MDS 202035 SHIULI SUBHRA GHOSH

1	2	3	4	5	6	7	8
4	5	5	5	5	3	5	5

$\frac{37}{40}$

Date:- 06.07.2021

Shiuli Subhra Ghosh
Final Sem Exam
Roll NO. MDS

2. Outliers are those which has a low affinity to all the clusters.

DBScan for detecting outliers :- (Assumed uniform density)

DBScan algorithm defines clusters in the region of uniform density. It identifies in a given volume how many data points are there.

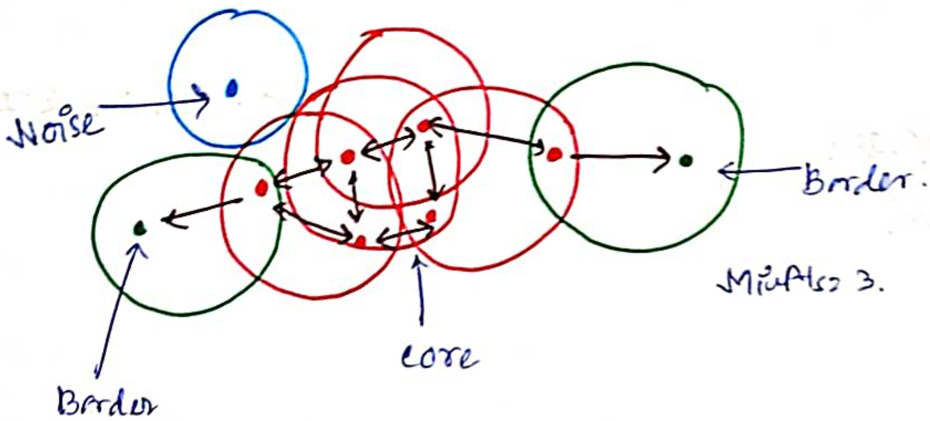
So, - We construct a small ball around each point of radius ϵ. Then we see how many points will lie inside each ball. Some may have many, some may have nothing. This region is called ϵ-neighbourhood.

- Then we identify a threshold for neighbours within the ball. ~~the~~

- Core points are those if it has least Minpts in its ϵ-neighbourhood. So, all the core points are located in the dense region.

- Border points are those who are attached to the core points but not core themselves.

- And Noise / Anomaly are those which are neigther core, nor border and doesnot have any minpts in its ~~E~~ neighbourhoods.



The dense point is a neighbour of dense points, so, if we get directed arrows in both side, then the points are dense points. This algorithm identifies noise as we have said earlier. This noise can also be interpreted as outliers. These are the points which don't come naturally into any of the clusters.

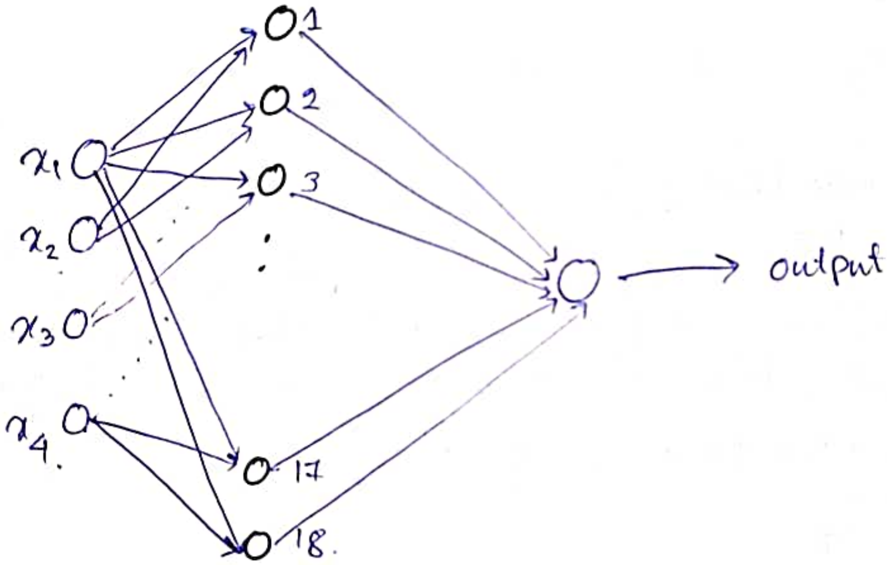
(b) Clustering using Gaussian Mixtures. :- (outlier identification)

