6.1 Perceptron Algorithm

6.1.1 Problem Setup

Given:

- A set of l-dimensional data samples:

$$\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\} \subset \mathbb{R}^l$$

- Two classes:

$$\omega = \{\omega_1, \omega_2\}$$

- A classification relation:

$$R : \mathcal{X} \mapsto \omega$$

  Do:
  Learn a decision hyper-plane:

$$g(\mathbf{x}) : \ \mathbf{w}^\top \mathbf{x} + w_0 = 0$$

  where $\mathbf{w}$ is the l-dimensional column weight vector and w_0 is the threshold.

Assume that the classes are linearly separable. Therefore:

$$\exists\ \mathbf{w}, w_0, \begin{cases} \mathbf{w}^\top \mathbf{x} + w_0 > 0 & \mathbf{x} \in \omega_1 \\ \\ \mathbf{w}^\top \mathbf{x} + w_0 < 0 & \mathbf{x} \in \omega_2 \end{cases}$$

We could omit the additional threshold by letting:

$$\mathbf{w}_{\text{new}} = \begin{bmatrix} \mathbf{w}_{\text{old}} \\ w_0 \end{bmatrix}$$

$$\mathbf{x}_{\text{new}} = \begin{bmatrix} \mathbf{x}_{\text{old}} \\ 1 \end{bmatrix}$$

Here:

- The original w_0 is inserted into the original $\mathbf{w}$, letting the new weight vector to grow by 1 dimension.
- Correspondingly, all the data samples are expanded by 1 dimension, with the new dimension being 1.
  Therefore, we could conclude that:

$$\mathbf{w}_{\text{old}}^\top \mathbf{x}_{\text{old}} + w_0 \equiv \mathbf{w}_{\text{new}}^\top \mathbf{x}_{\text{new}}$$

★ In general, the original linear separability could be expressed by:

$$\exists\, \mathbf{w} \in \mathbb{R}^{l+1}, \quad \begin{cases} \mathbf{w}^\top \mathbf{x} > 0 & \mathbf{x} \in \omega_1 \\ \mathbf{w}^\top \mathbf{x} < 0 & \mathbf{x} \in \omega_2 \end{cases}$$

Introduction to Perceptron Algorithm

Our goal is to compute such a solution. To reach this goal, we:

- Define a Cost Function.
- Choose an algorithm to *minimize* the cost function.
    - The minimum corresponds to a hyperplane solution.

6.1.2 Perceptron Cost Function

ℹ A "cost" is defined by the following:

$$\mathcal{J}(\mathbf{w}) = \sum_{\mathbf{x} \in \mathcal{Y}} \delta_{\mathbf{x}} \mathbf{w}^\top \mathbf{x}$$

where:

- $\mathcal{Y} \subset \mathcal{X}$ is the training vectors that's been *wrongly classified* by $\mathbf{w}$.
    - $\mathcal{Y} \in \emptyset$ means a solution is achieved.
- $\delta_{\mathbf{x}} = \begin{cases} -1 & \text{if } \mathbf{x} \in \omega_1 \\ +1 & \text{if } \mathbf{x} \in \omega_2 \end{cases}$

The gradient of the cost function:

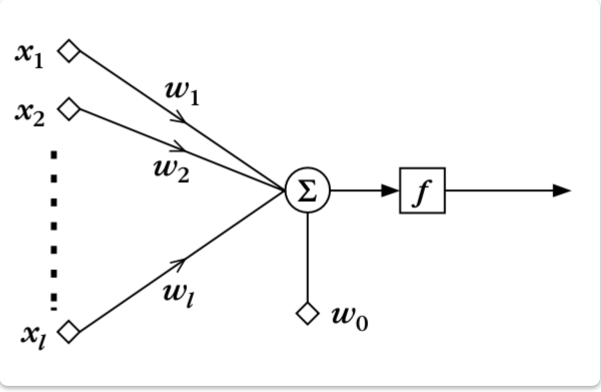
$$\begin{aligned} \frac{\partial \mathcal{J}(\mathbf{w})}{\partial \mathbf{w}} &= \frac{\partial}{\partial \mathbf{w}} \sum_{\mathbf{x} \in \mathcal{Y}} \delta_{\mathbf{x}} \mathbf{w}^\top \mathbf{x} \\ &= \sum_{\mathbf{x} \in \mathcal{Y}} \delta_{\mathbf{x}} \mathbf{x} \end{aligned}$$

★ The gradient descent algorithm would be:

$$\begin{aligned} \mathbf{w}_{t+1} &= \mathbf{w}_t - \rho_t \frac{\partial \mathcal{J}(\mathbf{w})}{\partial \mathbf{w}} \bigg|_{\mathbf{w}=\mathbf{w}_t} \\ &= \mathbf{w}_t - \rho_t \sum_{\mathbf{x} \in \mathcal{Y}} \delta_{\mathbf{x}} \mathbf{x} \end{aligned}$$

What exactly is a perceptron?

