# Machine Learning

PEI-YIN, TIEN
installbtien

February 2021

## Contents

# 1 Regression

## 1.1 Step 1: Model

- $y_{data} = b + w * x_{data}$

## 1.2 Step 2: Goodness of Function

- $L(f) = \Sigma(y_{data} - f(x_{data}))^2$

  $L(f)$: Loss function, input: a function, output: how bad it is
  $f(x_{data})$: estimated y based on the input function

- $L(w, b) = \Sigma(y_{data} - (b + w * x_{data}))^2$

## 1.3 Step 2: Regularization

$$y = b + \sum w_i x_i$$

- Redesign the loss function:

  $$L = \sum_n (y^n - (b + \sum w_i x_i))^2 + \lambda \sum (w_i)^2$$

  The functions with smaller $w_i$ are better

- The smaller $w_i$ is, the **smoother** a function is

  $$y + w_i \Delta x_i = b + \sum w_i (x_i + \Delta x_i)$$

- If some noises corrupt input $x_i$ when testing, a smoother function has less influence

## 1.4 Step 3: Gradient Descent

- Consider loss function $L(f)$ with one parameter w:

  1. (Randomly) Pick an initial value $w^0$
  2. Compute $\frac{dL}{dw}|_{w=w^0}$

     $w^1 \leftarrow w^0 - \eta \frac{dL}{dw}|_{w=w^0}$ , $\eta$: learning rate
  3. Compute $\frac{dL}{dw}|_{w=w^1}$

     $w^2 \leftarrow w^1 - \eta \frac{dL}{dw}|_{w=w^1}$

     ...many iteration

- How about two parameters? $w^*, b^* = arg\ minL(w, b)$

  1. (Randomly) Pick an initial value $w^0, b^0$
  2. Compute $\frac{\partial L}{\partial w}|_{w=w^0,b=b^0}$, $\frac{\partial L}{\partial b}|_{w=w^0,b=b^0}$

     $w^1 \leftarrow w^0 - \eta \frac{\partial L}{\partial w}|_{w=w^0,b=b^0}$

     $b^1 \leftarrow b^0 - \eta \frac{\partial L}{\partial b}|_{w=w^0,b=b^0}$
  3. Compute $\frac{\partial L}{\partial w}|_{w=w^1,b=b^1}$, $\frac{\partial L}{\partial b}|_{w=w^1,b=b^1}$

     $w^2 \leftarrow w^1 - \eta \frac{\partial L}{\partial w}|_{w=w^1,b=b^1}$

     $b^2 \leftarrow b^1 - \eta \frac{\partial L}{\partial b}|_{w=w^1,b=b^1}$

     ...many iteration

- Formulation of $\frac{\partial L}{\partial w}, \frac{\partial L}{\partial b}$

  $\frac{\partial L}{\partial w} = \Sigma 2(y_{data} - (b + w * x_{data}))(-x_{data})$

  $\frac{\partial L}{\partial b} = \Sigma 2(y_{data} - (b + w * x_{data}))$

- **Tip 1**: Tuning your learning rates

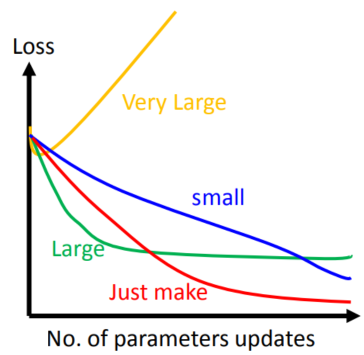



Figure 1: Set the learning rate $\eta$ carefully

Adaptive learning rates:

E.g. 1/t decay: $\eta^t = \eta/\sqrt{t+1}$

**Adagrad**: $w^{t+1} \leftarrow w^t - \frac{\eta}{\sqrt{\Sigma(g^i)^2}} g^t$, g: gradient

$[w^{t+1} \leftarrow w^t - \frac{\eta^t}{\sigma^t} g^t]$, $\sigma$: root mean square of the previous derivatives of parameter w

$w^1 \leftarrow w^0 - \frac{\eta^0}{\sigma^0} g^0 \quad \sigma^0 = \sqrt{(g^0)^2}$

$w^2 \leftarrow w^1 - \frac{\eta^1}{\sigma^1} g^1 \quad \sigma^1 = \sqrt{\frac{(g^0)^2 + (g^1)^2}{2}}$

$w^3 \leftarrow w^2 - \frac{\eta^2}{\sigma^2} g^2 \quad \sigma^2 = \sqrt{\frac{(g^0)^2 + (g^1)^2 + (g^2)^2}{3}}$

...

