Deep learning with Neural Networks

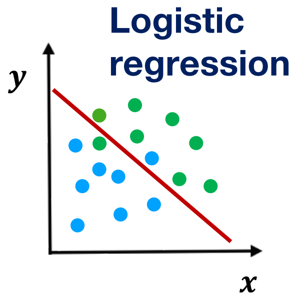
Logistic Regression, Stochastic Optimization & Regularization

Pablo Martínez Olmos, pamartin@ing.uc3m.es

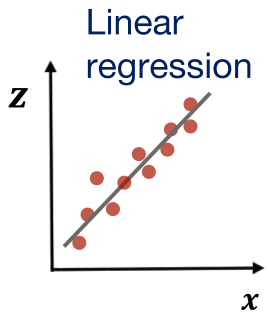
Binary Logistic Regression

- Binary classification method

x → y: 1: CAT, 0: NOT CAT

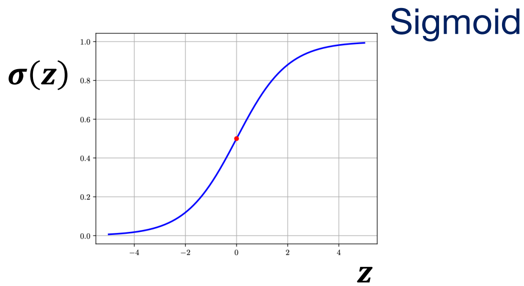


=



$z = w^T x + w_0$

+



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

$$p(y = 1 / x) = \sigma(w^T x + w_0) = \frac{1}{1 + e^{-(w^T x + w_0)}}$$

- Example

$$\underbrace{\begin{bmatrix} 1.5 & 0.3 & -2.1 & 0.0 \end{bmatrix}}_{w^T} \cdot \underbrace{\begin{bmatrix} 1.8 \\ 0.6 \\ 0.4 \\ 2 \end{bmatrix}}_{x} + \underbrace{0.2}_{w_0} = \underbrace{2.24}_{z} \xrightarrow{\sigma} \underbrace{0.90}_{p(y = 1|x)}$$

Binary Logistic Regression

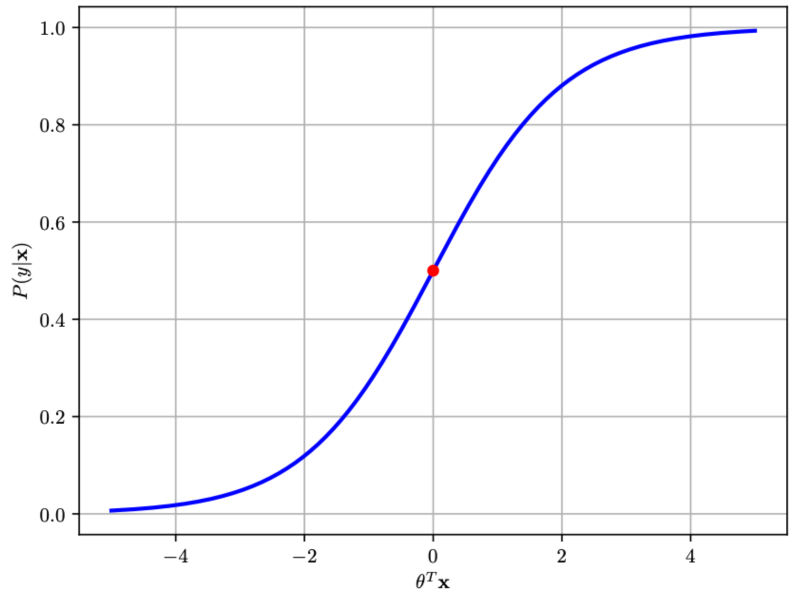
- Training database: $\mathcal{D} \doteq (\boldsymbol{x}^{(i)}, y^{(i)})_{i=1}^{N}$ $\qquad \boldsymbol{x}^{(i)} \in \mathbb{R}^m$ $\qquad y^{(i)} \in \{0, 1\}$

$$P(y=1|\boldsymbol{x}) = \frac{1}{1+\mathrm{e}^{-(\boldsymbol{w}^T\boldsymbol{x}+w_0)}} \doteq \sigma(\boldsymbol{w}^T\boldsymbol{x}+w_0)$$

