

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

# Deep Learning



---

Mohammad Ali Keyvanrad

Lecture 2:A Review of Artificial Neural Networks

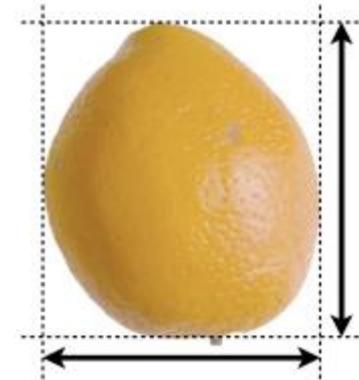
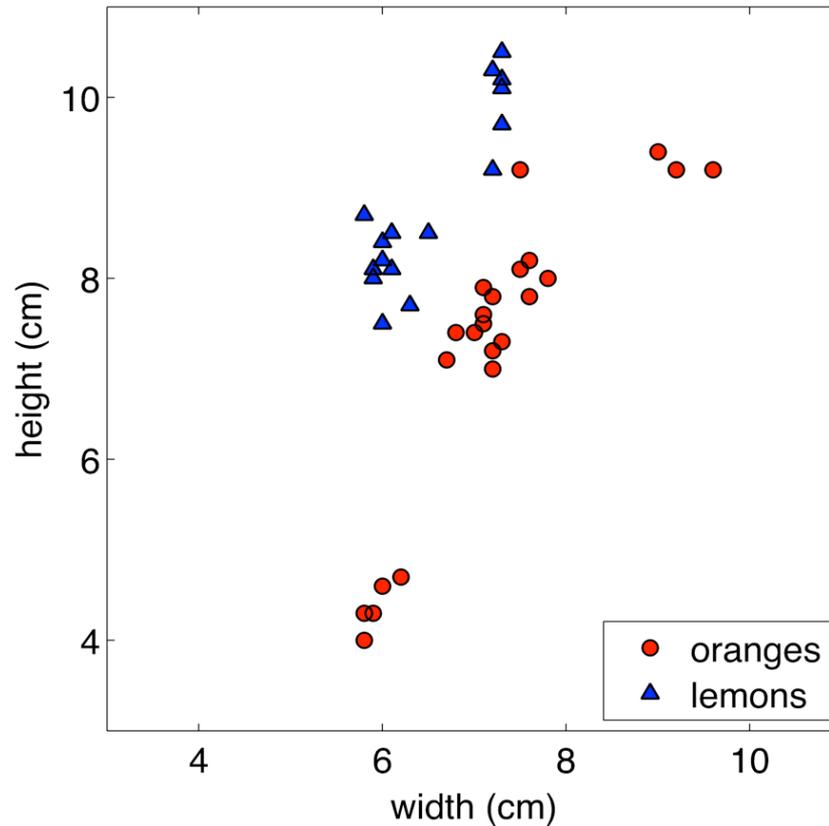
# OUTLINE

- Perceptron
- Stochastic Gradient Descent
- Backpropagation
  - Boxing
  - Sensitivity
  - Weight updates

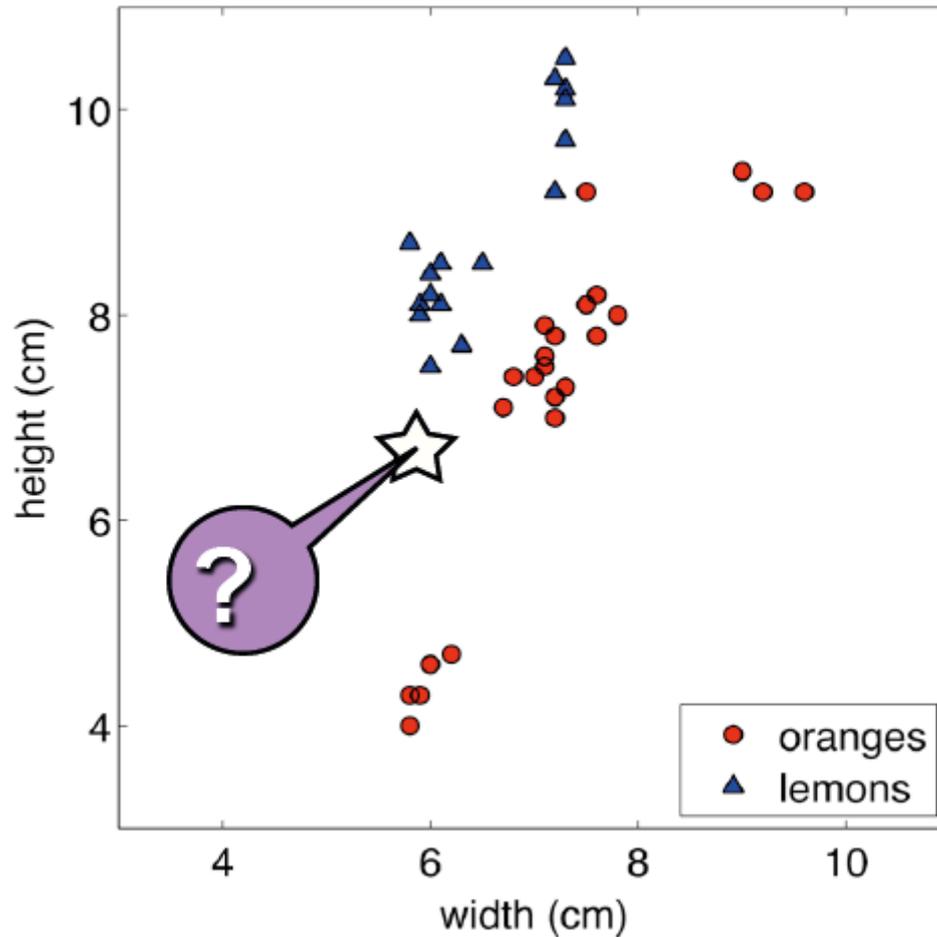
# OUTLINE

- **Perceptron**
- Stochastic Gradient Descent
- Backpropagation
  - Boxing
  - Sensitivity
  - Weight updates