$w^{t+1} \leftarrow w^t - \frac{\eta^t}{\sigma^t} g^t \quad \sigma^t = \sqrt{\frac{1}{t+1} \Sigma(g^i)^2} \quad \eta^t = \frac{\eta}{\sqrt{t+1}}$

$\Rightarrow w^{t+1} \leftarrow w^t - \frac{\eta}{\sqrt{\Sigma(g^i)^2}} g^t$

- **Tip 2**: Stochastic Gradient Descent (make the training faster)

  Loss is the summation over all training examples: $\sum_n (y^n - (b + \Sigma w_i x_i^n))^2$

  Pick an example $x^n$:

  $L^n = (y^n - (b + \sum_i w_i x_i^n))^2$

  $\theta^i = \theta^{i-1} - \eta \nabla L^n(\theta^{i-1})$

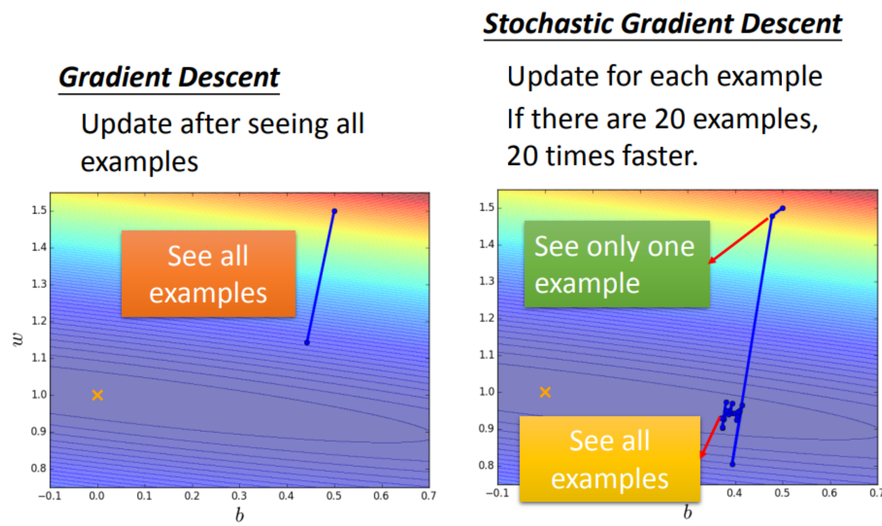
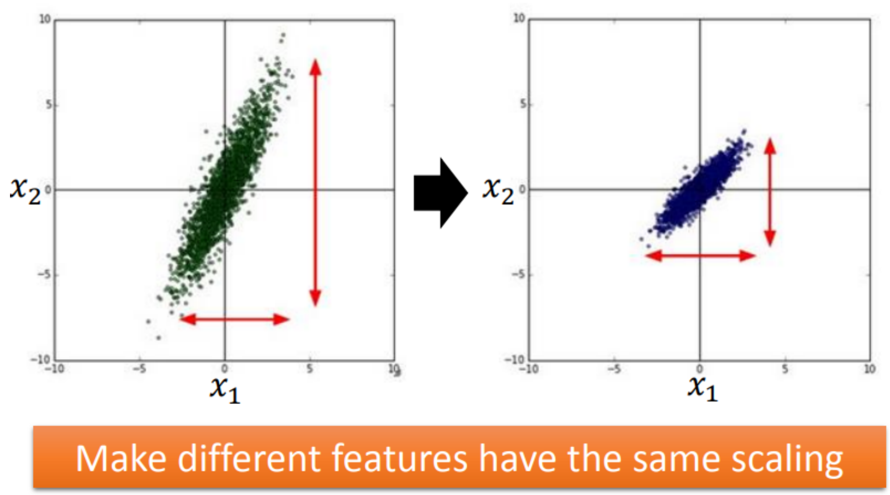



Figure 2: Comparison between *Gradient Descent* and *Stochastic Gradient Descent*

