第六讲：自编码器及其变体

Autoencoders and Its Variants

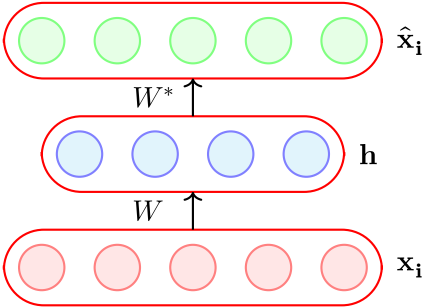
张盛平

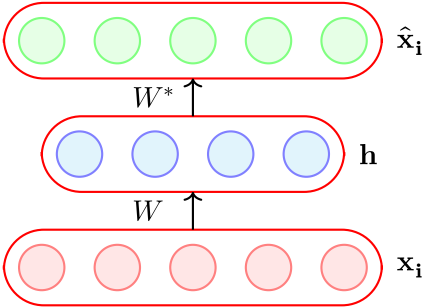
s.zhang@hit.edu.cn

计算学部
哈尔滨工业大学

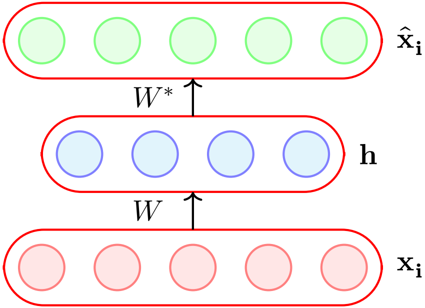
2021 年秋季学期

Introduction to Autoencoders

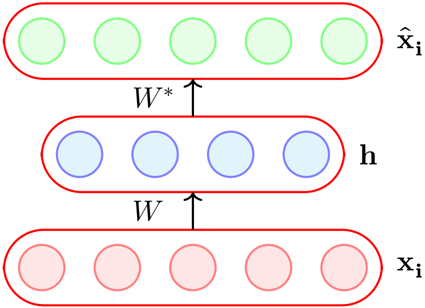

$\hat{\mathbf{x}}_\mathbf{i}$
W^*
$\mathbf{h}$
W
$\mathbf{x}_\mathbf{i}$



- 自编码器是一类特殊的前馈神经网络

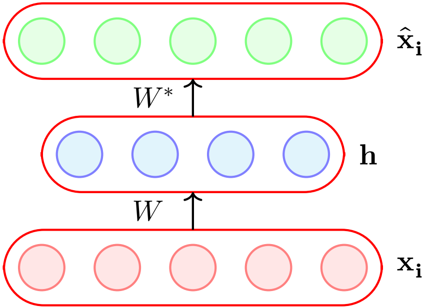


- 自编码器是一类特殊的前馈神经网络
- Encodes 输入 $\mathbf{x}_\mathbf{i}$ 得到隐含表示 $\mathbf{h}$



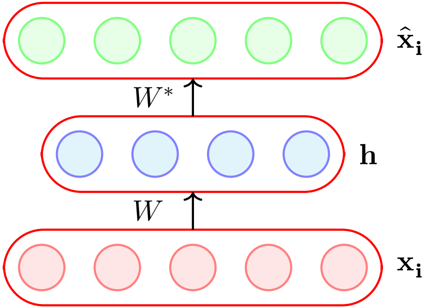
$$\mathbf{h} = g(W\mathbf{x}_\mathbf{i} + \mathbf{b})$$

- 自编码器是一类特殊的前馈神经网络
- Encodes 输入 $\mathbf{x}_\mathbf{i}$ 得到隐含表示 $\mathbf{h}$



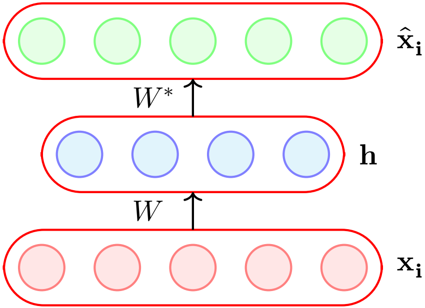
$$\mathbf{h} = g(W\mathbf{x}_\mathbf{i} + \mathbf{b})$$

- 自编码器是一类特殊的前馈神经网络
- Encodes 输入 $\mathbf{x}_\mathbf{i}$ 得到隐含表示 $\mathbf{h}$
- Decodes 输入，从隐含表示



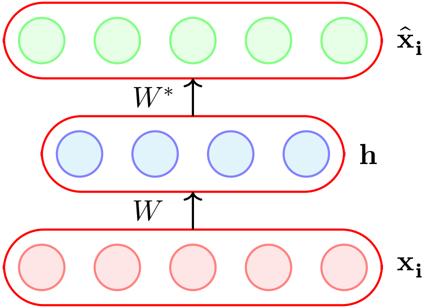
$$\mathbf{h} = g(W\mathbf{x}_\mathbf{i} + \mathbf{b})$$
$$\hat{\mathbf{x}}_\mathbf{i} = f(W^*\mathbf{h} + \mathbf{c})$$

- 自编码器是一类特殊的前馈神经网络
- Encodes 输入 $\mathbf{x}_\mathbf{i}$ 得到隐含表示 $\mathbf{h}$
- Decodes 输入，从隐含表示

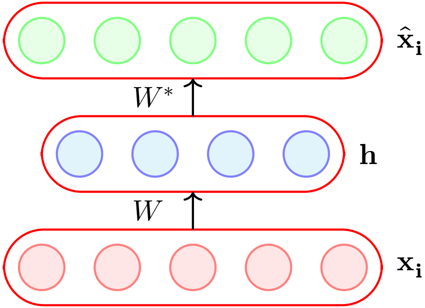


$$\mathbf{h} = g(W\mathbf{x}_\mathbf{i} + \mathbf{b})$$
$$\hat{\mathbf{x}}_\mathbf{i} = f(W^*\mathbf{h} + \mathbf{c})$$

- 自编码器是一类特殊的前馈神经网络
- Encodes 输入 $\mathbf{x}_\mathbf{i}$ 得到隐含表示 $\mathbf{h}$
- Decodes 输入，从隐含表示
- 最小化特定的损失函数确保 $\hat{\mathbf{x}}_\mathbf{i}$ 接近于 $\mathbf{x}_\mathbf{i}$

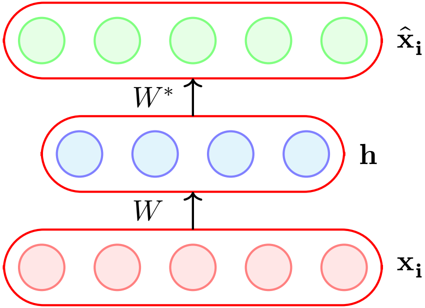


$$\mathbf{h} = g(W\mathbf{x}_\mathbf{i} + \mathbf{b})$$
$$\hat{\mathbf{x}}_\mathbf{i} = f(W^*\mathbf{h} + \mathbf{c})$$



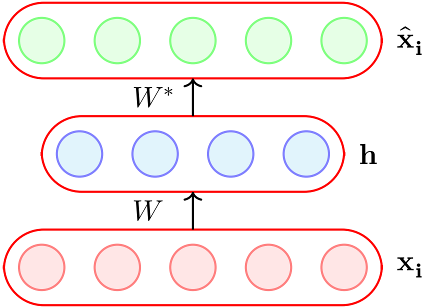
$$\mathbf{h} = g(W\mathbf{x}_\mathbf{i} + \mathbf{b})$$
$$\hat{\mathbf{x}}_\mathbf{i} = f(W^*\mathbf{h} + \mathbf{c})$$

- 考虑当 $\dim(\mathbf{h}) < \dim(\mathbf{x}_\mathbf{i})$ 的情况



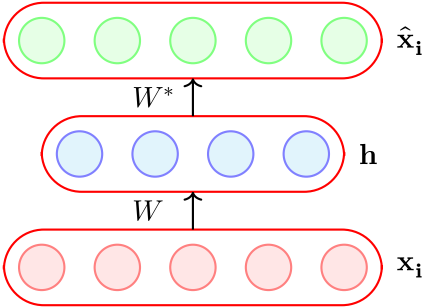
$$\mathbf{h} = g(W\mathbf{x}_\mathbf{i} + \mathbf{b})$$
$$\hat{\mathbf{x}}_\mathbf{i} = f(W^*\mathbf{h} + \mathbf{c})$$

- 考虑当 $\dim(\mathbf{h}) < \dim(\mathbf{x}_\mathbf{i})$ 的情况
- 如果我们能够从隐含表示 $\mathbf{h}$ 中重构出 $\hat{\mathbf{x}}_\mathbf{i}$，隐含表示 $\mathbf{h}$ 说明了什么？



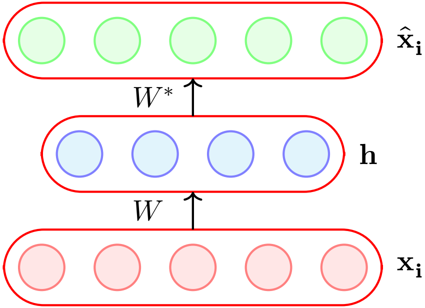
$$\mathbf{h} = g(W\mathbf{x}_\mathbf{i} + \mathbf{b})$$
$$\hat{\mathbf{x}}_\mathbf{i} = f(W^*\mathbf{h} + \mathbf{c})$$

- 考虑当 $\dim(\mathbf{h}) < \dim(\mathbf{x}_\mathbf{i})$ 的情况
- 如果我们能够从隐含表示 $\mathbf{h}$ 中重构出 $\hat{\mathbf{x}}_\mathbf{i}$，隐含表示 $\mathbf{h}$ 说明了什么?
- $\mathbf{h}$ 是 $\mathbf{x}_\mathbf{i}$ 的一个损失无关的编码，它提取了 $\mathbf{x}_\mathbf{i}$ 所有重要的信息



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

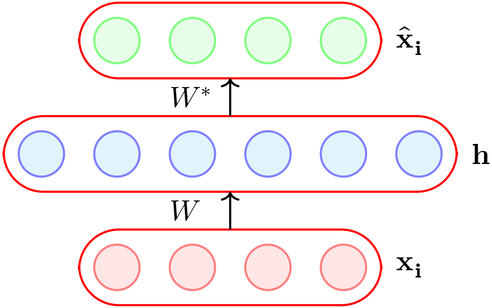
- 考虑当 $\dim(\mathbf{h}) < \dim(\mathbf{x}_\mathbf{i})$ 的情况
- 如果我们能够从隐含表示 $\mathbf{h}$ 中重构出 $\hat{\mathbf{x}}_\mathbf{i}$，隐含表示 $\mathbf{h}$ 说明了什么？
- $\mathbf{h}$ 是 $\mathbf{x}_\mathbf{i}$ 的一个损失无关的编码，它提取了 $\mathbf{x}_\mathbf{i}$ 所有重要的信息
- 是否与 PCA 类似？



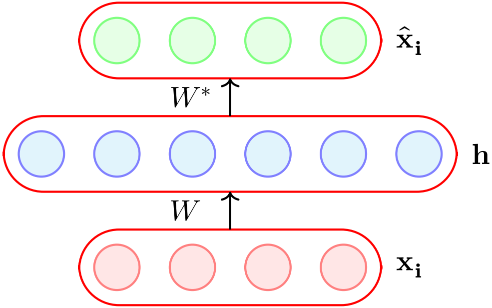
$$\mathbf{h} = g(W\mathbf{x}_\mathbf{i} + \mathbf{b})$$
$$\hat{\mathbf{x}}_\mathbf{i} = f(W^*\mathbf{h} + \mathbf{c})$$

当 $\dim(\mathbf{h}) < \dim(\mathbf{x}_\mathbf{i})$ 时，称为 under complete autoencoder

- 考虑当 $\dim(\mathbf{h}) < \dim(\mathbf{x}_\mathbf{i})$ 的情况
- 如果我们能够从隐含表示 $\mathbf{h}$ 中重构出 $\hat{\mathbf{x}}_\mathbf{i}$，隐含表示 $\mathbf{h}$ 说明了什么?
- $\mathbf{h}$ 是 $\mathbf{x}_\mathbf{i}$ 的一个损失无关的编码，它提取了 $\mathbf{x}_\mathbf{i}$ 所有重要的信息
- 是否与 PCA 类似?

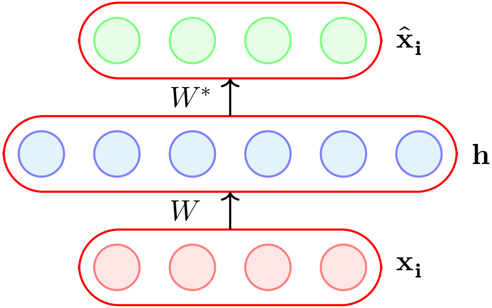


$$\mathbf{h} = g(W\mathbf{x}_\mathbf{i} + \mathbf{b})$$
$$\hat{\mathbf{x}}_\mathbf{i} = f(W^*\mathbf{h} + \mathbf{c})$$



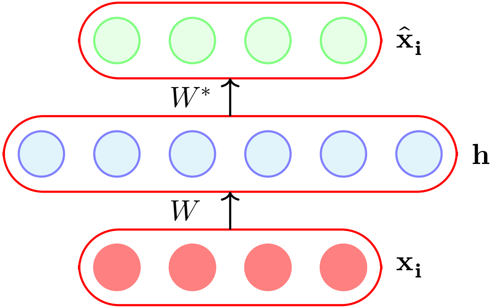
$$\mathbf{h} = g(W\mathbf{x}_{\mathbf{i}} + \mathbf{b})$$
$$\hat{\mathbf{x}}_{\mathbf{i}} = f(W^*\mathbf{h} + \mathbf{c})$$

- 当 $\dim(\mathbf{h}) \geq \dim(\mathbf{x}_{\mathbf{i}})$ 时



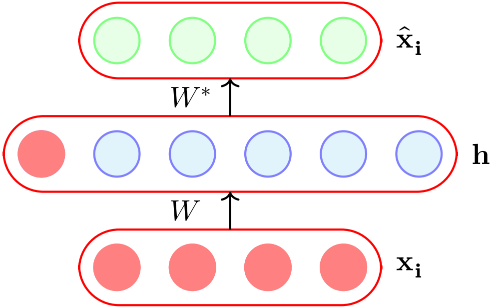
$$\mathbf{h} = g(W\mathbf{x}_\mathbf{i} + \mathbf{b})$$
$$\hat{\mathbf{x}}_\mathbf{i} = f(W^*\mathbf{h} + \mathbf{c})$$

- 当 $\dim(\mathbf{h}) \geq \dim(\mathbf{x}_\mathbf{i})$ 时
- 自编码器会学到 $\mathbf{x}_\mathbf{i}$ 的一个简单的编码，简单地将 $\mathbf{x}_\mathbf{i}$ 复制进 $\mathbf{h}$，然后再把 $\mathbf{h}$ 复制进 $\hat{\mathbf{x}}_\mathbf{i}$



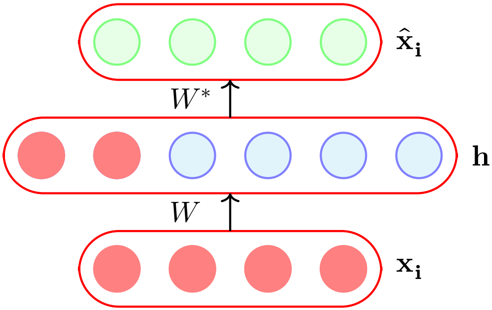
$$\mathbf{h} = g(W\mathbf{x}_\mathbf{i} + \mathbf{b})$$
$$\hat{\mathbf{x}}_\mathbf{i} = f(W^*\mathbf{h} + \mathbf{c})$$

- 当 $\dim(\mathbf{h}) \geq \dim(\mathbf{x}_\mathbf{i})$ 时
- 自编码器会学到 $\mathbf{x}_\mathbf{i}$ 的一个简单的编码，简单地将 $\mathbf{x}_\mathbf{i}$ 复制进 $\mathbf{h}$，然后再把 $\mathbf{h}$ 复制进 $\hat{\mathbf{x}}_\mathbf{i}$



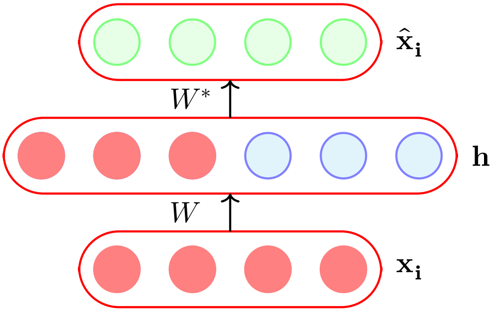
$$\mathbf{h} = g(W\mathbf{x}_\mathbf{i} + \mathbf{b})$$
$$\hat{\mathbf{x}}_\mathbf{i} = f(W^*\mathbf{h} + \mathbf{c})$$

- 当 $\dim(\mathbf{h}) \geq \dim(\mathbf{x}_\mathbf{i})$ 时
- 自编码器会学到 $\mathbf{x}_\mathbf{i}$ 的一个简单的编码，简单地将 $\mathbf{x}_\mathbf{i}$ 复制进 $\mathbf{h}$，然后再把 $\mathbf{h}$ 复制进 $\hat{\mathbf{x}}_\mathbf{i}$



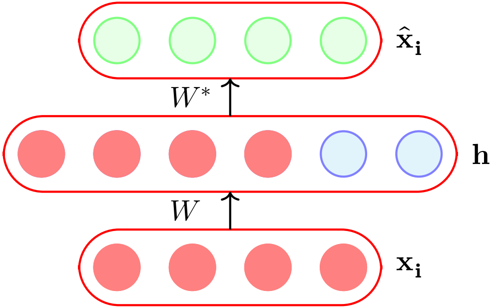
$$\mathbf{h} = g(W\mathbf{x}_\mathbf{i} + \mathbf{b})$$
$$\hat{\mathbf{x}}_\mathbf{i} = f(W^*\mathbf{h} + \mathbf{c})$$

