Multi-Class Classification

Shusen Wang

Multi-Class Classification

Tasks

Methods

Algorithms

Multi-Class Classification

Example 1: face recognition.

- #classes = #people

Multi-Class Classification

Example 2: hand-written digit recognition.

- #classes = 10

Can we use linear regression?

Looks like "3" ➔ $f = \mathbf{w}^T\mathbf{x}$ is close to 3

$\mathbf{x}$

Looks like "8" ➔ $f = \mathbf{w}^T\mathbf{x}$ is close to 8

Can we use linear regression?

Looks like “3” $\rightarrow$ $f = \mathbf{w}^T\mathbf{x}$ is close to 3

$f = \mathbf{w}^T\mathbf{x}$ is 4.835

$\mathbf{x}$

Looks like “8” $\rightarrow$ $f = \mathbf{w}^T\mathbf{x}$ is close to 8

Can we use linear regression?

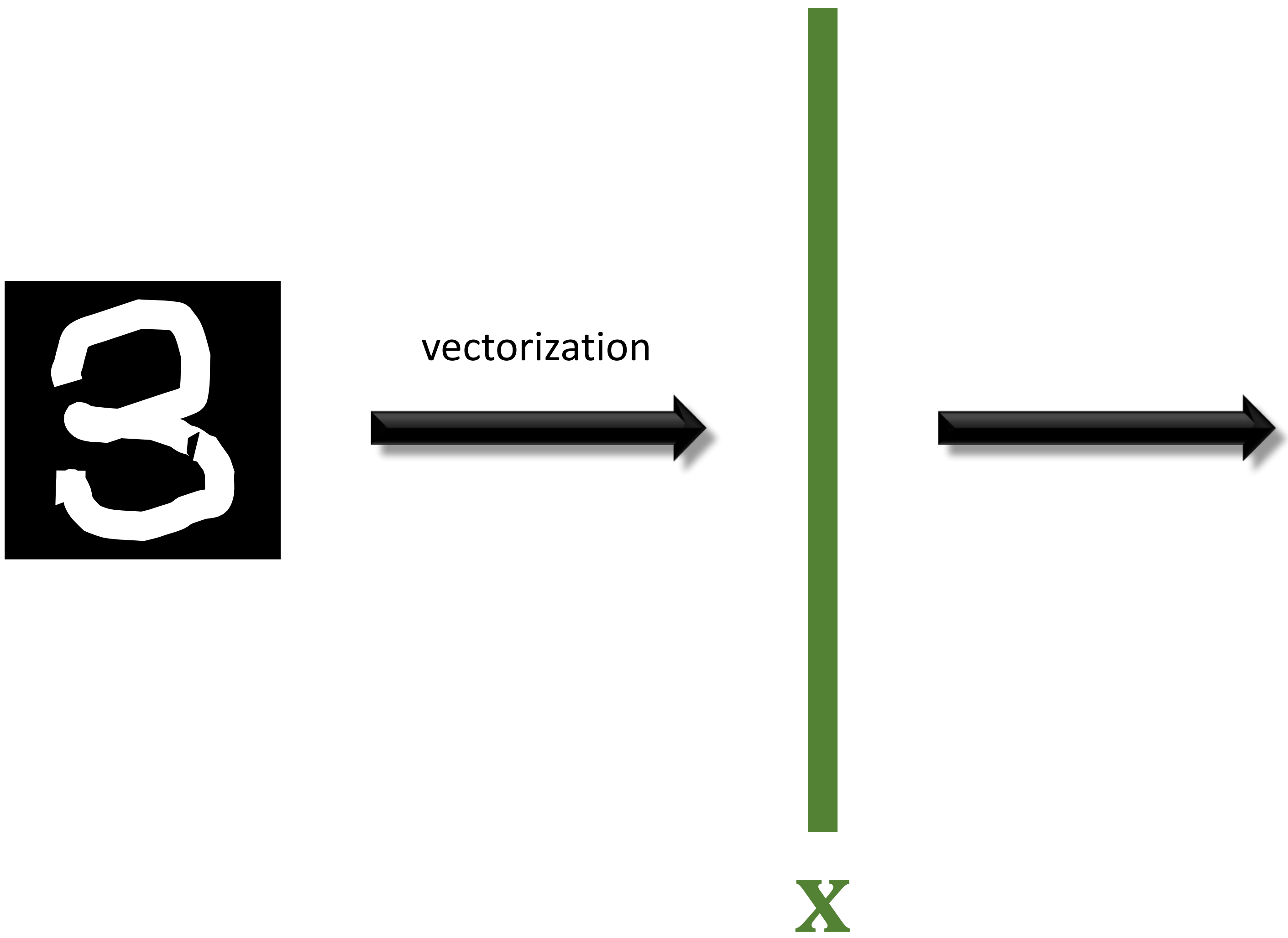
$\mathbf{x}$

Looks like "3" → $f = \mathbf{w}^T\mathbf{x}$ is close to 3

$f = \mathbf{w}^T\mathbf{x}$ is 4.835 $\xrightarrow{\text{rounding}}$ Linear regression believes $\mathbf{x}$ is "5"

Looks like "8" → $f = \mathbf{w}^T\mathbf{x}$ is close to 8

The Right Approach



$$\boldsymbol{\phi} = \mathbf{W}\mathbf{x} = \begin{bmatrix} -5 \\ -8 \\ 0.4 \\ 15 \\ -2 \\ 0.9 \\ 2.3 \\ -4.2 \\ 13.1 \\ 2.3 \end{bmatrix} \begin{array}{l} \\ \\ \\ \rightarrow \text{Digit “3”} \\ \\ \\ \\ \\ \rightarrow \text{Digit “8”} \\ \\ \end{array}$$

Preliminaries

准备工作(preliminary的名词复数)

