Neural Networks Report #3

Introduction

This homework has two problems, one is theory about Bayes classifier and its implementation in Perceptron and other is derivation of learning rules for multilayer perceptron. I will give either written solution for each part or some description of the code with brief analysis of the result. I again used Jupyter notebook for coding. In some of written solution, I used a code for visualization, it is saved in the cells of jupyter notebook just in case.

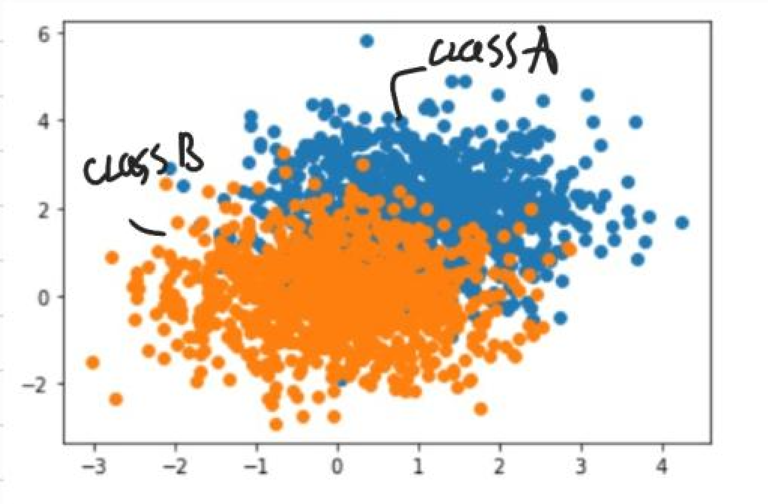
Note, I could not make the gaussian distribution by myself, so I generated data using the ready function gaussian(mean, std) that created a random number by this probability. As problems are connected, it is better to run them in order

Problem 1(a)

Here solution is straightforward and does not require any comments. It is below

Problem 1 (a)

Data is shown for good visualisation.



Let's start from general view

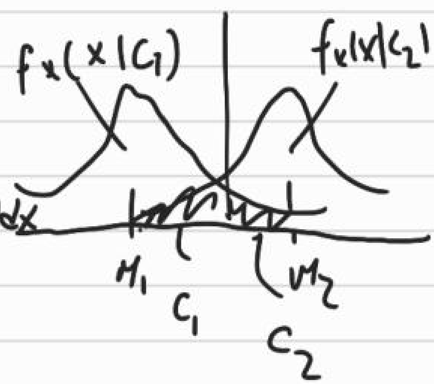
loss

$$R = c_{11}p_1 \int_{S_1} f_x(x|C_1)\,dx + c_{22}p_2 \int_{S_2} f_x(x|C_2)\,dx +$$

$$c_{21}p_1 \int_{S_2} f_x(x|C_1)\,dx + c_{12}p_2 \int_{S_1} f_x(x|C_2)\,dx$$

$S = S_1 \cup S_2$, $S_1 \cap S_2 = \emptyset$

Dec b



taking this we get

$$R = c_{21}p_1 + c_{22}p_2 + \int_{S_1} [p_2(c_{12}-c_{22})\, f_x(x|C_2)$$

$$- p_1(c_{21}-c_{11})\, f_x(x|C_1)]\,dx$$

since for point being in C_1 & C_2 has same cost

data is not biased

we take $c_{11}=c_{22}=0$, $c_{12}=c_{21}=c$

We use Gaussian f for classification as data is distributed so for data it doesn't has the same number for classes

so it has equal probability to be in either, so $P_1 = P_2 = 0.5$

$$R = \frac{c}{2} + \int_{S_1} \left[\frac{c}{2} f_x(x|C_2)\, dx - \frac{c}{2} f_x(x|C_1)\right] dx$$

$C_1 = A$

to minimize risk

$C_2 = B$

$$\frac{c}{2} f_x(x|C_1) > \frac{c}{2} f_x(x|C_2) \qquad x = \begin{bmatrix} x \\ y \end{bmatrix}$$

OR $\quad \Lambda(x) = \dfrac{f_x(x|C_1)}{f_x(x|C_2)} > 1$

$$\Lambda(x) \sim \exp\left(\frac{-1}{2}\left[\overbrace{(x-1)^2 + (y-2)^2}^{x^2-2x+1+y^2-4y+4} - x^2 - y^2\right]\right)$$

