

# Logistic Regression and Cross Entropy

Lê Anh Cường

2020

# Outline

1. Information Theory
2. Logistic function
3. Logistic classification with Cross Entropy Loss

# References

- <https://peterroelants.github.io/posts/cross-entropy-logistic/>
- <https://peterroelants.github.io/posts/cross-entropy-softmax/>

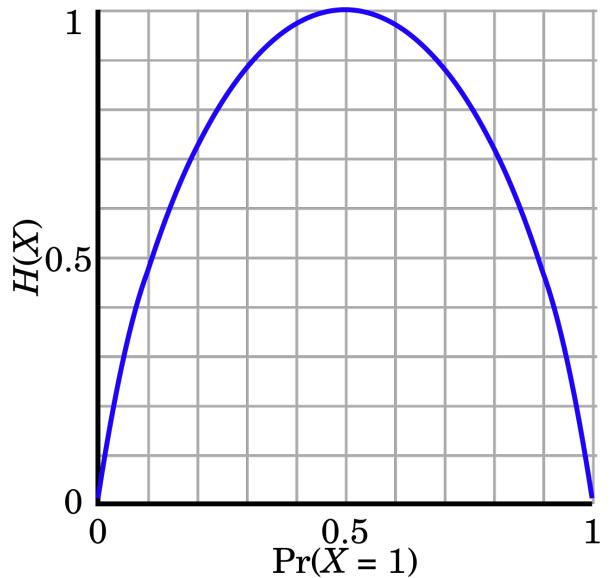
# Entropy

**Entropy** is a **measure** of the unpredictability of the state, or equivalently, of its average **information** content.

$$H(X) = - \sum_{i=1}^n P(x_i) \log_b P(x_i)$$

$p=0.7$ , then

$$\begin{aligned} H(X) &= -p \log_2(p) - q \log_2(q) \\ &= -0.7 \log_2(0.7) - 0.3 \log_2(0.3) \\ &\approx -0.7 \cdot (-0.515) - 0.3 \cdot (-1.737) \\ &= 0.8816 < 1 \end{aligned}$$



$$\begin{aligned} H(X) &= - \sum_{i=1}^n P(x_i) \log_b P(x_i) \\ &= - \sum_{i=1}^2 \frac{1}{2} \log_2 \frac{1}{2} \\ &= - \sum_{i=1}^2 \frac{1}{2} \cdot (-1) = 1 \end{aligned}$$

# Kullback–Leibler divergence

In mathematical statistics, the **Kullback–Leibler divergence** (also called **relative entropy**) is a measure of how one probability distribution is different from a second, reference probability distribution.<sup>[4]</sup>

For discrete probability distributions  $P$  and  $Q$  defined on the same probability space,  $\mathcal{X}$ , the Kullback–Leibler divergence from  $Q$  to  $P$  is defined<sup>[4]</sup> to be

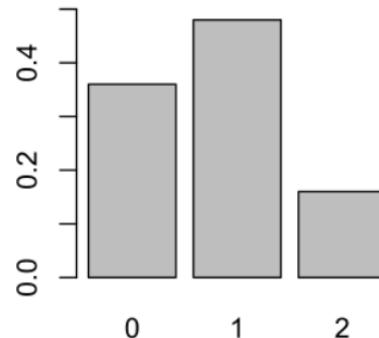
$$D_{\text{KL}}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \log\left(\frac{P(x)}{Q(x)}\right).$$

which is equivalent to

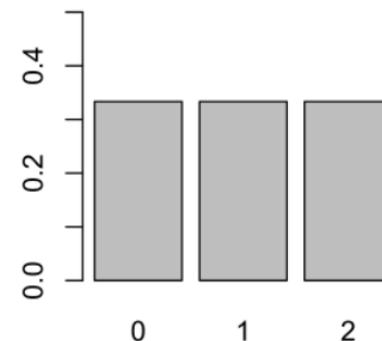
$$D_{\text{KL}}(P \parallel Q) = - \sum_{x \in \mathcal{X}} P(x) \log\left(\frac{Q(x)}{P(x)}\right)$$