A demonstration of a single perceptron:



A perceptron is:

- A linear combination of inputs and weights.

6.1.3 Example: A single update step of weight

Given:

- Current weight with bias:

$$\mathbf{w}_t = \begin{bmatrix} w_1 \\ w_2 \\ w_0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ -0.5 \end{bmatrix}$$

- Wrongly classified data samples:

$$\mathbf{x}_1 = \begin{bmatrix} -0.2 \\ 0.75 \end{bmatrix}, \ \mathbf{x}_2 = \begin{bmatrix} 0.4 \\ 0.05 \end{bmatrix} \in \mathcal{Y}$$

  where $\mathbf{x}_1 \in \omega_2, \ \mathbf{x}_2 \in \omega_1$
- The learning rate of the current step:

$$\rho_t = \rho = 0.7$$

  Do:
  Update the weights of the separating plane:

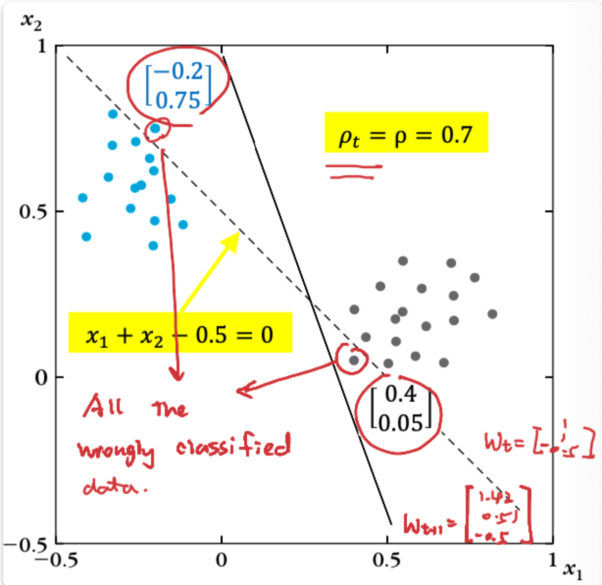
$$\mathbf{w}_{t+1} = \mathbf{w}_t - \rho \sum_{\mathbf{x} \in \mathcal{Y}} \delta_{\mathbf{x}} \mathbf{x}$$

$$= \begin{bmatrix} 1 \\ 1 \\ -0.5 \end{bmatrix} - 0.7 \times \left((+1) \begin{bmatrix} -0.2 \\ 0.75 \\ 1 \end{bmatrix} + (-1) \begin{bmatrix} 0.4 \\ 0.05 \\ 1 \end{bmatrix} \right)$$

$$= \begin{bmatrix} 1 \\ 1 \\ -0.5 \end{bmatrix} - 0.7 \times \begin{bmatrix} -0.2 - 0.4 \\ 0.75 - 0.05 \\ 1 - 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1.42 \\ 0.51 \\ -0.5 \end{bmatrix}$$

Visualization of this update:



6.2 XOR Problem and Multi-Layer Perceptron

6.2.1 XOR Problem

ℹ️ The XOR Problem illustrates:

- The inefficiency of a *Single Layer Perceptron* when,
- being faced with *Linear-Inseparable* data sets.

Given:

- Data samples:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \in \mathcal{X}$$

```
- where: $x_1, x_2\in\{0,1\}$
```

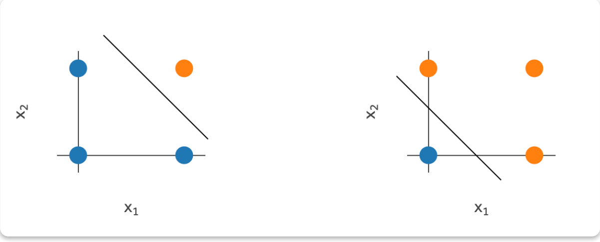
- The class of the data sample:

$$\omega_{\text{AND}}(\mathbf{x}) = x_1 \wedge x_2$$

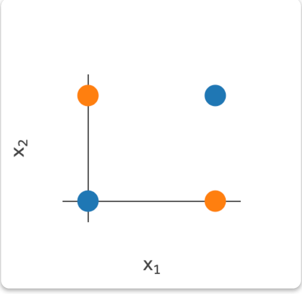
$$\omega_{\text{OR}}(\mathbf{x}) = x_1 \vee x_2$$

$$\omega_{\text{XOR}}(\mathbf{x}) = x_1 \oplus x_2$$

Do:
AND and OR could be solved by a single hyperplane.