- **Tip 3**: Feature Scaling

$$y = b + w_1x_1 + w_2x_2$$



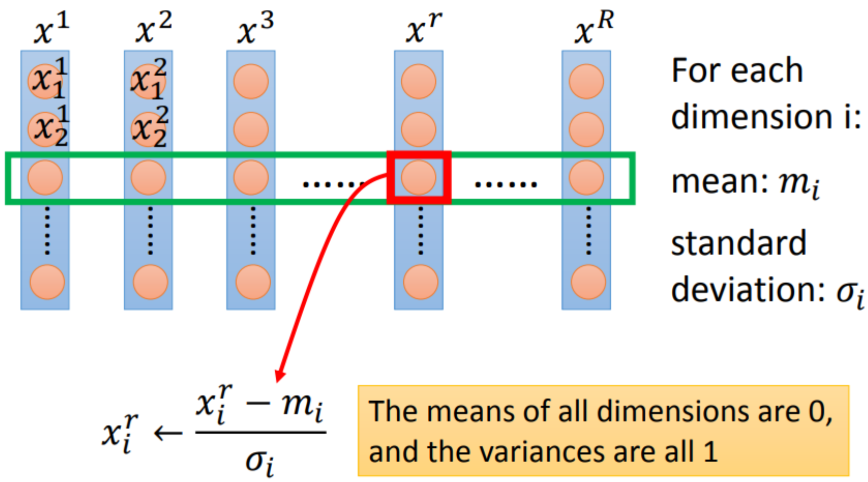



Figure 3: Normalization

# 2 Classification

## 2.1 Generative Model: Two Boxes (Classes)

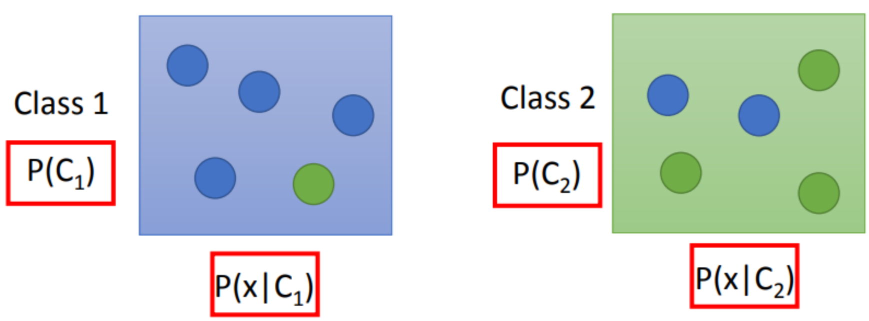



Figure 4: Estimating the probabilities from training data

Given an x, which class does it belong to

$$P(C_1|x) = \frac{P(x|C_1)P(C_1)}{P(x|C_1)P(C_1) + P(x|C_2)P(C_2)}$$

Generative Model

$$P(x) = P(x|C_1)P(C_1) + P(x|C_2)P(C_2)$$

- **Prior**:
  Water and Normal type with ID<400 for training, rest for testing
  Training: 79 Water, 61 Normal
  $P(C_1) = 79/(79 + 61) = 0.56$
  $P(C_2) = 64/(79 + 61) = 0.44$

- **Probability from Class**:

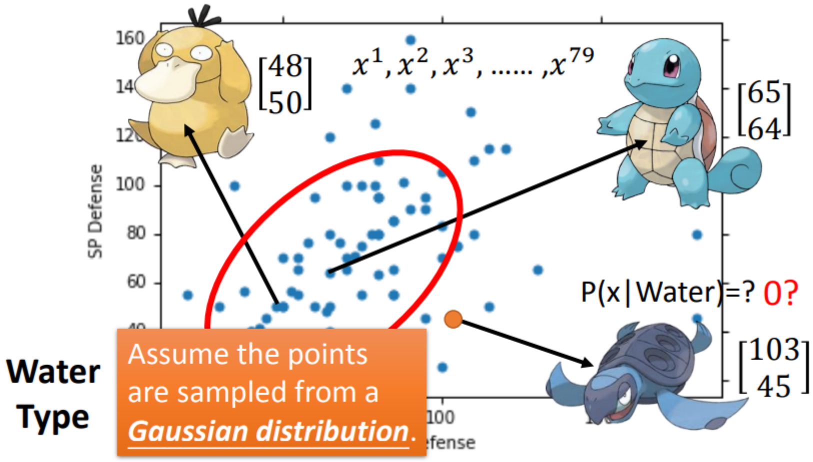



Figure 5: Considering *Defense* and *SP Defence*

1. **Gaussian Distribution**:

$$f_{\mu,\Sigma}(x) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\Sigma|^{1/2}} exp[-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)]$$

   Input: vector x, output: probability of sampling x
   The shape of the function determines by **mean** $\mu$ and **covariance matrix** $\Sigma$

2. **Maximum Likelihood**:

$$L(\mu, \Sigma) = f_{\mu,\Sigma}(x^1) f_{\mu,\Sigma}(x^2) f_{\mu,\Sigma}(x^3) \ldots f_{\mu,\Sigma}(x^{79})$$

   $\mu^*, \Sigma^* = arg\ max L(\mu, \Sigma)$
   $\mu^* = \frac{1}{79}\sum_{n=1}^{79} x^n \quad \Sigma^* = \frac{1}{79}\sum_{n=1}^{79}(x^n - \mu^*)(x^n - \mu^*)^T$

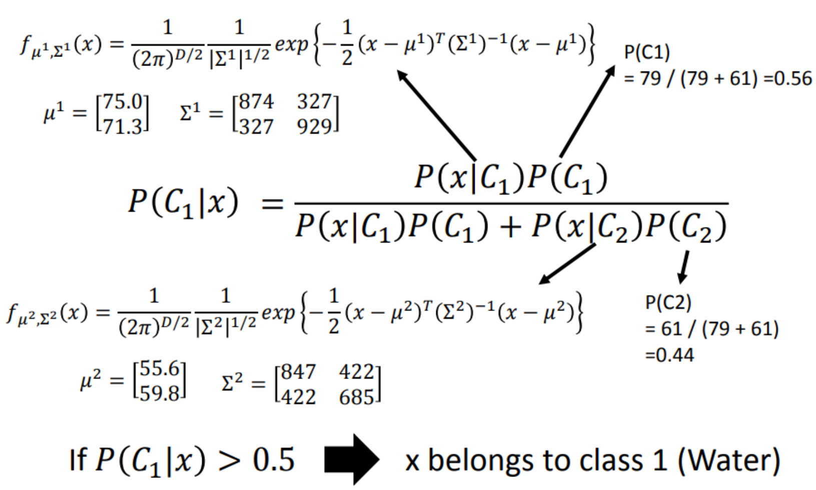



Figure 6: Flowchart

- **Modifying Model**: To avoid overfitting
  Find $\mu^1, \mu^2, \Sigma$ maximizing the likelihood $L(\mu^1, \mu^2, \Sigma)$

