CS771 HW-4

Sahil Dhull
160607

Ques- 1

If v is the eigenvector of $\left(\frac{1}{N} X X^T\right)$, then

$$\left(\frac{1}{N} X X^T\right) v = \lambda v$$

where $\lambda \in R$ is the eigenvalue of $\left(\frac{1}{N} X X^T\right)$, $v \in R^N$

Multiplying Both sides by X^T,

$$\Rightarrow \quad X^T \left(\frac{1}{N} X X^T\right) v = X^T (\lambda v)$$

$$\Rightarrow \quad \frac{1}{N}(X^T X X^T) v = \lambda (X^T v)$$

$$\Rightarrow \quad \left(\frac{1}{N} X^T X\right)(X^T v) = \lambda (X^T v)$$

$$\Rightarrow \quad \left(\frac{1}{N} X^T X\right) K = \lambda K \qquad \text{where } K = X^T v.$$

So, $K = X^T v$ is the eigenvector of $\frac{1}{N} X^T X = S$

or $X^T v$ is eigenvector of $S = \left(\frac{1}{N} X^T X\right)$

The advantage of this way to obtain eigenvector of S is that since $N < D$, computation of eigenvector of $\frac{1}{N} X X^T$ will take lesser time than computation of eigenvector of $\frac{1}{N} X^T X$ as latter is $D \times D$ dimensional and former is $N \times N$ dimensional. And once eigen vector of $\frac{1}{N} X X^T$ is obtained (ie v), we just need to multiply it to X^T which will take $D \times N$ time. So, overall time is less in this way.

CS771 HW-4 Sahil Dhull
160607

Ques- 2

We are given the activation function

$$h(x) = x\,\sigma(\beta x) = \frac{x}{1+\exp(-\beta x)}$$

(1) Approximating linear Activation function

For $h(x)$ to approximate linear activation function,

$$h(x) \propto x$$

let $h(x) = kx$ where $k \in \mathbb{R}$

$$\Rightarrow \frac{x}{1+\exp(-\beta x)} = kx$$

$$\Rightarrow \frac{1}{1+\exp(-\beta x)} = k \Rightarrow \exp(-\beta x) = \frac{1}{k} - 1$$

$$\Rightarrow -\beta x = \log\left(\frac{1}{k}-1\right).$$

$$\Rightarrow \beta x = -\log\left(\frac{1}{k}-1\right) = c.$$

$$\Rightarrow \beta x = c \qquad \forall x \in \mathbb{R}.$$

This is only possible when $\beta = 0$.

$$\Rightarrow c = 0 \Rightarrow \log\left(\frac{1}{k}-1\right) = 0 \Rightarrow k = \frac{1}{2}$$

$\Rightarrow$ when $\beta = 0$, $h(x) = \frac{x}{2}$ is the obtained linear activation function.

(2) To approximate ReLU activation function,

$$h(x) = \begin{cases} x, & x \geq 0 \\ 0, & x < 0. \end{cases}$$

$$\Rightarrow \frac{x}{1+\exp(-\beta x)} = \begin{cases} x, & x \geq 0 \\ 0, & x < 0 \end{cases}$$

CS 771 HW-4

Sahil Dhull
160607

Ques-2 (continued)

For $x=0$, $h(x) = \frac{x}{1+\exp(-\beta x)} = 0$. So, it is true.

For $x>0$,

$$\frac{x}{1+\exp(-\beta x)} = x$$

$$\Rightarrow \quad 1+\exp(-\beta x) = 1$$

$$\Rightarrow \quad \exp(-\beta x) = 0$$

$$\Rightarrow \quad -\beta x \rightarrow -\infty \qquad \forall x>0$$

$$\Rightarrow \quad \beta \rightarrow \infty. \qquad -(1)$$

For $x<0$,

$$\frac{x}{1+\exp(-\beta x)} = 0$$

$$\Rightarrow \quad 1+\exp(-\beta x) \rightarrow \infty \quad \text{or} \quad \underbrace{1+\exp(-\beta x) \rightarrow -\infty}_{\text{Not possible}}$$

$$\Rightarrow \quad \exp(-\beta x) \rightarrow \infty$$

$$\Rightarrow \quad -\beta x \rightarrow \infty \qquad \forall x<0$$

$$\Rightarrow \quad \beta \rightarrow \infty \qquad -(2)$$

So, from (1) & (2), $\underline{\underline{\beta \rightarrow \infty}}$. makes $h(x)$ approximate ReLU activation function.

