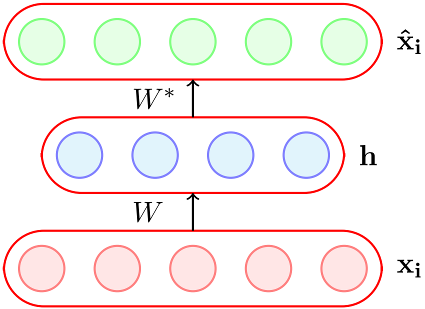
CS7015 (Deep Learning) : Lecture 7

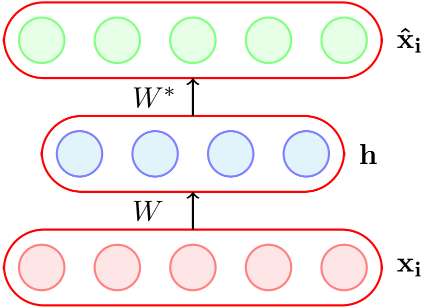
Autoencoders and relation to PCA, Regularization in autoencoders, Denoising autoencoders, Sparse autoencoders, Contractive autoencoders

Mitesh M. Khapra

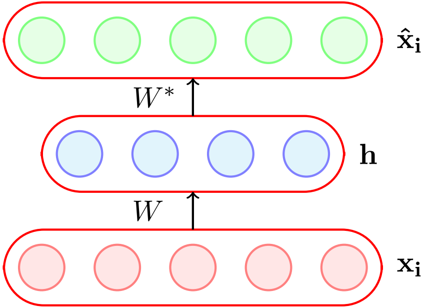
Department of Computer Science and Engineering
Indian Institute of Technology Madras

Module 7.1: Introduction to Autoencoders

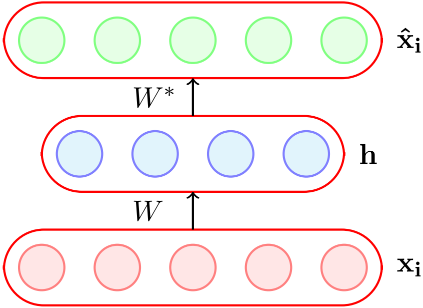

$\hat{\mathbf{x}}_\mathbf{i}$
W^*
$\mathbf{h}$
W
$\mathbf{x}_\mathbf{i}$



- An autoencoder is a special type of feed forward neural network which does the following

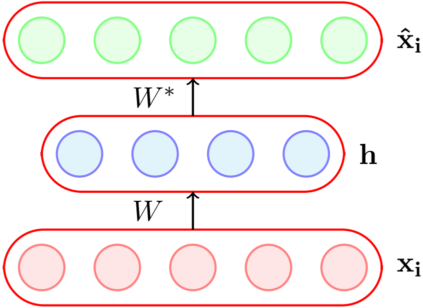


- An autoencoder is a special type of feed forward neural network which does the following
- Encodes its input $\mathbf{x_i}$ into a hidden representation $\mathbf{h}$



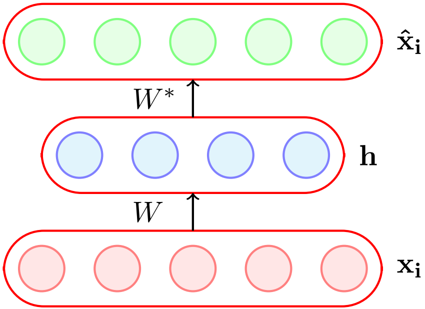
$$\mathbf{h} = g(W\mathbf{x_i} + \mathbf{b})$$

- An autoencoder is a special type of feed forward neural network which does the following
- Encodes its input $\mathbf{x_i}$ into a hidden representation $\mathbf{h}$



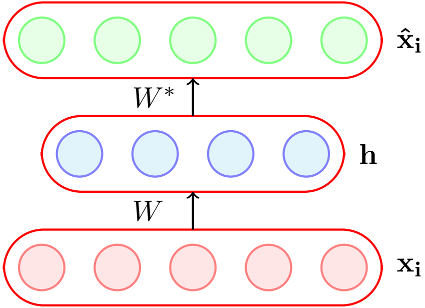
$$\mathbf{h} = g(W\mathbf{x_i} + \mathbf{b})$$

- An autoencoder is a special type of feed forward neural network which does the following
- Encodes its input $\mathbf{x_i}$ into a hidden representation $\mathbf{h}$
- Decodes the input again from this hidden representation



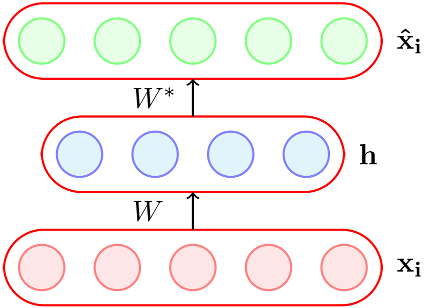
$$\mathbf{h} = g(W\mathbf{x}_\mathbf{i} + \mathbf{b})$$
$$\hat{\mathbf{x}}_\mathbf{i} = f(W^*\mathbf{h} + \mathbf{c})$$

- An autoencoder is a special type of feed forward neural network which does the following
- Encodes its input $\mathbf{x}_\mathbf{i}$ into a hidden representation $\mathbf{h}$
- Decodes the input again from this hidden representation

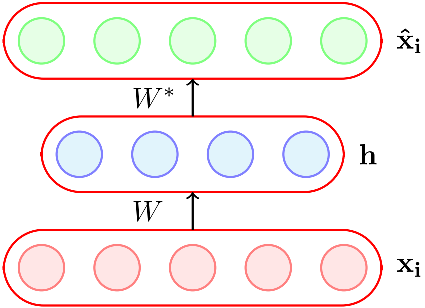


$$\mathbf{h} = g(W\mathbf{x}_\mathbf{i} + \mathbf{b})$$
$$\hat{\mathbf{x}}_\mathbf{i} = f(W^*\mathbf{h} + \mathbf{c})$$

- An autoencoder is a special type of feed forward neural network which does the following
- Encodes its input $\mathbf{x}_\mathbf{i}$ into a hidden representation $\mathbf{h}$
- Decodes the input again from this hidden representation
- The model is trained to minimize a certain loss function which will ensure that $\hat{\mathbf{x}}_\mathbf{i}$ is close to $\mathbf{x}_\mathbf{i}$ (we will see some such loss functions soon)

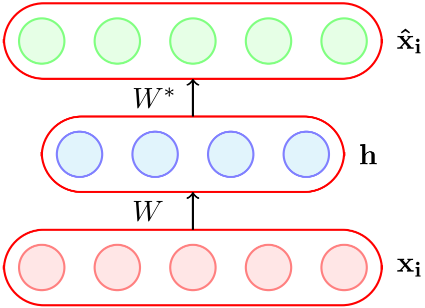


$$\mathbf{h} = g(W\mathbf{x_i} + \mathbf{b})$$
$$\hat{\mathbf{x}}_\mathbf{i} = f(W^*\mathbf{h} + \mathbf{c})$$



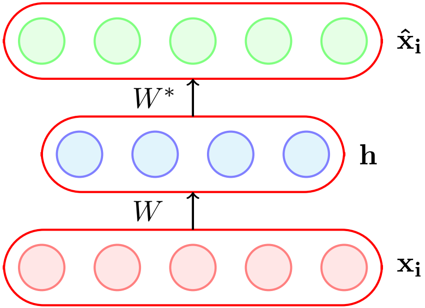
$$\mathbf{h} = g(W\mathbf{x}_\mathbf{i} + \mathbf{b})$$
$$\hat{\mathbf{x}}_\mathbf{i} = f(W^*\mathbf{h} + \mathbf{c})$$

- Let us consider the case where $\dim(\mathbf{h}) < \dim(\mathbf{x}_\mathbf{i})$



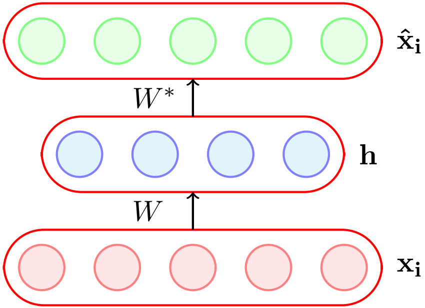
$$\mathbf{h} = g(W\mathbf{x}_\mathbf{i} + \mathbf{b})$$
$$\hat{\mathbf{x}}_\mathbf{i} = f(W^*\mathbf{h} + \mathbf{c})$$

- Let us consider the case where $\dim(\mathbf{h}) < \dim(\mathbf{x}_\mathbf{i})$
- If we are still able to reconstruct $\hat{\mathbf{x}}_\mathbf{i}$ perfectly from $\mathbf{h}$, then what does it say about $\mathbf{h}$?



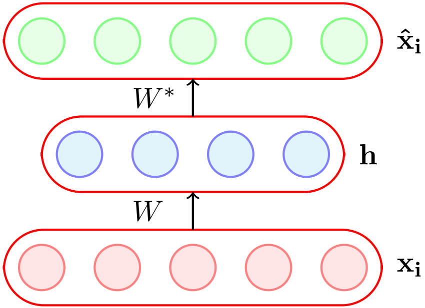
$$\mathbf{h} = g(W\mathbf{x}_\mathbf{i} + \mathbf{b})$$
$$\hat{\mathbf{x}}_\mathbf{i} = f(W^*\mathbf{h} + \mathbf{c})$$

- Let us consider the case where $\dim(\mathbf{h}) < \dim(\mathbf{x}_\mathbf{i})$
- If we are still able to reconstruct $\hat{\mathbf{x}}_\mathbf{i}$ perfectly from $\mathbf{h}$, then what does it say about $\mathbf{h}$?
- $\mathbf{h}$ is a loss-free encoding of $\mathbf{x}_\mathbf{i}$. It captures all the important characteristics of $\mathbf{x}_\mathbf{i}$



$$\mathbf{h} = g(W\mathbf{x}_\mathbf{i} + \mathbf{b})$$
$$\hat{\mathbf{x}}_\mathbf{i} = f(W^*\mathbf{h} + \mathbf{c})$$

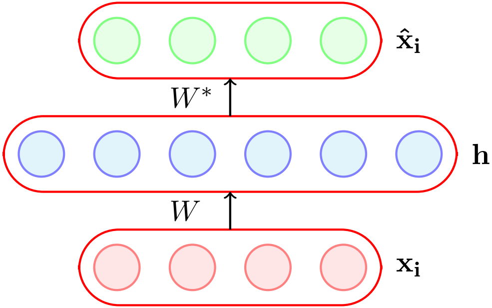
- Let us consider the case where $\dim(\mathbf{h}) < \dim(\mathbf{x}_\mathbf{i})$
- If we are still able to reconstruct $\hat{\mathbf{x}}_\mathbf{i}$ perfectly from $\mathbf{h}$, then what does it say about $\mathbf{h}$?
- $\mathbf{h}$ is a loss-free encoding of $\mathbf{x}_\mathbf{i}$. It captures all the important characteristics of $\mathbf{x}_\mathbf{i}$
- Do you see an analogy with PCA?



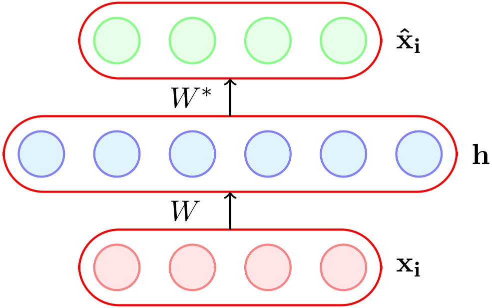
$$\mathbf{h} = g(W\mathbf{x}_\mathbf{i} + \mathbf{b})$$
$$\hat{\mathbf{x}}_\mathbf{i} = f(W^*\mathbf{h} + \mathbf{c})$$

An autoencoder where $\dim(\mathbf{h}) < \dim(\mathbf{x}_\mathbf{i})$ is called an <u>under complete</u> autoencoder

- Let us consider the case where $\dim(\mathbf{h}) < \dim(\mathbf{x}_\mathbf{i})$
- If we are still able to reconstruct $\hat{\mathbf{x}}_\mathbf{i}$ perfectly from $\mathbf{h}$, then what does it say about $\mathbf{h}$?
- $\mathbf{h}$ is a loss-free encoding of $\mathbf{x}_\mathbf{i}$. It captures all the important characteristics of $\mathbf{x}_\mathbf{i}$
- Do you see an analogy with PCA?