In mixture of Gaussians, ~~as~~ we use Expectation maximization to estimate the parameters of each Gaussian distribution. and then we assign all the data points to the best Gaussian. We can simultaneously estimate mean and variance of the Gaussians or we can estimate only the mean keeping σ to be equal.
Outliers are those points which are outside $K.\sigma$. for all the gaussians. K can be determined according to the choice of the situation. If some point is near to one of the zone it is OK to be included in that Gaussian. But if it is outside of all the gaussian zones, then the outliers can be labelled.

5

5. We have a network with 4 input features x_1, x_2, x_3, x_4 and a single output y. We assume that each pair of adjacent layers is completely connected and there is a single output layer.

(a) We need to estimate some parametes for a shallow network with 1 hidden layer.



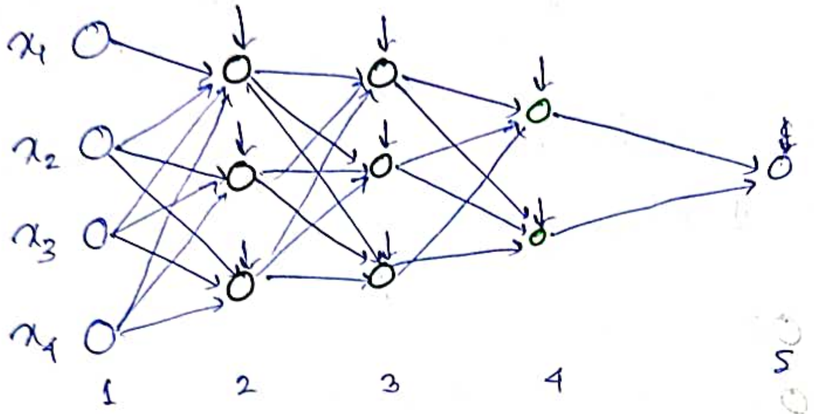
So, we need to assume all the weights and bias. for each node in the hidden layer 4 connections will be there. So, 4 weights for each node and one single bias.

so, till the 1st layer, $(\underset{\uparrow \text{weights}}{4\times 18} + \underset{\uparrow \text{bias}}{18}) = 72+18 = 90$ parameters.

then for the output layer again 18 more weights from the hidden layer will add up + one bias.

So, the total no. of parameters to be estimated $= 90+18+1$
$= 109$

(b) A deep network with 3 hidden layers, where the first 2 layers have 3 nodes each and the third layer has 2 nodes.



So, the total numbers of parameters to be estimated

$$\underbrace{\overbrace{(4\times 3)}^{1-2}}_{\text{weights}} + \underbrace{3}_{\text{bias}} + \underbrace{\overbrace{(3\times 3)}^{2-3}}_{\text{weights}} + \underbrace{3}_{\text{bias}} + \underbrace{\overbrace{(3\times 2)}^{3-4}}_{\text{weights}} + \underbrace{2}_{\text{bias}} + \underbrace{\overbrace{(2\times 1)}^{4-5}}_{\text{weights}} + \underbrace{1}_{\text{bias}}$$

$= 12 + 3 + 9 + 3 + 6 + 2 + 2 + 1$

$= 38$

5

6. We made following assumptions about the loss function C for Neural Networks.

- For input x, $C(x)$ is a function of only the output layer activation a^L.

For each training input (x_i, y_i) sum squared error is $(y_i - a_i^L)^2$. here x_i, y_i are fixed values and only a_i^L is a variable. This cost function is also a function of y but if x is fixed then y is also fixed for a given training sample. So, the cost function is only the function of the of the output a_i^L. Hence we can take this assumption.

Why not intermediate weights & biases?

