C3 Unsupervised Learning, Recommender System and Reinforcement Learning

- **Welcome!**
  - Unsupervised learning
    - Clustering
    - Anomaly detection
  - Recommender systems
  - Reinforcement learning
- **W1 Unsupervised Learning**
  - Clustering
    - What is clustering
      - Training set {x(1),..., x(m)} without the target label y
      - Applications of clustering
        - Grouping similar news
        - DNA analysis
        - Market segmentation
        - Astronomiacal data analysis
    - K-means algorithm
      - 1. Randomly initialize K cluster centroids μ1, μ2,.., μk
      - 2.
        Repeat {

        #assign points to cluster centroids
        for i =1 to m:
        c(i) := index (from 1 to K) of cluster centroid closet to x(i)

        #move cluster centroids
        for k=1 to K:
        μk := average(mean) of points assigned to cluster k
        }

- Optimization objective
  - c(i) = index of cluster(1,2..K) to which example x(i) is currently assigned
    μk = cluster centroid k
    μc(i) = cluster centroid of cluster to which example x(i) has been assigned
  - Cost function/distortion function(L2 norm):
    J(c(1),...,c(m),μ1,.. μk) = ∑||x(i)-μc(i)||^2/m
- Intialilizing K-means
  - Random initialization
    - Choose K<m
    - Randomly pick K training examples
    - Set μ1,.., μk equal to these K examples
  - #better than running K-means once
    For i = 1 to 100{
    randomly initialize K-means
    run K-means, get c(1),...,c(m), μ1,.., μk
    compute J(c(1),..,c(m), μ1,.., μk)}
    pick set of clusters that gave lowest cost J
- Choosing the number of clusters
  - Elbow method: calculate the cost function with K clusters
  - Evaluate k-means based on how well it performs on that later purpose

- Anomaly detection
  - Finding unusual events
    - Density estimation
      - p(x): probability of x being seen in dataset
      - p(x) < ε , anomaly point
      - p(x) >= ε, normal
    - Anomaly detection example
      - (Fraud detection)
        x(i) = features of user i's activities
        model p(x) dorm data
        identify unusual users by checking which have p(x) < ε
  - Gaussian distribution
  - Algorithm

- 1. Choose n features x(i) that you think might be indicative of anomalous examples
- 2. Fit parameters μ1, μ2,...μn, δ1^2,..., δn^2
- 3. Given new example x, compute p(x):
  p(x) = p(x1, μ1, δ1^2)* p(x2, μ2, δ2^2)*..* (xn, μn, δn^2)= N(μ1, δ1^2)*N(μ2, δ2^2)*..* N(μn, δn^2)
  anomaly if p(x) < ε
- If p(xi) is very small then the overall p(x) will be small

- Developing and evaluating an anomaly detection system
  - Fit model p(x) on training set
  - On a cross validation example x, predict:
    y = 1 if p(x) < ε else 0
  - Possible evaluation metrics:
    true postive,false postive , true negative
    precision/recall
    F1-score
  - Can also use cross validation set to choose parameter ε
- Anomaly detection vs supervised learning
  - The number of postive examples <= 20 then use anomaly detection
  - If the future anomalies look nothin like the examples before then use anomaly detection
- Choosing what features to use
  - Change the features to more-Gaussian features
  - Error analysis for anomaly detection
  - Monitoring computers in a data center
    - choose features that might take on unusually large or small values in the event of an anomaly
    - Deciding featur choice based on p(x):
      large for normal examples
      becomes small for anomaly in the cross validation set

W2 Recommender System

- Collaborative filtering
  - Using per-item features

- nu: number of users
  nm: number of movies
  n: number of features
- For user j: predict rating for movie j as:
  w(j)·x(i) + b(j)
- r(i,j) = 1 if user j has rated movie i
  y(i,j): rating given by user j on movie i
  x(i): feature vector for movie i
  w(j), b(j): parameters for user j
  m(j) = number of movies rated by user j
- Cost function to learn w(j),b(j) for user j:
  min J(w(j), b(j))
  = ∑(w(j)·x(i) + b(j)-y(i,j))^2/((2m(j)) + λ∑(wk(j))^2 / (2m(j)), k=1,2..n
- Cost function to learn w(1),..w(nu),b(1),..b(nu) for all users:
  min J(all parameters w,b)
  = ∑ ∑(w(j)·x(i) + b(j)-y(i,j))^2/2 + λ∑ ∑(wk(j))^2 / 2, k=1,2..n