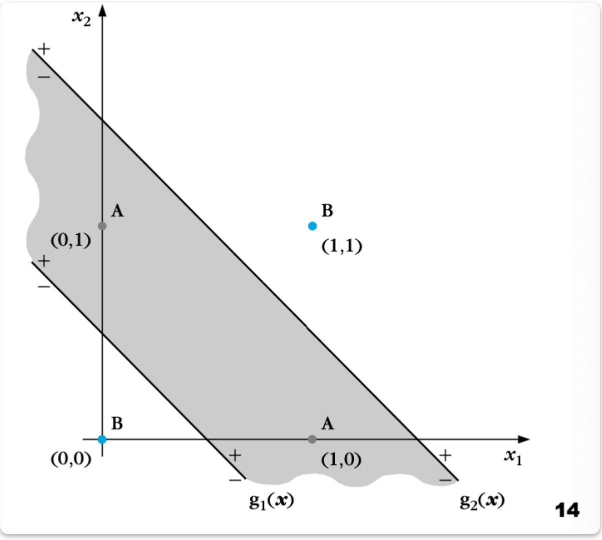
However, we could not decide a hyperplane that could separate XOR.



6.2.2 Two-Layer Perceptron

Key Point 1: Intuitive Solution of XOR Problem

To separate XOR, we need *two* lines, i.e. two hyperplanes.



The two hyperplanes:

$$g_1(\mathbf{x}) = \mathbf{w}_1^\top \mathbf{x} = 0$$

$$g_2(\mathbf{x}) = \mathbf{w}_2^\top \mathbf{x} = 0$$

Intuitively, we could get the observation solution by:

$$g_1(\mathbf{x}) > 0 \ \wedge \ g_2(\mathbf{x}) < 0 \implies \mathbf{x} \in \omega_1$$

$$g_1(\mathbf{x}) < 0 \ \vee \ g_2(\mathbf{x}) > 0 \implies \mathbf{x} \in \omega_2$$

After, we activate the outputs by:

$$y_i(\mathbf{x}) = f(g_i(\mathbf{x}))$$

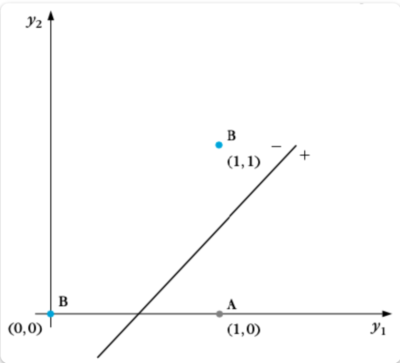
$$= \begin{cases} 0 & \text{if } g_i(\mathbf{x}) < 0 \\ 1 & \text{if } g_i(\mathbf{x}) > 0 \end{cases}$$

we could convert the solution as:

$$y_1(\mathbf{x}) = 1 \ \wedge \ y_2(\mathbf{x}) = 0 \implies \mathbf{x} \in \omega_1$$

$$y_1(\mathbf{x}) = 0 \ \vee \ g_2(\mathbf{x}) = 1 \implies \mathbf{x} \in \omega_2$$

With this expression, we could convert the solution into:



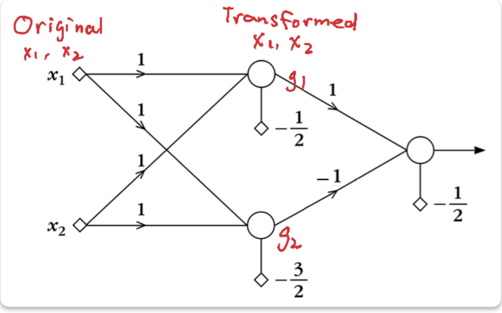
> *Note that $y_2 = 1 \implies y_1 \neq 0$, regions upper than the second line can't be lower than the first line. Therefore, the coordinate $(0, 1)$ is empty.*

This is a linear-separable case, which could be solved by the 1-layer perceptron.

