CS 6375 Final Examination
(Please do not forget to write your name.)

Answer all the questions.

I certify that I have not discussed this exam with any other students. I also certify that I have not used any other resources than the class lecture notes. I understand that I will get a 0 and get reported if found guilty of cheating of any sort.

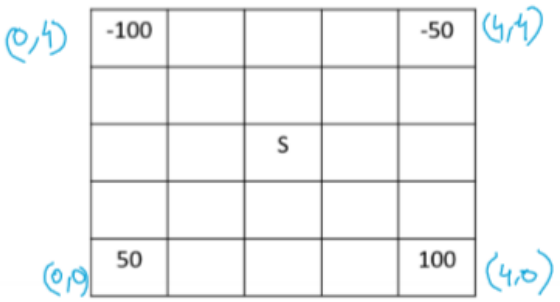
Kapil
KAPIL GAUTAM

Please sign and Print your name above

Intentionally left blank.

1. (RL and Supervised Learning 10 points)

- (6 points) Consider following gridworld. The agent starts in the center and huge rewards are present at the end of the corner cells. For the purposes of this problem, you can assume that the state transitions are deterministic.



(a) Design an MDP for this problem (recall the definition of an MDP and define all the components of an MDP clearly).

MDP = $< S, A, T, R >$

S = Observable States = $\{ x,y \to x+1,y \;;\; x-1,y \;;\; x,y-1 \;;\; x,y+1 \;\; \exists\, x \geq 0, y \geq 0, x < 5, y < 5 \}$

A = Action = {Up, Down, Right, Left}

T = Transition Probability (s', a, s) = 1 for each (state, action) pair as state transitions are deterministic

R = Reward (s, a) = { $R(0,0) = 50$, $R(0,4) = -100$, $R(4,4) = -50$, $R(4,0) = 100$, all other coordinates $R(x,y) = 0$ }

(b) Now, assume that the goal is minimize the number of steps. Will your MDP definition change? If so, what is the change?

Yes, to reach the goal in minimum number of steps we can put a -ve living reward

S = Observable State = Same as above

A = Action = {Up, Down, Right, Left}

T = Transition Probability (s', a, s) = 1 for each (state, action) pair as state transitions are deterministic

R = Reward (s, a) = { $R(0,0) = 50$, $R(0,4) = -100$, $R(4,4) = -50$, $R(4,0) = 100$, all other coordinates $R(x,y) = -1$ }

- (4 points) Assume that we have a large data set with k examples, in which each example i has one real-valued input x_i and one real valued output y_i. To fit this data, we assume the following model with an unknown parameter w that needs to be learned from the data.

$$y_i' = log(w^2 x_i)$$

Derive the gradient descent update for w using mean-squared error as the criterion.

$$w := w - \eta \nabla L(y', y) \quad , L = \frac{\sum (y' - y)^2}{k}$$

$$\frac{\partial L_i}{\partial w} = \frac{1}{k} \sum \frac{\partial}{\partial w} \left(\log(w^2 x_i) - y \right)^2$$

(Considering base e of logarithm, and single example)

$$\frac{\partial L}{\partial w} = 2 \left(\log(w^2 x) - y \right) \frac{\partial}{\partial w} \left(\log(w^2 x) \right)$$

$$\Rightarrow \frac{2 \left(\log(w^2 x) - y \right)}{w^2 x} \frac{\partial}{\partial w} (w^2 x)$$

$$\Rightarrow \frac{2 \log(w^2 x) - y}{w^2 x} (2wx) \frac{\partial}{\partial w} (w)$$

$$\Rightarrow \frac{4 \log(w^2 x) - y}{w^2 x} (wx) \Rightarrow \frac{4 \left(\log(w^2 x) - y \right)}{w}$$

$$\nabla L(y', y) \Rightarrow \frac{4}{k} \sum_{i=1}^{k} \frac{(y_i' - y)}{w}$$

$$w := w - \frac{4}{k} \eta \sum_{i=1}^{k} \frac{(y_i' - y)}{w}$$

2. (Unsupervised Learning 10 points)

- (4 pts) Can the starting cluster assignments affect the final output of K-means? If so, show a data set and two different starting cluster configurations such that running K-means (where k=3) yields two different results. If not, explain why the starting cluster assignments do not matter.