$$= \exp\left(\frac{-1}{2}(-2x - 4y + 5)\right)$$

for simplicity take log

$\log \Lambda(x) > 0$

$$\frac{-1}{2}(-2x - 4y + 5) > 0$$

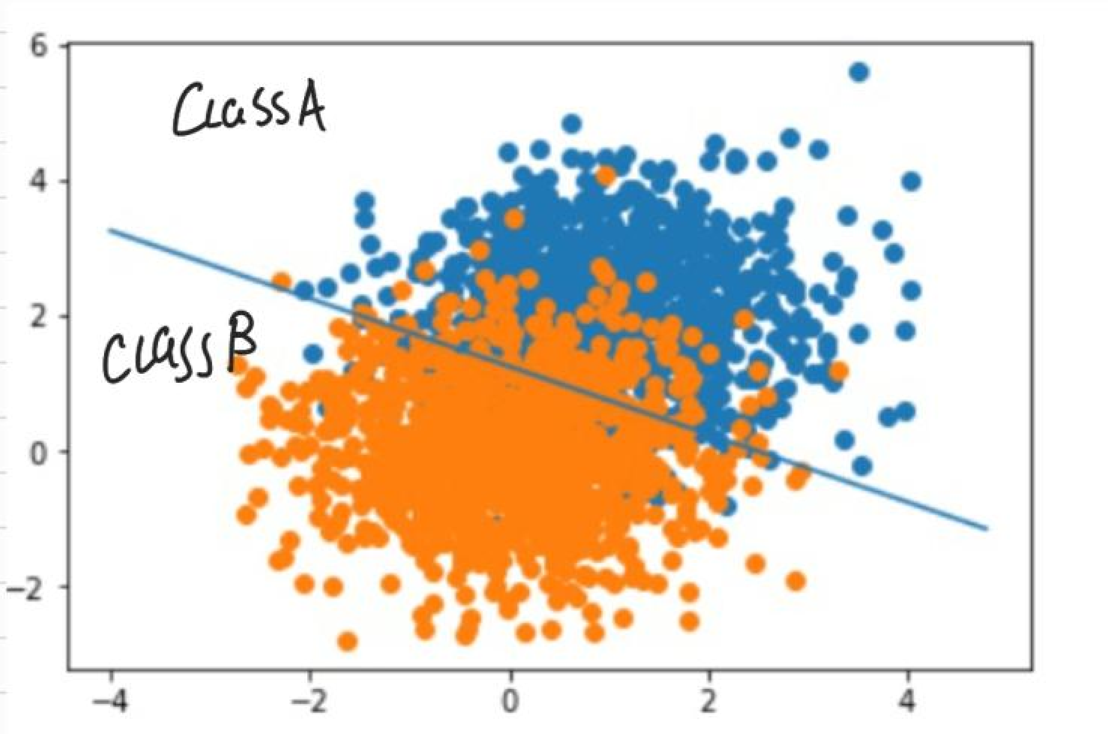
$$2x + 4y - 5 > 0$$

so to be in class 1 point coordinates are

$2x + 4y - 5 > 0$ & $2x + 4y - 5 < 0$ for C_2

line boundary is $2x+4y-5=0$

OR $y = \frac{5}{4} - \frac{x}{2}$ If we sketch it



Problem 1(b)

I firstly designed the functions that help to construct perceptron and that will be used in all other parts. The code is very simple, each weight is the coefficient of x coordinate, y coordinate and one extra bias. Then it uses algorithm that was shown in lecture notes.

Variables and Parameters:

$\mathbf{x}(n) = (m+1)$-by-1 input vector $= [+1, x_1(n), x_2(n), \ldots, x_m(n)]^T$

$\mathbf{w}(n) = (m+1)$-by-1 weight vector $= [b, w_1(n), w_2(n), \ldots, w_m(n)]^T$

b = bias

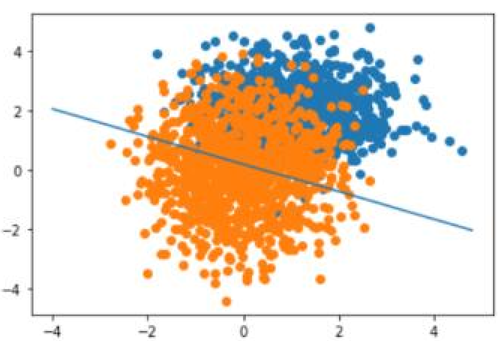
$y(n)$ = actual response (quantized)

$d(n)$ = desired response

η = learning-rate parameter, a positive constant

1. *Initialization.* Set $\mathbf{w}(0) = \mathbf{0}$. Then perform the following computations for time-step $n = 1, 2, \ldots$
2. *Activation.* At time-step n, activate the perceptron by applying input vector $\mathbf{x}(n)$ and desired response $d(n)$.
3. *Computation of Actual Response.* Compute the actual response of the perceptron as
$$y(n) = \text{sgn}[\mathbf{w}^T(n)\mathbf{x}(n)]$$
where sgn(·) is the signum function.
4. *Adaptation of Weight Vector.* Update the weight vector of the perceptron to obtain
$$\mathbf{w}(n+1) = \mathbf{w}(n) + \eta[d(n) - y(n)]\mathbf{x}(n)$$
where
$$d(n) = \begin{cases} +1 & \text{if } \mathbf{x}(n) \text{ belongs to class } \mathscr{C}_1 \\ -1 & \text{if } \mathbf{x}(n) \text{ belongs to class } \mathscr{C}_2 \end{cases}$$
5. *Continuation.* Increment time step n by one and go back to step 2.

I used very small learning rate and large amount of epoch so to avoid overfitting. To visualize I constructed the loss function and its behavior is strange. Possibly it is due to its sensitivity to small errors since the perceptron gives fixed value of result, either 1 or -1 so it gives big loss for each error. That's why the rate parameter should be very small. Result is satisfiable but still not so certain, the reason can be as data distribution is still chaotic.



Problem 1(c)

problem 1 (c)

We repeat the same steps but now with different fuctions for Likelyhood

$$R = C_{21}P_1 + C_{22}P_2 + \int_{S_1} [P_2(C_{12}-C_{22}) f_X(X|C_2)\,dX - P_1(C_{21}-C_{11}) \cdot f_X(X|C_1)]\,dX$$

It is the same cost for a point to be in either class : $C_{11} = C_{22} = 0$.

For making mistake it will have same risk as : $C_{12} = C_{21} = C$

As there is the same # of points in both classes, for a point it has same probability to be in any : $P_1 = P_2 = 0.5$

$$R = \frac{C}{2} + \int_{S_1} \frac{C}{2} f_X(X|C_2)\,dX - \frac{C}{2} f_X(X|C_1)]\,dX$$

to minimise

$$f_X(X|C_2)\,dX - f_X(X|C_1) < 0$$

4

$$\Lambda(x) = \frac{f_x(x|C_1)}{f_x(x|C_2)} > 1$$

for simplicity $\log \Lambda(x) > 0$

$f_x(x|C_1) \propto \exp[-((x-1)^2 + (y-2)^2)/2\sigma_1^2]$

$f_x(x|C_2) \propto \exp[-(y^2+x^2)/2\sigma_2^2]$

$\sigma_1 = 1 \quad \sigma_2 = \sqrt{2}$

$$\log \Lambda(x) = -\left(\frac{(x-1)^2 + (y-2)^2}{2} - \frac{y^2}{4} - \frac{x^2}{4}\right) > 0$$

