BÀI 10
Mạng Neuron Nhân Tạo

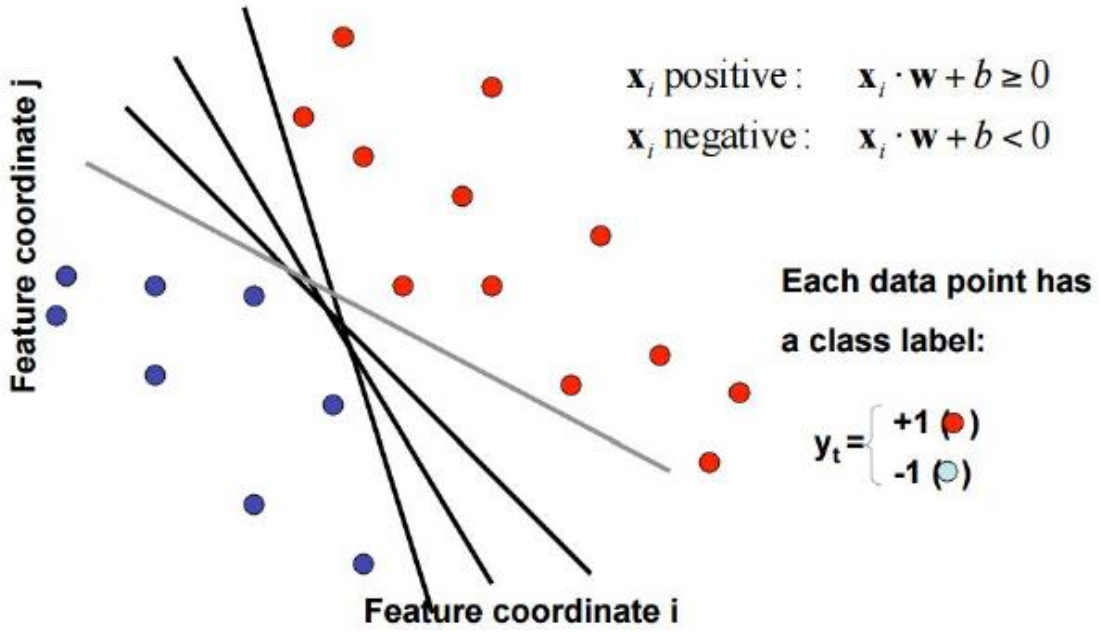
Nội dung buổi học

- Neural Network
  - ➢Bộ phân lớp tuyến tính
  - ➢Multiple Perceptron: Neural network
  - ➢Huấn luyện mạng với gradient descent
- Bài toán nhận dạng chữ số viết tay
  - Bộ dữ liệu MNIST
  - Định dạng dữ liệu
  - Phát biểu bài toán
  - Xây dựng mạng và thử nghiệm

Neural Network

Phân lớp tuyến tính (Linear Classification)

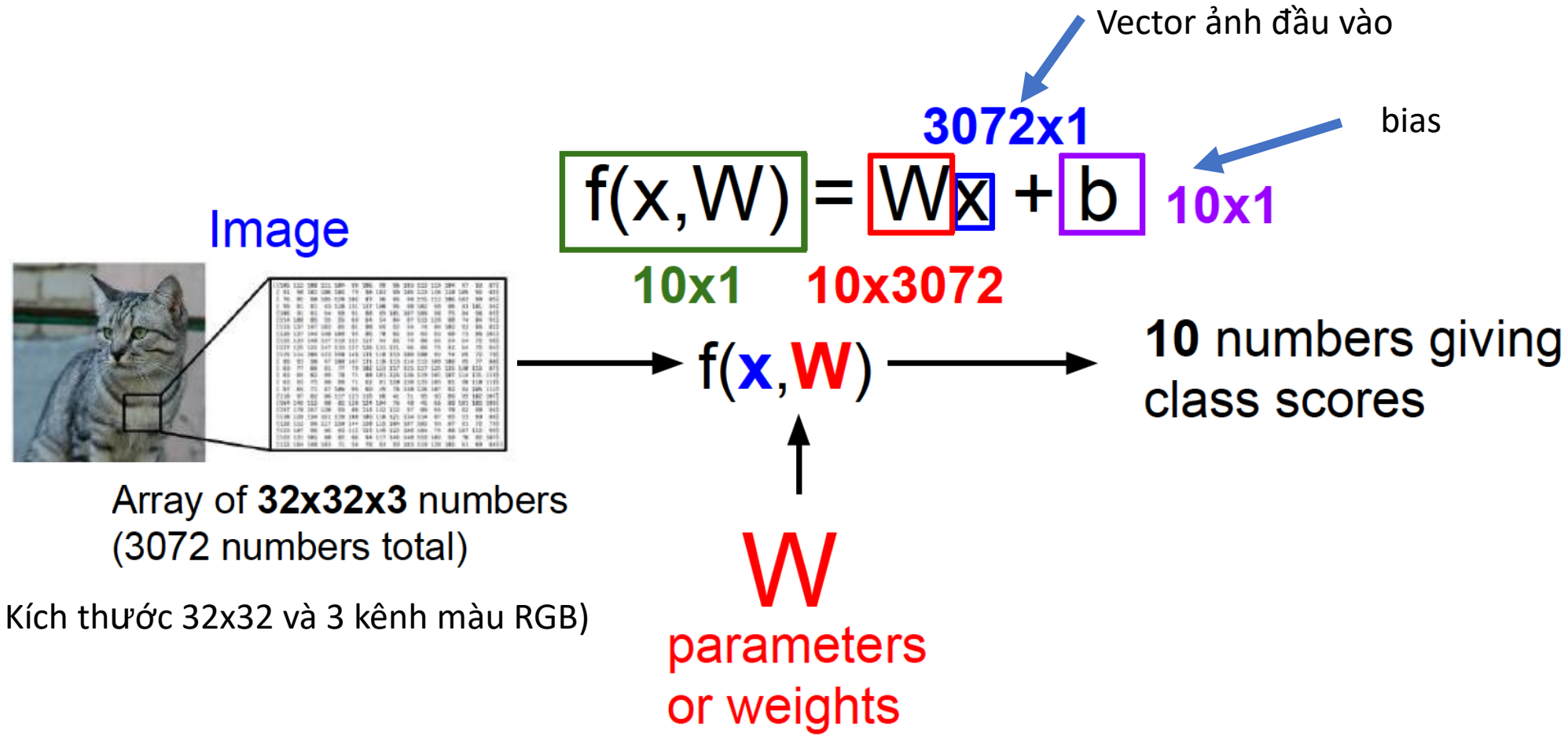
- Tìm một đường thẳng phân tách giữa hai lớp positive và negative



Phân lớp tuyến tính (Linear Classification)