▾ Collaborative filtering algorithm

- Given w(1),.. w(nu), b(1),..,b(nu) to learn x(i):#single feature
  J(x(i))
  = ∑(w(j)·x(i) + b(j)-y(i,j))^2/2 + λ∑(x(k))^2 / 2
- To learn x(1),..x(nm):# all features
  J(x(1),..,x(nm))
  = ∑ ∑(w(j)·x(i) + b(j)-y(i,j))^2/2 + λ∑ ∑(x(k))^2 / 2
- To learn w,b and x:
  J(w,b,x) = ∑(w(j)·x(i) + b(j)-y(i,j))^2/2 + λ∑ ∑(x(k))^2 / 2 + λ∑ ∑(wk(j))^2 / 2
- Gradient descent:
  repeat

$$w_i^{(j)} = w_i^{(j)} - \alpha \frac{\partial}{\partial w_i^{(j)}} J(w,b,x)$$
$$b^{(j)} = b^{(j)} - \alpha \frac{\partial}{\partial b^{(j)}} J(w,b,x)$$
$$x_k^{(i)} = x_k^{(i)} - \alpha \frac{\partial}{\partial x_k^{(i)}} J(w,b,x)$$

▾ Binary labels: favs, likes and clicks

- For binary labels: predict that the probability of y(i,j) = 1 is given g(w(j)·x(i) + b(j)) where g is a sigmoid function

- Cost function for single example:
  L(g(w(j)·x(i) + b(j),y(i,j)) = -y(i,j)log(g(w(j)·x(i) + b(j))) - (1-y(i,j))log(1-g(w(j)·x(i) + b(j)))
- Cost function for all example:
  J(w,b,x) = ∑ L(g(w(j)·x(i) + b(j),y(i,j))

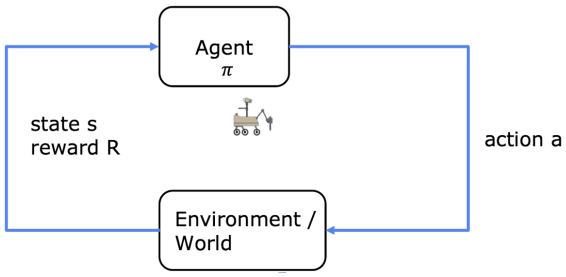
W3 Reinforcement Learning

key concepts

- policy π： what action (a=π(s)) to take in every state so as to maximize the return
- states\actions\rewards\discount factor γ\return\policy

markov decision process (MDP)

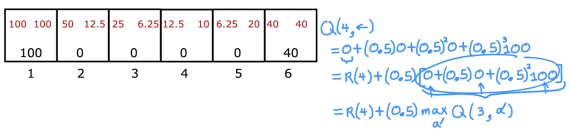
- the future only depends on the current state
-



state-action value function

- Q(s, a) = return if you
  .start in state s
  .take action a(once)
  .then behave optimally after that
- Bellman Equation:
  Q(s, a) = R(s) + γmaxQ(s', a')
-

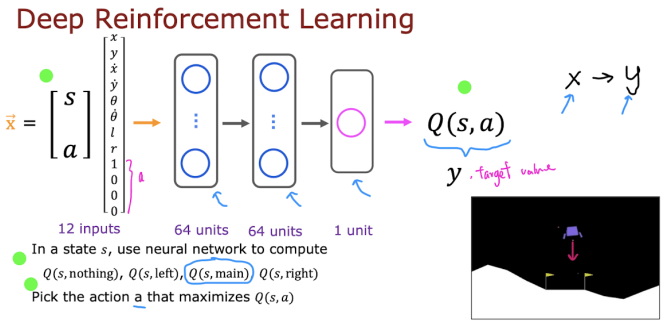
$$Q(s,a) = R(s) + \gamma \max_{a'} Q(s',a')$$