CS 771 HW-4

Sahil Dhull
160607

Ques- 3

Given : $z_n \sim \text{multinoulli}(\pi_1, \ldots, \pi_K)$

and $y_n \sim \text{Bernoulli}\left[\sigma(w_{z_n}^T x_n)\right]$

Now

$$p(y_n = 1 \mid x_n) = \sum_{k=1}^{K} p(y_n = 1 \mid x_n, z_n)\, p(z_n = k).$$

Since z_n is from multinoulli, $p(z_n = k) = \pi_k$.

and $p(y_n = 1 \mid x_n, z_n) = \sigma(w_{z_n}^T x_n)$

$$\Rightarrow \quad p(y_n = 1 \mid x_n) = \sum_{k=1}^{K} \sigma(w_k^T x_n)\, \pi_k$$

Therefore, the neural network has following layers:

1. Input layer $\rightarrow$ x_n which is D dimensional

2. Hidden Layer $\rightarrow$

   Weights : W ($K \times D$ dimensional)

   and $W_k = w_k$

   Activation: sigmoid (σ)

   Output : $\sigma(W x_n)$ which is K dimensional

3. Output layer $\rightarrow$

   Weights : Π ($K \times 1$ dimensional)

   where $\Pi_k = \pi_k$

   Activation : Identity (or linear)

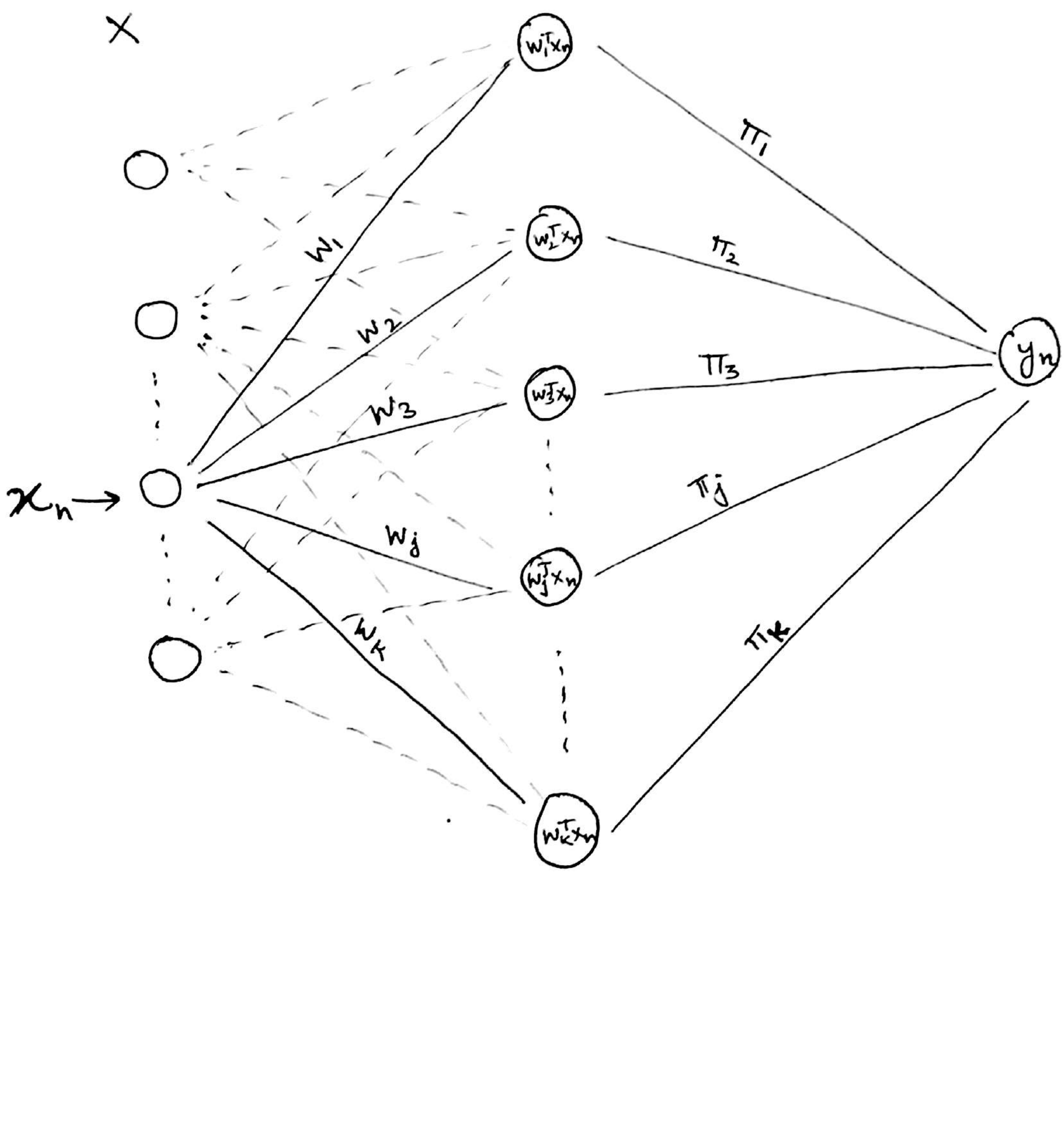
   Output : $y_n = \Pi^T \sigma(W x_n)$ which is 1 dimensional

