

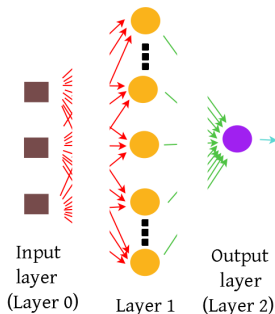
# Deep Learning - Theory and Practice

*IE 643*  
*Lectures 11, 12*

September 11 & 13, 2024.

- 1 Popular examples of MLP
  - MLP with Single Hidden Layer
  - Encoder-Decoder Architecture

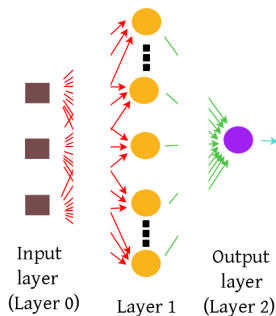
# MLP with Single Hidden Layer: Approximation Properties



In the MLP with single hidden layer, consider:

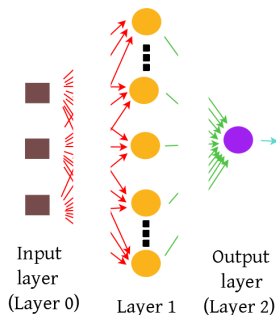
- $d$  neurons in input layer accepting inputs from  $[0, 1]^d$ .
- $N$  neurons in hidden layer.
- Single neuron in the output layer.

# MLP with Single Hidden Layer: Approximation Properties



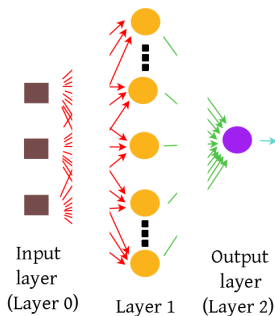
- All hidden layer neurons have sigmoidal activations.

# MLP with Single Hidden Layer: Approximation Properties



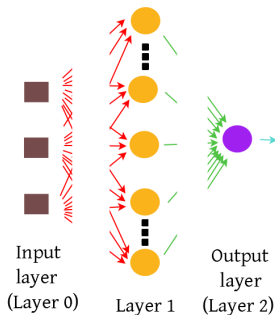
- All hidden layer neurons have sigmoidal activations.
  - ▶ **Recall:** A sigmoidal activation  $\sigma : \mathbb{R} \rightarrow (0, 1)$  is a monotonically increasing continuous function with the property  $\sigma(z) \rightarrow 0$  as  $z \rightarrow -\infty$  and  $\sigma(z) \rightarrow 1$  as  $z \rightarrow +\infty$ .

# MLP with Single Hidden Layer: Approximation Properties



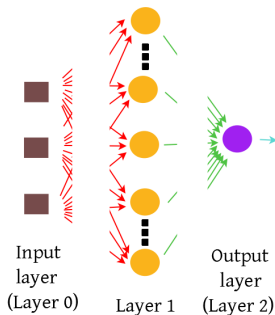
- All hidden layer neurons have sigmoidal activations.
  - ▶ **Recall:** A sigmoidal activation  $\sigma : \mathbb{R} \rightarrow (0, 1)$  is a monotonically increasing continuous function with the property  $\sigma(z) \rightarrow 0$  as  $z \rightarrow -\infty$  and  $\sigma(z) \rightarrow 1$  as  $z \rightarrow +\infty$ .
- The output layer neuron has linear activation function.

# MLP with Single Hidden Layer: Approximation Properties



- All hidden layer neurons have sigmoidal activations.
  - ▶ **Recall:** A sigmoidal activation  $\sigma : \mathbb{R} \rightarrow (0, 1)$  is a monotonically increasing continuous function with the property  $\sigma(z) \rightarrow 0$  as  $z \rightarrow -\infty$  and  $\sigma(z) \rightarrow 1$  as  $z \rightarrow +\infty$ .
- The output layer neuron has linear activation function.
  - ▶ **Recall:** A linear activation function  $\phi : \mathbb{R} \rightarrow \mathbb{R}$  is given by  $\phi(z) = z$ .

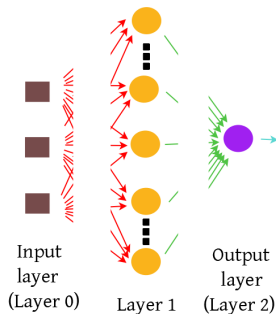
# MLP with Single Hidden Layer: Approximation Properties



- Weights connecting the input layer neurons to the  $j$ -th neuron in the hidden layer are collected into the vector  $w_j$  and the associated bias be  $b_j$ .
- Weight connecting the  $j$ -th neuron in hidden layer to the output layer neuron is denoted by  $\alpha_j$ .



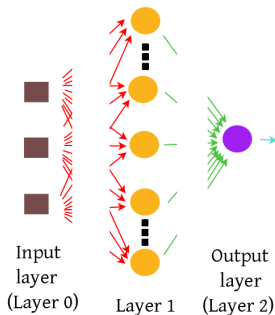
# MLP with Single Hidden Layer: Approximation Properties



- Then for an input  $x \in [0, 1]^d$  the prediction or last layer output can be represented as:

$$\hat{y} = G(x) = \sum_{j=1}^N \alpha_j \sigma(w_j^\top x + b_j).$$

# MLP with Single Hidden Layer: Approximation Properties

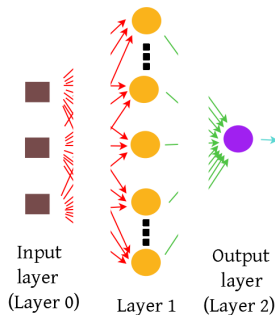


From a famous result of [Cybenko \(1989\)](#)<sup>†</sup> we have:

Let  $\mathcal{C}([0, 1]^d)$  denote the set of continuous functions over  $[0, 1]^d$  and let  $\epsilon > 0$ . Then for any  $f \in \mathcal{C}([0, 1]^d)$ , there is a sum of the form  $G(x) = \sum_{j=1}^N \alpha_j \sigma(w_j^\top x + b_j)$  such that  $|G(x) - f(x)| < \epsilon, \forall x \in [0, 1]^d$ .

<sup>†</sup> G. Cybenko, Approximations by Superpositions of a Sigmoidal Function, Math. Control Signals Systems, 1989.