Ví dụ với phân loại ảnh CIFAR-10

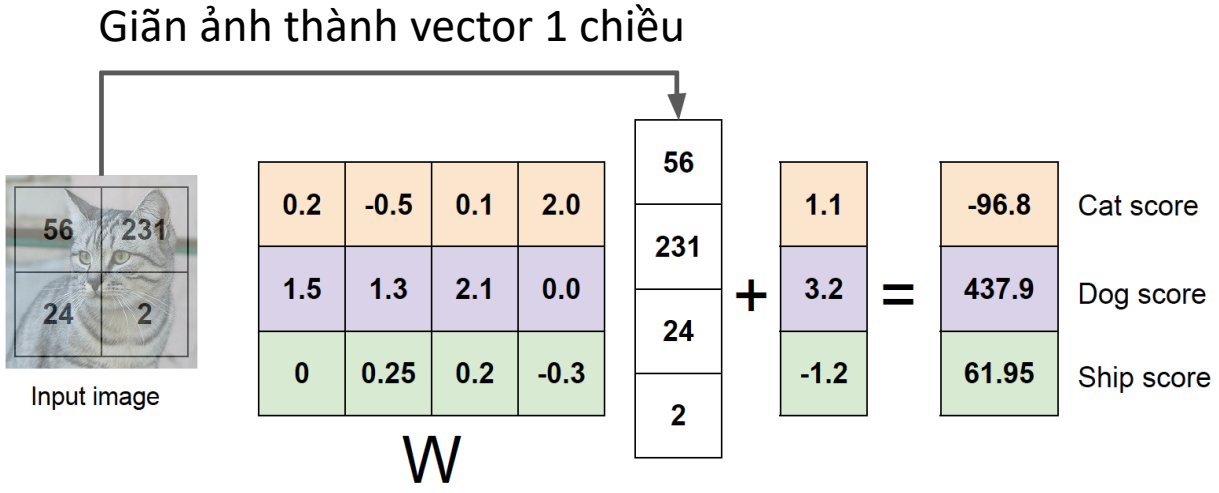
Phân lớp tuyến tính (Linear Classification)

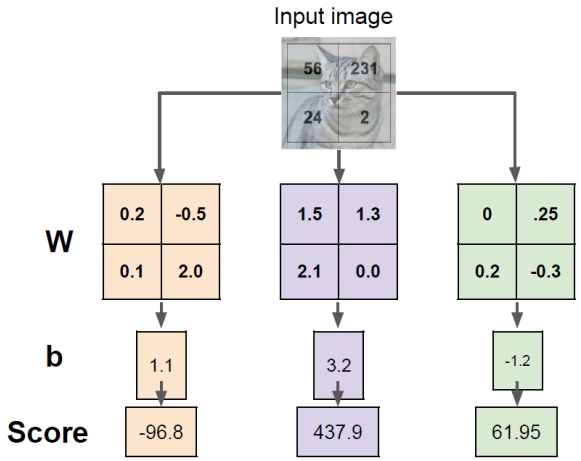


Kích thước 32x32 và 3 kênh màu RGB)

Linear Classification

- Ví dụ với phân loại ảnh có 4 pixels và 3 lớp đối tượng (cat/dog/ship)

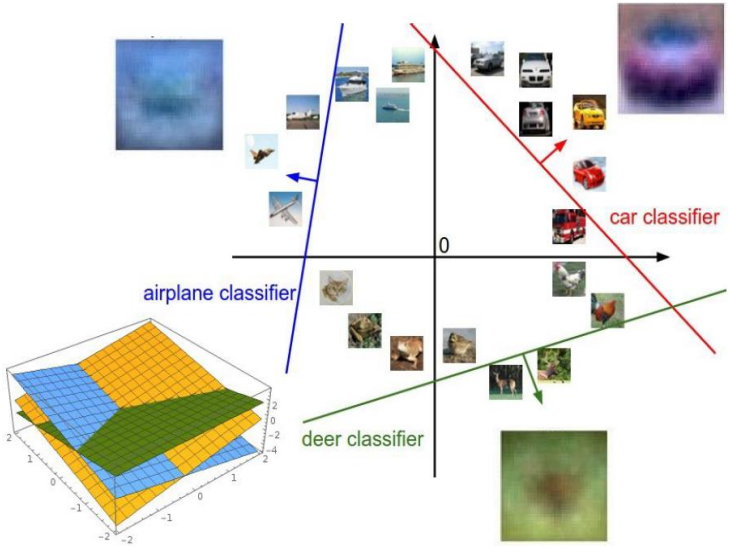




$$f(x,W) = Wx + b$$

Linear Classification

- Trực quan hóa bộ phân lớp với các đường phân tách



$$f(x,W) = Wx + b$$

Array of **32x32x3** numbers
(3072 numbers total)

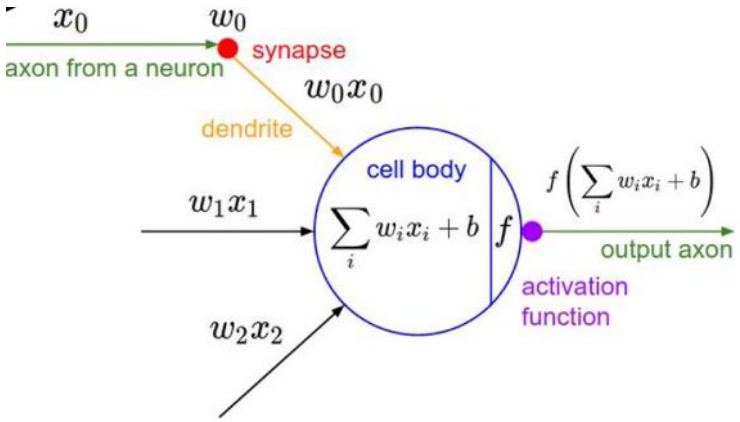
Mạng Neuron (Neural Network)

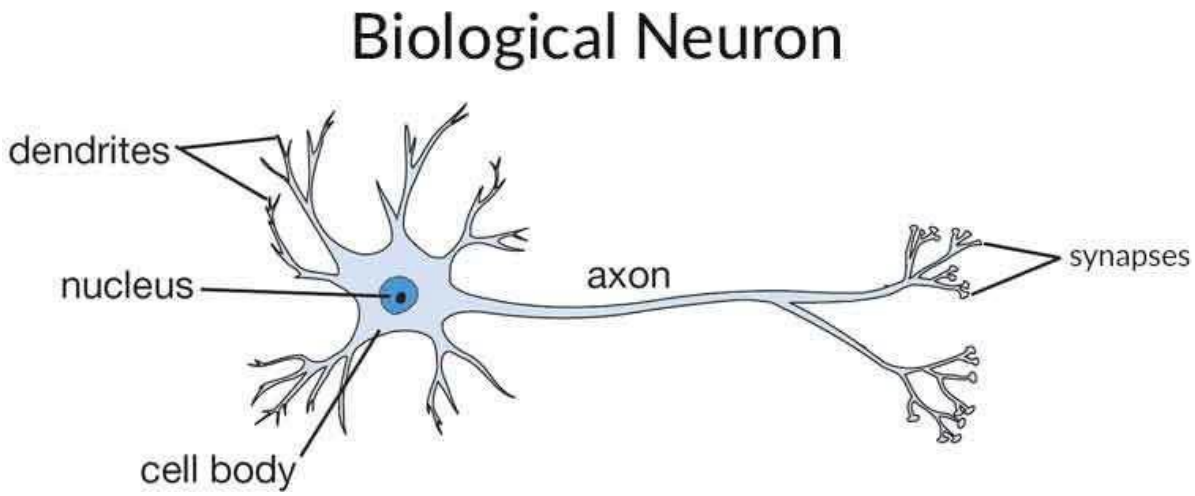
Linear classification

$$\text{output} = \begin{cases} 0 & \text{if } w \cdot x + b \leq 0 \\ 1 & \text{if } w \cdot x + b > 0 \end{cases}$$