One-Hot Encoding

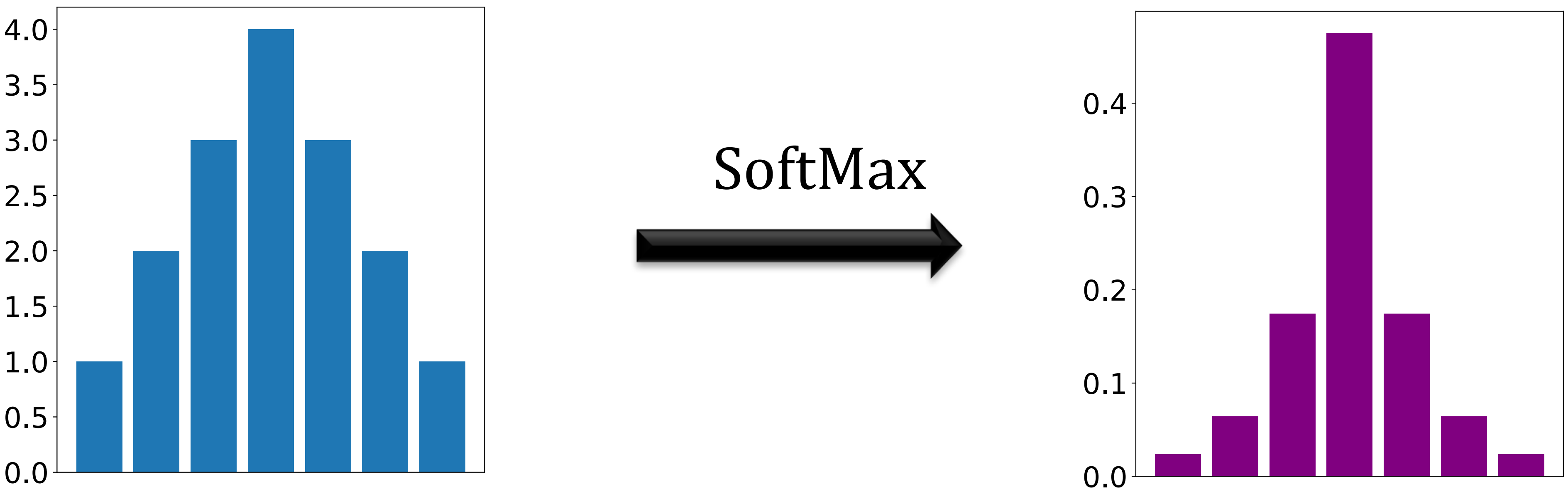
- #Class = 10 (e.g., in digit recognition).
- One-hot encode of $y = 3$:

$$\mathbf{y} \; = \; [0, 0, 0, 1, 0, 0, 0, 0, 0, 0]^T \; \in \; \{1,0\}^{10}$$

Softmax Function

- $\boldsymbol{\phi} \in \mathbb{R}^K$
- $\mathbf{p} = \mathrm{SoftMax}(\boldsymbol{\phi}) \in \mathbb{R}^K$; its entries are

$$p_k = \frac{\exp(\phi_k)}{\sum_{j=1}^{K} \exp(\phi_j)}, \text{ for } k = 1, \cdots, K.$$



Cross-Entropy

非负项

- The vectors $\mathbf{y}$ and $\mathbf{p}$ are K-dim vectors with nonnegative entries.

$$y_1 + \cdots + y_K = 1 \quad \text{and} \quad p_1 + \cdots + p_K = 1.$$

- Cross-entropy between $\mathbf{y}$ and $\mathbf{p}$:

公式：

$$\boxed{H(\mathbf{y}, \mathbf{p}) = -\sum_{l=1}^{K} y_l \log p_l \,.}$$

Cross-Entropy

- The vectors $\mathbf{y}$ and $\mathbf{p}$ are K-dim vectors with nonnegative entries.

$$y_1 + \cdots + y_K = 1 \quad \text{and} \quad p_1 + \cdots + p_K = 1.$$

- Cross-entropy between $\mathbf{y}$ and $\mathbf{p}$:

$$H(\mathbf{y}, \mathbf{p}) = -\sum_{l=1}^{K} y_l \log p_l \, .$$

- Cross-entropy measures the dissimilarity between $\mathbf{y}$ and $\mathbf{p}$.
- When used as loss function, $H(\mathbf{y}, \mathbf{p})$ can be replaced by

$$||\mathbf{y} - \mathbf{p}||_2^2 \quad \text{or} \quad ||\mathbf{y} - \mathbf{p}||_1.$$

Softmax Classifier: Model Formulation

Tasks

Methods

Algorithms

Softmax Classifier

Input: feature vectors $\mathbf{x}_1, \cdots, \mathbf{x}_n \in \mathbb{R}^d$ and labels $\mathbf{y}_1, \cdots, \mathbf{y}_n \in \mathbb{R}^K$.

n: #samples d: #features K: #classes

Remark: If the given labels are scalars $y_1, \cdots, y_n \in \{0, 1, \cdots, K-1\}$, turn them to K-dim vectors $\mathbf{y}_1, \cdots, \mathbf{y}_n \in \{0,1\}^K$ using one-hot encoding.

Example: One-hot encode of $y_i = 3$ (where K=10):

$$\mathbf{y}_i = [0, 0, 0, 1, 0, 0, 0, 0, 0, 0]^T \in \{1,0\}^{10}$$

Softmax Classifier

Input: feature vectors $\mathbf{x}_1, \cdots, \mathbf{x}_n \in \mathbb{R}^d$ and labels $\mathbf{y}_1, \cdots, \mathbf{y}_n \in \mathbb{R}^K$.

n: #samples d: #features K: #classes

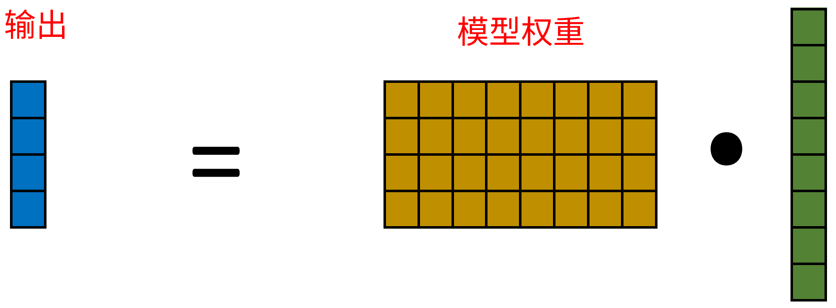
- $\boldsymbol{\phi} = \mathbf{W}\mathbf{x} \in \mathbb{R}^K$

Here, $\mathbf{x}$ is one of $\mathbf{x}_1, \cdots, \mathbf{x}_n \in \mathbb{R}^d$