$2(x^2 - 2x + 1 + y^2 - 4y + 4) - y^2 - x^2 < 0$

$x^2 + y^2 - 4x - 8y + 10 < 0$

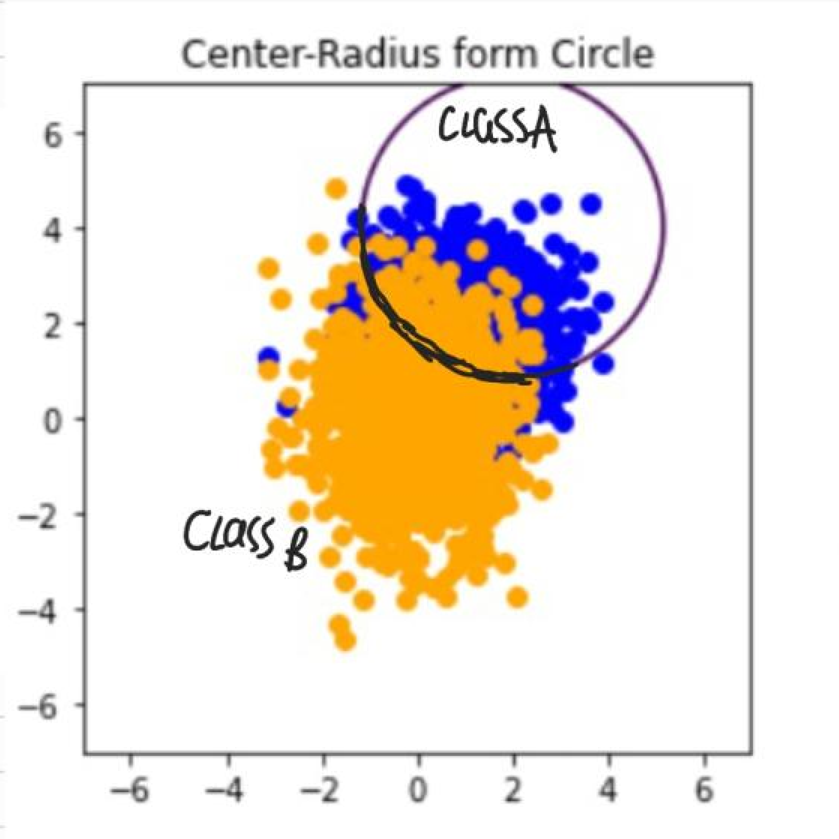
$y^2 - \overset{2 \cdot 4}{8}y + 16 - 6 + x^2 - 4x < 0$

$(y-4)^2 < 6 + 4x - x^2 - 4 + 4$

decision boundary is $(y-4)^2 = 10 - (x-2)^2$

If point is above, it is in class 1 if below at class 2

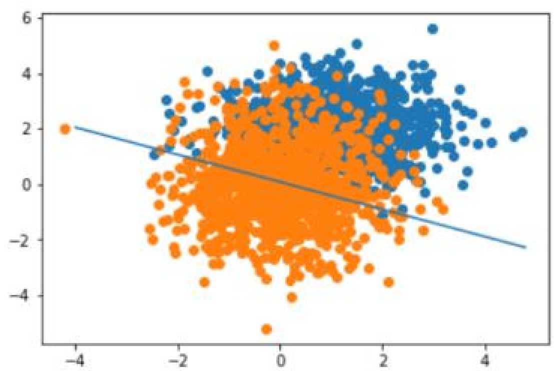
If we sketch, we get circular boundary


Center-Radius form Circle
CLASSA
Class B
6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6

Note that the result is not linear boundary which is fine as the Bayes classifier can produce such behavior

Problem 1(d)

This code is copy of code from part b except some parameters and data are changed. Since perceptron can give only linear result, it is different from the Bayes classifier solution.



Problem 1(e)

Problem 1(e)

Now the data probability to be in either class is different. $N_1 = 700$, $N_2 = 1400$

then $p_1 = \frac{700}{1400+700} = \frac{1}{3}$ $p_2 = 1 - p_1 = \frac{2}{3}$

$$R = \frac{c}{3} + \int_{S_1} \left[\frac{2c}{3} f_X(x|C_2)\,dx - \frac{c}{3} f_X(x|C_1) \right] dx$$

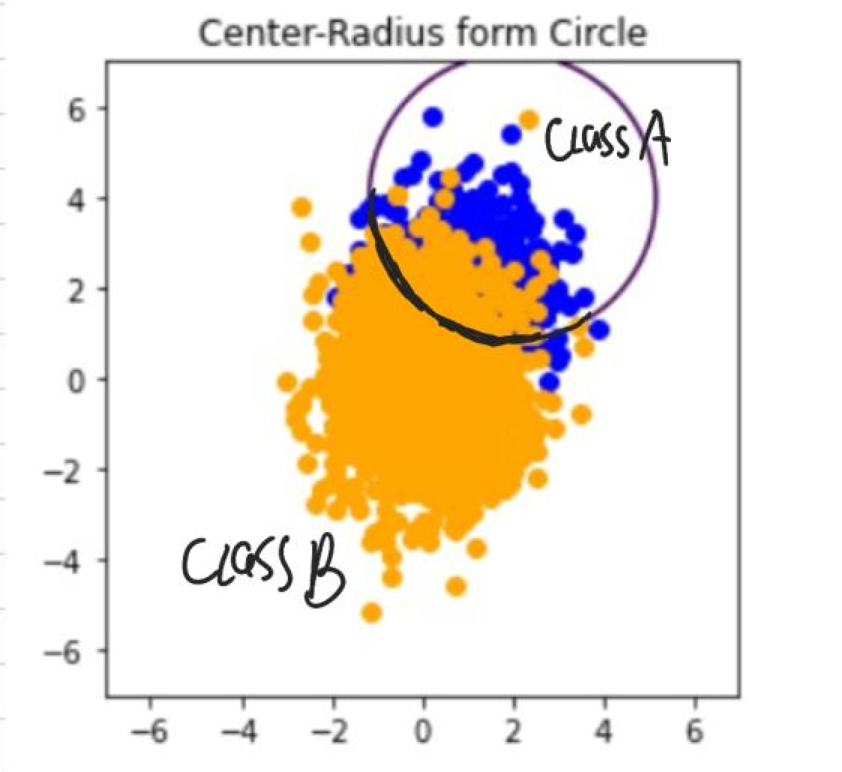
$$\Lambda(X) = \frac{f_X(X|C_1)}{f_X(X|C_2)} > 2$$

