Softmax Regression

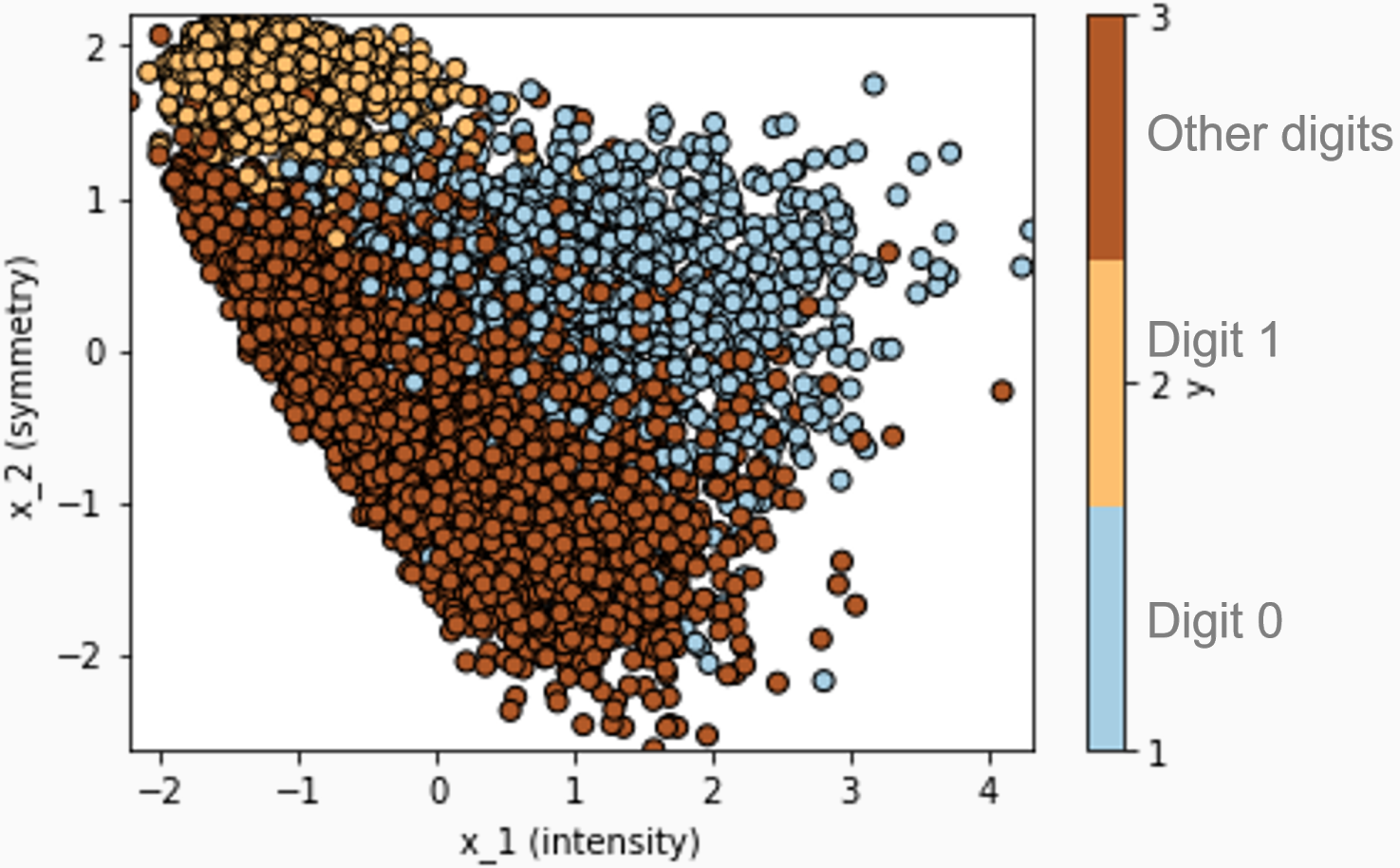
Trần Trung Kiên (ttkien@fit.hcmus.edu.vn)

