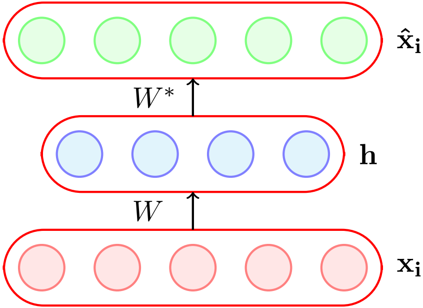
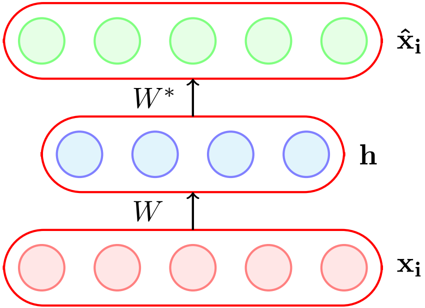
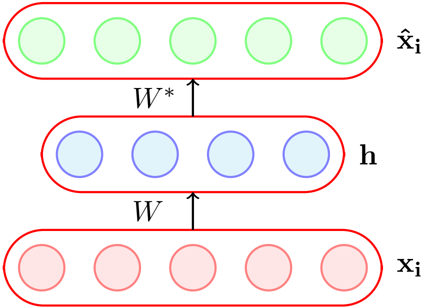
AUTOENCODER

Slide Courtesy of Mitesh Khapra, Deep Learning, IIT Madras

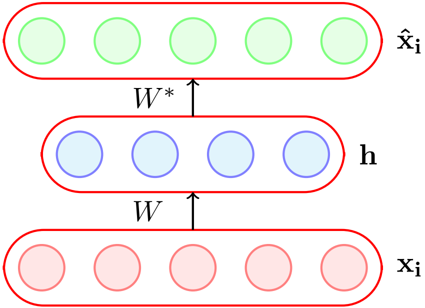

$\hat{\mathbf{x}}_\mathbf{i}$
W^*
$\mathbf{h}$
W
$\mathbf{x}_\mathbf{i}$



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

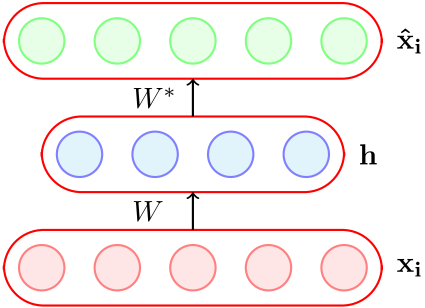


- An autoencoder is a special type of feed forward neural network which does the following
- Encodes its input $\mathbf{x_i}$ into a hidden representation $\mathbf{h}$



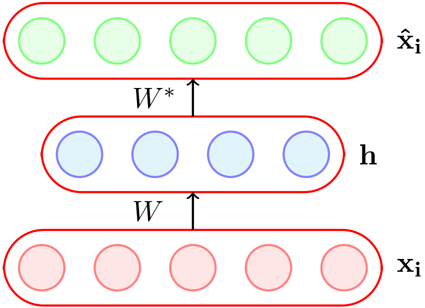
$$\mathbf{h} = g(W\mathbf{x_i} + \mathbf{b})$$

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



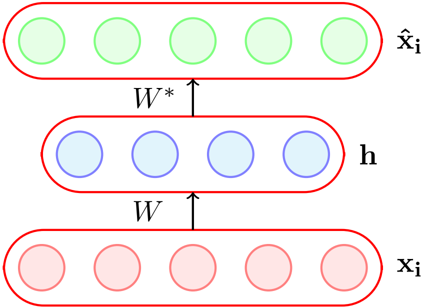
$$\mathbf{h} = g(W\mathbf{x}_\mathbf{i} + \mathbf{b})$$

- 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



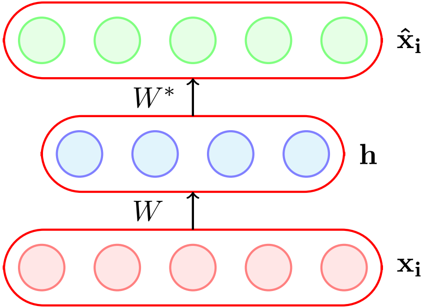
$$\mathbf{h} = g(W\mathbf{x}_\mathbf{i} + \mathbf{b})$$
$$\hat{\mathbf{x}}_\mathbf{i} = f(W^*\mathbf{h} + \mathbf{c})$$

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

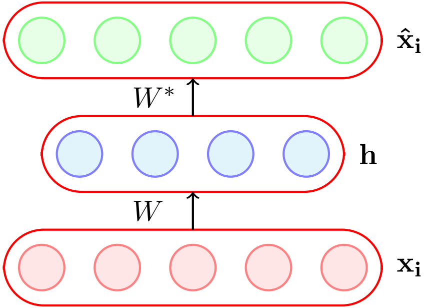


$$\mathbf{h} = g(W\mathbf{x}_\mathbf{i} + \mathbf{b})$$
$$\hat{\mathbf{x}}_\mathbf{i} = f(W^*\mathbf{h} + \mathbf{c})$$

- 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)

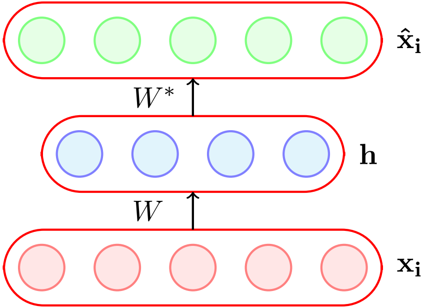


$$\mathbf{h} = g(W\mathbf{x}_\mathbf{i} + \mathbf{b})$$
$$\hat{\mathbf{x}}_\mathbf{i} = f(W^*\mathbf{h} + \mathbf{c})$$



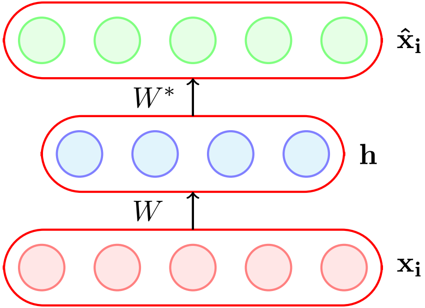
$$\mathbf{h} = g(W\mathbf{x}_\mathbf{i} + \mathbf{b})$$
$$\hat{\mathbf{x}}_\mathbf{i} = f(W^*\mathbf{h} + \mathbf{c})$$

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



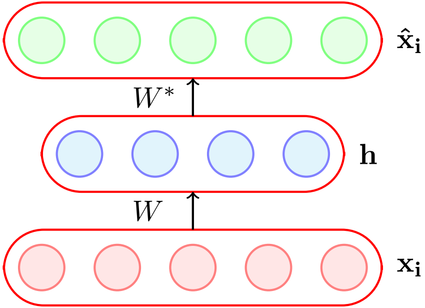
$$\mathbf{h} = g(W\mathbf{x}_\mathbf{i} + \mathbf{b})$$
$$\hat{\mathbf{x}}_\mathbf{i} = f(W^*\mathbf{h} + \mathbf{c})$$

- 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 $\mathbf{h}$?