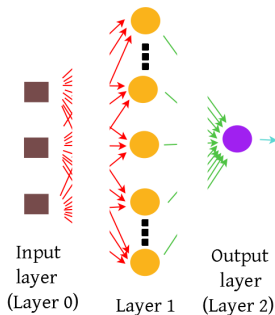
# MLP with Single Hidden Layer: Approximation Properties



The result of [Cybenko \(1989\)](#) implies:

- Single hidden layer networks where hidden layer neurons have sigmoidal activation functions and an output neuron with linear activation can approximate any continuous function over  $[0, 1]^d$  to any arbitrary precision  $\epsilon > 0$ .

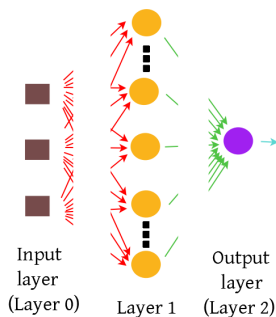
# MLP with Single Hidden Layer: Approximation Properties



The result of [Cybenko \(1989\)](#) implies:

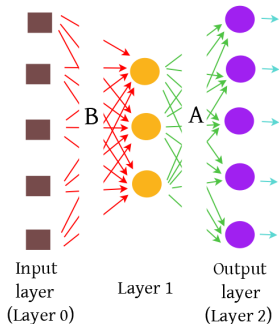
- Single hidden layer networks where hidden layer neurons has sigmoidal activation functions and an output neuron with linear activation can approximate any continuous function over  $[0, 1]^d$  to any arbitrarily precision  $\epsilon > 0$ .
- This result is about the **approximation capability** of single hidden layer networks.

# MLP with Single Hidden Layer: Approximation Properties



- **Caveat:** How many neurons  $N$  do we require in the hidden layer to achieve the approximation?

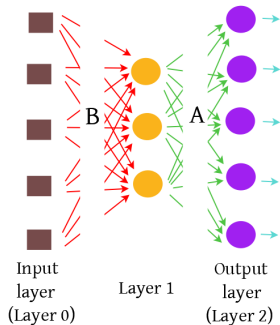
# Multi Layer Perceptron: Encoder-Decoder Architecture



We consider a simple MLP architecture with:

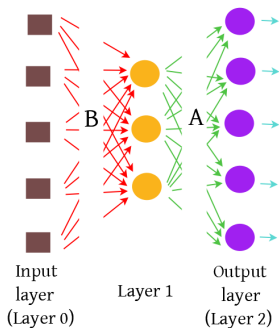
- An input layer with  $n$  neurons
- A hidden layer with  $p$  neurons, where  $p \leq n$
- An output layer with  $n$  neurons
- All neurons in hidden and output layers have linear activations.

# Multi Layer Perceptron: Encoder-Decoder Architecture



This architecture is popularly called **Encoder-Decoder Architecture**.

# Multi Layer Perceptron: Encoder-Decoder Architecture

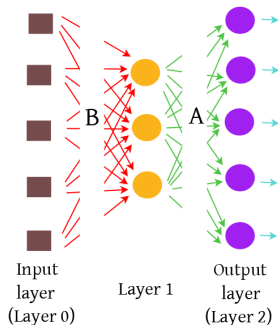


Let  $B$  denote the  $p \times n$  matrix connecting input layer and hidden layer.

Let  $A$  denote the  $n \times p$  matrix connecting hidden layer and output layer.

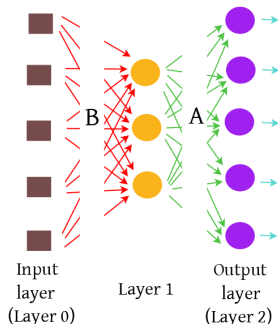


# Multi Layer Perceptron: Encoder-Decoder Architecture



- **Input:** Training Data  $D = \{(x^i, y^i)\}_{i=1}^S$ ,  $x^i \in \mathbb{R}^n$ ,  $y^i \in \mathbb{R}^n$ ,  $\forall i \in \{1, \dots, S\}$ .
- **MLP Parameters:** Weight matrices  $B$  and  $A$ .

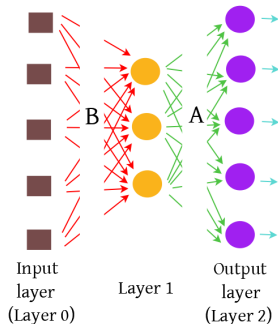
# Multi Layer Perceptron: Encoder-Decoder Architecture



- **Input:** Training Data  $D = \{(x^i, y^i)\}_{i=1}^S$ ,  $x^i \in \mathbb{R}^n$ ,  $y^i \in \mathbb{R}^n$ ,  $\forall i \in \{1, \dots, S\}$ .
- **MLP Parameters:** Weight matrices  $B$  and  $A$ .

$$\text{Error: } E(B, A) = \sum_{i=1}^S \|ABx^i - y^i\|_2^2.$$

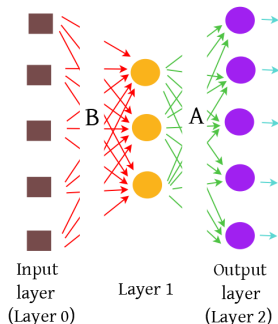
# Multi Layer Perceptron: Encoder-Decoder Architecture



- **Input:** Training Data  $D = \{(x^i, y^i)\}_{i=1}^S$ ,  $x^i \in \mathbb{R}^n$ ,  $y^i \in \mathbb{R}^n$ ,  $\forall i \in \{1, \dots, S\}$ .
- **MLP Parameters:** Weight matrices  $B$  and  $A$ .

**Special case:** When  $y^i = x^i$ ,  $\forall i$ , the architecture is called **Autoencoder**.

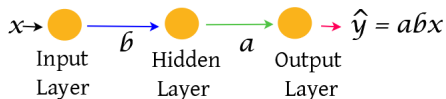
# Multi Layer Perceptron: Encoder-Decoder Architecture



- **Input:** Training Data  $D = \{(x^i, y^i)\}_{i=1}^S$ ,  $x^i \in \mathbb{R}^n$ ,  $y^i \in \mathbb{R}^n$ ,  $\forall i \in \{1, \dots, S\}$ .
- **MLP Parameters:** Weight matrices  $B$  and  $A$ .