# Classification: Oranges and Lemons



# Classification: Oranges and Lemons



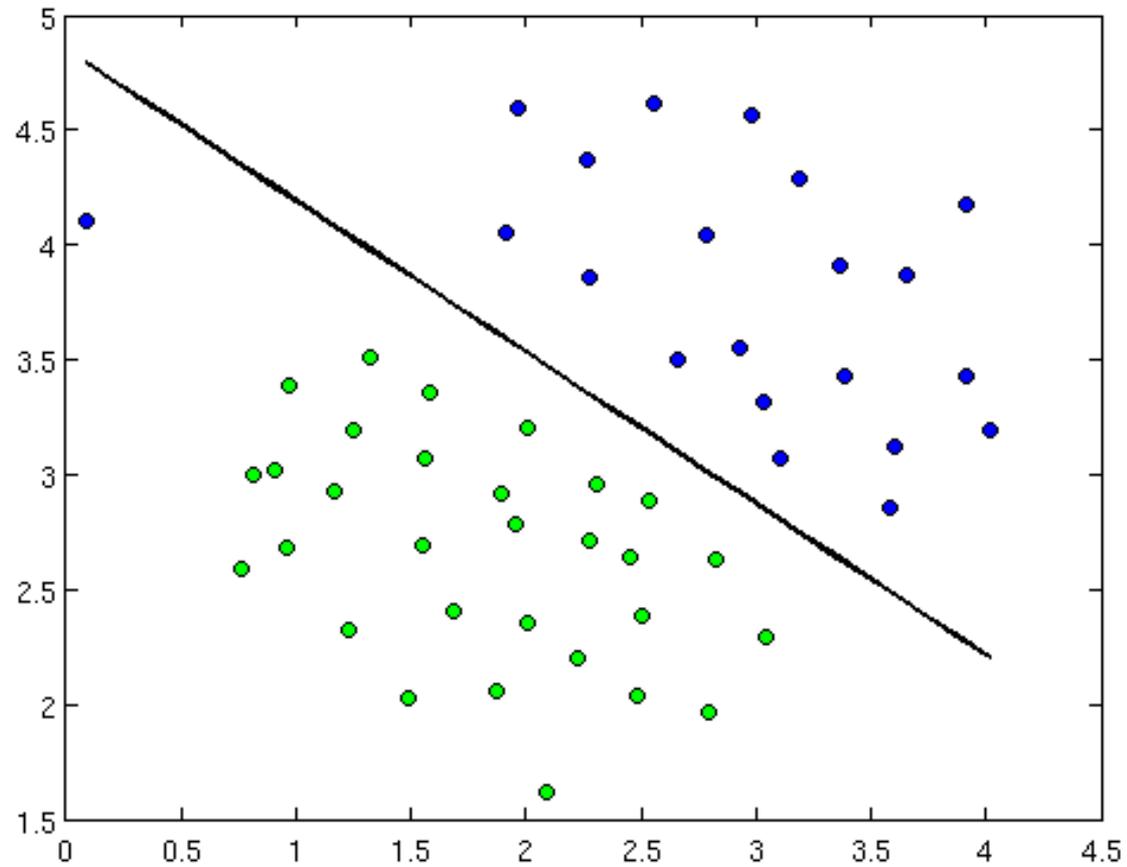
# Classification problem

---

- Given: Training set
  - labeled set of  $N$  input-output pairs
    - $D = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^N$
    - $y = \{1, \dots, K\}$
- Goal: Given an input  $\mathbf{x}$ , assign it to one of  $K$  classes
- Examples:
  - Spam filter
  - Handwritten digit recognition

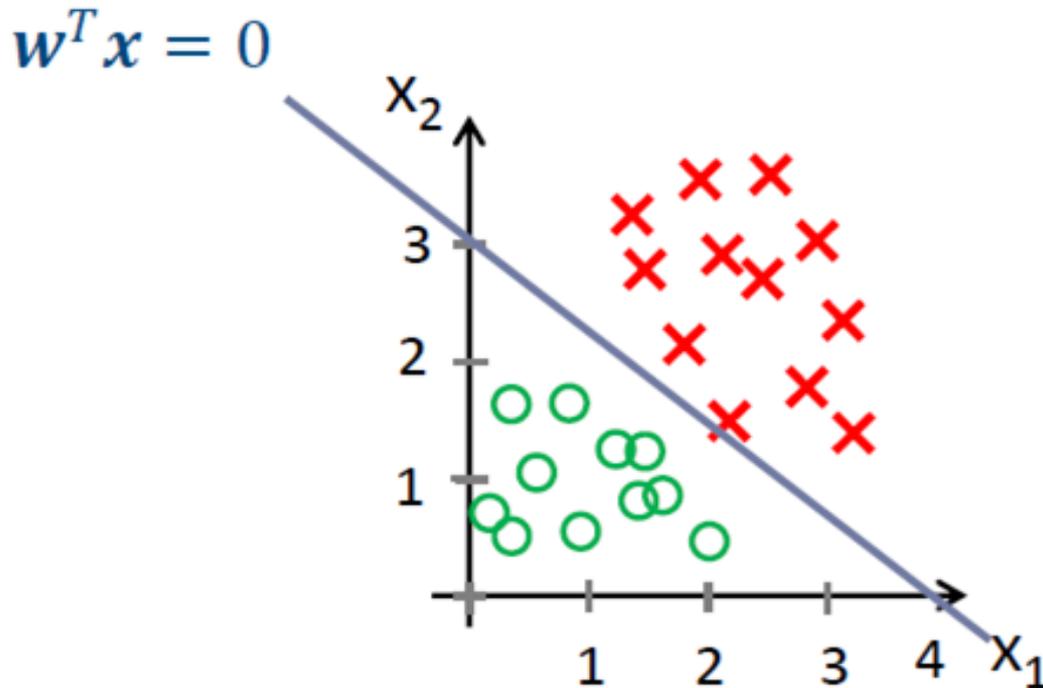
# Perceptron

---



# Decision boundary

- Linear classifier



$$-3 + \frac{3}{4}x_1 + x_2 = 0$$

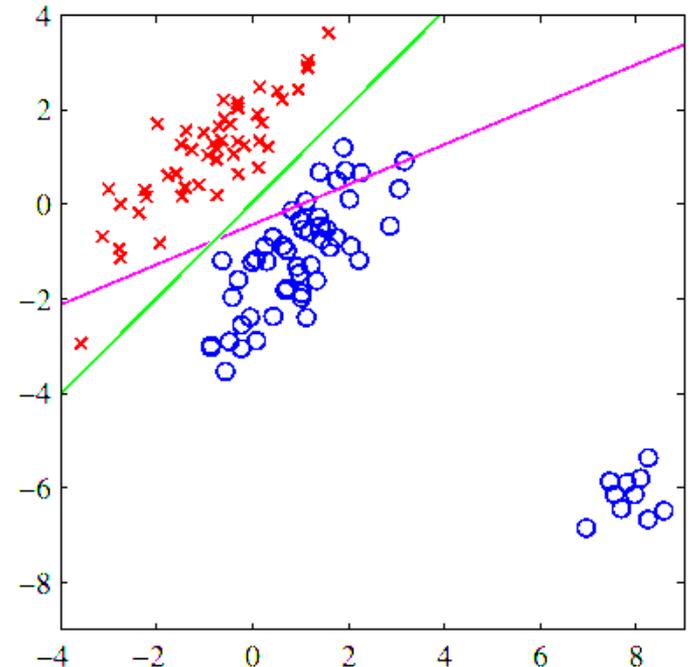
*if  $w^T x \geq 0$  then  $y = 1$   
Else  $y = -1$*

$$w = [-3, 0.75, 1]$$

# SSE cost function for classification

- SSE cost function is not suitable for classification
  - Sum of Squared Errors loss penalizes “too correct” predictions
  - SSE also lack robustness to noise