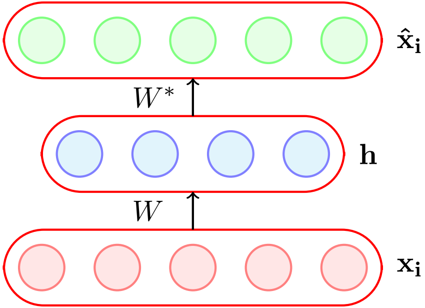
$$\mathbf{h} = g(W\mathbf{x}_\mathbf{i} + \mathbf{b})$$
$$\hat{\mathbf{x}}_\mathbf{i} = f(W^*\mathbf{h} + \mathbf{c})$$

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



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

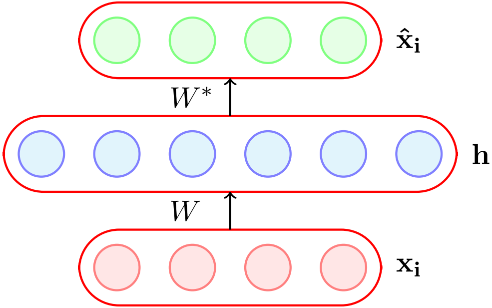
- Let us consider the case where $\dim(\mathbf{h}) < \dim(\mathbf{x}_\mathbf{i})$
- If we are still able to reconstruct $\hat{\mathbf{x}}_\mathbf{i}$ perfectly from $\mathbf{h}$, then what does it say about $\mathbf{h}$?
- $\mathbf{h}$ is a loss-free encoding of $\mathbf{x}_\mathbf{i}$. It captures all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



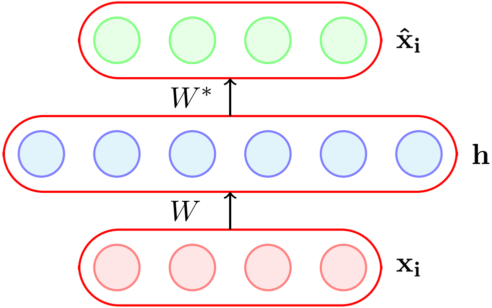
$$\mathbf{h} = g(W\mathbf{x}_\mathbf{i} + \mathbf{b})$$
$$\hat{\mathbf{x}}_\mathbf{i} = f(W^*\mathbf{h} + \mathbf{c})$$

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

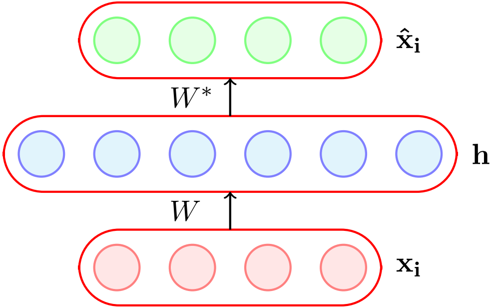


$$\mathbf{h} = g(W\mathbf{x}_\mathbf{i} + \mathbf{b})$$
$$\hat{\mathbf{x}}_\mathbf{i} = f(W^*\mathbf{h} + \mathbf{c})$$



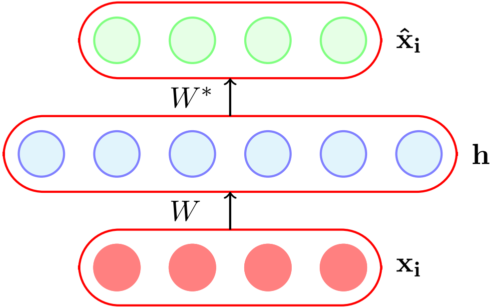
$$\mathbf{h} = g(W\mathbf{x}_\mathbf{i} + \mathbf{b})$$
$$\hat{\mathbf{x}}_\mathbf{i} = f(W^*\mathbf{h} + \mathbf{c})$$

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



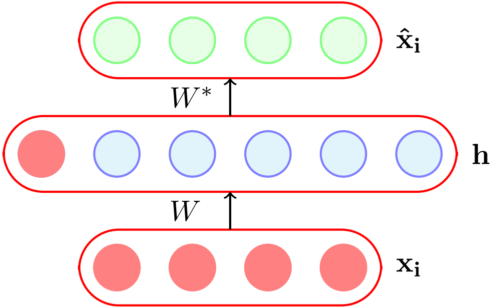
$$\mathbf{h} = g(W\mathbf{x}_\mathbf{i} + \mathbf{b})$$
$$\hat{\mathbf{x}}_\mathbf{i} = f(W^*\mathbf{h} + \mathbf{c})$$

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



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

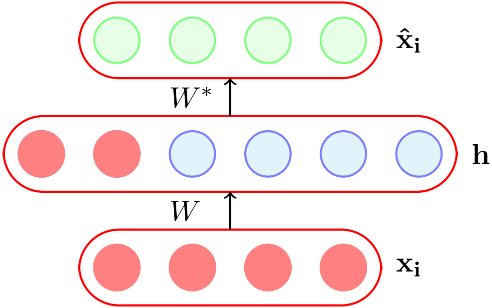
- Let us consider the case when $\dim(\mathbf{h}) \geq \dim(\mathbf{x}_\mathbf{i})$
- In such a case the autoencoder could learn a trivial encoding by simply copying $\mathbf{x}_\mathbf{i}$ into $\mathbf{h}$ and then copying $\mathbf{h}$ into $\hat{\mathbf{x}}_\mathbf{i}$



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

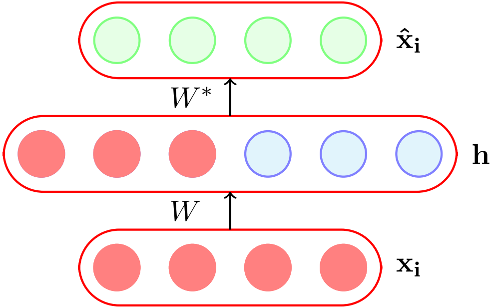
$$\hat{\mathbf{x}}_\mathbf{i} = f(W^*\mathbf{h} + \mathbf{c})$$

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



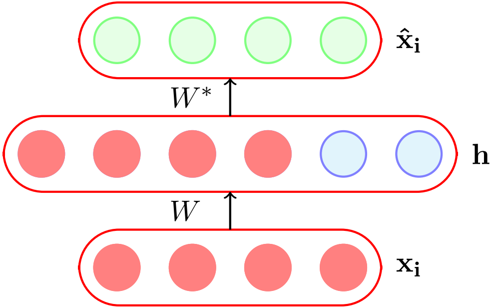
$$\mathbf{h} = g(W\mathbf{x}_\mathbf{i} + \mathbf{b})$$
$$\hat{\mathbf{x}}_\mathbf{i} = f(W^*\mathbf{h} + \mathbf{c})$$

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



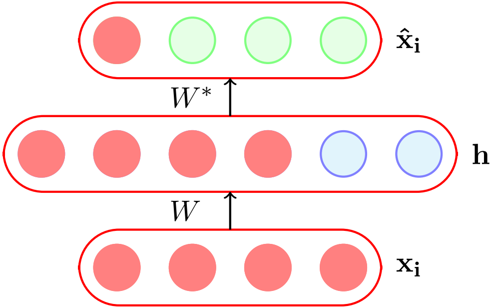
$$\mathbf{h} = g(W\mathbf{x}_\mathbf{i} + \mathbf{b})$$
$$\hat{\mathbf{x}}_\mathbf{i} = f(W^*\mathbf{h} + \mathbf{c})$$

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



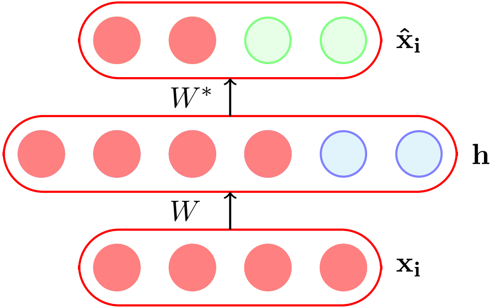
$$\mathbf{h} = g(W\mathbf{x_i} + \mathbf{b})$$
$$\hat{\mathbf{x}}_\mathbf{i} = f(W^*\mathbf{h} + \mathbf{c})$$

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



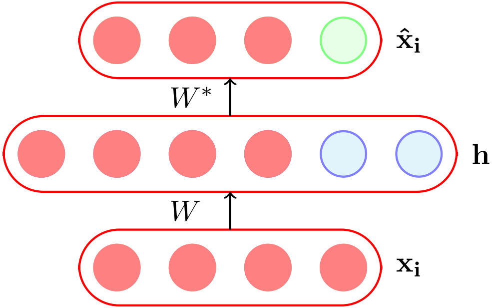
$$\mathbf{h} = g(W\mathbf{x}_\mathbf{i} + \mathbf{b})$$
$$\hat{\mathbf{x}}_\mathbf{i} = f(W^*\mathbf{h} + \mathbf{c})$$