- There are 2 axes.
- The value on each axes corresponds to:
  - The relative position of the original datapoint,
  - with respect to the separating hyperplane defined in the first layer.
  - e.g., $y_1 = 1$ means that the point is above the plane g_1.

Key Point 2: More details with the XOR Problem's Solution

The previous XOR problem solution could be illustrated in the following *2-layer perceptron.*



First Layer

The first layer has two neurons:

- Each neuron takes a 2-dimensional input, i.e., the data sample.
- Each neuron mimics a *Hyperplane.*
- The *output* of each neuron tells the *Region* separated by the hyperplane.

$$\text{First Neuron: } f(\begin{bmatrix}1 & 1 & -\frac{1}{2}\end{bmatrix}\begin{bmatrix}x_1\\x_2\\1\end{bmatrix}) = f(x_1 + x_2 - \frac{1}{2}) \in \{0,1\}$$

$$\text{Second Neuron: } f(\begin{bmatrix}1 & 1 & -\frac{3}{2}\end{bmatrix}\begin{bmatrix}x_1\\x_2\\1\end{bmatrix}) = f(x_1 + x_2 - \frac{3}{2}) \in \{0,1\}$$

The first layer *Maps* the linear inseparable data into linear separable ones.

- This is done with the help of f, the **Activation Function.**

$$f(z) = \begin{cases} 1 & \text{if } z > 0 \\ 0 & \text{if } z \le 0 \end{cases}$$

Second Layer

The second layer has only one neuron:

- It mimics the hyperplane that separates the *mapped data.*

$$\text{Output Neuron: } f(\begin{bmatrix}1 & -1 & -\frac{1}{2}\end{bmatrix}\begin{bmatrix}y_1\\y_2\\1\end{bmatrix}) = f(y_1 - y_2 - \frac{1}{2}) \in \{0,1\}$$

6.2.3 Three-Layer Perceptron

Key Point 1: Two-Phase Process of Two-Layered Network

Let's take a look at the general form of a two-layered network

Given:

- Three hyperplanes:

$$g_1(\mathbf{x}) = \begin{bmatrix}1 & -1 & 0\end{bmatrix}\begin{bmatrix}x_1\\x_2\\1\end{bmatrix} = 0$$

$$g_2(\mathbf{x}) = \begin{bmatrix}1 & 1 & 0\end{bmatrix}\begin{bmatrix}x_1\\x_2\\1\end{bmatrix} = 0$$

$$g_3(\mathbf{x}) = \begin{bmatrix}0 & 1 & -\frac{1}{4}\end{bmatrix}\begin{bmatrix}x_1\\x_2\\1\end{bmatrix} = 0$$

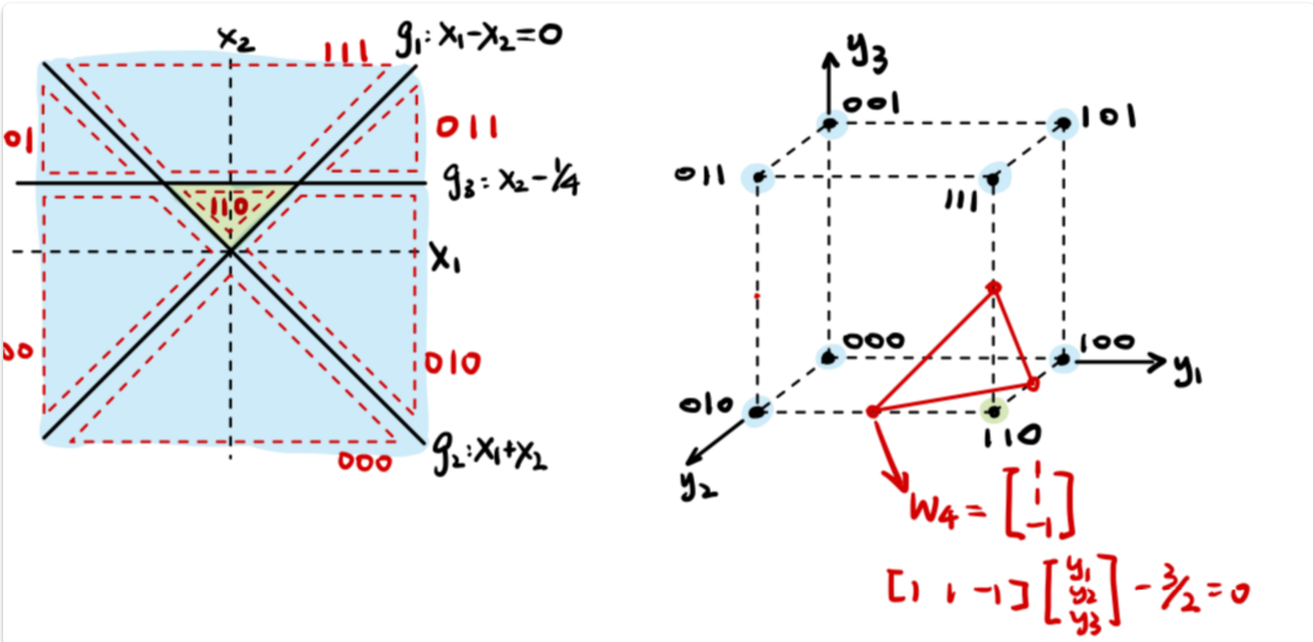
  Do:
- Separate the central regions from others using a two-layered perceptron.