- Total cost is average of individual input cost

Each input x_i incurs cost $C(x_i)$ and the total cost is $\cancel{\frac{1}{n}\sum_{i=1}^{n}(y_i - a_i^L)^2}$ $\frac{1}{n}\sum_{i=1}^{n} C(x_i)$. The mean square error can be considered, $\frac{1}{n}\sum_{i=1}^{n}(y_i - a_i^L)^2$. This assumption is taken because, back propagation allows us to compute $\frac{\partial C_x}{\partial w}$ and $\frac{\partial C_x}{\partial b}$ for a single training example. We then can recover $\frac{\partial C}{\partial w}$ and $\frac{\partial C}{\partial b}$ by averaging over training examples. We can also extrapolate change in individual cost $C(x)$ to change in the overall cost. in case of Stochastic Gradient descent. SGD make us of 2nd clause. It make sense to take a small no. of inputs and apply this update rather than waiting for the entire batch.

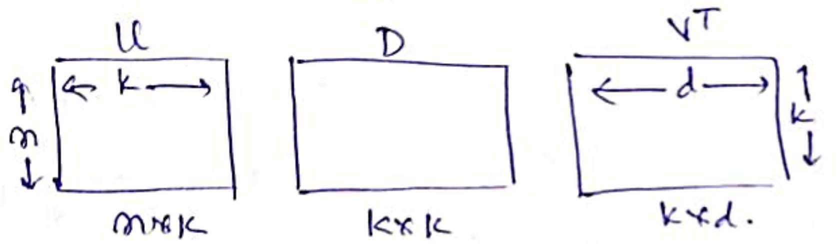
Sample mean approximates overall mean

3

4. We can decompose a matrix M as UDV^T

Where D is a $k \times k$ diagonal matrix, with +ve real entries.

- U is $m \times k$, V is $D \times k$
- columns of U, V are orthonormal.



Singular vector is a unit vector that passes through the origin. and we want to find the best k singular vectors to represent the feature space.

First singular vector :- is the unit vector through origin that maximizes the sum of projections of all rows in M. $v_1 = \arg\max_{|v|=1} |Mv|$. We find a vector which maximizes the length of $|Mv|$.

Second singular vector :- is the unit vector through origin parpendicular to v_1, that maximizes the sum of projections of all rows in M. $v_2 = \arg\max_{v \perp v_1, |v|=1} |Mv|$.

likewise we need to find k new dimensions which are mutually parpendicular to eachother.

likewise we can also find the 3rd singular vector that maximizes the sum of the projections of all rows in M. and we keep doing.

With each singular vectors, associated singular value is.

$\sigma_j = |M v_j|$.

we repeat this r times till we get $\max_{v \perp v_1, \ldots, v_r, |v|=1} |Mv| = 0$.

then r is the rank of M. and $\{v_1, v_2, \ldots, v_r\}$ are orthonormal right singular vectors.

let $\sigma_1 = |Mv_1|$
$\sigma_2 = |Mv_2|$ $\sigma_1 \geq \sigma_2$ this might be the case.

Let us assume $\sigma_1 < \sigma_2$, then we could have obtained σ_2 first. Because v_1 is the best among all the vectors which provides the max stretch. Then we get σ_2 and v_2 which is perpendicular to v_1 and best among all the perpendicular vectors.

like we can assume that at k-th point.

$\sigma_k \geq \sigma_{k+1}$

so, $v_k \perp (v_1 \cdots v_{k-1})$ and $v_{k+1} \perp (v_1 \cdots v_k)$

~~so, we need to proove if for $\sigma_{k+1} \geq \sigma_{k+2}$.~~

and if we got $\sigma_{k+1} > \sigma_k$ then we could have found σ_{k+1} in the position of σ_k.

hence we can show that $\sigma_1 \geq \sigma_2 \geq \sigma_3 \geq \cdots \geq \sigma_k \geq \sigma_{k+1} \geq \cdots \geq \sigma_r$.

