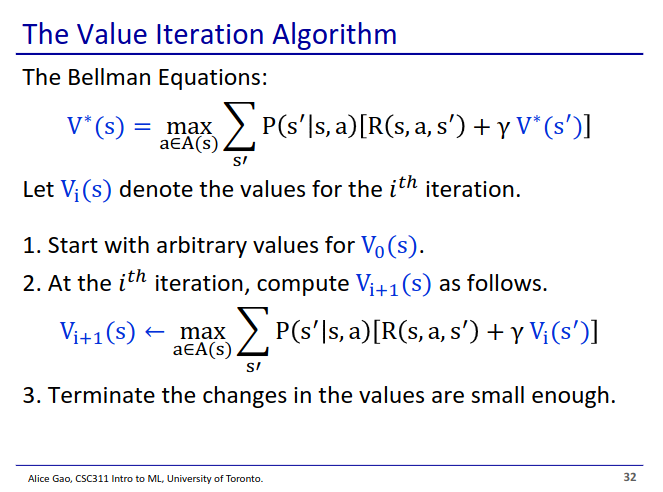
* Reinforcement learning is sort of in-between of supervised and unsupervised learning
* Key is a concept of time
  + We give a reward signal at some time interval
* Grid-world problem
  + What is the ideal policy?
* Bellman equations
  + Value (V\*)
  + Value of a state-action pair (Q\*)
  + V\* and Q\* are related to one another

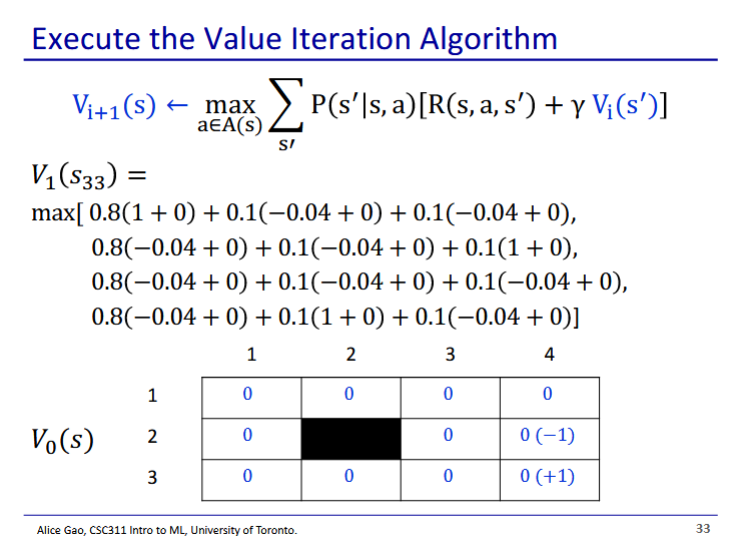
Final exam will not have calculations. Most it will have is plug-in numbers

* + 
  + Once we know V\* or Q\*, we can get the optimal policy
    - 
    - Optimal policy is easier to determine once we know Q (that’s why we use Q for Q learning)