$$L(\mu^1, \mu^2, \Sigma) = f_{\mu^1,\Sigma}(x^1) f_{\mu^1,\Sigma}(x^2) ... f_{\mu^1,\Sigma}(x^{79}) f_{\mu^2,\Sigma}(x^{80}) f_{\mu^2,\Sigma}(x^{81}) ... f_{\mu^2,\Sigma}(x^{140})$$

  $\mu^1$ and $\mu^2$ are same $\Sigma = \frac{79}{140}\Sigma^1 + \frac{61}{140}\Sigma^2$

## 2.2 Three Steps

1. **Function Set (Model)**:

$$x \Rightarrow P(C_1|x) = \frac{P(x|C_1)P(C_1)}{P(x|C_1)P(C_1) + P(x|C_2)P(C_2)}$$

If $P(C_1|x) > 0.5$, output: class 1
Otherwise, output: class 2

2. **Goodness of a function**:

   The mean $\mu^*$ and covariance $\Sigma^*$ that maximizing the likelihood (the probability of generating data)

   - Probability Distribution
     You can always use the distribution you like.
     For binary features, you may assume they are from **Bernoulli distributions**.
     If you assume all the dimensions are independent, then you are using **Naive Bayes Classifier**.
   - Posterior Probability

$$P(C_1|x) = \frac{P(x|C_1)P(C_1)}{P(x|C_1)P(C_1) + P(x|C_2)P(C_2)} = \frac{1}{1 + \frac{P(x|C_2)P(C_2)}{P(x|C_1)P(C_1)}} = \frac{1}{1 + exp(-z)} = \sigma(z)$$

$\sigma(z)$: Sigmoid function, $z = ln\frac{P(x|C_1)P(C_1)}{P(x|C_2)P(C_2)}$

   $\Sigma_1 = \Sigma_2 = \Sigma$

$$z = (\mu^1 - \mu^2)^T \Sigma^{-1} x - \frac{1}{2}(\mu^1)^T \Sigma^{-1} \mu^1 + \frac{1}{2}(\mu^2)^T \Sigma^{-1} \mu^2 + ln\frac{N_1}{N_2}$$

$$w^T = (\mu^1 - \mu^2)^T \Sigma^{-1}, \quad b = -\frac{1}{2}(\mu^1)^T \Sigma^{-1} \mu^1 + \frac{1}{2}(\mu^2)^T \Sigma^{-1} \mu^2 + ln\frac{N_1}{N_2}$$

$$\Rightarrow P(C_1|x) = \sigma(w \cdot x + b)$$

   In generative model, we estimate $N_1, N_2, \mu^1, \mu^2, \Sigma$, then we have w and b.

3. **Find the Best Function**: easy

### 2.3 Logistic Regression

1. **Step 1**: Function Set

   We want to find $P_{w,b}(C_1|x)$

   If $P_{w,b}(C_1|x) > 0.5$, output $C_1$

   Otherwise, output $C_2$

   $P_{w,b}(C_1|x) = \sigma(z), \quad z = w \cdot x + b, \quad \sigma(z) = \dfrac{1}{1 + exp(-z)}$

   **Function set**: $f_{w,b}(x) = P_{w,b}(C_1|x)$ , including all different w and b

   Output: between 0 and 1

2. **Step 2**: Goodness of a Function

   | **Training Data** | $x^1$ | $x^2$ | $x^3$ | ... ... | $x^N$ |
   |---|---|---|---|---|---|
   | | $C_1$ | $C_1$ | $C_2$ | ... ... | $C_1$ |

   $$\boxed{\begin{matrix} x^1 & x^2 & x^3 & \dots \dots \\ C_1 & C_1 & C_2 & \dots \dots \end{matrix}} \Rightarrow \boxed{\begin{matrix} x^1 & x^2 & x^3 \dots \dots \\ \hat{y}^1 = 1 & \hat{y}^2 = 1 & \hat{y}^3 = 0 \dots \dots \end{matrix}} \; \hat{y}^n\text{: 1 for class 1, 0 for class 2}$$

   $L(w, b) = f_{w,b}(x^1) f_{w,b}(x^2)(1 - f_{w,b}(x^3))...f_{w,b}(x^N)$

   $\boxed{w^*, b^* = arg\ maxL(w, b) \Rightarrow w^*, b^* = arg\ min\ \ -lnL(w, b)}$

   $$-lnL(w, b) = \sum_n \underline{-[\hat{y}^n ln f_{w,b}(x^n) + (1 - \hat{y}^n) ln(1 - f_{w,b}(x^n))]}$$