$\ln \Lambda(X) > \emptyset$

as it is the same condition as for (c) we again get decision boundary as

$$(y-4)^2 = 10 - (x-2)^2$$

And the sketch is:

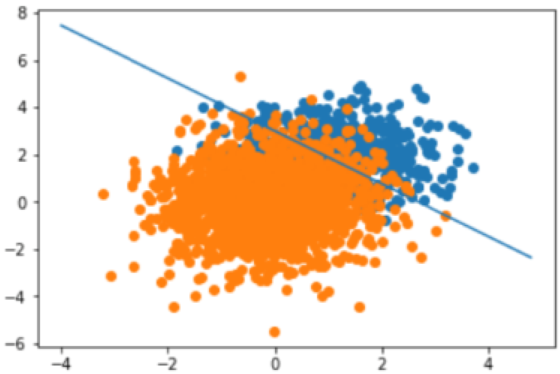


Center-Radius form Circle
Class A
Class B
6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6

Interesting to note that even with different number of data points and different probability to be in classes, Bayes classifier based on Gaussian function produces the same result as before and logarithm of constant is zero. Probably it is not sensitive to such cases in this situation

Problem 1(f)

Here the different number of points in each class made some change, perceptron is sensitive to it, and as before it could only produce a linear function.



Problem 1(g)(a)

Problem 1(g)

$p_1(x,y) = K_1$ for $(x-1)^2+(y-2)^2 < 2\sigma_1^2$

$p_2(x,y) = K_2$ for $x^2+y^2 < 2\sigma_2^2$

Problem 1(g)(a)

this distribution is similar to Homework 1 as it is constant & within a circle.

For this, we need a functions similar to distributions i.e.

$$f(x|C_1) = \begin{cases} K_1 & \text{if } (x-1)^2+(y-2)^2 < 2 \\ 0 & \text{oth} \end{cases}$$

$$f(x|C_2) = \begin{cases} K_2 & \text{if } x^2+y^2 < 2 \\ 0 & \text{otw} \end{cases}$$

as $N_1 = N_2 \rightarrow p_1 = p_2 = 0.5$ $C_{11} = C_{22} = 0$ $C_{12} = C_{21} = C$

$$R = \frac{C}{2} + \int_{S_1} \frac{C}{2} f_x(x|C_2)\,dx - \frac{C}{2} f_x(x|C_1)]\,dx$$

$$\Lambda(x) = \frac{f_x(x|C_1)}{f_x(x|C_2)} > 1$$

So we need conditions

But before, let's find K_1 & K_2 as

$\int p \, dS = 1 \rightarrow K \int dS = K \cdot S = 1$

$K = \frac{1}{S}$ $\quad K_1 = K_2 = \frac{1}{2\pi}$

so

$$\Lambda(x) = \begin{cases} 0 & \text{if } (x-1)^2 + (y-2)^2 > 2 \text{ only} \quad ① \\ \infty & \text{if } x^2+y^2 < 2 \text{ only} \quad ② \\ 1 & \text{if } (x-1)^2+(y-2)^2 < 2 \text{ \& } x^2+y^2 < 2 \quad ③ \end{cases}$$

We need to connect this function

let $a_1 = (x-1)^2 + (y-2)^2 - 2 = x^2 + y^2 - 2x - 4y + 3$

$a_2 = x^2 + y^2 - 2$

so $a_1 = a_2 - 2x - 4y + 5$

$$\Lambda(x) = \begin{cases} 0 & \text{if } a_1 \geq 0 \text{ \& } a_2 \geq 0 \\ \infty & \text{if } a_2 \geq 0 \text{ \& } a_1 < 0 \\ 1 & \text{if } a_1 < 0 \text{ \& } a_2 < 0 \end{cases}$$

We can make it with only one condition

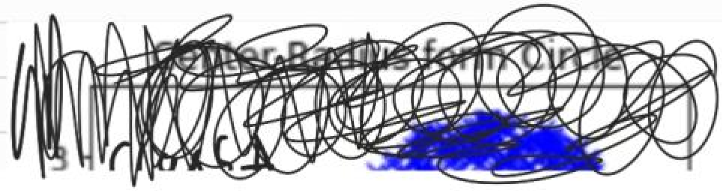
you see that decision boundary is at $a_1 = a_2$

OR $\quad$ OR

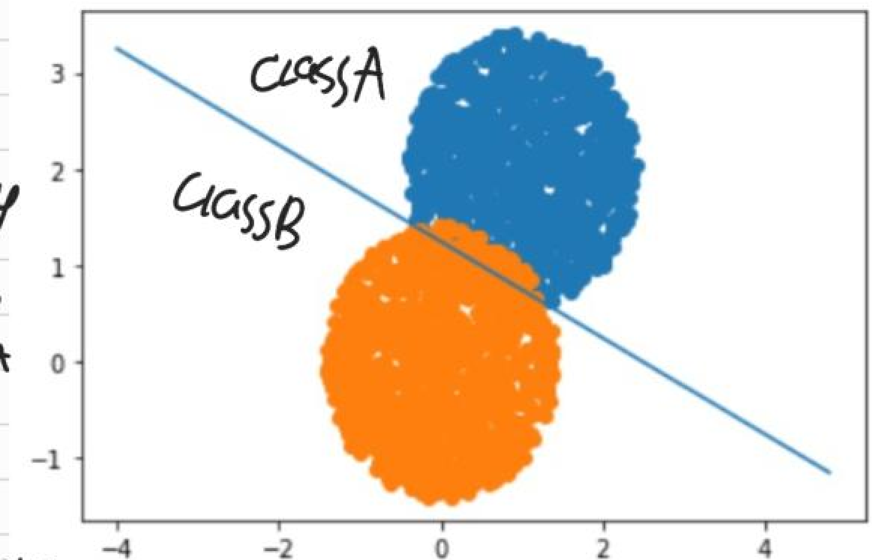
$-2x - 4y + 5 = 0$ $\qquad a_1 - a_2 < 0$

in C_1 if $y > \frac{5}{4} - \frac{x}{2}$ $\quad (a_1 < 0)$

in C_2 if $y \leq \frac{5}{4} - \frac{x}{2}$



Note that we can make another boundary by using the same $\Lambda(x)$, but this one fits all the parameters,