**Error for autoencoder:** 
$$E(B, A) = \sum_{i=1}^S \|ABx^i - x^i\|_2^2.$$

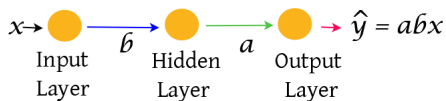
# Multi Layer Perceptron: Encoder-Decoder Architecture



**Example:** Consider the case:  $n = p = 1$ .

**Error:** 
$$E(b, a) = \sum_{i=1}^S (abx^i - y^i)^2.$$

# Multi Layer Perceptron: Encoder-Decoder Architecture

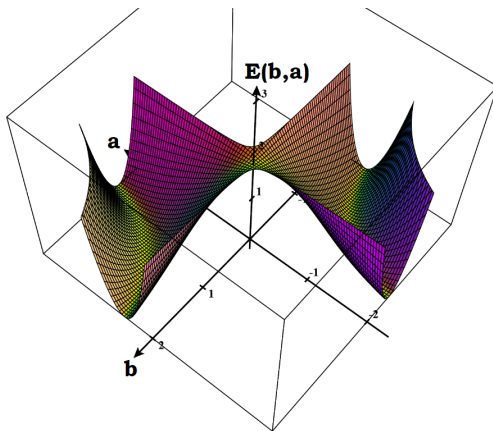


## Sample Training Data:

$x^i$	$y^i$
.708333	.708333
.583333	.583333
.166667	.166667
.458333	.458333
.875	.875

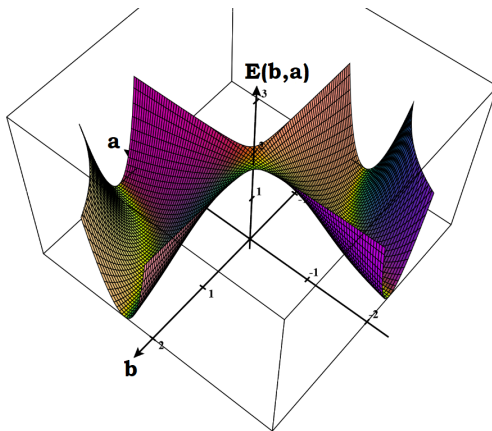
# Multi Layer Perceptron: Encoder-Decoder Architecture

Loss surface for the sample training data:



# Multi Layer Perceptron: Encoder-Decoder Architecture

Loss surface for the sample training data:

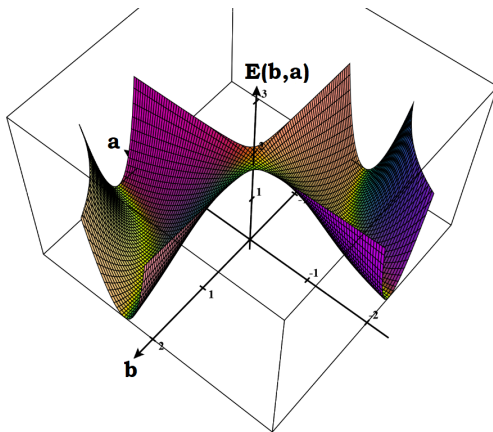


**Observation:** Though there are two valleys in the loss surface, they look symmetric.



# Multi Layer Perceptron: Encoder-Decoder Architecture

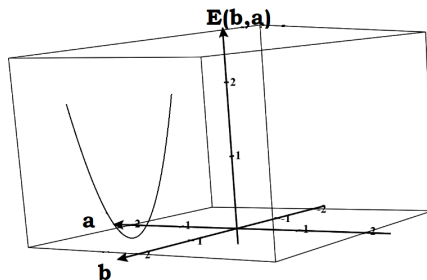
**Loss surface for the sample training data:**



**Question:** Does this extend to cases where  $y^i \neq x^i$  and for  $n \geq 2$ ,  $n \geq p$ ?

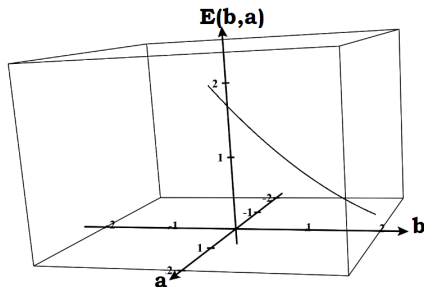
# Multi Layer Perceptron: Encoder-Decoder Architecture

However, when we fix  $a$ , we have:



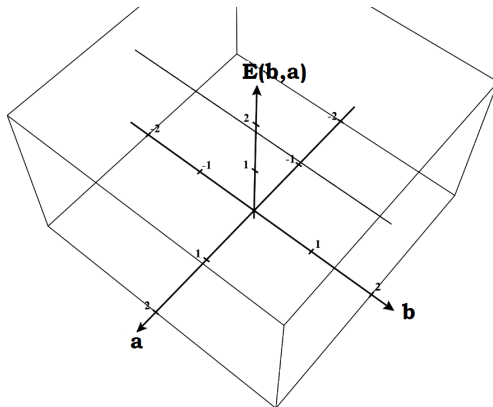
# Multi Layer Perceptron: Encoder-Decoder Architecture

However, when we fix  $a$ , we have:



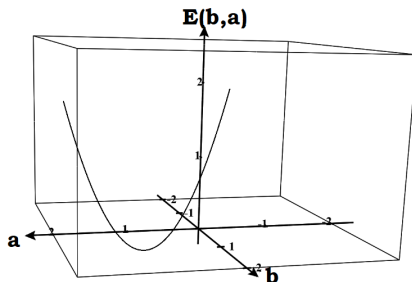
# Multi Layer Perceptron: Encoder-Decoder Architecture

However, when we fix  $a$ , we have:



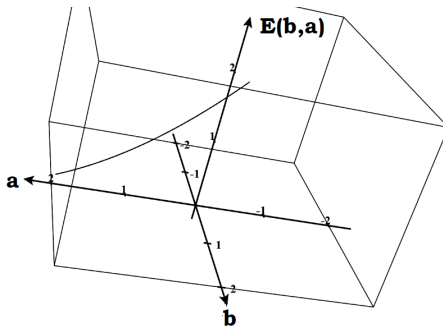
# Multi Layer Perceptron: Encoder-Decoder Architecture

Similarly, when we fix  $b$ , we have:



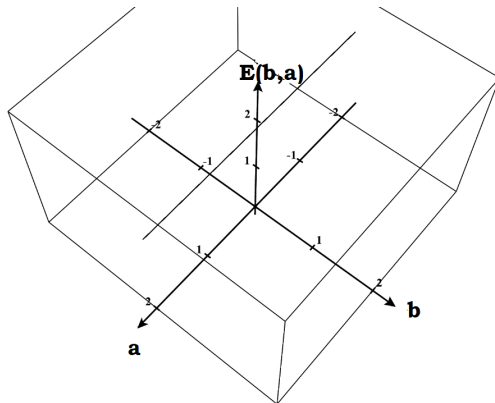
# Multi Layer Perceptron: Encoder-Decoder Architecture

Similarly, when we fix  $b$ , we have:

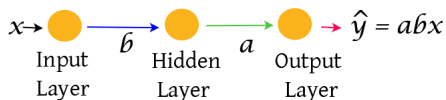


# Multi Layer Perceptron: Encoder-Decoder Architecture

Similarly, when we fix  $b$ , we have:



# Multi Layer Perceptron: Encoder-Decoder Architecture

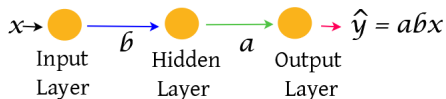


**Error:**  $E(b, a) = \sum_{i=1}^S (abx^i - y^i)^2.$

When we fix  $a$  or when we fix  $b$ , we observe that the graph does not contain multiple valleys.