   Cross entropy between two Bernoulli distribution

   | Distribution p: | | Distribution q: |
   |---|---|---|
   | $p(x = 1) = \hat{y}^n$ | $\Longleftrightarrow$ cross entropy | $q(x = 1) = f(x^n)$ |
   | $p(x = 0) = 1 - \hat{y}^n$ | | $q(x = 0) = 1 - f(x^n)$ |

   $$H(p, q) = -\sum_x p(x) ln(q(x))$$

3. **Step 3**: Find the Best Function

   $$\frac{\partial - lnL(w, b)}{\partial w_i} = \sum_n -\underbrace{(\hat{y}^n - f_{w,b}(x^n))}_{\text{Larger difference, larger update}} x_i^n$$

   $$w_i \leftarrow w_i - \eta \sum_n -(\hat{y}^n - f_{w,b}(x^n)) x_i^n$$

| | Logistic Regression | Linear Regression |
|---|---|---|
| Step 1 | $f_{w,b}(x) = \sigma(\sum_i w_i x_i + b)$<br>Output: between 0 and 1 | $f_{w,b}(x) = \sum_i w_i x_i + b$<br>Output: any value |
| Step 2 | $\hat{y}^n$: 1 for class 1, 0 for class 2<br>$L(f) = \sum_n C(f(x^n), \hat{y}^n)$ | $\hat{y}^n$: a real number<br>$L(f) = \frac{1}{2}\sum_n (f(x^n) - \hat{y}^n)^2$ |
| Step 3 | $w_i \leftarrow w_i - \eta \sum_n -(\hat{y}^n - f_{w,b}(x^n))x_i^n$ | $w_i \leftarrow w_i - \eta \sum_n -(\hat{y}^n - f_{w,b}(x^n))x_i^n$ |

Table 1: Comparison: *Logistic Regression* and *Linear Regression*

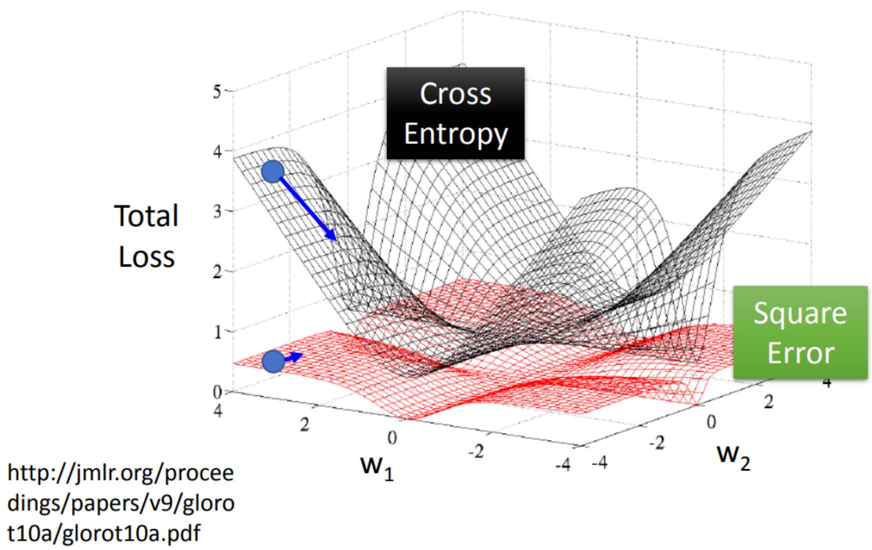
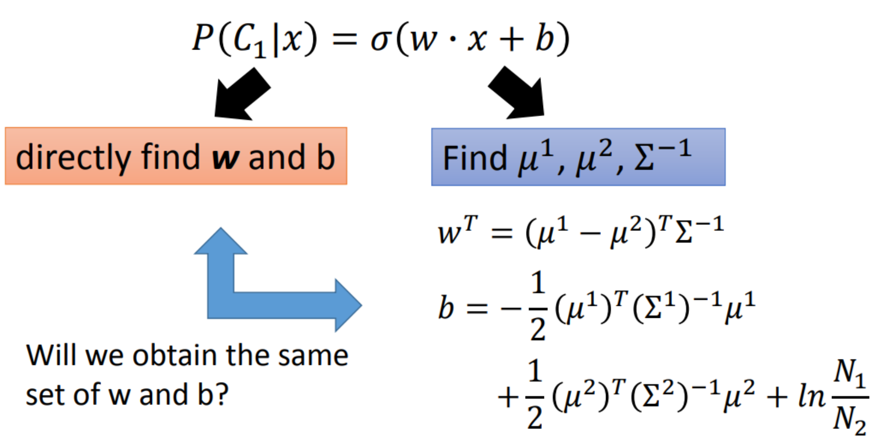



Figure 7: Cross Entropy v.s. Square Error

### 2.4 Generative v.s. Discriminative




The same model (function set), but different function is selected by the same training data.