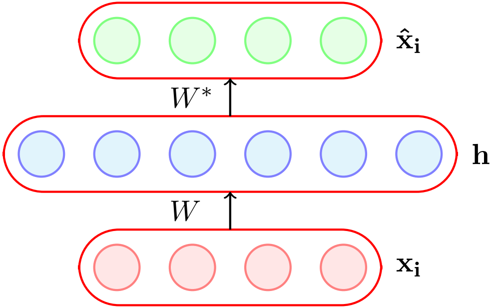


$$\mathbf{h} = g(W\mathbf{x}_\mathbf{i} + \mathbf{b})$$
$$\hat{\mathbf{x}}_\mathbf{i} = f(W^*\mathbf{h} + \mathbf{c})$$



$$\mathbf{h} = g(W\mathbf{x}_\mathbf{i} + \mathbf{b})$$
$$\hat{\mathbf{x}}_\mathbf{i} = f(W^*\mathbf{h} + \mathbf{c})$$

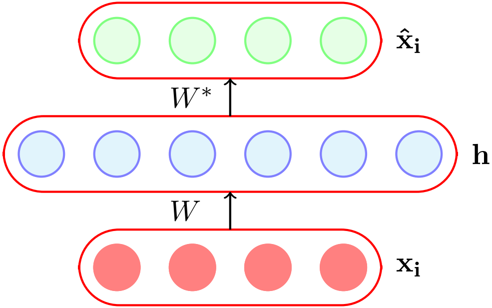
- Let us consider the case when $\dim(\mathbf{h}) \geq \dim(\mathbf{x}_\mathbf{i})$



$$\mathbf{h} = g(W\mathbf{x}_\mathbf{i} + \mathbf{b})$$

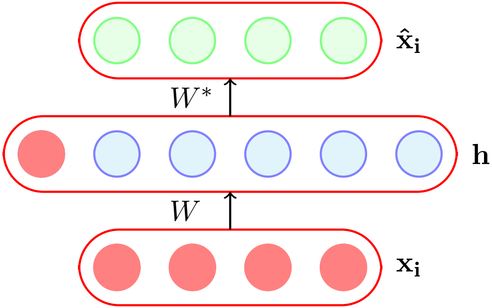
$$\hat{\mathbf{x}}_\mathbf{i} = f(W^*\mathbf{h} + \mathbf{c})$$

- Let us consider the case when $\dim(\mathbf{h}) \geq \dim(\mathbf{x}_\mathbf{i})$
- In such a case the autoencoder could learn a trivial encoding by simply copying $\mathbf{x}_\mathbf{i}$ into $\mathbf{h}$ and then copying $\mathbf{h}$ into $\hat{\mathbf{x}}_\mathbf{i}$



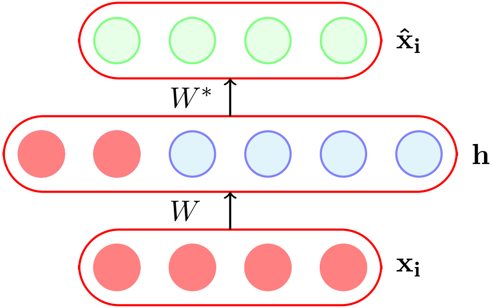
$$\mathbf{h} = g(W\mathbf{x}_\mathbf{i} + \mathbf{b})$$
$$\hat{\mathbf{x}}_\mathbf{i} = f(W^*\mathbf{h} + \mathbf{c})$$

- Let us consider the case when $\dim(\mathbf{h}) \geq \dim(\mathbf{x}_\mathbf{i})$
- In such a case the autoencoder could learn a trivial encoding by simply copying $\mathbf{x}_\mathbf{i}$ into $\mathbf{h}$ and then copying $\mathbf{h}$ into $\hat{\mathbf{x}}_\mathbf{i}$



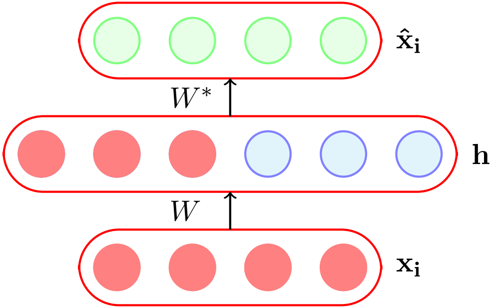
$$\mathbf{h} = g(W\mathbf{x}_\mathbf{i} + \mathbf{b})$$
$$\hat{\mathbf{x}}_\mathbf{i} = f(W^*\mathbf{h} + \mathbf{c})$$

- Let us consider the case when $\dim(\mathbf{h}) \geq \dim(\mathbf{x}_\mathbf{i})$
- In such a case the autoencoder could learn a trivial encoding by simply copying $\mathbf{x}_\mathbf{i}$ into $\mathbf{h}$ and then copying $\mathbf{h}$ into $\hat{\mathbf{x}}_\mathbf{i}$



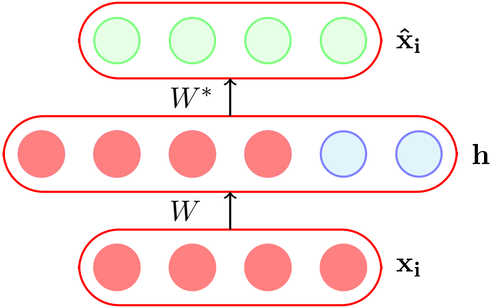
$$\mathbf{h} = g(W\mathbf{x}_\mathbf{i} + \mathbf{b})$$
$$\hat{\mathbf{x}}_\mathbf{i} = f(W^*\mathbf{h} + \mathbf{c})$$

- Let us consider the case when $\dim(\mathbf{h}) \geq \dim(\mathbf{x}_\mathbf{i})$
- In such a case the autoencoder could learn a trivial encoding by simply copying $\mathbf{x}_\mathbf{i}$ into $\mathbf{h}$ and then copying $\mathbf{h}$ into $\hat{\mathbf{x}}_\mathbf{i}$



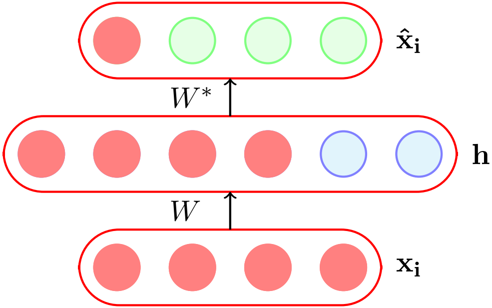
$$\mathbf{h} = g(W\mathbf{x}_\mathbf{i} + \mathbf{b})$$
$$\hat{\mathbf{x}}_\mathbf{i} = f(W^*\mathbf{h} + \mathbf{c})$$

- Let us consider the case when $\dim(\mathbf{h}) \geq \dim(\mathbf{x}_\mathbf{i})$
- In such a case the autoencoder could learn a trivial encoding by simply copying $\mathbf{x}_\mathbf{i}$ into $\mathbf{h}$ and then copying $\mathbf{h}$ into $\hat{\mathbf{x}}_\mathbf{i}$



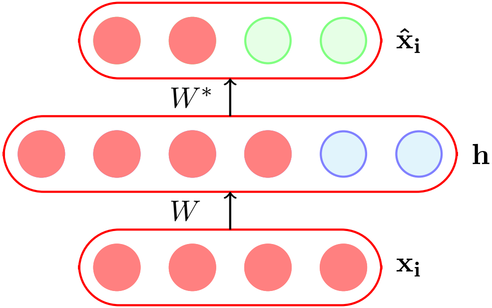
$$\mathbf{h} = g(W\mathbf{x}_\mathbf{i} + \mathbf{b})$$
$$\hat{\mathbf{x}}_\mathbf{i} = f(W^*\mathbf{h} + \mathbf{c})$$

- Let us consider the case when $\dim(\mathbf{h}) \geq \dim(\mathbf{x}_\mathbf{i})$
- In such a case the autoencoder could learn a trivial encoding by simply copying $\mathbf{x}_\mathbf{i}$ into $\mathbf{h}$ and then copying $\mathbf{h}$ into $\hat{\mathbf{x}}_\mathbf{i}$



$$\mathbf{h} = g(W\mathbf{x_i} + \mathbf{b})$$
$$\hat{\mathbf{x}}_\mathbf{i} = f(W^*\mathbf{h} + \mathbf{c})$$

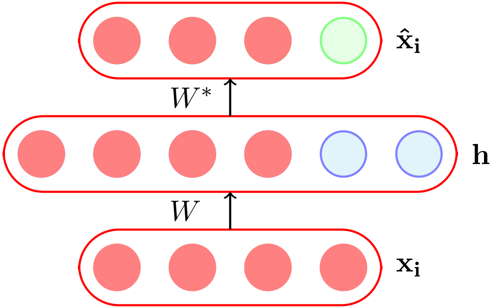
- Let us consider the case when $\dim(\mathbf{h}) \geq \dim(\mathbf{x_i})$
- In such a case the autoencoder could learn a trivial encoding by simply copying $\mathbf{x_i}$ into $\mathbf{h}$ and then copying $\mathbf{h}$ into $\hat{\mathbf{x}}_\mathbf{i}$



$$\mathbf{h} = g(W\mathbf{x}_\mathbf{i} + \mathbf{b})$$

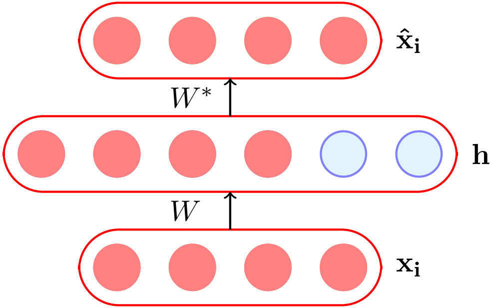
$$\hat{\mathbf{x}}_\mathbf{i} = f(W^*\mathbf{h} + \mathbf{c})$$

- Let us consider the case when $\dim(\mathbf{h}) \geq \dim(\mathbf{x}_\mathbf{i})$
- In such a case the autoencoder could learn a trivial encoding by simply copying $\mathbf{x}_\mathbf{i}$ into $\mathbf{h}$ and then copying $\mathbf{h}$ into $\hat{\mathbf{x}}_\mathbf{i}$



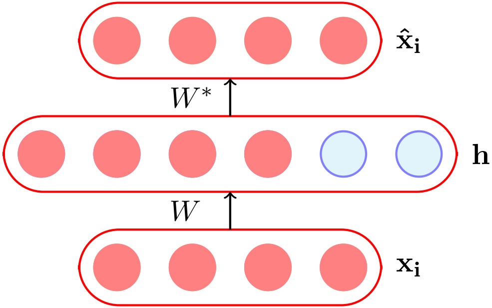
$$\mathbf{h} = g(W\mathbf{x}_\mathbf{i} + \mathbf{b})$$
$$\hat{\mathbf{x}}_\mathbf{i} = f(W^*\mathbf{h} + \mathbf{c})$$

- Let us consider the case when $\dim(\mathbf{h}) \geq \dim(\mathbf{x}_\mathbf{i})$
- In such a case the autoencoder could learn a trivial encoding by simply copying $\mathbf{x}_\mathbf{i}$ into $\mathbf{h}$ and then copying $\mathbf{h}$ into $\hat{\mathbf{x}}_\mathbf{i}$



$$\mathbf{h} = g(W\mathbf{x}_\mathbf{i} + \mathbf{b})$$
$$\hat{\mathbf{x}}_\mathbf{i} = f(W^*\mathbf{h} + \mathbf{c})$$

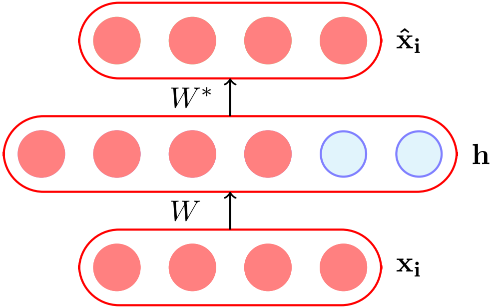
- Let us consider the case when $\dim(\mathbf{h}) \geq \dim(\mathbf{x}_\mathbf{i})$
- In such a case the autoencoder could learn a trivial encoding by simply copying $\mathbf{x}_\mathbf{i}$ into $\mathbf{h}$ and then copying $\mathbf{h}$ into $\hat{\mathbf{x}}_\mathbf{i}$



$$\mathbf{h} = g(W\mathbf{x}_\mathbf{i} + \mathbf{b})$$

$$\hat{\mathbf{x}}_\mathbf{i} = f(W^*\mathbf{h} + \mathbf{c})$$

- Let us consider the case when $\dim(\mathbf{h}) \geq \dim(\mathbf{x}_\mathbf{i})$
- In such a case the autoencoder could learn a trivial encoding by simply copying $\mathbf{x}_\mathbf{i}$ into $\mathbf{h}$ and then copying $\mathbf{h}$ into $\hat{\mathbf{x}}_\mathbf{i}$
- Such an identity encoding is useless in practice as it does not really tell us anything about the important characteristics of the data



$$\mathbf{h} = g(W\mathbf{x}_\mathbf{i} + \mathbf{b})$$
$$\hat{\mathbf{x}}_\mathbf{i} = f(W^*\mathbf{h} + \mathbf{c})$$

An autoencoder where $\dim(\mathbf{h}) \geq \dim(\mathbf{x}_\mathbf{i})$ is called an over complete autoencoder

- Let us consider the case when $\dim(\mathbf{h}) \geq \dim(\mathbf{x}_\mathbf{i})$
- In such a case the autoencoder could learn a trivial encoding by simply copying $\mathbf{x}_\mathbf{i}$ into $\mathbf{h}$ and then copying $\mathbf{h}$ into $\hat{\mathbf{x}}_\mathbf{i}$
- Such an identity encoding is useless in practice as it does not really tell us anything about the important characteristics of the data

The Road Ahead

The Road Ahead

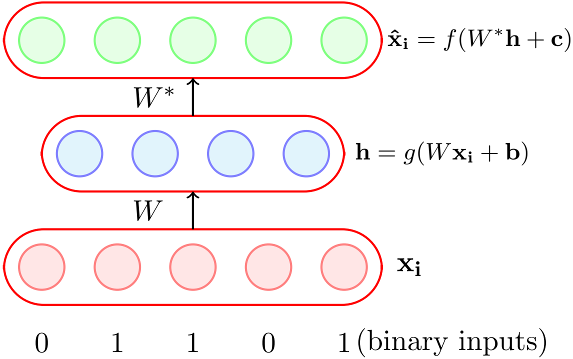
- Choice of $f(\mathbf{x_i})$ and $g(\mathbf{x_i})$