Output = sign(Wx+b)

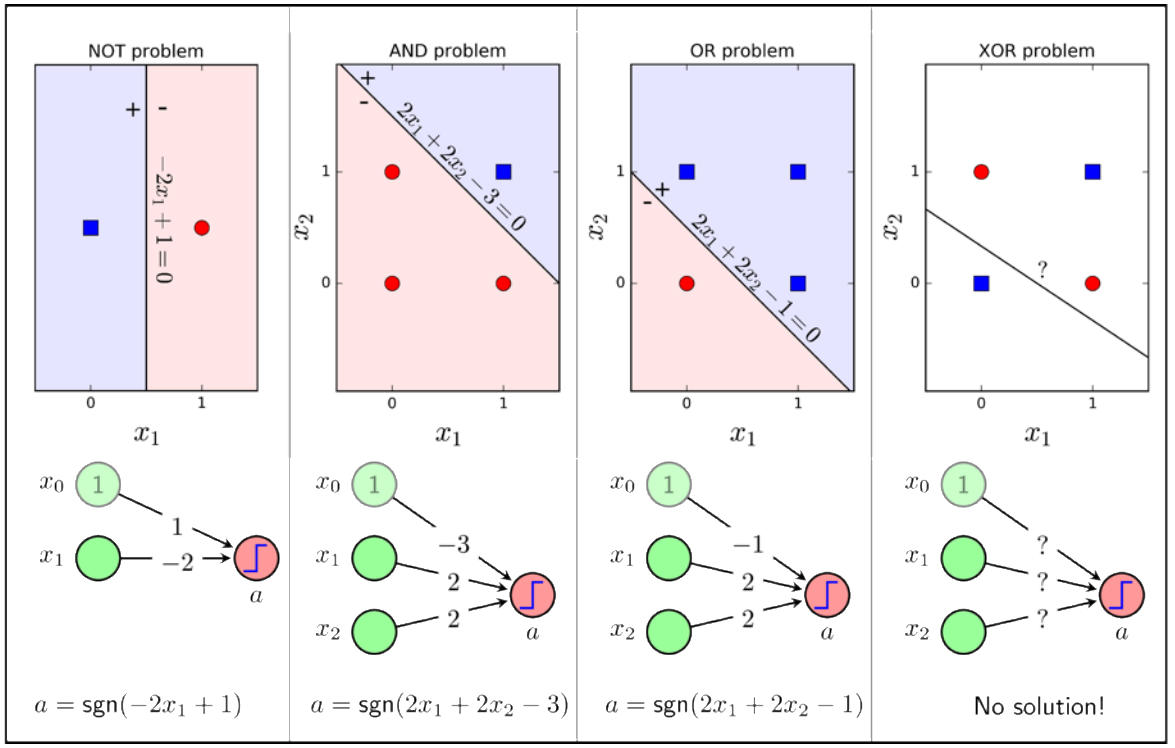
Perceptron





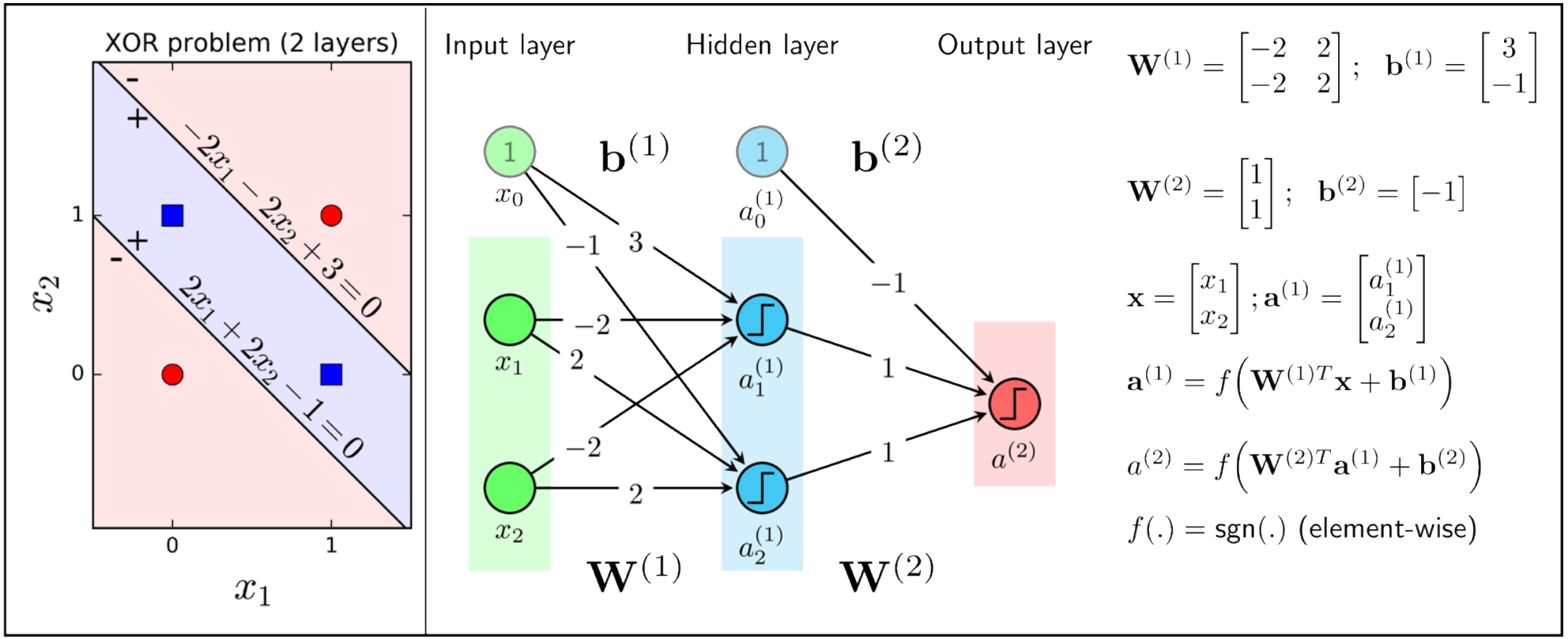
Mạng Neuron (Neural Networks)

- Vấn đề của Linear Classification
  - Một số phân bố dữ liệu có thể phân tách bằng các đường thẳng
  - Vấn đề với XOR: phân tách bằng non-linearly