和 y 是同维度的

输入

输出 模型权重



$\boldsymbol{\phi}$ $\mathbf{W}$ $\mathbf{x}$

$K \times 1$ $K \times d$ $d \times 1$

Softmax Classifier

Input: feature vectors $\mathbf{x}_1, \cdots, \mathbf{x}_n \in \mathbb{R}^d$ and labels $\mathbf{y}_1, \cdots, \mathbf{y}_n \in \mathbb{R}^K$.

n: #samples $\quad$ d: #features $\quad$ K: #classes

- $\boldsymbol{\phi} = \mathbf{W}\mathbf{x} \in \mathbb{R}^K$
- ϕ_k (the k-th entry of $\boldsymbol{\phi}$) indicates how likely $\mathbf{x}$ is in the k-th class.

$$\boldsymbol{\phi} = \mathbf{W}\mathbf{x} = \begin{bmatrix} -5 \\ -8 \\ 0.4 \\ 15 \\ -2 \\ 0.9 \\ 2.3 \\ -4.2 \\ 13.1 \\ 2.3 \end{bmatrix} \begin{array}{l} \\ \\ \\ \rightarrow \text{Digit “3”} \\ \\ \\ \\ \\ \rightarrow \text{Digit “8”} \\ \\ \end{array}$$

Softmax Classifier

Input: feature vectors $\mathbf{x}_1, \cdots, \mathbf{x}_n \in \mathbb{R}^d$ and labels $\mathbf{y}_1, \cdots, \mathbf{y}_n \in \mathbb{R}^K$.

n: #samples d: #features K: #classes

- $\boldsymbol{\phi} = \mathbf{W}\mathbf{x} \in \mathbb{R}^K$
- ϕ_k (the k-th entry of $\boldsymbol{\phi}$) indicates how likely $\mathbf{x}$ is in the k-th class.
- Softmax function: $p_k = \frac{\exp(\phi_k)}{\sum_{j=1}^{K} \exp(\phi_j)}$.

- $p_1 + \cdots + p_K = 1.$
- Thus $\mathbf{p} = [p_1, \cdots, p_K] \in \mathbb{R}^K$ is a distribution.

Softmax Classifier

Input: feature vectors $\mathbf{x}_1, \cdots, \mathbf{x}_n \in \mathbb{R}^d$ and labels $\mathbf{y}_1, \cdots, \mathbf{y}_n \in \mathbb{R}^K$.

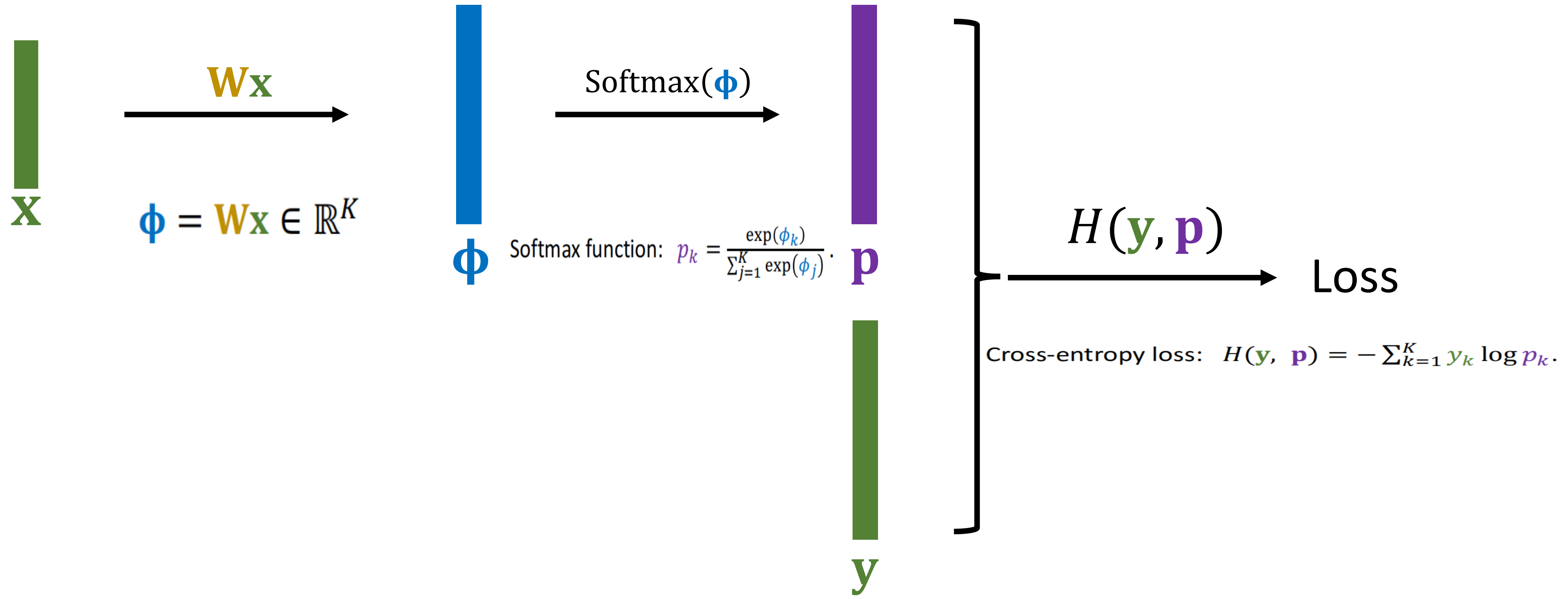
n: #samples d: #features K: #classes

- $\boldsymbol{\phi} = \mathbf{W}\mathbf{x} \in \mathbb{R}^K$
- ϕ_k (the k-th entry of $\boldsymbol{\phi}$) indicates how likely $\mathbf{x}$ is in the k-th class.
- Softmax function: $p_k = \frac{\exp(\phi_k)}{\sum_{j=1}^K \exp(\phi_j)}$.
- Cross-entropy loss: $H(\mathbf{y}, \mathbf{p}) = -\sum_{k=1}^K y_k \log p_k$.