# Multi Layer Perceptron: Encoder-Decoder Architecture

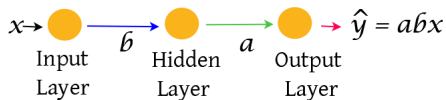


**Error:**  $E(b, a) = \sum_{i=1}^S (abx^i - y^i)^2.$

When we fix  $a$  or when we fix  $b$ , we observe that the graph does not contain multiple valleys.

Thus when we fix  $a$  or when we fix  $b$ , we observe that the graph looks as if every local optimum is a global optimum.

# Multi Layer Perceptron: Encoder-Decoder Architecture



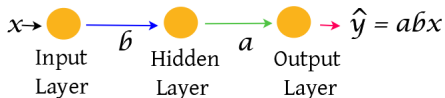
**Error:**  $E(b, a) = \sum_{i=1}^S (abx^i - y^i)^2.$

When we fix  $a$  or when we fix  $b$ , we observe that the graph does not contain multiple valleys.

Thus when we fix  $a$  or when we fix  $b$ , we observe that the graph looks as if every local optimum is a global optimum.

**Question:** Does this behavior extend to higher dimensions?

# Multi Layer Perceptron: Encoder-Decoder Architecture



**Error:**  $E(b, a) = \sum_{i=1}^S (abx^i - y^i)^2.$

When we fix  $a$  or when we fix  $b$ , we observe that the graph does not contain multiple valleys.

Thus when we fix  $a$  or when we fix  $b$ , we observe that the graph looks as if every local optimum is a global optimum.

**Question:** Does this behavior extend to higher dimensions?

We shall investigate this in higher dimensions !

# Multi Layer Perceptron: Encoder-Decoder Architecture

## Some notations:

$$\text{Let } X = \begin{bmatrix} \uparrow & \uparrow & \dots & \uparrow \\ x^1 & x^2 & \dots & x^S \\ \downarrow & \downarrow & \dots & \downarrow \end{bmatrix}, Y = \begin{bmatrix} \uparrow & \uparrow & \dots & \uparrow \\ y^1 & y^2 & \dots & y^S \\ \downarrow & \downarrow & \dots & \downarrow \end{bmatrix}.$$

**Note:**  $X$  and  $Y$  are  $n \times S$  matrices.

# Multi Layer Perceptron: Encoder-Decoder Architecture

## Some notations:

For a  $n \times q$  matrix  $H = \begin{bmatrix} h_{11} & h_{12} & \dots & h_{1q} \\ \vdots & \vdots & \dots & \vdots \\ h_{n1} & h_{n2} & \dots & h_{nq} \end{bmatrix}$ ,

let  $\text{vec}(H) = [h_{11} \dots h_{n1} \ h_{12} \dots h_{n2} \dots h_{1q} \dots h_{nq}]^T$

denote a  $nq \times 1$  vector.

**Note:**  $\text{vec}(H)$  contains the columns of matrix  $H$  stacked upon one another.

# Multi Layer Perceptron: Encoder-Decoder Architecture

## Recall:

- **Input:** Training Data  $D = \{(x^i, y^i)\}_{i=1}^S$ ,  $x^i \in \mathbb{R}^n$ ,  $y^i \in \mathbb{R}^n$ ,  $\forall i \in \{1, \dots, S\}$ .
- **MLP Parameters:** Weight matrices  $B$  and  $A$ .

**Error:**  $E = \sum_{i=1}^S \|ABx^i - y^i\|_2^2$ . (We have used  $E$  since the context is clear!)

# Multi Layer Perceptron: Encoder-Decoder Architecture

## Recall:

- **Input:** Training Data  $D = \{(x^i, y^i)\}_{i=1}^S$ ,  $x^i \in \mathbb{R}^n$ ,  $y^i \in \mathbb{R}^n$ ,  $\forall i \in \{1, \dots, S\}$ .
- **MLP Parameters:** Weight matrices  $B$  and  $A$ .

**Error:**  $E = \sum_{i=1}^S \|ABx^i - y^i\|_2^2$ . (We have used  $E$  since the context is clear!)

- Using the new notations, we write:  $E = \|\text{vec}(ABX - Y)\|_2^2$ .  
(Homework!)

**Note:** The norm is a simple vector norm.

# Multi Layer Perceptron: Encoder-Decoder Architecture

We have:  $E = \|\text{vec}(ABX - Y)\|_2^2$ .

Now note:  $\text{vec}(ABX - Y) = \text{vec}(ABX) - \text{vec}(Y)$ . (Homework: Verify this claim!)



# Multi Layer Perceptron: Encoder-Decoder Architecture

We have:  $E = \|\text{vec}(ABX - Y)\|_2^2$ .

Now note:  $\text{vec}(ABX - Y) = \text{vec}(ABX) - \text{vec}(Y)$ . (Homework: Verify this claim!)

Thus we have:  $E = \|\text{vec}(ABX) - \text{vec}(Y)\|_2^2$ .

# Multi Layer Perceptron: Encoder-Decoder Architecture

## Some more new notations:

For a  $m \times n$  matrix  $G = \begin{bmatrix} g_{11} & \dots & g_{1n} \\ \vdots & \dots & \vdots \\ g_{m1} & \dots & g_{mn} \end{bmatrix}$

and a  $p \times q$  matrix  $H = \begin{bmatrix} h_{11} & \dots & h_{1q} \\ \vdots & \dots & \vdots \\ h_{p1} & \dots & h_{pq} \end{bmatrix}$ ,

define:  $G \otimes H = \begin{bmatrix} g_{11}H & \dots & g_{1n}H \\ \vdots & \dots & \vdots \\ g_{m1}H & \dots & g_{mn}H \end{bmatrix}$  as Kronecker product of  $G$  and  $H$ .