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



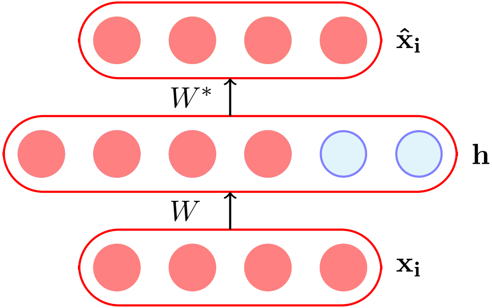
$$\mathbf{h} = g(W\mathbf{x}_\mathbf{i} + \mathbf{b})$$
$$\hat{\mathbf{x}}_\mathbf{i} = f(W^*\mathbf{h} + \mathbf{c})$$

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



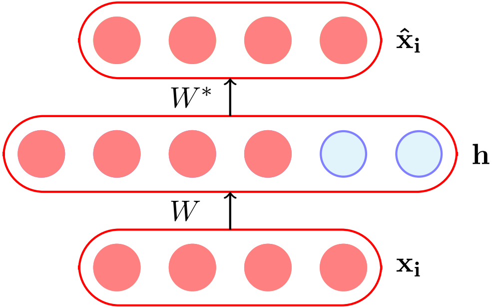
$$\mathbf{h} = g(W\mathbf{x}_\mathbf{i} + \mathbf{b})$$
$$\hat{\mathbf{x}}_\mathbf{i} = f(W^*\mathbf{h} + \mathbf{c})$$

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



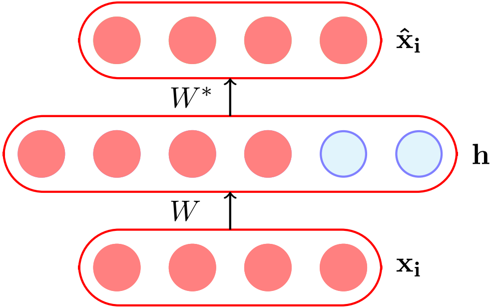
$$\mathbf{h} = g(W\mathbf{x}_\mathbf{i} + \mathbf{b})$$
$$\hat{\mathbf{x}}_\mathbf{i} = f(W^*\mathbf{h} + \mathbf{c})$$

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



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

- 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 <u>over complete</u> 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

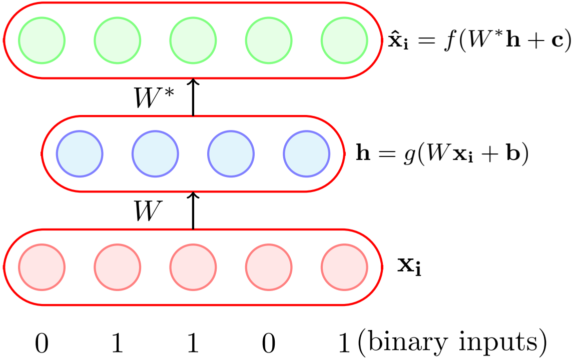
- Choice of $f(\mathbf{x_i})$ and $g(\mathbf{x_i})$

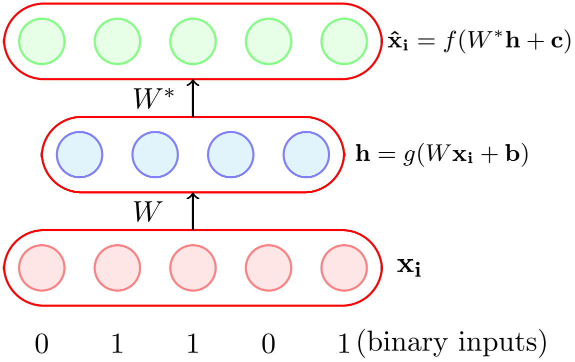
The Road Ahead

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

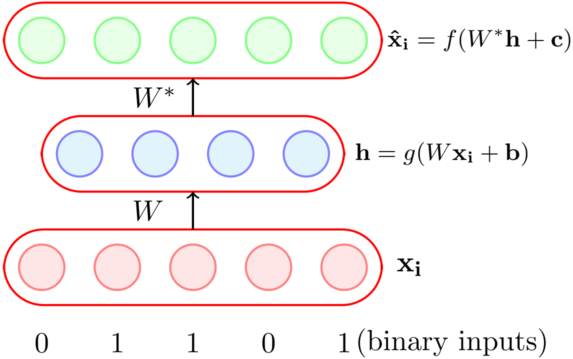
The Road Ahead

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

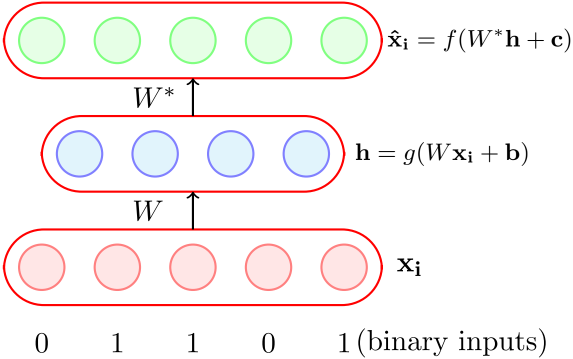

$\hat{\mathbf{x}}_\mathbf{i} = f(W^*\mathbf{h} + \mathbf{c})$
W^*
$\mathbf{h} = g(W\mathbf{x}_\mathbf{i} + \mathbf{b})$
W
$\mathbf{x}_\mathbf{i}$
0 1 1 0 1 (binary inputs)



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

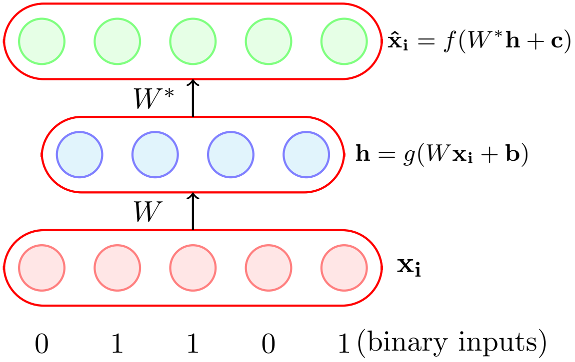


- Suppose all our inputs are binary (each $x_{ij} \in \{0, 1\}$)
- Which of the following functions would be most apt for the decoder?



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

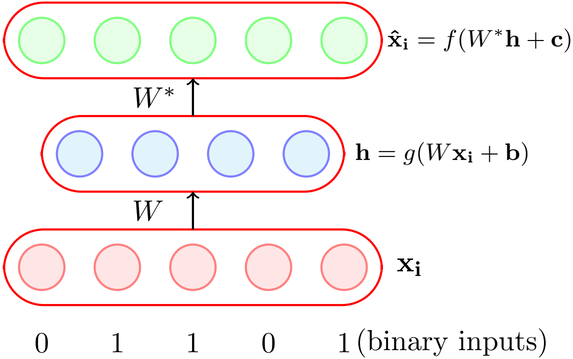
$$\hat{\mathbf{x}}_\mathbf{i} = \tanh(W^*\mathbf{h} + \mathbf{c})$$



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

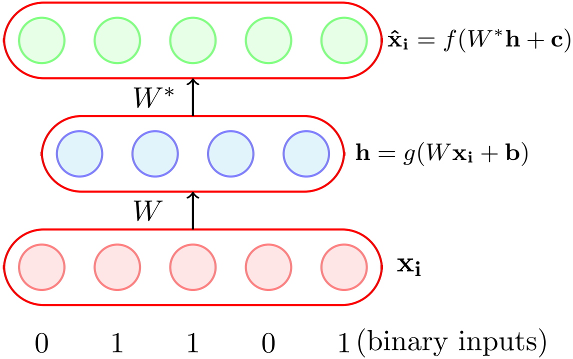
$$\hat{\mathbf{x}}_{\mathbf{i}} = \tanh(W^*\mathbf{h} + \mathbf{c})$$