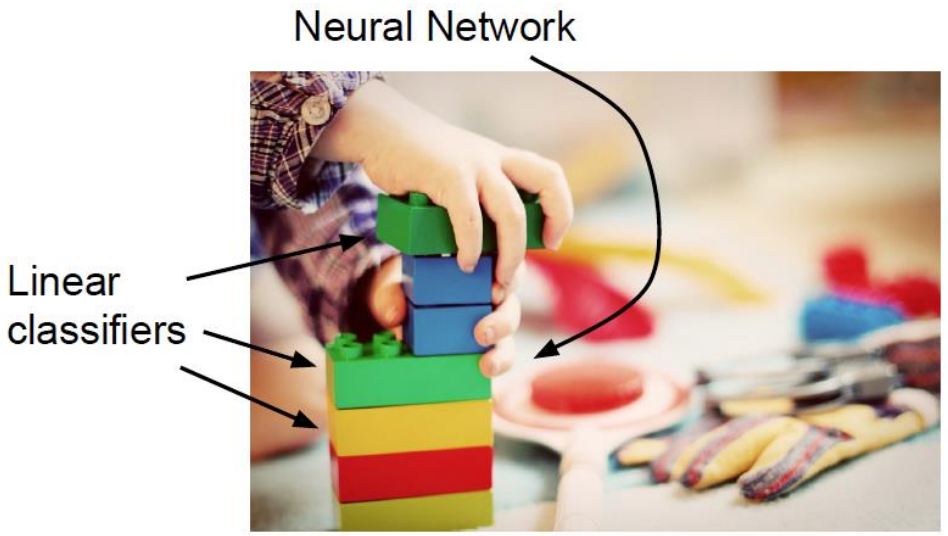
Neural Network

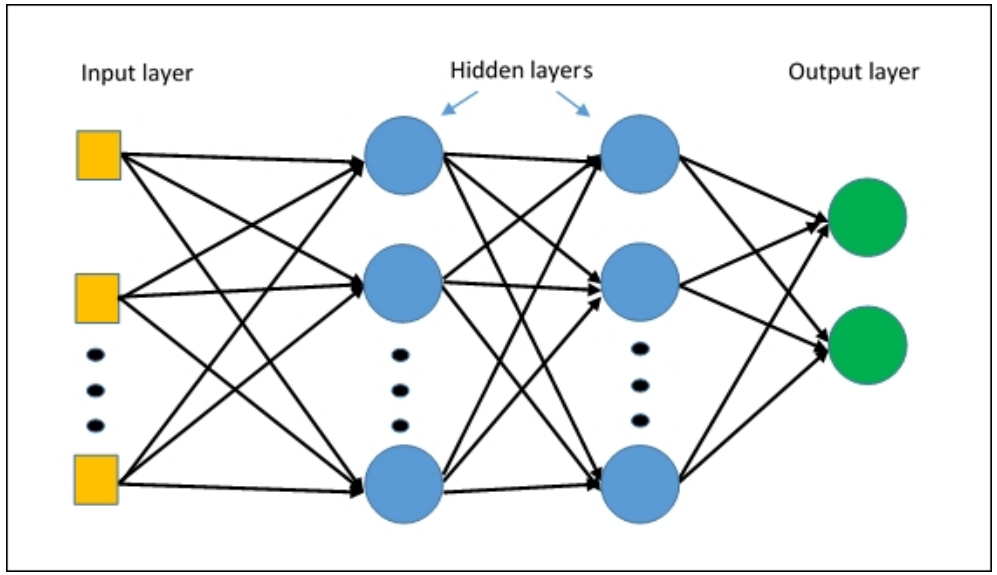
- Multiple perceptron



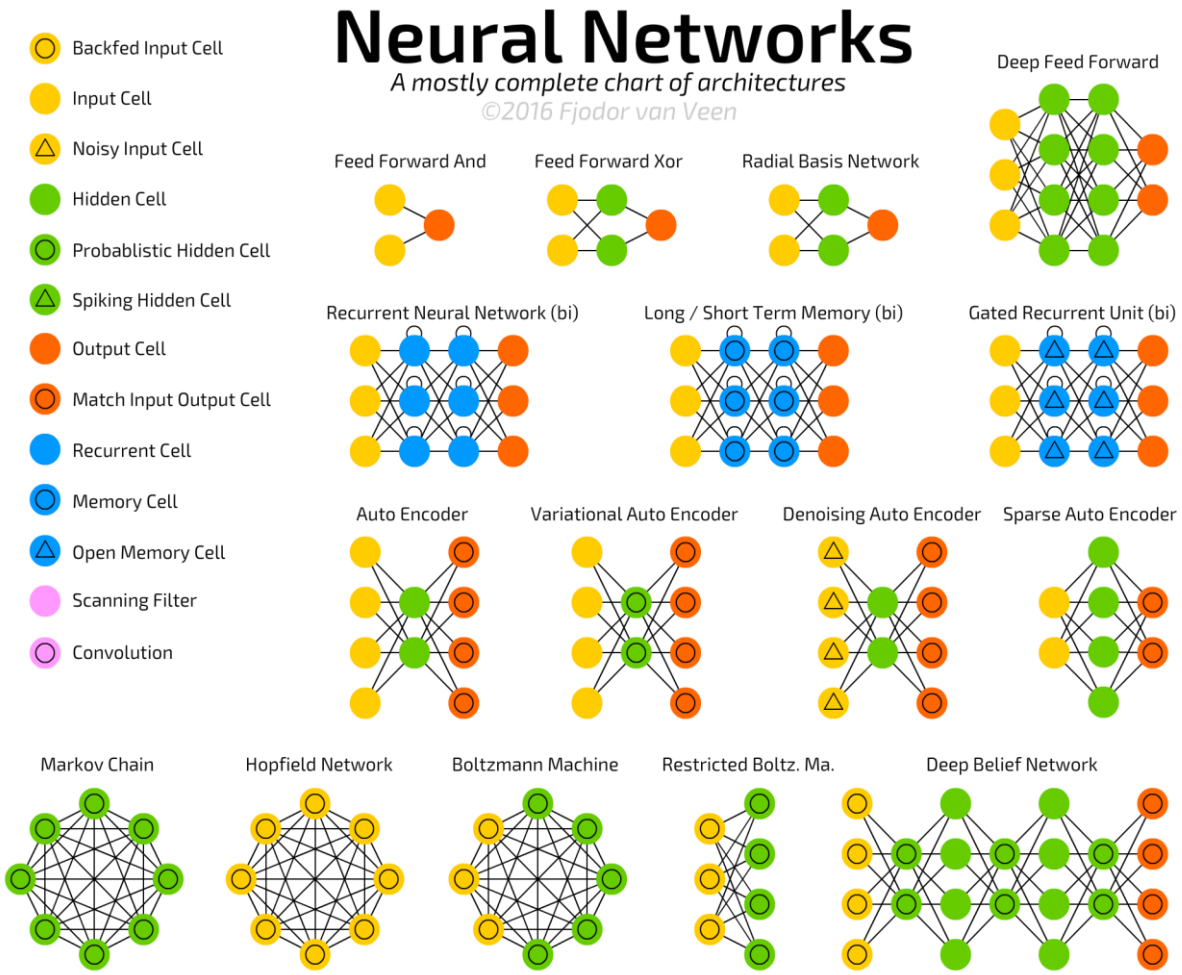
Neural Network

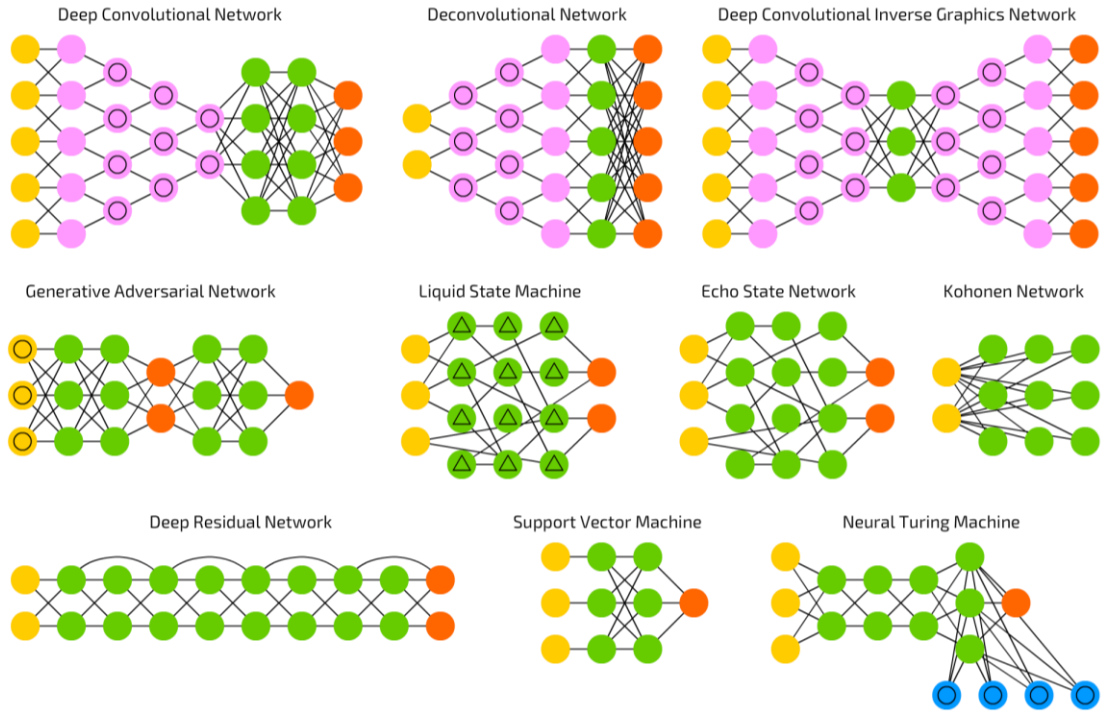
- Multi-layers Perceptron
  - Layers: Chơi xếp hình với các kiểu kết nối
  - Cấu trúc mạng: Thiết kế các lớp và các kết nối





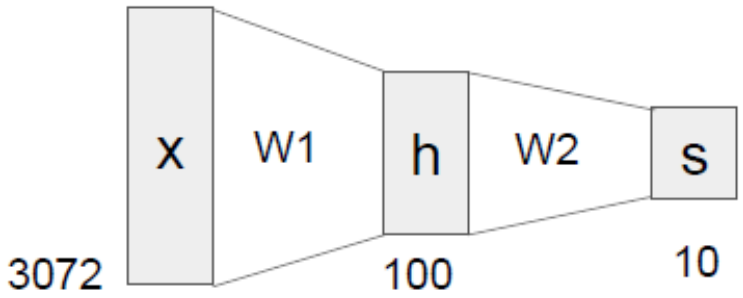
Các kiến trúc kết nối





Multi-perceptron

- Mạng neuron 1 lớp ẩn $f = Wx$
- Mạng neuron 2 lớp ẩn $f = W_2 \max(0, W_1 x)$



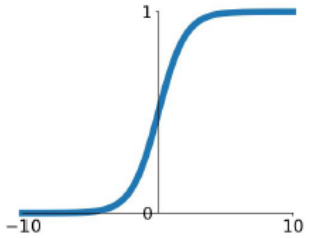
Hàm kích hoạt (Activate function)