- Decision boundary:

$$\{\boldsymbol{x} \in \mathbb{R}^m : \boldsymbol{w}^T\boldsymbol{x} + w_0 = \boldsymbol{0}\}$$

Hyperplane! Linear classifier



Logistic Regression: training

Binary Logistic Regression. Binary Cross Entropy Function

$$P(y=1|\boldsymbol{x}) = \frac{1}{1+\mathrm{e}^{-(\boldsymbol{w}^T\boldsymbol{x}+w_0)}} \doteq \sigma(\boldsymbol{w}^T\boldsymbol{x}+w_0)$$

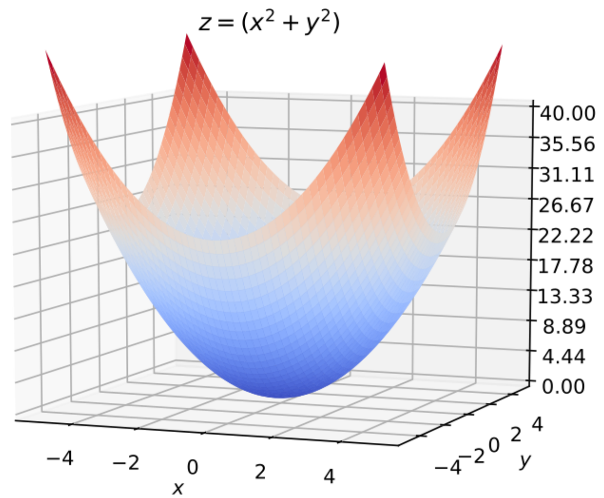
Binary Cross Entropy Function

$$\begin{aligned}\mathcal{L} &= -\log P(\boldsymbol{y}|\boldsymbol{X}) \\ &= -\sum_{i=1}^{N} \mathbb{1}[y^{(i)}=1]\log P(y^{(i)}=1|\boldsymbol{x}^{(i)}) + \mathbb{1}[y^{(i)}=0]\log P(y^{(i)}=0|\boldsymbol{x}^{(i)})\end{aligned}$$

$\mathcal{L}$ **is convex!!**

Logistic Regression Training

$$\hat{\boldsymbol{w}}, \hat{w}_0 = \arg\min_{\boldsymbol{w}, w_0} \mathcal{L}$$



Logistic Regression Training

Binary Cross Entropy Function

$$\begin{aligned}\mathcal{L} &= -\log P(\boldsymbol{y}|\boldsymbol{X}) \\ &= -\sum_{i=1}^{N} \mathbb{1}[y^{(i)} = 1] \log P(y^{(i)} = 1|\boldsymbol{x}^{(i)}) + \mathbb{1}[y^{(i)} = 0] \log P(y^{(i)} = 0|\boldsymbol{x}^{(i)})\end{aligned}$$

Logistic Regression Training

$$\hat{\boldsymbol{w}}, \hat{w}_0 = \arg\min_{\boldsymbol{w}, w_0} \mathcal{L} \quad \Longrightarrow \quad \nabla_{\boldsymbol{w}, w_0} \mathcal{L}(\hat{\boldsymbol{w}}, \hat{w}_0) = \mathbf{0}$$

No closed-form solution!