$$g_1(\mathbf{x}) > 0 \ \wedge \ g_2(\mathbf{x}) > 0 \ \wedge \ g_3(\mathbf{x}) < 0$$

- Namely, the regions named 110.

Phase 1.

- The original space is of l dimensions.
- Suppose there are p hyperplanes (of l dimensions) in the space.
- In the first layer, l-dimensional space will be transformed into a p-dimensional space.
  - The transformation is done with the help of Activation Function.



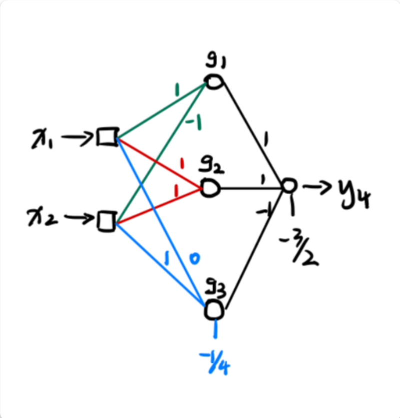
- By observation:
  - Regions in l-d space $\rightarrow$ Cube vertices in p-d space.
  - Separating regions in l-d space $\rightarrow$ Separating vertices in p-d space.

  Phase 2.

- The second layer, i.e., the output layer, separates vertices in p-dimensional space.
  - The single perceptron of the second layer mimics the hyperplane in the p-d space.
- In the given example, the hyperplane in p-d space is:

$$g_4(\mathbf{x}) = \begin{bmatrix} 1 & 1 & -1 & -\frac{3}{2} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ 1 \end{bmatrix} = 0$$

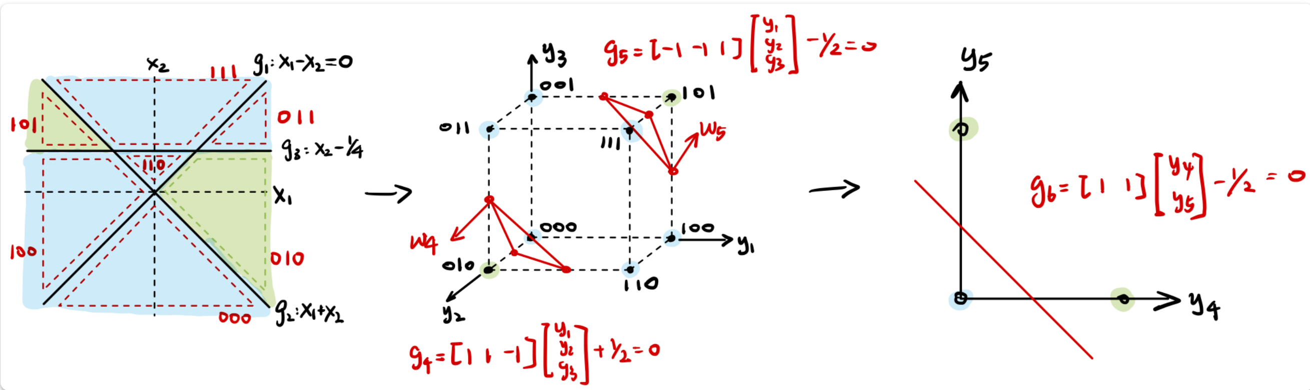
Summary.
The overall structure of the two-layered neural network is:



- The first layer mimics the three lines, transforming 2-d into 3-d.
- The second layer mimics the hyperplane in 3-d space that separates the vertices.