Softmax Classifier

Input: feature vectors $\mathbf{x}_1, \cdots, \mathbf{x}_n \in \mathbb{R}^d$ and labels $\mathbf{y}_1, \cdots, \mathbf{y}_n \in \mathbb{R}^K$.

n: #samples d: #features K: #classes



Softmax Classifier

Input: feature vectors $\mathbf{x}_1, \cdots, \mathbf{x}_n \in \mathbb{R}^d$ and labels $\mathbf{y}_1, \cdots, \mathbf{y}_n \in \mathbb{R}^K$.

- Softmax classifier: $\min_{\mathbf{W}} \sum_{i=1}^{n} H(\mathbf{y}_i, \mathbf{p}_i)$

$$\boldsymbol{\phi}_i = \mathbf{W}\mathbf{x}_i \in \mathbb{R}^K, \quad \mathbf{p}_i = \mathrm{SoftMax}(\boldsymbol{\phi}_i), \quad \text{and} \quad H(\mathbf{y}_i, \mathbf{p}_i) = -\sum_{k=1}^{K} y_{i,k} \log\left(p_{i,k}\right).$$

Softmax function

Cross-entropy loss

Softmax Classifier

Input: feature vectors $\mathbf{x}_1, \cdots, \mathbf{x}_n \in \mathbb{R}^d$ and labels $\mathbf{y}_1, \cdots, \mathbf{y}_n \in \mathbb{R}^K$.

- Softmax classifier: $\boxed{\min_{\mathbf{W}} \sum_{i=1}^{n} H(\mathbf{y}_i, \mathbf{p}_i)}$

$$\boldsymbol{\phi}_i = \mathbf{W}\mathbf{x}_i \in \mathbb{R}^K, \quad \mathbf{p}_i = \mathrm{SoftMax}(\boldsymbol{\phi}_i), \quad \text{and} \quad H(\mathbf{y}_i, \mathbf{p}_i) = -\sum_{k=1}^{K} y_{i,k} \log\left(p_{i,k}\right).$$

- The role of minimizing $H(\mathbf{y}_i, \mathbf{p}_i)$ is making $\mathbf{p}_i$ similar to $\mathbf{y}_i$.
- $H(\mathbf{y}_i, \mathbf{p}_i)$ can be replaced by $||\mathbf{y}_i - \mathbf{p}_i||_2^2$.
- In practice, cross-entropy works better.

Softmax Classifier: Algorithm

Tasks

Methods

Algorithms

Softmax Classifier

$$\boldsymbol{\phi} = \mathbf{W}\mathbf{x} \in \mathbb{R}^K, \qquad p_q = \frac{e^{\phi_q}}{\sum_{j=1}^{K} e^{\phi_j}}, \qquad \text{and} \qquad H(\mathbf{y}, \mathbf{p}) = -\sum_{q=1}^{K} y_q \log(p_q).$$

Softmax Classifier

$\boldsymbol{\phi} = \mathbf{W}\mathbf{x} \in \mathbb{R}^K$, $\boxed{p_q = \frac{e^{\phi_q}}{\sum_{j=1}^{K} e^{\phi_j}},}$ and $H(\mathbf{y}, \mathbf{p}) = -\sum_{q=1}^{K} y_q \log\left(\boxed{p_q}\right).$

- $H(\mathbf{y}, \mathbf{p}) = -\sum_{q=1}^{K} y_q \phi_q + \log\left(\sum_{j=1}^{K} e^{\phi_j}\right) \cdot \sum_{q=1}^{K} y_q.$

Softmax Classifier

$$\boldsymbol{\phi} = \mathbf{W}\mathbf{x} \in \mathbb{R}^K, \qquad p_q = \frac{e^{\phi_q}}{\sum_{j=1}^K e^{\phi_j}}, \qquad \text{and} \qquad H(\mathbf{y}, \mathbf{p}) = -\sum_{q=1}^K y_q \log\left(p_q\right).$$

- $H(\mathbf{y}, \mathbf{p}) = -\sum_{q=1}^K y_q \phi_q \; + \; \log\left(\sum_{j=1}^K e^{\phi_j}\right) \cdot \underbrace{\boxed{\sum_{q=1}^K y_q.}}_{=1}$

Softmax Classifier

$\boldsymbol{\phi} = \mathbf{W}\mathbf{x} \in \mathbb{R}^K$ and $H(\mathbf{y}, \mathbf{p}) = -\sum_{j=1}^{K} y_j \phi_j + \log\left(\sum_{j=1}^{K} e^{\phi_j}\right)$.

- $H(\mathbf{y}, \mathbf{p}) = -\sum_{q=1}^{K} y_q \phi_q + \log\left(\sum_{j=1}^{K} e^{\phi_j}\right) \cdot \underbrace{\boxed{\sum_{q=1}^{K} y_q}}_{=1}$.

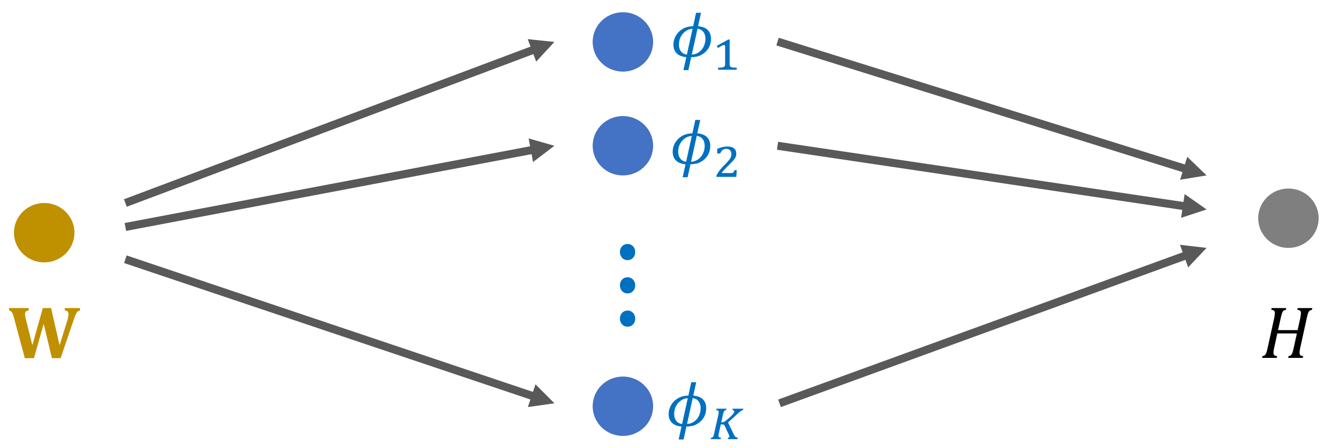
Softmax Classifier

$\boldsymbol{\phi} = \mathbf{W}\mathbf{x} \in \mathbb{R}^K$ and $H(\mathbf{y}, \mathbf{p}) = -\sum_{j=1}^{K} y_j \phi_j + \log\left(\sum_{j=1}^{K} e^{\phi_j}\right).$

$$\mathbf{W} \longrightarrow \boldsymbol{\phi} = \mathbf{W}\mathbf{x} \longrightarrow H$$