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



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

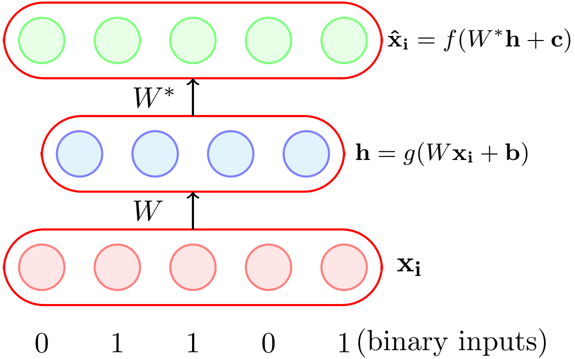
$$\hat{\mathbf{x}}_\mathbf{i} = \tanh(W^*\mathbf{h} + \mathbf{c})$$
$$\hat{\mathbf{x}}_\mathbf{i} = W^*\mathbf{h} + \mathbf{c}$$
$$\hat{\mathbf{x}}_\mathbf{i} = logistic(W^*\mathbf{h} + \mathbf{c})$$



- 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

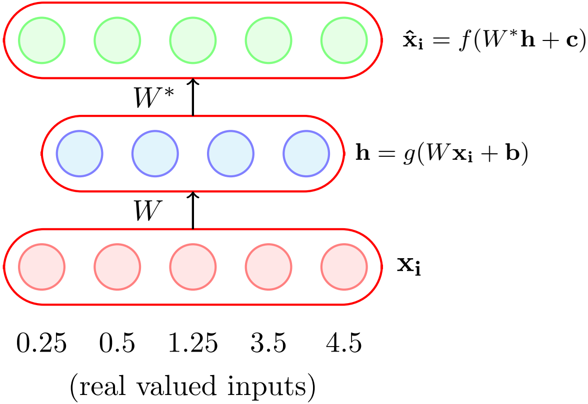


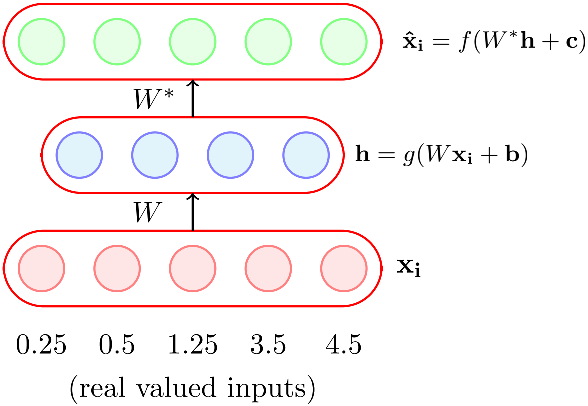
g is typically chosen as the sigmoid function

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

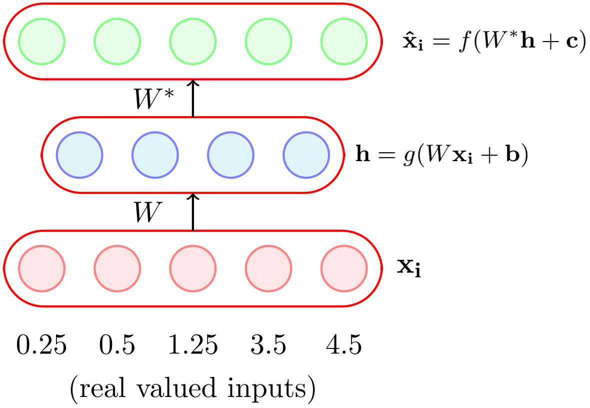
$$\hat{\mathbf{x}}_\mathbf{i} = \tanh(W^*\mathbf{h} + \mathbf{c})$$
$$\hat{\mathbf{x}}_\mathbf{i} = W^*\mathbf{h} + \mathbf{c}$$
$$\hat{\mathbf{x}}_\mathbf{i} = logistic(W^*\mathbf{h} + \mathbf{c})$$

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

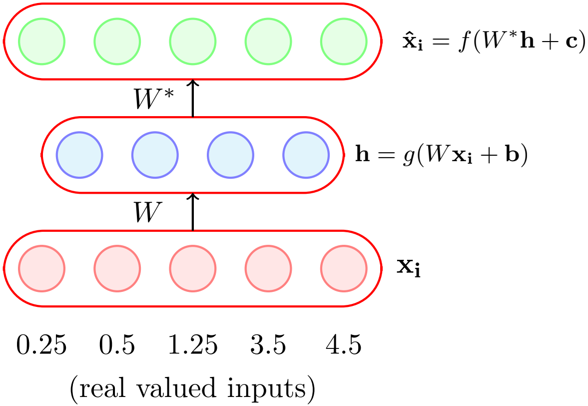

$\hat{\mathbf{x}}_{\mathbf{i}} = f(W^*\mathbf{h} + \mathbf{c})$
W^*
$\mathbf{h} = g(W\mathbf{x}_{\mathbf{i}} + \mathbf{b})$
W
$\mathbf{x}_{\mathbf{i}}$
0.25 0.5 1.25 3.5 4.5
(real valued inputs)



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

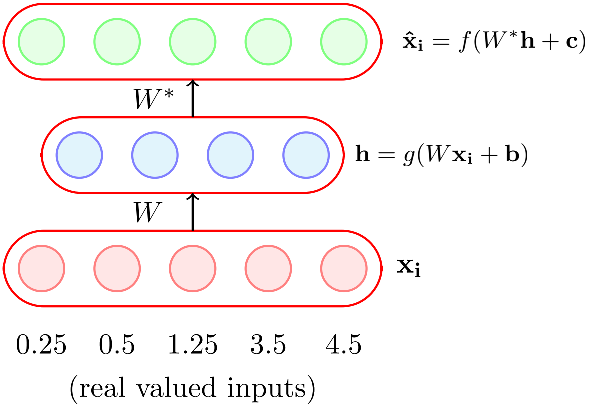


- Suppose all our inputs are real (each $x_{ij} \in \mathbb{R}$)
- Which of the following functions would be most apt for the decoder?



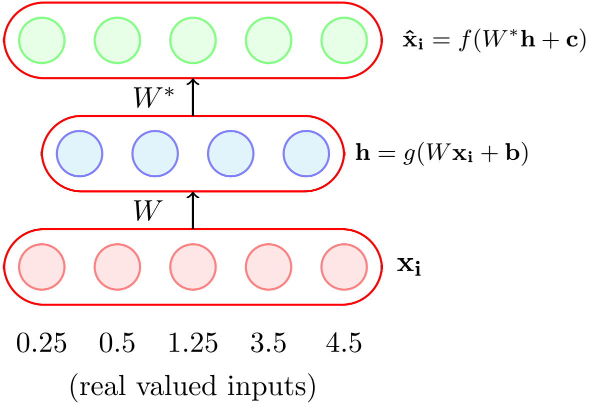
- Suppose all our inputs are real (each $x_{ij} \in \mathbb{R}$)
- Which of the following functions would be most apt for the decoder?

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



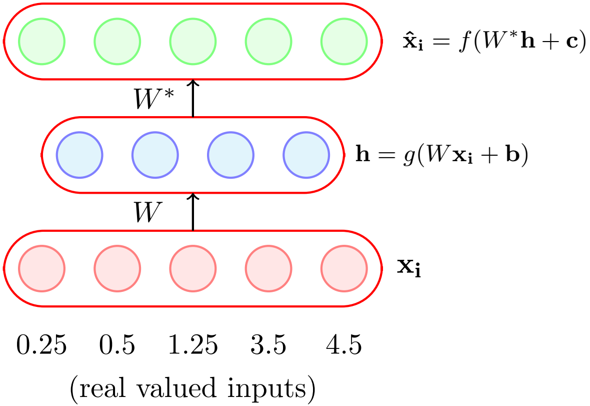
- Suppose all our inputs are real (each $x_{ij} \in \mathbb{R}$)
- Which of the following functions would be most apt for the decoder?