- 当 $\dim(\mathbf{h}) \geq \dim(\mathbf{x}_\mathbf{i})$ 时
- 自编码器会学到 $\mathbf{x}_\mathbf{i}$ 的一个简单的编码，简单地将 $\mathbf{x}_\mathbf{i}$ 复制进 $\mathbf{h}$，然后再把 $\mathbf{h}$ 复制进 $\hat{\mathbf{x}}_\mathbf{i}$



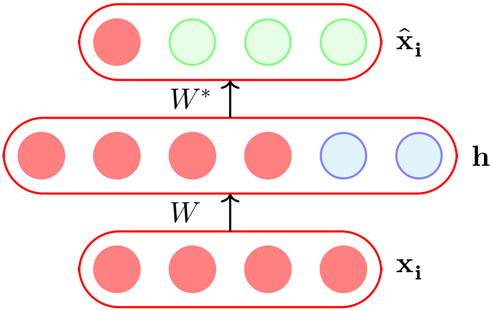
$$\mathbf{h} = g(W\mathbf{x}_\mathbf{i} + \mathbf{b})$$
$$\hat{\mathbf{x}}_\mathbf{i} = f(W^*\mathbf{h} + \mathbf{c})$$

- 当 $\dim(\mathbf{h}) \geq \dim(\mathbf{x}_\mathbf{i})$ 时
- 自编码器会学到 $\mathbf{x}_\mathbf{i}$ 的一个简单的编码，简单地将 $\mathbf{x}_\mathbf{i}$ 复制进 $\mathbf{h}$，然后再把 $\mathbf{h}$ 复制进 $\hat{\mathbf{x}}_\mathbf{i}$



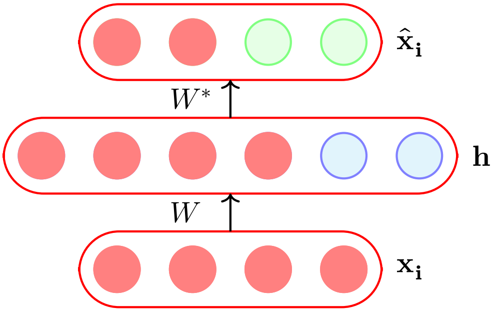
$$\mathbf{h} = g(W\mathbf{x}_\mathbf{i} + \mathbf{b})$$
$$\hat{\mathbf{x}}_\mathbf{i} = f(W^*\mathbf{h} + \mathbf{c})$$

- 当 $\dim(\mathbf{h}) \geq \dim(\mathbf{x}_\mathbf{i})$ 时
- 自编码器会学到 $\mathbf{x}_\mathbf{i}$ 的一个简单的编码，简单地将 $\mathbf{x}_\mathbf{i}$ 复制进 $\mathbf{h}$，然后再把 $\mathbf{h}$ 复制进 $\hat{\mathbf{x}}_\mathbf{i}$



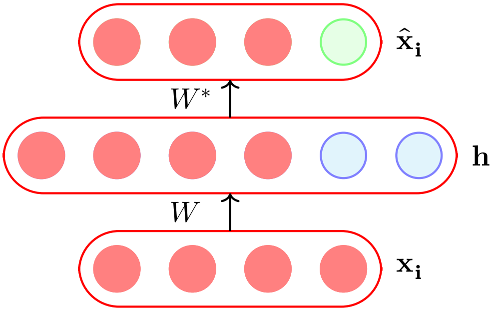
$$\mathbf{h} = g(W\mathbf{x}_\mathbf{i} + \mathbf{b})$$
$$\hat{\mathbf{x}}_\mathbf{i} = f(W^*\mathbf{h} + \mathbf{c})$$

- 当 $\dim(\mathbf{h}) \geq \dim(\mathbf{x}_\mathbf{i})$ 时
- 自编码器会学到 $\mathbf{x}_\mathbf{i}$ 的一个简单的编码，简单地将 $\mathbf{x}_\mathbf{i}$ 复制进 $\mathbf{h}$，然后再把 $\mathbf{h}$ 复制进 $\hat{\mathbf{x}}_\mathbf{i}$



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

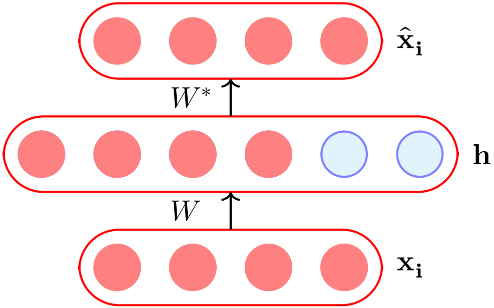
- 当 $\dim(\mathbf{h}) \geq \dim(\mathbf{x}_\mathbf{i})$ 时
- 自编码器会学到 $\mathbf{x}_\mathbf{i}$ 的一个简单的编码，简单地将 $\mathbf{x}_\mathbf{i}$ 复制进 $\mathbf{h}$，然后再把 $\mathbf{h}$ 复制进 $\hat{\mathbf{x}}_\mathbf{i}$



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

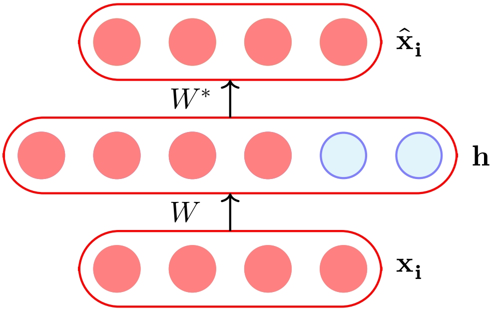
- 当 $\text{dim}(\mathbf{h}) \geq \text{dim}(\mathbf{x}_\mathbf{i})$ 时
- 自编码器会学到 $\mathbf{x}_\mathbf{i}$ 的一个简单的编码，简单地将 $\mathbf{x}_\mathbf{i}$ 复制进 $\mathbf{h}$，然后再把 $\mathbf{h}$ 复制进 $\hat{\mathbf{x}}_\mathbf{i}$

张盛平 s.zhang@hit.edu.cn



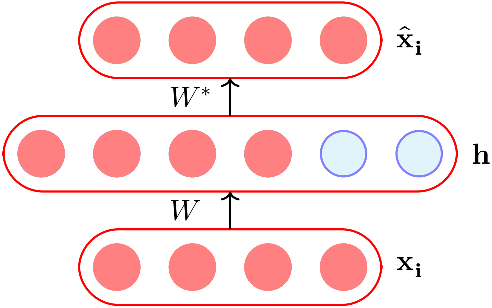
$$\mathbf{h} = g(W\mathbf{x}_\mathbf{i} + \mathbf{b})$$
$$\hat{\mathbf{x}}_\mathbf{i} = f(W^*\mathbf{h} + \mathbf{c})$$

- 当 $\dim(\mathbf{h}) \geq \dim(\mathbf{x}_\mathbf{i})$ 时
- 自编码器会学到 $\mathbf{x}_\mathbf{i}$ 的一个简单的编码，简单地将 $\mathbf{x}_\mathbf{i}$ 复制进 $\mathbf{h}$，然后再把 $\mathbf{h}$ 复制进 $\hat{\mathbf{x}}_\mathbf{i}$



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

- 当 $\dim(\mathbf{h}) \geq \dim(\mathbf{x}_\mathbf{i})$ 时
- 自编码器会学到 $\mathbf{x}_\mathbf{i}$ 的一个简单的编码，简单地将 $\mathbf{x}_\mathbf{i}$ 复制进 $\mathbf{h}$，然后再把 $\mathbf{h}$ 复制进 $\hat{\mathbf{x}}_\mathbf{i}$
- 这样一个对等编码 (identity encoding) 在实际中并没有什么用，因为它没有从数据中提取任何重要信息



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

- 当 $\dim(\mathbf{h}) \geq \dim(\mathbf{x}_\mathbf{i})$ 时
- 自编码器会学到 $\mathbf{x}_\mathbf{i}$ 的一个简单的编码，简单地将 $\mathbf{x}_\mathbf{i}$ 复制进 $\mathbf{h}$，然后再把 $\mathbf{h}$ 复制进 $\hat{\mathbf{x}}_\mathbf{i}$
- 这样一个对等编码 (identity encoding) 在实际中并没有什么用，因为它没有从数据中提取任何重要信息

当 $\dim(\mathbf{h}) \geq \dim(\mathbf{x}_\mathbf{i})$ 时，称为 over complete autoencoder

The Road Ahead

The Road Ahead

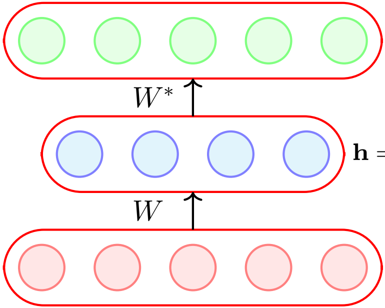
- 选择 $f(\mathbf{x_i})$ 和 $g(\mathbf{x_i})$

The Road Ahead

- 选择 $f(\mathbf{x_i})$ 和 $g(\mathbf{x_i})$
- 选择 loss function

The Road Ahead

- 选择 $f(\mathbf{x_i})$ 和 $g(\mathbf{x_i})$
- 选择 loss function



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

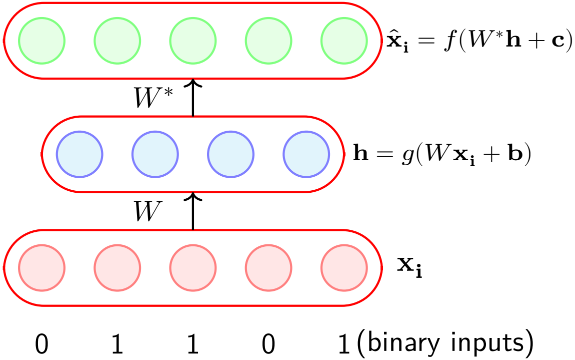
W^*

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

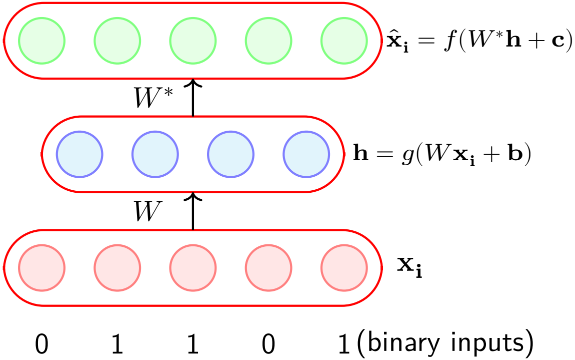
W

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

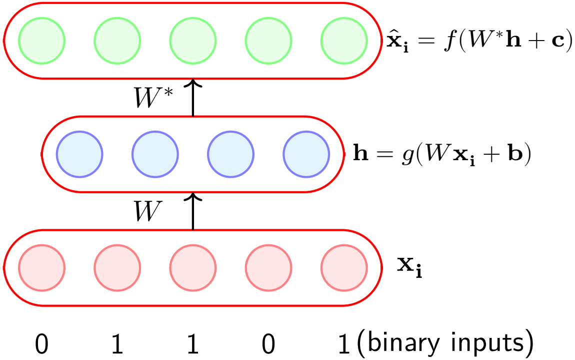
0 1 1 0 1 (binary inputs)



- 假定所有的输入是二值的（$x_{ij} \in \{0,1\}$）

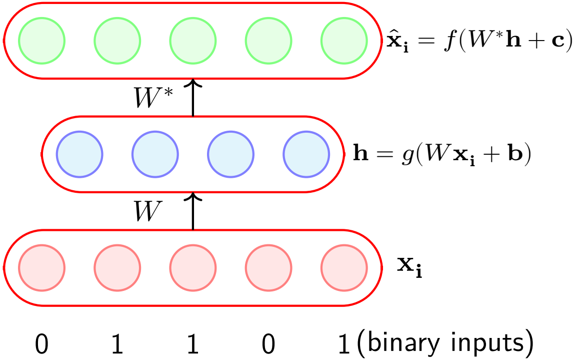


- 假定所有的输入是二值的（$x_{ij} \in \{0,1\}$）
- 下面哪个函数更适合作为 decoder?



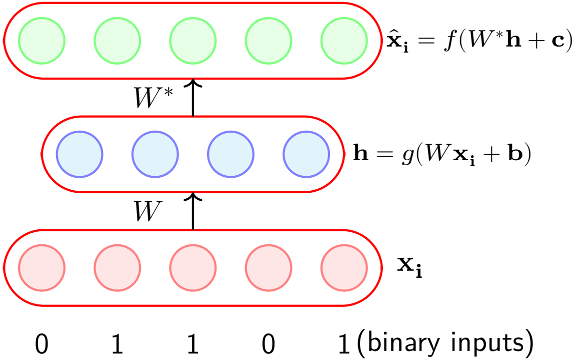
- 假定所有的输入是二值的（$x_{ij} \in \{0,1\}$）
- 下面哪个函数更适合作为 decoder?

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



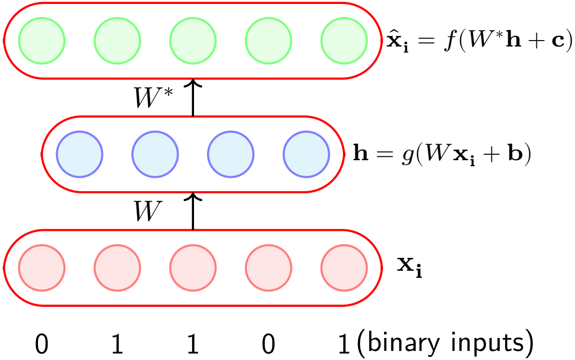
- 假定所有的输入是二值的（$x_{ij} \in \{0,1\}$）
- 下面哪个函数更适合作为 decoder?

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



- 假定所有的输入是二值的（ $x_{ij} \in \{0,1\}$）
- 下面哪个函数更适合作为 decoder?

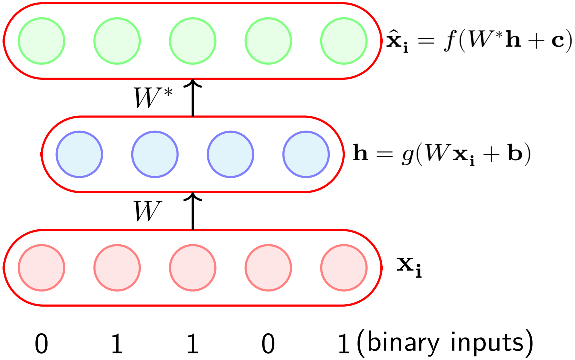
$$\hat{\mathbf{x}}_{\mathbf{i}} = tanh(W^*\mathbf{h} + \mathbf{c})$$
$$\hat{\mathbf{x}}_{\mathbf{i}} = W^*\mathbf{h} + \mathbf{c}$$
$$\hat{\mathbf{x}}_{\mathbf{i}} = logistic(W^*\mathbf{h} + \mathbf{c})$$



- 假定所有的输入是二值的（$x_{ij} \in \{0,1\}$）
- 下面哪个函数更适合作为 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 函数更适合。因为它将所有输出限定在 0 和 1 之间

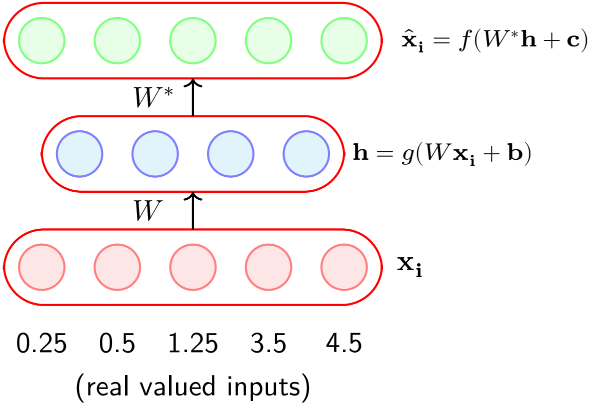


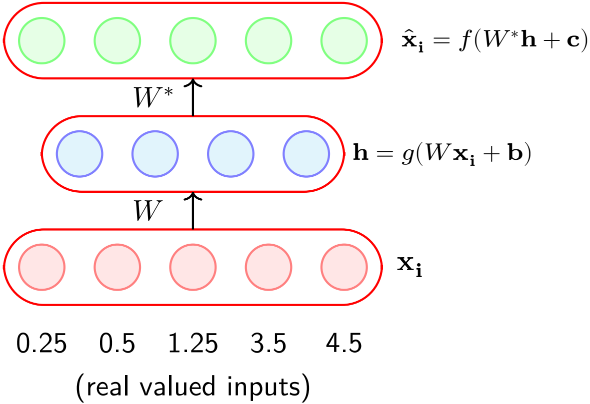
g 通常被选择为 sigmoid function

- 假定所有的输入是二值的（$x_{ij} \in \{0,1\}$）
- 下面哪个函数更适合作为 decoder?