- Mạng neuron 3 lớp ẩn $f = W_3 \max(0, W_2 \max(0, W_1 x))$

Hàm kích hoạt (Activate functions)

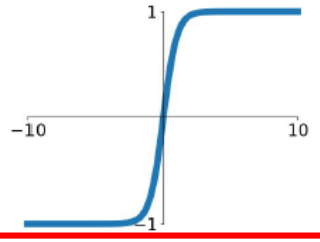
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



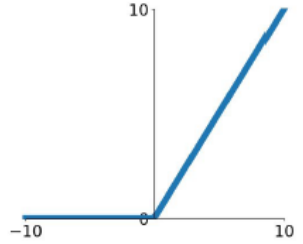
tanh

$$\tanh(x)$$



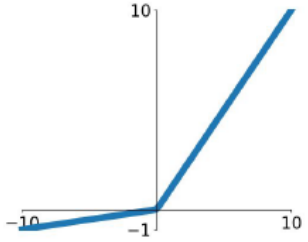
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

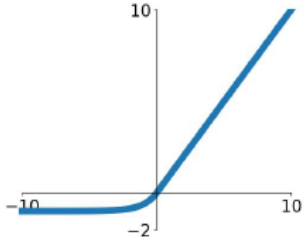


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

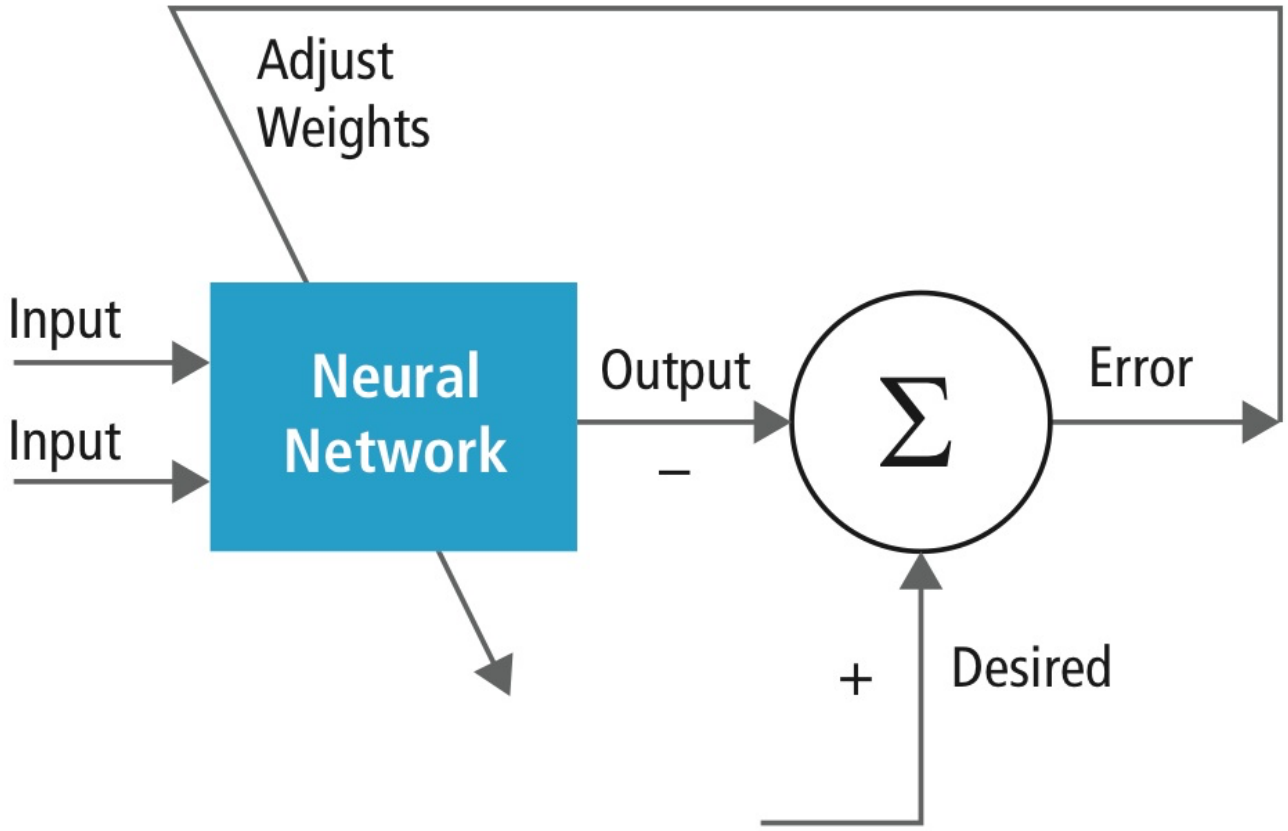
ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Thường sử dụng thực tế