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



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

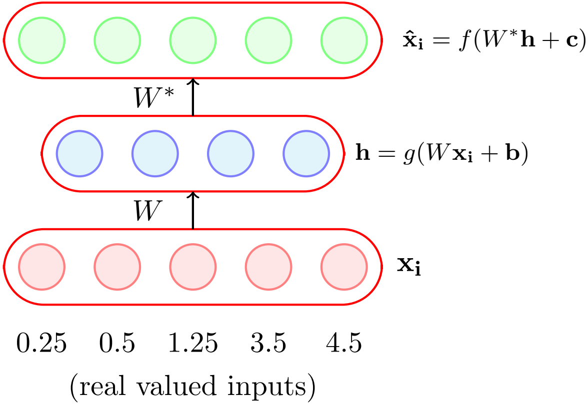
$$\hat{\mathbf{x}}_\mathbf{i} = \tanh(W^*\mathbf{h} + \mathbf{c})$$
$$\hat{\mathbf{x}}_\mathbf{i} = W^*\mathbf{h} + \mathbf{c}$$
$$\hat{\mathbf{x}}_\mathbf{i} = \text{logistic}(W^*\mathbf{h} + \mathbf{c})$$



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

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

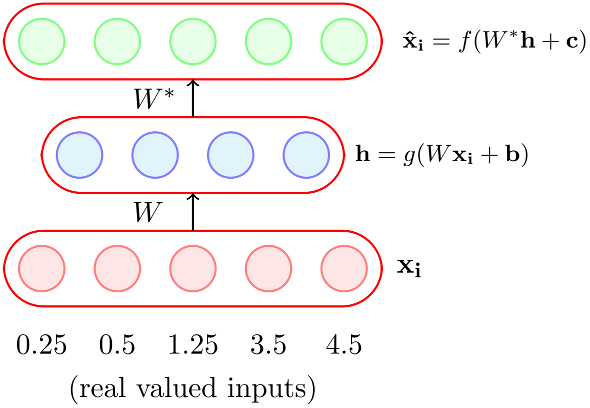
- What will logistic and tanh do?



- 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

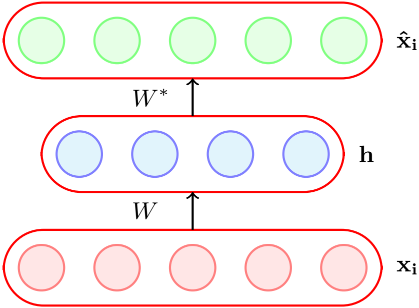
- Suppose all our inputs are real (each $x_{ij} \in \mathbb{R}$)
- Which of the following functions would be most apt for the decoder?

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

- 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 $\hat{\mathbf{x}}_\mathbf{i} \in \mathbb{R}^n$

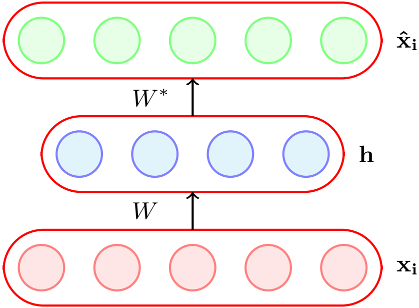
The Road Ahead

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



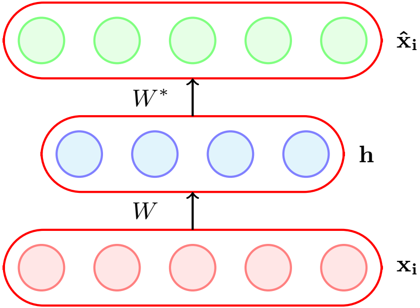
$$\mathbf{h} = g(W\mathbf{x}_\mathbf{i} + \mathbf{b})$$

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



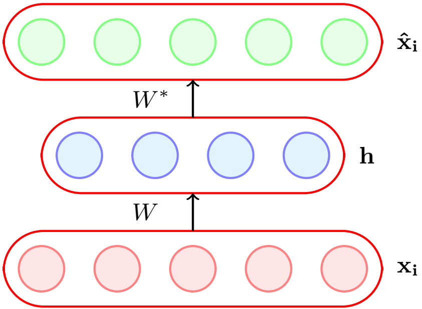
$$\mathbf{h} = g(W\mathbf{x}_\mathbf{i} + \mathbf{b})$$
$$\hat{\mathbf{x}}_\mathbf{i} = f(W^*\mathbf{h} + \mathbf{c})$$

- Consider the case when the inputs are real valued



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

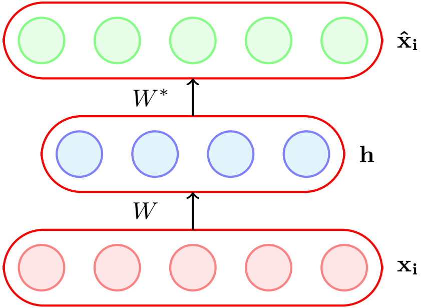
- Consider the case when the inputs are real valued
- The objective of the autoencoder is to reconstruct $\hat{\mathbf{x}}_\mathbf{i}$ to be as close to $\mathbf{x}_\mathbf{i}$ as possible



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

- 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:

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

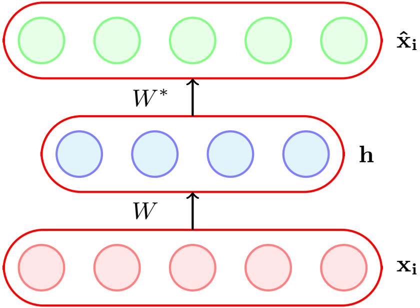


$$\mathbf{h} = g(W\mathbf{x}_\mathbf{i} + \mathbf{b})$$
$$\hat{\mathbf{x}}_\mathbf{i} = f(W^*\mathbf{h} + \mathbf{c})$$

- 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:

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

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



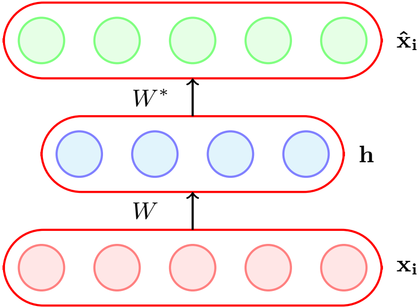
$$\mathbf{h} = g(W\mathbf{x}_\mathbf{i} + \mathbf{b})$$
$$\hat{\mathbf{x}}_\mathbf{i} = f(W^*\mathbf{h} + \mathbf{c})$$

- 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



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

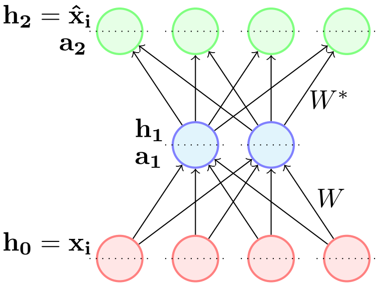
- Consider the case when the inputs are real valued
- The objective of the autoencoder is to reconstruct $\hat{\mathbf{x}}_\mathbf{i}$ to be as close to $\mathbf{x_i}$ as possible
- This can be formalized using the following objective function:

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

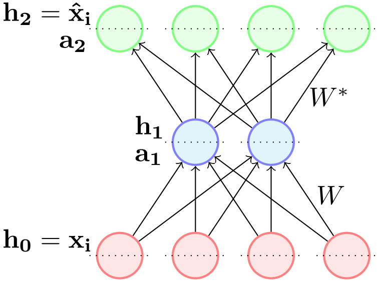
$$i.e., \min_{W, W^*, \mathbf{c}, \mathbf{b}} \frac{1}{m} \sum_{i=1}^{m} (\hat{\mathbf{x}}_\mathbf{i} - \mathbf{x_i})^T (\hat{\mathbf{x}}_\mathbf{i} - \mathbf{x_i})$$