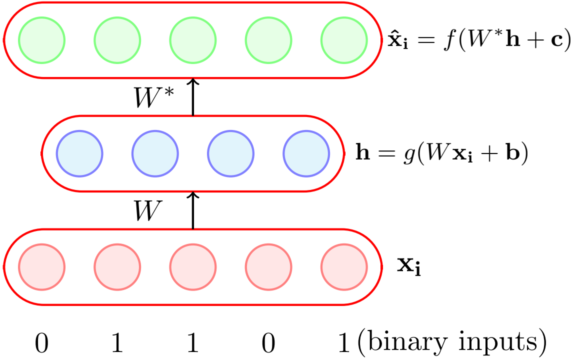
The Road Ahead

- Choice of $f(\mathbf{x_i})$ and $g(\mathbf{x_i})$
- Choice of loss function

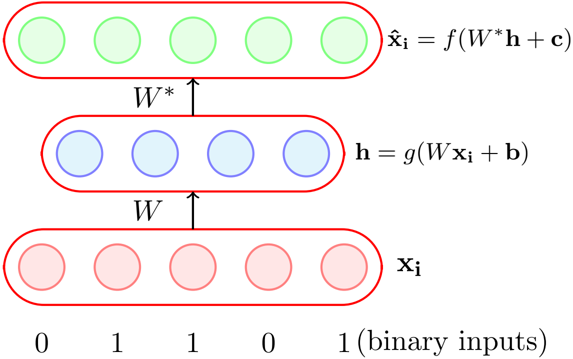
The Road Ahead

- Choice of $f(\mathbf{x_i})$ and $g(\mathbf{x_i})$
- Choice of loss function

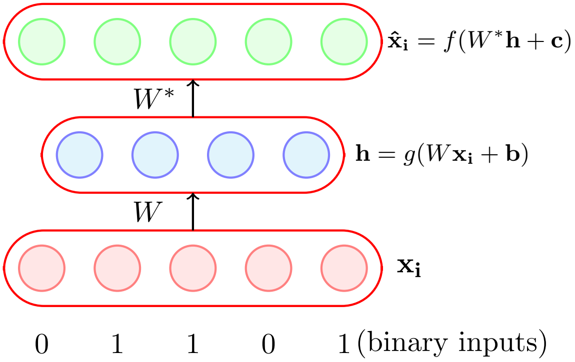

$\hat{\mathbf{x}}_\mathbf{i} = f(W^*\mathbf{h} + \mathbf{c})$
W^*
$\mathbf{h} = g(W\mathbf{x}_\mathbf{i} + \mathbf{b})$
W
$\mathbf{x}_\mathbf{i}$
0 1 1 0 1 (binary inputs)



- Suppose all our inputs are binary (each $x_{ij} \in \{0, 1\}$)

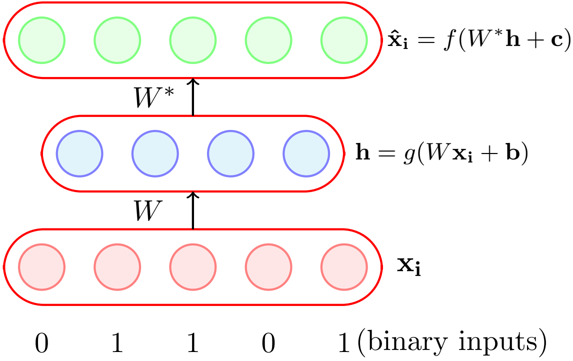


- Suppose all our inputs are binary (each $x_{ij} \in \{0, 1\}$)
- Which of the following functions would be most apt for the decoder?



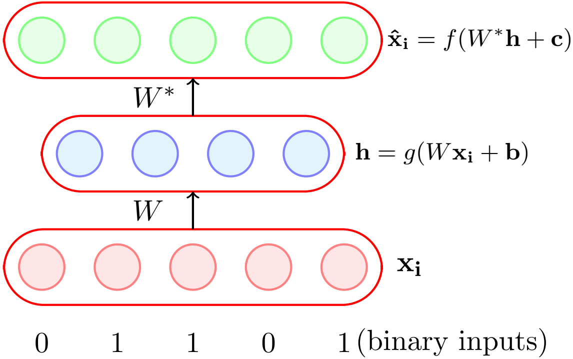
- Suppose all our inputs are binary (each $x_{ij} \in \{0, 1\}$)
- Which of the following functions would be most apt for the decoder?

$$\hat{\mathbf{x}}_{\mathbf{i}} = \tanh(W^*\mathbf{h} + \mathbf{c})$$



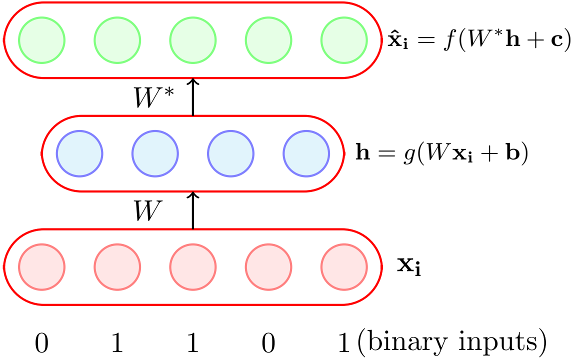
- Suppose all our inputs are binary (each $x_{ij} \in \{0, 1\}$)
- Which of the following functions would be most apt for the decoder?

$$\hat{\mathbf{x}}_\mathbf{i} = \tanh(W^*\mathbf{h} + \mathbf{c})$$
$$\hat{\mathbf{x}}_\mathbf{i} = W^*\mathbf{h} + \mathbf{c}$$



- Suppose all our inputs are binary (each $x_{ij} \in \{0, 1\}$)
- Which of the following functions would be most apt for the decoder?

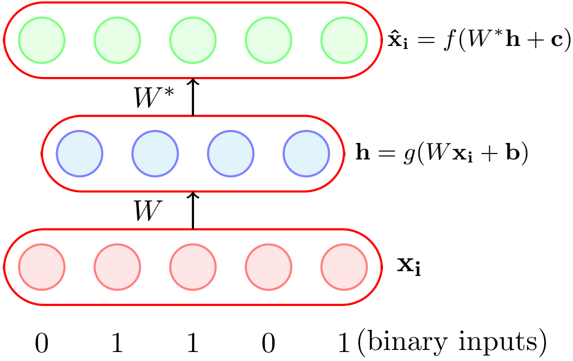
$$\hat{\mathbf{x}}_\mathbf{i} = \tanh(W^*\mathbf{h} + \mathbf{c})$$
$$\hat{\mathbf{x}}_\mathbf{i} = W^*\mathbf{h} + \mathbf{c}$$
$$\hat{\mathbf{x}}_\mathbf{i} = logistic(W^*\mathbf{h} + \mathbf{c})$$



- Suppose all our inputs are binary (each $x_{ij} \in \{0, 1\}$)
- Which of the following functions would be most apt for the decoder?

$$\hat{\mathbf{x}}_{\mathbf{i}} = \tanh(W^*\mathbf{h} + \mathbf{c})$$
$$\hat{\mathbf{x}}_{\mathbf{i}} = W^*\mathbf{h} + \mathbf{c}$$
$$\hat{\mathbf{x}}_{\mathbf{i}} = logistic(W^*\mathbf{h} + \mathbf{c})$$

- Logistic as it naturally restricts all outputs to be between 0 and 1

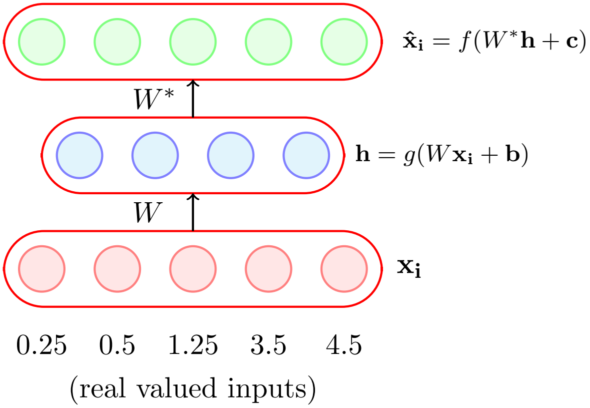


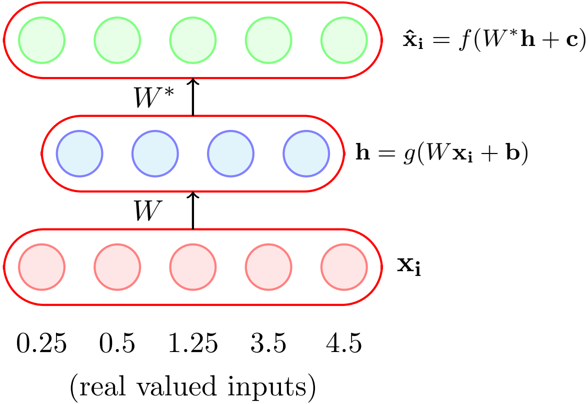
g is typically chosen as the sigmoid function

- Suppose all our inputs are binary (each $x_{ij} \in \{0, 1\}$)
- Which of the following functions would be most apt for the decoder?

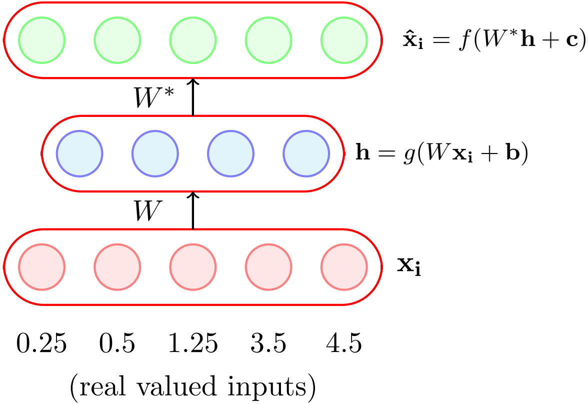
$$\hat{\mathbf{x}}_\mathbf{i} = \tanh(W^*\mathbf{h} + \mathbf{c})$$
$$\hat{\mathbf{x}}_\mathbf{i} = W^*\mathbf{h} + \mathbf{c}$$
$$\hat{\mathbf{x}}_\mathbf{i} = logistic(W^*\mathbf{h} + \mathbf{c})$$

- Logistic as it naturally restricts all outputs to be between 0 and 1

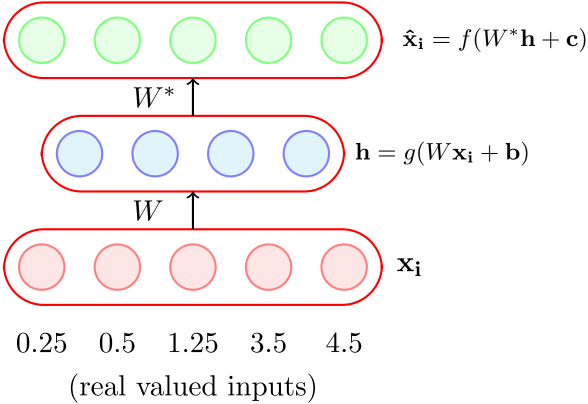

$\hat{\mathbf{x}}_\mathbf{i} = f(W^*\mathbf{h} + \mathbf{c})$
W^*
$\mathbf{h} = g(W\mathbf{x_i} + \mathbf{b})$
W
$\mathbf{x_i}$
0.25 0.5 1.25 3.5 4.5
(real valued inputs)



- Suppose all our inputs are real (each $x_{ij} \in \mathbb{R}$)

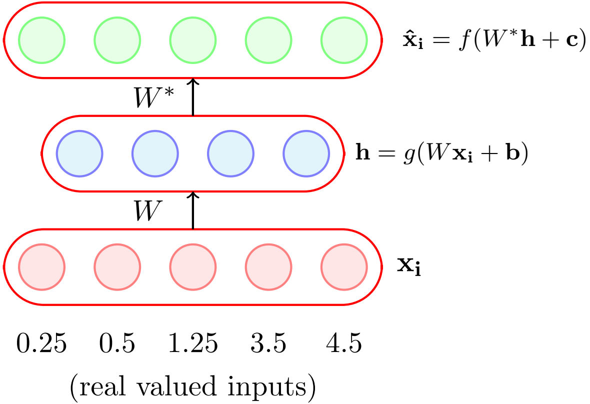


- Suppose all our inputs are real (each $x_{ij} \in \mathbb{R}$)
- Which of the following functions would be most apt for the decoder?



- Suppose all our inputs are real (each $x_{ij} \in \mathbb{R}$)
- Which of the following functions would be most apt for the decoder?

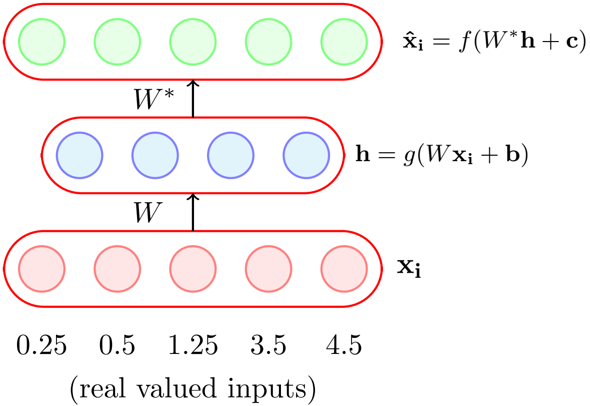
$$\hat{\mathbf{x}}_\mathbf{i} = \tanh(W^*\mathbf{h} + \mathbf{c})$$



- Suppose all our inputs are real (each $x_{ij} \in \mathbb{R}$)
- Which of the following functions would be most apt for the decoder?