Last update: March 2, 2024

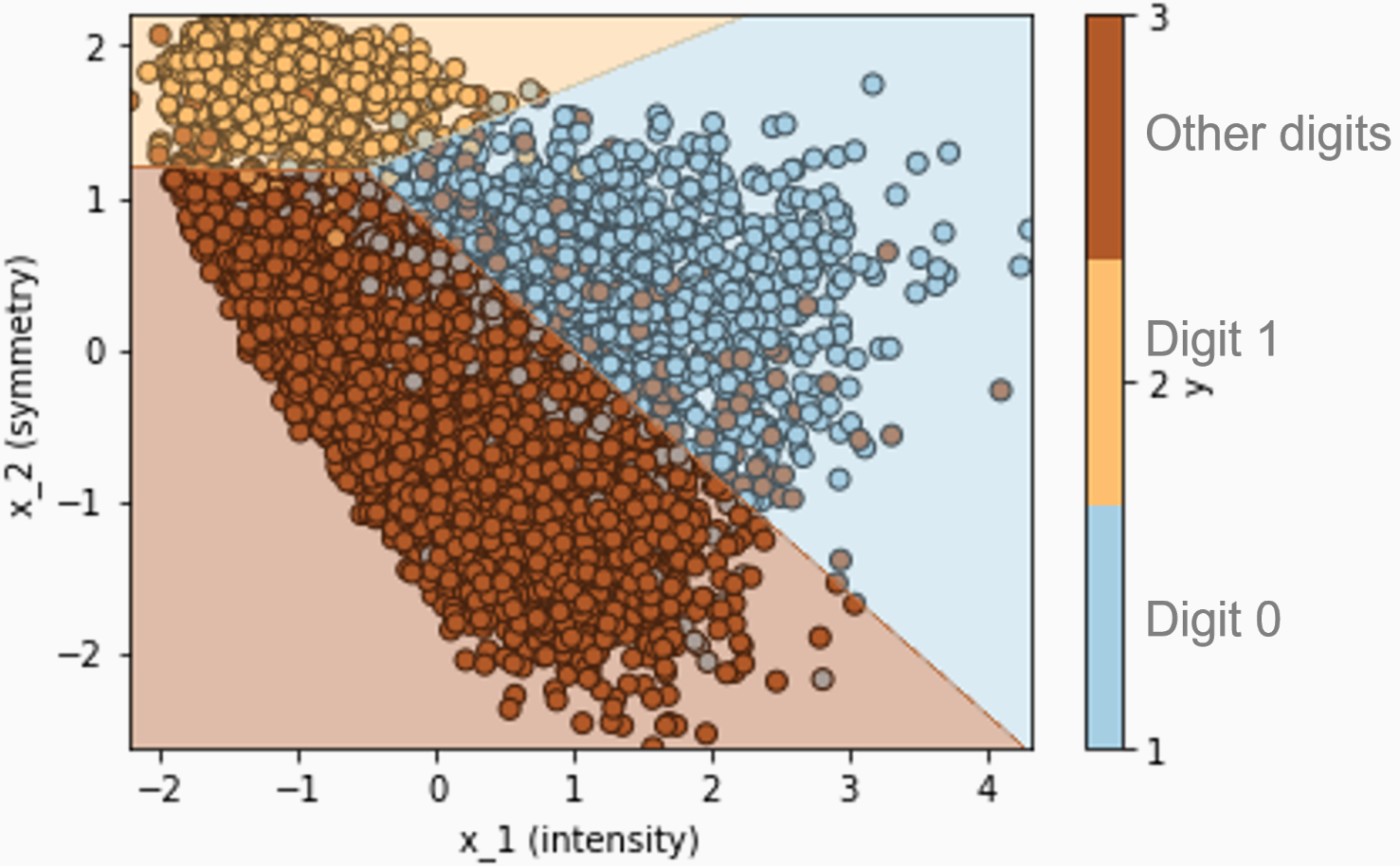
KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