Key Point 2: Deficiencies Two-Layered Network

By the example above, we could predict that the two-layered neural network can't separate all classes. For instance, in the following case:



- For disconnected regions in the l-space, their corresponding vertices are also "disconnected" in the p-space.
- We need 2 hyperplanes in the p-space to separate them!
  Therefore, we need a 3-Phase Process.

Key Point 3: Three-Phase Process of Three-Layered Network

Phase 1.

- In the 2-d space, there are 3 hyperplanes.
- The first layer uses 3 perceptrons to mimic the 3 hyperplanes.
- The original 2-d space is thus transformed into a 3-d space.
  Phase 2.
- In the 3-d space, we realized that we need to use two p-dimensional separating planes ($p = 2$) to separate the two vertices 010 and 101.

$$g_4(\mathbf{x}) = \begin{bmatrix} 1 & 1 & -1 & \frac{1}{2} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ 1 \end{bmatrix} = 0$$

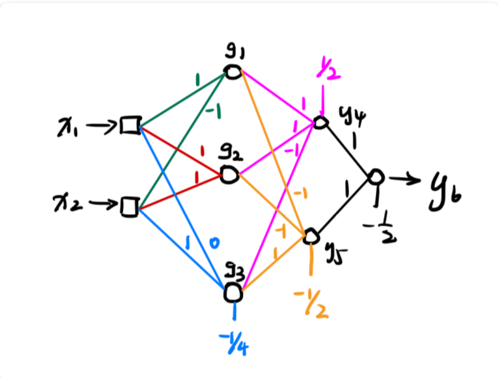
$$g_5(\mathbf{x}) = \begin{bmatrix} -1 & -1 & 1 & -\frac{1}{2} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ 1 \end{bmatrix} = 0$$

Phase 3.
- The second layer transposes the intermediate 3-d space into a new 2-d space, considering there are 2 separating planes.
- Separate the new linear-separable 2-d vertices with the hyperplane:

$$g_6(\mathbf{x}) = \begin{bmatrix} 1 & 1 & -\frac{1}{2} \end{bmatrix} \begin{bmatrix} y_4 \\ y_5 \\ 1 \end{bmatrix} = 0$$

The separation process is thus done.

Summary.
The overall structure of the three-layered neural network is:

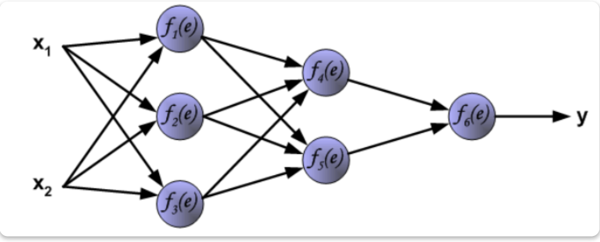


- The first layer mimics the three lines, transforming 2-d into 3-d.
- The second layer mimics the two hyperplanes in the 3-d space, transforming 3-d into 2-d.
- The third layer, i.e., the output layer, mimics the separating line in the 2-d space from the second layer.

6.3 Back Propagation Algorithm

6.3.1 Forward Propagation

We construct a neural network as follows.



Key Point 1: Detailed Explanation of NN Structure

There are three layers in this neural network. Initially, weights are randomized.

Layer 1: $2 \to 3$

- Weights:

$$\mathbf{W}_1 = \begin{bmatrix} \mathbf{w}_{11}^\top \\ \mathbf{w}_{12}^\top \\ \mathbf{w}_{13}^\top \end{bmatrix} = \begin{bmatrix} w_{11,1} & w_{11,2} & b_{11} \\ w_{12,1} & w_{12,2} & b_{12} \\ w_{13,1} & w_{13,2} & b_{13} \end{bmatrix}$$

Layer 2: $3 \to 2$

- Weights:

$$\mathbf{W}_2 = \begin{bmatrix} \mathbf{w}_{21}^\top \\ \mathbf{w}_{22}^\top \end{bmatrix} = \begin{bmatrix} w_{21,1} & w_{21,2} & w_{21,3} & b_{21} \\ w_{22,1} & w_{22,2} & w_{22,3} & b_{22} \end{bmatrix}$$

Layer 3: $2 \to 1$

- Weights:

$$\mathbf{W}_3 = \mathbf{w}_{31}^\top = \begin{bmatrix} w_{31,1} & w_{31,2} & b_{31} \end{bmatrix}$$

Key Point 2: Forward Propagation Process

- Original Data with ground truth:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix}, y_d$$