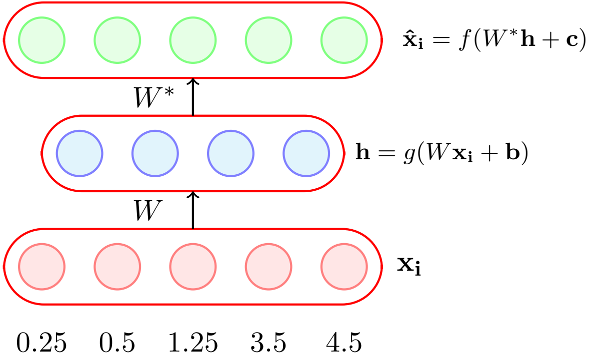
$$\hat{\mathbf{x}}_\mathbf{i} = \tanh(W^*\mathbf{h} + \mathbf{c})$$

$$\hat{\mathbf{x}}_\mathbf{i} = W^*\mathbf{h} + \mathbf{c}$$



- Suppose all our inputs are real (each $x_{ij} \in \mathbb{R}$)
- Which of the following functions would be most apt for the decoder?

$$\hat{\mathbf{x}}_\mathbf{i} = \tanh(W^*\mathbf{h} + \mathbf{c})$$
$$\hat{\mathbf{x}}_\mathbf{i} = W^*\mathbf{h} + \mathbf{c}$$
$$\hat{\mathbf{x}}_\mathbf{i} = \text{logistic}(W^*\mathbf{h} + \mathbf{c})$$

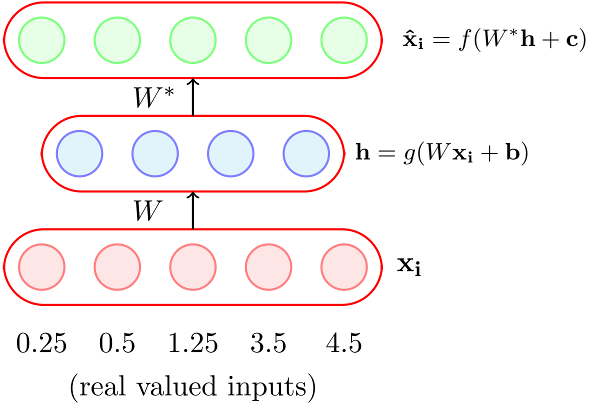


(real valued inputs)

- Suppose all our inputs are real (each $x_{ij} \in \mathbb{R}$)
- Which of the following functions would be most apt for the decoder?

$$\hat{\mathbf{x}}_\mathbf{i} = \tanh(W^*\mathbf{h} + \mathbf{c})$$
$$\hat{\mathbf{x}}_\mathbf{i} = W^*\mathbf{h} + \mathbf{c}$$
$$\hat{\mathbf{x}}_\mathbf{i} = \text{logistic}(W^*\mathbf{h} + \mathbf{c})$$

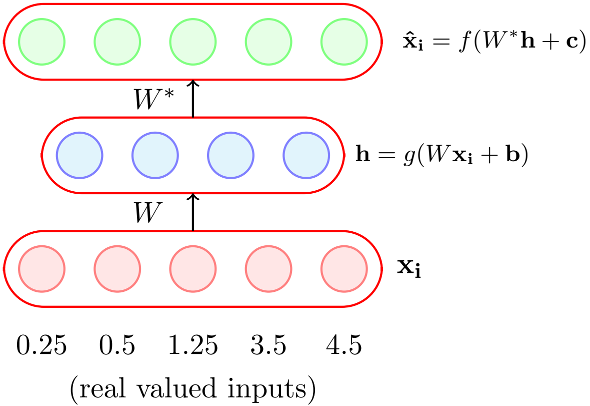
- What will logistic and tanh do?



- Suppose all our inputs are real (each $x_{ij} \in \mathbb{R}$)
- Which of the following functions would be most apt for the decoder?

$$\hat{\mathbf{x}}_\mathbf{i} = \tanh(W^*\mathbf{h} + \mathbf{c})$$
$$\hat{\mathbf{x}}_\mathbf{i} = W^*\mathbf{h} + \mathbf{c}$$
$$\hat{\mathbf{x}}_\mathbf{i} = \text{logistic}(W^*\mathbf{h} + \mathbf{c})$$

- What will logistic and tanh do?
- They will restrict the reconstructed $\hat{\mathbf{x}}_\mathbf{i}$ to lie between [0,1] or [-1,1] whereas we want $\hat{\mathbf{x}}_\mathbf{i} \in \mathbb{R}^n$



Again, g is typically chosen as the sigmoid function

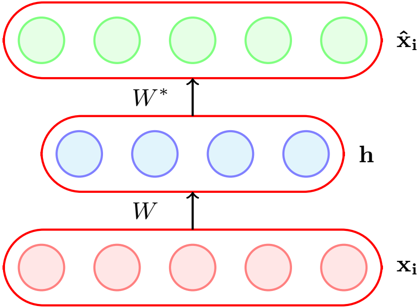
- Suppose all our inputs are real (each $x_{ij} \in \mathbb{R}$)
- Which of the following functions would be most apt for the decoder?

$$\hat{\mathbf{x}}_\mathbf{i} = \tanh(W^*\mathbf{h} + \mathbf{c})$$
$$\hat{\mathbf{x}}_\mathbf{i} = W^*\mathbf{h} + \mathbf{c}$$
$$\hat{\mathbf{x}}_\mathbf{i} = \text{logistic}(W^*\mathbf{h} + \mathbf{c})$$

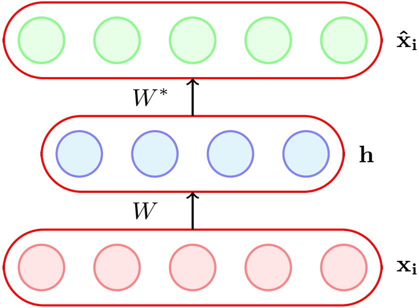
- What will logistic and tanh do?
- They will restrict the reconstructed $\hat{\mathbf{x}}_\mathbf{i}$ to lie between [0,1] or [-1,1] whereas we want $\hat{\mathbf{x}}_\mathbf{i} \in \mathbb{R}^n$

The Road Ahead

- Choice of $f(\mathbf{x_i})$ and $g(\mathbf{x_i})$
- Choice of loss function

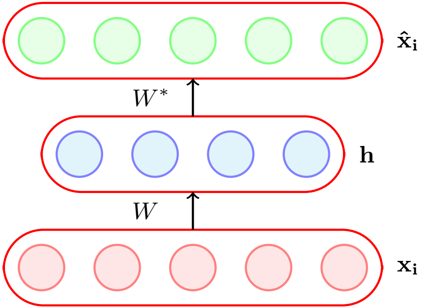


$$\mathbf{h} = g(W\mathbf{x}_\mathbf{i} + \mathbf{b})$$
$$\hat{\mathbf{x}}_\mathbf{i} = f(W^*\mathbf{h} + \mathbf{c})$$



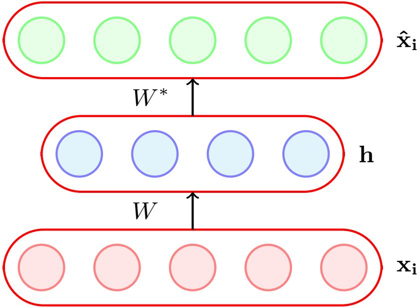
$$\mathbf{h} = g(W\mathbf{x}_\mathbf{i} + \mathbf{b})$$
$$\hat{\mathbf{x}}_\mathbf{i} = f(W^*\mathbf{h} + \mathbf{c})$$

- Consider the case when the inputs are real valued



$$\mathbf{h} = g(W\mathbf{x}_\mathbf{i} + \mathbf{b})$$
$$\hat{\mathbf{x}}_\mathbf{i} = f(W^*\mathbf{h} + \mathbf{c})$$

- Consider the case when the inputs are real valued
- The objective of the autoencoder is to reconstruct $\hat{\mathbf{x}}_\mathbf{i}$ to be as close to $\mathbf{x}_\mathbf{i}$ as possible

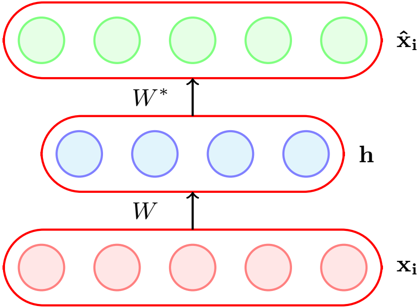


$$\mathbf{h} = g(W\mathbf{x}_\mathbf{i} + \mathbf{b})$$

$$\hat{\mathbf{x}}_\mathbf{i} = f(W^*\mathbf{h} + \mathbf{c})$$

- Consider the case when the inputs are real valued
- The objective of the autoencoder is to reconstruct $\hat{\mathbf{x}}_\mathbf{i}$ to be as close to $\mathbf{x}_\mathbf{i}$ as possible
- This can be formalized using the following objective function:

$$\min_{W,W^*,\mathbf{c},\mathbf{b}} \frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{n}(\hat{x}_{ij} - x_{ij})^2$$

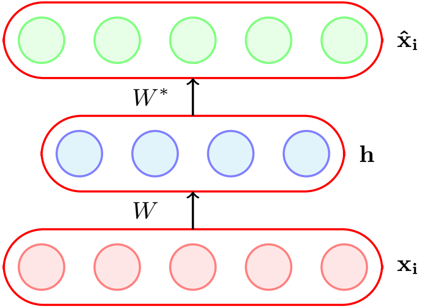


$$\mathbf{h} = g(W\mathbf{x}_\mathbf{i} + \mathbf{b})$$
$$\hat{\mathbf{x}}_\mathbf{i} = f(W^*\mathbf{h} + \mathbf{c})$$

- Consider the case when the inputs are real valued
- The objective of the autoencoder is to reconstruct $\hat{\mathbf{x}}_\mathbf{i}$ to be as close to $\mathbf{x}_\mathbf{i}$ as possible
- This can be formalized using the following objective function:

$$\min_{W,W^*,\mathbf{c},\mathbf{b}} \frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{n}(\hat{x}_{ij} - x_{ij})^2$$

$$i.e., \min_{W,W^*,\mathbf{c},\mathbf{b}} \frac{1}{m}\sum_{i=1}^{m}(\hat{\mathbf{x}}_\mathbf{i} - \mathbf{x}_\mathbf{i})^T(\hat{\mathbf{x}}_\mathbf{i} - \mathbf{x}_\mathbf{i})$$



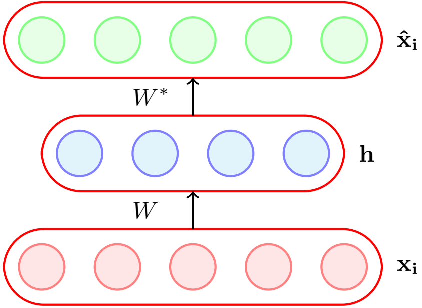
$$\mathbf{h} = g(W\mathbf{x}_\mathbf{i} + \mathbf{b})$$
$$\hat{\mathbf{x}}_\mathbf{i} = f(W^*\mathbf{h} + \mathbf{c})$$

- Consider the case when the inputs are real valued
- The objective of the autoencoder is to reconstruct $\hat{\mathbf{x}}_\mathbf{i}$ to be as close to $\mathbf{x}_\mathbf{i}$ as possible
- This can be formalized using the following objective function:

$$\min_{W,W^*,\mathbf{c},\mathbf{b}} \frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{n}(\hat{x}_{ij} - x_{ij})^2$$

$$i.e., \min_{W,W^*,\mathbf{c},\mathbf{b}} \frac{1}{m}\sum_{i=1}^{m}(\hat{\mathbf{x}}_\mathbf{i} - \mathbf{x}_\mathbf{i})^T(\hat{\mathbf{x}}_\mathbf{i} - \mathbf{x}_\mathbf{i})$$

- We can then train the autoencoder just like a regular feedforward network using backpropagation



$$\mathbf{h} = g(W\mathbf{x_i} + \mathbf{b})$$
$$\hat{\mathbf{x}}_\mathbf{i} = f(W^*\mathbf{h} + \mathbf{c})$$

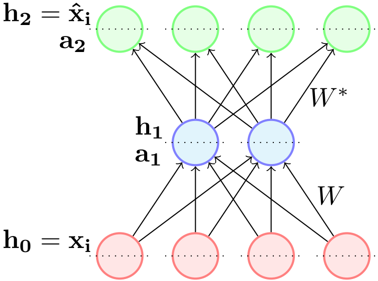
- Consider the case when the inputs are real valued
- The objective of the autoencoder is to reconstruct $\hat{\mathbf{x}}_\mathbf{i}$ to be as close to $\mathbf{x_i}$ as possible
- This can be formalized using the following objective function:

$$\min_{W,W^*,\mathbf{c},\mathbf{b}} \frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{n}(\hat{x}_{ij} - x_{ij})^2$$

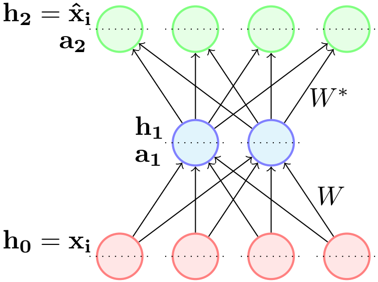
$$i.e., \min_{W,W^*,\mathbf{c},\mathbf{b}} \frac{1}{m}\sum_{i=1}^{m}(\hat{\mathbf{x}}_\mathbf{i} - \mathbf{x_i})^T(\hat{\mathbf{x}}_\mathbf{i} - \mathbf{x_i})$$

- We can then train the autoencoder just like a regular feedforward network using backpropagation
- All we need is a formula for $\frac{\partial \mathscr{L}(\theta)}{\partial W^*}$ and $\frac{\partial \mathscr{L}(\theta)}{\partial W}$ which we will see now

$$\mathscr{L}(\theta) = (\hat{\mathbf{x}}_\mathbf{i} - \mathbf{x}_\mathbf{i})^T(\hat{\mathbf{x}}_\mathbf{i} - \mathbf{x}_\mathbf{i})$$

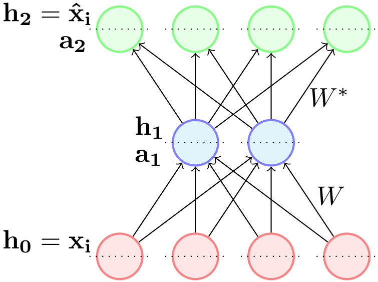


$$\mathscr{L}(\theta) = (\hat{\mathbf{x}}_\mathbf{i} - \mathbf{x}_\mathbf{i})^T(\hat{\mathbf{x}}_\mathbf{i} - \mathbf{x}_\mathbf{i})$$



- Note that the loss function is shown for only one training example.