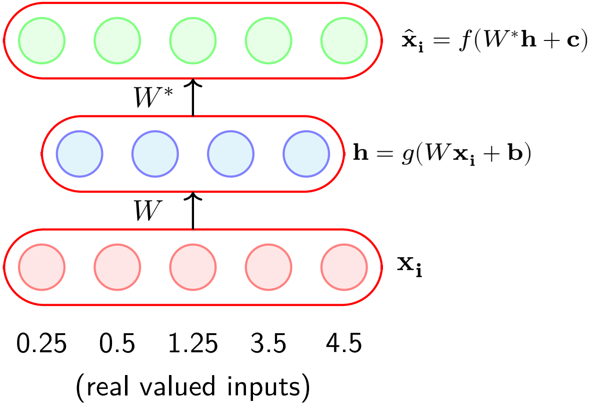
$$\hat{\mathbf{x}}_\mathbf{i} = tanh(W^*\mathbf{h} + \mathbf{c})$$
$$\hat{\mathbf{x}}_\mathbf{i} = W^*\mathbf{h} + \mathbf{c}$$
$$\hat{\mathbf{x}}_\mathbf{i} = logistic(W^*\mathbf{h} + \mathbf{c})$$

- Logistic 函数更适合。因为它将所有输出限定在 0 和 1 之间

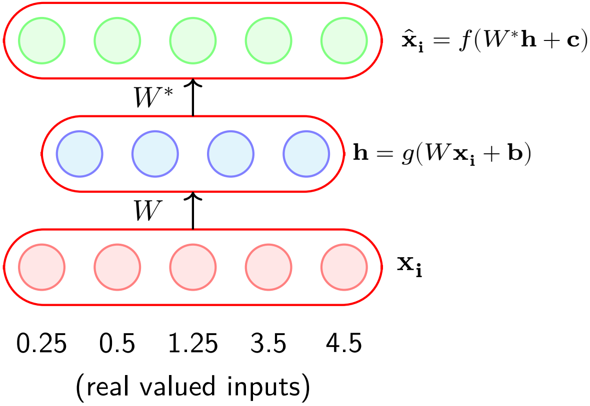

$\hat{\mathbf{x}}_{\mathbf{i}} = f(W^*\mathbf{h} + \mathbf{c})$
W^*
$\mathbf{h} = g(W\mathbf{x}_{\mathbf{i}} + \mathbf{b})$
W
$\mathbf{x}_{\mathbf{i}}$
0.25 0.5 1.25 3.5 4.5
(real valued inputs)



- 假定所有的输入是实数（$x_{ij} \in \mathbb{R}$）

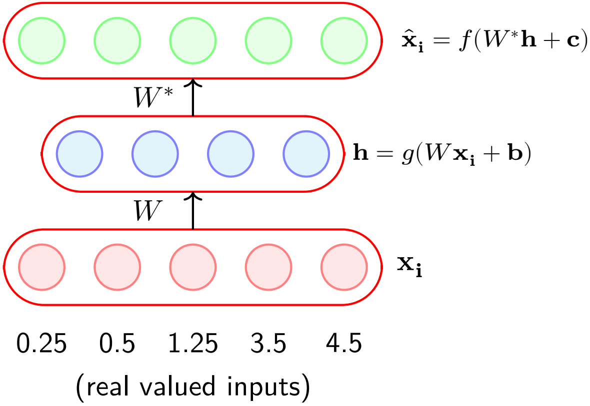


- 假定所有的输入是实数 ($x_{ij} \in \mathbb{R}$)
- 下面哪个函数更适合作为 decoder?



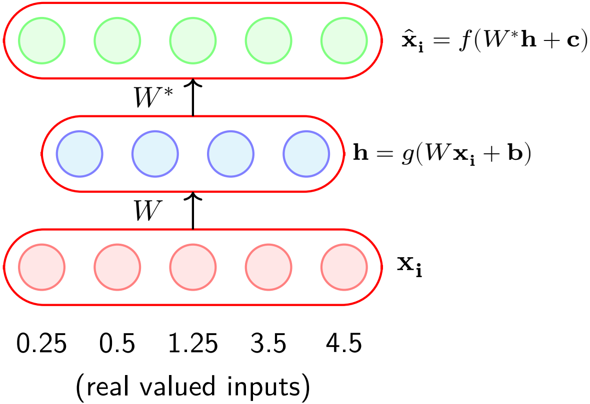
- 假定所有的输入是实数（$x_{ij} \in \mathbb{R}$）
- 下面哪个函数更适合作为 decoder?

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



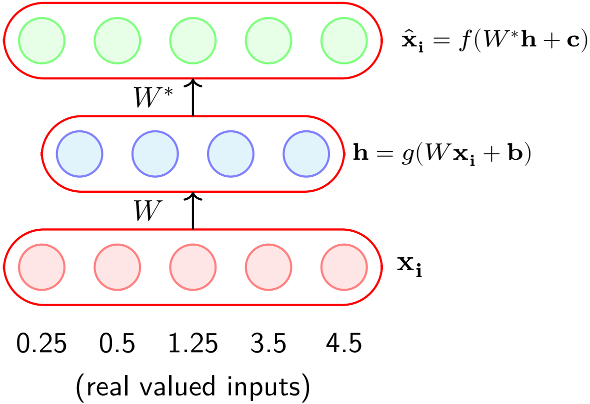
- 假定所有的输入是实数（$x_{ij} \in \mathbb{R}$）
- 下面哪个函数更适合作为 decoder?

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



- 假定所有的输入是实数 ($x_{ij} \in \mathbb{R}$)
- 下面哪个函数更适合作为 decoder?

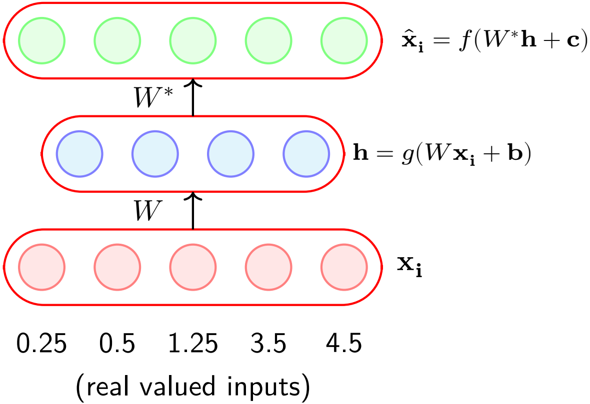
$$\hat{\mathbf{x}}_\mathbf{i} = tanh(W^*\mathbf{h} + \mathbf{c})$$
$$\hat{\mathbf{x}}_\mathbf{i} = W^*\mathbf{h} + \mathbf{c}$$
$$\hat{\mathbf{x}}_\mathbf{i} = \text{logistic}(W^*\mathbf{h} + \mathbf{c})$$



- 假定所有的输入是实数（$x_{ij} \in \mathbb{R}$）
- 下面哪个函数更适合作为 decoder?

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

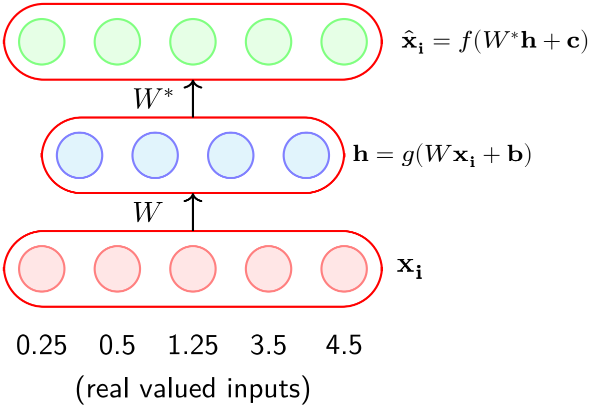
- logistic 和 *tanh* 如何?



- 假定所有的输入是实数 ($x_{ij} \in \mathbb{R}$)
- 下面哪个函数更适合作为 decoder?

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

- logistic 和 *tanh* 如何?
- 它们将限制重构的 $\hat{\mathbf{x}}_{\mathbf{i}}$ 在 [0,1] 或 [-1,1] 范围内，然而我们希望的是 $\hat{\mathbf{x}}_{\mathbf{i}} \in \mathbb{R}^n$



g 通常被选择为 sigmoid function

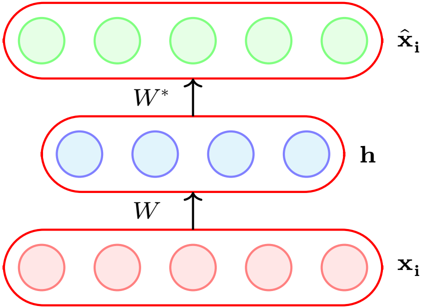
- 假定所有的输入是实数 ($x_{ij} \in \mathbb{R}$)
- 下面哪个函数更适合作为 decoder?

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

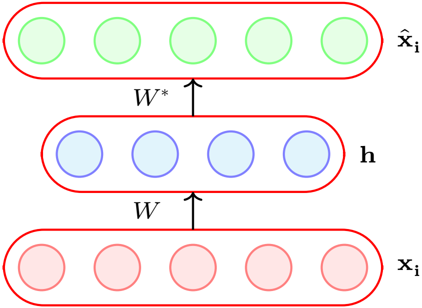
- logistic 和 *tanh* 如何?
- 它们将限制重构的 $\hat{\mathbf{x}}_\mathbf{i}$ 在 [0,1] 或 [-1,1] 范围内，然而我们希望的是 $\hat{\mathbf{x}}_\mathbf{i} \in \mathbb{R}^n$

The Road Ahead

- 选择 $f(\mathbf{x_i})$ 和 $g(\mathbf{x_i})$
- 选择 loss function

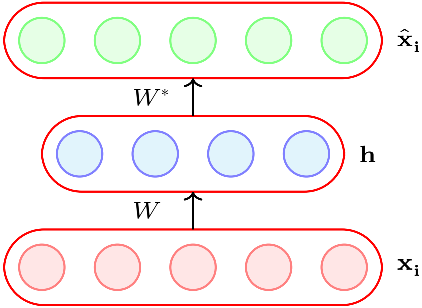


$$\mathbf{h} = g(W\mathbf{x}_\mathbf{i} + \mathbf{b})$$
$$\hat{\mathbf{x}}_\mathbf{i} = f(W^*\mathbf{h} + \mathbf{c})$$



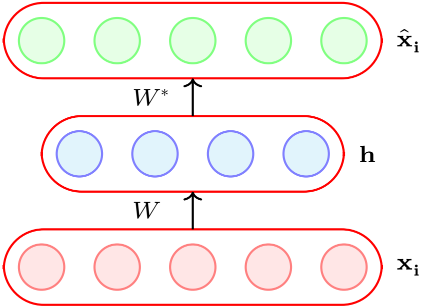
$$\mathbf{h} = g(W\mathbf{x}_\mathbf{i} + \mathbf{b})$$
$$\hat{\mathbf{x}}_\mathbf{i} = f(W^*\mathbf{h} + \mathbf{c})$$

- 考虑当输入是实数的情况



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

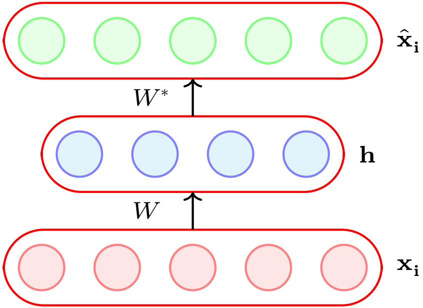
- 考虑当输入是实数的情况
- 自编码器的目标是让重构出的 $\hat{\mathbf{x}}_\mathbf{i}$ 尽可能的接近输入 $\mathbf{x}_\mathbf{i}$



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

- 考虑当输入是实数的情况
- 自编码器的目标是让重构出的 $\hat{\mathbf{x}}_\mathbf{i}$ 尽可能的接近输入 $\mathbf{x}_\mathbf{i}$
- 可以通过最小化下面的目标函数:

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

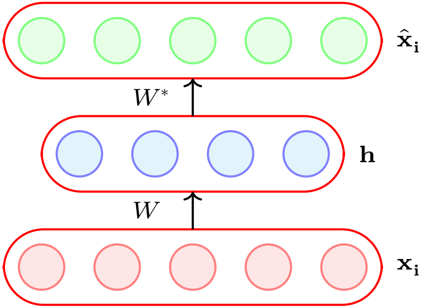


$$\mathbf{h} = g(W\mathbf{x}_\mathbf{i} + \mathbf{b})$$
$$\hat{\mathbf{x}}_\mathbf{i} = f(W^*\mathbf{h} + \mathbf{c})$$

- 考虑当输入是实数的情况
- 自编码器的目标是让重构出的 $\hat{\mathbf{x}}_\mathbf{i}$ 尽可能的接近输入 $\mathbf{x}_\mathbf{i}$
- 可以通过最小化下面的目标函数:

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

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



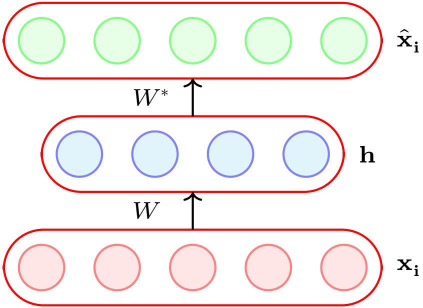
$$\mathbf{h} = g(W\mathbf{x}_\mathbf{i} + \mathbf{b})$$
$$\hat{\mathbf{x}}_\mathbf{i} = f(W^*\mathbf{h} + \mathbf{c})$$

- 考虑当输入是实数的情况
- 自编码器的目标是让重构出的 $\hat{\mathbf{x}}_\mathbf{i}$ 尽可能的接近输入 $\mathbf{x}_\mathbf{i}$
- 可以通过最小化下面的目标函数:

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

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

- 训练自编码器就像使用反向传播训练前馈神经网络一样



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

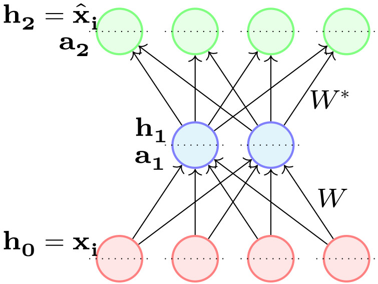
- 考虑当输入是实数的情况
- 自编码器的目标是让重构出的 $\hat{\mathbf{x}}_\mathbf{i}$ 尽可能的接近输入 $\mathbf{x}_\mathbf{i}$
- 可以通过最小化下面的目标函数:

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

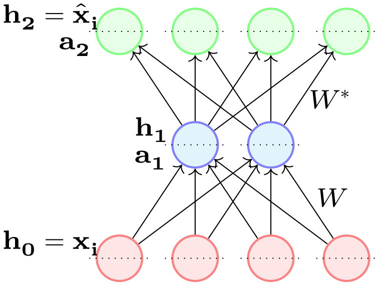
$$i.e., \underset{W,W^*,\mathbf{c},\mathbf{b}}{min} \frac{1}{m}\sum_{i=1}^{m}(\hat{\mathbf{x}}_\mathbf{i} - \mathbf{x}_\mathbf{i})^T(\hat{\mathbf{x}}_\mathbf{i} - \mathbf{x}_\mathbf{i})$$

- 训练自编码器就像使用反向传播训练前馈神经网络一样
- 所需要的只是求出 $\frac{\partial \mathcal{L}(\theta)}{\partial W^*}$ 和 $\frac{\partial \mathcal{L}(\theta)}{\partial W}$

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

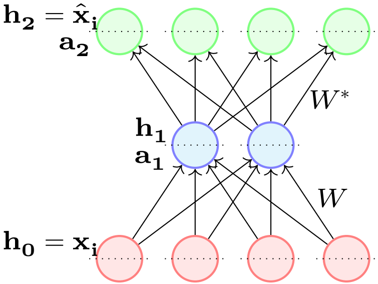


$$\mathcal{L}(\theta) = (\hat{\mathbf{x}}_\mathbf{i} - \mathbf{x}_\mathbf{i})^T(\hat{\mathbf{x}}_\mathbf{i} - \mathbf{x}_\mathbf{i})$$



- 注意：损失函数只由一个训练样本计算而得

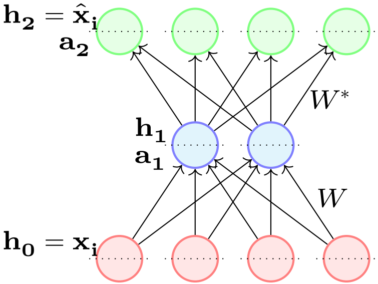
$$\mathcal{L}(\theta) = (\hat{\mathbf{x}}_\mathbf{i} - \mathbf{x}_\mathbf{i})^T (\hat{\mathbf{x}}_\mathbf{i} - \mathbf{x}_\mathbf{i})$$



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

- 注意：损失函数只由一个训练样本计算而得

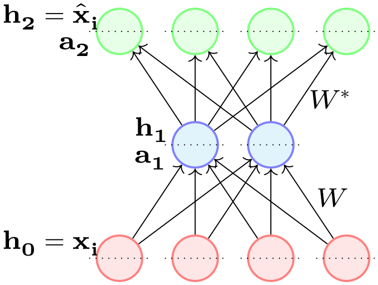
$$\mathcal{L}(\theta) = (\hat{\mathbf{x}}_\mathbf{i} - \mathbf{x}_\mathbf{i})^T(\hat{\mathbf{x}}_\mathbf{i} - \mathbf{x}_\mathbf{i})$$



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

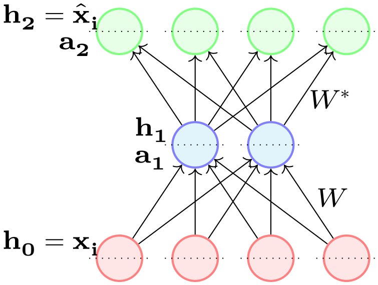
- 注意：损失函数只由一个训练样本计算而得

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



- 注意：损失函数只由一个训练样本计算而得

- $$\frac{\partial \mathcal{L}(\theta)}{\partial W^*} = \frac{\partial \mathcal{L}(\theta)}{\partial \mathbf{h_2}} \boxed{\frac{\partial \mathbf{h_2}}{\partial \mathbf{a_2}} \frac{\partial \mathbf{a_2}}{\partial W^*}}$$
- $$\frac{\partial \mathcal{L}(\theta)}{\partial W} = \frac{\partial \mathcal{L}(\theta)}{\partial \mathbf{h_2}} \boxed{\frac{\partial \mathbf{h_2}}{\partial \mathbf{a_2}} \frac{\partial \mathbf{a_2}}{\partial \mathbf{h_1}} \frac{\partial \mathbf{h_1}}{\partial \mathbf{a_1}} \frac{\partial \mathbf{a_1}}{\partial W}}$$
- 之前在讲反向传播时已经讲过如何计算框中的梯度