$$J(\mathbf{w}) = \sum_{i=1}^N (\mathbf{w}^T \mathbf{x}^{(i)} - y^{(i)})^2$$

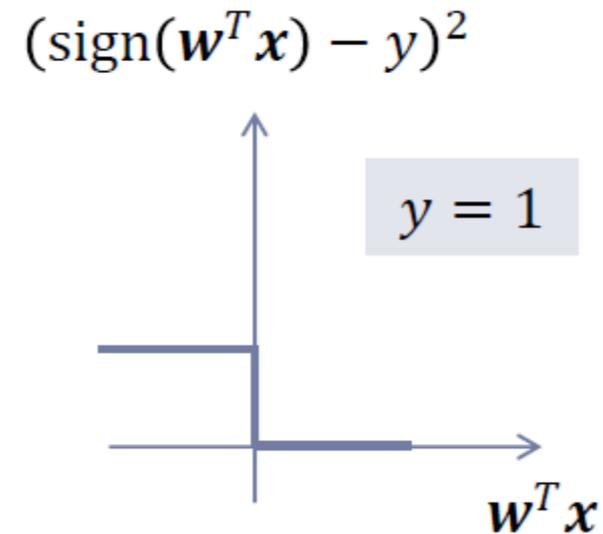


# SSE cost function for classification

- Is it more suitable if we set  $f(\mathbf{x}; \mathbf{w}) = g(\mathbf{w}^T \mathbf{x})$ ?

$$J(\mathbf{w}) = \sum_{i=1}^N (g(\mathbf{w}^T \mathbf{x}^{(i)}) - y^{(i)})^2$$

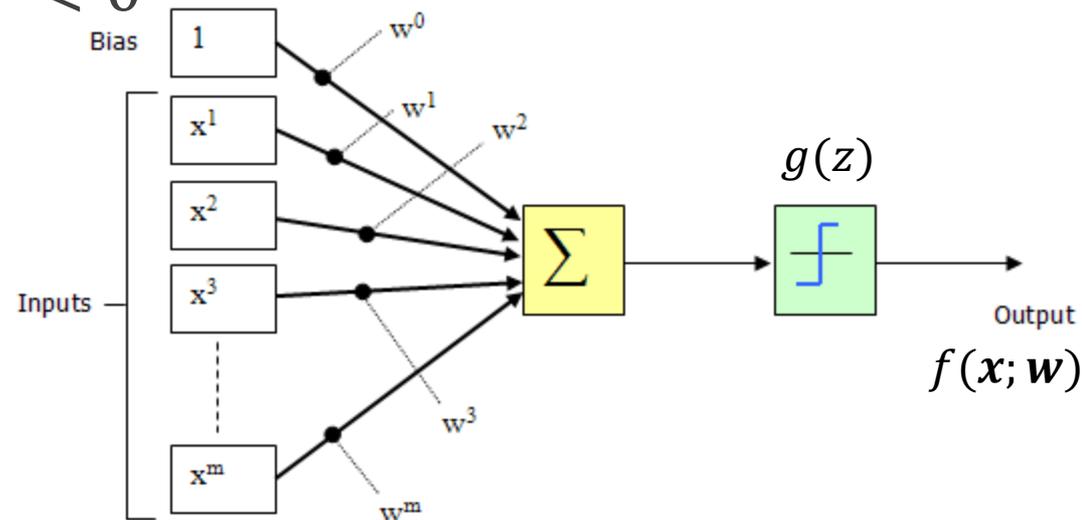
$$g(z) = \text{sign}(z) = \begin{cases} -1, & z < 0 \\ 1, & z \geq 0 \end{cases}$$



# Perceptron algorithm

- Linear classifier
- Two-class:  $y \in \{-1, 1\}$ ;  $y = -1$  for  $C_2$ ,  $y = 1$  for  $C_1$
- Goal
  - $\forall i, \mathbf{x}^{(i)} \in C_1 \Rightarrow \mathbf{w}^T \mathbf{x}^{(i)} > 0$
  - $\forall i, \mathbf{x}^{(i)} \in C_2 \Rightarrow \mathbf{w}^T \mathbf{x}^{(i)} < 0$

- $f(\mathbf{x}; \mathbf{w}) = g(\mathbf{w}^T \mathbf{x})$ 
  - $g(z) = \begin{cases} -1, & z < 0 \\ +1, & z \geq 0 \end{cases}$



# Perceptron criterion

---

- *Misclassification*

- $\forall i, \mathbf{x}^{(i)} \in C_1 \Rightarrow y^{(i)} = +1$  but  $\mathbf{w}^T \mathbf{x}^{(i)} < 0$

- $\forall i, \mathbf{x}^{(i)} \in C_2 \Rightarrow y^{(i)} = -1$  but  $\mathbf{w}^T \mathbf{x}^{(i)} > 0$


$$J_P(\mathbf{w}) = - \sum_{i \in \mathcal{M}} \mathbf{w}^T \mathbf{x}^{(i)} y^{(i)}$$

$\mathcal{M}$ : subset of training data that are misclassified

# OUTLINE

- Perceptron
- **Stochastic Gradient Descent**
- Backpropagation
  - Boxing
  - Sensitivity
  - Weight updates

# Batch gradient for descent Perceptron

---

- “Gradient Descent” to solve the optimization problem

$$\mathbf{w}^{t+1} = \mathbf{w}^t - \eta \nabla_{\mathbf{w}} J_P(\mathbf{w}^t)$$

$$\nabla_{\mathbf{w}} J_P(\mathbf{w}) = - \sum_{i \in \mathcal{M}} \mathbf{x}^{(i)} y^{(i)}$$

- Batch Perceptron converges in finite number of steps for linearly separable data

```
Initialize  $\mathbf{w}, t \leftarrow 0$   
Repeat  
     $\mathbf{w} = \mathbf{w} + \eta \sum_{i \in \mathcal{M}} \mathbf{x}^{(i)} y^{(i)}$   
     $t \leftarrow t + 1$   
Until  $\eta \sum_{i \in \mathcal{M}} \mathbf{x}^{(i)} y^{(i)} < \theta$ 
```