Gradient Descent (GD)

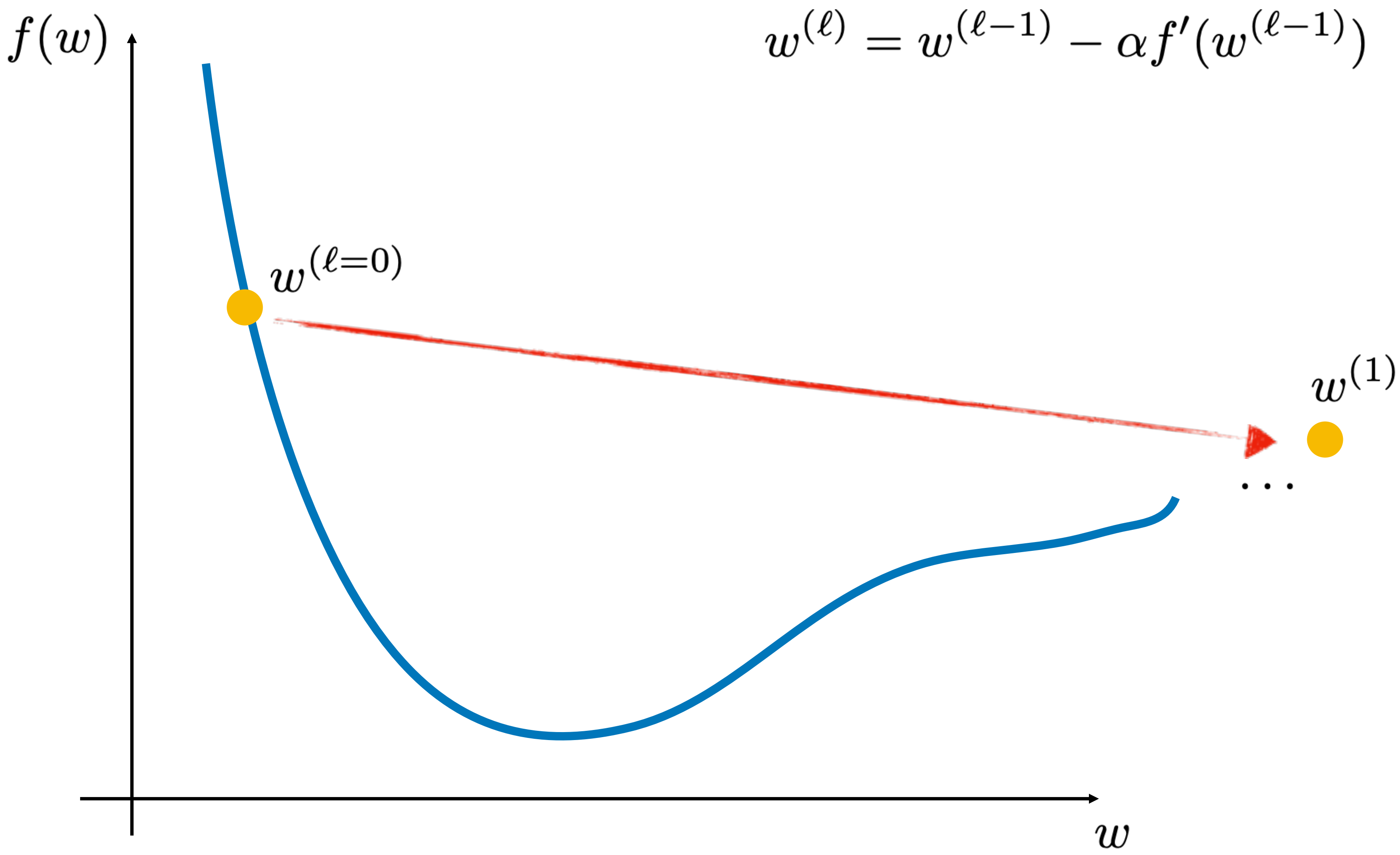
$$\boldsymbol{w}_+ \doteq \begin{bmatrix} w_0 & \boldsymbol{w} \end{bmatrix} \Rightarrow \nabla_{\boldsymbol{w}_+} \mathcal{L} = \mathbf{0}$$

the step size

$$\boldsymbol{w}_+^{(\ell)} = \boldsymbol{w}_+^{(\ell-1)} - (\alpha \nabla \mathcal{L})\,\big|_{\boldsymbol{w}_+^{(\ell-1)}}$$