# Kullback–Leibler divergence

**Distribution P**  
Binomial with  $p = 0.4$ ,  $N = 2$



**Distribution Q**  
Uniform with  $p = 1/3$



x	0	1	2
Distribution $P(x)$	0.36	0.48	0.16
Distribution $Q(x)$	0.333	0.333	0.333

$$\begin{aligned} D_{\text{KL}}(P \parallel Q) &= \sum_{x \in \mathcal{X}} P(x) \ln \left( \frac{P(x)}{Q(x)} \right) \\ &= 0.36 \ln \left( \frac{0.36}{0.333} \right) + 0.48 \ln \left( \frac{0.48}{0.333} \right) + 0.16 \ln \left( \frac{0.16}{0.333} \right) \\ &= 0.0852996 \end{aligned}$$

# Kullback–Leibler divergence

x	0	1	2
Distribution $P(x)$	0.36	0.48	0.16
Distribution $Q(x)$	0.333	0.333	0.333

$$\begin{aligned} D_{\text{KL}}(P \parallel Q) &= \sum_{x \in \mathcal{X}} P(x) \ln \left( \frac{P(x)}{Q(x)} \right) \\ &= 0.36 \ln \left( \frac{0.36}{0.333} \right) + 0.48 \ln \left( \frac{0.48}{0.333} \right) + 0.16 \ln \left( \frac{0.16}{0.333} \right) \\ &= 0.0852996 \end{aligned}$$

## Interpretations [ edit ]

The Kullback–Leibler divergence from  $Q$  to  $P$  is often denoted  $D_{\text{KL}}(P \parallel Q)$ .

In the context of [machine learning](#),  $D_{\text{KL}}(P \parallel Q)$  is often called the [information gain](#) achieved if  $Q$  is used instead of  $P$ . By analogy with information theory, it is also called the **relative entropy** of  $P$  with respect to  $Q$ . In the context of [coding theory](#),  $D_{\text{KL}}(P \parallel Q)$  can be constructed by measuring the expected number of extra [bits](#) required to [code](#) samples from  $P$  using a code optimized for  $Q$  rather than the code optimized for  $P$ .

# Cross Entropy

In [information theory](#), the **cross entropy** between two [probability distributions](#)  $p$  and  $q$  over the same underlying set of events measures the average number of [bits](#) needed to identify an event drawn from the set if a coding scheme used for the set is optimized for an estimated probability distribution  $q$ , rather than the true distribution  $p$ .

$$H(p, q) = H(p) + D_{\text{KL}}(p\|q) ,$$

where  $H(p)$  is the [entropy](#) of  $p$ .

# Cross Entropy

The cross entropy formula takes in two distributions,  $p(x)$ , the true distribution, and  $q(x)$ , the estimated distribution, defined over the discrete variable  $x$  and is given by

$$H(p, q) = - \sum_{\forall x} p(x) \log(q(x))$$

$$H(p, q) = H(p) + D_{\text{KL}}(p\|q)$$

$$D_{\text{KL}}(P \| Q) = \sum_{x \in \mathcal{X}} P(x) \ln\left(\frac{P(x)}{Q(x)}\right)$$

$$\begin{aligned} H(p) &= - \sum_x P(x) \cdot \log P(x) \\ &= \sum_x P(x) \cdot \log\left(\frac{1}{P(x)}\right) \end{aligned}$$

# Cross Entropy: especial case

Having set up our notation,  $p \in \{y, 1 - y\}$  and  $q \in \{\hat{y}, 1 - \hat{y}\}$ , we can use cross entropy to get a measure of dissimilarity between  $p$  and  $q$ :

$$H(p, q) = - \sum_i p_i \log q_i = -y \log \hat{y} - (1 - y) \log(1 - \hat{y})$$

# Cross Entropy: machine learning

The cross entropy formula takes in two distributions,  $p(x)$ , the true distribution, and  $q(x)$ , the estimated distribution, defined over the discrete variable  $x$  and is given by

$$H(p, q) = - \sum_{\forall x} p(x) \log(q(x))$$

$$H(p, q) = H(p) + D_{\text{KL}}(p\|q)$$

# Cross Entropy: machine learning

$$H(p, q) = - \sum_{\forall x} p(x) \log(q(x)) \rightarrow \text{try to minimize}$$

$$H(p, q) = \underbrace{H(p)}_{\text{constant}} + \underbrace{D_{\text{KL}}(p||q)}_{\rightarrow \text{try to minimize}}$$