# Stochastic gradient descent for Perceptron

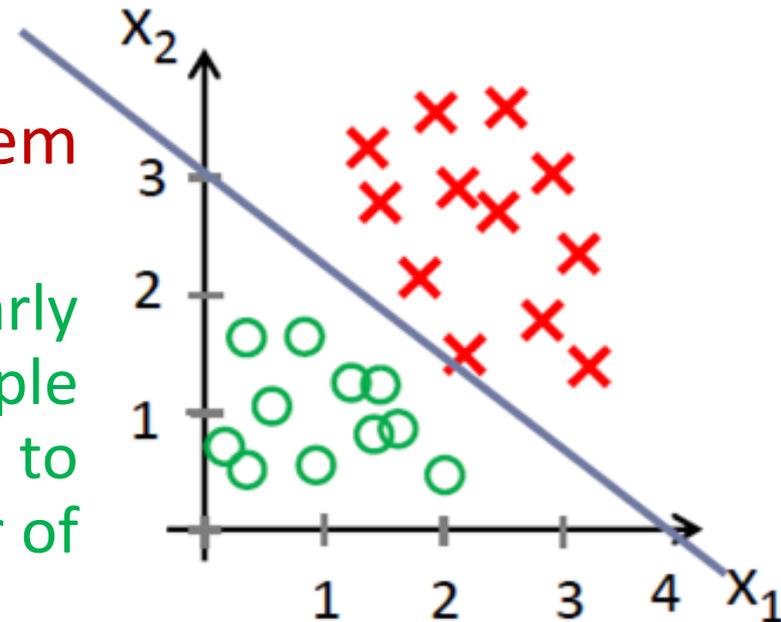
- Single-sample perceptron

- If  $\mathbf{x}^{(i)}$  is misclassified

$$\mathbf{w}^{t+1} = \mathbf{w}^t + \eta \mathbf{x}^{(i)} y^{(i)}$$

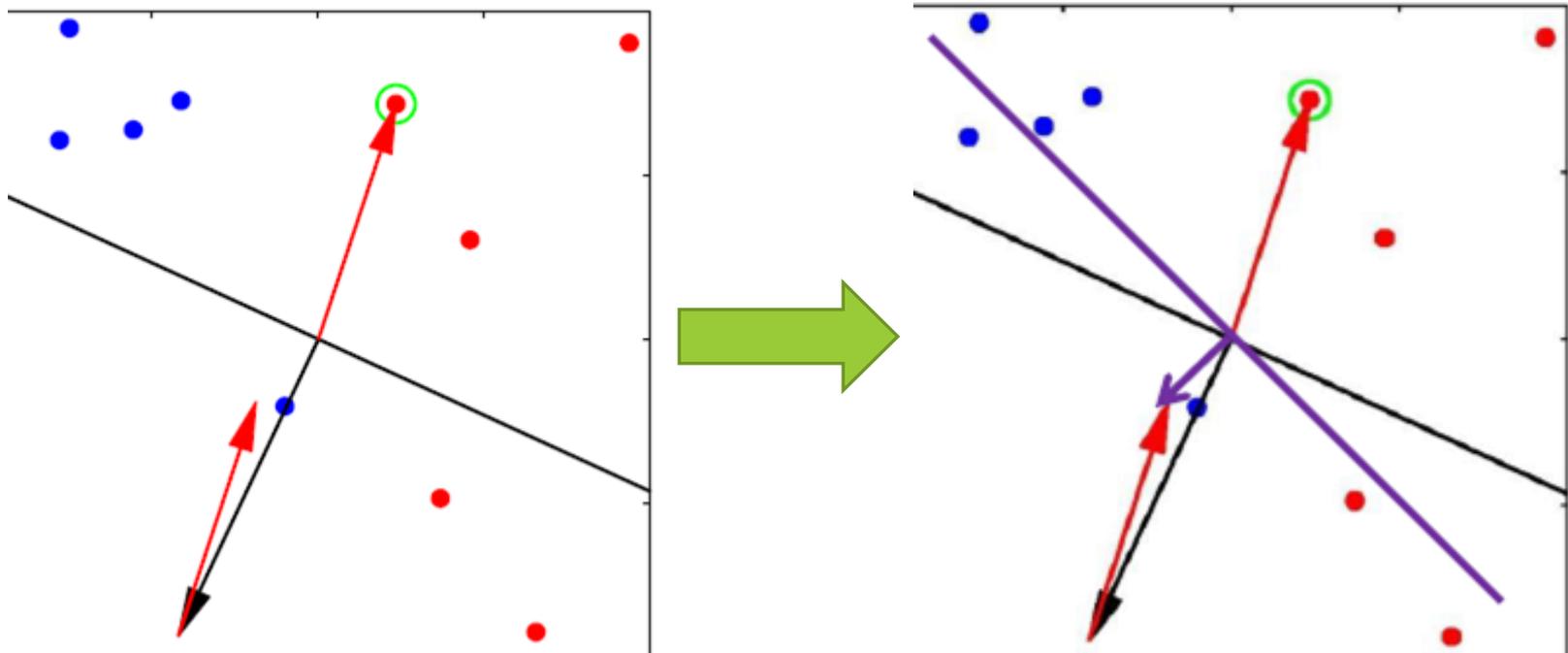
- **Perceptron convergence theorem**  
(for linearly separable data)

- If training data are linearly separable, the single-sample perceptron is also guaranteed to find a solution in a finite number of steps.