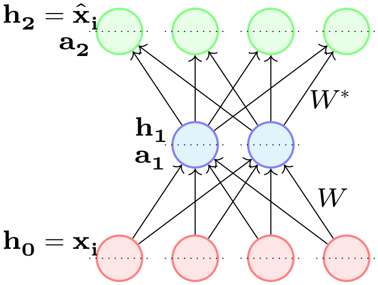


- $$\frac{\partial \mathcal{L}(\theta)}{\partial W^*} = \frac{\partial \mathcal{L}(\theta)}{\partial \mathbf{h_2}} \boxed{\frac{\partial \mathbf{h_2}}{\partial \mathbf{a_2}} \frac{\partial \mathbf{a_2}}{\partial W^*}}$$
- $$\frac{\partial \mathcal{L}(\theta)}{\partial W} = \frac{\partial \mathcal{L}(\theta)}{\partial \mathbf{h_2}} \boxed{\frac{\partial \mathbf{h_2}}{\partial \mathbf{a_2}} \frac{\partial \mathbf{a_2}}{\partial \mathbf{h_1}} \frac{\partial \mathbf{h_1}}{\partial \mathbf{a_1}} \frac{\partial \mathbf{a_1}}{\partial W}}$$
- 之前在讲反向传播时已经讲过如何计算框中的梯度

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

- 注意：损失函数只由一个训练样本计算而得

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

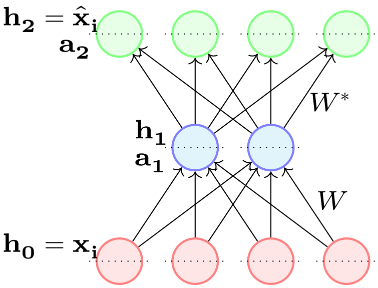


- 注意：损失函数只由一个训练样本计算而得

- $$\frac{\partial \mathcal{L}(\theta)}{\partial W^*} = \frac{\partial \mathcal{L}(\theta)}{\partial \mathbf{h_2}} \boxed{\frac{\partial \mathbf{h_2}}{\partial \mathbf{a_2}} \frac{\partial \mathbf{a_2}}{\partial W^*}}$$
- $$\frac{\partial \mathcal{L}(\theta)}{\partial W} = \frac{\partial \mathcal{L}(\theta)}{\partial \mathbf{h_2}} \boxed{\frac{\partial \mathbf{h_2}}{\partial \mathbf{a_2}} \frac{\partial \mathbf{a_2}}{\partial \mathbf{h_1}} \frac{\partial \mathbf{h_1}}{\partial \mathbf{a_1}} \frac{\partial \mathbf{a_1}}{\partial W}}$$
- 之前在讲反向传播时已经讲过如何计算框中的梯度

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

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



- 注意：损失函数只由一个训练样本计算而得

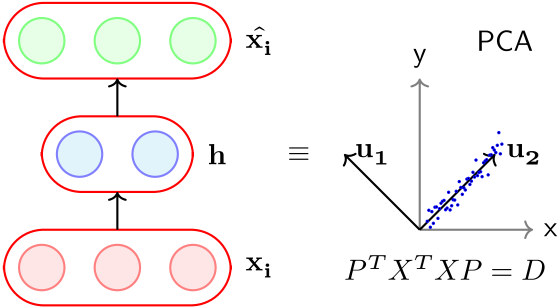
- $$\frac{\partial \mathcal{L}(\theta)}{\partial W^*} = \frac{\partial \mathcal{L}(\theta)}{\partial \mathbf{h_2}} \boxed{\frac{\partial \mathbf{h_2}}{\partial \mathbf{a_2}} \frac{\partial \mathbf{a_2}}{\partial W^*}}$$

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

- 之前在讲反向传播时已经讲过如何计算框中的梯度

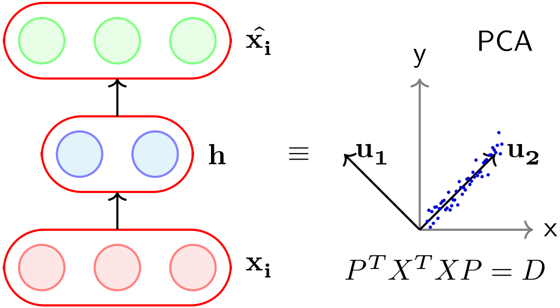
$$\begin{aligned}\frac{\partial \mathcal{L}(\theta)}{\partial \mathbf{h_2}} &= \frac{\partial \mathcal{L}(\theta)}{\partial \hat{\mathbf{x}}_\mathbf{i}} \\ &= \nabla_{\hat{\mathbf{x}}_\mathbf{i}} \{(\hat{\mathbf{x}}_\mathbf{i} - \mathbf{x}_\mathbf{i})^T(\hat{\mathbf{x}}_\mathbf{i} - \mathbf{x}_\mathbf{i})\}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

Link between PCA and Autoencoders

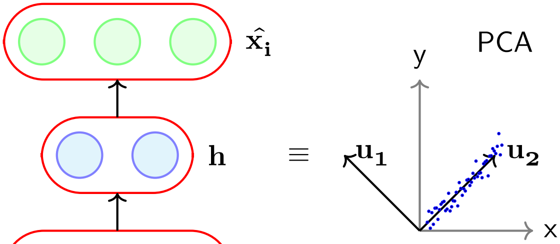


$$P^T X^T X P = D$$

- 自编码器的编码器部分等价于 PCA，如果满足

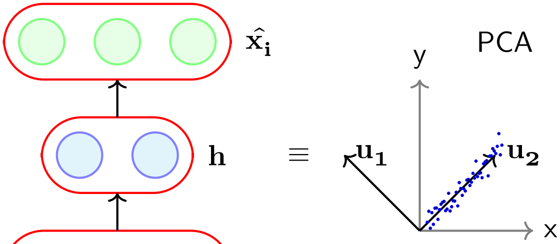


- 自编码器的编码器部分等价于 PCA，如果满足
  - 使用线性的编码器



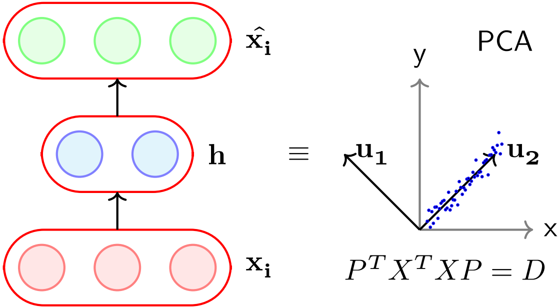
$$P^T X^T X P = D$$

- 自编码器的编码器部分等价于 PCA，如果满足
  - 使用线性的编码器
  - 使用线性的解码器



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

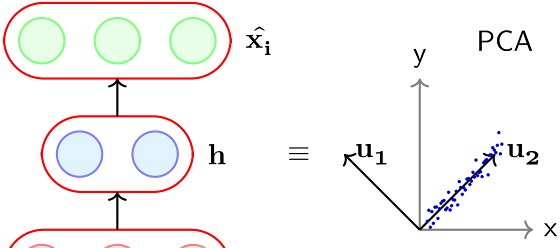
- 自编码器的编码器部分等价于 PCA，如果满足
  - 使用线性的编码器
  - 使用线性的解码器
  - 使用平方根误差损失函数 u



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

- 自编码器的编码器部分等价于 PCA，如果满足
  - 使用线性的编码器
  - 使用线性的解码器
  - 使用平方根误差损失函数 u
  - 对输入进行归一化

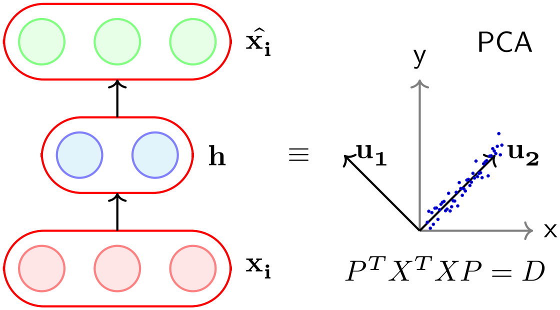
$$\hat{x}_{ij} = \frac{1}{\sqrt{m}}\left(x_{ij} - \frac{1}{m}\sum_{k=1}^{m} x_{kj}\right)$$



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

- 考虑对输入进行归一化的影响

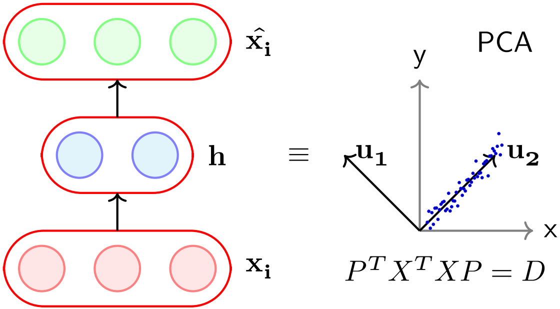
$$\hat{x}_{ij} = \frac{1}{\sqrt{m}}\left(x_{ij} - \frac{1}{m}\sum_{k=1}^{m} x_{kj}\right)$$



- 考虑对输入进行归一化的影响

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

- 括号中的操作确保数据沿着每个维度 j 有 0 均值 (减掉均值)

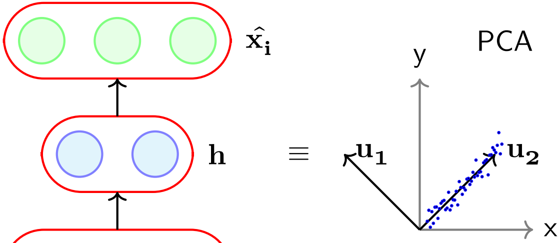


$$P^T X^T X P = D$$

- 考虑对输入进行归一化的影响

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

- 括号中的操作确保数据沿着每个维度 j 有 0 均值 (减掉均值)
- Let X' 是零均值归一化后的数据矩阵，则 $X = \frac{1}{\sqrt{m}} X'$

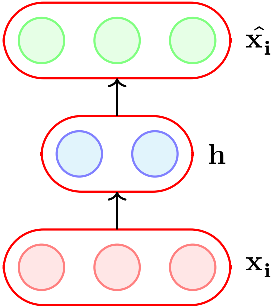


$$P^T X^T X P = D$$

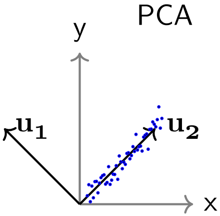
- 考虑对输入进行归一化的影响

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

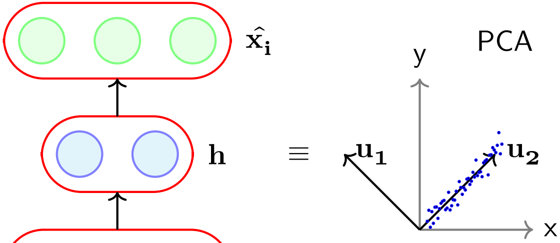
- 括号中的操作确保数据沿着每个维度 j 有 0 均值 (减掉均值)
- Let X' 是零均值归一化后的数据矩阵，则 $X = \frac{1}{\sqrt{m}} X'$
- 现在 $(X)^T X = \frac{1}{m}(X')^T X'$ 是协方差矩阵 (回忆 PCA 中的协方差矩阵)



$\equiv$

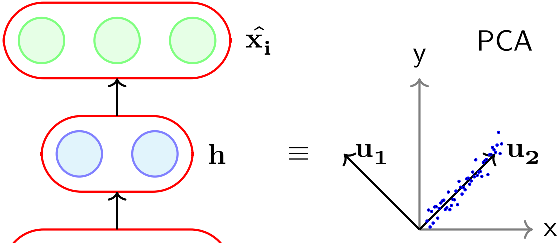


$$P^T X^T X P = D$$



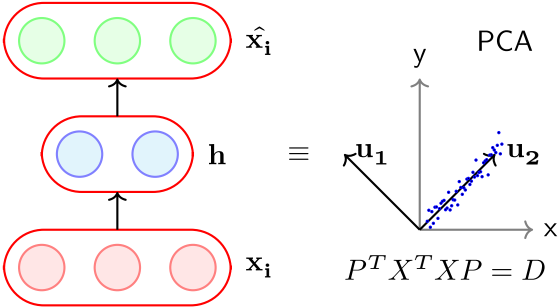
$$P^T X^T X P = D$$

- 当使用线性编码器和平方根误差损失函数时



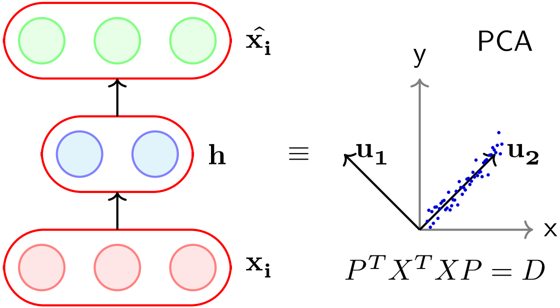
$$P^T X^T X P = D$$

- 当使用线性编码器和平方根误差损失函数时
- 优化解需要求解下面的目标函数



- 当使用线性编码器和平方根误差损失函数时
- 优化解需要求解下面的目标函数

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



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

- 当使用线性编码器和平方根误差损失函数时
- 优化解需要求解下面的目标函数

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

当使用线性编码器时可以得到其解

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

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

- 等价于

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

- 等价于

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

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

- 等价于

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

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

- 等价于

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

(将公式 (1) 写成矩阵形式，引入符号 $||A||_F$) (忽略了 biases)

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

- 等价于

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

(将公式 (1) 写成矩阵形式，引入符号 $||A||_F$) (忽略了 biases)

- 使用 SVD，上面优化问题的解是

$$HW^* = U_{.,\leq k} \Sigma_{k,k} V_{.,\leq k}^T$$

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

- 等价于

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

(将公式 (1) 写成矩阵形式，引入符号 $||A||_F$) (忽略了 biases)

- 使用 SVD，上面优化问题的解是

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

- 一个可能的解是

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

H 是一个线性编码器，求解编码器的权重 W

H 是一个线性编码器，求解编码器的权重 W

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

H 是一个线性编码器，求解编码器的权重 W

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

H 是一个线性编码器，求解编码器的权重 W

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

H 是一个线性编码器，求解编码器的权重 W

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

H 是一个线性编码器，求解编码器的权重 W

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

H 是一个线性编码器，求解编码器的权重 W

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

H 是一个线性编码器，求解编码器的权重 W

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

H 是一个线性编码器，求解编码器的权重 W

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

H 是一个线性编码器，求解编码器的权重 W

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

H 是一个线性编码器，求解编码器的权重 W

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

H 是一个线性编码器，求解编码器的权重 W

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

因此 H 是一个线性变换，$W = V_{\cdot,\leq k}$

- 编码器权重 $W = V_{\cdot,\leq k}$

- 编码器权重 $W = V_{\cdot,\leq k}$
- 根据 SVD, V 是由 $X^T X$ 的特征向量构成的矩阵

- 编码器权重 $W = V_{\cdot,\leq k}$
- 根据 SVD, V 是由 $X^T X$ 的特征向量构成的矩阵
- 根据 PCA, P 是由协方差矩阵的特征向量构成的矩阵

- 编码器权重 $W = V_{\cdot,\leq k}$
- 根据 SVD, V 是由 X^TX 的特征向量构成的矩阵
- 根据 PCA, P 是由协方差矩阵的特征向量构成的矩阵
- 之前看到，如果 X 经过如下归一化

- 编码器权重 $W = V_{\cdot,\leq k}$
- 根据 SVD, V 是由 X^TX 的特征向量构成的矩阵
- 根据 PCA, P 是由协方差矩阵的特征向量构成的矩阵
- 之前看到，如果 X 经过如下归一化

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

- 编码器权重 $W = V_{\cdot,\leq k}$
- 根据 SVD, V 是由 $X^T X$ 的特征向量构成的矩阵
- 根据 PCA, P 是由协方差矩阵的特征向量构成的矩阵
- 之前看到，如果 X 经过如下归一化

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

$X^T X$ 实际上就是样本的协方差矩阵

- 编码器权重 $W = V_{\cdot,\leq k}$
- 根据 SVD, V 是由 $X^T X$ 的特征向量构成的矩阵
- 根据 PCA, P 是由协方差矩阵的特征向量构成的矩阵
- 之前看到，如果 X 经过如下归一化