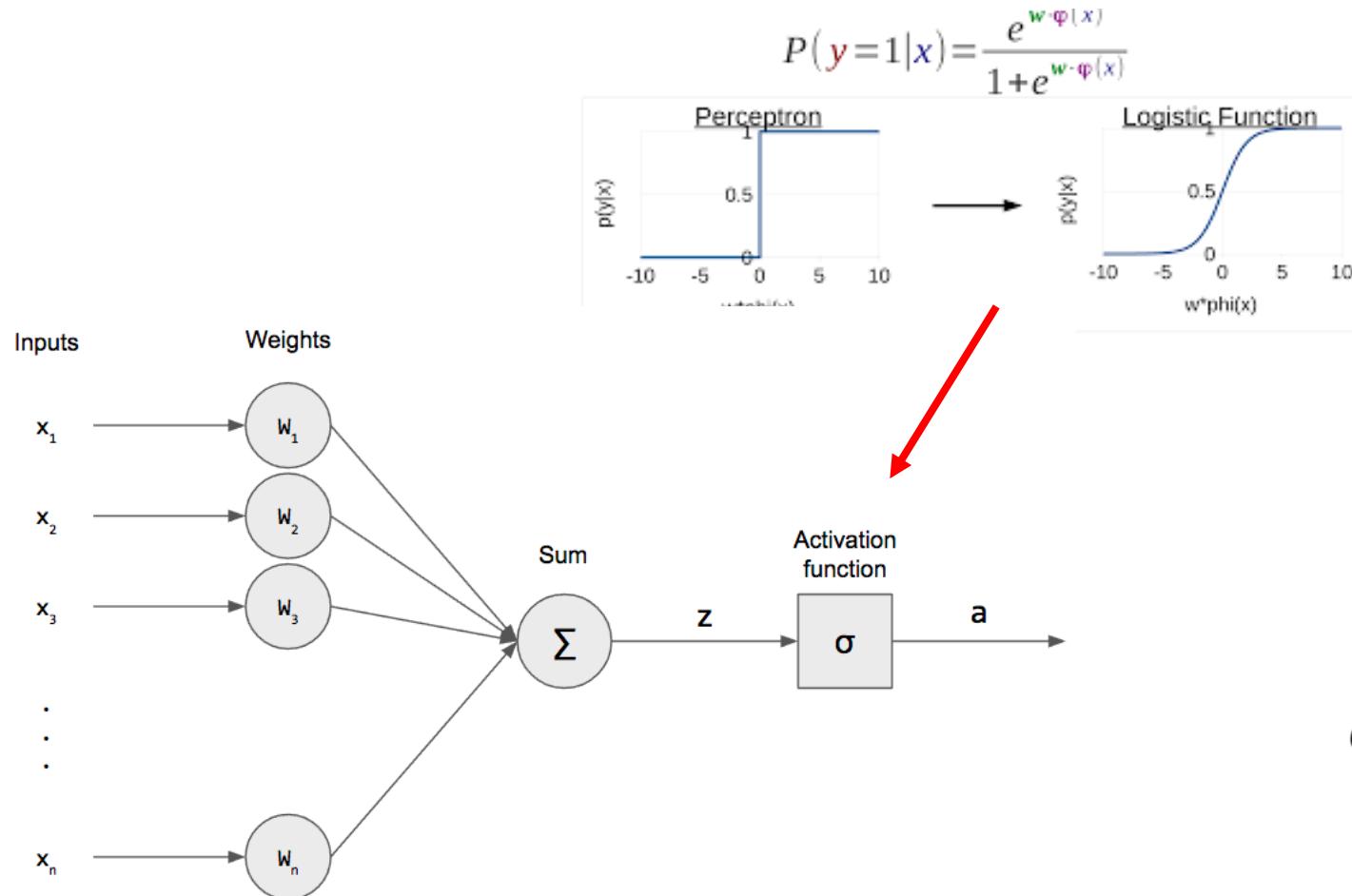
# Cross Entropy: especial case and machine learning

