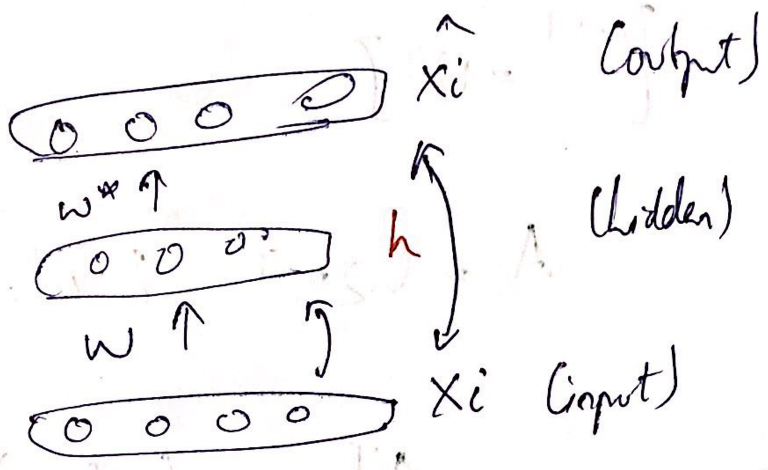
Lecture 7

7.1 Introduction to Autoencoders



Special type of feed forward ~~ntt~~ neural network

Encodes its input x_i into a hidden representation h

Encoder function

$$h = g(Wx_i + b)$$

$x_i \in \mathbb{R}^n$ $\quad h \in \mathbb{R}^d$ $\quad W \in \mathbb{R}^{d \times n}$

$b \in \mathbb{R}^{d \times n}$

$h \in \mathbb{R}^{d \times 1}$ $\quad \hat{x_i} = f(W^* h + c)$ $\quad W \in \mathbb{R}^{n \times d}$

$x_i \in \mathbb{R}^n$

Decoder, decodes the input again from the hidden representation

The model is trained to minimise a certain loss function which will ensure that $\hat{x_i}$ is close to x_i

~~Vot~~ Under Complete Autoencoder.

if $\dim(h) < \dim(x_i)$

if we are able to reconstruct $\hat{x_i}$ perfectly from ~~ha~~, h, then h is a loss-free encoding of x_i, It captures all the ~~important~~ important characteristics of x_i.

Analogy with PCA.

Over Complete Autoencoder.

$\dim(h) \geq \dim(x_i)$

Identity encoding is useless.

→

→ Choice of $f(x_i)$ & $g(x_i)$

→ Choice of loss function

If i/p binary → function for decoder is ~~fun~~ 'Logistic function'

if i/p real number,
the function will be linear
g is typically chosen as sigmoid.

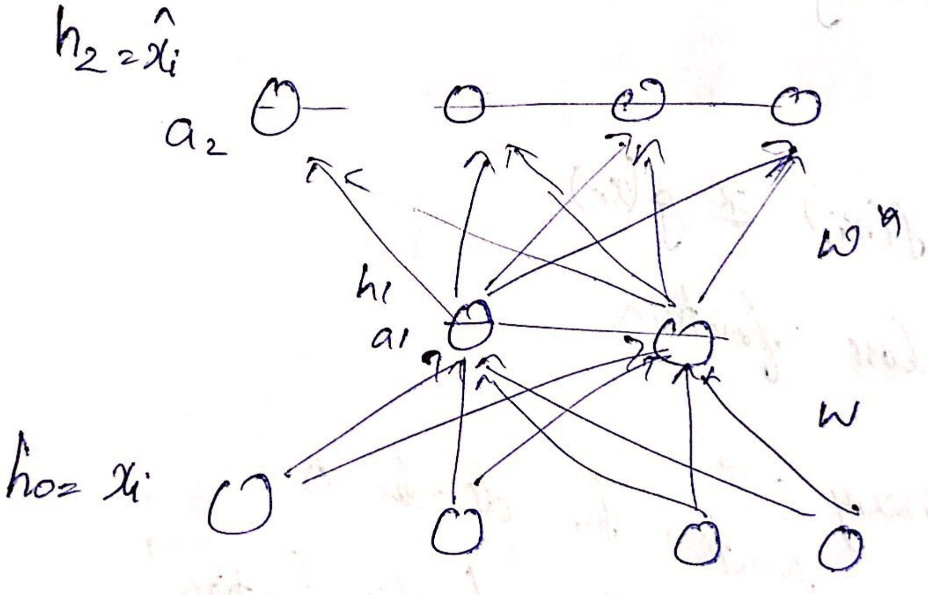
Loss function

i/p are real-valued.
Squared error loss

We can train the autoencoder just like a regular feedforward network.

