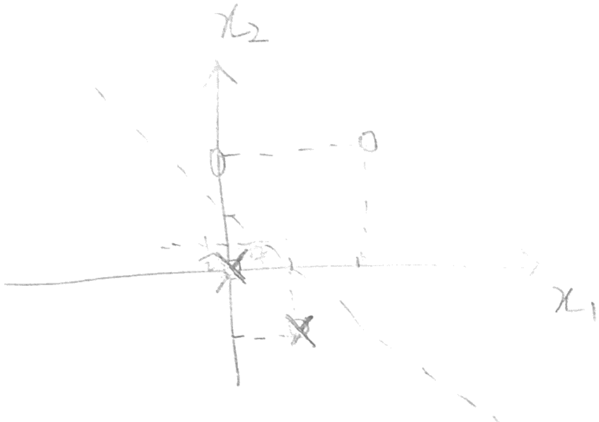
1. Maximum Margin Classifier and Support Vector Machine

(a) Yes, it is possible to use a linear function $x_{i1}+x_{i2}+b=0$ to prefectly separate the two classes for some parameter b.



Since there are two point $(0,0)$ and $(0,2)$,

then exist a linear function

$x_{i1}+x_{i2}+b=0$ pass $(0,1)$

Hence, $0+1+b=0 \Rightarrow b=-1$

And the maximum margin is

$\frac{\sqrt{2}}{2}\times 1=\boxed{\frac{\sqrt{2}}{2}}$.

(b)

$$\hat{y} = \begin{cases} 1 & \text{if } z > 0 \\ -1 & \text{if } z \leq 0 \end{cases}, \quad z = \sum_{i=1}^{N} \alpha_i y_i K(x, x_i) + b$$
$$= \alpha_i y_i + b$$

$$K(x, x_i) := \begin{cases} 1 & \text{if } \|x - x_i\|^2 \leq r^2 \\ 0 & \text{if } \|x - x_i\|^2 > r^2 \end{cases}$$

Assume $r = 0.5$, $b = -0.5$, for each sample, then

$$z_1 = -\alpha_1 - 0.5 \leq 0$$

$$z_2 = \alpha_2 - 0.5 > 0$$

$$z_3 = -\alpha_3 - 0.5 \leq 0$$

$$z_4 = \alpha_4 - 0.5 > 0$$

then we set $\alpha_1 = 0$, $\alpha_2 = 1$, $\alpha_3 = 0$, $\alpha_4 = 1$

Hence, $\alpha = [0, 1, 0, 1]$, $b = -0.5$ and $r = 0.5$ to make the classifier no errors on the training data.

2. $\boxed{\text{Neural Network}}$

(a) $$z^H = \sum_{i=1}^{N} W^H x_i + b^H$$

$$= \begin{bmatrix} x_1 \times 0 + x_2 \times 1 - 2 \\ x_1 \times 1 + x_2 \times 1 - 1 \\ x_1 \times (-1) + x_2 \times 1 + 1 \end{bmatrix}$$

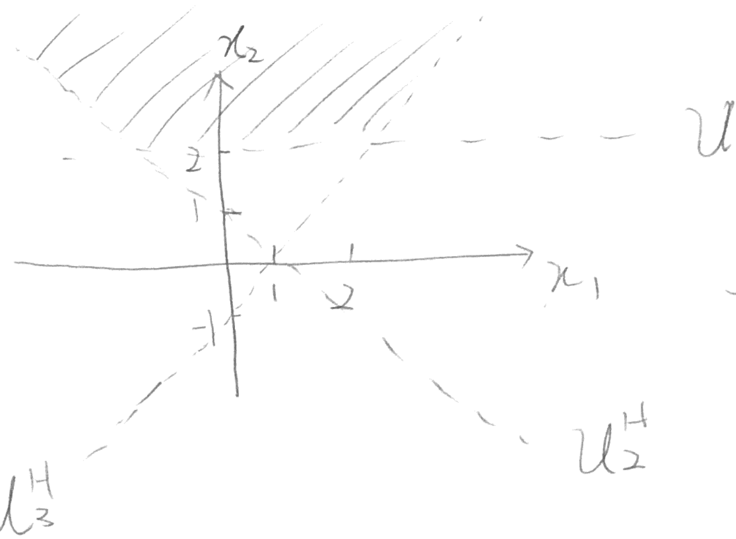
$$= \begin{bmatrix} x_2 - 2 \\ x_1 + x_2 - 1 \\ -x_1 + x_2 + 1 \end{bmatrix}$$

Hence,

$$u_1^H = 1 \iff x_2 > 2$$

$$u_2^H = 1 \iff x_1 + x_2 > 1 \iff x_2 > 1 - x_1$$

$$u_3^H = 1 \iff -x_1 + x_2 > -1 \iff x_2 > x_1 - 1$$



The region is shown above.