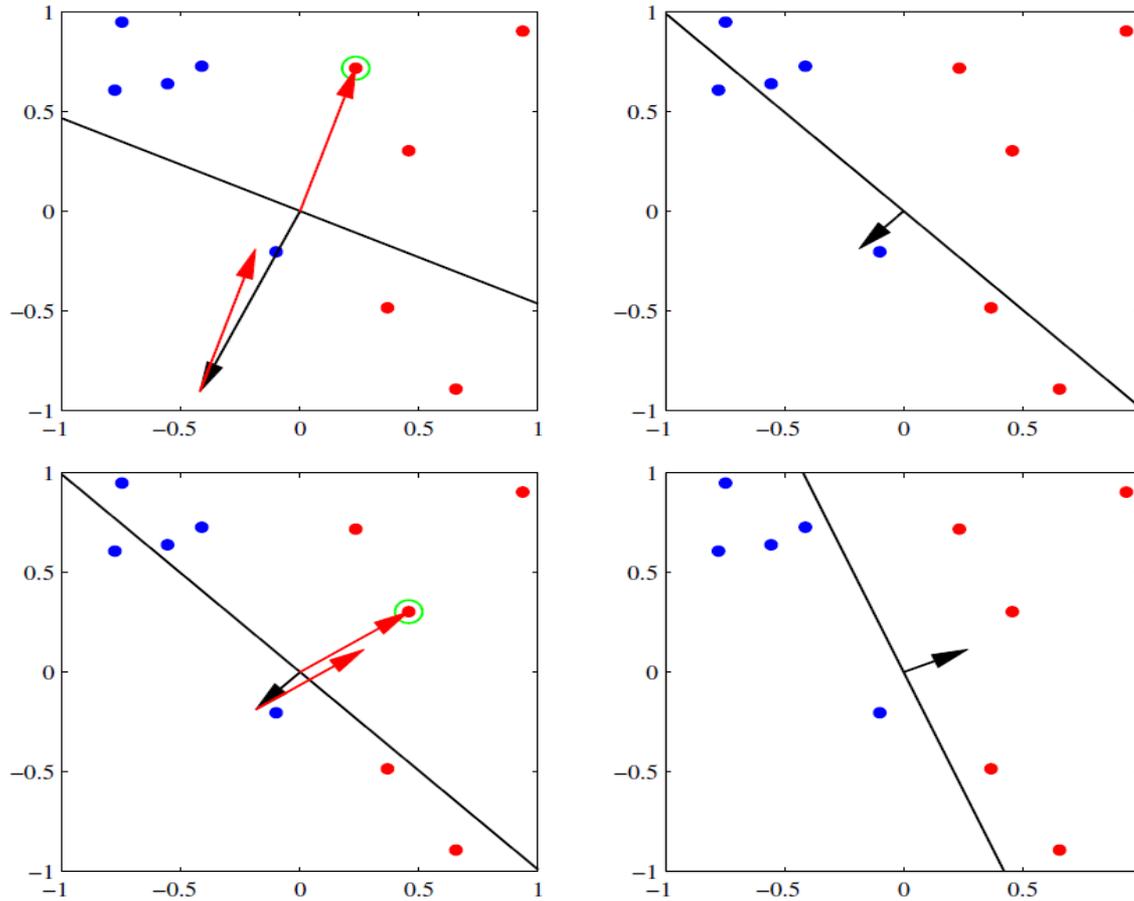


# Convergence of Perceptron

- Change  $w$  in a direction that corrects the error



# Convergence of Perceptron

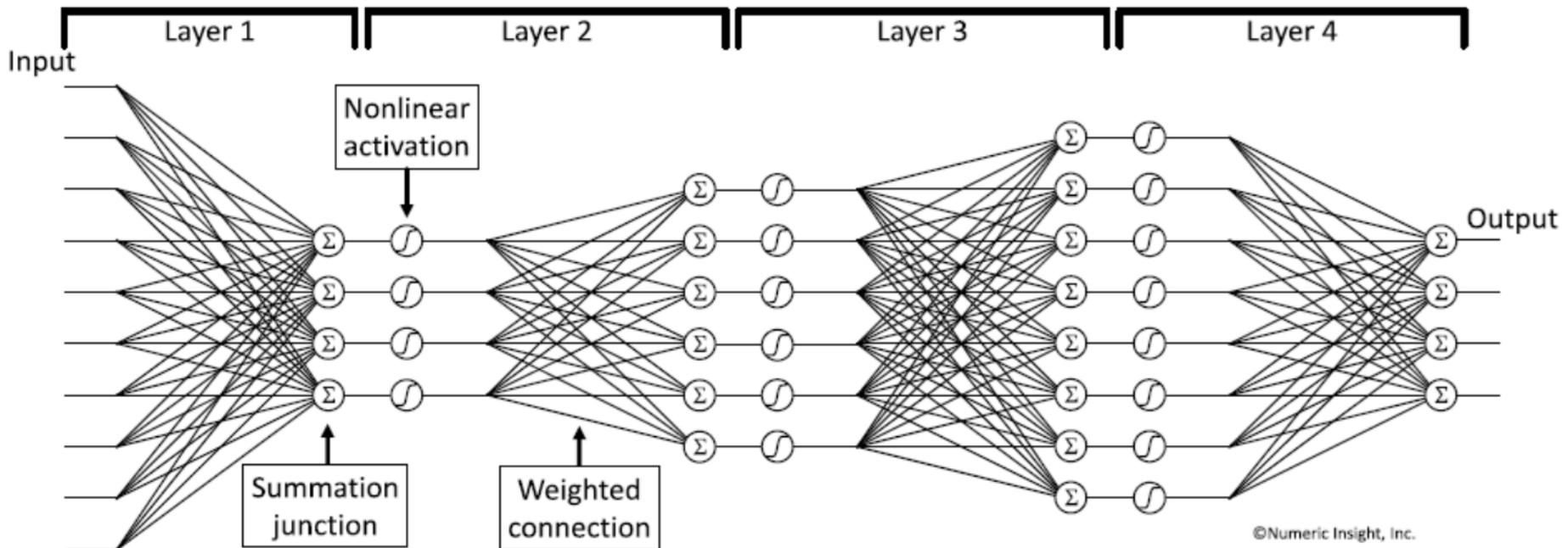


# OUTLINE

- Perceptron
- Stochastic Gradient Descent
- **Backpropagation**
  - Boxing
  - Sensitivity
  - Weight updates

# Backpropagation

- An example of a multi-layered neural network
  - Input: 10 numbers
  - Output: 4 decisions or predictions



# OUTLINE

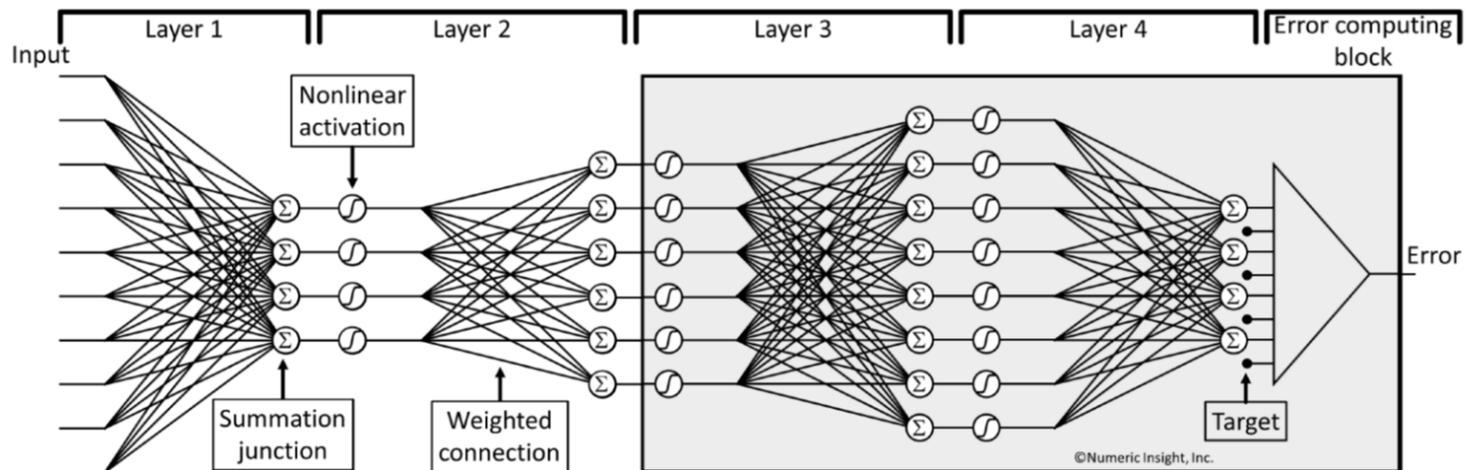
- Perceptron
- Stochastic Gradient Descent
- Backpropagation
  - **Boxing**
  - Sensitivity
  - Weight updates