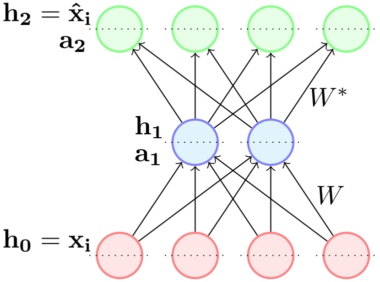
$$\mathscr{L}(\theta) = (\hat{\mathbf{x}}_\mathbf{i} - \mathbf{x}_\mathbf{i})^T(\hat{\mathbf{x}}_\mathbf{i} - \mathbf{x}_\mathbf{i})$$



- Note that the loss function is shown for only one training example.

- $$\frac{\partial \mathscr{L}(\theta)}{\partial W^*} = \frac{\partial \mathscr{L}(\theta)}{\partial \mathbf{h_2}} \boxed{\frac{\partial \mathbf{h_2}}{\partial \mathbf{a_2}} \frac{\partial \mathbf{a_2}}{\partial W^*}}$$

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

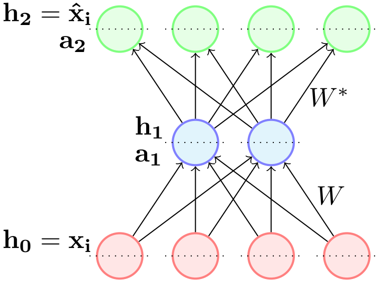


- Note that the loss function is shown for only one training example.

- $$\frac{\partial \mathscr{L}(\theta)}{\partial W^*} = \frac{\partial \mathscr{L}(\theta)}{\partial \mathbf{h_2}} \boxed{\frac{\partial \mathbf{h_2}}{\partial \mathbf{a_2}} \frac{\partial \mathbf{a_2}}{\partial W^*}}$$

- $$\frac{\partial \mathscr{L}(\theta)}{\partial W} = \frac{\partial \mathscr{L}(\theta)}{\partial \mathbf{h_2}} \boxed{\frac{\partial \mathbf{h_2}}{\partial \mathbf{a_2}} \frac{\partial \mathbf{a_2}}{\partial \mathbf{h_1}} \frac{\partial \mathbf{h_1}}{\partial \mathbf{a_1}} \frac{\partial \mathbf{a_1}}{\partial W}}$$

$$\mathscr{L}(\theta) = (\hat{\mathbf{x}}_\mathbf{i} - \mathbf{x}_\mathbf{i})^T(\hat{\mathbf{x}}_\mathbf{i} - \mathbf{x}_\mathbf{i})$$



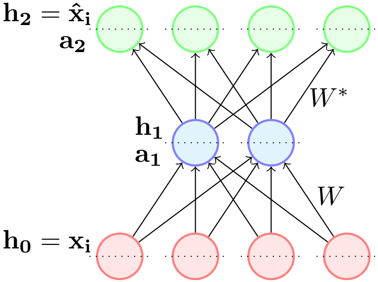
- Note that the loss function is shown for only one training example.

- $$\frac{\partial \mathscr{L}(\theta)}{\partial W^*} = \frac{\partial \mathscr{L}(\theta)}{\partial \mathbf{h_2}} \boxed{\frac{\partial \mathbf{h_2}}{\partial \mathbf{a_2}} \frac{\partial \mathbf{a_2}}{\partial W^*}}$$

- $$\frac{\partial \mathscr{L}(\theta)}{\partial W} = \frac{\partial \mathscr{L}(\theta)}{\partial \mathbf{h_2}} \boxed{\frac{\partial \mathbf{h_2}}{\partial \mathbf{a_2}} \frac{\partial \mathbf{a_2}}{\partial \mathbf{h_1}} \frac{\partial \mathbf{h_1}}{\partial \mathbf{a_1}} \frac{\partial \mathbf{a_1}}{\partial W}}$$

- We have already seen how to calculate the expression in the boxes when we learnt backpropagation

$$\mathscr{L}(\theta) = (\hat{\mathbf{x}}_\mathbf{i} - \mathbf{x}_\mathbf{i})^T(\hat{\mathbf{x}}_\mathbf{i} - \mathbf{x}_\mathbf{i})$$



- Note that the loss function is shown for only one training example.

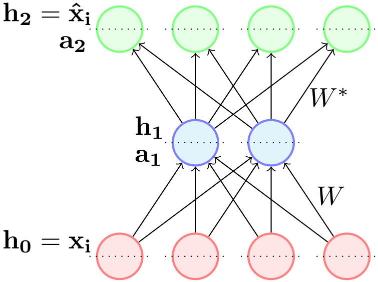
- $$\frac{\partial \mathscr{L}(\theta)}{\partial W^*} = \frac{\partial \mathscr{L}(\theta)}{\partial \mathbf{h_2}} \boxed{\frac{\partial \mathbf{h_2}}{\partial \mathbf{a_2}} \frac{\partial \mathbf{a_2}}{\partial W^*}}$$

- $$\frac{\partial \mathscr{L}(\theta)}{\partial W} = \frac{\partial \mathscr{L}(\theta)}{\partial \mathbf{h_2}} \boxed{\frac{\partial \mathbf{h_2}}{\partial \mathbf{a_2}} \frac{\partial \mathbf{a_2}}{\partial \mathbf{h_1}} \frac{\partial \mathbf{h_1}}{\partial \mathbf{a_1}} \frac{\partial \mathbf{a_1}}{\partial W}}$$

- We have already seen how to calculate the expression in the boxes when we learnt backpropagation

$$\frac{\partial \mathscr{L}(\theta)}{\partial \mathbf{h_2}} = \frac{\partial \mathscr{L}(\theta)}{\partial \hat{\mathbf{x}}_\mathbf{i}}$$

$$\mathscr{L}(\theta) = (\hat{\mathbf{x}}_\mathbf{i} - \mathbf{x}_\mathbf{i})^T(\hat{\mathbf{x}}_\mathbf{i} - \mathbf{x}_\mathbf{i})$$



- Note that the loss function is shown for only one training example.

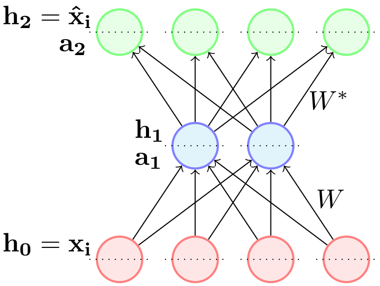
- $$\frac{\partial \mathscr{L}(\theta)}{\partial W^*} = \frac{\partial \mathscr{L}(\theta)}{\partial \mathbf{h_2}} \boxed{\frac{\partial \mathbf{h_2}}{\partial \mathbf{a_2}} \frac{\partial \mathbf{a_2}}{\partial W^*}}$$

- $$\frac{\partial \mathscr{L}(\theta)}{\partial W} = \frac{\partial \mathscr{L}(\theta)}{\partial \mathbf{h_2}} \boxed{\frac{\partial \mathbf{h_2}}{\partial \mathbf{a_2}} \frac{\partial \mathbf{a_2}}{\partial \mathbf{h_1}} \frac{\partial \mathbf{h_1}}{\partial \mathbf{a_1}} \frac{\partial \mathbf{a_1}}{\partial W}}$$

- We have already seen how to calculate the expression in the boxes when we learnt backpropagation

$$\begin{aligned}\frac{\partial \mathscr{L}(\theta)}{\partial \mathbf{h_2}} &= \frac{\partial \mathscr{L}(\theta)}{\partial \hat{\mathbf{x}}_\mathbf{i}} \\ &= \nabla_{\hat{\mathbf{x}}_\mathbf{i}} \{(\hat{\mathbf{x}}_\mathbf{i} - \mathbf{x}_\mathbf{i})^T(\hat{\mathbf{x}}_\mathbf{i} - \mathbf{x}_\mathbf{i})\}\end{aligned}$$

$$\mathscr{L}(\theta) = (\hat{\mathbf{x}}_\mathbf{i} - \mathbf{x}_\mathbf{i})^T(\hat{\mathbf{x}}_\mathbf{i} - \mathbf{x}_\mathbf{i})$$



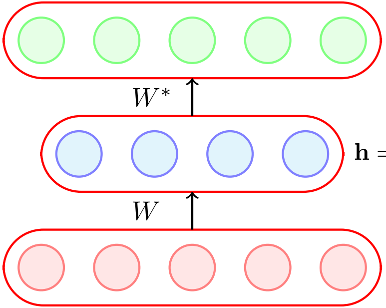
- Note that the loss function is shown for only one training example.

- $$\frac{\partial \mathscr{L}(\theta)}{\partial W^*} = \frac{\partial \mathscr{L}(\theta)}{\partial \mathbf{h_2}} \boxed{\frac{\partial \mathbf{h_2}}{\partial \mathbf{a_2}} \frac{\partial \mathbf{a_2}}{\partial W^*}}$$

- $$\frac{\partial \mathscr{L}(\theta)}{\partial W} = \frac{\partial \mathscr{L}(\theta)}{\partial \mathbf{h_2}} \boxed{\frac{\partial \mathbf{h_2}}{\partial \mathbf{a_2}} \frac{\partial \mathbf{a_2}}{\partial \mathbf{h_1}} \frac{\partial \mathbf{h_1}}{\partial \mathbf{a_1}} \frac{\partial \mathbf{a_1}}{\partial W}}$$

- We have already seen how to calculate the expression in the boxes when we learnt backpropagation

$$\begin{aligned}
\frac{\partial \mathscr{L}(\theta)}{\partial \mathbf{h_2}} &= \frac{\partial \mathscr{L}(\theta)}{\partial \hat{\mathbf{x}}_\mathbf{i}} \\
&= \nabla_{\hat{\mathbf{x}}_\mathbf{i}} \{(\hat{\mathbf{x}}_\mathbf{i} - \mathbf{x}_\mathbf{i})^T(\hat{\mathbf{x}}_\mathbf{i} - \mathbf{x}_\mathbf{i})\} \\
&= 2(\hat{\mathbf{x}}_\mathbf{i} - \mathbf{x}_\mathbf{i})
\end{aligned}$$

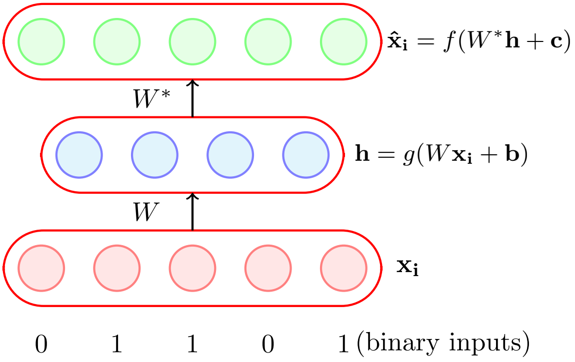


$\hat{\mathbf{x}}_{\mathbf{i}} = f(W^*\mathbf{h} + \mathbf{c})$

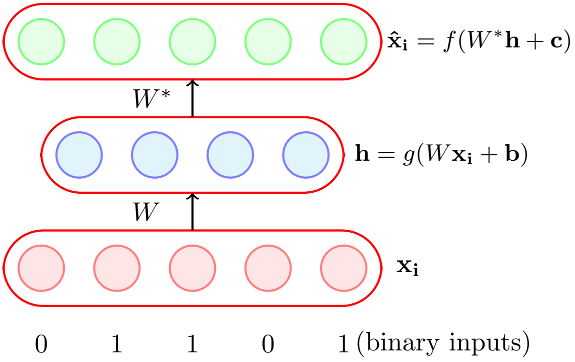
$\mathbf{h} = g(W\mathbf{x}_{\mathbf{i}} + \mathbf{b})$

$\mathbf{x}_{\mathbf{i}}$

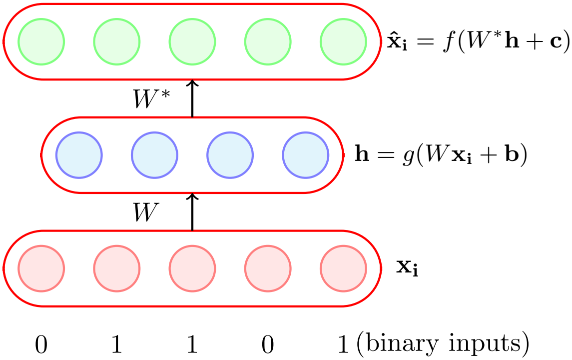
0 1 1 0 1 (binary inputs)



- Consider the case when the inputs are binary

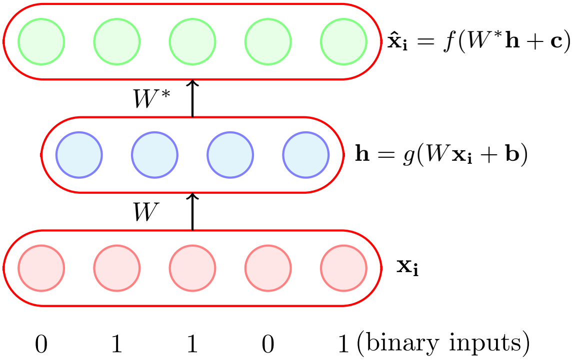


- Consider the case when the inputs are binary
- We use a sigmoid decoder which will produce outputs between 0 and 1, and can be interpreted as probabilities.