(b)

i	1	2	3	4
x_{i1}	0	2	1	1
x_{i2}	0	0	3	$\frac{1}{2}$
u_1^H	-1	-1	1	-1
u_2^H	-1	1	1	1
u_3^H	1	-1	1	1
$\hat{y}_i$	0	0	0	1

From the table we know that:

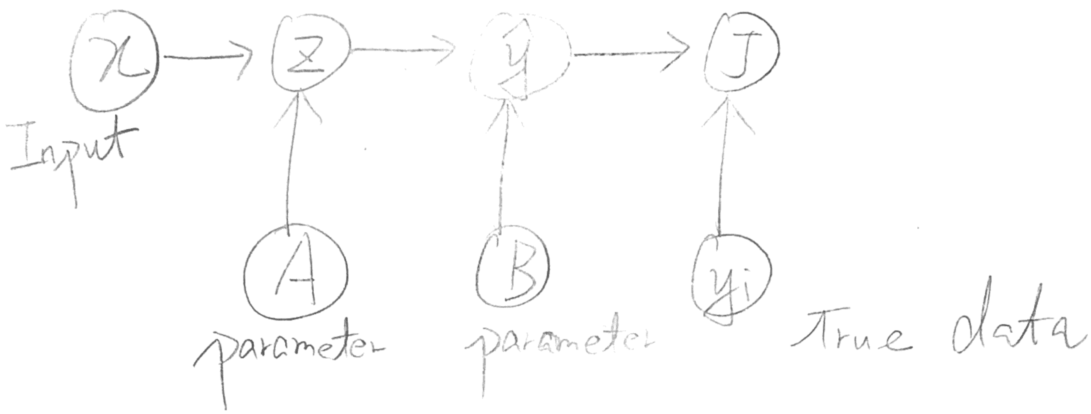
$$\begin{cases} -w_1^o - w_2^o + w_3^o + b^o \leq 0 \\ -w_1^o + w_2^o - w_3^o + b^o \leq 0 \\ w_1^o + w_2^o + w_3^o + b^o \leq 0 \\ -w_1^o + w_2^o + w_3^o + b^o > 0 \end{cases}$$

Then, we would get $W^o = [-1, 1, 1]$,

and $b^o = -2.5$

3. Backpropagation

(a)



(b)

$$\frac{\partial J}{\partial A_{il}} = \frac{\partial J}{\partial z_{il}} \cdot \frac{\partial z_{il}}{\partial A_{il}}, \quad z_{il} = \sum_{j=1}^{D} x_{ij} A_{jl}$$

$$\frac{\partial z_{il}}{\partial A_{il}} = x_{ij},$$

$$\Rightarrow \frac{\partial J}{\partial A_{il}} = \frac{\partial J}{\partial z_{il}} \times x_{ij}$$

(c)

$$\frac{\partial J}{\partial B_l} = \sum_i \frac{\partial J}{\partial \hat{y}_i} \cdot \frac{\partial \hat{y}_i}{\partial B_l}, \quad \frac{\partial \hat{y}}{\partial B_l} = e^{B_l z_l} \cdot z_l$$

$$\frac{\partial J}{\partial \hat{y}} = 2(y_i - \hat{y}_i) \cdot (-1) = -2(y_i - \hat{y}_i)$$

5.

1

$$\boxed{\frac{\partial J}{\partial B_1} = \sum_{i=1}^{N} 2(\hat{y}_i - y_i) \cdot z_i \cdot e^{B_l z_l}}$$

4. (a) $\boxed{\text{Convolutional ~~net~~ neural network}}$

For Conv1 layer, we will have

Input: (samples, x pixels, y pixels, channels)

$= (100, 256, 256, 3)$

Output: (sample, x pixels, y pixels, channels)