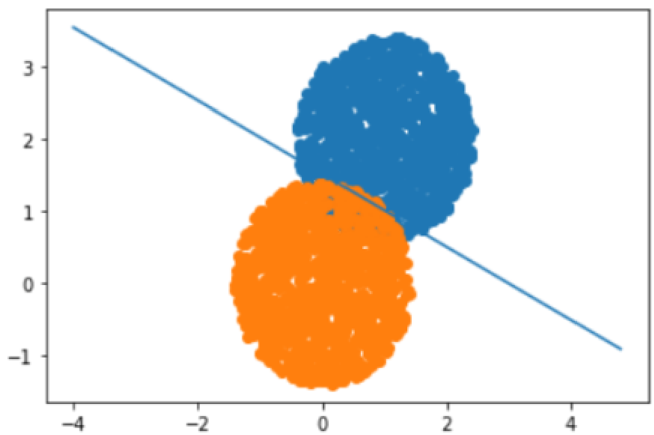
so it is more trivial case & that's why should be preferred.

Linear is prefered is the risk is still minimized.

Different distribution, but in similar space classes gives the same result. That's is interesting

Problem 1(g)(b)

Again, code is slightly adjusted from previous parts. This time, the loss converged to minimum and didn't show any strange behavior, since the data is distributed uniformly, so lower the noise which creates such errors. Result is divided perfectly and agrees with Bayes classifier



Problem 1(g)(c)

Problem 1(g)(c)

Now we have

$p_1(x,y) = k_1$ if $(x-1)^2+(y-2)^2<2$

$p_2(x,y) = k_2$ If $x^2+y^2<4$

Choosing same functions & same conditions as in part (a) we get

$R = \dots \int \cdots (f(x|C_1) - f(x|C_2) \cdots$

$$\Lambda(x) = \frac{f(x|C_1)}{f(x|C_2)} > 1$$

$$f(x|C_1) = \begin{cases} k_1 & \text{if } (x-1)^2+(y-2)^2<2 \\ 0 & \text{oth} \end{cases}$$

$$f(x|C_2) = \begin{cases} k_2 & \text{if } x^2+y^2<4 \\ 0 & \text{oth} \end{cases}$$

Let's find k_1 & k_2

$k = \frac{1}{\pi r^2} \Rightarrow k_1 = \frac{1}{2\pi} \quad k_2 = \frac{1}{4\pi}$

Let $a_1 = (x-1)^2+(y-2)^2-2$
$a_2 = x^2+y^2-4$

$$\Lambda(X) = \begin{cases} 0 & a_1 > 0 \\ \infty & a_1 < 0 \ \& \ a_2 < 0 \\ 2 & a_1 < 0 \ \& \ a_2 > 0 \end{cases}$$

$a_1 = x^2 + y^2 - 2x - 4y + 5 - 2 = a_2 - 2x - 4y + 7$

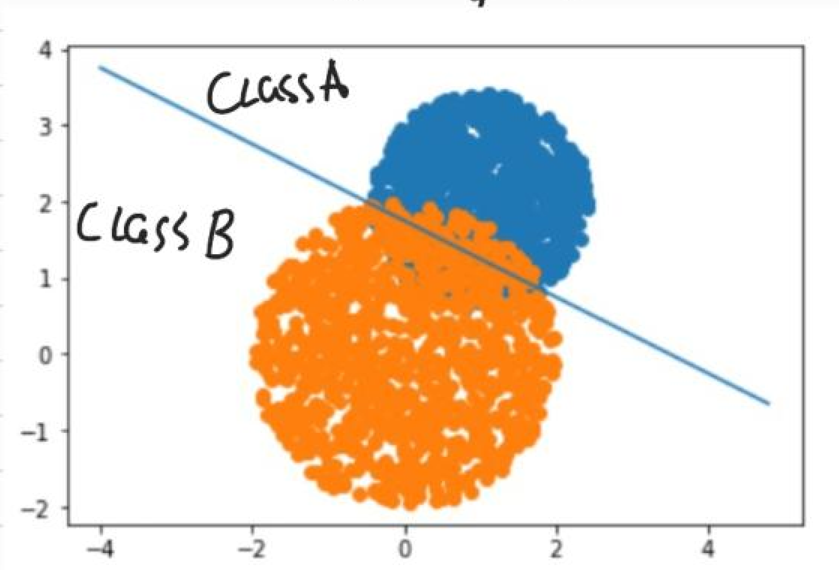
for $\Lambda(x) > 1$ $\ a_1 - a_2 < 0 \rightarrow$ simpler to satisfy

so decision boundary is $\swarrow$ difference

$4y + 2x - 7 = 0$ $\nearrow$ $y \geq 7/4 - x/2$ is in C_1

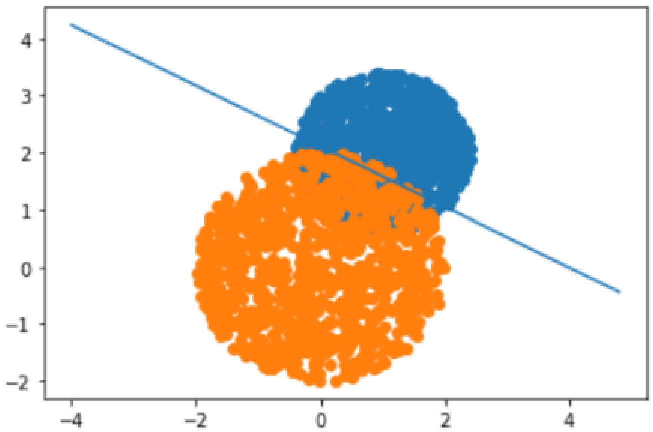
$\searrow$ $y < 7/4 - x/2$ is in C_2

sketching



Problem 1(g)(d)

This is similar as part a, but Loss is noisier as class B is more chaotic, but result still agrees with Bayes prediction and has a good performance.