CS771 HW-4

Sahil Dhull
160607

Ques- 3 (continued)

Neural Network:



CS771 HW-4 Sahil Dhull
160607

Ques- 4

We are given

$$p(X_{nm} \mid u_n, v_m, \theta_n, \phi_m) = N(X_{nm} \mid \theta_n + \phi_m + u_n^T v_m, \lambda^{-1})$$

$$= \sqrt{\frac{\lambda}{2\pi}} \exp\left(-\frac{\lambda}{2}(X_{nm} - \theta_n - \phi_m - u_n^T v_m)^2\right)$$

and priors on u_n and v_m are

$$p(u_n) = N(u_n \mid W_u a_n, \lambda_u^{-1} I_K)$$

$$p(v_m) = N(v_m \mid W_v b_m, \lambda_v^{-1} I_K)$$

Parameters are ~~u_n ($\forall$ $n \in \{1, N$~~

u_n, θ_n ($\forall$ n from 1 to N) , v_m, ϕ_m ($\forall$ m from 1 to M)

and W_u, W_v

Let Θ denote all the parameters.

Now we know

$$p(\Theta \mid X) = \frac{p(X \mid \Theta)\, p(\Theta)}{p(X)} \qquad \text{— (1)}$$

where

$p(\Theta)$ denotes priors of parameters and we only have priors for u_n and v_m.

Now $p(X)$ term will not be used (since it will be constant w.r.t. parameters)

$$p(\Theta) = \prod_{n=1}^{N} p(u_n) \prod_{m=1}^{M} p(v_m)$$

$$= \prod_{n=1}^{N} N(u_n \mid W_u a_n, \lambda_u^{-1} I_K) \prod_{m=1}^{M} N(v_m \mid W_v b_m, \lambda_u^{-1} I_K)$$

CS771 HW-4

Sahil Dhull
160607

Ques-4 (continued)

and $$p(X|\theta) = \prod_{n,m \in \Omega} p(X_{nm} | u_n, v_m, \theta_n, \phi_m)$$

$$= \prod_{(n,m)\in\Omega} N(X_{nm} | \theta_n + \phi_m + u_n^T v_m, \lambda_x^{-1})$$

$$= \prod_{(n,m)\in\Omega} \sqrt{\frac{\lambda_x}{2\pi}} \exp\left(-\frac{\lambda_x}{2}(X_{nm} - \theta_n - \phi_m - u_n^T v_m)^2\right)$$