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

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

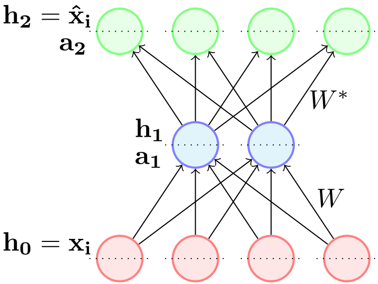


$$\mathscr{L}(\theta) = (\mathbf{\hat{x}_i} - \mathbf{x_i})^T(\mathbf{\hat{x}_i} - \mathbf{x_i})$$



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

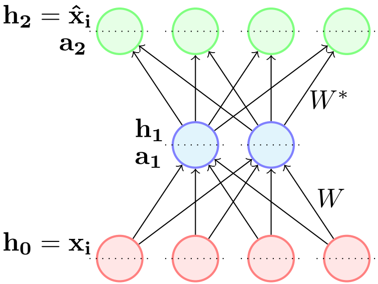
$$\mathscr{L}(\theta) = (\mathbf{\hat{x}_i} - \mathbf{x_i})^T(\mathbf{\hat{x}_i} - \mathbf{x_i})$$



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

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

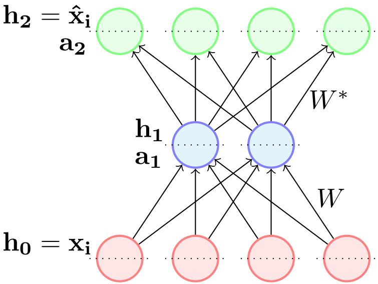
$$\mathscr{L}(\theta) = (\mathbf{\hat{x}_i} - \mathbf{x_i})^T(\mathbf{\hat{x}_i} - \mathbf{x_i})$$



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

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

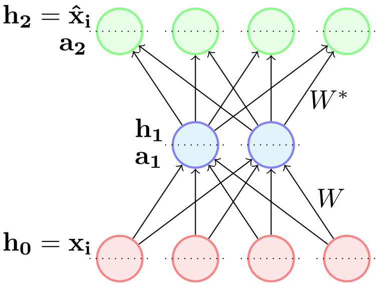
$$\mathscr{L}(\theta) = (\hat{\mathbf{x}}_\mathbf{i} - \mathbf{x}_\mathbf{i})^T(\hat{\mathbf{x}}_\mathbf{i} - \mathbf{x}_\mathbf{i})$$



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

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

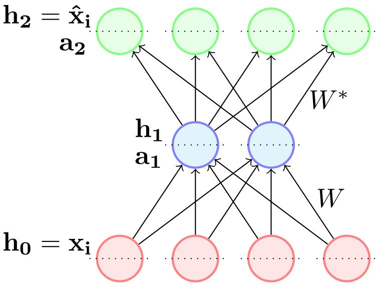
- $$\frac{\partial \mathscr{L}(\theta)}{\partial W^*} = \frac{\partial \mathscr{L}(\theta)}{\partial \mathbf{h_2}} \boxed{\frac{\partial \mathbf{h_2}}{\partial \mathbf{a_2}} \frac{\partial \mathbf{a_2}}{\partial W^*}}$$

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

- 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.

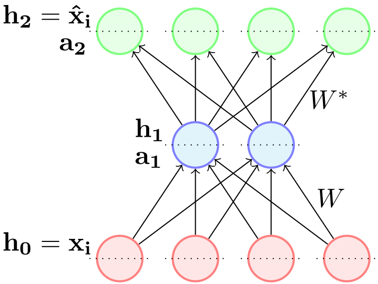
- $$\frac{\partial \mathscr{L}(\theta)}{\partial W^*} = \frac{\partial \mathscr{L}(\theta)}{\partial \mathbf{h_2}} \boxed{\frac{\partial \mathbf{h_2}}{\partial \mathbf{a_2}} \frac{\partial \mathbf{a_2}}{\partial W^*}}$$

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

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

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

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



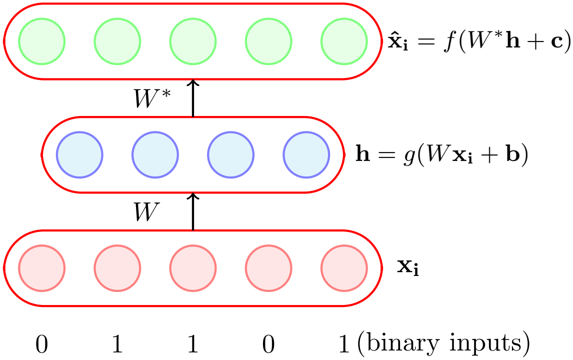
- Note that the loss function is shown for only one training example.

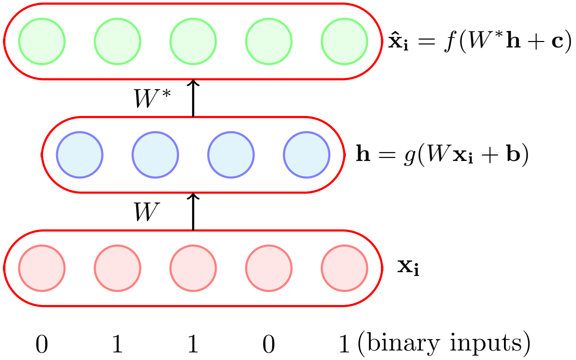
- $$\frac{\partial \mathscr{L}(\theta)}{\partial W^*} = \frac{\partial \mathscr{L}(\theta)}{\partial \mathbf{h_2}} \boxed{\frac{\partial \mathbf{h_2}}{\partial \mathbf{a_2}} \frac{\partial \mathbf{a_2}}{\partial W^*}}$$

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

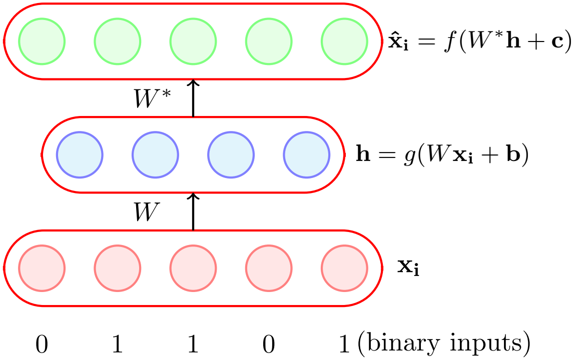
- We have already seen how to calculate the expression in the boxes when we learnt backpropagation

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

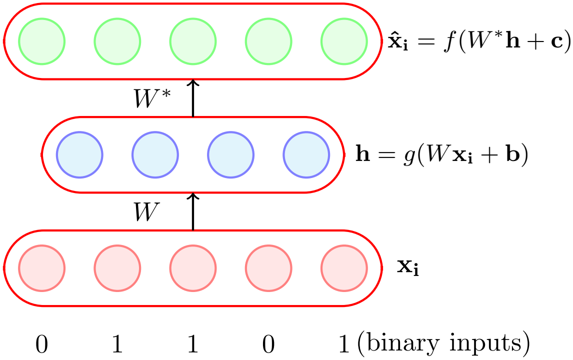

$\hat{\mathbf{x}}_\mathbf{i} = f(W^*\mathbf{h} + \mathbf{c})$
W^*
$\mathbf{h} = g(W\mathbf{x}_\mathbf{i} + \mathbf{b})$
W
$\mathbf{x}_\mathbf{i}$
0 1 1 0 1 (binary inputs)



- Consider the case when the inputs are binary

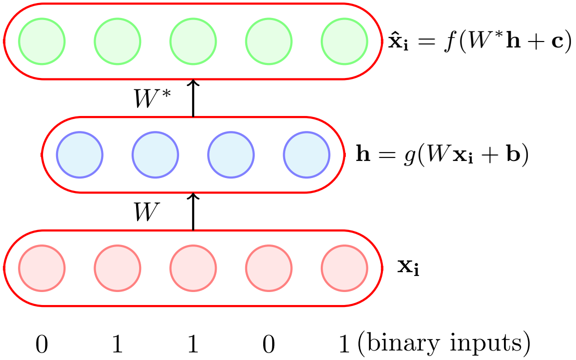


- Consider the case when the inputs are binary
- We use a sigmoid decoder which will produce outputs between 0 and 1, and can be interpreted as probabilities.