**Note:**  $G \otimes H$  is of size  $mp \times nq$ .

# Multi Layer Perceptron: Encoder-Decoder Architecture

**Claim:**

$$\text{vec}(ABX) = (X^{\top} \otimes A)\text{vec}(B).$$

**Proof idea:**

# Multi Layer Perceptron: Encoder-Decoder Architecture

**Using:**

$$\text{vec}(ABX) = (X^{\top} \otimes A)\text{vec}(B).$$

we can write:

$$\begin{aligned} E &= \|\text{vec}(ABX - Y)\|_2^2 = \|\text{vec}(ABX) - \text{vec}(Y)\|_2^2 \\ &= \|(X^{\top} \otimes A)\text{vec}(B) - \text{vec}(Y)\|_2^2. \end{aligned}$$

# Multi Layer Perceptron: Encoder-Decoder Architecture

**Using:**

$$\text{vec}(ABX) = (X^\top \otimes A)\text{vec}(B).$$

we can write:

$$\begin{aligned} E &= \|\text{vec}(ABX - Y)\|_2^2 = \|\text{vec}(ABX) - \text{vec}(Y)\|_2^2 \\ &= \|(X^\top \otimes A)\text{vec}(B) - \text{vec}(Y)\|_2^2. \end{aligned}$$

This is of the form:  $F(z) = \|Mz - c\|_2^2$ , where  $M = (X^\top \otimes A)$ ,  $z = \text{vec}(B)$  and  $c = \text{vec}(Y)$ .

# Multi Layer Perceptron: Encoder-Decoder Architecture

**Using:**

$$\text{vec}(ABX) = (X^\top \otimes A)\text{vec}(B).$$

we can write:

$$\begin{aligned} E &= \|\text{vec}(ABX - Y)\|_2^2 = \|\text{vec}(ABX) - \text{vec}(Y)\|_2^2 \\ &= \|(X^\top \otimes A)\text{vec}(B) - \text{vec}(Y)\|_2^2. \end{aligned}$$

This is of the form:  $F(z) = \|Mz - c\|_2^2$ , where  $M = (X^\top \otimes A)$ ,  $z = \text{vec}(B)$  and  $c = \text{vec}(Y)$ .

**Observe:**  $M$  is of size  $nS \times np$ ,  $z$  is a  $np \times 1$  vector and  $c$  is a  $nS \times 1$  vector.

# Multi Layer Perceptron: Encoder-Decoder Architecture

Consider the function  $F(z) = \|Mz - c\|_2^2$ . We have the following result:

Convexity of function  $F$

The function  $F(z)$  is convex.

# Multi Layer Perceptron: Encoder-Decoder Architecture

Consider the function  $F(z) = \|Mz - c\|_2^2$ . We have the following result:

Convexity of function  $F$

The function  $F(z)$  is convex.

**Question:** What is a convex function?



# Multi Layer Perceptron: Encoder-Decoder Architecture

Consider the problem  $F(z) = \|Mz - c\|_2^2$ . We have the following result:

## Convexity of function $F$

The function  $F(z)$  is convex.

**Question:** What is a convex function?

**We discuss convex sets and convex functions in Part 2 of this lecture !**

# Multi Layer Perceptron: Encoder-Decoder Architecture

Consider the function  $F(z) = \|Mz - c\|_2^2$ . We have the following result:

## Convexity of function $F$

The function  $F(z)$  is convex.

Hence  $\min_z F(z)$  is a **convex optimization problem**.

# Multi Layer Perceptron: Encoder-Decoder Architecture

Consider the function  $F(z) = \|Mz - c\|_2^2$ . We have the following result:

## Convexity of function $F$

The function  $F(z)$  is convex.

Hence  $\min_z F(z)$  is a **convex optimization problem**.

## Note:

$$\operatorname{argmin}_z F(z) = \operatorname{argmin}_z \|Mz - c\|_2^2$$

# Multi Layer Perceptron: Encoder-Decoder Architecture

Consider the function  $F(z) = \|Mz - c\|_2^2$ . We have the following result:

## Convexity of function $F$

The function  $F(z)$  is convex.

Hence  $\min_z F(z)$  is a **convex optimization problem**.

## Note:

$$\operatorname{argmin}_z F(z) = \operatorname{argmin}_z \|Mz - c\|_2^2 = \operatorname{argmin}_z (Mz - c)^\top (Mz - c)$$

# Multi Layer Perceptron: Encoder-Decoder Architecture

Consider the function  $F(z) = \|Mz - c\|_2^2$ . We have the following result:

## Convexity of function $F$

The function  $F(z)$  is convex.

Hence  $\min_z F(z)$  is a **convex optimization problem**.

## Note:

$$\begin{aligned}\operatorname{argmin}_z F(z) &= \operatorname{argmin}_z \|Mz - c\|_2^2 = \operatorname{argmin}_z (Mz - c)^\top (Mz - c) \\ &= \operatorname{argmin}_z z^\top M^\top Mz - 2z^\top M^\top c + c^\top c\end{aligned}$$

# Multi Layer Perceptron: Encoder-Decoder Architecture

Consider the function  $F(z) = \|Mz - c\|_2^2$ . We have the following result:

## Convexity of function $F$

The function  $F(z)$  is convex.

Hence  $\min_z F(z)$  is a **convex optimization problem**.

## Note:

$$\begin{aligned}\operatorname{argmin}_z F(z) &= \min_z \|Mz - c\|_2^2 = \operatorname{argmin}_z (Mz - c)^\top (Mz - c) \\ &= \operatorname{argmin}_z z^\top M^\top Mz - 2z^\top M^\top c + c^\top c \\ &= \operatorname{argmin}_z z^\top M^\top Mz - 2z^\top M^\top c\end{aligned}$$

# Multi Layer Perceptron: Encoder-Decoder Architecture

Consider the function  $F(z) = \|Mz - c\|_2^2$ . We have the following result:

## Convexity of function $F$

The function  $F(z)$  is convex.

Hence  $\min_z F(z)$  is a **convex optimization problem**.

**Note:**

$$\operatorname{argmin}_z F(z) = \min_z \|Mz - c\|_2^2 = \operatorname{argmin}_z z^\top M^\top Mz - 2z^\top M^\top c$$

- By first-order optimality characterization, we have  $z^*$  is a solution of  $\min_z F(z)$  if and only if  $\nabla F(z^*) = 0$ .

# Multi Layer Perceptron: Encoder-Decoder Architecture

Consider the function  $F(z) = \|Mz - c\|_2^2$ . We have the following result:

## Convexity of function $F$

The function  $F(z)$  is convex.