Problem 1(g)(e)

Problem 1 (g) (e)

By same reasoning as in (e) $P_1 = \frac{1}{3}$, $P_2 = \frac{2}{3}$

$\Lambda(x) > \xi$

$\frac{f(x|C_1)}{f(x|C_2)} > 2$ Let $a_1 = (x-1)^2+(y-2)^2-2$
$a_2 = x^2+y^2-4$

$$\Lambda(X) = \begin{cases} 0 & a_1 > 0 \\ \infty & a_1 < 0 \ \& \ a_2 < 0 \\ 2 & a_1 < 0 \ \& \ a_2 > 0 \end{cases}$$

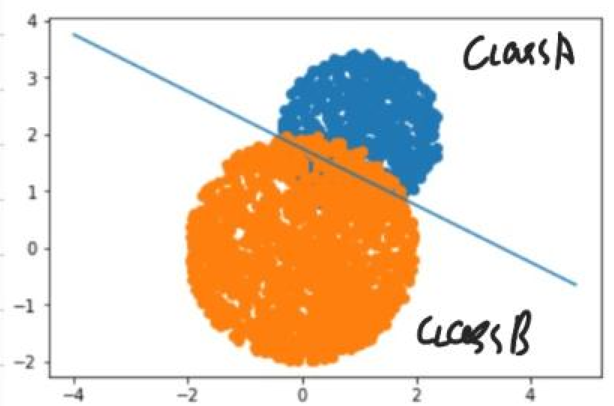
So $a_1 < 0$ & $a_2 < 0 \rightarrow a_1 - a_2 < 0$

We again get boundary at $4y+2x-7=0$

$y > -\frac{x}{2} + \frac{7}{4} \rightarrow$ class A

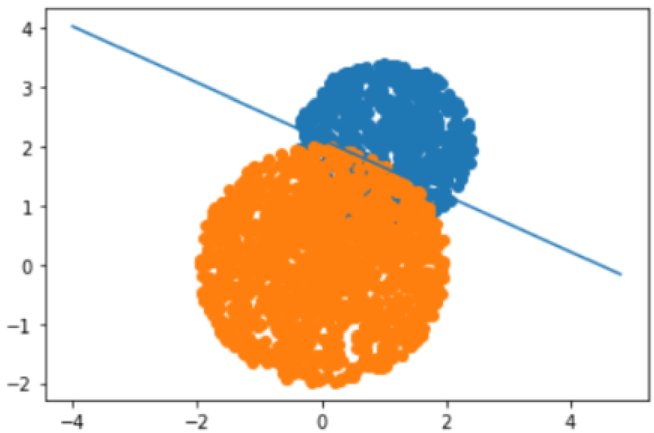
$y \leq \frac{7}{4} - \frac{x}{2} \rightarrow$ class B

the difference from previous is equality factor!



Problem 1(g)(f)

This time result is more uncertain and does not agree with Bayes prediction much as the perceptron is sensitive to point number and very sensitive to the "noise" produced from B class distribution.



Analysis of Part (g)

As in g part the data is distributed without huge variety, i.e. in one fixed space, perceptron easily learns how t solve the task in all cases unlike what happened in gaussian distribution case where the widely positioned data caused a lot of confusion, so the loss contained noise and could not converge on the graph. Note that gaussian and uniform distribution in circle region are very similar in visualization, only gaussian is distributed more chaotically. G part also has some fluctuations around the minimum in loss graph, but they are very small compared to gauss and can be treated naturally.

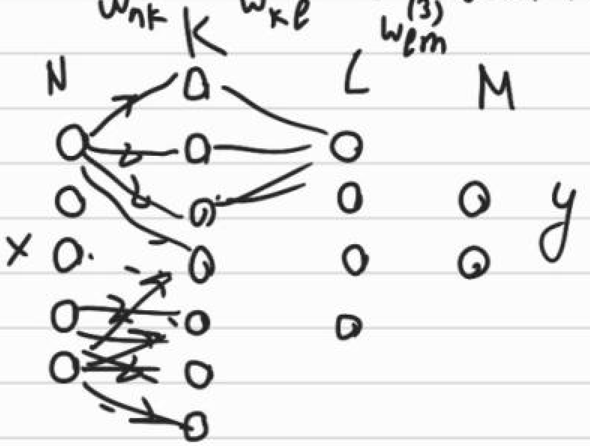
Bayesian part was also different, in other parts gaussian function were used and it gave non-linear result in two last cases as Bayesian can give such results, perceptron however can only work with linear. In part g it was more complex, there was a choice in conditions how to combine in likelihood function, eventually I choose the simple case to avoid circle in the border, so the theoretical result became simple line. If I choose another condition it could be simple circle around one distribution and I think it is not that fair as this requires the knowledge of data before distribution, so quite different from machine learning concepts. Perceptron here, made a good agreement with Bayes result and they didn't have any huge variations.

Conclusion

This problem showed how to construct perceptron, use Bates classifier and their difference and opportunities of each. Different distributions explain many details of concepts and show importance of good parameters.