Substituting in ①,

$$p(\theta|X) = \frac{\left[\prod_{(n,m)\in\Omega} \sqrt{\frac{\lambda_x}{2\pi}} \exp\left(-\frac{\lambda_x}{2}(X_{nm} - \theta_n - \phi_m - u_n^T v_m)^2\right)\right] \times \left[\prod_{n=1}^{N} N(u_n | W_u a_n, \lambda_u^{-1} I_K) \prod_{m=1}^{M} N(v_m | W_v b_m, \lambda_v^{-1} I_K)\right]}{p(X)}.$$

And loss function in this case will be

$$L \propto -\log(p(\theta|X))$$

or we simply write

$$L = -\log(p(\theta|X))$$

$$= -\log(p(X|\theta)) - \log(p(\theta))$$

$$\Rightarrow \boxed{L = \sum_{(n,m)\in\Omega} \lambda_x (X_{nm} - \theta_n - \phi_m - u_n^T v_m)^2 + \sum_{n=1}^{N} \lambda_u (u_n - W_u a_n)^2 + \sum_{m=1}^{M} \lambda_v (v_m - W_v b_m)^2}$$

This is the loss function.

CS771 HW-4

Sahil Dhull
160607

Ques-4 (continued)

Now we have obtained the loss function. We will be using ALT-OPT, for that we will need to differentiate loss function w.r.t. each parameter, keeping other parameters constant.

Diff. w.r.t. u_n,

$$\frac{\partial \mathcal{L}}{\partial u_n} = \sum_{(n,m)\in\Omega} \frac{\partial}{\partial u_n}\left(\lambda_x \left(X_{nm} - \theta_n - \phi_m - u_n^T v_m\right)^2\right)$$

$$+ \sum_{k=1}^{N} \frac{\partial}{\partial u_n}\left(\lambda_u \left(u_k - W_u a_k\right)^2\right) + \sum_{m=1}^{M} \frac{\partial}{\partial u_n}\left(\lambda_v \left(v_m - W_v b_m\right)^2\right)$$

The 3rd term will be 0 ($\because$ v_m is not dependent on u_n).
and in 2nd term, only term with $k=n$ will be left,

since $\frac{\partial u_k}{\partial u_n} = 0 \quad \forall\ k \neq n$.

In the 1st term, only those m which belong to Ω_{r_n} will be left.

$$\Rightarrow \frac{\partial \mathcal{L}}{\partial u_n} = \sum_{m\in\Omega_{r_n}} -2\lambda_x\left(X_{nm} - (\theta_n + \phi_m + u_n^T v_m)\right)v_m + 2\lambda_u\left(u_n - W_u a_n\right)$$

$$= 0$$

u_n is $K\times 1$ and v_m is $K \times 1$. $\Rightarrow u_n^T v_m = v_m^T u_n \in \mathbb{R}$.

$$\Rightarrow \left(\sum_{m\in\Omega_{r_n}} \lambda_x\left(X_{nm} - (\theta_n + \phi_m)\right)v_m + \lambda_u W_u a_n\right)$$

$$= \left(\sum_{m\in\Omega_{r_n}} \lambda_x v_m v_m^T + \lambda_u I_K\right) u_n$$

$$\Rightarrow \boxed{u_n = \left(\sum_{m\in\Omega_{r_n}} \lambda_x v_m v_m^T + \lambda_u I_K\right)^{-1}\left(\sum_{m\in\Omega_{r_n}} \lambda_x (X_{nm} - \theta_n - \phi_m) v_m + \lambda_u W_u a_n\right)}$$

CS771 HW-4

Sahil Dhull
160607

Ques 4 (continued)