# Backpropagation (concept 1)

- **Boxing**

- Tacking on an extra computational block to calculate the error
- Next, we choose one of the layers (say Layer 3) and enclose that layer and all following layers in a box

$$e = (o_1 - t_1)^2 + (o_2 - t_2)^2 + (o_3 - t_3)^2 + (o_4 - t_4)^2$$



# Backpropagation (concept 1)

- **Boxing (cont.)**

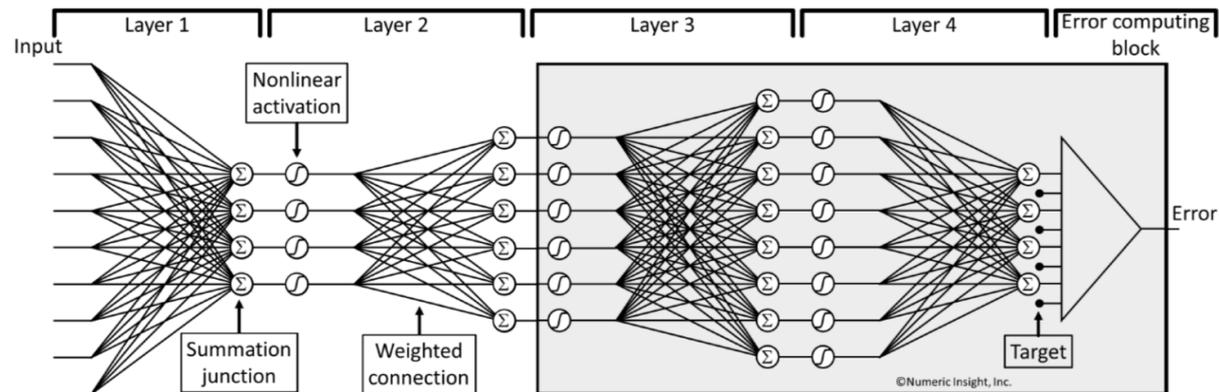
- This box: **current box**

- Input to this box: **current input**

- $\{c_1, c_2, c_3, c_4, c_5, c_6\}$

- Relationship between the current input to the box and the output

- $e = E(c_1, c_2, c_3, c_4, c_5, c_6)$



# OUTLINE

- Perceptron
- Stochastic Gradient Descent
- Backpropagation
  - Boxing
  - **Sensitivity**
  - Weight updates

# Backpropagation (concept 2)

- **Sensitivity**

- partial derivatives

- $\frac{\partial E}{\partial c_1}, \frac{\partial E}{\partial c_2}, \frac{\partial E}{\partial c_3}, \dots$

- given a name for partial derivatives

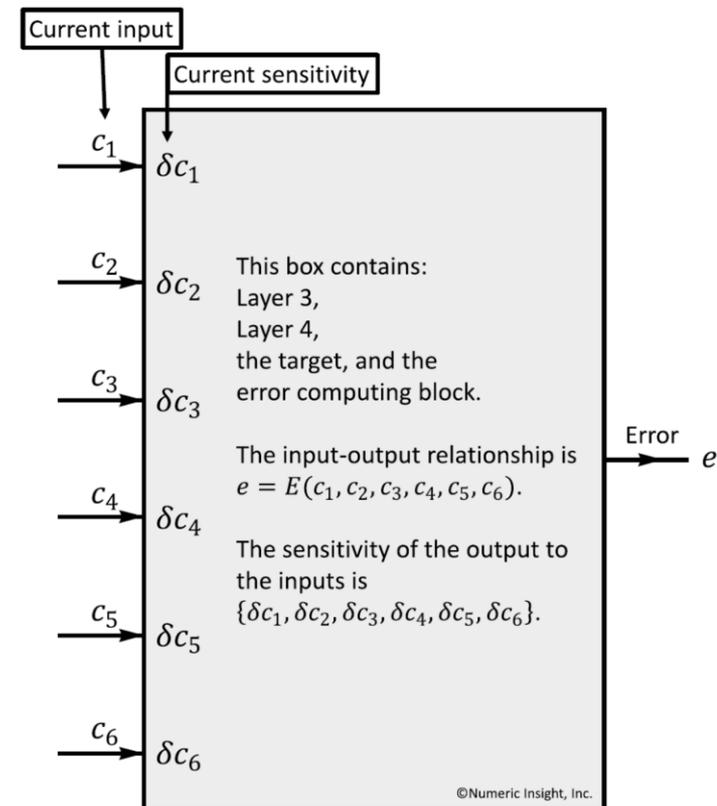
- Sensitivity:  $\{\delta c_1, \delta c_2, \delta c_3, \delta c_4, \delta c_5, \delta c_6\}$

current input changes  
by the small amount

expect the error  
change at the output

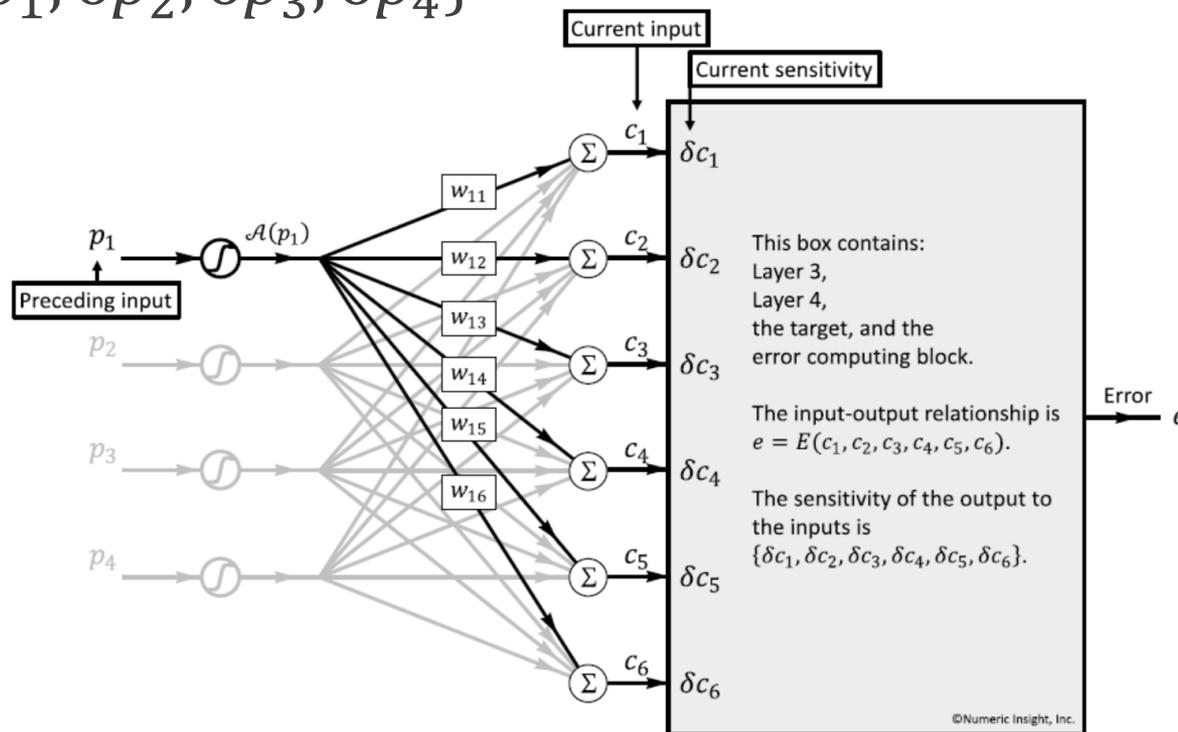
$$\Delta c_1 \text{ -----} \rightarrow \Delta c_1 \delta c_1$$