Having set up our notation,  $p \in \{y, 1 - y\}$  and  $q \in \{\hat{y}, 1 - \hat{y}\}$ , we can use cross entropy to get a measure of dissimilarity between  $p$  and  $q$ :

$$H(p, q) = - \sum_i p_i \log q_i = -y \log \hat{y} - (1 - y) \log(1 - \hat{y})$$

learning = minimize

# Logistic Regression



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

# Logistic Function

The goal is to predict the target class  $t$  from an input  $z$ . The probability  $P(t = 1|z)$  that input  $z$  is classified as class  $t = 1$  is represented by the output  $y$  of the logistic function computed as  $y = \sigma(z)$ . The [logistic function](#)  $\sigma$  is defined as:

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

We can write the probabilities that the class is  $t = 1$  or  $t = 0$  given input  $z$  as:

$$P(t = 1|z) = \sigma(z) = \frac{1}{1 + e^{-z}}$$

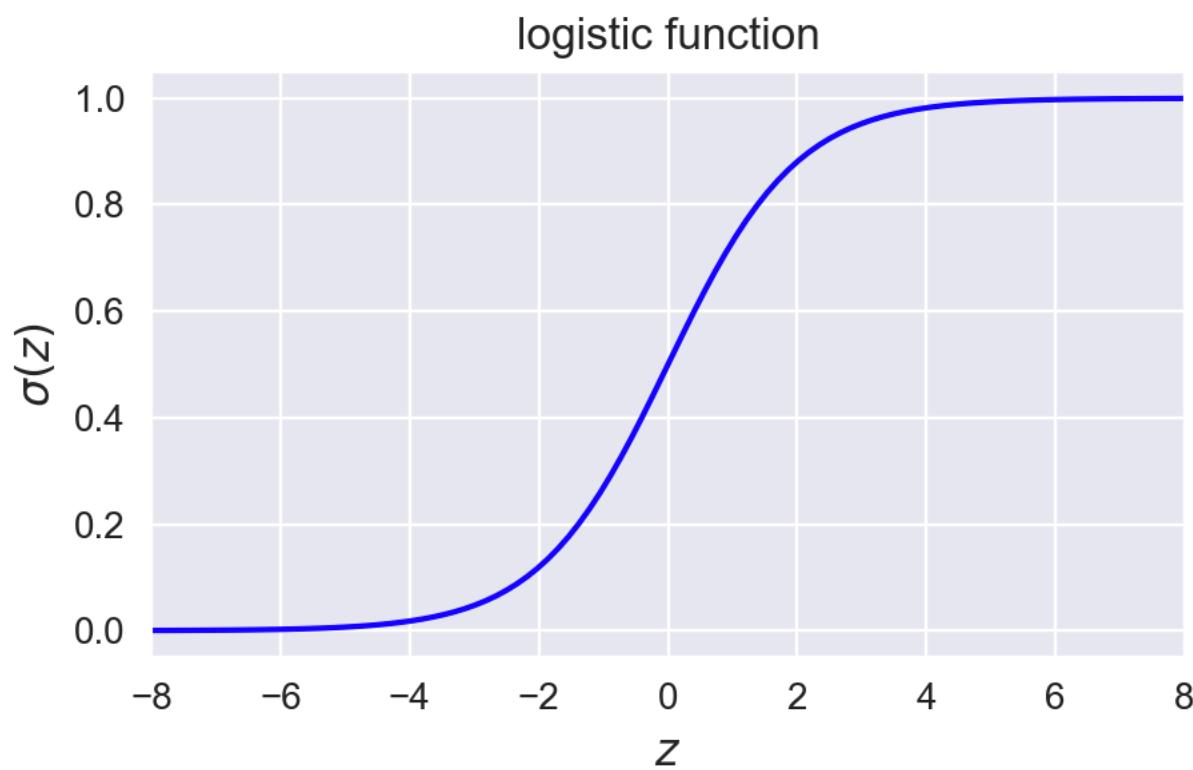
$$P(t = 0|z) = 1 - \sigma(z) = \frac{e^{-z}}{1 + e^{-z}}$$

Note that input  $z$  to the logistic function corresponds to the log **odds ratio** of  $P(t = 1|z)$  over  $P(t = 0|z)$ .