Similarly for v_m,

$$\frac{\partial L}{\partial v_m} = \sum_{n \in \Omega_{cm}} -2\lambda_x (X_{nm} - \theta_n - \phi_m - u_n^T v_m) u_n + 2\lambda_v (v_m - W_v b_m) = 0$$

$$\Rightarrow \boxed{v_m = \left(\sum_{n \in \Omega_{cm}} \lambda_x u_n u_n^T + \lambda_v I_K\right)^{-1} \left(\sum_{n \in \Omega_{cm}} \lambda_x (X_{nm} - \theta_n - \phi_m) u_n + \lambda_v W_v b_m\right)}$$

Diff. w.r.t. θ_n,

$$\frac{\partial L}{\partial \theta_n} = \sum_{m \in \Omega_{rn}} -2\lambda_x (X_{nm} - \theta_n - \phi_m - u_n^T v_m) = 0$$

$$\Rightarrow \boxed{\theta_n = \frac{1}{|\Omega_{rn}|} \left(\sum_{m \in \Omega_{rn}} (X_{nm} - \phi_m - u_n^T v_m)\right)}$$

Diff. w.r.t. ϕ_m,

$$\frac{\partial L}{\partial \phi_m} = \sum_{n \in \Omega_{cm}} -2\lambda_x (X_{nm} - \theta_n - \phi_m - u_n^T v_m) = 0$$

$$\Rightarrow \boxed{\phi_m = \frac{1}{|\Omega_{cm}|} \left(\sum_{n \in \Omega_{cm}} (X_{nm} - \theta_n - u_n^T v_m)\right)}$$

Diff. w.r.t. W_u,

$$\cancel{\frac{\partial L}{\partial W_u} = \sum_{n=1}^{N} |\Omega_{rn}| (-u_n a_n^T + W_u a_n a_n^T) = 0}$$

$$\Rightarrow \cancel{\boxed{W_u = \left(\sum_{n=1}^{N} |\Omega_{rn}| u_n a_n^T\right)\left(\sum_{n=1}^{N} |\Omega_{rn}| a_n a_n^T\right)^{-1}}}$$

CS771 HW-4 Sahil Dhull 160607

Ques 4 (continued)

Diff. w.r.t. W_u,

$$\frac{\partial \mathcal{L}}{\partial W_u} = \sum_{n=1}^{N} (-u_n a_n^T + W_u a_n a_n^T) = 0$$

$$\Rightarrow \boxed{W_u = \left(\sum_{n=1}^{N} u_n a_n^T\right)\left(\sum_{n=1}^{N} a_n a_n^T\right)^{-1}}$$

Diff. w.r.t. W_v,

$$\frac{\partial \mathcal{L}}{\partial W_v} = \sum_{m=1}^{M} (-v_m b_m^T + W_v b_m b_m^T) = 0$$

$$\Rightarrow \boxed{W_v = \left(\sum_{m=1}^{M} v_m b_m^T\right)\left(\sum_{m=1}^{M} b_m b_m^T\right)^{-1}}$$

Now ALT-OPT ~~of~~ steps:

1. Initialize $\theta^{(0)} = \{u_n, \theta_n\}_{n=1}^{N}$, $\{v_m, \phi_m\}_{m=1}^{M}$, W_u and W_v and ~~t=0~~, $t=1$
2. ~~Compute $\theta^{(t)}$ based on~~
   Compute the loss function for $\theta^{(t-1)}$
3. Update all parameters sequentially as per the update equations.
   Choose any random n, update $u_n^{(t)}$ and $\theta_n^{(t)}$
   Choose any random m, update $v_m^{(t)}$ and $\phi_m^{(t)}$
   Update $W_u^{(t)}$ and $W_v^{(t)}$.
4. Set $t = t+1$ and repeat from step 2 until convergence

CS 771 HW-4 Sahil Dhull

160607

Ques- 5

Programming Problem 1:

The more the number of dimensions, better are the results. Reason is that lowering the dimensions leads to more loss of information and thus a lower dimensional image is insufficient to capture all of main features of image.

Programming Problem 2:

For small MNIST dataset, tSNE does better separation in clusters obtained by K-Means as compared to PCA.