Problem2

Problem 2



Let's say the function use is f It can be ReLU, tanh etc

$$x \to W^{(1)} \to h^{(1)} \to W^{(2)} \to h^{(2)} \to W^{(3)} \to y$$

From forward propagation we have

$$h_k^{(1)} = f\left(\sum_{n=0}^{N} W_{nk}^{(1)} x_n\right) = f(\hat{h}_k^{(1)})$$

$$h_l^{(2)} = f\left(\sum_{k=0}^{K} w_{kl}^{(2)} h_k^{(1)}\right) = f(\hat{h}_l^{(2)})$$

$$y_m = f\left(\sum_{l=0}^{L} w_{lm}^{(3)} h_l^{(2)}\right) = f(\hat{y}_m)$$

In backpropogation we start from y & go back to the beginning

a) Chain rule

Let's say at some iteration network has loss $\mathcal{L}_e$ then to update each weigh we use gradient descent

$$w_{ij}^{(s)}(n+1) = w_{ij}^{(s)}(n) - \frac{\partial \mathcal{L}_e}{\partial w_{ij}^{(s)}} \cdot \eta$$

learning rate

However $\mathcal{L}_e$ is function of $y \to \mathcal{L}_e(y)$
Using chain rule

$$\frac{\partial \mathcal{L}_e}{\partial w_{lm}^{(3)}} = \sum \frac{\partial \mathcal{L}_e}{\partial y_m} \cdot \frac{\partial y_m}{\partial w_{lm}^{(3)}}$$

Let's use a mean square error

$$\mathcal{L}_e = \frac{1}{2} \sum_{1}^{N} (d^i - y^i)^2 = \frac{1}{2} \sum_{1}^{S} \sum_{k=1}^{L} (d_k^s - y_k^s)^2$$

where S is total number of data points

$$\frac{\partial \mathcal{L}_e}{\partial y_m} = \frac{2}{2} \sum_{1}^{S} \sum_{m=1}^{M} -(d_m^s - y_m^s) = -\sum_{s=1}^{S} (d_M^s - y_m^s)$$

$$\frac{\partial y_m}{\partial w_{lm}^{(3)}} = f'(\hat{y}_m) \cdot \frac{\partial \hat{y}_m}{\partial w_{lm}} = f'(y_m) \cdot h_l^{(2)}$$

so

$$\frac{\partial L_e}{\partial W_{\ell m}^{(3)}} = -\sum_{s=1}^{S} \delta_\ell^{(3)s} h_\ell^{(2)s} \quad \delta_\ell^{(3)s} \equiv f'(\hat{y}_\ell^s)(d_i^s - y_\ell^s)$$

similary

sum because we need for every neuron, multidim

$$\frac{\partial L}{\partial W_{k\ell}^{(2)}} = \sum_m \frac{\partial L_e}{\partial \hat{y}_m} \cdot \frac{\partial \hat{y}_m}{\partial W_{k\ell}^{(2)}}$$

$$\frac{\partial L_e}{\partial \hat{y}_m} = \frac{\partial L_e}{\partial y_m} \cdot \frac{\partial y_m}{\partial \hat{y}_m} = \left(-\sum_s d_m^s - y_m^s\right) \cdot f'(\hat{y}_m^s)$$

for 1 data point (ignore s)

$$\frac{\partial \hat{y}_m}{\partial w_{k\ell}^{(2)}} = \frac{\partial}{\partial w_{k\ell}^{(2)}}\left(\sum_\ell w_{\ell m}^{(3)} h_\ell^{(2)}\right) = w_{\ell m}^{(3)} \frac{\partial h_\ell^{(2)}}{\partial w_{k\ell}^{(2)}}$$

$$= w_{\ell m}^{(3)} \cdot f'(\hat{h}_\ell^{(2)}) \cdot \frac{\partial}{\partial w_{k\ell}^{(2)}}\left(\sum w_{k\ell}^{(2)} \cdot h_1^{(k)}\right) \quad \to h_1^{(k)}$$

so

$$\frac{\partial L_e}{\partial w_{k\ell}^{(2)}} = \sum_m \underbrace{\left(-\sum_s d_m^s - y_m^s\right) \cdot f'(\hat{y}_m^s)}_{\delta_m^{(3)}} \cdot w_{\ell m}^{(3)} \cdot f'(\hat{h}_\ell^{(2)}) \cdot h_1^{(k)}$$

N K L M

separating terms

$$\frac{\partial L_e}{\partial W_{k\ell}^{(2)}} = -\sum_{s=1}^{S} \delta_k^{(2)s} h_k^{(1)s}$$

$$\delta_\ell^{2,s} = f'(\hat{h}_\ell^{(2)s}) \sum_{m=1}^{M} \delta_m^{(3)} W_{m\ell}^{(3)}$$

To update $w^{(1)}$ we make similar step

back

$$\frac{\partial \mathcal{L}_0}{\partial w_{nk}^{(1)}} = \sum_{s=1}^{S} \sum_{\ell}^{L} \frac{\partial \mathcal{L}_0}{\partial \hat{h}_\ell^{(2)s}} \cdot \frac{\partial \hat{h}_\ell^{(2)}}{\partial w_{nk}^{(1)}}$$ summed over all ℓ to get rid of ℓ (chain rule)

where $\frac{\partial \mathcal{L}_0}{\partial \hat{h}_\ell^{(2)}}$ is continuation of previous

$$\frac{\partial \mathcal{L}_0}{\partial w_{k\ell}^{(2)}} = -\sum_{s=1}^{S} \delta_k^{(2)s} h_k^{(1)s} = -\sum \delta_k^{(2)s} f(\hat{h}_k^{(1)s})$$

to includ all demonsions

$$\frac{\partial \mathcal{L}_0}{\partial \hat{h}_\ell^{(2)}} = \sum_m \frac{\partial \mathcal{L}_0}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial \hat{h}_\ell^{(2)}} = \sum \left(-\sum_s d_m^s - y_m^s\right) \cdot f'(\hat{y}_m^s) \cdot w_{\ell m}^{(3)}$$

h_k^1

$$\frac{\partial \hat{h}_\ell^{(2)}}{\partial w_{nk}^{(1)}} = f'(h_\ell^{(2)}) \cdot w_{k\ell}^{(2)} \cdot \frac{\partial}{\partial w_{nk}^{(1)}}\left(\sum w_{nk}^{(1)} x_n\right) \cdot f'(\hat{h}_k^{(1)})$$

$$= f'(h_\ell^{(2)}) \cdot w_{k\ell}^{(2)} \cdot x_n$$

Putting all together $\quad M \; K \; \ell$

$$\mathcal{L}_0 = \sum_s \sum_\ell \sum_m -(d_m^s - y_m^s) \cdot f'(\hat{y}_m) \, w_{\ell m}^{(3)} \, f'(h_\ell^{(2)}) w_{k\ell}^{(2)} \, x_n \, f'(h_k^{(1)})$$