$$\begin{aligned}\log \frac{P(t = 1|z)}{P(t = 0|z)} &= \log \frac{\frac{1}{1+e^{-z}}}{\frac{e^{-z}}{1+e^{-z}}} = \log \frac{1}{e^{-z}} \\ &= \log(1) - \log(e^{-z}) = z\end{aligned}$$

This means that the logg odds ratio  $\log(P(t = 1|z)/P(t = 0|z))$  changes linearly with  $z$ . And if  $z = x \cdot w$  as in neural networks, this means that the logg odds ratio changes linearly with the parameters  $w$  and input samples  $x$ .

```
def logistic(z):
    """Logistic function."""
    return 1 / (1 + np.exp(-z))
```



# Derivative of the logistic function

Since neural networks typically use [gradient](#) based optimization techniques such as [gradient descent](#) it is important to define the [derivative](#) of the output  $y$  of the logistic function with respect to its input  $z$ .  $\partial y / \partial z$  can be calculated as:

$$\frac{\partial y}{\partial z} = \frac{\partial \sigma(z)}{\partial z} = \frac{\partial \frac{1}{1+e^{-z}}}{\partial z} = \frac{-1}{(1+e^{-z})^2} \cdot e^{-z} \cdot -1 = \frac{1}{1+e^{-z}} \frac{e^{-z}}{1+e^{-z}}$$

And since  $1 - \sigma(z) = 1 - 1/(1 + e^{-z}) = e^{-z}/(1 + e^{-z})$  this can be rewritten as:

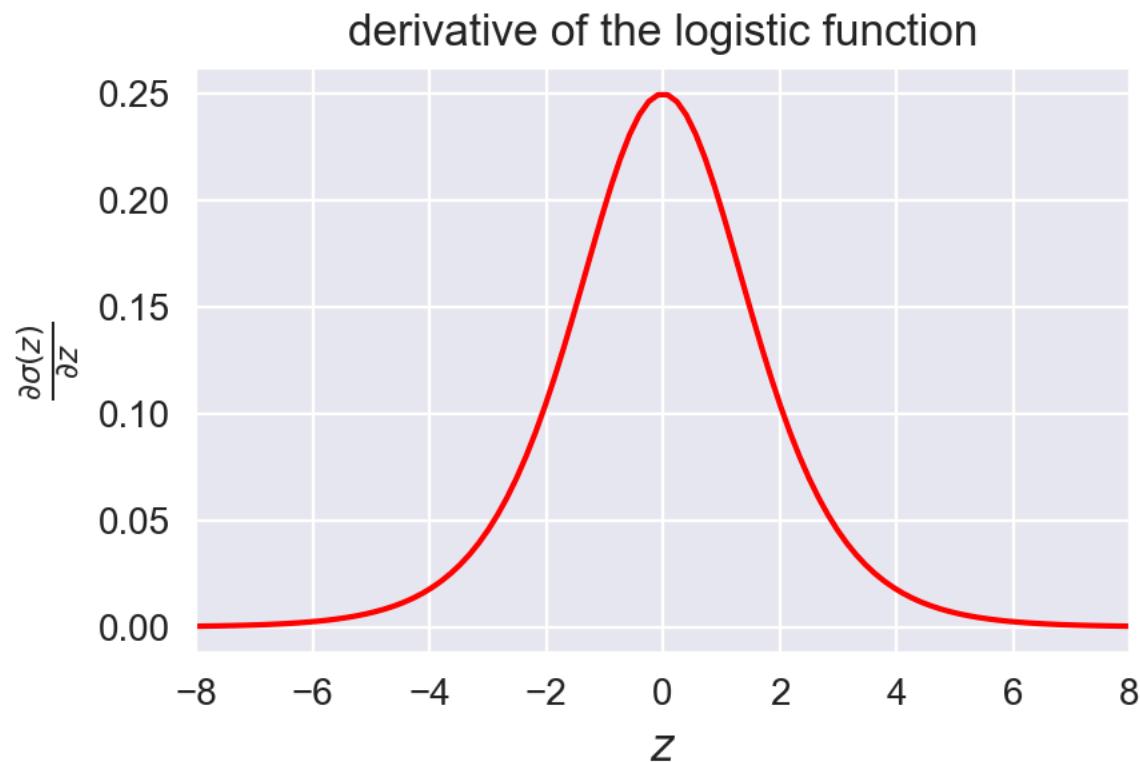
$$\frac{\partial y}{\partial z} = \frac{1}{1+e^{-z}} \frac{e^{-z}}{1+e^{-z}} = \sigma(z) \cdot (1 - \sigma(z)) = y(1 - y)$$