Choosing the step size manually is a **hard problem!**

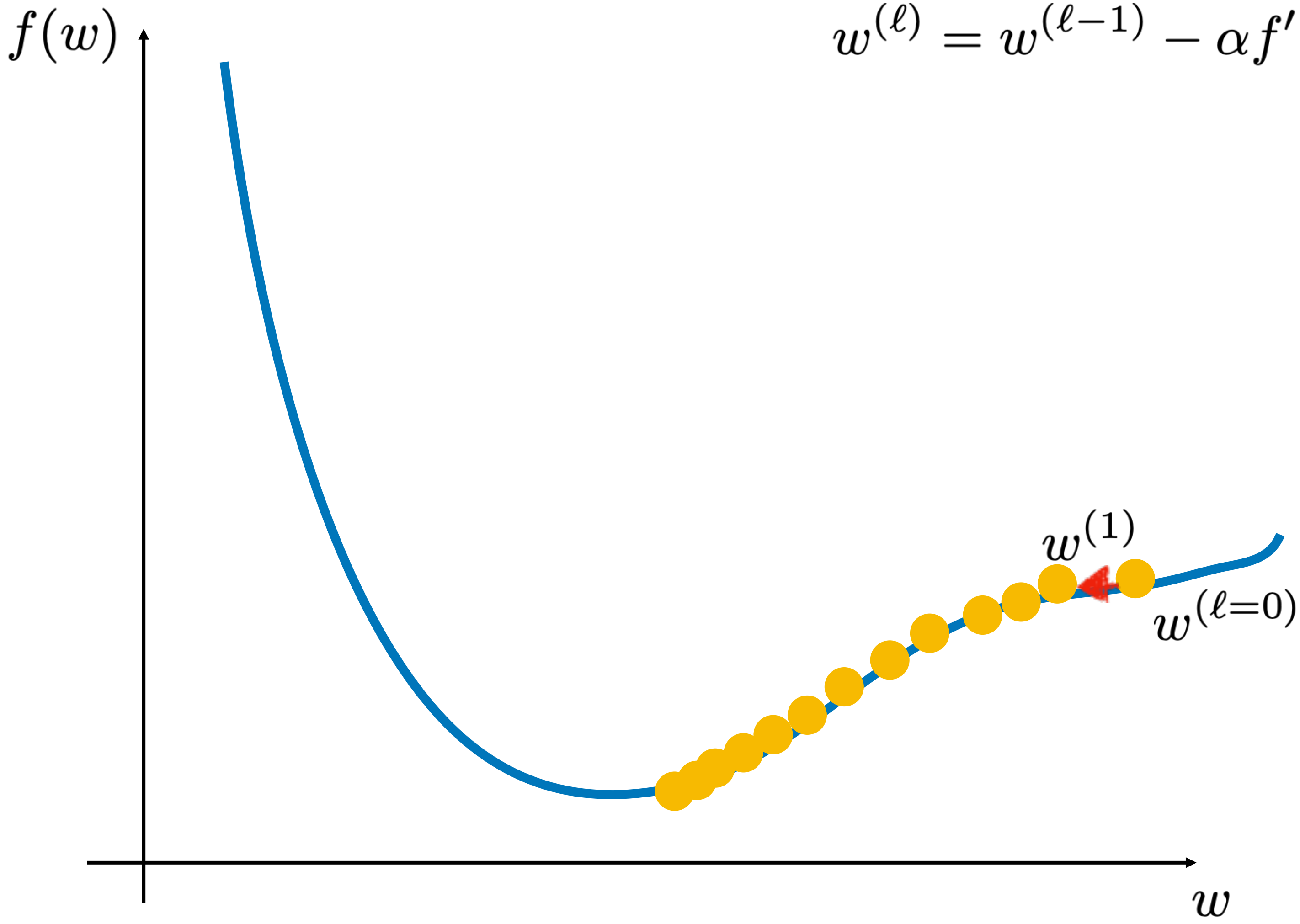
Gradient Descent



If α is too large ….

Gradient Descent

$$w^{(\ell)} = w^{(\ell-1)} - \alpha f'(w^{(\ell-1)})$$



If α is too small ….