Huấn luyện mạng



Huấn luyện mạng

- Định nghĩa hàm mất mát (**loss function**): định lượng về độ lệch của dự đoán đối với nhãn thực của tập dữ liệu huấn luyện
- Nhiệm vụ: tìm bộ tham số sao cho loss function là nhỏ nhất có thể (tối ưu hóa – **optimization**)

airplane	-3.45	-0.51	3.42
automobile	-8.87	**6.04**	4.64
bird	0.09	5.31	2.65
cat	**2.9**	-4.22	5.1
deer	4.48	-4.19	2.64
dog	8.02	3.58	5.55
frog	3.78	4.49	**-4.34**
horse	1.06	-4.37	-1.5
ship	-0.36	-2.09	-4.79
truck	-0.72	-2.93	6.14

Loss Function

- Bài toán: Tập huấn luyện gồm 3 ảnh với 3 lớp đối tượng
- Sử dụng parameter W với hàm $f(x,W)=Wx$



cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Losses:	2.9	0	12.9

- Multiple SVM loss
- Tập huấn luyện (x_i, y_i) với
  - x_i là dữ liệu vector ảnh
  - y_i là nhãn
- Loss tổng:

$$L = \frac{1}{N}\sum_{i=1}^{N} L_i$$

L = (2.9 + 0 + 12.9)/3
= **5.27**

Softmax classifier

- Chuẩn hóa từ score sang miền xác suất

$$s = f(x_i; W)$$

$$P(Y = k|X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

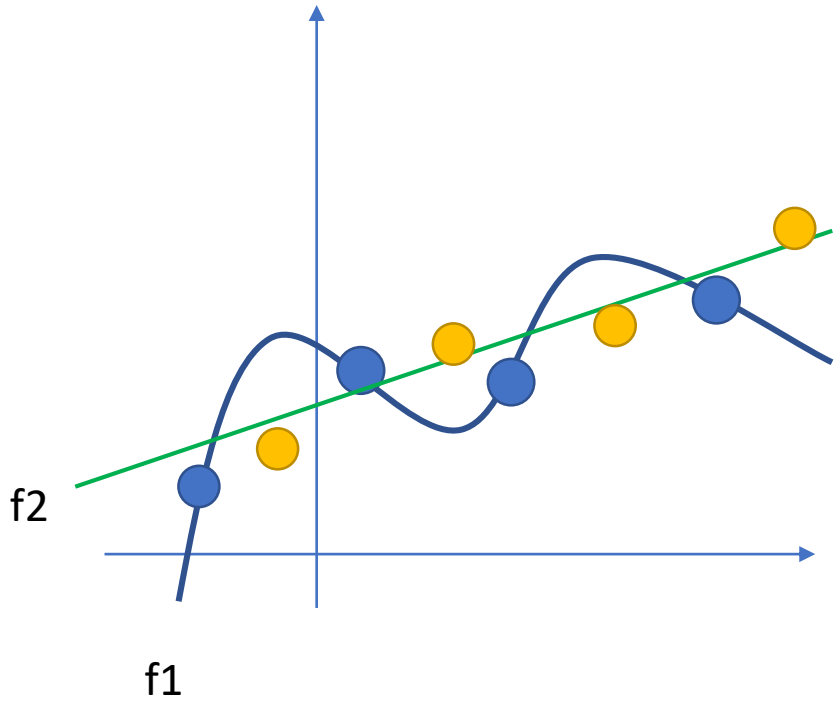
Softmax Function

	Score		Xác suất đoán (P)		Xác suất đúng (Q)
cat	**3.2**	→	**0.13**		**1.00**
car	5.1		0.87		0.00
frog	-1.7		0.00		0.00

So sánh

Cross Entropy H(P,Q)