$= (100, (256-9+1), (256-9+1), 12)$

$= (100, 248, 248, 12)$

5. $\boxed{\text{PCA}}$

(a) sample mean $= \left[\dfrac{4+2+5+1}{4}, \dfrac{1+3+4+0}{4}\right] = [3, 2]$

Covariance matrix: $\tilde{x} = x - x_m$

$$= \begin{bmatrix} 1, & -1 \\ -1, & 1 \\ 2, & 2 \\ -2, & -2 \end{bmatrix}$$

$$Q = \frac{1}{4} \tilde{x}^T \tilde{x} = \frac{1}{4} \times \begin{bmatrix} 1 & -1 \\ -1 & 1 \\ 2 & 2 \\ -2 & -2 \end{bmatrix} \times \begin{bmatrix} 1 & -1 & 2 & -2 \\ -1 & 1 & 2 & -2 \end{bmatrix}$$

$$= \begin{bmatrix} 1\times1+(-1)\times(-1)+2\times2+(-2)\times(-2), \\ (-1)\times1+1\times(-1)+2\times2+(-2)\times(-2), \\ (-1)\times1+1\times(-1)+2\times2+(-2)\times(-2), \\ (-1)\times(-1)+1\times1+2\times2+(-2)\times(-2) \end{bmatrix}$$

$$= \frac{1}{4}\begin{bmatrix} 10, & 6 \\ 6, & 10 \end{bmatrix} = \begin{bmatrix} \frac{5}{2}, & \frac{3}{2} \\ \frac{3}{2}, & \frac{5}{2} \end{bmatrix}$$

(b)

Since $\lambda_1 > \lambda_2$, we choose V_1, then

$$\alpha = \frac{V^T x_1}{V^T V} = \frac{[1,1] \times [4,1]^T}{[1,1] \times [1,1]} = \frac{4+1}{2} = \frac{5}{2}$$

Project $\hat{x}_1 = \alpha V_1 = [2.5, 2.5]^T$

(c)

The projection of all samples in the direction of maximum variance is:

$$\alpha = \frac{V^T x}{V^T V} = \frac{[1,1] \times \begin{bmatrix} 4, 1 \\ 2, 3 \\ 5, 4 \\ 1, 0 \end{bmatrix}^T}{[1,1]^T \times [1,1]} = \frac{[5,5,9,1]}{2}$$

$$= \frac{[\frac{5}{2}, \frac{5}{2}, \frac{9}{2}, \frac{1}{2}]}{}$$

$$\begin{bmatrix} 4, 2, 5, 1 \\ 1, 3, 4, 0 \end{bmatrix} \times [1,1] = [5,5,9,1]$$

Projection $\hat{x} = \alpha V_1 = [\frac{5}{2}, \frac{5}{2}, \frac{9}{2}, \frac{1}{2}] \times [1,1]^T$

$$= \begin{bmatrix} \frac{5}{2}, \frac{5}{2}, \frac{9}{2}, \frac{1}{2} \\ \frac{5}{2}, \frac{5}{2}, \frac{9}{2}, \frac{1}{2} \end{bmatrix}$$

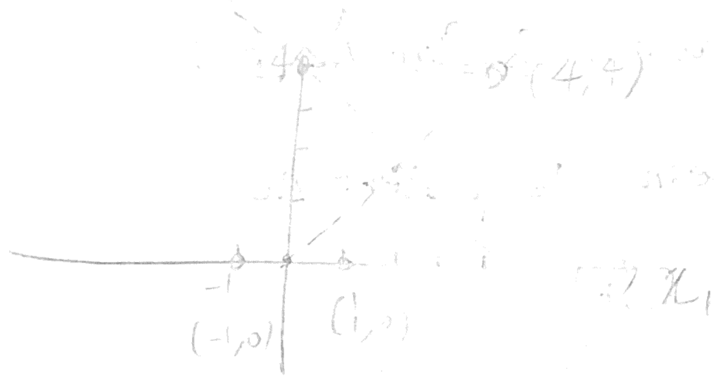
$$= 2^2 + 2^2 + 2^2 + 2^2$$
$$= \boxed{32}$$

$\hat{x}_m = [\frac{5}{2}, \frac{5}{2}]^T$

Thus, the total variance is: $\sum^{N} \| \hat{x}_i - \hat{x}_m \|^2 = \begin{bmatrix} 0, 0, \frac{4}{2}, -\frac{4}{2} \\ 0, 0, \frac{4}{2}, -\frac{4}{2} \end{bmatrix}^2$

6. K-means

(a)



After K-mean finishes on the training data,

the two clusters centers are:

$\mu_1 = \frac{1}{2}[(-1,0)+(1,0)] = (0,0)$

$\mu_2 = (4,4)$

And the equation of the boundary of the two clusters will be: $x_1+x_2=4$

when $x_1+x_2>4$, then it is belong to μ_2

when $x_1+x_2<4$, then it is belong to μ_1

(b)

```
def outlier_detect(x, xtr, nc, t):
    km = kMeans(n_cluster=nc)
    km.fit(xtr)
    mu = km.cluster_centers_
    dsq = np.sum((x[:, None, :] - mu[None, :, :])**2,
                 axis=2)

    dmin = np.min(dsq, axis=1)
    outlier = (dmin > t**2)
    return outlier
```