- Consider the case when the inputs are binary
- We use a sigmoid decoder which will produce outputs between 0 and 1, and can be interpreted as probabilities.
- For a single n-dimensional i^{th} input we can use the following loss function

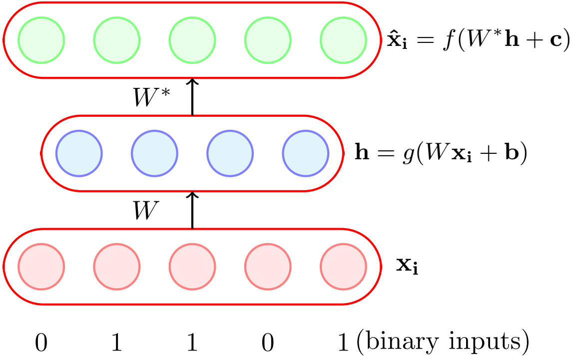
$$\min\{-\sum_{j=1}^{n}(x_{ij}\log\hat{x}_{ij} + (1 - x_{ij})\log(1 - \hat{x}_{ij}))\}$$



What value of $\hat{x}_{ij}$ will minimize this function?

- Consider the case when the inputs are binary
- We use a sigmoid decoder which will produce outputs between 0 and 1, and can be interpreted as probabilities.
- For a single n-dimensional i^{th} input we can use the following loss function

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

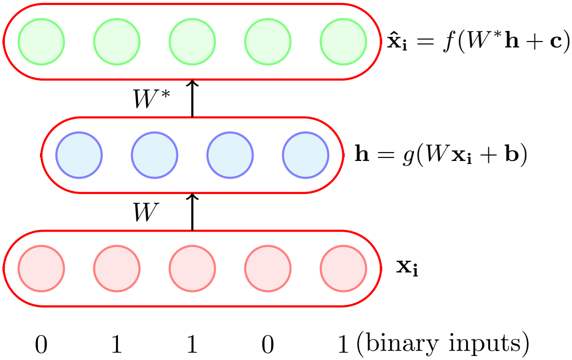


What value of $\hat{x}_{ij}$ will minimize this function?

- If $x_{ij} = 1$?

- Consider the case when the inputs are binary
- We use a sigmoid decoder which will produce outputs between 0 and 1, and can be interpreted as probabilities.
- For a single n-dimensional i^{th} input we can use the following loss function

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

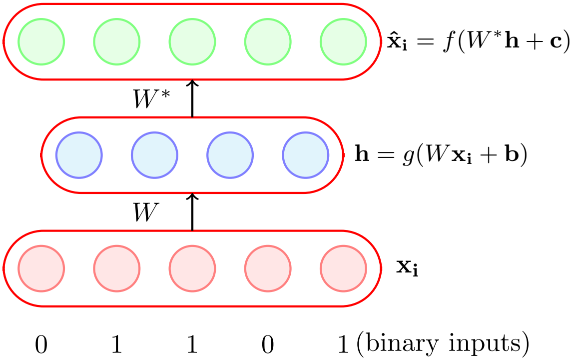


What value of $\hat{x}_{ij}$ will minimize this function?

- If $x_{ij} = 1$?
- If $x_{ij} = 0$?

- Consider the case when the inputs are binary
- We use a sigmoid decoder which will produce outputs between 0 and 1, and can be interpreted as probabilities.
- For a single n-dimensional i^{th} input we can use the following loss function

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



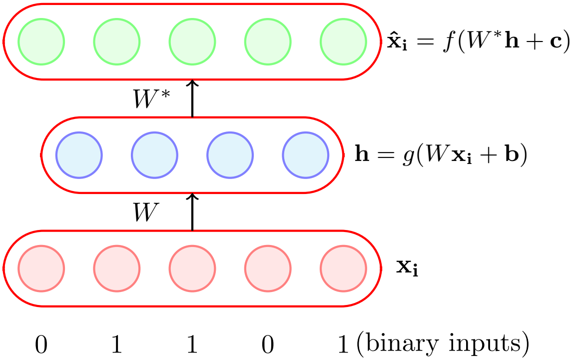
What value of $\hat{x}_{ij}$ will minimize this function?

- If $x_{ij} = 1$?
- If $x_{ij} = 0$?

- Consider the case when the inputs are binary
- We use a sigmoid decoder which will produce outputs between 0 and 1, and can be interpreted as probabilities.
- For a single n-dimensional i^{th} input we can use the following loss function

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

- Again we need is a formula for $\frac{\partial\mathscr{L}(\theta)}{\partial W^*}$ and $\frac{\partial\mathscr{L}(\theta)}{\partial W}$ to use backpropagation



What value of $\hat{x}_{ij}$ will minimize this function?

- If $x_{ij} = 1$?
- If $x_{ij} = 0$?

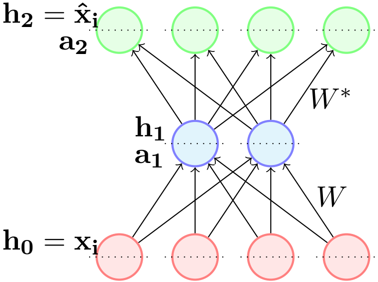
Indeed the above function will be minimized when $\hat{x}_{ij} = x_{ij}$!

- Consider the case when the inputs are binary
- We use a sigmoid decoder which will produce outputs between 0 and 1, and can be interpreted as probabilities.
- For a single n-dimensional i^{th} input we can use the following loss function

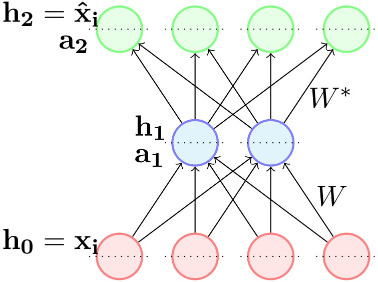
$$\min\{-\sum_{j=1}^{n}(x_{ij}\log\hat{x}_{ij} + (1 - x_{ij})\log(1 - \hat{x}_{ij}))\}$$

- Again we need is a formula for $\frac{\partial \mathscr{L}(\theta)}{\partial W^*}$ and $\frac{\partial \mathscr{L}(\theta)}{\partial W}$ to use backpropagation

$$\mathscr{L}(\theta) = -\sum_{j=1}^{n} (x_{ij} \log \hat{x}_{ij} + (1 - x_{ij}) \log(1 - \hat{x}_{ij}))$$

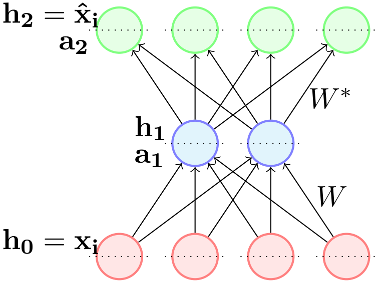


$$\mathscr{L}(\theta) = -\sum_{j=1}^{n}(x_{ij}\log\hat{x}_{ij} + (1-x_{ij})\log(1-\hat{x}_{ij}))$$



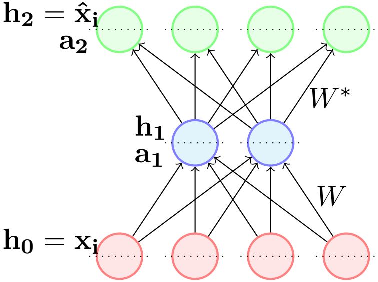
- $$\frac{\partial \mathscr{L}(\theta)}{\partial W^*} = \frac{\partial \mathscr{L}(\theta)}{\partial \mathbf{h_2}} \frac{\partial \mathbf{h_2}}{\partial \mathbf{a_2}} \boxed{\frac{\partial \mathbf{a_2}}{\partial W^*}}$$

$$\mathscr{L}(\theta) = -\sum_{j=1}^{n}(x_{ij}\log\hat{x}_{ij} + (1-x_{ij})\log(1-\hat{x}_{ij}))$$



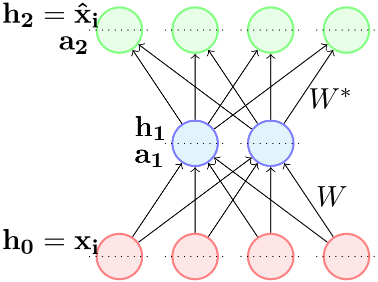
- $$\frac{\partial \mathscr{L}(\theta)}{\partial W^*} = \frac{\partial \mathscr{L}(\theta)}{\partial \mathbf{h_2}}\frac{\partial \mathbf{h_2}}{\partial \mathbf{a_2}}\boxed{\frac{\partial \mathbf{a_2}}{\partial W^*}}$$
- $$\frac{\partial \mathscr{L}(\theta)}{\partial W} = \frac{\partial \mathscr{L}(\theta)}{\partial \mathbf{h_2}}\frac{\partial \mathbf{h_2}}{\partial \mathbf{a_2}}\boxed{\frac{\partial \mathbf{a_2}}{\partial \mathbf{h_1}}\frac{\partial \mathbf{h_1}}{\partial \mathbf{a_1}}\frac{\partial \mathbf{a_1}}{\partial W}}$$

$$\mathscr{L}(\theta) = -\sum_{j=1}^{n} (x_{ij} \log \hat{x}_{ij} + (1 - x_{ij}) \log(1 - \hat{x}_{ij}))$$



- $$\frac{\partial \mathscr{L}(\theta)}{\partial W^*} = \frac{\partial \mathscr{L}(\theta)}{\partial \mathbf{h_2}} \frac{\partial \mathbf{h_2}}{\partial \mathbf{a_2}} \boxed{\frac{\partial \mathbf{a_2}}{\partial W^*}}$$
- $$\frac{\partial \mathscr{L}(\theta)}{\partial W} = \frac{\partial \mathscr{L}(\theta)}{\partial \mathbf{h_2}} \frac{\partial \mathbf{h_2}}{\partial \mathbf{a_2}} \boxed{\frac{\partial \mathbf{a_2}}{\partial \mathbf{h_1}} \frac{\partial \mathbf{h_1}}{\partial \mathbf{a_1}} \frac{\partial \mathbf{a_1}}{\partial W}}$$
- We have already seen how to calculate the expressions in the square boxes when we learnt BP

$$\mathscr{L}(\theta) = -\sum_{j=1}^{n} (x_{ij} \log \hat{x}_{ij} + (1 - x_{ij}) \log(1 - \hat{x}_{ij}))$$

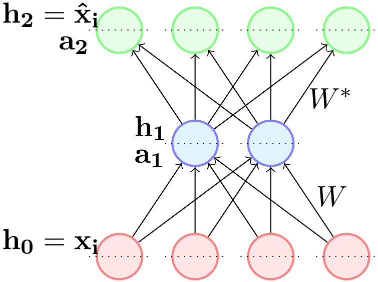


- $$\frac{\partial \mathscr{L}(\theta)}{\partial W^*} = \frac{\partial \mathscr{L}(\theta)}{\partial \mathbf{h_2}} \frac{\partial \mathbf{h_2}}{\partial \mathbf{a_2}} \boxed{\frac{\partial \mathbf{a_2}}{\partial W^*}}$$
- $$\frac{\partial \mathscr{L}(\theta)}{\partial W} = \frac{\partial \mathscr{L}(\theta)}{\partial \mathbf{h_2}} \frac{\partial \mathbf{h_2}}{\partial \mathbf{a_2}} \boxed{\frac{\partial \mathbf{a_2}}{\partial \mathbf{h_1}} \frac{\partial \mathbf{h_1}}{\partial \mathbf{a_1}} \frac{\partial \mathbf{a_1}}{\partial W}}$$
- We have already seen how to calculate the expressions in the square boxes when we learnt BP
- The first two terms on RHS can be computed as:

$$\frac{\partial \mathscr{L}(\theta)}{\partial h_{2j}} = -\frac{x_{ij}}{\hat{x}_{ij}} + \frac{1 - x_{ij}}{1 - \hat{x}_{ij}}$$

$$\frac{\partial h_{2j}}{\partial a_{2j}} = \sigma(a_{2j})(1 - \sigma(a_{2j}))$$

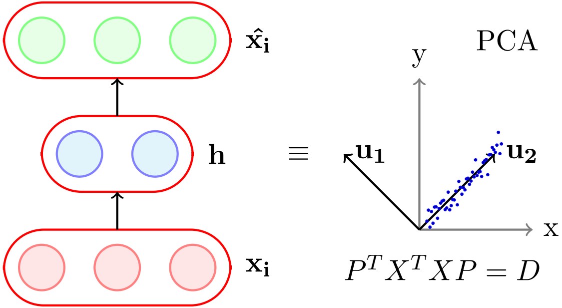
$$\mathscr{L}(\theta) = -\sum_{j=1}^{n} (x_{ij} \log \hat{x}_{ij} + (1 - x_{ij}) \log(1 - \hat{x}_{ij}))$$



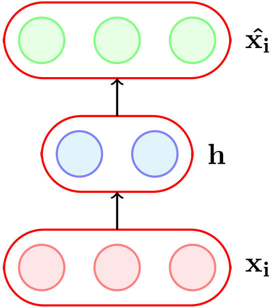
$$\frac{\partial \mathscr{L}(\theta)}{\partial \mathbf{h_2}} = \begin{pmatrix} \frac{\partial \mathscr{L}(\theta)}{\partial h_{21}} \\ \frac{\partial \mathscr{L}(\theta)}{\partial h_{22}} \\ \vdots \\ \frac{\partial \mathscr{L}(\theta)}{\partial h_{2n}} \end{pmatrix}$$