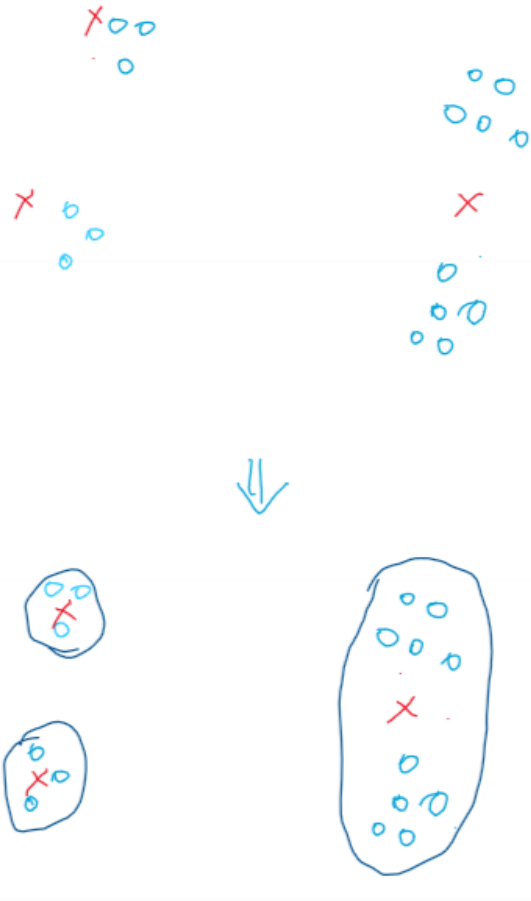
Yes, starting cluster assignments can affect the final output of K-means.