$$\frac{\partial \mathcal{L}(\theta)}{\partial W^*} \quad \& \quad \frac{\partial \mathcal{L}(\theta)}{\partial W}$$

$$\mathcal{L}(\theta) = (\hat{x}_i - x_i)^T (\hat{x}_i - x_i)$$



$$\frac{\partial \mathcal{L}(\theta)}{\partial W^*} = \frac{\partial \mathcal{L}(\theta)}{\partial h_2} \left[\frac{\partial h_2}{\partial a_2} \frac{\partial a_2}{\partial W^*} \right]$$

$$\frac{\partial \mathcal{L}(\theta)}{\partial W} = \frac{\partial \mathcal{L}(\theta)}{\partial h_2} \left[\frac{\partial h_2}{\partial a_2} \frac{\partial a_2}{\partial h_1} \frac{\partial h_1}{\partial a_1} \frac{\partial a_1}{\partial W} \right]$$

$$\frac{\partial \mathcal{L}(\theta)}{\partial h_2} = \frac{\partial \mathcal{L}(\theta)}{\partial \hat{x}_i}$$

$$= \nabla_{\hat{x}_i} \left\{ (\hat{x}_i - x_i)^T (\hat{x}_i - x_i) \right\}$$

$$= 2 (\hat{x}_i - x_i)$$

Input - Binary

→ ~~Sigmoid~~ Logistic decoder output b/w 0 & 1

$$\min = \left\{ - \sum_{j=1}^{n} (x_{ij} \log \hat{x}_{ij} + (1 - x_{ij}) \log (1 - \hat{x}_{ij})) \right\}$$

$p = [1, 0]$ (p_0 p_1)

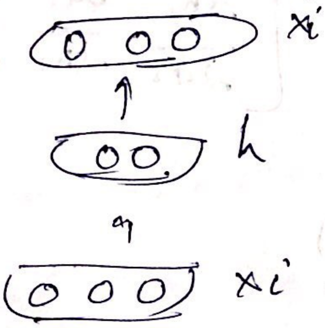
$q = [0.8, 0.2]$ (q_0 q_1)

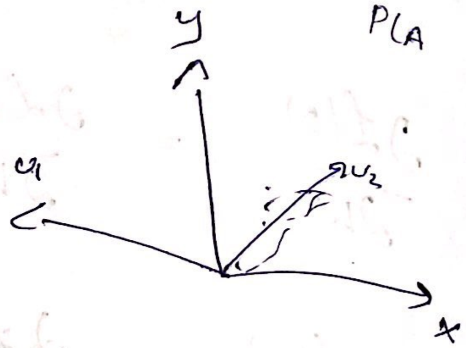
$$-\sum_{i=0}^{1} p_i \log q_i$$

Cross Entropy Function

for input is binary

7.2 Link b/w PCA & Auto Encoders





The encoder part of an autoencoder is equivalent to PCA if we

→ use a linear encoder

→ use a linear decoder

→ using squared error loss

→ normalize the inputs to

$$\hat{x}_{ij} = \frac{1}{\sqrt{m}}\left(x_{ij} - \frac{1}{m}\sum_{k=1}^{m} x_{kj}\right)$$

$$(X^T)X = \frac{1}{m}(X')^T X'$$

~~Optimal~~

Linear Decoder & Squared Error loss function

the the optimal solution to the following objective function

$$\frac{1}{m} \sum_{i=1}^{m} \sum_{j=1}^{n} (x_{ij} - \hat{x}_{ij})^2$$

when we use of a linear encoder

$$\min_{\theta} \sum_{i=1}^{m} \sum_{j=1}^{n} (x_{ij} - \hat{x}_{ij})^2$$

This is equivalent to

$$\min_{W, H} (\| X - HW^* \|_F)^2$$

$$HW^* = U_{\cdot, \leq k}$$

H is actually a linear encoding & find an expression for the encoder weights W.

$$H = U_{\cdot, \leq k} \Sigma_{k,k} = (XX^T)(XX^T)^{-1} U_{\cdot, \leq k} \Sigma_{k,k}$$

$$= XVI$$

7.3

Regularization in Autoencoders

to generalization

~~training~~ time

~~the~~

if less generalization (then overfitting)

Regularization is done to avoid overfitting

if the no of parameters are high,

it leads to overfitting

(over complete autoencoders a regularization is required as the number of parameters are very high) . But there are situations where we need regularization for under complete auto encoders