Layer 1:

$$\mathbf{y}_1' = f(\mathbf{W}_1\mathbf{x})$$

$$= f(\begin{bmatrix} w_{11,1} & w_{11,2} & b_{11} \\ w_{12,1} & w_{12,2} & b_{12} \\ w_{13,1} & w_{13,2} & b_{13} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix})$$

$$= f(\begin{bmatrix} w_{11,1} \cdot x_1 + w_{11,2} \cdot x_2 + b_{11} \\ w_{12,1} \cdot x_1 + w_{12,2} \cdot x_2 + b_{12} \\ w_{13,1} \cdot x_1 + w_{13,2} \cdot x_2 + b_{13} \end{bmatrix})$$

$$= f(\begin{bmatrix} g_{11} \\ g_{12} \\ g_{13} \end{bmatrix})$$

$$= \begin{bmatrix} y_{11} \\ y_{12} \\ y_{13} \end{bmatrix}$$

$$\mathbf{y}_1 = \begin{bmatrix} y_{11} \\ y_{12} \\ y_{13} \\ 1 \end{bmatrix}$$

Layer 2:

$$\mathbf{y}_2' = f(\mathbf{W}_2\mathbf{y}_1)$$

$$= f(\begin{bmatrix} w_{21,1} & w_{21,2} & w_{21,3} & b_{21} \\ w_{22,1} & w_{22,2} & w_{22,3} & b_{22} \end{bmatrix} \begin{bmatrix} y_{11} \\ y_{12} \\ y_{13} \\ 1 \end{bmatrix})$$

$$= f(\begin{bmatrix} w_{21,1} \cdot y_{11} + w_{21,2} \cdot y_{12} + w_{21,3} \cdot y_{13} + b_{21} \\ w_{22,1} \cdot y_{11} + w_{22,2} \cdot y_{12} + w_{22,3} \cdot y_{13} + b_{22} \end{bmatrix})$$

$$= f(\begin{bmatrix} g_{21} \\ g_{22} \end{bmatrix})$$

$$= \begin{bmatrix} y_{21} \\ y_{22} \end{bmatrix}$$

$$\mathbf{y}_2 = \begin{bmatrix} y_{21} \\ y_{22} \\ 1 \end{bmatrix}$$

Layer 3:

$$\mathbf{y}_3' = f(\mathbf{W}_3\mathbf{y}_2)$$

$$= f(\begin{bmatrix} w_{31,1} & w_{31,2} & b_{31} \end{bmatrix} \begin{bmatrix} y_{21} \\ y_{22} \\ 1 \end{bmatrix})$$

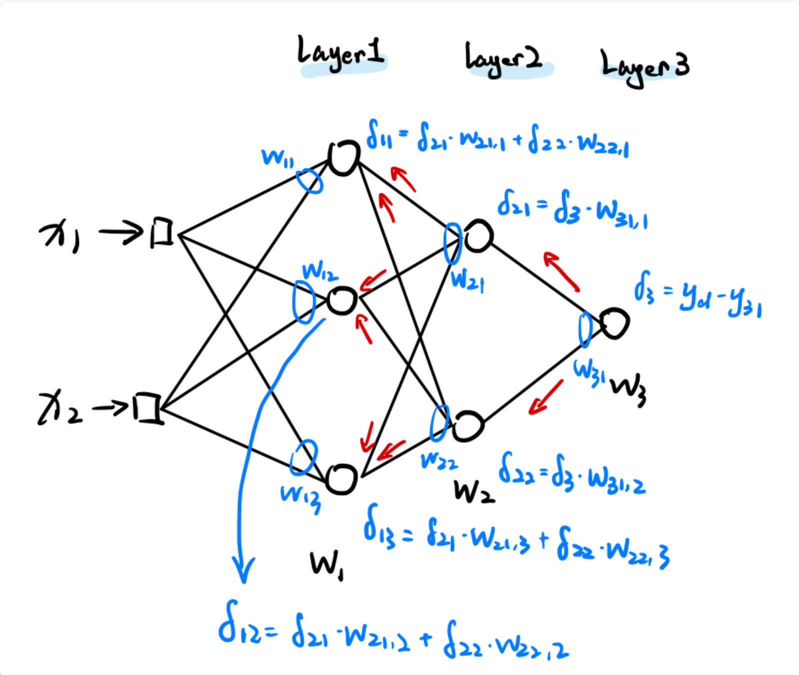
$$= f(w_{31,1} \cdot y_{21} + w_{31,2} \cdot y_{22} + b_{31})$$