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

$X^T X$ 实际上就是样本的协方差矩阵

- 因此, 线性自编码器的编码器权重矩阵 (W) 和 PCA 的投影变换矩阵 (P) 实际上是一样的. 证毕

总结

线性自编码器的编码器等价于 PCA，如果满足

总结

线性自编码器的编码器等价于 PCA，如果满足

- 使用线性编码器

总结

线性自编码器的编码器等价于 PCA，如果满足

- 使用线性编码器
- 使用线性解码器

总结

线性自编码器的编码器等价于 PCA，如果满足

- 使用线性编码器
- 使用线性解码器
- 使用平方根误差损失函数

总结

线性自编码器的编码器等价于 PCA，如果满足

- 使用线性编码器
- 使用线性解码器
- 使用平方根误差损失函数
- 对输入进行如下归一化

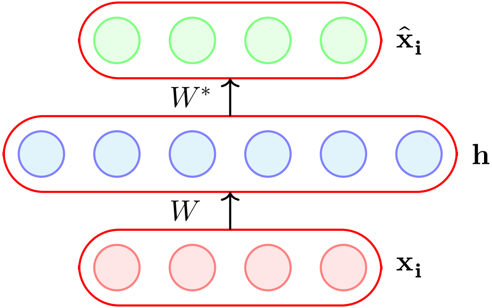
总结

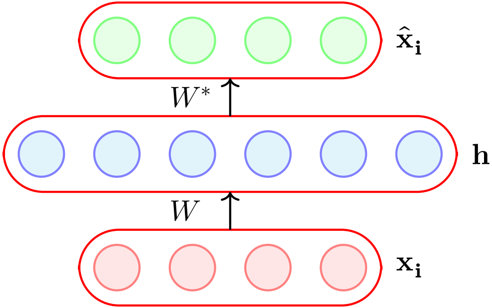
线性自编码器的编码器等价于 PCA，如果满足

- 使用线性编码器
- 使用线性解码器
- 使用平方根误差损失函数
- 对输入进行如下归一化

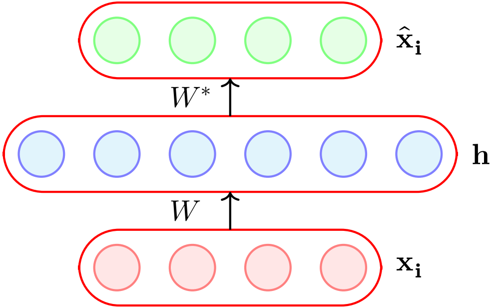
$$\hat{x}_{ij} = \frac{1}{\sqrt{m}}\left(x_{ij} - \frac{1}{m}\sum_{k=1}^{m} x_{kj}\right)$$

Regularization in autoencoders (Motivation)

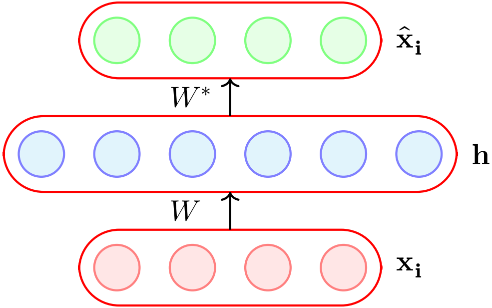

$\hat{\mathbf{x}}_\mathbf{i}$
W^*
$\mathbf{h}$
W
$\mathbf{x}_\mathbf{i}$



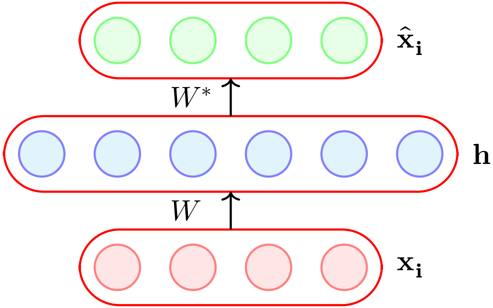
- 自编码器的泛化能力强差，特别是对于过完备的自编码器



- 自编码器的泛化能力强差，特别是对于过完备的自编码器
- 对于过完备的自编码器，模型会简单地将 $\mathbf{x}_\mathbf{i}$ 复制进 $\mathbf{h}$，然后再把 $\mathbf{h}$ 复制进 $\hat{\mathbf{x}}_\mathbf{i}$

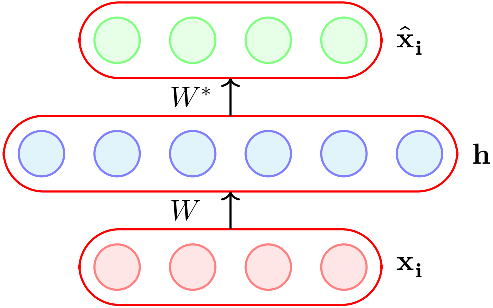


- 自编码器的泛化能力强差，特别是对于过完备的自编码器
- 对于过完备的自编码器，模型会简单地将 $\mathbf{x}_\mathbf{i}$ 复制进 $\mathbf{h}$，然后再把 $\mathbf{h}$ 复制进 $\hat{\mathbf{x}}_\mathbf{i}$
- 为了避免较差的泛化能力，可以在目标函数中引入正则项



- 最简单的正则项是在目标函数中增加一个 L_2-regularization 项

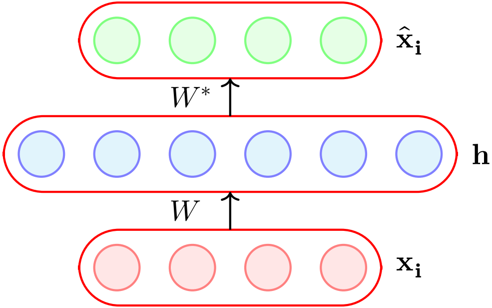
$$\min_{\theta=\{W,W^*,\mathbf{b},\mathbf{c}\}} \frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{n}(\hat{x}_{ij}-x_{ij})^2+\lambda\|\theta\|^2$$



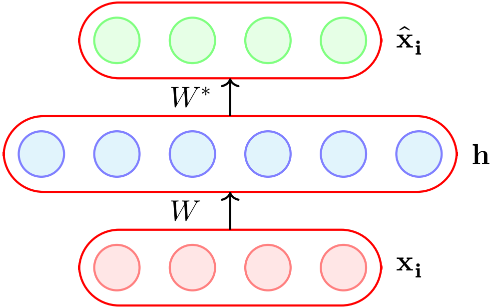
- 最简单的正则项是在目标函数中增加一个 L$_2$-regularization 项

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

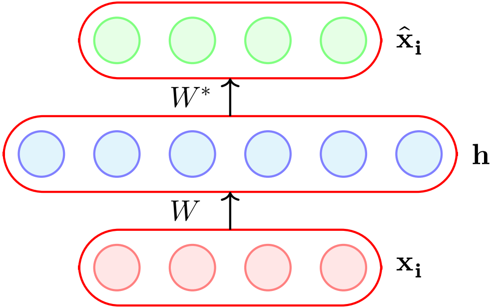
- 在使用梯度下降法求解最优参数时，只需要在梯度项 $\frac{\partial\mathcal{L}(\theta)}{\partial W}$ 中增加一项 λW (求解其他参数时，有类似的操作)



- 另一个方案是对编码器和解码器的权重进行捆绑（tie the weights），

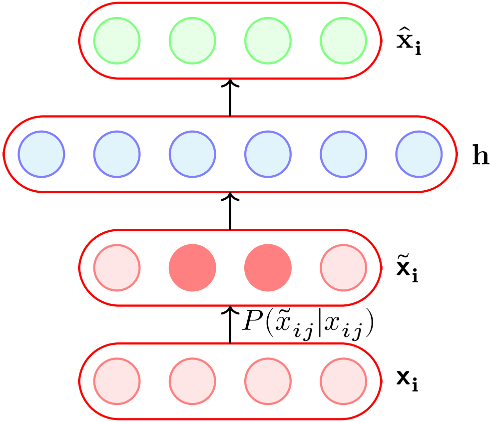


- 另一个方案是对编码器和解码器的权重进行捆绑（tie the weights），i.e., $W^* = W^T$

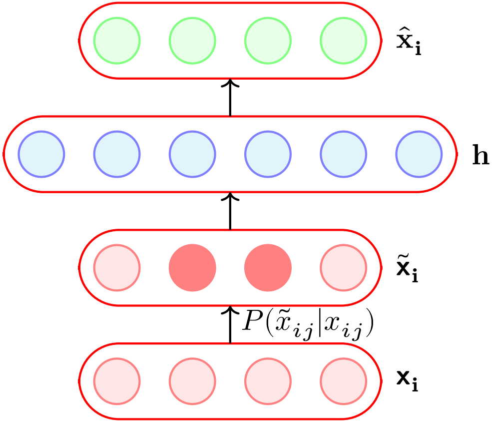


- 另一个方案是对编码器和解码器的权重进行捆绑（tie the weights），i.e., $W^* = W^T$
- 这种方案有效地减少了自编码器的容量，也能起到正则化的作用

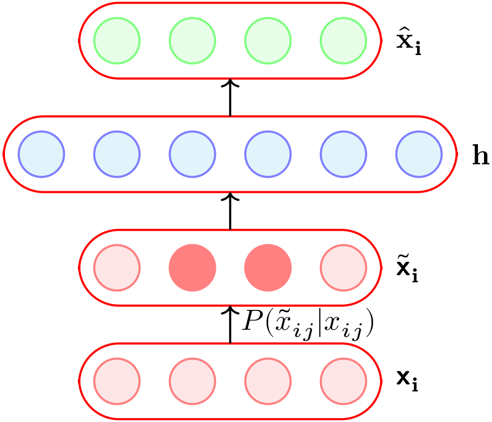
去噪自编码器（Denoising Autoencoders）



- 一个去噪编码器在处理输入数据时，简单地使用一个概率分布 $(P(\tilde{x}_{ij}|x_{ij}))$ 对输入数据施加噪声

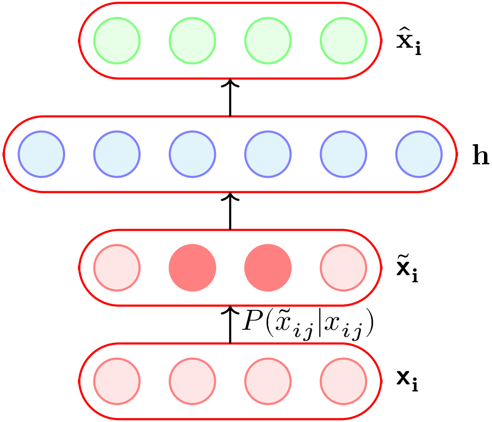


- 一个去噪编码器在处理输入数据时，简单地使用一个概率分布 $(P(\tilde{x}_{ij}|x_{ij}))$ 对输入数据施加噪声
- 在实际中，一个简单的概率分布 $P(\tilde{x}_{ij}|x_{ij})$ 可以使用下面的



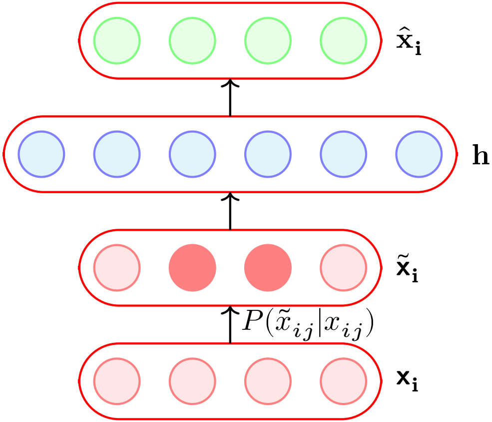
- 一个去噪编码器在处理输入数里时，简单地使用一个概率分布 $(P(\tilde{x}_{ij}|x_{ij}))$ 对输入数据施加噪声
- 在实际中，一个简单的概率分布 $P(\tilde{x}_{ij}|x_{ij})$ 可以使用下面的

$$P(\tilde{x}_{ij}=0|x_{ij})=q$$



- 一个去噪编码器在处理输入数据时，简单地使用一个概率分布 $(P(\tilde{x}_{ij}|x_{ij}))$ 对输入数据施加噪声
- 在实际中，一个简单的概率分布 $P(\tilde{x}_{ij}|x_{ij})$ 可以使用下面的

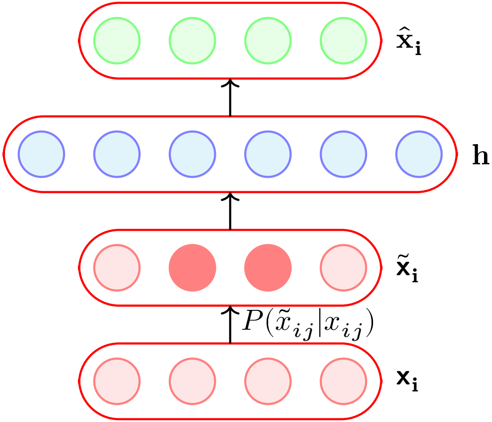
$$P(\tilde{x}_{ij} = 0|x_{ij}) = q$$
$$P(\tilde{x}_{ij} = x_{ij}|x_{ij}) = 1 - q$$



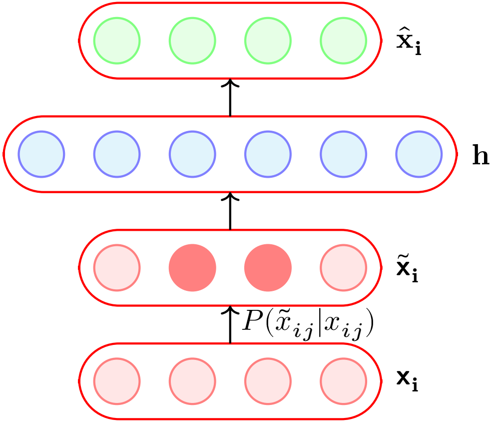
- 一个去噪编码器在处理输入数据时，简单地使用一个概率分布 $(P(\tilde{x}_{ij}|x_{ij}))$ 对输入数据施加噪声
- 在实际中，一个简单的概率分布 $P(\tilde{x}_{ij}|x_{ij})$ 可以使用下面的

$$P(\tilde{x}_{ij} = 0|x_{ij}) = q$$
$$P(\tilde{x}_{ij} = x_{ij}|x_{ij}) = 1 - q$$

- 换句话说，以概率 q 将输入数据置成 0，以概率 $(1-q)$ 保持原来的输入

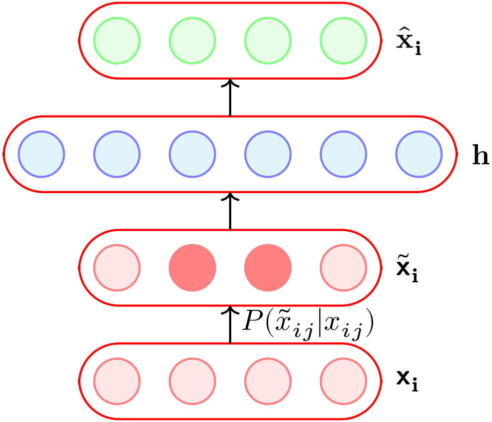


- 对输入数据施加噪声有什么作用？



- 对输入数据施加噪声有什么作用？
- 优化的目标仍然是重构原始的输入（没有加噪声）$\mathbf{x}_i$

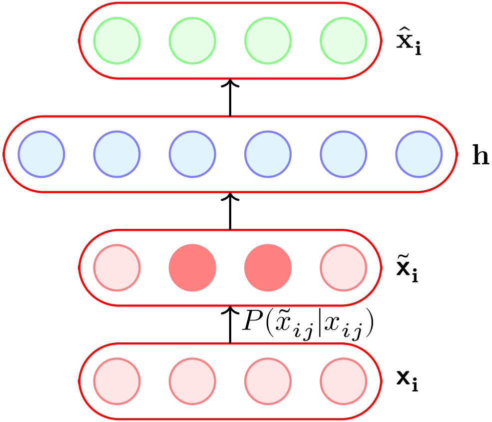
$$\underset{\theta}{arg\,min}\frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{n}(\hat{x}_{ij}-x_{ij})^2$$



- 对输入数据施加噪声有什么作用？
- 优化的目标仍然是重构原始的输入（没有加噪声）$\mathbf{x}_i$

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

- 因此，如果模型简单地将加噪声之后的数据 I $\widetilde{\mathbf{x}_i}$ 复制进 $h(\widetilde{\mathbf{x}_i})$，然后再复制进 $\hat{\mathbf{x}}_i$，会导致目标函数不是最优的，因此能够避免这种简单的复制



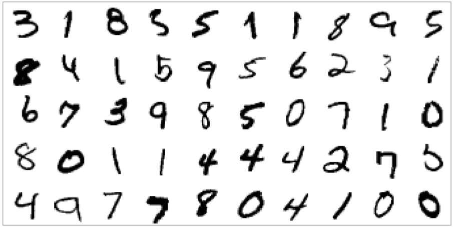
- 对输入数据施加噪声有什么作用？
- 优化的目标仍然是重构原始的输入（没有加噪声）$\mathbf{x}_i$

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

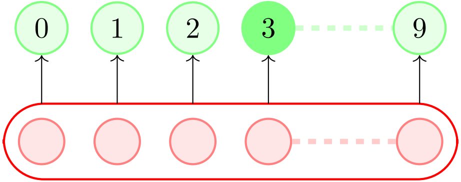
- 因此，如果模型简单地将加噪声之后的数据 I $\widetilde{\mathbf{x}_i}$ 复制进 $h(\widetilde{\mathbf{x}_i})$，然后再复制进 $\hat{\mathbf{x}}_i$，会导致目标函数不是最优的，因此能够避免这种简单的复制
- 模型将会正确地捕获输入数据的特性

下面看一个使用自编码器的实际应用，然后比较去噪自编码器和标准自编码器

任务：Hand-written digit recognition



MNIST Data

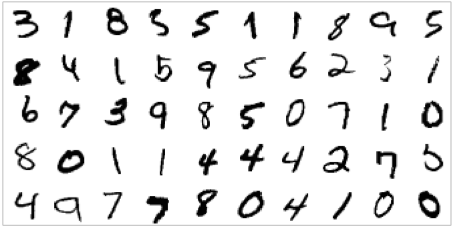


$$|\mathbf{x}_i| = 784 = 28 \times 28$$

28*28

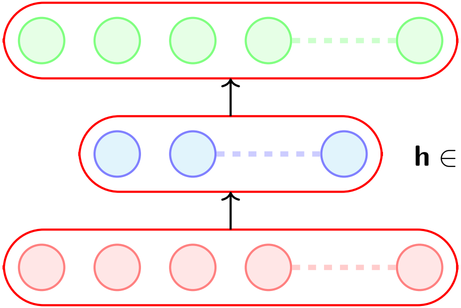
Basic approach (使用图像灰度作为特征)