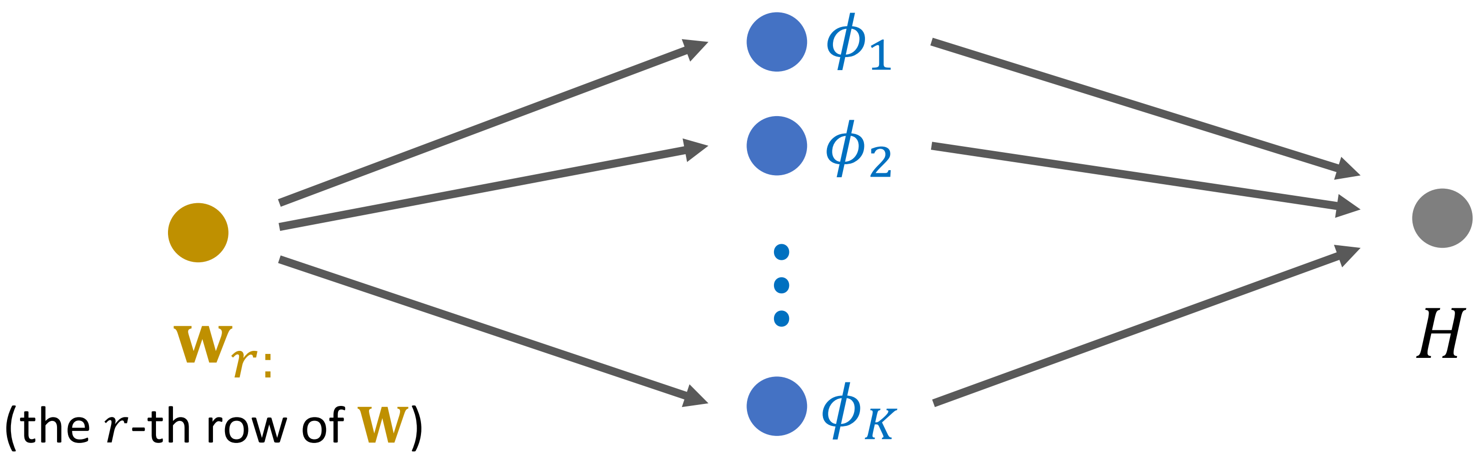
Softmax Classifier

$\boldsymbol{\phi} = \mathbf{W}\mathbf{x} \in \mathbb{R}^K$ and $H(\mathbf{y}, \mathbf{p}) = -\sum_{j=1}^{K} y_j \phi_j + \log\left(\sum_{j=1}^{K} e^{\phi_j}\right).$



Gradient

$\boldsymbol{\phi} = \mathbf{W}\mathbf{x} \in \mathbb{R}^K$ and $H(\mathbf{y}, \mathbf{p}) = -\sum_{j=1}^{K} y_j \phi_j + \log\left(\sum_{j=1}^{K} e^{\phi_j}\right).$

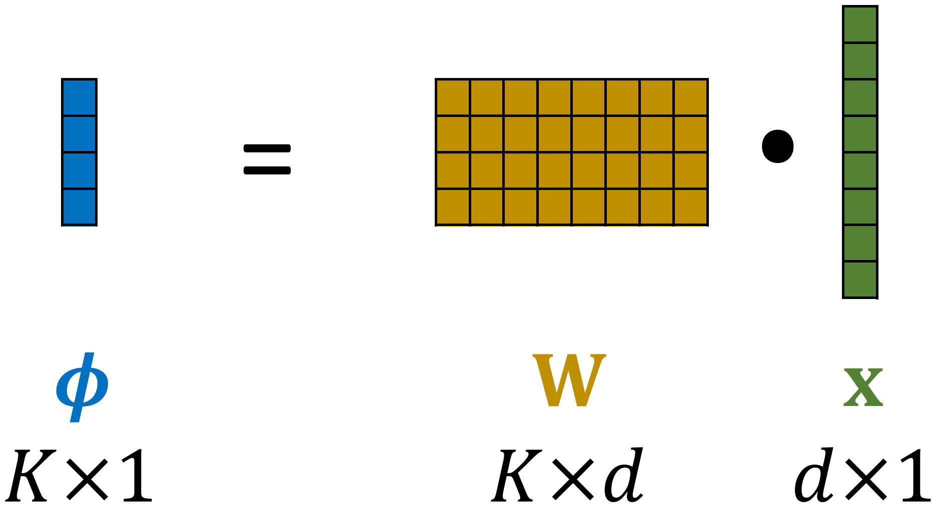


Chain rule: $\frac{\partial H}{\partial \mathbf{w}_{r:}} = \sum_{q=1}^{K} \frac{\partial \phi_q}{\partial \mathbf{w}_{r:}} \cdot \frac{\partial H}{\partial \phi_q}$

Gradient

$\boldsymbol{\phi} = \mathbf{W}\mathbf{x} \in \mathbb{R}^K$ and $H(\mathbf{y}, \mathbf{p}) = -\sum_{j=1}^{K} y_j \phi_j + \log\left(\sum_{j=1}^{K} e^{\phi_j}\right)$.

- $\phi_q = \mathbf{x}^T \mathbf{w}_{q:}$.