Hence  $\min_z F(z)$  is a **convex optimization problem**.

**Note:**

$$\operatorname{argmin}_z F(z) = \min_z \|Mz - c\|_2^2 = \operatorname{argmin}_z z^\top M^\top Mz - 2z^\top M^\top c$$

- By first-order optimality characterization, we have  $z^*$  is a solution of  $\min_z F(z)$  if and only if  $\nabla F(z^*) = 0$ .
- $\implies 2M^\top Mz^* - 2M^\top c = 0$ .



# Multi Layer Perceptron: Encoder-Decoder Architecture

Consider the function  $F(z) = \|Mz - c\|_2^2$ . We have the following result:

## Convexity of function $F$

The function  $F(z)$  is convex.

Hence  $\min_z F(z)$  is a **convex optimization problem**.

**Note:**

$$\operatorname{argmin}_z F(z) = \min_z \|Mz - c\|_2^2 = \operatorname{argmin}_z z^\top M^\top Mz - 2z^\top M^\top c$$

- By first-order optimality characterization, we have  $z^*$  is a solution of  $\min_z F(z)$  if and only if  $\nabla F(z^*) = 0$ .
- $\implies 2M^\top Mz^* - 2M^\top c = 0$ .
- $\implies M^\top Mz^* = M^\top c$ .

# Multi Layer Perceptron: Encoder-Decoder Architecture

Consider the function  $F(z) = \|Mz - c\|_2^2$ . We have the following result:

## Convexity of function $F$

The function  $F(z)$  is convex.

Hence  $\min_z F(z)$  is a **convex optimization problem**.

**Note:**

$$\operatorname{argmin}_z F(z) = \min_z \|Mz - c\|_2^2 = \operatorname{argmin}_z z^\top M^\top Mz - 2z^\top M^\top c$$

- By first-order optimality characterization, we have  $z^*$  is a solution of  $\min_z F(z)$  if and only if  $\nabla F(z^*) = 0$ .
- $\implies 2M^\top Mz^* - 2M^\top c = 0$ .
- $\implies M^\top Mz^* = M^\top c$ .

Further if  $M^\top M$  is positive definite,  $z^*$  is the unique optimal solution and is given by  $z^* = (M^\top M)^{-1} M^\top c$ .

# Multi Layer Perceptron: Encoder-Decoder Architecture

## Recall:

$$\min_z F(z) = \|Mz - c\|_2^2 \text{ is same as } \min_{\text{vec}(B)} \|(X^\top \otimes A)\text{vec}(B) - \text{vec}(Y)\|_2^2.$$

Hence from  $M^\top Mz^* = M^\top c$ , we have:

$$(X^\top \otimes A)^\top (X^\top \otimes A)z^* = (X^\top \otimes A)^\top c.$$

# Multi Layer Perceptron: Encoder-Decoder Architecture

## Some properties of Kronecker Product:

(P1)  $\text{vec}(ABC) = (C^T \otimes A)\text{vec}(B).$

(P2)  $(G \otimes H)^T = (G^T \otimes H^T).$

(P3)  $(G \otimes H)^{-1} = (G^{-1} \otimes H^{-1}).$

(P4)  $(G \otimes H)(U \otimes V) = (GU \otimes HV)$

(P5) If  $G$  and  $H$  are symmetric and positive (semi-)definite, then  $(G \otimes H)$  is also symmetric and positive (semi-)definite.

# Multi Layer Perceptron: Encoder-Decoder Architecture

## Recall:

$$\min_z F(z) = \|Mz - c\|_2^2 \text{ is same as } \min_{\text{vec}(B)} \|(X^\top \otimes A)\text{vec}(B) - \text{vec}(Y)\|_2^2.$$

Hence from  $M^\top Mz^* = M^\top c$ , we have:

$$\begin{aligned} (X^\top \otimes A)^\top (X^\top \otimes A)z^* &= (X^\top \otimes A)^\top c \\ \implies (X \otimes A^\top)(X^\top \otimes A)z^* &= (X \otimes A^\top)c \text{ (from (P2))} \\ \implies (XX^\top \otimes A^\top A)z^* &= (X \otimes A^\top)c \text{ (from (P4))} \end{aligned}$$

# Multi Layer Perceptron: Encoder-Decoder Architecture

**Recall:**  $X = \begin{bmatrix} \uparrow & \uparrow & \dots & \uparrow \\ x^1 & x^2 & \dots & x^S \\ \downarrow & \downarrow & \dots & \downarrow \end{bmatrix}, Y = \begin{bmatrix} \uparrow & \uparrow & \dots & \uparrow \\ y^1 & y^2 & \dots & y^S \\ \downarrow & \downarrow & \dots & \downarrow \end{bmatrix}.$

**Also note:**

$$XX^T = \sum_{i=1}^S x^i (x^i)^T,$$

$$YY^T = \sum_{i=1}^S y^i (y^i)^T,$$

$$XY^T = \sum_{i=1}^S x^i (y^i)^T$$

$$\text{and } YX^T = \sum_{i=1}^S y^i (x^i)^T.$$

(Homework: try to see if these equalities are true!)

Denote  $XX^T$  by  $\Sigma_{XX}$ ,  $YY^T$  by  $\Sigma_{YY}$ ,  $XY^T$  by  $\Sigma_{XY}$  and  $YX^T$  by  $\Sigma_{YX}$ .

# Multi Layer Perceptron: Encoder-Decoder Architecture

**Recall:**  $X = \begin{bmatrix} \uparrow & \uparrow & \dots & \uparrow \\ x^1 & x^2 & \dots & x^S \\ \downarrow & \downarrow & \dots & \downarrow \end{bmatrix}, Y = \begin{bmatrix} \uparrow & \uparrow & \dots & \uparrow \\ y^1 & y^2 & \dots & y^S \\ \downarrow & \downarrow & \dots & \downarrow \end{bmatrix}.$

**Also note:**

$$XX^T = \sum_{i=1}^S x^i (x^i)^T,$$

$$YY^T = \sum_{i=1}^S y^i (y^i)^T,$$

$$XY^T = \sum_{i=1}^S x^i (y^i)^T$$

$$\text{and } YX^T = \sum_{i=1}^S y^i (x^i)^T.$$

(Homework: try to see if these equalities are true!)

Denote  $XX^T$  by  $\Sigma_{XX}$ ,  $YY^T$  by  $\Sigma_{YY}$ ,  $XY^T$  by  $\Sigma_{XY}$  and  $YX^T$  by  $\Sigma_{YX}$ .