Task: Hand-written digit recognition



MNIST Data

$\hat{\mathbf{x}}_i \in \mathbb{R}^{784}$

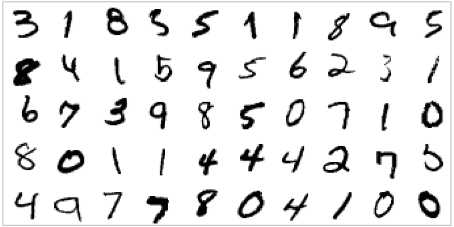


$\mathbf{h} \in \mathbb{R}^d$

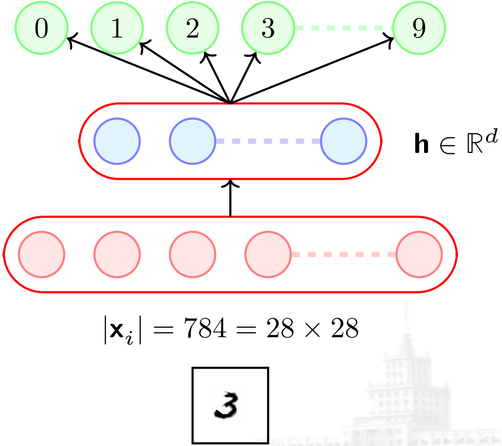
$|\mathbf{x}_i| = 784 = 28 \times 28$

AE approach (首先从数据中学习重要特征)

Task: Hand-written digit recognition

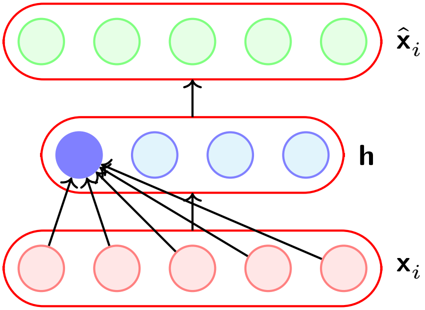


MNIST Data

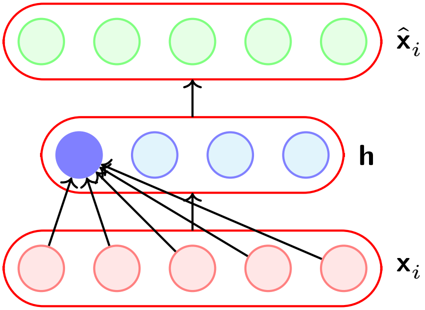


AE approach (然后，在隐含表示上学习一个分类器)

下面对 AEs 进行可视化，比较不同的 AEs



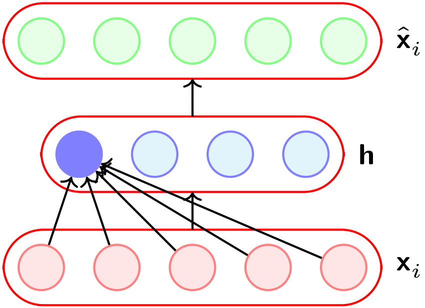
- 隐含层的每一个神经元可以当作是一个 filter，能够对输入 $\mathbf{x}_i$ 的一种特征配置进行响应



- 隐含层的每一个神经元可以当作是一个 filter，能够对输入 $\mathbf{x}_i$ 的一种特征配置进行响应
- 例如，

$$\mathbf{h}_1 = \sigma(W_1^T \mathbf{x}_i) \ [ignoring \ bias \ b]$$

其中 W_1 是学习到的权重参数，用于将输入数据与第一个隐神经元联系起来

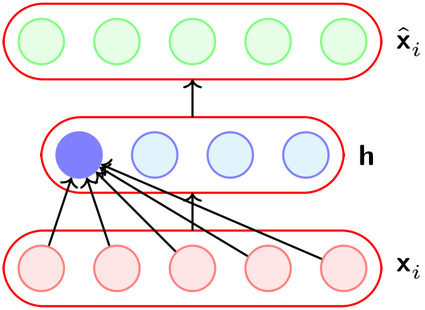


- 隐含层的每一个神经元可以当作是一个 filter，能够对输入 $\mathbf{x}_i$ 的一种特征配置进行响应
- 例如，

$$\mathbf{h}_1 = \sigma(W_1^T \mathbf{x}_i) \ [ignoring\ bias\ b]$$

其中 W_1 是学习到的权重参数，用于将输入数据与第一个隐神经元联系起来

- 什么样的 $\mathbf{x}_i$ 将导致 $\mathbf{h}_1$ 最大 (或者最大化地被激活)

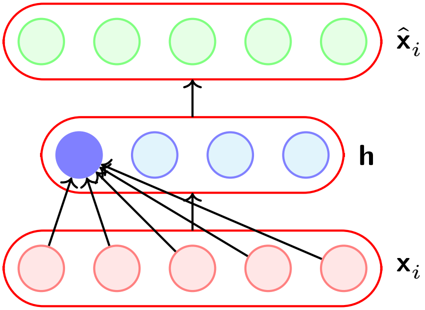


- 隐含层的每一个神经元可以当作是一个 filter，能够对输入 $\mathbf{x}_i$ 的一种特征配置进行响应
- 例如，

$$\mathbf{h}_1 = \sigma(W_1^T \mathbf{x}_i) \ [ignoring\ bias\ b]$$

其中 W_1 是学习到的权重参数，用于将输入数据与第一个隐神经元联系起来

- 什么样的 $\mathbf{x}_i$ 将导致 $\mathbf{h}_1$ 最大 (或者最大化地被激活)
- 假设输入数据已经归一化，使得 $\|\mathbf{x}_i\| = 1$



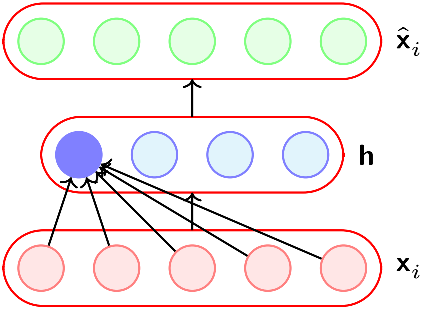
$$\max_{\mathbf{x}_i} \quad \{W_1^T \mathbf{x}_i\}$$

$$s.t. \quad ||\mathbf{x}_i||^2 = \mathbf{x}_i^T \mathbf{x}_i = 1$$

- 隐含层的每一个神经元可以当作是一个 filter，能够对输入 $\mathbf{x}_i$ 的一种特征配置进行响应
- 例如，

$$\mathbf{h}_1 = \sigma(W_1^T \mathbf{x}_i) \; [ignoring \; bias \; b]$$

  其中 W_1 是学习到的权重参数，用于将输入数据与第一个隐神经元联系起来
- 什么样的 $\mathbf{x}_i$ 将导致 $\mathbf{h}_1$ 最大 (或者最大化地被激活)
- 假设输入数据已经归一化，使得 $\|\mathbf{x}_i\| = 1$



$$\max_{\mathbf{x}_i} \quad \{W_1^T \mathbf{x}_i\}$$

$$s.t. \quad ||\mathbf{x}_i||^2 = \mathbf{x}_i^T \mathbf{x}_i = 1$$

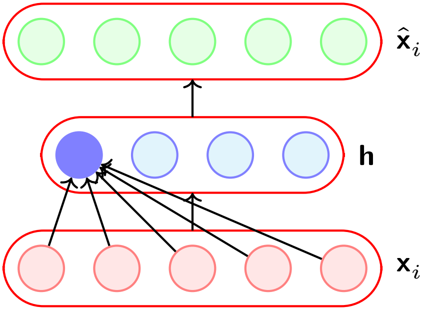
$$\text{Solution:} \quad \mathbf{x}_i = \frac{W_1}{\sqrt{W_1^T W_1}}$$

- 隐含层的每一个神经元可以当作是一个 filter，能够对输入 $\mathbf{x}_i$ 的一种特征配置进行响应
- 例如，

$$\mathbf{h}_1 = \sigma(W_1^T \mathbf{x}_i) \; [ignoring \; bias \; b]$$

其中 W_1 是学习到的权重参数，用于将输入数据与第一个隐神经元联系起来

- 什么样的 $\mathbf{x}_i$ 将导致 $\mathbf{h}_1$ 最大 (或者最大化地被激活)
- 假设输入数据已经归一化，使得 $\|\mathbf{x}_i\| = 1$



$$\max_{\mathbf{x}_i} \quad \{W_1^T \mathbf{x}_i\}$$

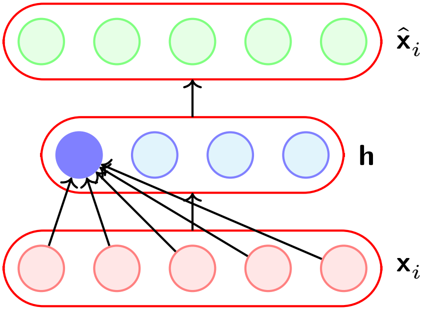
$$s.t. \quad ||\mathbf{x}_i||^2 = \mathbf{x}_i^T \mathbf{x}_i = 1$$

$$\text{Solution:} \quad \mathbf{x}_i = \frac{W_1}{\sqrt{W_1^T W_1}}$$

- 因此，输入

$$\mathbf{x}_i = \frac{W_1}{\sqrt{W_1^T W_1}}, \frac{W_2}{\sqrt{W_2^T W_2}}, \cdots \frac{W_n}{\sqrt{W_n^T W_n}}$$

将分别使得所有隐含神经元 $1, 2, \ldots, n$ 最大化



$$\max_{\mathbf{x}_i} \quad \{W_1^T \mathbf{x}_i\}$$

$$s.t. \quad ||\mathbf{x}_i||^2 = \mathbf{x}_i^T \mathbf{x}_i = 1$$

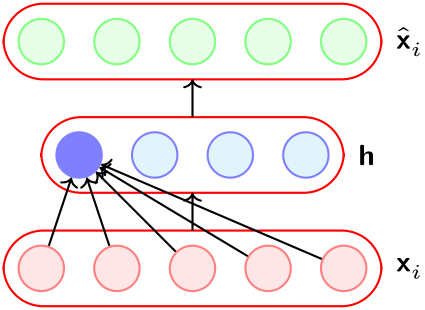
$$\text{Solution:} \quad \mathbf{x}_i = \frac{W_1}{\sqrt{W_1^T W_1}}$$

- 因此，输入

$$\mathbf{x}_i = \frac{W_1}{\sqrt{W_1^T W_1}}, \frac{W_2}{\sqrt{W_2^T W_2}}, \ldots \frac{W_n}{\sqrt{W_n^T W_n}}$$

将分别使得所有隐含神经元 1, 2, ..., n 最大化

- 可视化能够最大化前 k 个隐含神经元的图像 ($\mathbf{x}_i$'s)



$$\max_{\mathbf{x}_i} \quad \{W_1^T \mathbf{x}_i\}$$

$$s.t. \quad ||\mathbf{x}_i||^2 = \mathbf{x}_i^T \mathbf{x}_i = 1$$

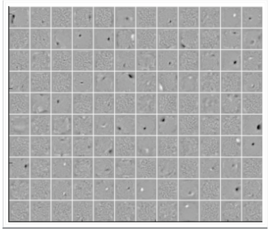
$$\text{Solution:} \quad \mathbf{x}_i = \frac{W_1}{\sqrt{W_1^T W_1}}$$

- 因此，输入

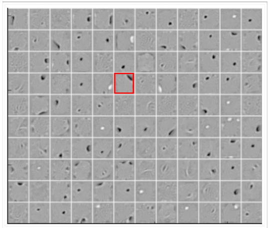
$$\mathbf{x}_i = \frac{W_1}{\sqrt{W_1^T W_1}}, \frac{W_2}{\sqrt{W_2^T W_2}}, \ldots \frac{W_n}{\sqrt{W_n^T W_n}}$$

将分别使得所有隐含神经元 $1, 2, \ldots, n$ 最大化

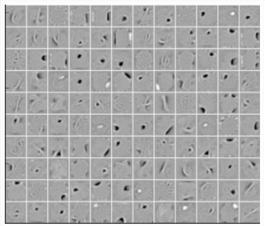
- 可视化能够最大化前 k 个隐含神经元的图像 ($\mathbf{x}_i$'s)
- 这些 $\mathbf{x}_i$'s 通过上面的公式计算，其中的权重 $(W_1, W_2 \ldots W_k)$ 由不同的自编码器学习而得



Vanilla AE (No noise)

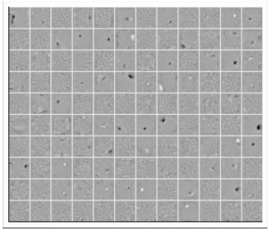


25% Denoising AE (q=0.25)

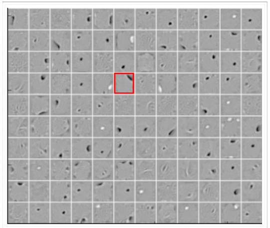


50% Denoising AE (q=0.5)

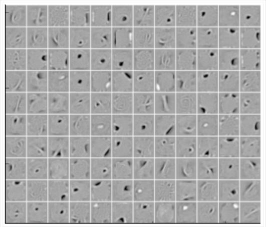
- The vanilla AE 没有学习到很多有意义的模式



Vanilla AE (No noise)

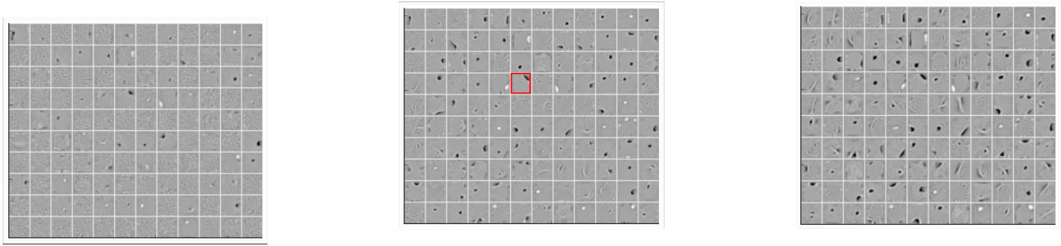


25% Denoising AE (q=0.25)



50% Denoising AE (q=0.5)

- The vanilla AE 没有学习到很多有意义的模式
- denoising AEs 的隐含神经元似乎像 pen-stroke detectors (例如，图中红色框标记的神经元，其黑色的模式像书写'0'，'2'，'3'，'8' 或'9' 时右上角会出现的一个笔触)



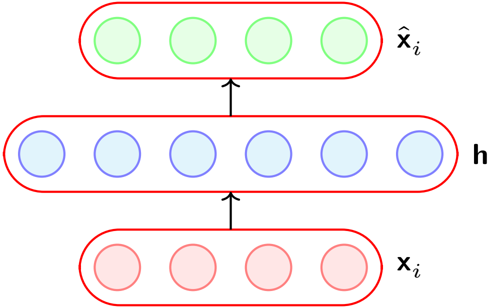
Vanilla AE (No noise)

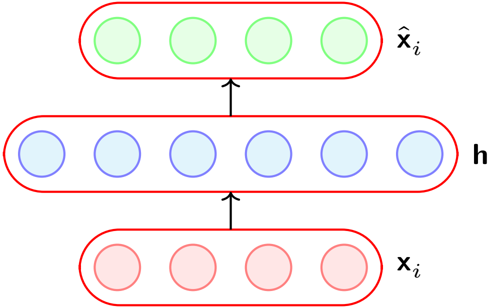
25% Denoising AE (q=0.25)

50% Denoising AE (q=0.5)

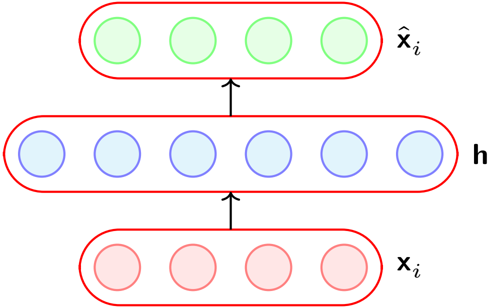
- The vanilla AE 没有学习到很多有意义的模式
- denoising AEs 的隐含神经元似乎像 pen-stroke detectors (例如，图中红色框标记的神经元，其黑色的模式像书写'0'，'2'，'3'，'8' 或'9' 时右上角会出现的一个笔触)
- 随着噪声的增加，filters 变得更宽，因为神经元需要依赖更多的相邻像素去表示一个笔触

稀疏自动编码器（Sparse Autoencoder）

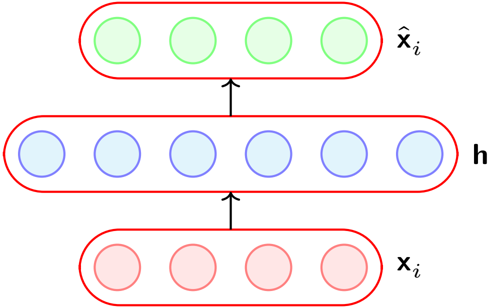

$\hat{\mathbf{x}}_i$
$\mathbf{h}$
$\mathbf{x}_i$



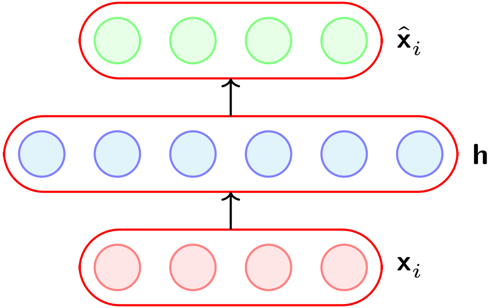
- 隐含层经过 Sigmoid 激活后，输出结果在 0 到 1 之间



- 隐含层经过 Sigmoid 激活后，输出结果在 0 到 1 之间
- 当神经元的输出接近 1，则认为该神经元被激活；当输出接近 0 时，则认为该神经元没有被激活