$$\Delta c_4 \text{ -----} \rightarrow \Delta c_4 \delta c_4$$



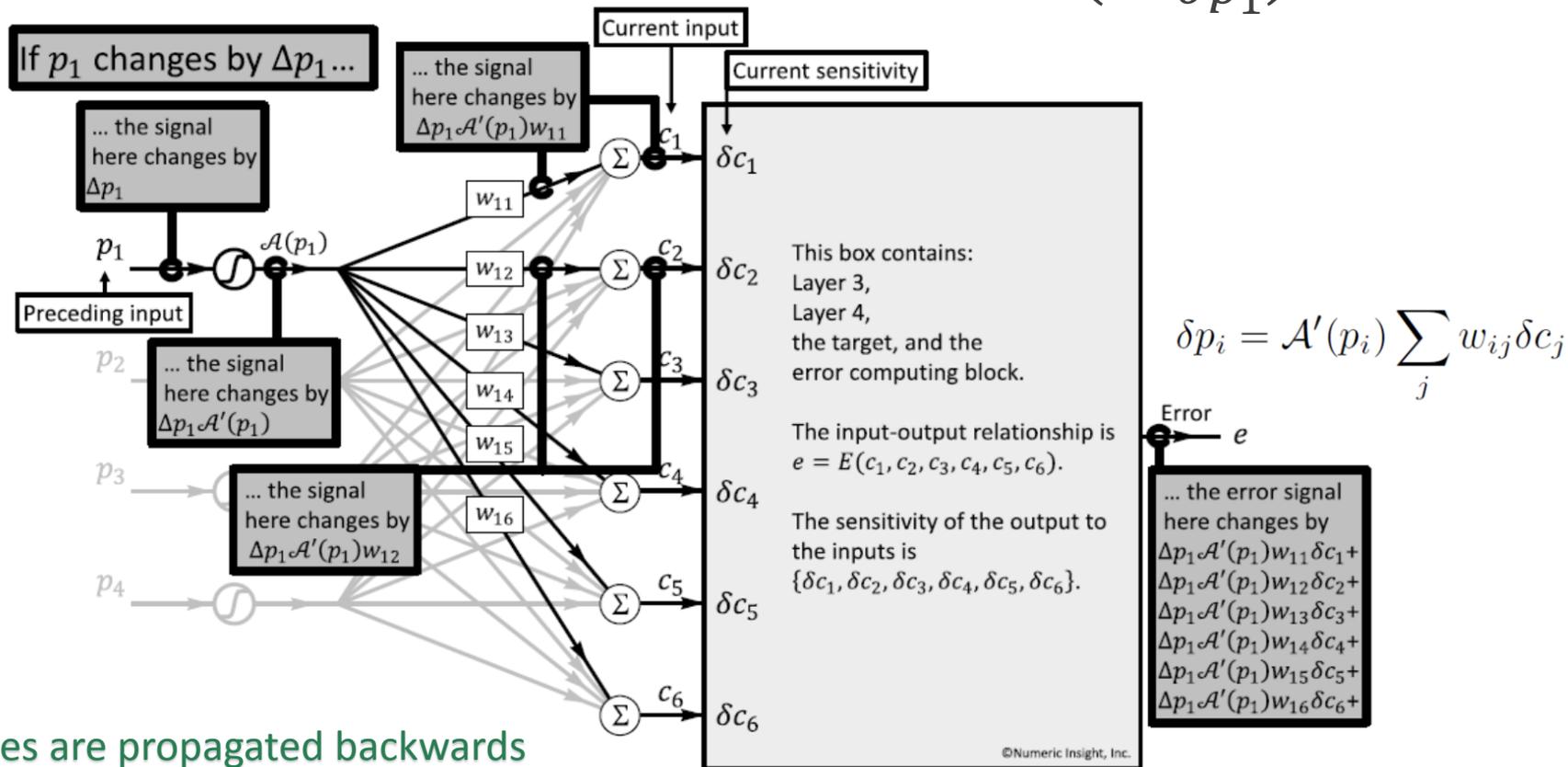
# Backpropagation (concept 2)

- Sensitivity of the preceding box
  - Layer 2 and all following layers
  - $\{\delta p_1, \delta p_2, \delta p_3, \delta p_4\}$



# Backpropagation (concept 2)

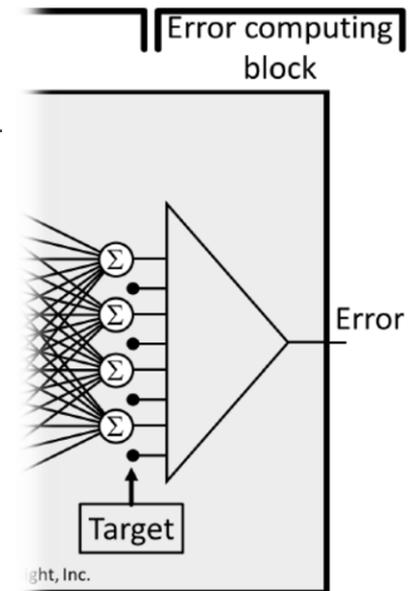
- Computing sensitivity component of  $\delta p_1 \left( = \frac{\partial E}{\partial p_1} \right)$



sensitivities are propagated backwards

# Backpropagation (concept 2)

- Computing sensitivity component of starting point
  - consider the error computing block itself
    - Input :  $\{o_1, o_2, o_3, o_4\}$
    - Output:  $e$
  - sensitivity of the error computing block
    - $\{2(o_1 - t_1), 2(o_2 - t_2), 2(o_3 - t_3), 2(o_4 - t_4)\}$