Also note  $\Sigma_{XX}, \Sigma_{YY}$  are symmetric and  $(\Sigma_{XY})^T = (XY^T)^T = YX^T = \Sigma_{YX}$ .

# Multi Layer Perceptron: Encoder-Decoder Architecture

Recall:

$$\min_z F(z) = \|Mz - c\|_2^2 \text{ is same as } \min_{\text{vec}(B)} \|(X^\top \otimes A)\text{vec}(B) - \text{vec}(Y)\|_2^2.$$

Hence from  $M^\top Mz^* = M^\top c$ , we have:

$$\begin{aligned} (X^\top \otimes A)^\top (X^\top \otimes A)z^* &= (X^\top \otimes A)^\top c \\ \implies (X \otimes A^\top)(X^\top \otimes A)z^* &= (X \otimes A^\top)c \text{ (from (P2))} \\ \implies (XX^\top \otimes A^\top A)z^* &= (X \otimes A^\top)c \text{ (from (P4))} \end{aligned}$$



# Multi Layer Perceptron: Encoder-Decoder Architecture

Recall:

$$\min_z F(z) = \|Mz - c\|_2^2 \text{ is same as } \min_{\text{vec}(B)} \|(X^\top \otimes A)\text{vec}(B) - \text{vec}(Y)\|_2^2.$$

Hence from  $M^\top Mz^* = M^\top c$ , we have:

$$\begin{aligned} (X^\top \otimes A)^\top (X^\top \otimes A)z^* &= (X^\top \otimes A)^\top c \\ \implies (X \otimes A^\top)(X^\top \otimes A)z^* &= (X \otimes A^\top)c \text{ (from (P2))} \\ \implies (XX^\top \otimes A^\top A)z^* &= (X \otimes A^\top)c \text{ (from (P4))} \\ \implies (\Sigma_{XX} \otimes A^\top A)z^* &= (X \otimes A^\top)c \end{aligned}$$

# Multi Layer Perceptron: Encoder-Decoder Architecture

Recall:

$$\min_z F(z) = \|Mz - c\|_2^2 \text{ is same as } \min_{\text{vec}(B)} \|(X^\top \otimes A)\text{vec}(B) - \text{vec}(Y)\|_2^2.$$

Hence from  $M^\top Mz^* = M^\top c$ , we have:

$$\begin{aligned} (X^\top \otimes A)^\top (X^\top \otimes A)z^* &= (X^\top \otimes A)^\top c \\ \implies (X \otimes A^\top)(X^\top \otimes A)z^* &= (X \otimes A^\top)c \text{ (from (P2))} \\ \implies (XX^\top \otimes A^\top A)z^* &= (X \otimes A^\top)c \text{ (from (P4))} \\ \implies (\Sigma_{XX} \otimes A^\top A)z^* &= (X \otimes A^\top)c \\ \implies (\Sigma_{XX} \otimes A^\top A)(\text{vec}(B^*)) &= (X \otimes A^\top)(\text{vec}(Y)) \end{aligned}$$

# Multi Layer Perceptron: Encoder-Decoder Architecture

Recall:

$$\min_z F(z) = \|Mz - c\|_2^2 \text{ is same as } \min_{\text{vec}(B)} \|(X^\top \otimes A)\text{vec}(B) - \text{vec}(Y)\|_2^2.$$

Hence from  $M^\top Mz^* = M^\top c$ , we have:

$$\begin{aligned} (X^\top \otimes A)^\top (X^\top \otimes A)z^* &= (X^\top \otimes A)^\top c \\ \implies (X \otimes A^\top)(X^\top \otimes A)z^* &= (X \otimes A^\top)c \text{ (from (P2))} \\ \implies (XX^\top \otimes A^\top A)z^* &= (X \otimes A^\top)c \text{ (from (P4))} \\ \implies (\Sigma_{XX} \otimes A^\top A)z^* &= (X \otimes A^\top)c \\ \implies (\Sigma_{XX} \otimes A^\top A)(\text{vec}(B^*)) &= (X \otimes A^\top)(\text{vec}(Y)) \\ \implies A^\top AB^*\Sigma_{XX} &= A^\top YX^\top \text{ (from (P1))} \end{aligned}$$

# Multi Layer Perceptron: Encoder-Decoder Architecture

Recall:

$$\min_z F(z) = \|Mz - c\|_2^2 \text{ is same as } \min_{\text{vec}(B)} \|(X^\top \otimes A)\text{vec}(B) - \text{vec}(Y)\|_2^2.$$

Hence from  $M^\top Mz^* = M^\top c$ , we have:

$$\begin{aligned} (X^\top \otimes A)^\top (X^\top \otimes A)z^* &= (X^\top \otimes A)^\top c \\ \implies (X \otimes A^\top)(X^\top \otimes A)z^* &= (X \otimes A^\top)c \text{ (from (P2))} \\ \implies (XX^\top \otimes A^\top A)z^* &= (X \otimes A^\top)c \text{ (from (P4))} \\ \implies (\Sigma_{XX} \otimes A^\top A)z^* &= (X \otimes A^\top)c \\ \implies (\Sigma_{XX} \otimes A^\top A)(\text{vec}(B^*)) &= (X \otimes A^\top)(\text{vec}(Y)) \\ \implies A^\top AB^*\Sigma_{XX} &= A^\top YX^\top \text{ (from (P1))} \\ \implies A^\top AB^*\Sigma_{XX} &= A^\top \Sigma_{YX} \end{aligned}$$

# Multi Layer Perceptron: Encoder-Decoder Architecture

We have the following result:

The minimizer  $z^*$  of  $\min_z \|(X^\top \otimes A)z - \text{vec}(Y)\|_2^2$  satisfies

$$A^\top AB^* \Sigma_{XX} = A^\top \Sigma_{YX}$$

where  $\text{vec}(B^*) = z^*$ .

# Multi Layer Perceptron: Encoder-Decoder Architecture

## Assume:

- $\Sigma_{XX}$  is invertible.

# Multi Layer Perceptron: Encoder-Decoder Architecture

## Assume:

- $\Sigma_{XX}$  is invertible.
  - ▶ Recall:  $\Sigma_{XX} = XX^T = \sum_{j=1}^S x^j (x^j)^T$ .

# Multi Layer Perceptron: Encoder-Decoder Architecture

## Assume:

- $\Sigma_{XX}$  is invertible.
  - ▶ Recall:  $\Sigma_{XX} = XX^\top = \sum_{j=1}^S x^j (x^j)^\top$ .
  - ▶ Hence  $\Sigma_{XX}$  is symmetric and positive semi-definite.