Gradient

$\boldsymbol{\phi} = \mathbf{W}\mathbf{x} \in \mathbb{R}^K$ and $H(\mathbf{y}, \mathbf{p}) = -\sum_{j=1}^{K} y_j \phi_j + \log\left(\sum_{j=1}^{K} e^{\phi_j}\right).$

- $\phi_q = \mathbf{x}^T \mathbf{w}_{q:}$.
- $\dfrac{\partial H}{\partial \mathbf{w}_{1:}} = \sum_{q=1}^{K} \dfrac{\partial \phi_q}{\partial \mathbf{w}_{1:}} \cdot \dfrac{\partial H}{\partial \phi_q}$

Gradient

$$\boldsymbol{\phi} = \mathbf{W}\mathbf{x} \in \mathbb{R}^K \qquad \text{and} \qquad H(\mathbf{y}, \mathbf{p}) = -\sum_{j=1}^{K} y_j \phi_j + \log\left(\sum_{j=1}^{K} e^{\phi_j}\right).$$

- $\phi_q = \mathbf{x}^T \mathbf{w}_{q:}$.
- $\frac{\partial H}{\partial \mathbf{w}_{1:}} = \sum_{q=1}^{K} \frac{\partial \phi_q}{\partial \mathbf{w}_{1:}} \cdot \frac{\partial H}{\partial \phi_q} = \frac{\partial \phi_1}{\partial \mathbf{w}_{1:}} \cdot \frac{\partial H}{\partial \phi_1} + \sum_{q \neq 1} \frac{\partial \phi_q}{\partial \mathbf{w}_{1:}} \cdot \frac{\partial H}{\partial \phi_q}.$

Gradient

$\boldsymbol{\phi} = \mathbf{W}\mathbf{x} \in \mathbb{R}^K$ and $H(\mathbf{y}, \mathbf{p}) = -\sum_{j=1}^{K} y_j \phi_j + \log\left(\sum_{j=1}^{K} e^{\phi_j}\right).$

- $\phi_q = \mathbf{x}^T \mathbf{w}_{q:}$.
- $\frac{\partial H}{\partial \mathbf{w}_{1:}} = \sum_{q=1}^{K} \frac{\partial \phi_q}{\partial \mathbf{w}_{1:}} \cdot \frac{\partial H}{\partial \phi_q} = \boxed{\frac{\partial \phi_1}{\partial \mathbf{w}_{1:}}} \cdot \frac{\partial H}{\partial \phi_1} + \sum_{q \neq 1} \frac{\partial \phi_q}{\partial \mathbf{w}_{1:}} \cdot \frac{\partial H}{\partial \phi_q}.$

$$\boxed{= \frac{\partial\, \mathbf{x}^T \mathbf{w}_{1:}}{\partial\, \mathbf{w}_{1:}} = \mathbf{x}}$$

Gradient

$\boldsymbol{\phi} = \mathbf{W}\mathbf{x} \in \mathbb{R}^K$ and $H(\mathbf{y}, \mathbf{p}) = -\sum_{j=1}^{K} y_j \phi_j + \log\left(\sum_{j=1}^{K} e^{\phi_j}\right)$.

- $\phi_q = \mathbf{x}^T \mathbf{w}_{q:}$.
- $\frac{\partial H}{\partial \mathbf{w}_{1:}} = \sum_{q=1}^{K} \frac{\partial \phi_q}{\partial \mathbf{w}_{1:}} \cdot \frac{\partial H}{\partial \phi_q} = \boxed{\frac{\partial \phi_1}{\partial \mathbf{w}_{1:}}} \cdot \frac{\partial H}{\partial \phi_1} + \sum_{q \neq 1} \boxed{\frac{\partial \phi_q}{\partial \mathbf{w}_{1:}}} \cdot \frac{\partial H}{\partial \phi_q}$.

$\boxed{= \frac{\partial \mathbf{x}^T \mathbf{w}_{1:}}{\partial \mathbf{w}_{1:}} = \mathbf{x}}$ $\boxed{= \frac{\partial \mathbf{x}^T \mathbf{w}_{q:}}{\partial \mathbf{w}_{1:}} = \mathbf{0}}$

Gradient

$$\boldsymbol{\phi} = \mathbf{W}\mathbf{x} \in \mathbb{R}^K \qquad \text{and} \qquad H(\mathbf{y}, \mathbf{p}) = -\sum_{j=1}^{K} y_j \phi_j + \log\left(\sum_{j=1}^{K} e^{\phi_j}\right).$$

- $\phi_q = \mathbf{x}^T \mathbf{w}_{q:}$.
- $\frac{\partial H}{\partial \mathbf{w}_{1:}} = \sum_{q=1}^{K} \frac{\partial \phi_q}{\partial \mathbf{w}_{1:}} \cdot \frac{\partial H}{\partial \phi_q} = \boxed{\frac{\partial \phi_1}{\partial \mathbf{w}_{1:}}} \cdot \frac{\partial H}{\partial \phi_1} + \sum_{q \neq 1} \boxed{\frac{\partial \phi_q}{\partial \mathbf{w}_{1:}}} \cdot \frac{\partial H}{\partial \phi_q}.$

$$\boxed{= \frac{\partial\, \mathbf{x}^T \mathbf{w}_{1:}}{\partial\, \mathbf{w}_{1:}} = \mathbf{x}} \qquad \boxed{= \frac{\partial\, \mathbf{x}^T \mathbf{w}_{q:}}{\partial\, \mathbf{w}_{1:}} = \mathbf{0}}$$

- $\frac{\partial H}{\partial \mathbf{w}_{1:}} = \mathbf{x} \cdot \frac{\partial H}{\partial \phi_1}.$

Gradient

$$\boldsymbol{\phi} = \mathbf{W}\mathbf{x} \in \mathbb{R}^K \qquad \text{and} \qquad H(\mathbf{y}, \mathbf{p}) = -\sum_{j=1}^{K} y_j \phi_j + \log\left(\sum_{j=1}^{K} e^{\phi_j}\right).$$

- $\frac{\partial H}{\partial \mathbf{w}_{q:}} = \mathbf{x} \cdot \frac{\partial H}{\partial \phi_q}$.

Gradient

$\boldsymbol{\phi} = \mathbf{W}\mathbf{x} \in \mathbb{R}^K$ and $H(\mathbf{y}, \mathbf{p}) = -\sum_{j=1}^{K} y_j \phi_j + \log\left(\sum_{j=1}^{K} e^{\phi_j}\right)$.