5

3. There are 3 biased coins c_1, c_2 and c_3. I have given a sequence of 1000 coin tosses, where each outcome corresponds to tossing one of $\{c_1, c_2, c_3\}$, chosen uniformly at random. Let $\{P_1, P_2, P_3\}$ be the probabilities of heads for the coins $\{c_1, c_2, c_3\}$ respectively.

$P_1 < 0.5$

$P_2 \geq 0.5$

$P_3 > 0.5$

So, we need an iterative procedure to estimate $\{P_1, P_2, P_3\}$

Let us assume $P_1 = 0.4$ $P_2 = 0.7$ $P_3 = 0.8$. ~~These~~ These are taken randomly and will affect the result accordingly. So the corresponding probability of tail is $q_1 = 0.6$ $q_2 = 0.3$ $q_3 = 0.2$.

We will Estimate the biases.

$$\hat{P_1} = \frac{0.4}{0.4+0.7+0.8} = \frac{4}{19}. \qquad \hat{q_1} = \frac{0.6}{0.6+0.3+0.2} = \frac{6}{11}$$

$$\hat{P_2} = \frac{0.7}{0.4+0.7+0.8} = \frac{7}{19}. \qquad \hat{q_2} = \frac{0.3}{0.6+0.3+0.2} = \frac{3}{11}$$

$$\hat{P_3} = \frac{0.8}{0.4+0.7+0.8} = \frac{8}{19}. \qquad \hat{q_3} = \frac{0.2}{0.6+0.3+0.2} = \frac{2}{11}$$

Now we will assign the fractional counts to the heads and tails separately. for each category.

now if out of 1000 the number of heads = 750 and tail 250.

C_1: $750 \times \frac{4}{19}$ heads and $250 \times \frac{6}{11}$ tails.

C_2: $750 \times \frac{7}{19}$ heads and $250 \times \frac{3}{11}$ tails.

C_3: $750 \times \frac{8}{19}$ heads and $250 \times \frac{2}{11}$ tails.

now we reestimate the parameter as,

$$\hat{P}_1 = \frac{750 \times \frac{4}{19}}{750 \times \frac{4}{19} + 250 \times \frac{6}{11}} = \frac{22}{41} \qquad \hat{q}_1 = \left(1 - \frac{22}{41}\right) = \frac{19}{41}$$

$$\hat{P}_2 = \frac{750 \times \frac{7}{19}}{750 \times \frac{7}{19} + 250 \times \frac{6}{11}} =$$ likewise we can calculate.

$$\hat{P}_3 = \frac{750 \times \frac{8}{19}}{750 \times \frac{8}{19} + 250 \times \frac{2}{11}}$$

Now with the estimates we repeat the whole procedure again until convergence. we repeat till there is no significant difference between the old and the new estimates of the parameters.

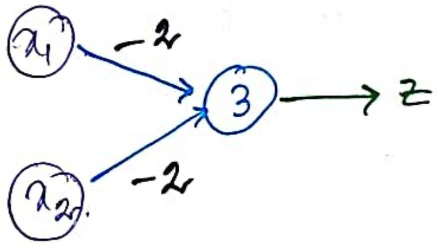
5

7. Let $f(x_1, x_2, \cdots, x_k)$ be boolean formula with k inputs. The set of inputs for which f_k is true defines a concept class $c_f \subseteq \{0,1\}^k$ of k-dimensional bit vectors.

Let ~~$c \in$~~ $c \subseteq \{0,1\}^k$ denote an arbitrary concept class.

This concept class can be represented by a neural network.

We know that NAND gate is the universal logic gate as we can design all circuits using NAND. We can design this NAND Gate using Artificial Neural Network as,



x_1	x_2	Z	Y
0	0	3	1
0	1	1	1
1	0	1	1
1	1	-1	0

If we take the ~~Nand~~ output is True if $Z>0$ and False if $Z<0$ then. Y is the output of a logic gate NAND.

And as we can represent any circuit using this NAND Gate, we can express any arbitrary concept class using this NAND function.