Line Search methods: automatically running the step size

$$\boldsymbol{w}_{+}^{(\ell)} = \boldsymbol{w}_{+}^{(\ell-1)} - \left(\alpha^{(\ell)} \nabla \mathcal{L}\right)\Big|_{\boldsymbol{w}_{+}^{(\ell-1)}}$$

- Estimate the largest possible step size that guarantees that the function decreases
- Every state-of-the-art Deep Learning library contains implementations of various algorithms to optimize gradient descent
- Momentum, Adam, Adagrad, …

Check out in Aula Global two excellent posts on SGD methods for Deep Learning

(Link1, Link2)

Logistic Regression:
Stochastic gradient descent

Stochastic Optimization (Mini-batch Optimization)

Binary Cross Entropy Function

$$
\begin{aligned}
\mathcal{L} &= -\log P(\boldsymbol{y}|\boldsymbol{X}) \\
&= -\sum_{i=1}^{N} \mathbb{1}[y^{(i)} = 1] \log P(y^{(i)} = 1|\boldsymbol{x}^{(i)}) + \mathbb{1}[y^{(i)} = 0] \log P(y^{(i)} = 0|\boldsymbol{x}^{(i)})
\end{aligned}
$$

Expensive for very large databases!!

$$
\nabla_{\boldsymbol{w}_+}\mathcal{L} = \sum_{i=1}^{N} \left(\sigma \left(\boldsymbol{w}_+ \begin{bmatrix} 1.0 \\ \boldsymbol{x}^{(i)} \end{bmatrix} \right) - y^{(i)} \right) \begin{bmatrix} 1.0 \\ \boldsymbol{x}^{(i)} \end{bmatrix}
$$

Mini-batch optimization

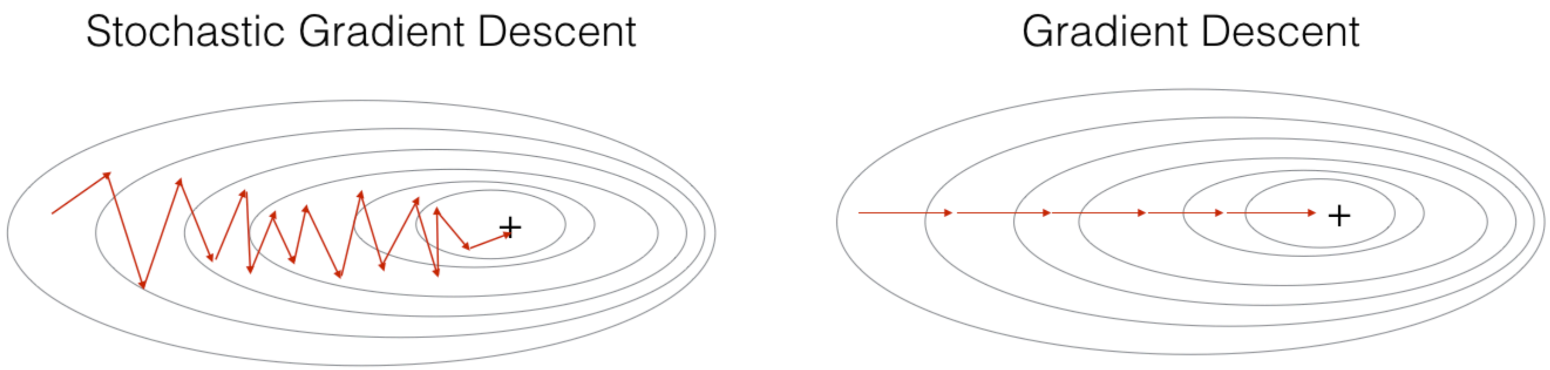
Select at random a mini-batch $\mathcal{B}$ of data at every SGD iteration