- $\dfrac{\partial H}{\partial \mathbf{w}_{q:}} = \mathbf{x} \cdot \dfrac{\partial H}{\partial \phi_q}$.
- $\dfrac{\partial H}{\partial \phi_q} = -y_q + \dfrac{e^{\phi_q}}{\sum_{j=1}^{K} e^{\phi_j}} = -y_q + p_q$.

Gradient

$\boldsymbol{\phi} = \mathbf{W}\mathbf{x} \in \mathbb{R}^K$ and $H(\mathbf{y}, \mathbf{p}) = -\sum_{j=1}^{K} y_j \phi_j + \log\left(\sum_{j=1}^{K} e^{\phi_j}\right).$

- $\frac{\partial H}{\partial \mathbf{w}_{q:}} = \mathbf{x} \cdot \boxed{\frac{\partial H}{\partial \phi_q}}.$
- $\boxed{\frac{\partial H}{\partial \phi_q}} = -y_q + \frac{e^{\phi_q}}{\sum_{j=1}^{K} e^{\phi_j}} = -y_q + p_q.$
- $\Rightarrow \frac{\partial H}{\partial \mathbf{w}_{q:}} = \mathbf{x} \cdot \left(-y_q + p_q\right)$

Gradient

$$\boldsymbol{\phi} = \mathbf{W}\mathbf{x} \in \mathbb{R}^K \qquad \text{and} \qquad H(\mathbf{y}, \mathbf{p}) = -\sum_{j=1}^{K} y_j \phi_j + \log\left(\sum_{j=1}^{K} e^{\phi_j}\right).$$

- $$\frac{\partial H}{\partial \mathbf{w}_{q:}} = \left(p_q - y_q\right) \cdot \mathbf{x}.$$

- $$\frac{\partial H}{\partial \mathbf{W}} = \begin{bmatrix} \frac{\partial H}{\partial \mathbf{w}_{1:}^T} \\ \frac{\partial H}{\partial \mathbf{w}_{2:}^T} \\ \vdots \\ \frac{\partial H}{\partial \mathbf{w}_{K:}^T} \end{bmatrix} = \begin{bmatrix} (p_1 - y_1) \cdot \mathbf{x}^T \\ (p_2 - y_2) \cdot \mathbf{x}^T \\ \vdots \\ (p_K - y_K) \cdot \mathbf{x}^T \end{bmatrix} = (\mathbf{p} - \mathbf{y}) \cdot \mathbf{x}^T.$$

终于 算出了 梯度：
损失函数 对w的偏导数

Stochastic Gradient Descent

Input: feature vectors $\mathbf{x}_1, \cdots, \mathbf{x}_n \in \mathbb{R}^d$ and labels $\mathbf{y}_1, \cdots, \mathbf{y}_n \in \mathbb{R}^K$.

- Model: $\min_{\mathbf{W}} \frac{1}{n} \sum_{i=1}^{n} H(\mathbf{y}_i, \mathbf{p}_i)$

  $\boldsymbol{\phi}_i = \mathbf{W}\mathbf{x}_i \in \mathbb{R}^K$, $\mathbf{p}_i = \text{SoftMax}(\boldsymbol{\phi}_i)$, and $H(\mathbf{y}_i, \mathbf{p}_i) = -\sum_{k=1}^{K} y_{i,k} \log(p_{i,k})$.

- A (stochastic) gradient: $\mathbf{G}_i = \frac{\partial H(\mathbf{y}_i, \mathbf{p}_i)}{\partial \mathbf{W}} = (\mathbf{p}_i - \mathbf{y}_i) \cdot \mathbf{x}_i^T \in \mathbb{R}^{K \times d}$.

随机梯度

Stochastic Gradient Descent

Input: feature vectors $\mathbf{x}_1, \cdots, \mathbf{x}_n \in \mathbb{R}^d$ and labels $\mathbf{y}_1, \cdots, \mathbf{y}_n \in \mathbb{R}^K$.

- Model: $\min_{\mathbf{W}} \frac{1}{n} \sum_{i=1}^{n} H(\mathbf{y}_i, \mathbf{p}_i)$

  $\boldsymbol{\phi}_i = \mathbf{W}\mathbf{x}_i \in \mathbb{R}^K$, $\mathbf{p}_i = \mathrm{SoftMax}(\boldsymbol{\phi}_i)$, and $H(\mathbf{y}_i, \mathbf{p}_i) = -\sum_{k=1}^{K} y_{i,k} \log(p_{i,k})$.

- A (stochastic) gradient: $\mathbf{G}_i = \frac{\partial H(\mathbf{y}_i, \mathbf{p}_i)}{\partial \mathbf{W}} = (\mathbf{p}_i - \mathbf{y}_i) \cdot \mathbf{x}_i^T \in \mathbb{R}^{K \times d}$.

SGD Algorithm:.

1. Randomly sample i from $\{1, 2, \cdots, n\}$.
2. Compute $\mathbf{G}_i$ using $(\mathbf{x}_i, y_i)$.
3. $\mathbf{W} \leftarrow \mathbf{W} - \alpha\, \mathbf{G}_i$.

Softmax Classifier: Train and Test

- Train (given feature vectors $\mathbf{x}_1, \cdots, \mathbf{x}_n \in \mathbb{R}^d$ and labels $\mathbf{y}_1, \cdots, \mathbf{y}_n \in \mathbb{R}^K$)
  - Compute $\mathbf{W}^\star = \underset{\mathbf{W}}{\operatorname{argmin}} \sum_{i=1}^{n} H(\mathbf{y}_i, \mathbf{p}_i)$ by SGD or other algorithms.
- Test (for a sample $\mathbf{x}' \in \mathbb{R}^d$)
  - $\boldsymbol{\phi}' = \mathbf{W}^\star \mathbf{x}' \in \mathbb{R}^K$.
  - Return the index of the largest entry of $\boldsymbol{\phi}'$.

Softmax Classifier: Train and Test

- Train (given feature vectors $\mathbf{x}_1, \cdots, \mathbf{x}_n \in \mathbb{R}^d$ and labels $\mathbf{y}_1, \cdots, \mathbf{y}_n \in \mathbb{R}^K$)
  - Compute $\mathbf{W}^\star = \underset{\mathbf{W}}{\operatorname{argmin}} \sum_{i=1}^{n} H(\mathbf{y}_i, \mathbf{p}_i)$ by SGD or other algorithms.
- Test (for a sample $\mathbf{x}' \in \mathbb{R}^d$)
  - $\boldsymbol{\phi}' = \mathbf{W}^\star \mathbf{x}' \in \mathbb{R}^K$.
  - Return the index of the largest entry of $\boldsymbol{\phi}'$.