Simplest solution add L2 regularization

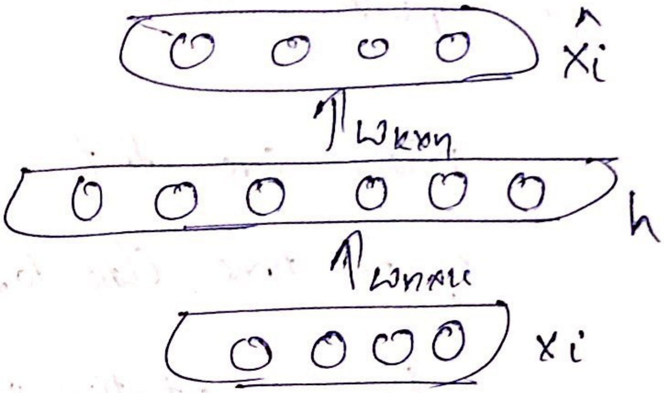
$$\left[\min_{\theta, W, W^*, b, c} \frac{1}{m} \sum_{i=1}^{m} \sum_{j=1}^{n} (\hat{x}_{ij} - x_{ij})^2\right] + \lambda \|\theta\|^2$$

objective function $\mathcal{L}(\theta)$ L(L2) $\Omega(\theta)$

$\theta = (W_1, W_2 \ldots \ldots)$

$\frac{\partial \mathcal{L}(\theta)}{\partial W_1}$ adds the term $2\lambda W$ to the gradient $\partial \mathcal{L}(\theta)$

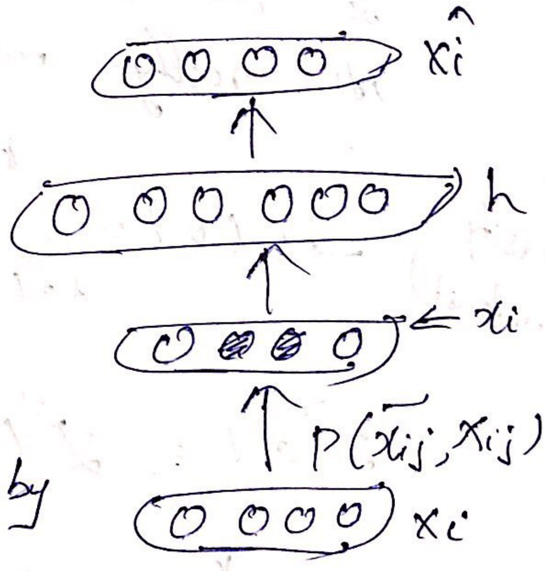
$W^* = W^T$



7.4 Denoising Autoencoders

A denoising encoder simply corrupts the input data using a probabilistic process $(P(\tilde{x}_{ij} | x_{ij}))$ before feeding it to the network

[Robust to change at test times]



No longer get away by memorising the data.

- Or add Gaussian Noise

$$\tilde{x}_{ij} = x_{ij} + \mathcal{N}(0,1)$$

Gaussian Noise is better than L2 regularization

7.5 Sparse Encoder

Make sure that neuron is inactive most of the time close to zero

Average activation of neuron is close to zero.

Average value activation of neuron $\Rightarrow$ $\hat{p_i} = \frac{1}{m} \sum_{i=1}^{m} h(x_i)_e$

This is kind of regularization, So its like whenever its active, its really meaningful patterns.

(Sparsity constant) $\Omega(\theta) = \sum_{l=1}^{k} p \log \frac{p}{\hat{p_i}} + (1-p) \log \frac{1-p}{1-\hat{p_i}}$

$$\mathcal{L}(\theta) = \mathcal{L}'(\theta) + \Omega(\theta)$$

$$\Omega(\theta) = \sum_{l=1}^{k} p \log \left(\frac{p}{\hat{p_i}}\right) + (1-p) \log \frac{(1-p)}{(1-\hat{p_i})}$$

7.6 Contractive Autoencoders

Some kind of regularization

$$\Omega(\theta) = \| J_x(h) \|_F^2$$

$\mathbb{R}^k$

(Jacobian) $\mathbb{R}^n$

$$\| J_x(h) \|_F^2 = \sum_{j=1}^{n} \sum_{l=1}^{k} \left(\frac{\partial h_l}{\partial x_j} \right)^2$$

Not very sensitive to variations.

Regularization

Weight decay $\Omega(\theta) = \lambda \| \theta \|^2$

Sparse $\Omega(\theta) = \sum_{l=1}^{k} \rho \log \frac{\rho}{\hat{\rho}_l} + (1-\rho) \log \frac{1-\rho}{1-\hat{\rho}_l}$

Contractive $\Omega(\theta) = \sum_{j=1}^{n} \sum_{l=1}^{k} \left(\frac{\partial h_l}{\partial x_j} \right)^2$