Separating terms we get

$$\frac{\partial \mathcal{L}_0}{\partial w_{mk}^{(1)}} = -\sum_{s=1}^{S} \delta_k^{(3)s} x_n \;, \quad \delta_k^{(3)s} = f'(h_k^{(1)s}) \sum_{\ell=1}^{L} \delta_\ell^{(2)s} w_{\ell k}^{(2)}$$

b) Let's assume that we slightly changes $w_{\ell m}^{(3)}$ by $\Delta w_{\ell m}^{(3)}$

only one weight is changes

as $y_m = \sum_{\ell=1}^{L} w_{\ell m} h_\ell^{(2)} \rightarrow \Delta \hat{y}_m = \Delta w_{\ell m}^{(3)} h_\ell^{(2)}$

as $L = \frac{1}{2}\sum_s \sum_m (d_m^s - y_m^s)^2$ so $\Delta y_m = f(\Delta \hat{y}_m) = f(\hat{y}_m)\Delta y_m$

if we take the small change on Δy_m

$$\Delta L = \frac{1}{2}\sum_s (d_m^s - y_m^s - \Delta y_m^s)^2$$

$$= \frac{1}{2}\sum_s (d_m^s - y_m^s)^2 \left(1 - \frac{\Delta y_m^s}{d_m^s - y_m^s}\right)^2 \approx \frac{\cancel{2}}{\cancel{2}}\sum_s (d_m^s - y_m^s)\Delta y_m^s$$

so $\Delta L = \sum_s (d_m^s - y_m^s)\Delta y_m^s$

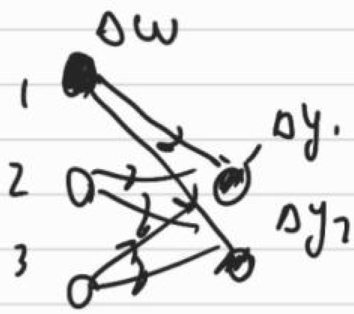
then

$$\Delta L = \underbrace{\sum_s}_{\text{for all data points}} (d_m^s - y_m^s) \cdot h_e^{(2)s} \Delta w_{em}^{(3)} f'(\hat{y}_m^s)$$

$\frac{\Delta L}{\Delta w_{em}^{(3)}} = \sum_s \delta_m^{(3)s} h_e^{(2)s}$ $\delta_m^{(3)} = f'(\hat{y}_m)(d_m^s - y_m^s)$

So if we wanna make loss smaller by ΔL we update weight 3 by same amount

Now assume we changed $w_{ke}^{(2)}$ slightly by $\Delta w_{ke}^{(2)}$. then $\hat{h}_l^{(2)}$ get's changed & subsequent parts due to sense



so $\Delta \hat{h}_e^{(2)} = \Delta w_{ke} \cdot h_k^{(1)}$

$\Delta h_e^{(2)} = f(\Delta \hat{h}_e^{(2)}) = f'(h^{(2)}) \cdot \Delta w_{ke} h_k^{(1)}$

So $\Delta \hat{y} = \sum_m W_{em} \cdot \Delta h_\ell$ as Δ is small (N K L M

← as m nerons go from h_ℓ

$\Delta y = f(\Delta \hat{y}) = f(\hat{y} + \Delta \hat{y}) - f(\Delta y) = f'(\hat{y}) \Delta y$

since $f'(y) = \lim_{\Delta y \to 0} \frac{f(\hat{y} + \Delta \hat{y}) - f(\hat{y})}{\Delta y}$

$\Delta h_e = \sum_s (d_m^s - y_m^s) \Delta y = \sum_s \sum_m f'(\hat{y}) \cdot w_{em} \cdot h_k^{(1)} \cdot \Delta w_{k\ell}^{(1)}$

Introducing constants:

N K L M

$\frac{\Delta R_0}{\Delta w_{ke}^{(2)}} = -\sum_{s=1}^{S} \delta_k^{(2)s} h_k^{(1)s}$

$\delta_\ell^{2,s} = f'(\hat{h}_\ell^{(2)s}) \sum_{m=1}^{M} \delta_m^{(3)} W_{em}^{(3)}$

of one neuron

Next, let's slightly change w_{nk}^{W} & see how

h_e is sensitive to $\Delta w_{k\ell}^{(1)}$

$\Delta \hat{h}_k^{(1)} = \Delta w_{nk} x_n$

$\Delta h_k^{(1)} = f(\Delta \hat{h}_1) = f'(\hat{h}_1) \cdot \Delta \hat{h}_1$

$\Delta \hat{h}_\ell^{(2)} = \sum_\ell w_{k\ell} \Delta \hat{h}_k^{(1)}$ as all one neuron k gives

change to ℓ neurons in next layer

Then repeating previous steps

$\Delta\hat{y} = \sum_m w_{\ell m} \Delta\hat{h}_\ell^{(2)}$

$y = f'(\hat{y})\, \Delta\hat{y}$

$\Delta\mathcal{L}_e = \sum_s -(d_m^s - y_m^s)\, \Delta y$ Going in reverse

$= \sum_s - \sum_m \sum_\ell (d_m^s - y_m^s)\, f'(\hat{y}) \cdot w_{\ell m} \cdot w_{k\ell}\, f'(\hat{h}_1) \cdot x_n\, \Delta w_{nk}$

$\dfrac{\Delta\mathcal{L}_e}{\Delta w_{nk}^{(1)}} = -\sum_{s=1}^{S} \delta_k^{(3)s} x_n \,, \quad \delta_k^{(3)s} = f'(h_k^{(1)s}) \sum_{\ell=1}^{L} \delta_L^{(2)s} w_{\ell k}^{(2)}$

So taking limit for Δw so $\Delta w \to 0$ we get derivative $\dfrac{\partial\mathcal{L}_e}{\partial w^{()}}$ & update each weight by using GD. $\dfrac{\Delta\mathcal{L}_e}{\Delta w} \to \dfrac{\partial\mathcal{L}_e}{\partial w}$