- 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

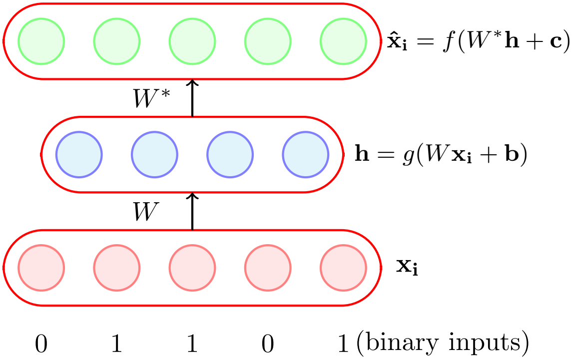
$$\min\{-\sum_{j=1}^{n}(x_{ij}\log\hat{x}_{ij} + (1 - x_{ij})\log(1 - \hat{x}_{ij}))\}$$



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

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

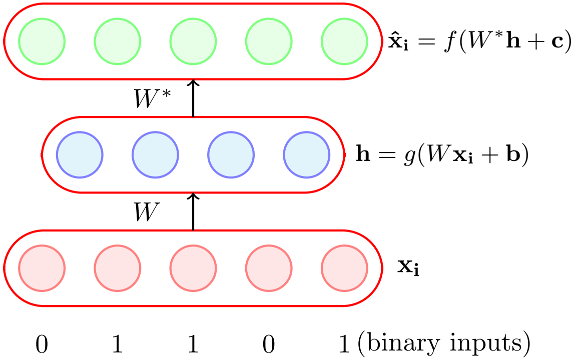


What value of $\hat{x}_{ij}$ will minimize this function?

- 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

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

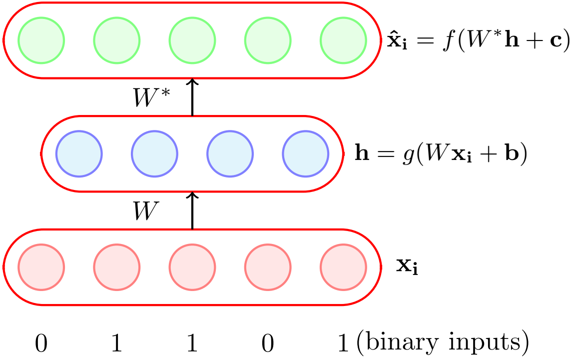


What value of $\hat{x}_{ij}$ will minimize this function?

- 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

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



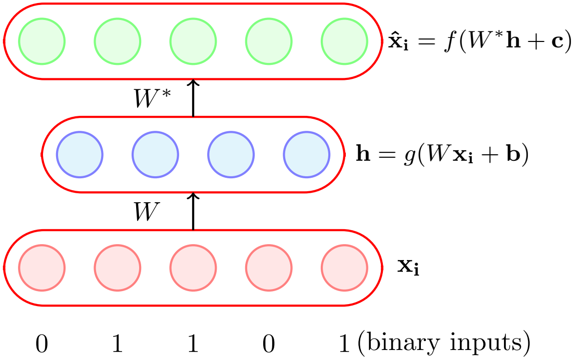
What value of $\hat{x}_{ij}$ will minimize this function?

- 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 $x_{ij} = 0$?

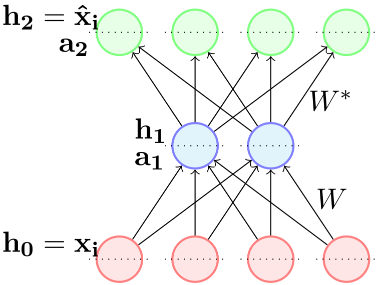
Indeed the above function will be minimized when $\hat{x}_{ij} = x_{ij}$!

- 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

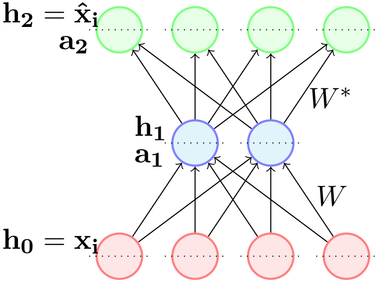
$$\min\{-\sum_{j=1}^{n}(x_{ij}\log\hat{x}_{ij} + (1 - x_{ij})\log(1 - \hat{x}_{ij}))\}$$

- 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

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

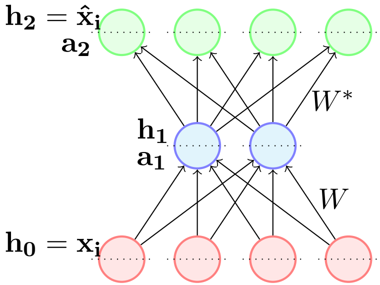


$$\mathscr{L}(\theta) = -\sum_{j=1}^{n}(x_{ij}\log\hat{x}_{ij} + (1-x_{ij})\log(1-\hat{x}_{ij}))$$



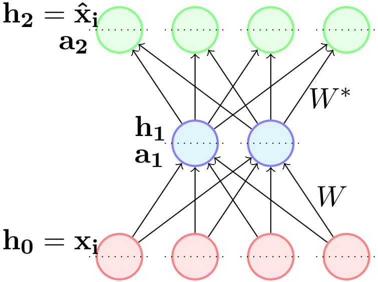
- $$\frac{\partial \mathscr{L}(\theta)}{\partial W^*} = \frac{\partial \mathscr{L}(\theta)}{\partial \mathbf{h_2}}\frac{\partial \mathbf{h_2}}{\partial \mathbf{a_2}}\boxed{\frac{\partial \mathbf{a_2}}{\partial W^*}}$$

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



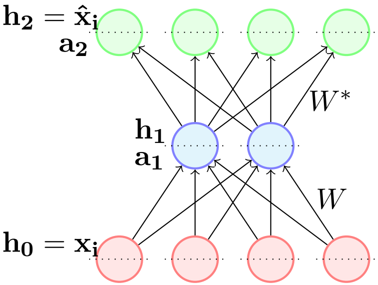
- $$\frac{\partial \mathscr{L}(\theta)}{\partial W^*} = \frac{\partial \mathscr{L}(\theta)}{\partial \mathbf{h_2}} \frac{\partial \mathbf{h_2}}{\partial \mathbf{a_2}} \boxed{\frac{\partial \mathbf{a_2}}{\partial W^*}}$$
- $$\frac{\partial \mathscr{L}(\theta)}{\partial W} = \frac{\partial \mathscr{L}(\theta)}{\partial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

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



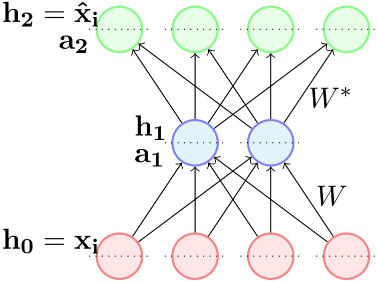
- $$\frac{\partial \mathscr{L}(\theta)}{\partial W^*} = \frac{\partial \mathscr{L}(\theta)}{\partial \mathbf{h_2}} \frac{\partial \mathbf{h_2}}{\partial \mathbf{a_2}} \boxed{\frac{\partial \mathbf{a_2}}{\partial W^*}}$$
- $$\frac{\partial \mathscr{L}(\theta)}{\partial W} = \frac{\partial \mathscr{L}(\theta)}{\partial \mathbf{h_2}} \frac{\partial \mathbf{h_2}}{\partial \mathbf{a_2}} \boxed{\frac{\partial \mathbf{a_2}}{\partial \mathbf{h_1}} \frac{\partial \mathbf{h_1}}{\partial \mathbf{a_1}} \frac{\partial \mathbf{a_1}}{\partial W}}$$
- 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:
$$\frac{\partial \mathscr{L}(\theta)}{\partial h_{2j}} = -\frac{x_{ij}}{\hat{x}_{ij}} + \frac{1 - x_{ij}}{1 - \hat{x}_{ij}}$$
$$\frac{\partial h_{2j}}{\partial a_{2j}} = \sigma(a_{2j})(1 - \sigma(a_{2j}))$$