# Multi Layer Perceptron: Encoder-Decoder Architecture

## Assume:

- $\Sigma_{XX}$  is invertible.
  - ▶ Recall:  $\Sigma_{XX} = XX^\top = \sum_{j=1}^S x^j (x^j)^\top$ .
  - ▶ Hence  $\Sigma_{XX}$  is symmetric and positive semi-definite.
  - ▶ Now invertibility of  $\Sigma_{XX}$  implies that  $\Sigma_{XX}$  is symmetric and positive definite.

# Multi Layer Perceptron: Encoder-Decoder Architecture

## Assume:

- $\Sigma_{XX}$  is invertible.
- $A$  is full rank.

# Multi Layer Perceptron: Encoder-Decoder Architecture

## Assume:

- $\Sigma_{XX}$  is invertible.
- $A$  is full rank.
  - ▶ Recall: Rank of matrix  $A$  is the number of linearly independent columns or rows in  $A$ .

# Multi Layer Perceptron: Encoder-Decoder Architecture

## Assume:

- $\Sigma_{XX}$  is invertible.
- $A$  is full rank.
  - ▶ Recall: Rank of matrix  $A$  is the number of linearly independent columns or rows in  $A$ .
  - ▶ Also recall:  $A$  is of size  $n \times p$  and  $p \leq n$ .

# Multi Layer Perceptron: Encoder-Decoder Architecture

## Assume:

- $\Sigma_{XX}$  is invertible.
- $A$  is full rank.
  - ▶ Recall: Rank of matrix  $A$  is the number of linearly independent columns or rows in  $A$ .
  - ▶ Also recall:  $A$  is of size  $n \times p$  and  $p \leq n$ .
  - ▶  $A$  is full rank  $\implies \text{rank}(A) = \min\{n, p\} = p$ .

# Multi Layer Perceptron: Encoder-Decoder Architecture

## Assume:

- $\Sigma_{XX}$  is invertible.
- $A$  is full rank.
  - ▶ Recall: Rank of matrix  $A$  is the number of linearly independent columns or rows in  $A$ .
  - ▶ Also recall:  $A$  is of size  $n \times p$  and  $p \leq n$ .
  - ▶  $A$  is full rank  $\implies \text{rank}(A) = \min\{n, p\} = p$ .

The full rank assumption on  $A \implies A^T A$  is symmetric and positive definite and hence invertible.

# Multi Layer Perceptron: Encoder-Decoder Architecture

Recall:

$$\min_z F(z) = \|Mz - c\|_2^2 \text{ is same as } \min_{\text{vec}(B)} \|(X^\top \otimes A)\text{vec}(B) - \text{vec}(Y)\|_2^2.$$

Hence from  $M^\top Mz^* = M^\top c$ , we have:

$$\begin{aligned} (X^\top \otimes A)^\top (X^\top \otimes A)z^* &= (X^\top \otimes A)^\top c \\ \Rightarrow (X \otimes A^\top)(X^\top \otimes A)z^* &= (X \otimes A^\top)c \text{ (from (P2))} \\ \Rightarrow (XX^\top \otimes A^\top A)z^* &= (X \otimes A^\top)c \text{ (from (P4))} \\ \Rightarrow (\Sigma_{XX} \otimes A^\top A)z^* &= (X \otimes A^\top)c \end{aligned}$$

If the assumptions that  $\Sigma_{XX}$  is invertible and  $A$  is full rank hold, then we have  $\Sigma_{XX}$  and  $A^\top A$  are symmetric and positive definite, hence

$M^\top M = (\Sigma_{XX} \otimes A^\top A)$  is also symmetric and positive definite (by (P5)).

# Multi Layer Perceptron: Encoder-Decoder Architecture

## Recall:

$$\min_z F(z) = \|Mz - c\|_2^2 \text{ is same as } \min_{\text{vec}(B)} \|(X^\top \otimes A)\text{vec}(B) - \text{vec}(Y)\|_2^2.$$

Since  $M^\top M$  is positive definite, the function  $\|Mz - c\|_2^2$  is strictly convex and admits a unique minimizer.



# Multi Layer Perceptron: Encoder-Decoder Architecture

Consider the result:

For a fixed  $A$ , the minimizer  $z^*$  of  $\min_z \|(X^\top \otimes A)z - \text{vec}(Y)\|_2^2$  satisfies

$$A^\top A B^* \Sigma_{XX} = A^\top \Sigma_{YX}$$

where  $\text{vec}(B^*) = z^*$ .

If the assumptions that  $\Sigma_{XX}$  is invertible and  $A$  is full rank hold, then we have

$$B^* = (A^\top A)^{-1} A^\top \Sigma_{YX} \Sigma_{XX}^{-1}$$

as the unique minimizer.

# Multi Layer Perceptron: Encoder-Decoder Architecture

**Recall:**  $E = \|\text{vec}(ABX) - \text{vec}(Y)\|_2^2$ .

**Homework:** Fix  $B$  and vary weights of  $A$  matrix and check for convexity of the loss function and solution characteristics of the associated minimization problem.

# Multi Layer Perceptron: Encoder-Decoder Architecture

We have the corresponding result:

For a fixed  $B$ , the loss function is convex with respect to  $\text{vec}(A)$  and the minimizer  $A$  satisfies the following relation:

$$AB\Sigma_{XX}B^\top = \Sigma_{YX}B^\top$$

If the assumptions that  $\Sigma_{XX}$  is invertible and  $B$  is full rank hold, then we have

$$A^* = \Sigma_{YX}B^\top(B\Sigma_{XX}B^\top)^{-1}$$

as the unique minimizer.

# Multi Layer Perceptron: Encoder-Decoder Architecture

Further, we have the corresponding result:

Assume  $\Sigma_{XX}$  is invertible and  $A$  is of full rank. Then  $A$  and  $B$  denote the critical points of the loss (or error) function  $E$ , that is,

$$\frac{\partial E}{\partial A_{ij}} = 0, \forall i \in \{1, \dots, n\}, j \in \{1, \dots, p\} \text{ and}$$

$$\frac{\partial E}{\partial B_{ij}} = 0, \forall i \in \{1, \dots, p\}, j \in \{1, \dots, n\}, \text{ if and only if } A \text{ satisfies:}$$

$$P_A \Sigma = \Sigma P_A = P_A \Sigma P_A, \text{ where } P_A = A(A^\top A)^{-1} A^\top, \Sigma = \Sigma_{YX} \Sigma_{XX}^{-1} \Sigma_{XY}.$$

and  $W = AB$  is of the form:

$$W = P_A \Sigma_{YX} \Sigma_{XX}^{-1}.$$