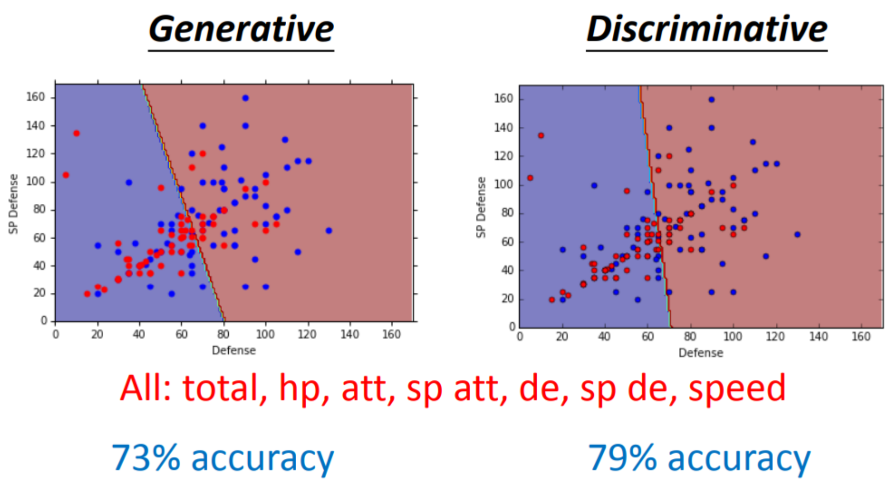



Figure 8: Discriminative v.s. Generative

- Benefits of generative model
  1. With the assumption of probability distribution, less training data is needed
  2. With the assumption of probability distribution, more robust to the noise
  3. Priors and class-dependent probabilities can be estimated from different sources.

## 2.5 Multi-class Classification (3 classes as example)

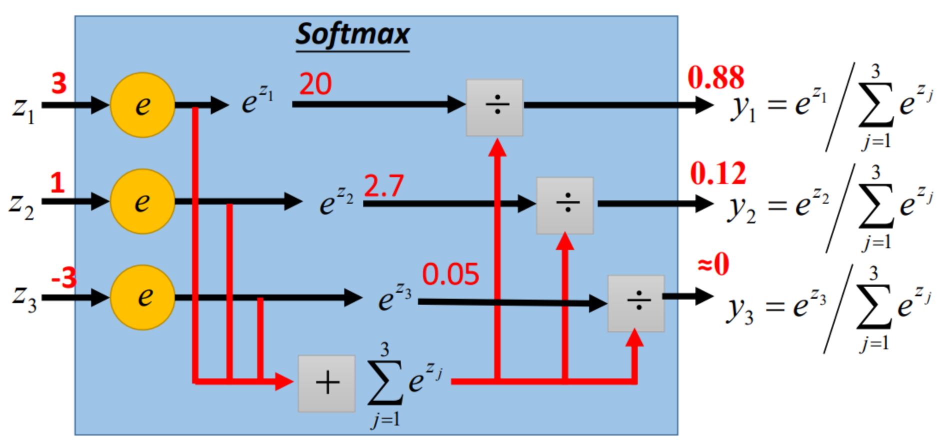
$C_1$: $w^1, b_1$ $\quad z_1 = w^1 \cdot x + b_1$

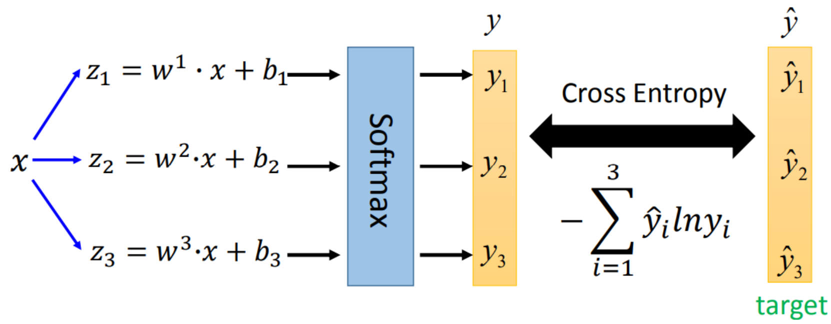
$C_2$: $w^2, b_2$ $\quad z_2 = w^2 \cdot x + b_2$

$C_3$: $w^3, b_3$ $\quad z_3 = w^3 \cdot x + b_3$

**Probability**:

- $1 > y_i > 0$
- $\sum_i y_i = 1 \qquad y_i = P(C_i|x)$






If $x \in$ class 1 $\hat{y} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$ If $x \in$ class 2 $\hat{y} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$ If $x \in$ class 3 $\hat{y} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$

# 3 Deep Learning

## 3.1 Brief Introduction to Deep Learning

- Three Steps for Deep Learning

| Step 1: define a set of function Neural Network | $\Rightarrow$ | Step 2: goodness of function | $\Rightarrow$ | Step 3: pick the best function |
|---|---|---|---|---|