fit@hcmus

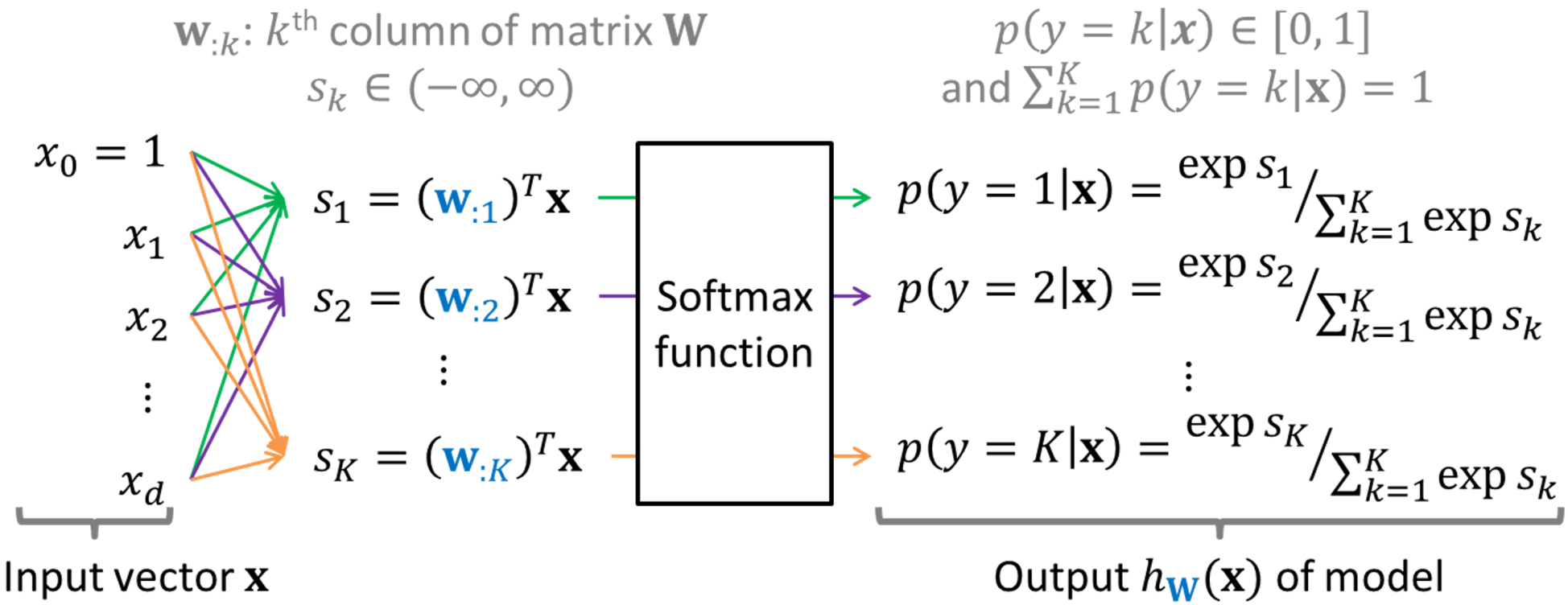
Softmax Regression — idea



Look at training data and choose *lines* (planes in 3D space, hyper-planes in >3D space), each line separates two classes well


x_2 (symmetry)
x_1 (intensity)
2
1
0
−1
−2
−2
−1
0
1
2
3
4
3
Other digits
Digit 1
2
y
Digit 0
1

Softmax Regression — model form


$\mathbf{w}_{:k}$: k^{th} column of matrix $\mathbf{W}$
$s_k \in (-\infty, \infty)$
$p(y = k|\boldsymbol{x}) \in [0, 1]$
and $\sum_{k=1}^{K} p(y = k|\mathbf{x}) = 1$
$x_0 = 1$
x_1
x_2
x_d
$s_1 = (\mathbf{w}_{:1})^T\mathbf{x}$
$s_2 = (\mathbf{w}_{:2})^T\mathbf{x}$
$s_K = (\mathbf{w}_{:K})^T\mathbf{x}$
Softmax function
$p(y = 1|\mathbf{x}) = \exp s_1 / \sum_{k=1}^{K} \exp s_k$
$p(y = 2|\mathbf{x}) = \exp s_2 / \sum_{k=1}^{K} \exp s_k$
$p(y = K|\mathbf{x}) = \exp s_K / \sum_{k=1}^{K} \exp s_k$
Input vector $\mathbf{x}$
Output $h_{\mathbf{W}}(\mathbf{x})$ of model

Softmax Regression — training

Given training data: $\{(\mathbf{x}^{(1)}, y^{(1)}), ..., (\mathbf{x}^{(N)}, y^{(N)})\}$
where $\mathbf{x}^{(n)} \in \mathbb{R}^{d+1}$ and $y^{(n)} \in \{1, 2, ..., K\}$

We should choose model $h_{\mathbf{W}}(\mathbf{x})$ with $\mathbf{W} = ?$

Steps to find $\mathbf{W}$:

1. Define $E_{train}(\mathbf{W})$ — error function on training data
2. $\min_{\mathbf{W}} E_{train}(\mathbf{W})$

Step 1: Define $E_{train}(\mathbf{W})$ — error function on training data

People often use *Cross-entropy Error*

Cross-entropy Error on a data point $(\mathbf{x}, y)$:

$$e(h_{\mathbf{W}}(\mathbf{x}), y) = -\ln(h_{\mathbf{W}}(\mathbf{x})_y)$$

Equivalently:

$$e(h_{\mathbf{W}}(\mathbf{x}), y) = -\sum_{j=1}^{K} onehot(y)_j \ln(h_{\mathbf{W}}(\mathbf{x})_j)$$

Intuition: with correct class y,

- If $h_{\mathbf{W}}(\mathbf{x})_y \to 0$ then $e \to \infty$
- If $h_{\mathbf{W}}(\mathbf{x})_y \to 1$ then $e \to 0$

Cross-entropy Error on training data:

$$E_{train}(\mathbf{W}) = \frac{1}{N} \sum_{n=1}^{N} e(h_{\mathbf{W}}(\mathbf{x}^{(n)}), y^{(n)})$$

Step 2: $\min_{\mathbf{w}} E_{train}(\mathbf{w})$

We can use iterative gradient-based algorithms such as Gradient Descent

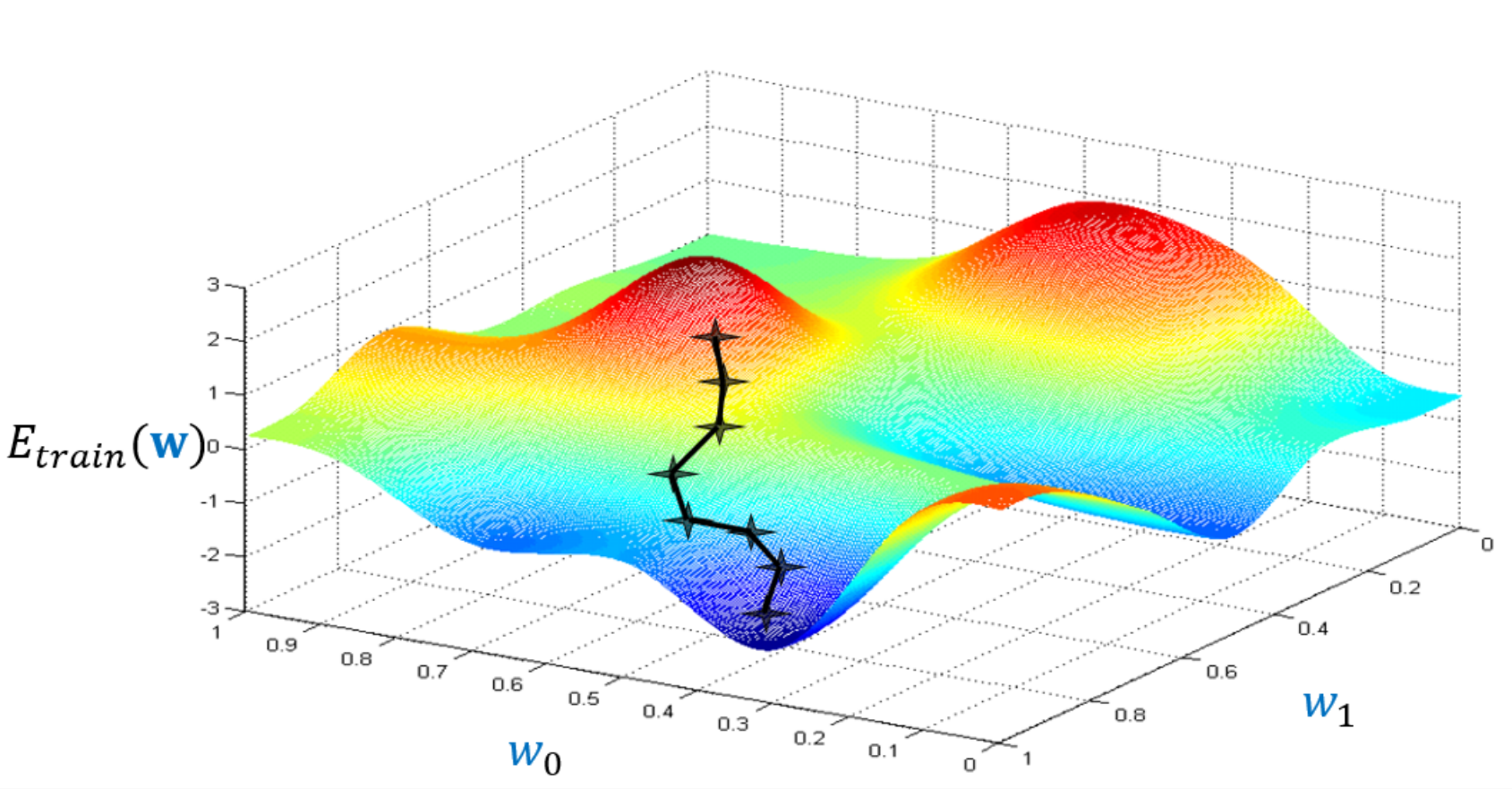


Image source: Andrew Ng, Machine Learning, MOOC

Step 2: $\min_{\mathbf{W}} E_{train}(\mathbf{W})$

Gradient Descent algorithm:

1. Initialize $\mathbf{W}$
2. Repeat until termination criteria are satisfied:

$$\mathbf{W} \leftarrow \mathbf{W} - \alpha \nabla E_{train}(\mathbf{W})$$

where:

- α is a number > 0; it is called "learning rate" and is a hyper-parameter which partially controlling step size
- $\nabla E_{train}(\mathbf{W})$ is a matrix containing partial derivaties of E_{train} w.r.t each w_{ij} of $\mathbf{W}$; it is called "gradient"