K=3

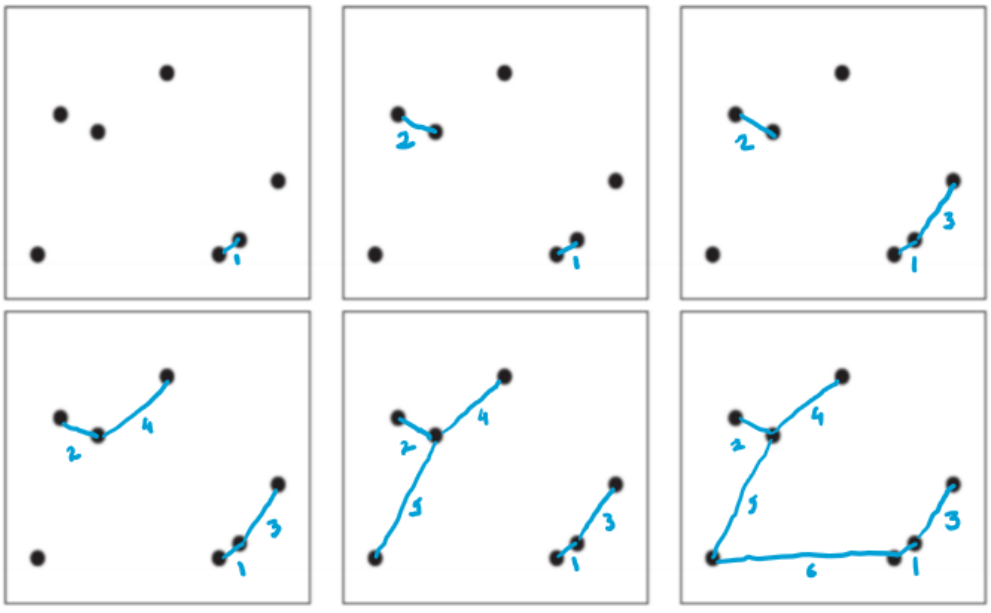
Dataset Starting Configuration 1

K=3

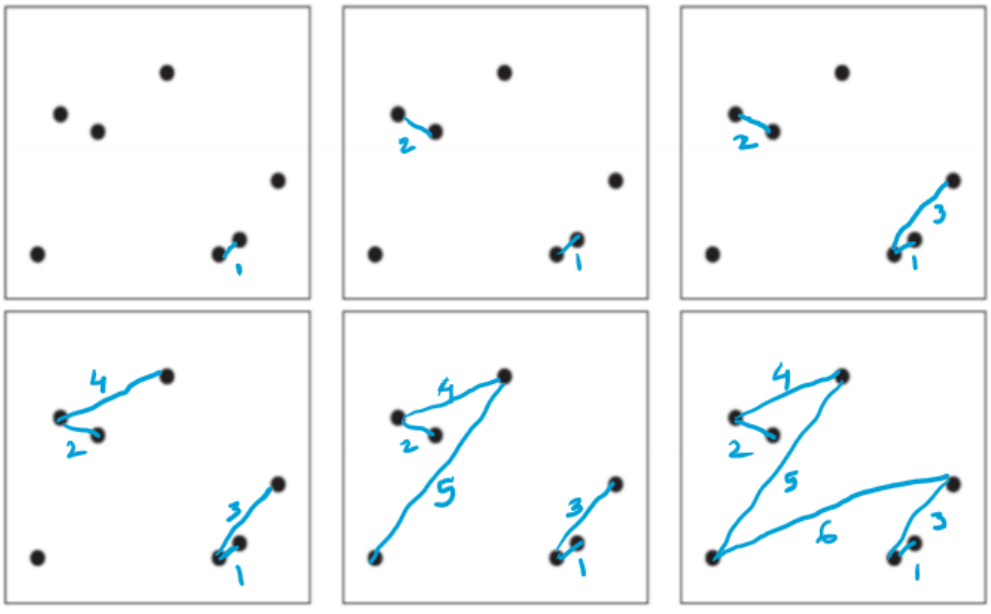
Dataset Starting Configuration 2



- (6 points) Use single-link and complete agglomerative clustering to group the data described by the following distance matrix. Stop when converged or after 6 steps, whichever comes first. Show each step separately.



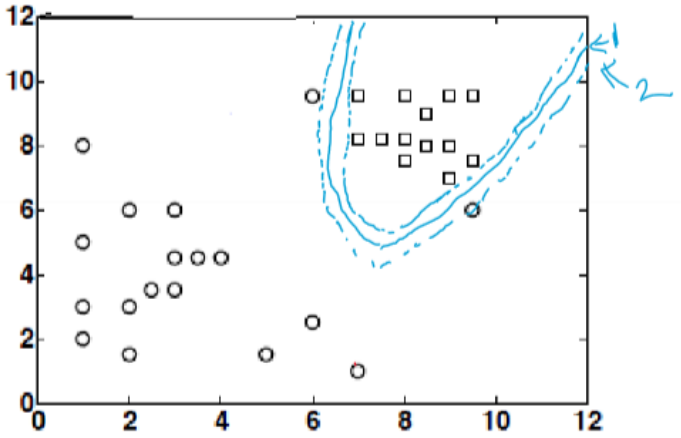
Single Link (Closest Points)



Complete Link (Furthest Points)

3. (Supervised Learning 10 points)

- (4 points) You are given the following data set. The data could be error prone so we cannot trust any point too much. Let us assume that we are learning a SVM with quadratic kernel (hint: In this figure it will look like a curve from the top axis to the right axis).



(a) Where would your decision boundary be for very large values of C (i.e., as $C \to \infty$)? Draw this in the figure and label it as 1. Soft margin, Min. Error (overfit)

(b) Where would your decision boundary be for $C = 0$? Draw this in the figure and label it as 2. Same as (a) Hard margin (Underfit)

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in \mathcal{X}$, $y_i \in \{-1, +1\}$.
Initialize: $D_1(i) = 1/m$ for $i = 1, \ldots, m$.
For $t = 1, \ldots, T$:

- Train weak learner using distribution D_t.
- Get weak hypothesis $h_t : \mathcal{X} \to \{-1, +1\}$.
- Aim: select h_t with low weighted error:

$$\varepsilon_t = \Pr_{i \sim D_t}\left[h_t(x_i) \neq y_i\right].$$

- Choose $\alpha_t = \frac{1}{2} \ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$.
- Update, for $i = 1, \ldots, m$:

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution).