- Neural Network

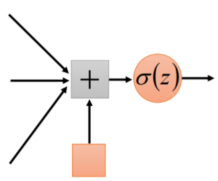
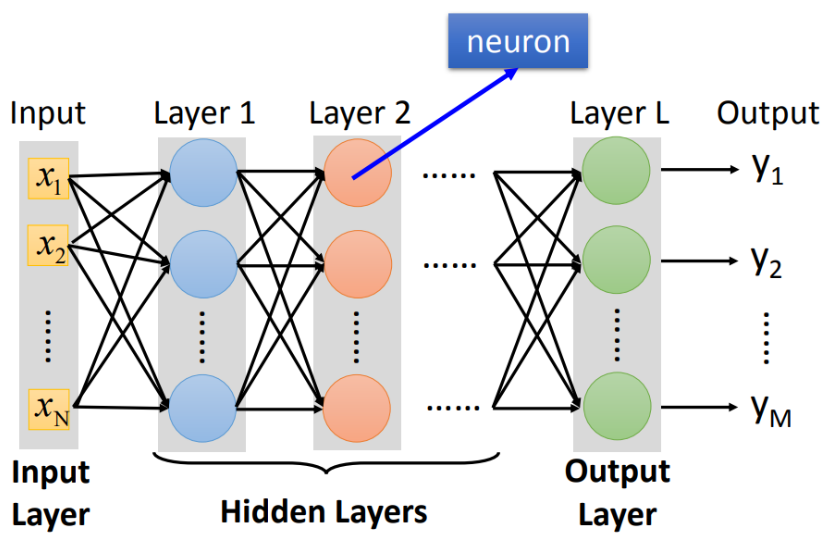



Figure 9: Neuron

Different connection leads to different network structures

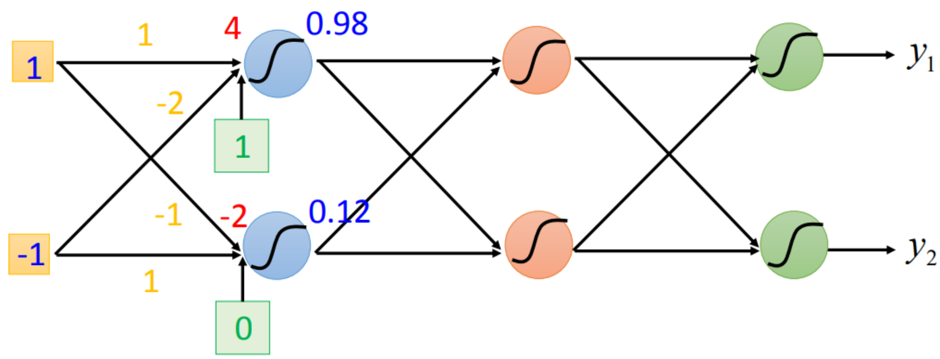
Network parameter $\theta$: all the weights and biases in the "neurons"