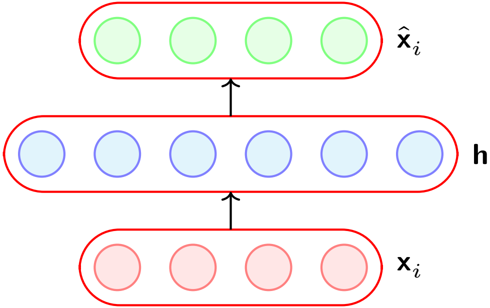
- 隐含层经过 Sigmoid 激活后，输出结果在 0 到 1 之间
- 当神经元的输出接近 1，则认为该神经元被激活；当输出接近 0 时，则认为该神经元没有被激活
- 稀疏自动编码器的目的是确保神经元在大部分时间处于非激活状态



第 l 个神经元的平均激活值:

$$\hat{\rho}_l = \frac{1}{m}\sum_{i=1}^{m} h(\mathbf{x}_i)_l$$

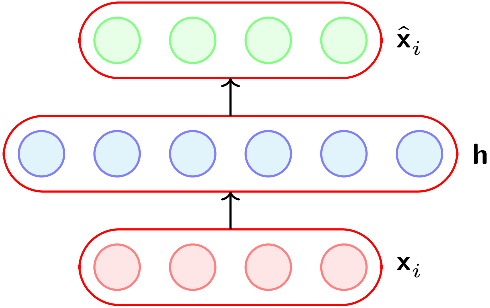
- 如果神经元 l 是稀疏的（大部分时间处于非激活状态），那么 $\hat{\rho}_l \to 0$



- 如果神经元 l 是稀疏的（大部分时间处于非激活状态），那么 $\hat{\rho}_l \to 0$
- 稀疏自编码器使用一个稀疏度参数 ρ（通常 ρ 非常接近于 0，例如 0.005）

第 l 个神经元的平均激活值：

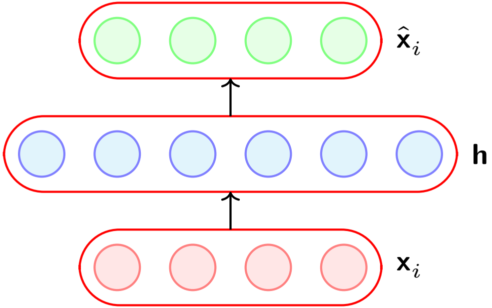
$$\hat{\rho}_l = \frac{1}{m}\sum_{i=1}^{m} h(\mathbf{x}_i)_l$$



- 如果神经元 l 是稀疏的（大部分时间处于非激活状态），那么 $\hat{\rho}_l \to 0$
- 稀疏自编码器使用一个稀疏度参数 ρ（通常 ρ 非常接近于 0，例如 0.005）
- 希望能满足 $\hat{\rho}_l = \rho$

第 l 个神经元的平均激活值：

$$\hat{\rho}_l = \frac{1}{m}\sum_{i=1}^{m} h(\mathbf{x}_i)_l$$



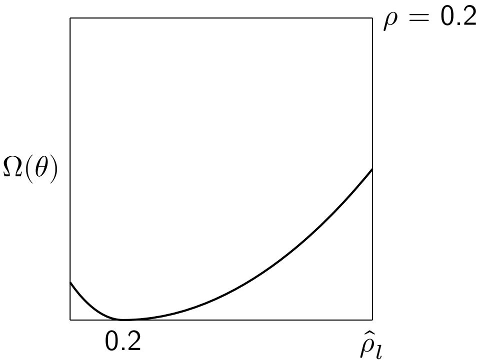
- 如果神经元 l 是稀疏的（大部分时间处于非激活状态），那么 $\hat{\rho}_l \to 0$
- 稀疏自编码器使用一个稀疏度参数 ρ（通常 ρ 非常接近于 0，例如 0.005）
- 希望能满足 $\hat{\rho}_l = \rho$
- ：可以通过将下面的这一项加入目标函数中来实现

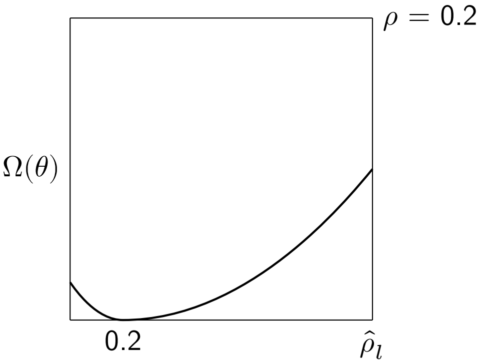
第 l 个神经元的平均激活值：

$$\hat{\rho}_l = \frac{1}{m}\sum_{i=1}^{m} h(\mathbf{x}_i)_l$$

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

- 这一项何时能达到最小值？最小值又是多少？可以将它的图画出来


$\rho = 0.2$
$\Omega(\theta)$
0.2
$\hat{\rho}_l$



- 当 $\hat{\rho}_l = \rho$ 时，函数值达到最小。

- 现在，新的目标函数为

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

- 现在，新的目标函数为

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

- $\mathcal{L}(\theta)$ 可以是平方误差损失或交叉熵损失，$\Omega(\theta)$ 是稀疏约束

- 现在，新的目标函数为

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

- $\mathcal{L}(\theta)$ 可以是平方误差损失或交叉熵损失，$\Omega(\theta)$ 是稀疏约束
- $\frac{\partial \mathcal{L}(\theta)}{\partial W}$ 之前计算过

- 现在，新的目标函数为

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

- $\mathcal{L}(\theta)$ 可以是平方误差损失或交叉熵损失，$\Omega(\theta)$ 是稀疏约束
- $\frac{\partial \mathcal{L}(\theta)}{\partial W}$ 之前计算过
- 如何计算 $\frac{\partial \Omega(\theta)}{\partial W}$?

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

- 现在，新的目标函数为

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

- $\mathcal{L}(\theta)$ 可以是平方误差损失或交叉熵损失，$\Omega(\theta)$ 是稀疏约束
- $\frac{\partial \mathcal{L}(\theta)}{\partial W}$ 之前计算过
- 如何计算 $\frac{\partial \Omega(\theta)}{\partial W}$ ？

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

重写为:

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

- 现在，新的目标函数为

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

- $\mathcal{L}(\theta)$ 可以是平方误差损失或交叉熵损失，$\Omega(\theta)$ 是稀疏约束
- $\frac{\partial\mathcal{L}(\theta)}{\partial W}$ 之前计算过
- 如何计算 $\frac{\partial\Omega(\theta)}{\partial W}$ ？

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

重写为:

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

通过链式法则:

$$\frac{\partial \Omega(\theta)}{\partial W} = \frac{\partial \Omega(\theta)}{\partial \hat{\boldsymbol{\rho}}} \cdot \frac{\partial \hat{\boldsymbol{\rho}}}{\partial W}$$

- 现在，新的目标函数为

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

- $\mathcal{L}(\theta)$ 可以是平方误差损失或交叉熵损失，$\Omega(\theta)$ 是稀疏约束
- $\frac{\partial \mathcal{L}(\theta)}{\partial W}$ 之前计算过
- 如何计算 $\frac{\partial \Omega(\theta)}{\partial W}$ ？

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

重写为:

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

通过链式法则:

$$\frac{\partial\Omega(\theta)}{\partial W}=\frac{\partial\Omega(\theta)}{\partial\hat{\boldsymbol{\rho}}}.\frac{\partial\hat{\boldsymbol{\rho}}}{\partial W}$$

$$\frac{\partial\Omega(\theta)}{\partial\hat{\boldsymbol{\rho}}}=\left[\frac{\partial\Omega(\theta)}{\partial\hat{\rho}_1},\frac{\partial\Omega(\theta)}{\partial\hat{\rho}_2},\dots\frac{\partial\Omega(\theta)}{\partial\hat{\rho}_k}\right]^T$$

- 现在，新的目标函数为

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

- $\mathcal{L}(\theta)$ 可以是平方误差损失或交叉熵损失，$\Omega(\theta)$ 是稀疏约束
- $\frac{\partial\mathcal{L}(\theta)}{\partial W}$ 之前计算过
- 如何计算 $\frac{\partial\Omega(\theta)}{\partial W}$?

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

重写为:

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

通过链式法则:

$$\frac{\partial\Omega(\theta)}{\partial W}=\frac{\partial\Omega(\theta)}{\partial\hat{\boldsymbol{\rho}}}.\frac{\partial\hat{\boldsymbol{\rho}}}{\partial W}$$

$$\frac{\partial\Omega(\theta)}{\partial\hat{\boldsymbol{\rho}}}=\left[\frac{\partial\Omega(\theta)}{\partial\hat{\rho}_1},\frac{\partial\Omega(\theta)}{\partial\hat{\rho}_2},\ldots\frac{\partial\Omega(\theta)}{\partial\hat{\rho}_k}\right]^T$$

对隐含层的每一个神经元 $l\in 1\ldots k$, 得到

- 现在，新的目标函数为

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

- $\mathcal{L}(\theta)$ 可以是平方误差损失或交叉熵损失，$\Omega(\theta)$ 是稀疏约束
- $\frac{\partial\mathcal{L}(\theta)}{\partial W}$ 之前计算过
- 如何计算 $\frac{\partial\Omega(\theta)}{\partial W}$?

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

重写为:

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

通过链式法则:

$$\frac{\partial \Omega(\theta)}{\partial W} = \frac{\partial \Omega(\theta)}{\partial \hat{\boldsymbol{\rho}}} . \frac{\partial \hat{\boldsymbol{\rho}}}{\partial W}$$

$$\frac{\partial \Omega(\theta)}{\partial \hat{\boldsymbol{\rho}}} = \left[\frac{\partial \Omega(\theta)}{\partial \hat{\rho}_1}, \frac{\partial \Omega(\theta)}{\partial \hat{\rho}_2}, \ldots \frac{\partial \Omega(\theta)}{\partial \hat{\rho}_k} \right]^T$$

对隐含层的每一个神经元 $l \in 1 \ldots k$, 得到

$$\frac{\partial \Omega(\theta)}{\partial \hat{\rho}_l} = -\frac{\rho}{\hat{\rho}_l} + \frac{(1-\rho)}{1-\hat{\rho}_l}$$

- 现在，新的目标函数为

$$\hat{\mathscr{L}}(\theta) = \mathscr{L}(\theta) + \Omega(\theta)$$

- $\mathscr{L}(\theta)$ 可以是平方误差损失或交叉熵损失，$\Omega(\theta)$ 是稀疏约束
- $\frac{\partial \mathscr{L}(\theta)}{\partial W}$ 之前计算过
- 如何计算 $\frac{\partial \Omega(\theta)}{\partial W}$?

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

重写为:

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

通过链式法则:

$$\frac{\partial\Omega(\theta)}{\partial W}=\frac{\partial\Omega(\theta)}{\partial\hat{\boldsymbol{\rho}}}.\frac{\partial\hat{\boldsymbol{\rho}}}{\partial W}$$

$$\frac{\partial\Omega(\theta)}{\partial\hat{\boldsymbol{\rho}}}=\left[\frac{\partial\Omega(\theta)}{\partial\hat{\rho}_1},\frac{\partial\Omega(\theta)}{\partial\hat{\rho}_2},\ldots\frac{\partial\Omega(\theta)}{\partial\hat{\rho}_k}\right]^T$$

对隐含层的每一个神经元 $l\in 1\ldots k$，得到

$$\frac{\partial\Omega(\theta)}{\partial\hat{\rho}_l}=-\frac{\rho}{\hat{\rho}_l}+\frac{(1-\rho)}{1-\hat{\rho}_l}$$

及 $\frac{\partial\hat{\rho}_l}{\partial W}=\mathbf{x}_i(g'(W^T\mathbf{x}_i+\mathbf{b}))^T$(推导过程见下一页)

- 现在，新的目标函数为

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

- $\mathcal{L}(\theta)$ 可以是平方误差损失或交叉熵损失，$\Omega(\theta)$ 是稀疏约束
- $\frac{\partial\mathcal{L}(\theta)}{\partial W}$ 之前计算过
- 如何计算 $\frac{\partial\Omega(\theta)}{\partial W}$ ？

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

重写为:

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

通过链式法则:

$$\frac{\partial \Omega(\theta)}{\partial W} = \frac{\partial \Omega(\theta)}{\partial \hat{\boldsymbol{\rho}}} \cdot \frac{\partial \hat{\boldsymbol{\rho}}}{\partial W}$$

$$\frac{\partial \Omega(\theta)}{\partial \hat{\boldsymbol{\rho}}} = \left[\frac{\partial \Omega(\theta)}{\partial \hat{\rho}_1}, \frac{\partial \Omega(\theta)}{\partial \hat{\rho}_2}, \ldots \frac{\partial \Omega(\theta)}{\partial \hat{\rho}_k} \right]^T$$

对隐含层的每一个神经元 $l \in 1 \ldots k$, 得到

$$\frac{\partial \Omega(\theta)}{\partial \hat{\rho}_l} = -\frac{\rho}{\hat{\rho}_l} + \frac{(1-\rho)}{1-\hat{\rho}_l}$$

及 $\frac{\partial \hat{\rho}_l}{\partial W} = \mathbf{x}_i (g'(W^T \mathbf{x}_i + \mathbf{b}))^T$(推导过程见下一页)

- 现在，新的目标函数为

$$\hat{\mathscr{L}}(\theta) = \mathscr{L}(\theta) + \Omega(\theta)$$

- $\mathscr{L}(\theta)$ 可以是平方误差损失或交叉熵损失，$\Omega(\theta)$ 是稀疏约束
- $\frac{\partial \mathscr{L}(\theta)}{\partial W}$ 之前计算过
- 如何计算 $\frac{\partial \Omega(\theta)}{\partial W}$?
- 最终,

$$\frac{\partial \hat{\mathscr{L}}(\theta)}{\partial W} = \frac{\partial \mathscr{L}(\theta)}{\partial W} + \frac{\partial \Omega(\theta)}{\partial W}$$

求导过程

$$\frac{\partial \hat{\rho}}{\partial W} = \left[\frac{\partial \hat{\rho}_1}{\partial W} \quad \frac{\partial \hat{\rho}_2}{\partial W} \cdots \frac{\partial \hat{\rho}_k}{\partial W}\right]$$

对上述矩阵中的每个元素，计算偏导数 $\frac{\partial \hat{\rho}_l}{\partial W}$
其中，对矩阵 W 中的每个元素 W_{jl}

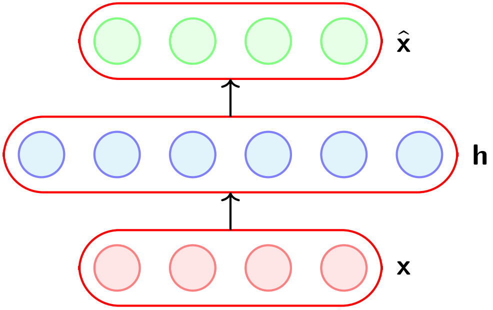
$$\begin{aligned}
\frac{\partial \hat{\rho}_l}{\partial W_{jl}} &= \frac{\partial\left[\frac{1}{m}\sum_{i=1}^{m} g(W_{:,l}^T \mathbf{x_i} + b_l)\right]}{\partial W_{jl}} \\
&= \frac{1}{m}\sum_{i=1}^{m} \frac{\partial\left[g(W_{:,l}^T \mathbf{x_i} + b_l)\right]}{\partial W_{jl}} \\
&= \frac{1}{m}\sum_{i=1}^{m} g'(W_{:,l}^T \mathbf{x_i} + b_l) x_{ij}
\end{aligned}$$

矩阵形式:

$$\frac{\partial \hat{\rho}_l}{\partial W} = \mathbf{x}_i (g'(W^T \mathbf{x}_i + \mathbf{b}))^T$$

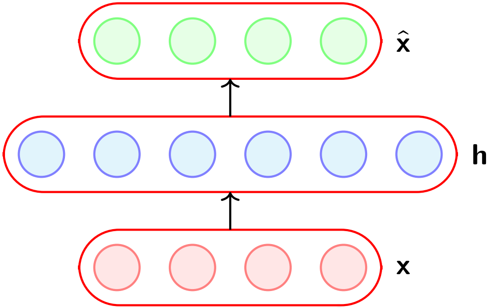
收缩自编码器（Contractive Autoencoders）

- 收缩自编码器的设计是为了防止学习特定映射函数时，产生 overcomplete (隐藏层神经元个数大于输入层神经元个数)



- 收缩自编码器的设计是为了防止学习特定映射函数时，产生 overcomplete（隐藏层神经元个数大于输入层神经元个数）
- 为了实现该目的，可以向损失函数添加以下正则化项

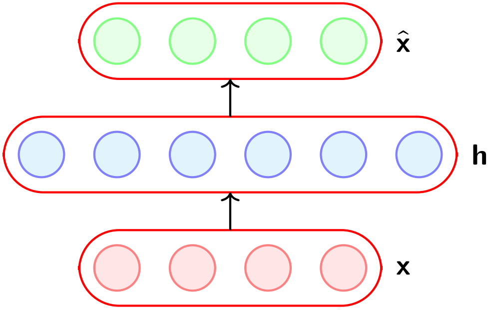
$$\Omega(\theta) = \|J_{\mathbf{x}}(\mathbf{h})\|_F^2$$