Output the final hypothesis:

$$H(x) = \operatorname{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right).$$

- (3 points) **AdaBoost Classifier** At iteration t of AdaBoost, if the weak hypothesis h_t performs **worse than random guessing** (that is, $\varepsilon_t > \frac{1}{2}$), then correctly classified examples will be weighted more than misclassified examples. **True** or **False**? Explain your reasoning.

True, for $\varepsilon > \frac{1}{2}$, $\alpha < 0$

(e.g. $\varepsilon = 0.7 \Rightarrow \alpha = \frac{1}{2} \ln\left(\frac{0.3}{0.7}\right) = -0.423$)

For correct classification $\Rightarrow$ $y = 1, h(x) = 1$ OR $y = -1, h(x) = -1$ $\Rightarrow (y)(h(x)) = +ve$

$D_{t+1} = \frac{D_t}{Z_t} e^{(-\alpha(+ve))} \Rightarrow \frac{D_t}{Z_t} e^{(-(-ve)(+ve))} \Rightarrow \frac{D_t}{Z_t} e^{(+ve)} \Rightarrow$ Increase

- (3 points) Suppose your training set consists of 200 examples of class Red, 300 of class Green and 400 of class Blue. What is the maximum possible information gain of any attribute. (Note that you do not have to calculate the final value. Just write out the expression).

#Red = 200, # Green = 300, # Blue = 400

Information Gain = $H(Y) - H(Y|X)$

To max. Information gain, $H(Y|X) \downarrow$ and $H(Y) \uparrow$

$IG_{max} = H(Y) - 0$

For multiclass, we can compute by One class vs others approach

⇒ #Red = 200, # Not Red = 700 → Say this Gives IG_1

Green = 300, #Not Green = 600 → Say this Gives IG_2

Blue = 400, # Not Blue = 500 → Say this Gives IG_3

Max Info Gain = $Max(IG_1, IG_2, IG_3)$

$= Max(H(Y=Red), H(Y=Green), H(Y=Blue))$

4. Supervised Learning (10 points)

- (4 pts) Consider the algorithm for k-NN presented below. Describe the potential problem with this algorithm (hint: Think of the case where there are duplicates in the data set). Assume that the distance function returns 0 for the distances between objects that are the same. How will you fix this problem.

Algorithm for finding K nearest neighbors.

for i = 1 to number of data objects **do**
  Find the distances of the i^{th} object to all other objects.
  Sort these distances in decreasing order.
  (Keep track of which object is associated with each distance.)
  return the objects associated with the first K distances of the sorted list
end for

① For Duplicates, we can do a pre-processing step to just have the unique examples, and remove the duplicates for the algorithm.

② For KNN, if we are returning first k distances, then we need to sort in ascending order of distances.

③ Loop is from 1 to n, while algo says to compute distance to all other object, only i^{th} to n^{th} are considered here. It should say to compute for all remaining objects

④ Edge case, if in this algo - with no pre processing, and having k duplicates with Increasing order, then it will keep clustering itself (duplicates).

- What is the difference between gradient-boosting with loglikelihood and Adaboost (use the weight equations to explain)?

Adaboost ⇒ Weight is redistributed

$$D_{t+1}(i) = \frac{D_t(i)\exp(-\alpha y h(x))}{Z_t}$$

Gradient Boosting ⇒ Gradient is adjusted

$$\hat{y}_m = \hat{y}_{m-1} - \eta\left(-\frac{\partial L(y, F_{m-1}(x))}{\partial (F_{m-1}(x)}\right)$$

$F_{m-1}(x)$ can be logistic loss f^m

$$\hat{y}_m = \hat{y}_{m-1} - \eta\left(y_i - P(y=1 \mid x_i)\right)$$

- Explain the effect of bagging and boosting on bias and variance?

Bagging → Decrease variance (Slightly increase Bias)

Boosting → Decrease Bias (Slightly increase Variance)

- Given a particular classifier (say with low bias) what ensemble method would you employ? What will you use with high variance classifier?

Already low bias,

Low Bias → Use Bagging

High Variance → Use Bagging (To reduce variance)