$$\nabla E_{train}(\mathbf{W}) = \begin{bmatrix} \frac{\partial E_{train}}{\partial w_{11}} & \cdots & \frac{\partial E_{train}}{\partial w_{1K}} \\ \vdots & \ddots & \vdots \\ \frac{\partial E_{train}}{\partial w_{(d+1)1}} & \cdots & \frac{\partial E_{train}}{\partial w_{(d+1)K}} \end{bmatrix}$$

$\nabla E_{train}(\mathbf{W}) = ?$

After doing difficult math with pencil and paper:

$$\frac{\partial E_{train}}{\partial w_{ij}} = \frac{1}{N} \sum_{n=1}^{N} x_i^{(n)} (h_{\mathbf{W}}(\mathbf{x}^{(n)})_j - onehot(y^{(n)})_j)$$

Based on this formula, we can write Python/Numpy code to compute $\nabla E_{train}(\mathbf{W})$ with 3 nested loops :-(

$$\nabla E_{train}(\mathbf{W}) = ?$$

After thinking hard from previous formula:

$$\nabla E_{train}(\mathbf{W}) = \frac{1}{N} \sum_{n=1}^{N} \mathbf{x}^{(n)} \left(h_{\mathbf{W}}(\mathbf{x}^{(n)}) - onehot(y^{(n)}) \right)^T$$

Based on this formula, we can write Python/Numpy code to compute $\nabla E_{train}(\mathbf{W})$ with 1 loop

$$\nabla E_{train}(\mathbf{W}) = ?$$

After thinking hard from previous formula (your job):

$$\nabla E_{train}(\mathbf{W}) = ...\mathbf{X}...\hat{\mathbf{Y}}...\mathbf{Y}...$$

where:

$$\mathbf{X} = \begin{bmatrix} — & \left(\mathbf{x}^{(1)}\right)^T & — \\ & \vdots & \\ — & \left(\mathbf{x}^{(N)}\right)^T & — \end{bmatrix} \quad \hat{\mathbf{Y}} = \begin{bmatrix} — & \left(h_{\mathbf{W}}(\mathbf{x}^{(1)})\right)^T & — \\ & \vdots & \\ — & \left(h_{\mathbf{W}}(\mathbf{x}^{(N)})\right)^T & — \end{bmatrix}$$

$$\mathbf{Y} = \begin{bmatrix} — & \left(onehot(y^{(1)})\right)^T & — \\ & \vdots & \\ — & \left(onehot(y^{(N)})\right)^T & — \end{bmatrix}$$

Now we can write Python/Numpy code to compute $\nabla E_{train}(\mathbf{W})$ with 0 loop :-)

Done!