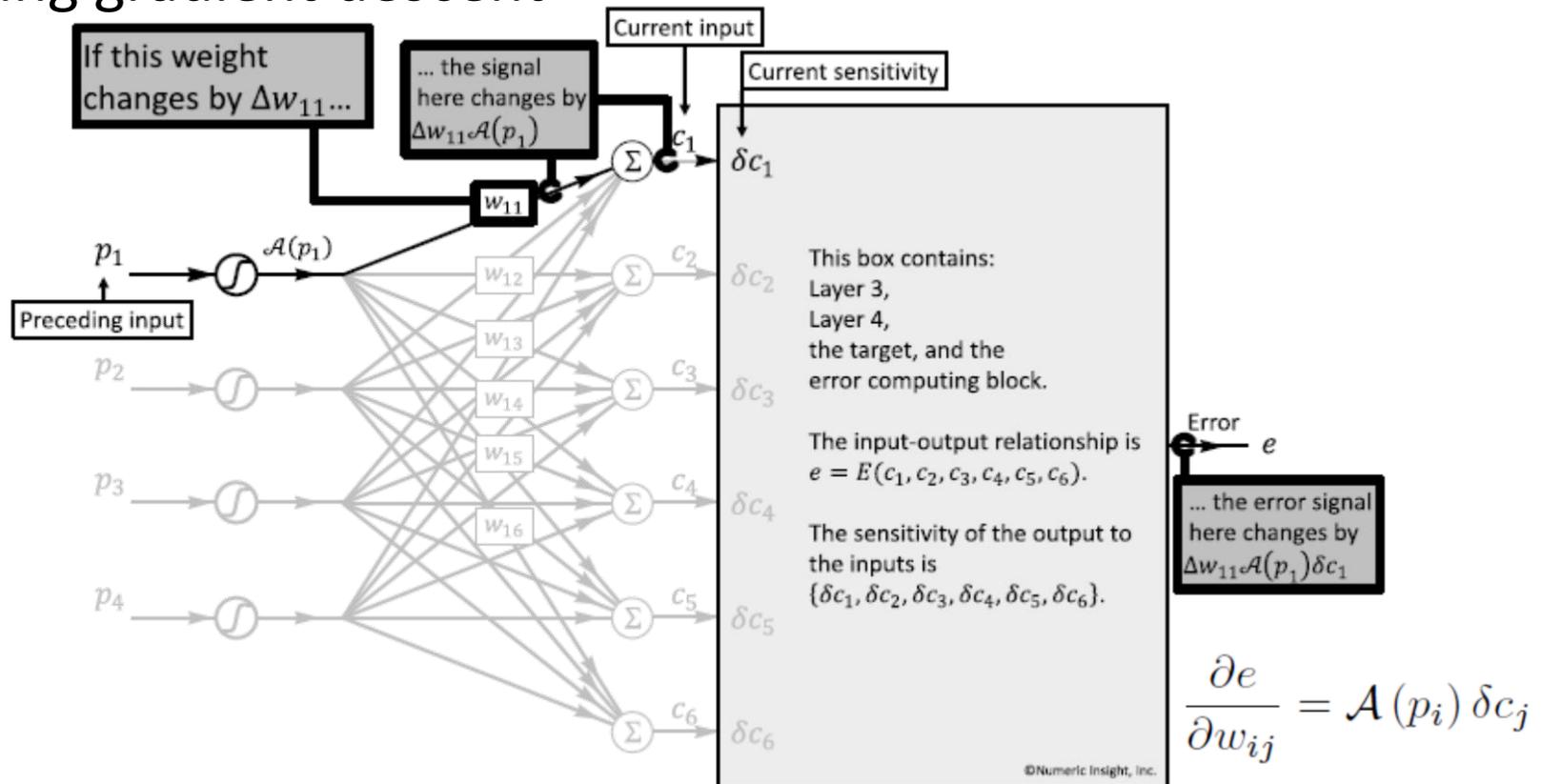


# OUTLINE

- Perceptron
- Stochastic Gradient Descent
- Backpropagation
  - Boxing
  - Sensitivity
  - **Weight updates**

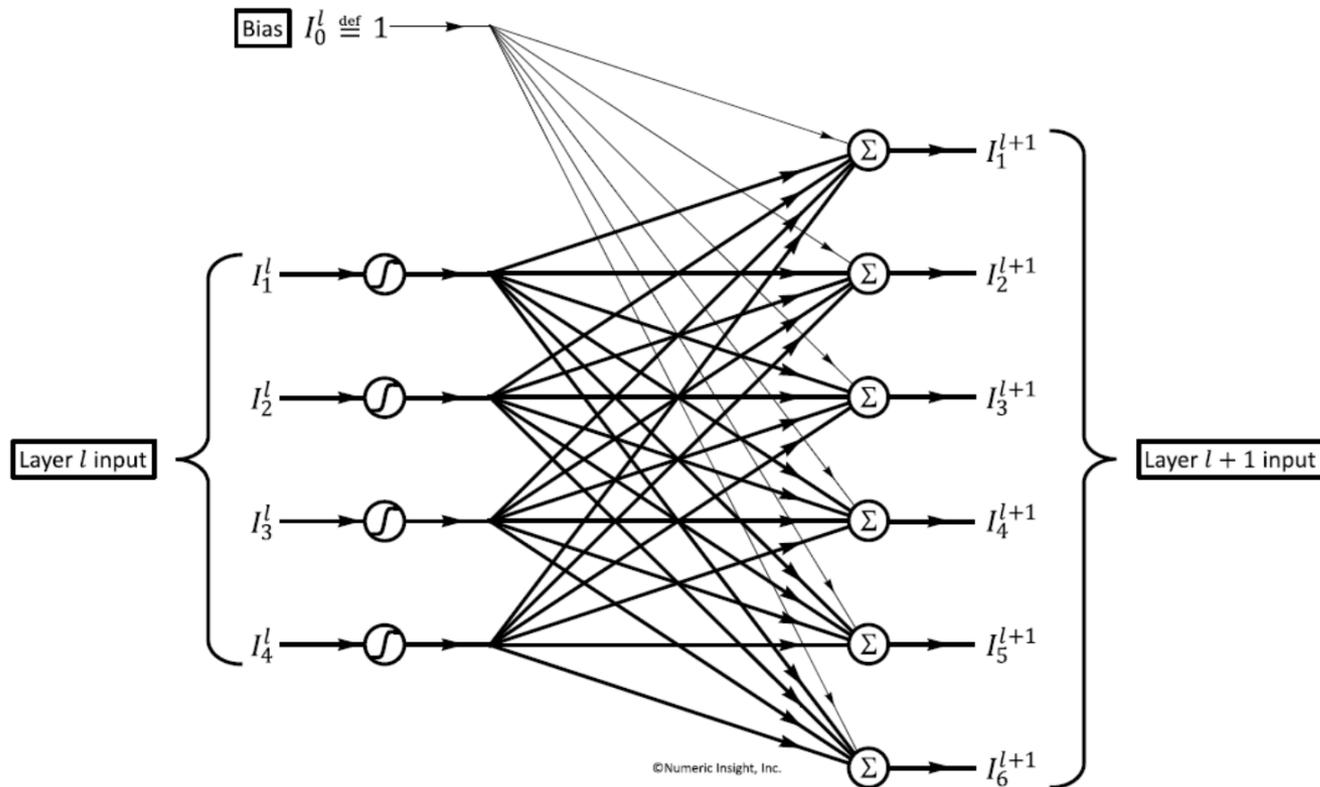
# Backpropagation (concept 3)

- **Weight updates**
  - Using gradient descent



# Backpropagation (Addition of bias)

- Use of bias generally results in a faster convergence of the training process.



# Backpropagation (Summary)

---

- **Step 1:** Initialize the weights and set the learning rate and the stopping criteria.
- **Step 2:** Randomly choose an input and the corresponding target.
- **Step 3:** Compute the input to each layer and the output of the final layer.
- **Step 4:** Compute the sensitivity components.
- **Step 5:** Compute the gradient components and update the weights.
- **Step 6:** Check against the stopping criteria. Exit and return the weights or loop back to Step 2.

# References

---

- MIT Machine Learning course ([Classification and Linear Classifiers](#))
- Sathyanarayana, Shashi. "A Gentle Introduction to Backpropagation." (2014).

امام حسین (ع):

إِنَّ حَوَائِجَ النَّاسِ إِلَيْكُمْ مِنْ نِعَمِ اللَّهِ عَلَيْكُمْ، فَلَا تَمَلُّوا النُّعْمَ.

در خواست‌های مردم از شما از نعمت‌های خدا بر شماست، پس از این نعمت‌ها خسته و ملول نشوید.

The fact that people need you is one of the blessings of Allah upon you. Therefore, do not feel any grievance about it.

بحار الأنوار، ج ۷۴، ص ۳۱۸