- $$\frac{\partial \mathscr{L}(\theta)}{\partial W^*} = \frac{\partial \mathscr{L}(\theta)}{\partial \mathbf{h_2}} \frac{\partial \mathbf{h_2}}{\partial \mathbf{a_2}} \boxed{\frac{\partial \mathbf{a_2}}{\partial W^*}}$$
- $$\frac{\partial \mathscr{L}(\theta)}{\partial W} = \frac{\partial \mathscr{L}(\theta)}{\partial \mathbf{h_2}} \frac{\partial \mathbf{h_2}}{\partial \mathbf{a_2}} \boxed{\frac{\partial \mathbf{a_2}}{\partial \mathbf{h_1}} \frac{\partial \mathbf{h_1}}{\partial \mathbf{a_1}} \frac{\partial \mathbf{a_1}}{\partial W}}$$
- We have already seen how to calculate the expressions in the square boxes when we learnt BP
- The first two terms on RHS can be computed as:
$$\frac{\partial \mathscr{L}(\theta)}{\partial h_{2j}} = -\frac{x_{ij}}{\hat{x}_{ij}} + \frac{1 - x_{ij}}{1 - \hat{x}_{ij}}$$
$$\frac{\partial h_{2j}}{\partial a_{2j}} = \sigma(a_{2j})(1 - \sigma(a_{2j}))$$

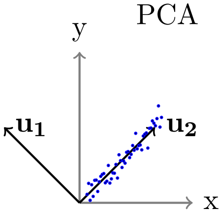
Module 7.2: Link between PCA and Autoencoders



- We will now see that the encoder part of an autoencoder is equivalent to PCA if we

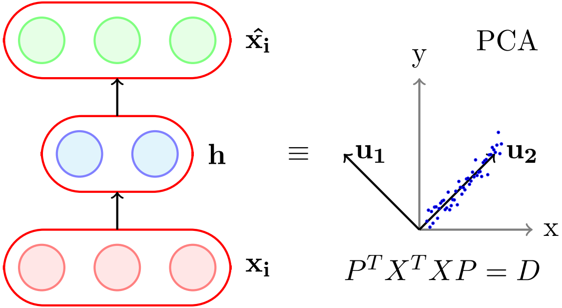


$\equiv$

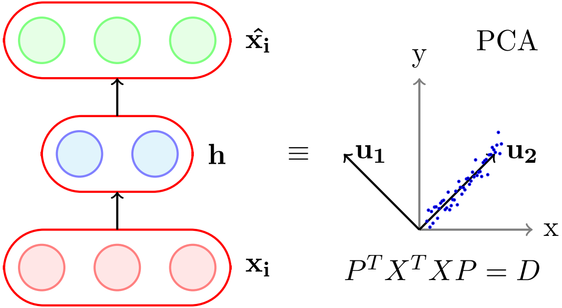


$$P^T X^T X P = D$$

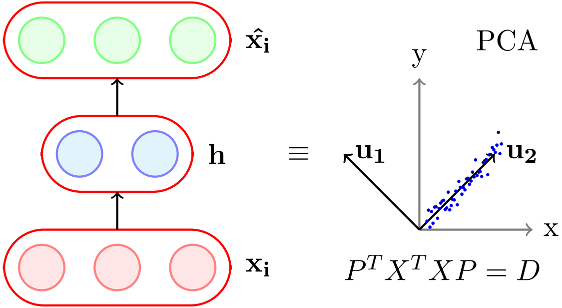
- We will now see that the encoder part of an autoencoder is equivalent to PCA if we
  - use a linear encoder



- We will now see that the encoder part of an autoencoder is equivalent to PCA if we
  - use a linear encoder
  - use a linear decoder



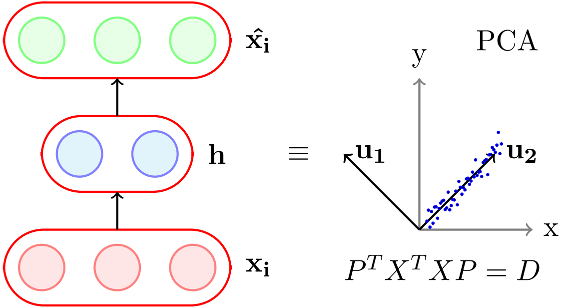
- We will now see that the encoder part of an autoencoder is equivalent to PCA if we
  - use a linear encoder
  - use a linear decoder
  - use squared error loss function



$$P^T X^T X P = D$$

- We will now see that the encoder part of an autoencoder is equivalent to PCA if we
  - use a linear encoder
  - use a linear decoder
  - use squared error loss function
  - normalize the inputs to

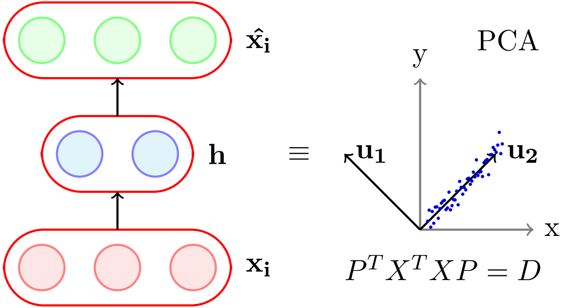
$$\hat{x}_{ij} = \frac{1}{\sqrt{m}}\left(x_{ij} - \frac{1}{m}\sum_{k=1}^{m} x_{kj}\right)$$



$$P^T X^T X P = D$$

- First let us consider the implication of normalizing the inputs to

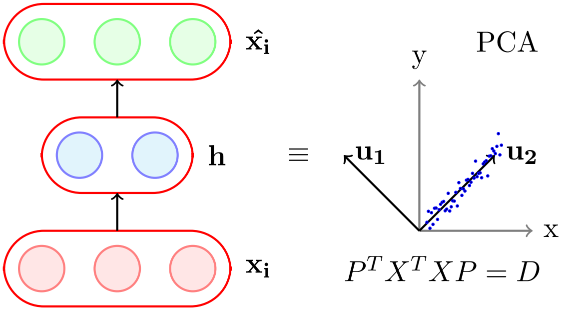
$$\hat{x}_{ij} = \frac{1}{\sqrt{m}}\left(x_{ij} - \frac{1}{m}\sum_{k=1}^{m} x_{kj}\right)$$



- First let us consider the implication of normalizing the inputs to

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

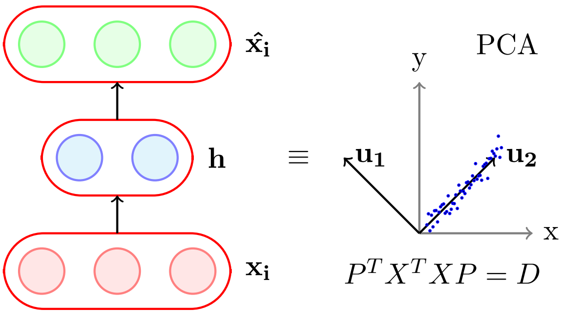
- The operation in the bracket ensures that the data now has 0 mean along each dimension j (we are subtracting the mean)



- First let us consider the implication of normalizing the inputs to

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

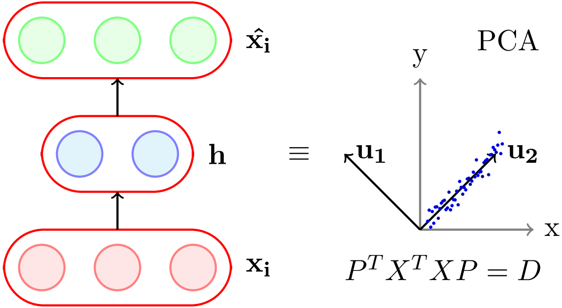
- The operation in the bracket ensures that the data now has 0 mean along each dimension j (we are subtracting the mean)
- Let X' be this zero mean data matrix then what the above normalization gives us is $X = \frac{1}{\sqrt{m}}X'$



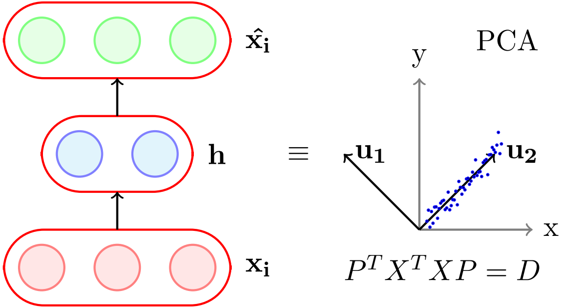
- First let us consider the implication of normalizing the inputs to

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

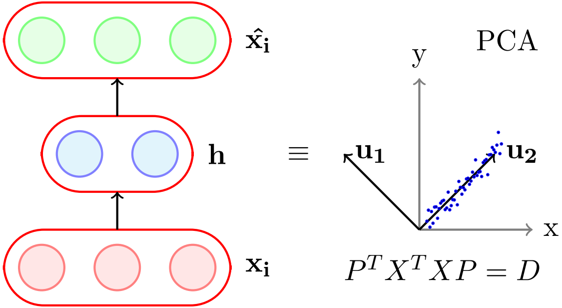
- The operation in the bracket ensures that the data now has 0 mean along each dimension j (we are subtracting the mean)
- Let X' be this zero mean data matrix then what the above normalization gives us is $X = \frac{1}{\sqrt{m}} X'$
- Now $(X)^T X = \frac{1}{m}(X')^T X'$ is the covariance matrix (recall that covariance matrix plays an important role in PCA)


$\hat{\mathbf{x}}_\mathbf{i}$
$\mathbf{h}$
$\mathbf{x_i}$
$\equiv$
PCA
y
x
$\mathbf{u_1}$
$\mathbf{u_2}$

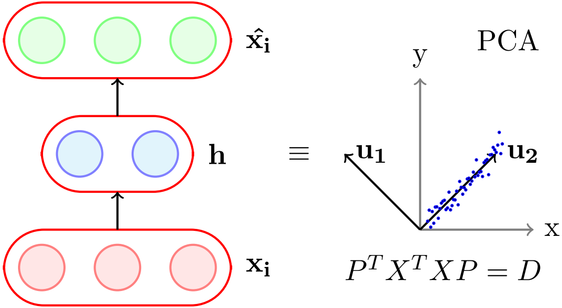
$$P^T X^T X P = D$$



- First we will show that if we use linear decoder and a squared error loss function then

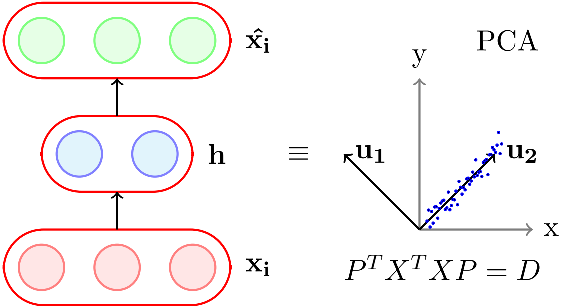


- First we will show that if we use linear decoder and a squared error loss function then
- The optimal solution to the following objective function



- First we will show that if we use linear decoder and a squared error loss function then
- The optimal solution to the following objective function

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



- First we will show that if we use linear decoder and a squared error loss function then
- The optimal solution to the following objective function

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

is obtained when we use a linear encoder.

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

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

- This is equivalent to

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

- This is equivalent to

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

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

- This is equivalent to

$$\min_{W^* H} (\|X - HW^*\|_F)^2 \qquad\qquad \|A\|_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij}^2}$$

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

- This is equivalent to

$$\min_{W^* H} (\|X - HW^*\|_F)^2 \qquad\qquad \|A\|_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij}^2}$$

(just writing the expression (1) in matrix form and using the definition of $||A||_F$) (we are ignoring the biases)

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

- This is equivalent to

$$\min_{W^*H} (\|X - HW^*\|_F)^2 \qquad \qquad \|A\|_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij}^2}$$

(just writing the expression (1) in matrix form and using the definition of $||A||_F$) (we are ignoring the biases)

- From SVD we know that optimal solution to the above problem is given by

$$HW^* = U_{.,\leq k} \Sigma_{k,k} V^T_{.,\leq k}$$

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

- This is equivalent to

$$\min_{W^*H} (\|X - HW^*\|_F)^2 \qquad \|A\|_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij}^2}$$

(just writing the expression (1) in matrix form and using the definition of $||A||_F$) (we are ignoring the biases)

- From SVD we know that optimal solution to the above problem is given by

$$HW^* = U_{.,\leq k} \Sigma_{k,k} V^T_{.,\leq k}$$

- By matching variables one possible solution is

$$H = U_{.,\leq k} \Sigma_{k,k}$$
$$W^* = V^T_{.,\leq k}$$

We will now show that H is a linear encoding and find an expression for the encoder weights W

We will now show that H is a linear encoding and find an expression for the encoder weights W

$$H = U_{.,\leq k}\Sigma_{k,k}$$

We will now show that H is a linear encoding and find an expression for the encoder weights W

$$
\begin{aligned}
H &= U_{.,\leq k}\Sigma_{k,k} \\
&= (XX^T)(XX^T)^{-1}U_{.,\leq K}\Sigma_{k,k} && \textit{(pre-multiplying } (XX^T)(XX^T)^{-1} = I\textit{)}
\end{aligned}
$$

We will now show that H is a linear encoding and find an expression for the encoder weights W

$$\begin{aligned} H &= U_{.,\leq k}\Sigma_{k,k} \\ &= (XX^T)(XX^T)^{-1}U_{.,\leq K}\Sigma_{k,k} && \textit{(pre-multiplying } (XX^T)(XX^T)^{-1} = I\textit{)} \\ &= (XV\Sigma^T U^T)(U\Sigma V^T V\Sigma^T U^T)^{-1}U_{.,\leq k}\Sigma_{k,k} && \textit{(using } X = U\Sigma V^T\textit{)} \end{aligned}$$