Stochastic Optimization (Mini-batch Optimization)

$$\boldsymbol{w}_+ \doteq \left[\begin{array}{cc} w_0 & \boldsymbol{w} \end{array}\right] \Rightarrow \nabla_{\boldsymbol{w}_+}\mathcal{L} = \mathbf{0}$$

$$\boldsymbol{w}_+^{(\ell)} = \boldsymbol{w}_+^{(\ell-1)} - (\alpha \nabla \mathcal{L})\,\big|_{\boldsymbol{w}_+^{(\ell-1)}}$$

$$\nabla_{\boldsymbol{w}_+}\mathcal{L} \approx \sum_{i \in \mathcal{B}} \left(\sigma\left(\boldsymbol{w}_+ \left[\begin{array}{c} 1.0 \\ \boldsymbol{x}^{(i)} \end{array}\right]\right) - y^{(i)}\right) \left[\begin{array}{c} 1.0 \\ \boldsymbol{x}^{(i)} \end{array}\right]$$



Source: this post

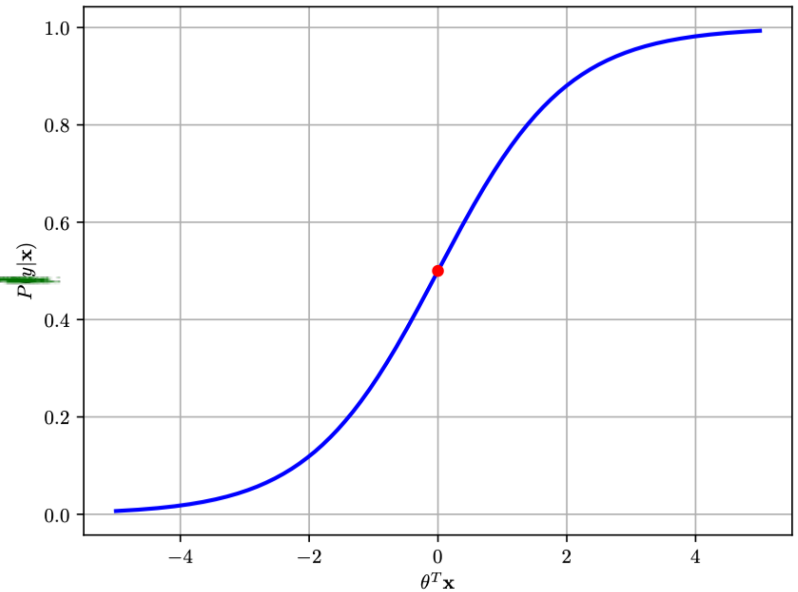
Logistic Regression: multiple classes

Binary Logistic Regression

- Discriminative classifier:

$$P(y=1|\boldsymbol{x}) = \frac{1}{1+\mathrm{e}^{-(\boldsymbol{w}^T\boldsymbol{x}+w_0)}} \doteq \sigma(\boldsymbol{w}^T\boldsymbol{x}+w_0)$$

Generalized linear model



- Decision boundary:

$$\{\boldsymbol{x} \in \mathbb{R}^m : \boldsymbol{w}^T\boldsymbol{x} + w_0 = \mathbf{0}\}$$

Hyperplane! Linear classifier

Multiclass Logistic Regression

- Training database: $\mathcal{D} \doteq (\boldsymbol{x}^{(i)}, y^{(i)})_{i=1}^{N}$ $\quad \boldsymbol{x}^{(i)} \in \mathbb{R}^m \quad y^{(i)} \in \{1, \ldots, K\}$

- Discriminative classifier based on the **Softmax function**:

$$P(y = k|\boldsymbol{x}) = \frac{\mathrm{e}^{-(z_k)}}{\sum_{j=1}^{K} \mathrm{e}^{-(z_j)}}, \quad \boldsymbol{z} = \boldsymbol{W} \begin{bmatrix} 1.0 \\ \boldsymbol{x}^{(i)} \end{bmatrix}$$

$$K \times (m+1)$$

- Loss function (Cross entropy):

$$\mathcal{L} = -\log P(\boldsymbol{y}|\boldsymbol{X}) = -\sum_{n=1}^{N} \sum_{k=1}^{K} \mathbb{1}[y^{(n)} == k] \log P(y = k|\boldsymbol{x})$$

- Similar gradient expression