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



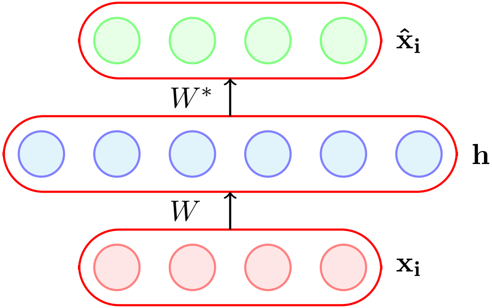
$$\frac{\partial \mathscr{L}(\theta)}{\partial \mathbf{h_2}} = \begin{pmatrix} \frac{\partial \mathscr{L}(\theta)}{\partial h_{21}} \\ \frac{\partial \mathscr{L}(\theta)}{\partial h_{22}} \\ \vdots \\ \frac{\partial \mathscr{L}(\theta)}{\partial h_{2n}} \end{pmatrix}$$

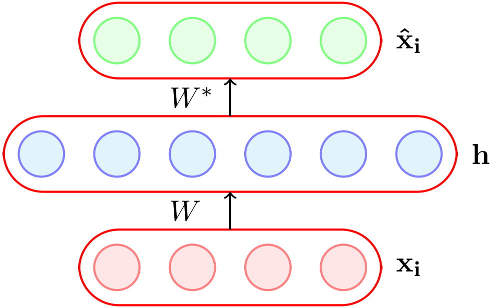
- $$\frac{\partial \mathscr{L}(\theta)}{\partial W^*} = \frac{\partial \mathscr{L}(\theta)}{\partial \mathbf{h_2}} \frac{\partial \mathbf{h_2}}{\partial \mathbf{a_2}} \boxed{\frac{\partial \mathbf{a_2}}{\partial W^*}}$$
- $$\frac{\partial \mathscr{L}(\theta)}{\partial W} = \frac{\partial \mathscr{L}(\theta)}{\partial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

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

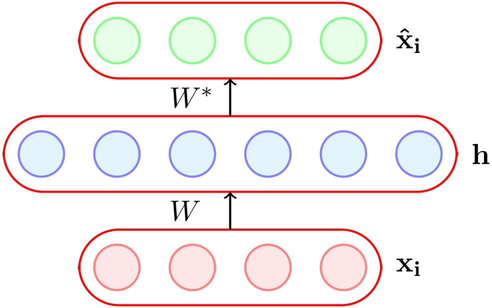
$$\frac{\partial h_{2j}}{\partial a_{2j}} = \sigma(a_{2j})(1 - \sigma(a_{2j}))$$

REGULARIZATION

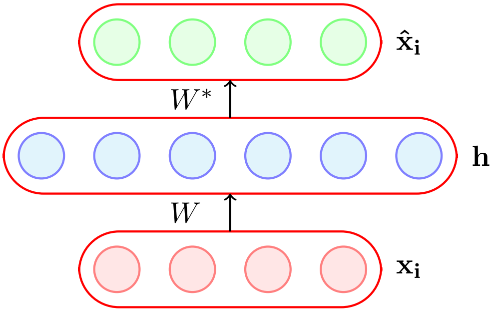

$\hat{\mathbf{x}}_\mathbf{i}$
W^*
$\mathbf{h}$
W
$\mathbf{x}_\mathbf{i}$



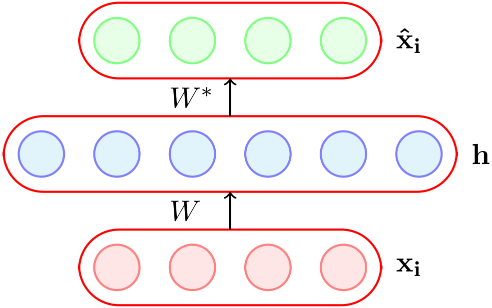
- While poor generalization could happen even in undercomplete autoencoders it is an even more serious problem for overcomplete auto encoders



- While poor generalization could happen even in undercomplete autoencoders it is an even more serious problem for overcomplete auto encoders
- Here, (as stated earlier) the model can simply learn to copy $\mathbf{x_i}$ to $\mathbf{h}$ and then $\mathbf{h}$ to $\hat{\mathbf{x}}_\mathbf{i}$

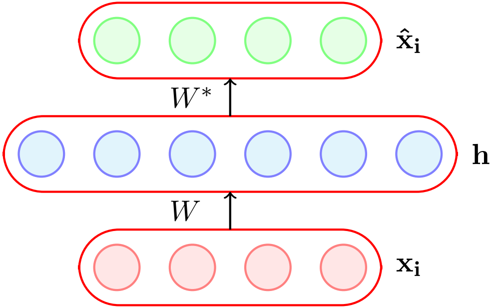


- While poor generalization could happen even in undercomplete autoencoders it is an even more serious problem for overcomplete auto encoders
- Here, (as stated earlier) the model can simply learn to copy $\mathbf{x}_\mathbf{i}$ to $\mathbf{h}$ and then $\mathbf{h}$ to $\hat{\mathbf{x}}_\mathbf{i}$
- To avoid poor generalization, we need to introduce regularization



- The simplest solution is to add a L_2-regularization term to the objective function

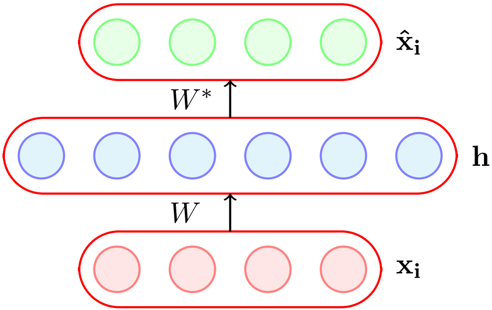
$$\min_{\theta, w, w^*, \mathbf{b}, \mathbf{c}} \frac{1}{m} \sum_{i=1}^{m} \sum_{j=1}^{n} (\hat{x}_{ij} - x_{ij})^2 + \lambda \|\theta\|^2$$



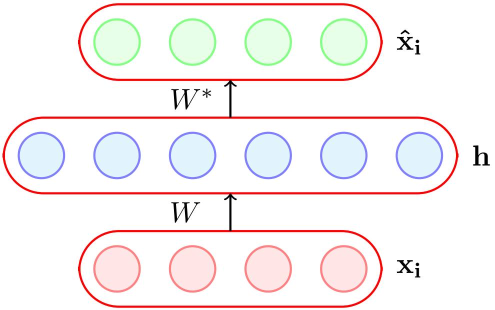
- The simplest solution is to add a L_2-regularization term to the objective function

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

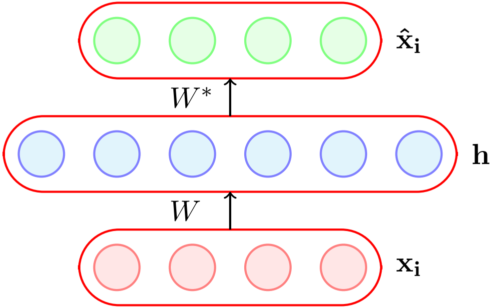
- This is very easy to implement and just adds a term λW to the gradient $\frac{\partial \mathscr{L}(\theta)}{\partial W}$ (and similarly for other parameters)



- Another trick is to tie the weights of the encoder and decoder



- Another trick is to tie the weights of the encoder and decoder i.e., $W^* = W^T$



- Another trick is to tie the weights of the encoder and decoder i.e., $W^* = W^T$
- This effectively reduces the capacity of Autoencoder and acts as a regularizer