We will now show that H is a linear encoding and find an expression for the encoder weights W

$$
\begin{aligned}
H &= U_{.,\leq k}\Sigma_{k,k} \\
&= (XX^T)(XX^T)^{-1}U_{.,\leq K}\Sigma_{k,k} && \textit{(pre-multiplying } (XX^T)(XX^T)^{-1} = I\textit{)} \\
&= (XV\Sigma^T U^T)(U\Sigma V^T V\Sigma^T U^T)^{-1}U_{.,\leq k}\Sigma_{k,k} && \textit{(using } X = U\Sigma V^T\textit{)} \\
&= XV\Sigma^T U^T (U\Sigma\Sigma^T U^T)^{-1}U_{.,\leq k}\Sigma_{k,k} && (V^T V = I)
\end{aligned}
$$

We will now show that H is a linear encoding and find an expression for the encoder weights W

$$
\begin{aligned}
H &= U_{.,\leq k}\Sigma_{k,k} \\
&= (XX^T)(XX^T)^{-1}U_{.,\leq K}\Sigma_{k,k} && \textit{(pre-multiplying } (XX^T)(XX^T)^{-1} = I\textit{)} \\
&= (XV\Sigma^T U^T)(U\Sigma V^T V\Sigma^T U^T)^{-1}U_{.,\leq k}\Sigma_{k,k} && \textit{(using } X = U\Sigma V^T\textit{)} \\
&= XV\Sigma^T U^T(U\Sigma\Sigma^T U^T)^{-1}U_{.,\leq k}\Sigma_{k,k} && (V^T V = I) \\
&= XV\Sigma^T U^T U(\Sigma\Sigma^T)^{-1}U^T U_{.,\leq k}\Sigma_{k,k} && ((ABC)^{-1} = C^{-1}B^{-1}A^{-1})
\end{aligned}
$$

We will now show that H is a linear encoding and find an expression for the encoder weights W

$$
\begin{aligned}
H &= U_{.,\leq k}\Sigma_{k,k} \\
&= (XX^T)(XX^T)^{-1}U_{.,\leq K}\Sigma_{k,k} && \textit{(pre-multiplying } (XX^T)(XX^T)^{-1} = I\textit{)} \\
&= (XV\Sigma^T U^T)(U\Sigma V^T V\Sigma^T U^T)^{-1}U_{.,\leq k}\Sigma_{k,k} && \textit{(using } X = U\Sigma V^T\textit{)} \\
&= XV\Sigma^T U^T(U\Sigma\Sigma^T U^T)^{-1}U_{.,\leq k}\Sigma_{k,k} && (V^T V = I) \\
&= XV\Sigma^T U^T U(\Sigma\Sigma^T)^{-1}U^T U_{.,\leq k}\Sigma_{k,k} && ((ABC)^{-1} = C^{-1}B^{-1}A^{-1}) \\
&= XV\Sigma^T(\Sigma\Sigma^T)^{-1}U^T U_{.,\leq k}\Sigma_{k,k} && (U^T U = I)
\end{aligned}
$$

We will now show that H is a linear encoding and find an expression for the encoder weights W

$$
\begin{aligned}
H &= U_{.,\leq k}\Sigma_{k,k} \\
&= (XX^T)(XX^T)^{-1}U_{.,\leq K}\Sigma_{k,k} && \textit{(pre-multiplying } (XX^T)(XX^T)^{-1} = I\textit{)} \\
&= (XV\Sigma^T U^T)(U\Sigma V^T V\Sigma^T U^T)^{-1}U_{.,\leq k}\Sigma_{k,k} && \textit{(using } X = U\Sigma V^T\textit{)} \\
&= XV\Sigma^T U^T(U\Sigma\Sigma^T U^T)^{-1}U_{.,\leq k}\Sigma_{k,k} && (V^T V = I) \\
&= XV\Sigma^T U^T U(\Sigma\Sigma^T)^{-1}U^T U_{.,\leq k}\Sigma_{k,k} && ((ABC)^{-1} = C^{-1}B^{-1}A^{-1}) \\
&= XV\Sigma^T(\Sigma\Sigma^T)^{-1}U^T U_{.,\leq k}\Sigma_{k,k} && (U^T U = I) \\
&= XV\Sigma^T\Sigma^{T^{-1}}\Sigma^{-1}U^T U_{.,\leq k}\Sigma_{k,k} && ((AB)^{-1} = B^{-1}A^{-1})
\end{aligned}
$$

We will now show that H is a linear encoding and find an expression for the encoder weights W

$$
\begin{aligned}
H &= U_{.,\leq k}\Sigma_{k,k} \\
&= (XX^T)(XX^T)^{-1}U_{.,\leq K}\Sigma_{k,k} && \textit{(pre-multiplying } (XX^T)(XX^T)^{-1} = I\textit{)} \\
&= (XV\Sigma^T U^T)(U\Sigma V^T V\Sigma^T U^T)^{-1}U_{.,\leq k}\Sigma_{k,k} && \textit{(using } X = U\Sigma V^T\textit{)} \\
&= XV\Sigma^T U^T(U\Sigma\Sigma^T U^T)^{-1}U_{.,\leq k}\Sigma_{k,k} && (V^TV = I) \\
&= XV\Sigma^T U^T U(\Sigma\Sigma^T)^{-1}U^T U_{.,\leq k}\Sigma_{k,k} && ((ABC)^{-1} = C^{-1}B^{-1}A^{-1}) \\
&= XV\Sigma^T(\Sigma\Sigma^T)^{-1}U^T U_{.,\leq k}\Sigma_{k,k} && (U^TU = I) \\
&= XV\Sigma^T\Sigma^{T^{-1}}\Sigma^{-1}U^T U_{.,\leq k}\Sigma_{k,k} && ((AB)^{-1} = B^{-1}A^{-1}) \\
&= XV\Sigma^{-1}I_{.,\leq k}\Sigma_{k,k} && (U^TU_{.,\leq k} = I_{.,\leq k})
\end{aligned}
$$

We will now show that H is a linear encoding and find an expression for the encoder weights W

$$
\begin{aligned}
H &= U_{.,\leq k}\Sigma_{k,k} \\
&= (XX^T)(XX^T)^{-1}U_{.,\leq K}\Sigma_{k,k} && \textit{(pre-multiplying } (XX^T)(XX^T)^{-1} = I\textit{)} \\
&= (XV\Sigma^T U^T)(U\Sigma V^T V\Sigma^T U^T)^{-1}U_{.,\leq k}\Sigma_{k,k} && \textit{(using } X = U\Sigma V^T\textit{)} \\
&= XV\Sigma^T U^T (U\Sigma\Sigma^T U^T)^{-1}U_{.,\leq k}\Sigma_{k,k} && (V^T V = I) \\
&= XV\Sigma^T U^T U(\Sigma\Sigma^T)^{-1}U^T U_{.,\leq k}\Sigma_{k,k} && ((ABC)^{-1} = C^{-1}B^{-1}A^{-1}) \\
&= XV\Sigma^T(\Sigma\Sigma^T)^{-1}U^T U_{.,\leq k}\Sigma_{k,k} && (U^T U = I) \\
&= XV\Sigma^T\Sigma^{T^{-1}}\Sigma^{-1}U^T U_{.,\leq k}\Sigma_{k,k} && ((AB)^{-1} = B^{-1}A^{-1}) \\
&= XV\Sigma^{-1}I_{.,\leq k}\Sigma_{k,k} && (U^T U_{.,\leq k} = I_{.,\leq k}) \\
&= XVI_{.,\leq k} && (\Sigma^{-1}I_{.,\leq k} = \Sigma^{-1}_{k,k})
\end{aligned}
$$

We will now show that H is a linear encoding and find an expression for the encoder weights W

$$
\begin{aligned}
H &= U_{.,\leq k}\Sigma_{k,k} \\
&= (XX^T)(XX^T)^{-1}U_{.,\leq K}\Sigma_{k,k} && \textit{(pre-multiplying } (XX^T)(XX^T)^{-1} = I\textit{)} \\
&= (XV\Sigma^T U^T)(U\Sigma V^T V\Sigma^T U^T)^{-1}U_{.,\leq k}\Sigma_{k,k} && \textit{(using } X = U\Sigma V^T\textit{)} \\
&= XV\Sigma^T U^T(U\Sigma\Sigma^T U^T)^{-1}U_{.,\leq k}\Sigma_{k,k} && (V^TV = I) \\
&= XV\Sigma^T U^T U(\Sigma\Sigma^T)^{-1}U^T U_{.,\leq k}\Sigma_{k,k} && ((ABC)^{-1} = C^{-1}B^{-1}A^{-1}) \\
&= XV\Sigma^T(\Sigma\Sigma^T)^{-1}U^T U_{.,\leq k}\Sigma_{k,k} && (U^TU = I) \\
&= XV\Sigma^T\Sigma^{T^{-1}}\Sigma^{-1}U^T U_{.,\leq k}\Sigma_{k,k} && ((AB)^{-1} = B^{-1}A^{-1}) \\
&= XV\Sigma^{-1}I_{.,\leq k}\Sigma_{k,k} && (U^TU_{.,\leq k} = I_{.,\leq k}) \\
&= XVI_{.,\leq k} && (\Sigma^{-1}I_{.,\leq k} = \Sigma^{-1}_{k,k}) \\
H &= XV_{.,\leq k}
\end{aligned}
$$

We will now show that H is a linear encoding and find an expression for the encoder weights W

$$
\begin{aligned}
H &= U_{.,\leq k}\Sigma_{k,k} \\
&= (XX^T)(XX^T)^{-1}U_{.,\leq K}\Sigma_{k,k} && \textit{(pre-multiplying } (XX^T)(XX^T)^{-1} = I) \\
&= (XV\Sigma^T U^T)(U\Sigma V^T V\Sigma^T U^T)^{-1}U_{.,\leq k}\Sigma_{k,k} && \textit{(using } X = U\Sigma V^T) \\
&= XV\Sigma^T U^T(U\Sigma\Sigma^T U^T)^{-1}U_{.,\leq k}\Sigma_{k,k} && (V^T V = I) \\
&= XV\Sigma^T U^T U(\Sigma\Sigma^T)^{-1}U^T U_{.,\leq k}\Sigma_{k,k} && ((ABC)^{-1} = C^{-1}B^{-1}A^{-1}) \\
&= XV\Sigma^T(\Sigma\Sigma^T)^{-1}U^T U_{.,\leq k}\Sigma_{k,k} && (U^T U = I) \\
&= XV\Sigma^T\Sigma^{T^{-1}}\Sigma^{-1}U^T U_{.,\leq k}\Sigma_{k,k} && ((AB)^{-1} = B^{-1}A^{-1}) \\
&= XV\Sigma^{-1}I_{.,\leq k}\Sigma_{k,k} && (U^T U_{.,\leq k} = I_{.,\leq k}) \\
&= XVI_{.,\leq k} && (\Sigma^{-1}I_{.,\leq k} = \Sigma^{-1}_{k,k}) \\
H &= XV_{.,\leq k}
\end{aligned}
$$

Thus H is a linear transformation of X and $W = V_{.,\leq k}$

- We have encoder $W = V_{\cdot,\leq k}$

- We have encoder $W = V_{.,\leq k}$
- From SVD, we know that V is the matrix of eigen vectors of X^TX

- We have encoder $W = V_{\cdot,\leq k}$
- From SVD, we know that V is the matrix of eigen vectors of X^TX
- From PCA, we know that P is the matrix of the eigen vectors of the covariance matrix

- We have encoder $W = V_{.,\leq k}$
- From SVD, we know that V is the matrix of eigen vectors of X^TX
- From PCA, we know that P is the matrix of the eigen vectors of the covariance matrix
- We saw earlier that, if entries of X are normalized by

- We have encoder $W = V_{\cdot,\leq k}$
- From SVD, we know that V is the matrix of eigen vectors of X^TX
- From PCA, we know that P is the matrix of the eigen vectors of the covariance matrix
- We saw earlier that, if entries of X are normalized by

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

- We have encoder $W = V_{.,\leq k}$
- From SVD, we know that V is the matrix of eigen vectors of $X^T X$
- From PCA, we know that P is the matrix of the eigen vectors of the covariance matrix
- We saw earlier that, if entries of X are normalized by

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

then $X^T X$ is indeed the covariance matrix

- We have encoder $W = V_{.,\leq k}$
- From SVD, we know that V is the matrix of eigen vectors of $X^T X$
- From PCA, we know that P is the matrix of the eigen vectors of the covariance matrix
- We saw earlier that, if entries of X are normalized by

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

then $X^T X$ is indeed the covariance matrix

- Thus, the encoder matrix for linear autoencoder(W) and the projection matrix(P) for PCA could indeed be the same. Hence proved

Remember

The encoder of a linear autoencoder is equivalent to PCA if we

Remember

The encoder of a linear autoencoder is equivalent to PCA if we

- use a linear encoder

Remember

The encoder of a linear autoencoder is equivalent to PCA if we

- use a linear encoder
- use a linear decoder

Remember

The encoder of a linear autoencoder is equivalent to PCA if we

- use a linear encoder
- use a linear decoder
- use a squared error loss function

Remember

The encoder of a linear autoencoder is equivalent to PCA if we

- use a linear encoder
- use a linear decoder
- use a squared error loss function
- and normalize the inputs to

Remember

The encoder of a linear autoencoder is equivalent to PCA if we

- use a linear encoder
- use a linear decoder
- use a squared error loss function
- and normalize the inputs to

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