This derivative is implemented as `logistic_derivative(z)` and is plotted below.

```
def logistic_derivative(z):
    """Derivative of the logistic function."""
    return logistic(z) * (1 - logistic(z))
```

```
def logistic_derivative(z):
    """Derivative of the logistic function."""
    return logistic(z) * (1 - logistic(z))
```

```
# Plot the derivative of the logistic function
```



# Cross-entropy loss function for the logistic function

Having set up our notation,  $p \in \{y, 1 - y\}$  and  $q \in \{\hat{y}, 1 - \hat{y}\}$ , we can use cross entropy to get a measure of dissimilarity between  $p$  and  $q$ :

$$H(p, q) = - \sum_i p_i \log q_i = -y \log \hat{y} - (1 - y) \log(1 - \hat{y})$$

learning  $\Rightarrow$  minimize

# Cross-entropy ~ Log likelihood

The neural network model will be optimized by maximizing the [likelihood](#) that a given set of parameters  $\theta$  of the model can result in a prediction of the correct class of each input sample. The parameters  $\theta$  transform each input sample  $i$  into an input to the logistic function  $z_i$ . The likelihood maximization can be written as:

$$\operatorname{argmax}_{\theta} \mathcal{L}(\theta|t, z) = \operatorname{argmax}_{\theta} \prod_{i=1}^n \mathcal{L}(\theta|t_i, z_i)$$

$$\operatorname{argmax}_{\theta} \mathcal{L}(\theta|t, z) = \operatorname{argmax}_{\theta} \prod_{i=1}^n \mathcal{L}(\theta|t_i, z_i)$$

The likelihood  $\mathcal{L}(\theta|t, z)$  can be rewritten as the **joint probability** of generating  $t$  and  $z$  given the parameters  $\theta$ :  $P(t, z|\theta)$ . Since  $P(A, B) = P(A|B)P(B)$  this can be written as:

$$P(t, z|\theta) = P(t|z, \theta)P(z|\theta)$$

Since we are not interested in the probability of  $z$  we can reduce this to:  $\mathcal{L}(\theta|t, z) = P(t|z, \theta) = \prod_{i=1}^n P(t_i|z_i, \theta)$ . Since  $t_i$  is a **Bernoulli variable**, and the probability  $P(t|z) = y$  is fixed for a given  $\theta$  we can rewrite this as:

$$\begin{aligned} P(t|z) &= \prod_{i=1}^n P(t_i = 1|z_i)^{t_i} \cdot (1 - P(t_i = 1|z_i))^{1-t_i} \\ &= \prod_{i=1}^n y_i^{t_i} \cdot (1 - y_i)^{1-t_i} \end{aligned}$$

Since the logarithmic function is a monotone increasing function we can optimize the log-likelihood function  $\underset{\theta}{\operatorname{argmax}} \log \mathcal{L}(\theta|t, z)$ . This maximum will be the same as the maximum from the regular likelihood function. The benefit of using the log-likelihood is that it can prevent numerical **underflow** when the probabilities are low. The log-likelihood function can be written as:

$$\begin{aligned}\log \mathcal{L}(\theta|t, z) &= \log \prod_{i=1}^n y_i^{t_i} \cdot (1 - y_i)^{1-t_i} \\ &= \sum_{i=1}^n t_i \log(y_i) + (1 - t_i) \log(1 - y_i)\end{aligned}$$

→ maximize

Having set up our notation,  $p \in \{y, 1 - y\}$  and  $q \in \{\hat{y}, 1 - \hat{y}\}$ , we can use cross entropy to get a measure of dissimilarity between  $p$  and  $q$ :

$$H(p, q) = - \sum_i p_i \log q_i = -y \log \hat{y} - (1 - y) \log(1 - \hat{y})$$

learning → minimize

# Gradient Descent

$$J(\theta) = -\frac{1}{m} \left[ \sum_{i=1}^m y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)})) \right]$$