5

1. We have a dataset $X = \{x_1, x_2, \dots, x_N\}$ equipped with a symmetric distance function. $d(x_i, x_j) = d(x_j, x_i)$ is the distance between x_i and x_j. We construct an $N \times N$ matrix D such that $D[i,j] = d(x_i, x_j)$.

We can cluster the N columns D using the Euclidian distance, since each column is a vector of length N.

Each columns of D represents a single data point and its distance from all. and D is of the form

$$\begin{array}{ccc} \begin{bmatrix} d(x_1,x_1) & d(x_1,x_2) & \dots & d(x_1,x_n) \\ d(x_2,x_1) & d(x_2,x_2) & & d(x_2,x_n) \\ \vdots & \vdots & & \vdots \\ d(x_n,x_1) & d(x_n,x_2) & & d(x_n,x_n) \end{bmatrix} \\ (x_1,y_1) \quad (x_2,y_2) \dots \quad (x_n,y_n) \end{array}$$

So, If we ~~cluster 2~~ wish to add 2 columns in a similar cluster, ~~its distan~~. both of its distances from all the other points will be nearly similar. for (x_1,y_1) and (x_2,y_2) to be in a same cluster, the distance of (x_3,y_3) $(x_4,y_4) \dots (x_n,y_n)$ will be similar from (x_1,y_1) and (x_2,y_2)

So, $d(x_3,x_1) \approx d(x_3,x_2)$
$\vdots \qquad \vdots$
$d(x_n,x_1) \approx d(x_n,x_2)$

and the distance between the points in the cluster will be less compared to the other points.

$d(x_1,x_2) = d(x_2,x_1) << d(x_i,x_1)$
$\sim, d(x_i,x_2)$

Where $i \neq 1,2$ in this case as we have assumed only 1 & 2 are in the same cluster, likewise we can find all the other points near to the cluster. that will follow the same criterion. like let pth point will be included ~~in~~ with 1 & 2 then. $d(P,1)$ and $d(P,2)$ will be small compared to $d(P,K)$ where $K \neq 1,2,P$.

and the distances of other points to P will be same as it is for. ~~d(P,~~ 1 and 2.

Be more clear 4

8. Consider a neural network that is layered and completely connected. Suppose we initialize two nodes n_1 and n_2 from the same layer with the same weights and biases. Then the final weights and biases ~~is~~ learnt will be the same.

We use chain rule to run backpropagation algorithm. Given an input, we execute the network from left to right to compute all outputs. If we initialize ~~same layers inputs to~~ two nodes of the same layer with same weights and biases, forward pass will execute the Neural Network and will produce the same outputs for the same weight and bias initialized nodes.

then we go backward and plug in all the values and get the derivatives and then take the derivatives and apply some δ and get the new weights.

In this case at $l = L$, δ_j^L

$$\frac{\partial c}{\partial z_j^L} = \frac{\partial c}{\partial a_j^L} \cdot \frac{\partial a_j^L}{\partial z_j^L}$$

let we have initialized. z_3 and z_5 same

then. $\frac{\partial c}{\partial z_3^L} = \frac{\partial c}{\partial a_3^l} \cdot \frac{\partial a_3^l}{\partial z_j^L}$

$$\frac{\partial c}{\partial z_5^L} = \frac{\partial c}{\partial a_5^l} \cdot \frac{\partial a_5^l}{\partial z_5^l}$$

so, $\frac{\partial c}{\partial z_3^L} = \frac{\partial c}{\partial z_5^L}$. as for that layer

$$\frac{\partial c}{\partial z_3^L} = 2(y_3 - a_3^L)(-1)$$

$$\frac{\partial c}{\partial z_5^l} = 2(y_5 - a_5^l)(-1)$$

$y_3 = y_5$ as the weights, ~~initialized~~ and biased are initialized same. and the computed a_3^l and a_5^l will be same. Also for each iteration for both the neurons updates will also be same.. ~~The update direction of~~ the learning will get disrupted because of these reasons and direction will be bounded ~~by~~ for each nuron by the others.

so, we need to initialize random weights and biases ~~so~~ to avoid this kind of disrupted learning

5