- Fully Connect Feedforward Network

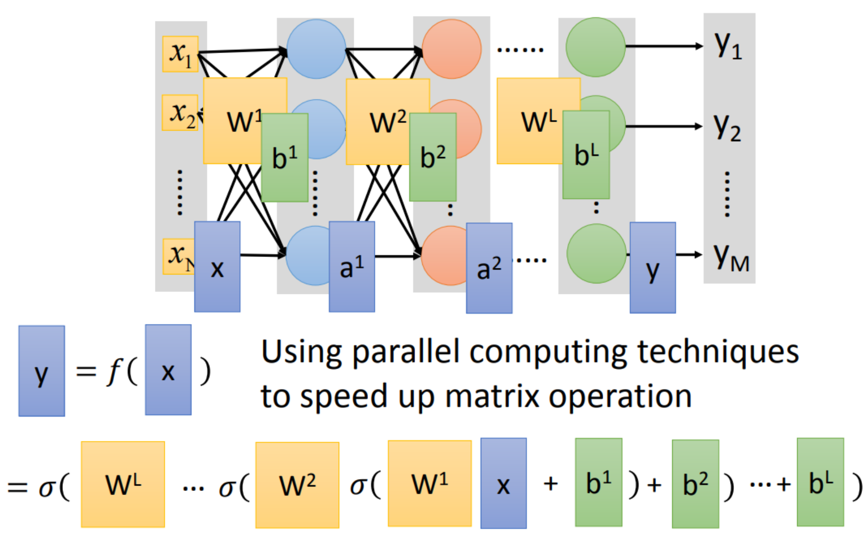


Deep = Many hidden layers

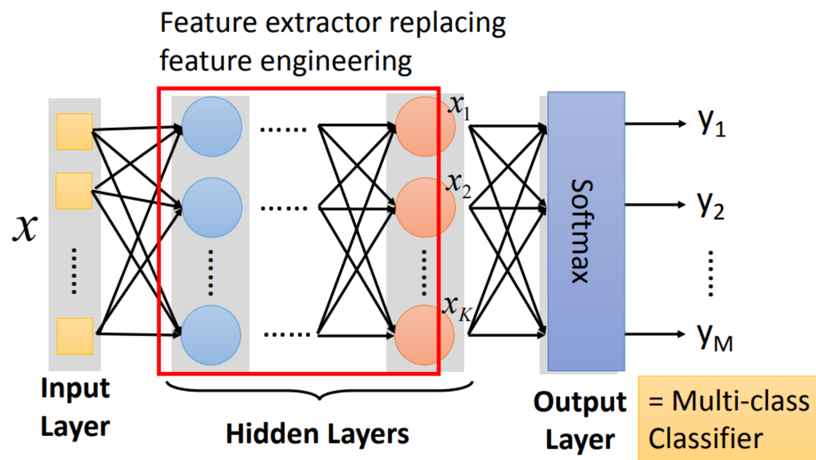
- Matrix Operation



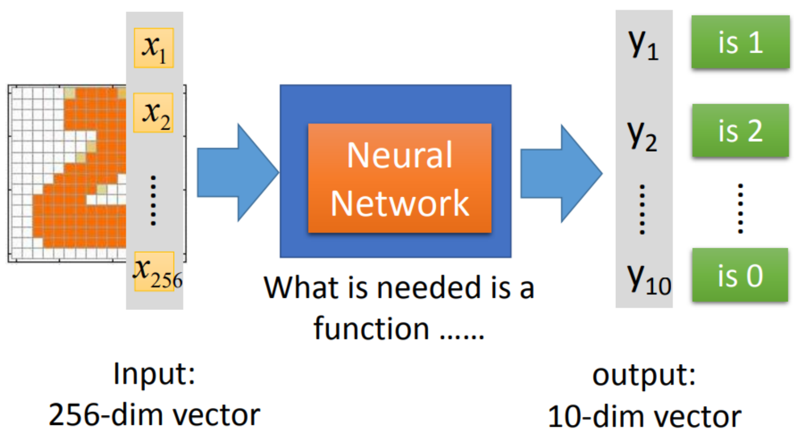

$$\sigma\left(\underbrace{\begin{bmatrix}1&-2\\-1&1\end{bmatrix}\begin{bmatrix}1\\-1\end{bmatrix}+\begin{bmatrix}1\\0\end{bmatrix}}_{\begin{bmatrix}4\\-2\end{bmatrix}}\right)=\begin{bmatrix}0.98\\0.12\end{bmatrix}$$




- Output Layer

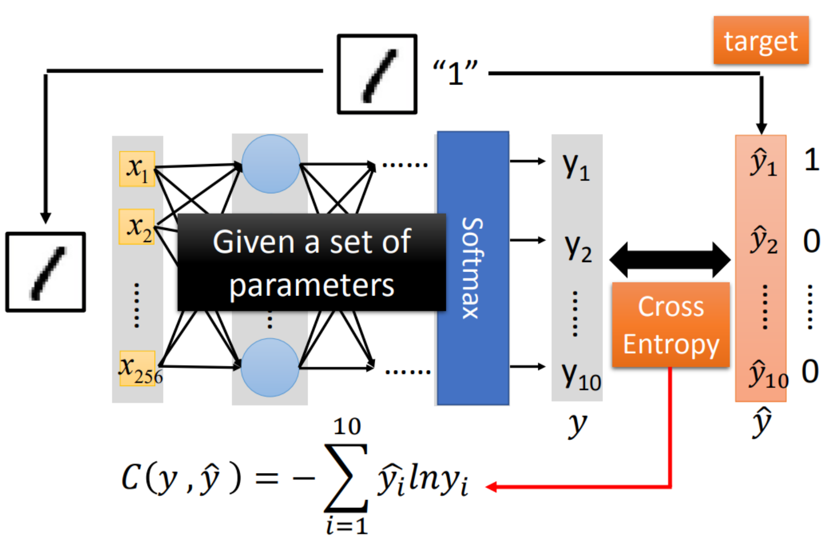



- Example Application: Handwriting Digit Recogition




You need to decide the network structure to let a good function in your function set.

- Loss



Total Loss:

$\boxed{L = \sum_{n=1}^{N} C^n}$ $\Rightarrow$ $\boxed{\text{Find } \textit{a function in function set} \text{ that minimizes total loss L}}$ $\Rightarrow$ $\boxed{\text{Find } \textit{the network parameters } \theta^* \text{ that minimize total loss L}}$

- Gradient Descent

  Gradient:

$$\nabla L = \begin{bmatrix} \frac{\partial L}{\partial w_1} \\ \frac{\partial L}{\partial w_2} \\ \cdot \\ \cdot \\ \cdot \\ \frac{\partial L}{\partial b_1} \\ \cdot \\ \cdot \\ \cdot \end{bmatrix}$$

- Universality Theorem

  Any continuous function $f : R^N \rightarrow R^M$

  can be realized by a network with one hidden layer. (given **enough** hidden neurons)

# 4 Convolutional Neural Network (CNN)

# 5 Recurrent Neural Network (RNN)