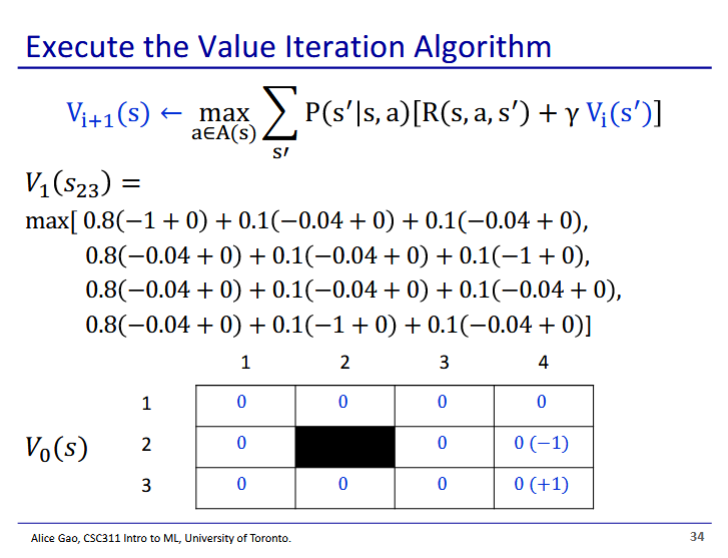


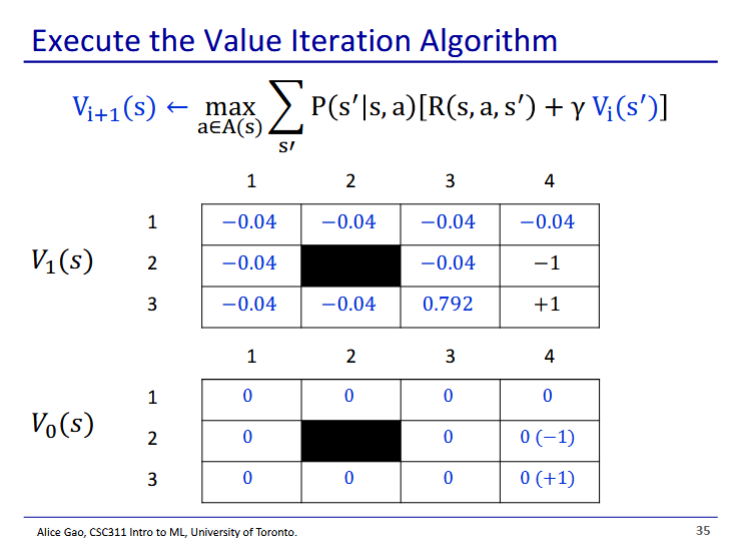




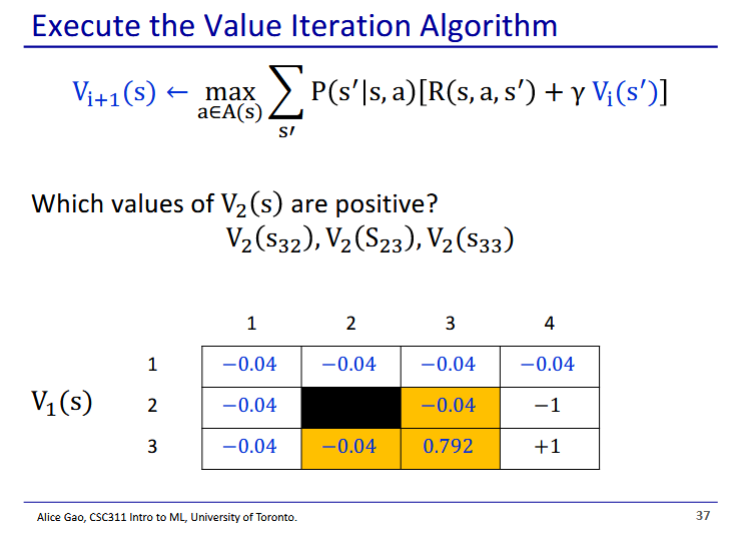
****

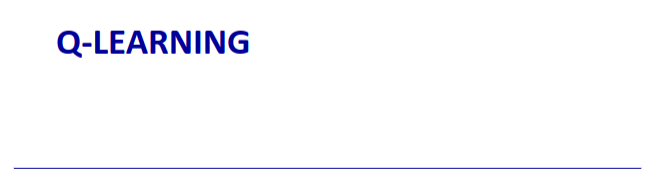
* We first initialise all V\* to 0
  + Initialisation values specifically doesn’t really matter
* Gamma is 1 in this case, so we don’t have to include it
* We then find the action that results in the largest value
  + Cell’s new value becomes that value
  + Repeat for all cells

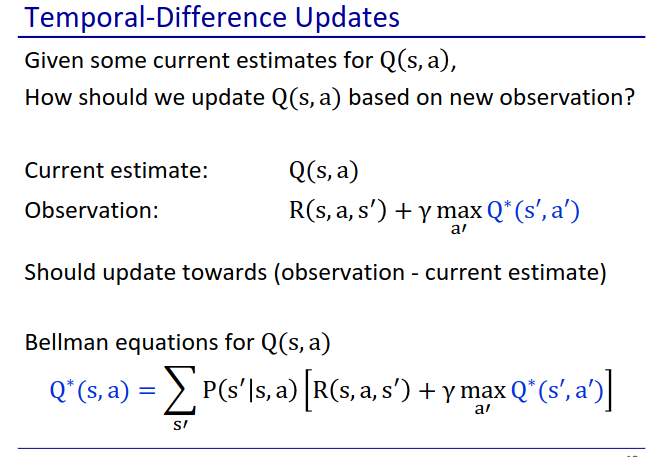
****



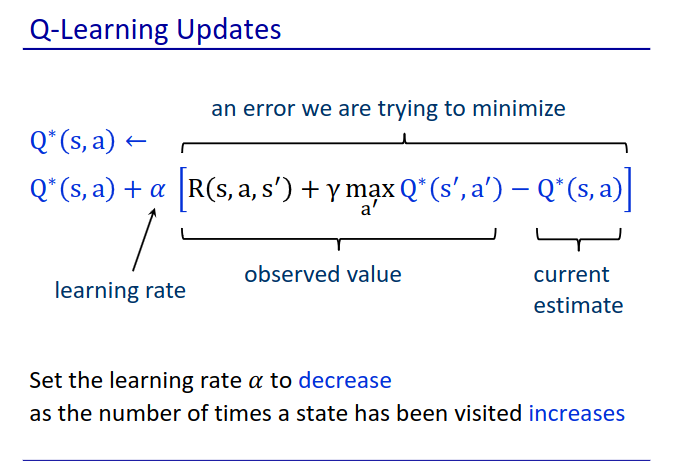
* After first update, the only positive value is the one next to the exit
  + All other cells have only max of 0 value neighbours
* All other states accumulate a negative value
  + The only state that has found the exit is the one right next to the exit