- 收缩自编码器的设计是为了防止学习特定映射函数时，产生 overcomplete（隐藏层神经元个数大于输入层神经元个数）
- 为了实现该目的，可以向损失函数添加以下正则化项

$$\Omega(\theta) = \|J_{\mathbf{x}}(\mathbf{h})\|_F^2$$

其中 $J_{\mathbf{x}}(\mathbf{h})$ 是编码器的 Jacobian 矩阵



- 假设输入为 n 维，隐藏层为 k 维，则

- 假设输入为 n 维，隐藏层为 k 维，则

$$J_{\mathbf{x}}(\mathbf{h}) = \begin{bmatrix} \frac{\partial h_1}{\partial x_1} & \cdots & \cdots & \cdots & \frac{\partial h_1}{\partial x_n} \\ \frac{\partial h_2}{\partial x_1} & \cdots & \cdots & \cdots & \frac{\partial h_2}{\partial x_n} \\ \vdots & & \ddots & & \vdots \\ \frac{\partial h_k}{\partial x_1} & \cdots & \cdots & \cdots & \frac{\partial h_k}{\partial x_n} \end{bmatrix}$$

- 假设输入为 n 维，隐藏层为 k 维，则
- Jacobian 矩阵的 (l,j) 项能够反映第 j 个输入的小变化量对第 l 个神经元的输出的影响

$$J_{\mathbf{x}}(\mathbf{h}) = \begin{bmatrix} \frac{\partial h_1}{\partial x_1} & \cdots & \cdots & \cdots & \frac{\partial h_1}{\partial x_n} \\ \frac{\partial h_2}{\partial x_1} & \cdots & \cdots & \cdots & \frac{\partial h_2}{\partial x_n} \\ \vdots & & \ddots & & \vdots \\ \frac{\partial h_k}{\partial x_1} & \cdots & \cdots & \cdots & \frac{\partial h_k}{\partial x_n} \end{bmatrix}$$

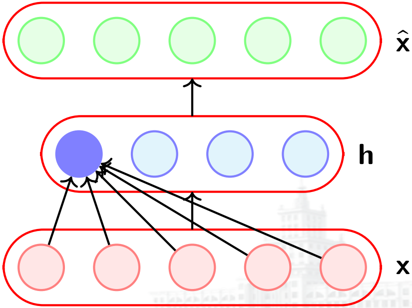
- 假设输入为 n 维，隐藏层为 k 维，则
- Jacobian 矩阵的 (l, j) 项能够反映第 j 个输入的小变化量对第 l 个神经元的输出的影响

$$J_{\mathbf{x}}(\mathbf{h}) = \begin{bmatrix} \frac{\partial h_1}{\partial x_1} & \cdots & \cdots & \cdots & \frac{\partial h_1}{\partial x_n} \\ \frac{\partial h_2}{\partial x_1} & \cdots & \cdots & \cdots & \frac{\partial h_2}{\partial x_n} \\ \vdots & & \ddots & & \vdots \\ \frac{\partial h_k}{\partial x_1} & \cdots & \cdots & \cdots & \frac{\partial h_k}{\partial x_n} \end{bmatrix}$$

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

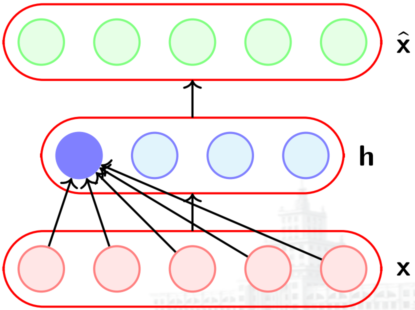
- 如何理解上述内容?

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



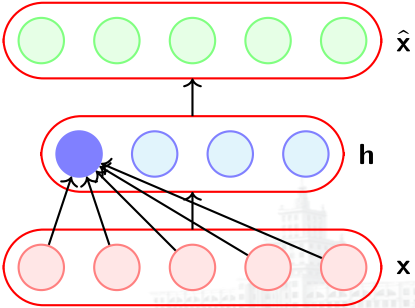
- 如何理解上述内容?
- 对于右式中的 $\frac{\partial h_1}{\partial x_1}$，当 $\frac{\partial h_1}{\partial x_1}=0$ 时，意味着这个神经元对于输入 x_1 不敏感

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

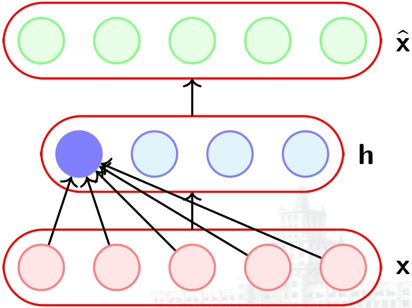


- 如何理解上述内容?
- 对于右式中的 $\frac{\partial h_1}{\partial x_1}$，当 $\frac{\partial h_1}{\partial x_1} = 0$ 时，意味着这个神经元对于输入 x_1 不敏感
- 但是想要最小化 $\mathcal{L}(\theta)$，要求 $\mathbf{h}$ 捕获输入的变化

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

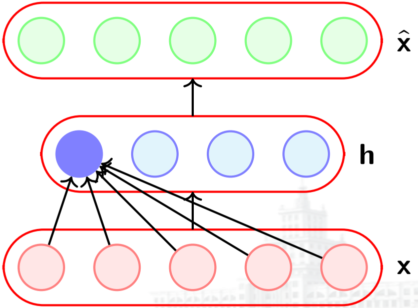


$$\|J_{\mathbf{x}}(\mathbf{h})\|_F^2 = \sum_{j=1}^{n}\sum_{l=1}^{k}\left(\frac{\partial h_l}{\partial x_j}\right)^2$$



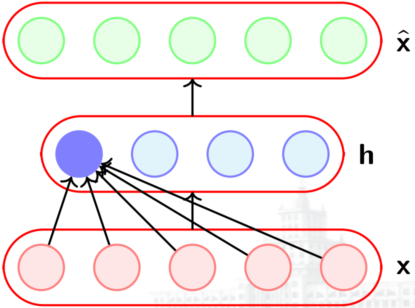
- 因此，通过将这两个目标对立，我们确保 **h** 仅对非常重要的变化敏感。

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



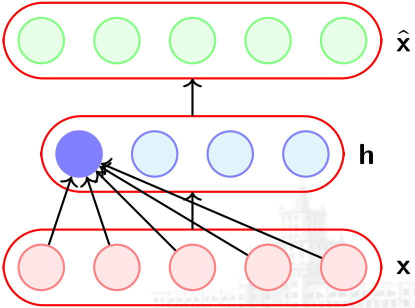
- 因此，通过将这两个目标对立，我们确保 **h** 仅对非常重要的变化敏感。
- $\mathcal{L}(\theta)$ - 捕获数据中重要的变化

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



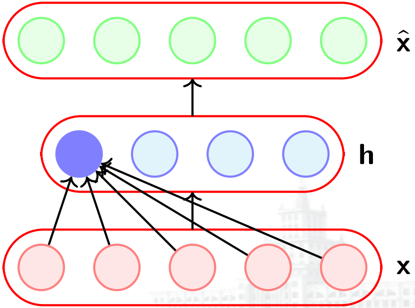
- 因此，通过将这两个目标对立，我们确保 **h** 仅对非常重要的变化敏感。
- $\mathcal{L}(\theta)$ - 捕获数据中重要的变化
- $\Omega(\theta)$ - 不捕获数据中的变化

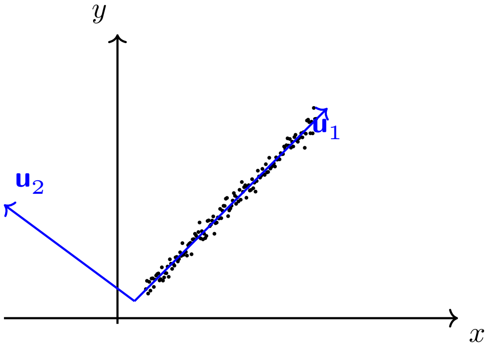
$$\|J_{\mathbf{x}}(\mathbf{h})\|_F^2 = \sum_{j=1}^{n}\sum_{l=1}^{k}\left(\frac{\partial h_l}{\partial x_j}\right)^2$$

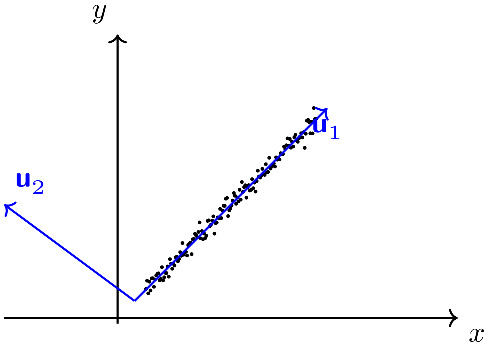


- 因此，通过将这两个目标对立，我们确保 **h** 仅对非常重要的变化敏感。
- $\mathcal{L}(\theta)$ - 捕获数据中重要的变化
- $\Omega(\theta)$ - 不捕获数据中的变化
- 结合上述两个约束，最终**仅仅**捕获数据中**非常重要**的变化

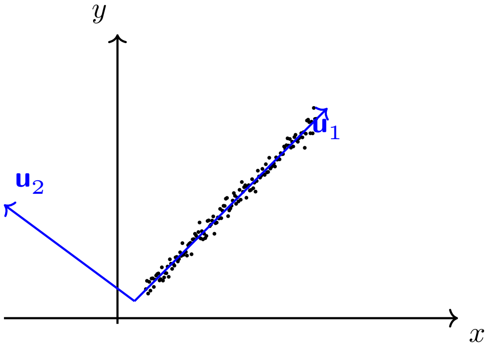
$$\|J_{\mathbf{x}}(\mathbf{h})\|_F^2 = \sum_{j=1}^{n}\sum_{l=1}^{k}\left(\frac{\partial h_l}{\partial x_j}\right)^2$$



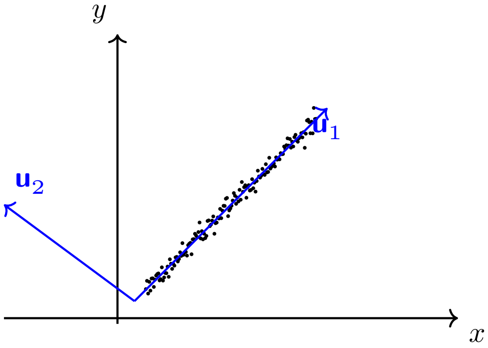

y
x
$\mathbf{u}_1$
$\mathbf{u}_2$



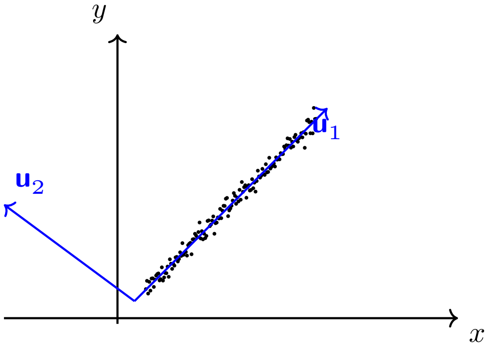
- 假设数据的变化 $\mathbf{u}_1$ 和 $\mathbf{u}_2$ 是沿着图中的方向



- 假设数据的变化 $\mathbf{u}_1$ 和 $\mathbf{u}_2$ 是沿着图中的方向
- 最大化神经元对于 $\mathbf{u}_1$ 变化的敏感度

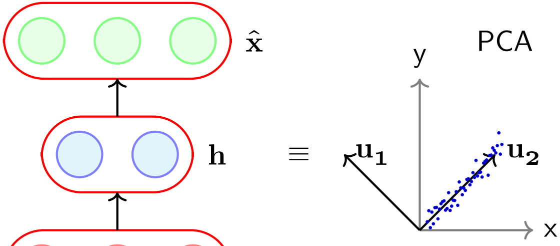


- 假设数据的变化 $\mathbf{u}_1$ 和 $\mathbf{u}_2$ 是沿着图中的方向
- 最大化神经元对于 $\mathbf{u}_1$ 变化的敏感度
- 抑制神经元对于 $\mathbf{u}_2$ 变化的敏感程度（例如图像重建中微弱的噪声扰动）

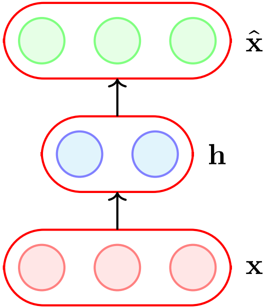


- 假设数据的变化 $\mathbf{u}_1$ 和 $\mathbf{u}_2$ 是沿着图中的方向
- 最大化神经元对于 $\mathbf{u}_1$ 变化的敏感度
- 抑制神经元对于 $\mathbf{u}_2$ 变化的敏感程度（例如图像重建中微弱的噪声扰动）
- 将二者相结合，可以在良好重建和低灵敏度这两个相互矛盾的目标之间取得平衡

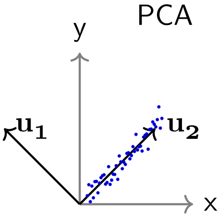
总结



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

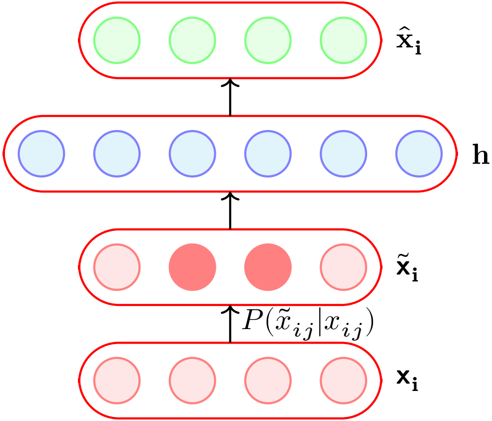


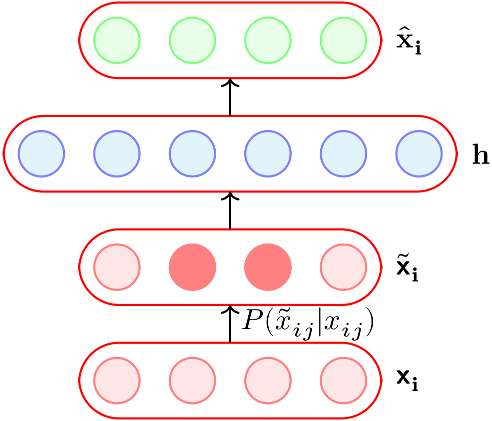
$\equiv$



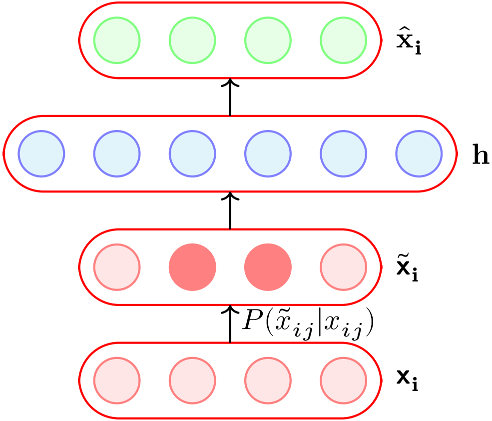
$$P^T X^T X P = D$$

$$\min_{\theta} \| X - \underbrace{HW^*}_{\substack{U\Sigma V^T \\ \text{(SVD)}}} \|_F^2$$


$\hat{\mathbf{x}}_\mathbf{i}$
$\mathbf{h}$
$\tilde{\mathbf{x}}_\mathbf{i}$
$P(\tilde{x}_{ij}|x_{ij})$
$\mathbf{x}_\mathbf{i}$

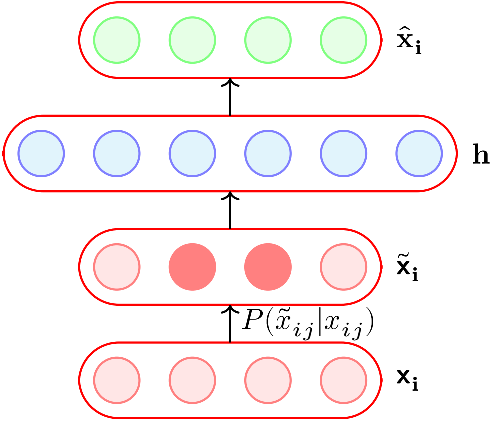

$\hat{\mathbf{x}}_\mathbf{i}$
$\mathbf{h}$
$\tilde{\mathbf{x}}_\mathbf{i}$
$P(\tilde{x}_{ij}|x_{ij})$
$\mathbf{x}_\mathbf{i}$

<u>正则化</u>



正则化

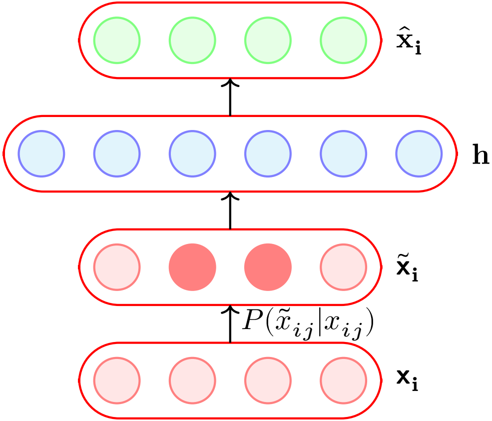
$\Omega(\theta) = \lambda\|\theta\|^2$ 权重衰减



正则化

$$\Omega(\theta) = \lambda\|\theta\|^2 \quad \boxed{\text{权重衰减}}$$

$$\Omega(\theta) = \sum_{l=1}^{k} \rho \, log \frac{\rho}{\hat{\rho}_l} + (1-\rho) \, log \frac{1-\rho}{1-\hat{\rho}_l} \quad \boxed{\text{稀疏}}$$



正则化

$$\Omega(\theta) = \lambda\|\theta\|^2$$ 权重衰减

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

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