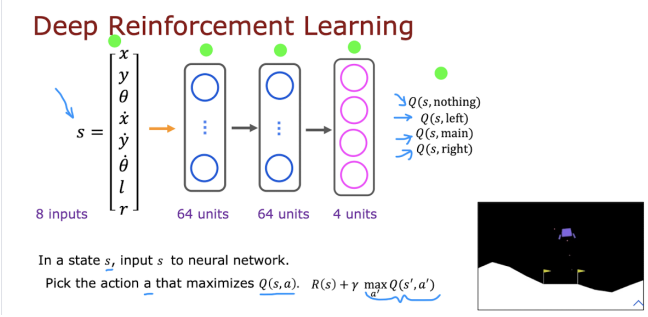
random(stochastic) environment

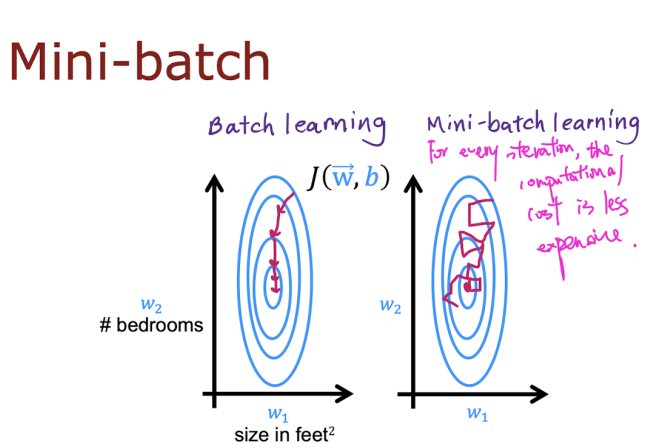
- expected return
  = average(R1+ γR2+ γ^2R3+ γ^3R4+ ...)
  = E(R1+ γR2+ γ^2R3+ γ^3R4+ ...)

- maximum the average value
- Bellman Equation:
  Q(s, a) = R(s) + γE[maxQ(s', a')]

- continuous state spaces
  - Lunar Lander Problem
    - Learn a policy π
    - given s =[x, y, x', y',θ, θ', l, r]
    - picks action a=π(s) so as to maximum the return
    - γ = 0.985
    - deep reinforcement learning
      -
      

      - Bellman Equation:
        Q(s, a) = R(s) + γmaxQ(s', a')
        x = [s, a]
        y = R(s) + γmaxQ(s', a')]
      - learning algorithm
        Initialize neural network randomly as guess of Q(s, a)
        Repeat{
        Take actions in the lunar lander.Get(s, a, R(s), s')
        Store 10000 most recent (s, a, R(s), s') tuples *replay buffer
        Train neural network:
        create training set of 10000 examples using
        x = (s, a) and y = R(s) + γmaxQ(s', a')
        Train Qnew such that Qnew ~ y
        set Q = Qnew
        }
  - improved neural network architecture

- 

- algorithm refinement: ε -greedy policy
  - Take actions in the lunar lander
    >>>
    1. with probability 1-ε, pick the action a that maximizes Q(s, a)
    2. with probability ε, pick an action a randomly
  - 1 for greedy exploitation
    2 for exploration
  - greedy 1-ε of the time and exlporing ε of the time
  - start ε very high and gradually decrease
- mini-batch and soft update
  - if the sample size is too much and when repeating gradient descent, it only minus a little
  - so for each iteration, only choose the subset of training set
    create training set of 10000 examples
    >>> create training set of 1000 examples
  - 

- (soft update)set Q = Qnew
  >>>
  if Q(W, B) then Q(Wnew, Bnew)
  w = 0.01Wnew+0.99W
  B=0.01Bnew+0.99B
  *0.01 and 0.99 can be modified

- further reading
  - Mnih, V., Kavukcuoglu, K., Silver, D. et al. Human-level control through deep reinforcement learning. Nature 518, 529–533 (2015).
    Lillicrap, T. P., Hunt, J. J., Pritzel, A., et al. Continuous Control with Deep Reinforcement Learning. ICLR (2016).
    Mnih, V., Kavukcuoglu, K., Silver, D. et al. Playing Atari with Deep Reinforcement Learning. arXiv e-prints. arXiv:1312.5602 (2013).