$$= f(g_{31})$$

$$= y_{31}$$

$$\mathbf{y}_3 = y_{31}$$

Key Point 3: Back Propagation Algorithm



Layer 3:

$$\delta_3 = y_d - y_{31}$$

Layer 2:

$$\delta_{21} = \delta_3 \cdot w_{31,1}$$

$$\delta_{22} = \delta_3 \cdot w_{31,2}$$

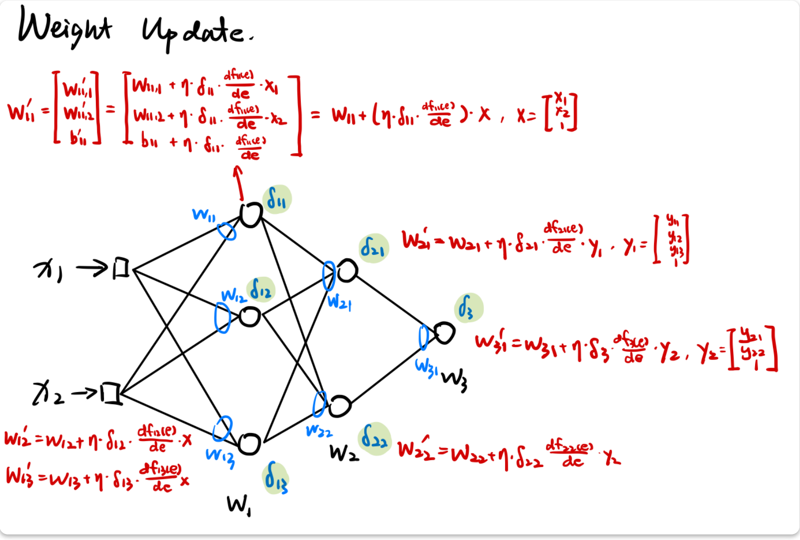
Layer 1:

$$\delta_{11} = \delta_{21} \cdot w_{21,1} + \delta_{22} \cdot w_{22,1}$$

$$\delta_{12} = \delta_{21} \cdot w_{21,2} + \delta_{22} \cdot w_{22,2}$$

$$\delta_{13} = \delta_{21} \cdot w_{21,3} + \delta_{22} \cdot w_{22,3}$$

Key Point 4: Weight Update



Layer 1

$$
\mathbf{w}'_{11} = \begin{bmatrix} w'_{11,1} \\ w'_{11,2} \\ b'_{11} \end{bmatrix}
$$

$$
= \begin{bmatrix} w_{11,1} + \eta \cdot \delta_{11} \cdot \dfrac{df_{11}(e)}{de} \cdot x_1 \\ w_{11,2} + \eta \cdot \delta_{11} \cdot \dfrac{df_{11}(e)}{de} \cdot x_2 \\ b_{11} + \eta \cdot \delta_{11} \cdot \dfrac{df_{11}(e)}{de} \cdot 1 \end{bmatrix}
$$

$$
= \mathbf{w}_{11} + \eta \cdot \delta_{11} \cdot \frac{df_{11}(e)}{de} \cdot \mathbf{x}
$$

$$
\mathbf{w}'_{12} = \mathbf{w}_{12} + \eta \cdot \delta_{12} \cdot \frac{df_{12}(e)}{de} \cdot \mathbf{x}
$$

$$
\mathbf{w}'_{13} = \mathbf{w}_{13} + \eta \cdot \delta_{13} \cdot \frac{df_{13}(e)}{de} \cdot \mathbf{x}
$$

where,

$$
\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix}
$$

Layer 2

$$
\mathbf{w}'_{21} = \mathbf{w}_{21} + \eta \cdot \delta_{21} \cdot \frac{df_{21}(e)}{de} \cdot \mathbf{y}_1
$$

$$
\mathbf{w}'_{22} = \mathbf{w}_{22} + \eta \cdot \delta_{22} \cdot \frac{df_{22}(e)}{de} \cdot \mathbf{y}_1
$$

where,

$$\mathbf{y}_1 = \begin{bmatrix} y_{11} \\ y_{12} \\ y_{13} \\ 1 \end{bmatrix}$$

Layer 3

$$\mathbf{w}'_{31} = \mathbf{w}_{31} + \eta \cdot \delta_3 \cdot \frac{df_{31}(e)}{de} \cdot \mathbf{y}_2$$

where,

$$\mathbf{y}_2 = \begin{bmatrix} y_{21} \\ y_{22} \\ 1 \end{bmatrix}$$