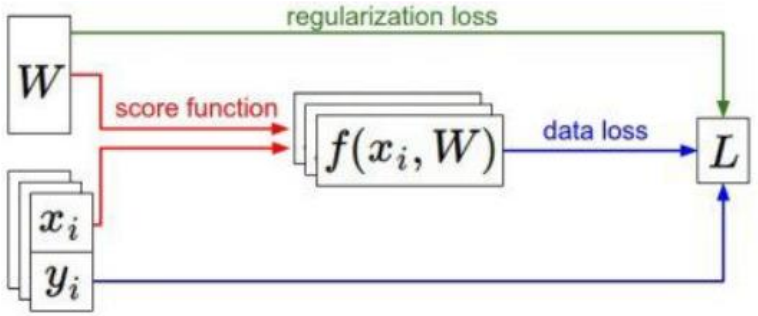
Regularization



$$L(W) = \frac{1}{N}\sum_{i=1}^{N} L_i(f(x_i, W), y_i) + \lambda R(W)$$

Data loss:
Dự đoán từ mô hình

Regularization:
Nhằm tránh mô hình biểu diễn quá tốt trên dữ liệu huấn luyện



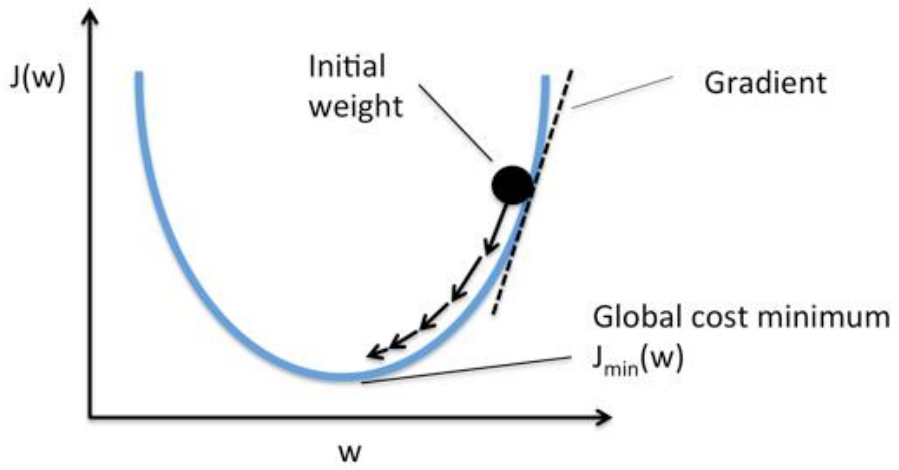
Tham khảo

L2 regularization: $R(W) = \sum_k \sum_l W_{k,l}^2$

L1 regularization: $R(W) = \sum_k \sum_l |W_{k,l}|$

Elastic net (L1 + L2): $R(W) = \sum_k \sum_l \beta W_{k,l}^2 + |W_{k,l}|$

Gradient Descent

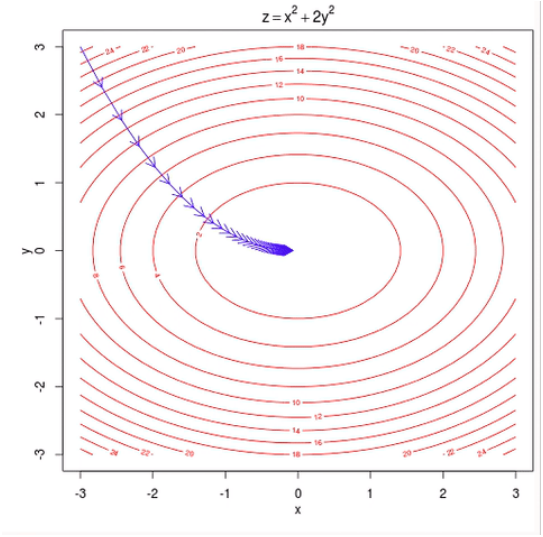


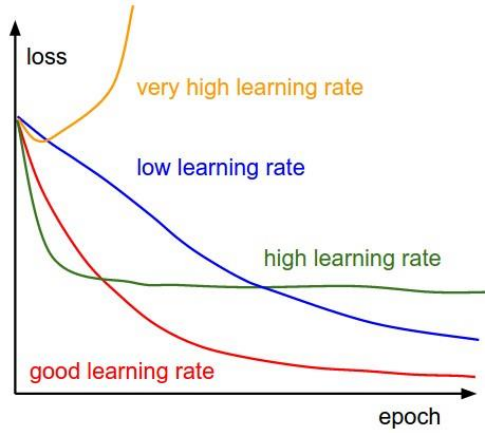
Algorithm: Gradient Descent

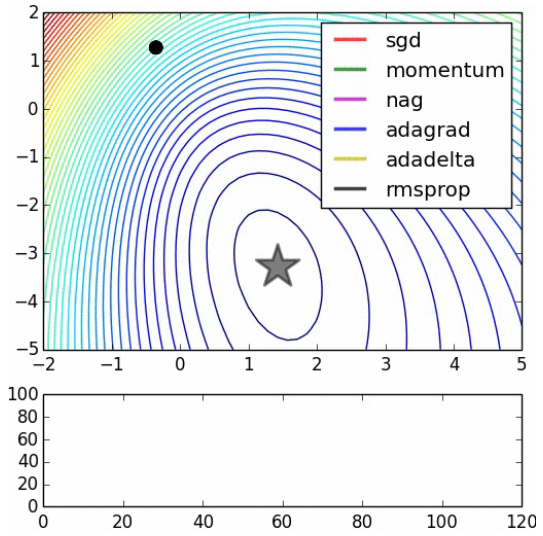
Input: $Y, \Theta, \mathbf{X}, \alpha$, tolerance, max iterations
Output: $\mathbf{\Theta}$

```
for i = 0; i < max iterations; i++ do
    current cost = Cost(Y, X, Θ)
    if current cost < tolerance then
        break
    else
        gradient = Gradient(Y, X, Θ)
        θ_j ← θ_j − α · gradient
```

Learning rate: bước nhảy dài hay ngắn





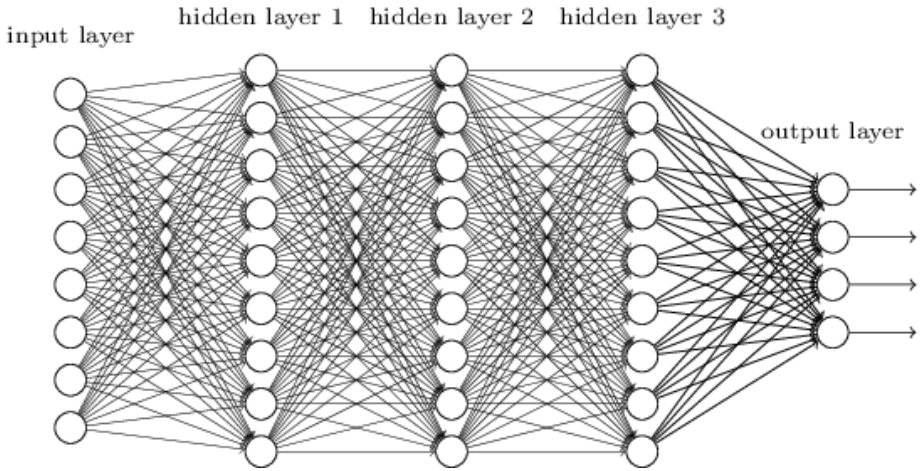


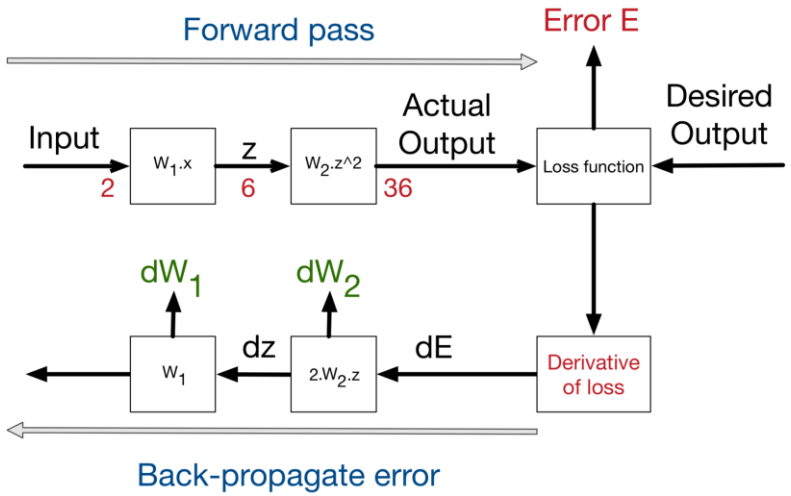
- Batch SGD: thay vì input từng ảnh, input một tập ảnh
- Minibatch: dùng batch nhỏ để huấn luyện

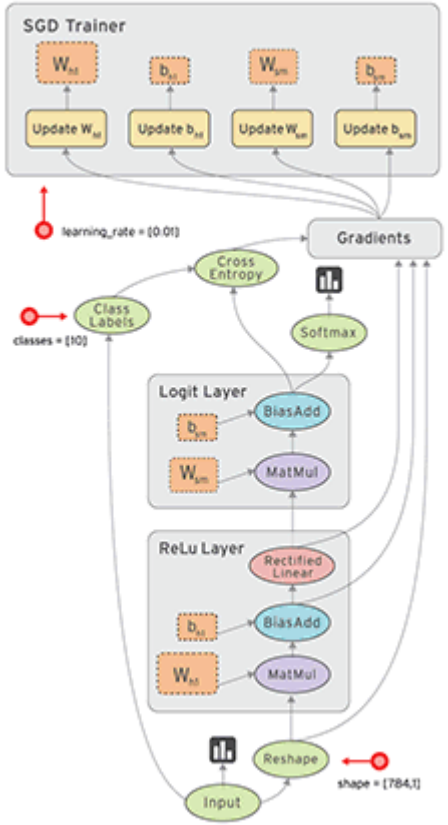
Lan truyền ngược (Back propagation)

- Tính gradient cho từng tham số w_i với mạng nhiều lớp

  (Có thể coi là hộp đen nếu bạn chưa muốn tìm hiểu ngay)