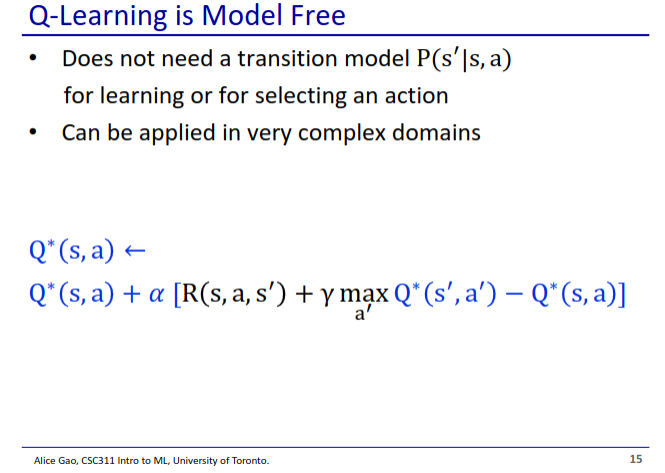




* Q learning works with the Q values (duh)
* Our goal is to learn the optimal Q\* values
  + These can then be used to create the optimal policy
  + We learn the optimal Q by repeatedly updating our Q based on our observations
* Note the observation is without the probability
  + To get an observation, we stand at s and take action a
  + Once we have taken the action, there is no probability attached to where we end up since we know where we ended up

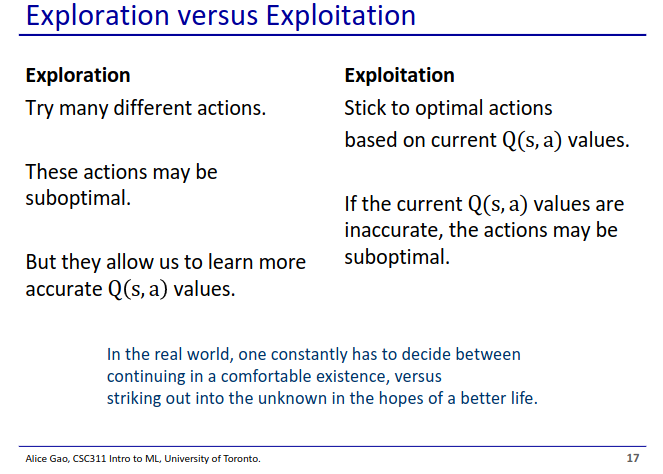


* We update our estimate based on the difference between our observed value and our current estimate
* We set the learning rate to decrease as we have visited the same state more times
  + When we don’t have a lot of data, each observation is more important
  + When we have a lot of data, each observation is less important
* We set our model to just repeatedly explore around the world, and update Q\*(s, a) after each action

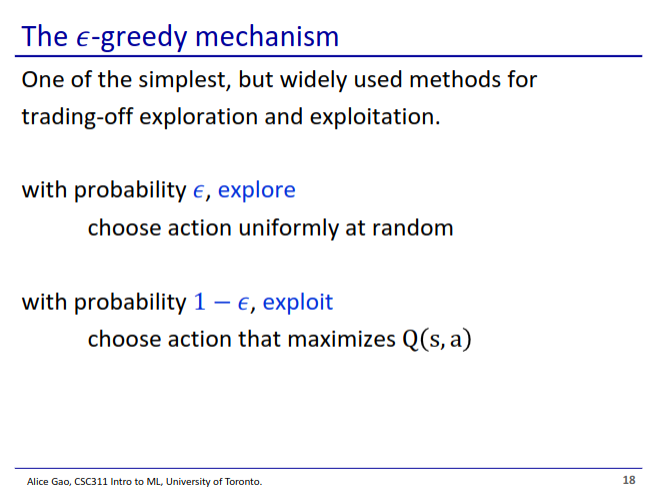


* A big advantage of Q-learning is that it is model free
* We don’t need to know P(s’|s, a)
  + We don’t need to know the transition probability
* This is what makes Q learning different than the value iteration algorithm
  + For value iteration, we need to know the probabilities attached to each action
  + For Q learning, we don’t need the probabilities
    - We put the agent in a already modelled world and let it do things





* When our model learns, there is a tradeoff between exploration and exploitation
* Exploration means our model tries a wide range of different actions
  + It will spend a lot of time trying suboptimal actions, but may find a better path
* Exploitation means our model tries to optimise based off of what it already sees
  + It might miss out on better paths it has not considered
* Usually we try to get a healthy mix of the two



* Each time we need to do an action, we toss a coin
  + With some small probability e, we explore (take some random action)
  + With some larger probability, we exploit (take the current predicted best action)