Want  $\min_{\theta} J(\theta)$ :

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

$$\begin{aligned}
\frac{\partial}{\partial \theta} J(\theta) &= -\frac{\partial}{\partial \theta} \frac{1}{m} \sum_{i=1}^m \left( y^{(i)} \log(h_\theta(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)})) \right) \\
&= -\frac{1}{m} \sum_{i=1}^m \left( y^{(i)} \frac{\partial}{\partial \theta} \log(h_\theta(x^{(i)})) + (1 - y^{(i)}) \frac{\partial}{\partial \theta} \log(1 - h_\theta(x^{(i)})) \right)
\end{aligned}$$

$$\frac{d}{dz} \log(z) = \frac{1}{z}$$

$$\frac{d}{dz} h(z) = h(z)(1 - h(z))$$

$$\begin{aligned}
\frac{\partial \log(h(z))}{\partial \theta} &= \frac{\partial \log h(z)}{\partial z} \cdot \frac{\partial z}{\partial \theta} \\
&= \frac{\partial \log h(z)}{\partial h(z)} \cdot \frac{\partial h(z)}{\partial z} \cdot \frac{\partial z}{\partial \theta}
\end{aligned}$$

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

$$\frac{\partial y}{\partial z} = \frac{1}{1 + e^{-z}} \frac{e^{-z}}{1 + e^{-z}} = \sigma(z) \cdot (1 - \sigma(z)) = y(1 - y)$$

$$\begin{aligned}
\frac{\partial}{\partial \theta} J(\theta) &= -\frac{\partial}{\partial \theta} \frac{1}{m} \sum_{i=1}^m \left( y^{(i)} \log(h_\theta(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)})) \right) \\
&= -\frac{1}{m} \sum_{i=1}^m \left( y^{(i)} \frac{\partial}{\partial \theta} \log(h_\theta(x^{(i)})) + (1 - y^{(i)}) \frac{\partial}{\partial \theta} \log(1 - h_\theta(x^{(i)})) \right)
\end{aligned}$$

$$\frac{d}{dz} \log(z) = \frac{1}{z}$$

$$\frac{d}{dz} h(z) = h(z)(1 - h(z))$$

$$\begin{aligned}
\frac{\partial \log(h(z))}{\partial \theta} &= \frac{\partial \log h(z)}{\partial z} \cdot \frac{\partial z}{\partial \theta} \\
&= \frac{\partial \log h(z)}{\partial h(z)} \cdot \frac{\partial h(z)}{\partial z} \cdot \frac{\partial z}{\partial \theta}
\end{aligned}$$

$$\begin{aligned}
\frac{\partial}{\partial \theta_j} J(\theta) &= -\frac{1}{m} \sum_{i=1}^m y^{(i)} \frac{x_j^{(i)}}{h(x^{(i)})} h(x^{(i)})(1 - h(x^{(i)})) + (1 - y^{(i)}) \frac{x_j^{(i)}}{1 - h(x^{(i)})} (-h(x^{(i)})(1 - h(x^{(i)}))) \\
&= -\frac{1}{m} \sum_{i=1}^m (y^{(i)} (1 - h(x^{(i)})) - (1 - y^{(i)}) h(x^{(i)})) x_j^{(i)} \\
&= -\frac{1}{m} \sum_{i=1}^m \left( y^{(i)} - y^{(i)} h(x^{(i)}) - h(x^{(i)}) + y^{(i)} h(x^{(i)}) \right) x_j^{(i)} \\
&= \frac{1}{m} \sum_{i=1}^m \left( h(x^{(i)}) - y^{(i)} \right) x_j^{(i)}
\end{aligned}$$

## Gradient Descent

$$J(\theta) = -\frac{1}{m} \left[ \sum_{i=1}^m y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)})) \right]$$

Want  $\min_{\theta} J(\theta)$ :

Repeat {

$$\theta_j := \theta_j - \alpha \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

}

(simultaneously update all  $\theta_j$ )