Ưu nhược điểm của Neural Networks

- Dễ dàng xây dựng được mô hình biểu diễn, tính toán
- Mô hình đa dạng, có thể sử dụng trong nhiều bài toán khác nhau (phân lớp, phân cụm, phân tích chuỗi)
- Sử dụng trong nhiều ứng dụng khác nhau: ảnh, text, voice …

- Vấn đề hộp đen (blackbox): không giải thích được, không suy diễn được
- Cần nhiều dữ liệu tính toán
- Huấn luyện mô hình phức tạp

BÀI TOÁN PHÂN LOẠI CHỮ VIẾT TAY

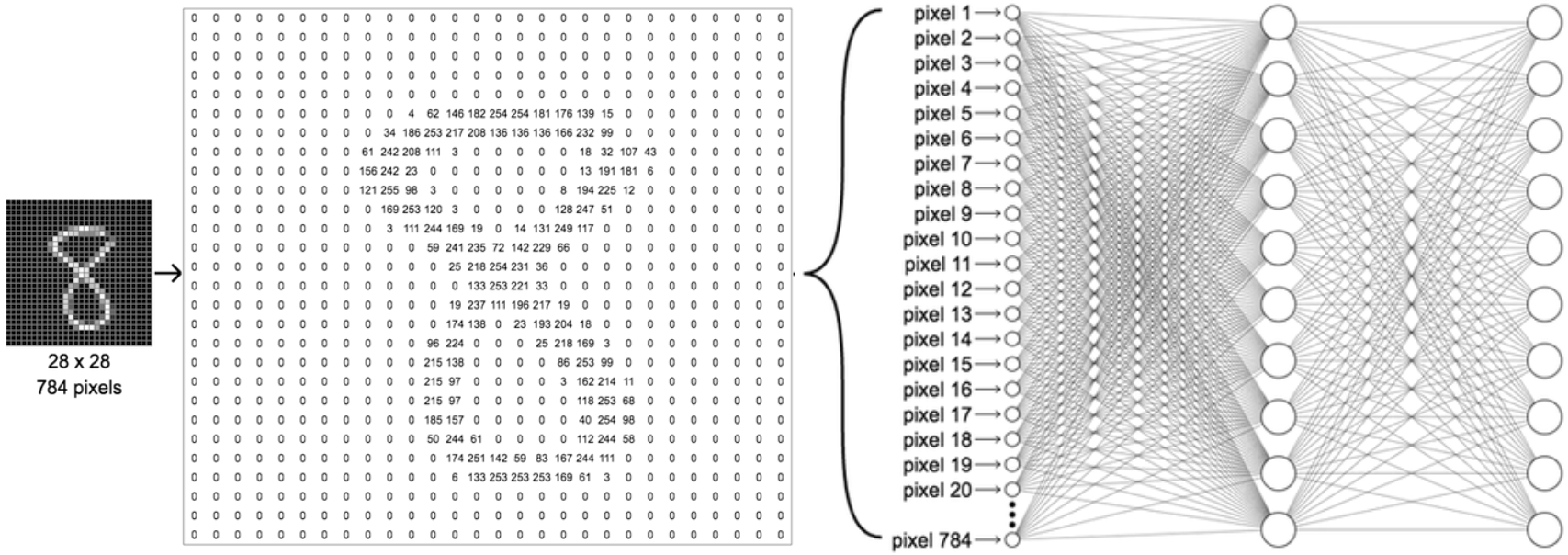
Bài toán phân loại chữ viết tay

- Dữ liệu: Bộ dữ liệu MNIST
- http://yann.lecun.com/exdb/mnist/

- 60,000 ảnh cho huấn luyện
- 10,000 ảnh cho kiểm thử mô hình
- Mọi ảnh chữ số viết tay trong tập dữ liệu được chuẩn hóa về kích thước, cụ thể là (28x28) với giá trị pixels nằm trong khoảng 0 đến 1

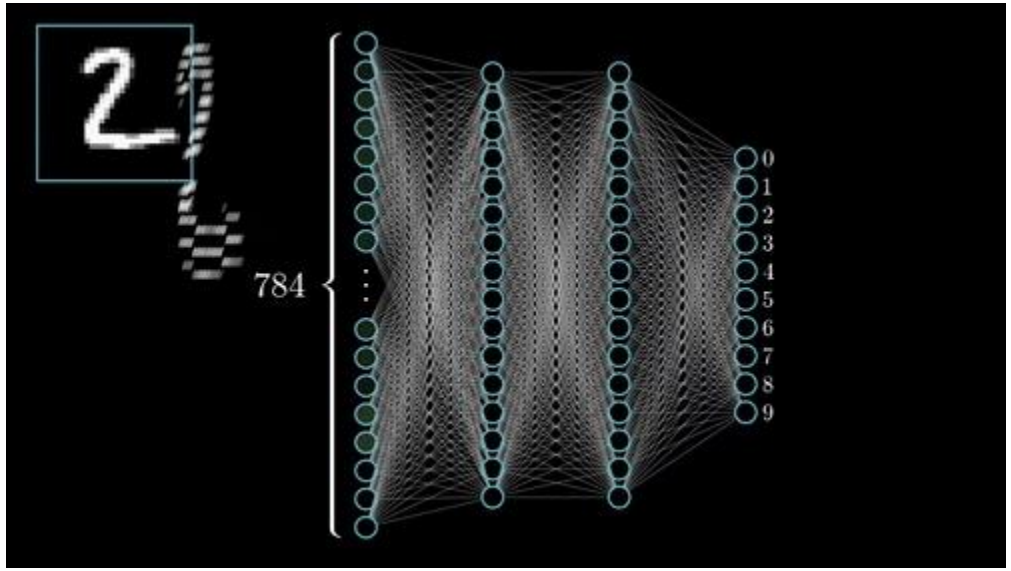
Dữ liệu đầu vào cho mạng neuron

- Vector đầu vào: 784 feature với các số trong khoảng từ 0-255
- Nhãn từ 0 đến 9



Xây dựng mô hình

- Input: vector 784 features
- Hai lớp ẩn với hàm kích hoạt là ReLU
- Lớp output với softmax classifier



Keras

```
# build model
from keras.layers import Dense, Flatten
from keras.models import Sequential

model = Sequential()
model.add(Flatten(input_shape=(28, 28)))
model.add(Dense(n_hidden_1, activation='relu'))  # hidden layer1
model.add(Dense(n_hidden_2, activation='relu'))  # hidden layer2
model.add(Dense(num_classes, activation='softmax'))  # output layer
```

Huấn luyện mô hình

- Hàm mất mát (loss function): cross entropy
- Cập nhật tham số bằng Stochastic Gradient Desent
- Các siêu tham số:
  - learning_rate = 0.1
  - num_epoch = 10
  - batch_size = 128

```
# loss, optimizers
model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.SGD(lr=learning_rate),
              metrics=['accuracy'])

model.fit(x_train, y_train, batch_size=batch_size, epochs=num_epoch)
```

- Tham khảo các bước còn lại trên github

Bài tập

- Quan sát và làm lại mẫu bài tập trên website
- Tìm và thay thế các phần thiếu hoặc thay thế các tham số, các phương pháp huấn luyện
- Thử với các bài toán trong các buổi học trước

Q&A

Thank you!