What is the predicted class?

$$\boldsymbol{\phi}' = \begin{bmatrix} 0.9 \\ -5 \\ 10 \\ -20 \\ 0 \end{bmatrix}$$

Softmax Classifier: Train and Test

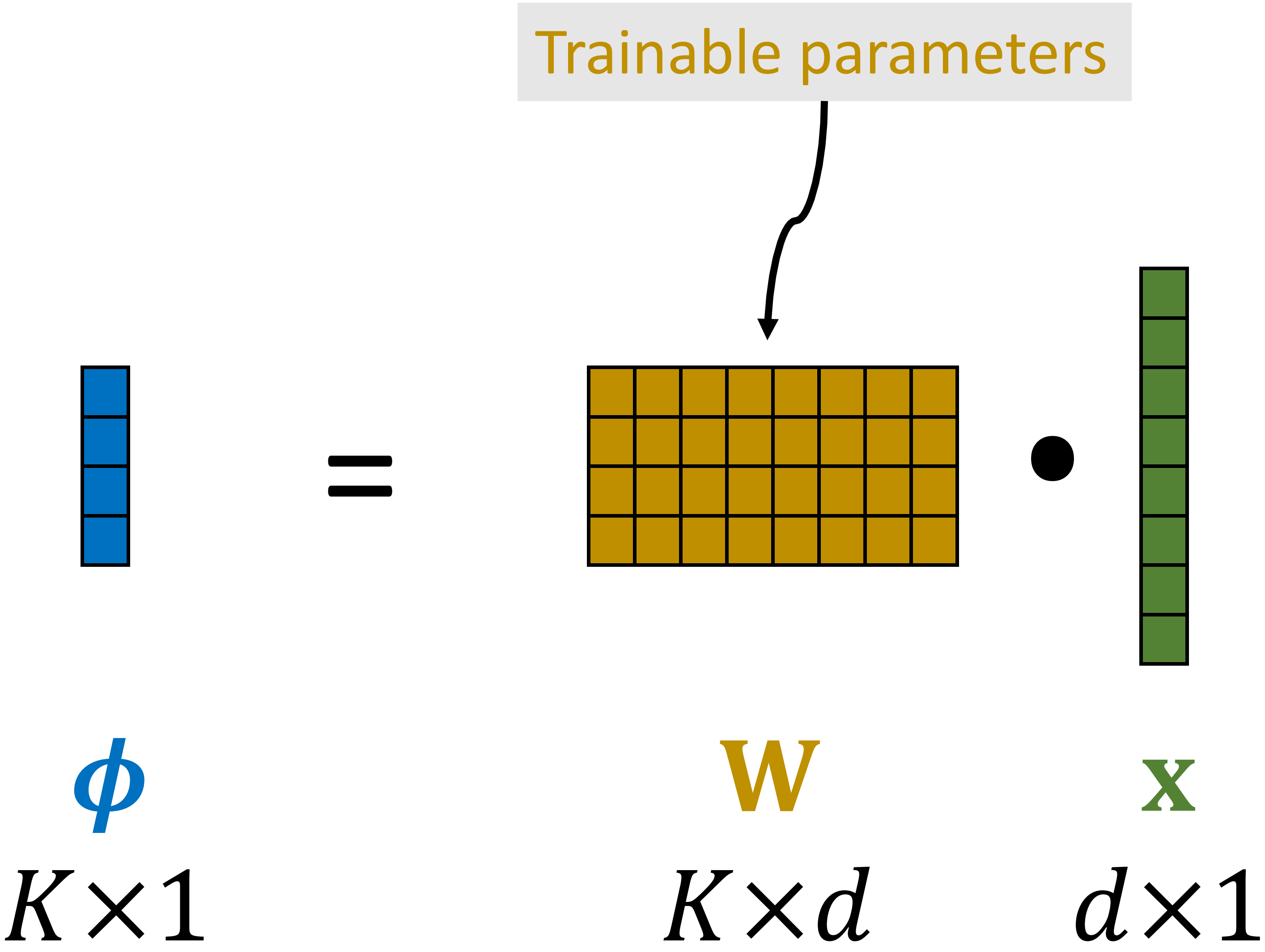
- Train (given feature vectors $\mathbf{x}_1, \cdots, \mathbf{x}_n \in \mathbb{R}^d$ and labels $\mathbf{y}_1, \cdots, \mathbf{y}_n \in \mathbb{R}^K$)
  - Compute $\mathbf{W}^\star = \underset{\mathbf{W}}{\operatorname{argmin}} \sum_{i=1}^{n} H(\mathbf{y}_i, \mathbf{p}_i)$ by SGD or other algorithms.
- Test (for a sample $\mathbf{x}' \in \mathbb{R}^d$)
  - $\boldsymbol{\phi}' = \mathbf{W}^\star \mathbf{x}' \in \mathbb{R}^K$.
  - Return the index of the largest entry of $\boldsymbol{\phi}'$.

What is the predicted class?

$$\boldsymbol{\phi}' = \begin{bmatrix} 0.9 \\ -5 \\ \boxed{10} \\ -20 \\ 0 \end{bmatrix}$$

Limitations of Softmax Classifier

#Parameter v.s. #Classes



#Parameter v.s. #Classes

- Suppose $\#\text{features} = 1K$.
- Suppose $\#\text{classes} = 10$ (e.g., digit recognition).
  - $1K \times 10 = 10K$ parameters.
- Suppose $\#\text{classes} = 1K$ (e.g., ImageNet image recognition).
  - $1K \times 1K = 1M$ parameters.
- Suppose $\#\text{classes} = 1M$ (e.g., face recognition).
  - $1K \times 1M = 1G$ parameters ➔ Heavy computation and memory costs.

沉重的计算和内存成本

#Parameter v.s. #Classes

- Suppose $\#\text{features} = 1K$.
- Suppose $\#\text{classes} = 10$ (e.g., digit recognition).
  - $1K \times 10 = 10K$ parameters.
- Suppose $\#\text{classes} = 1K$ (e.g., ImageNet image recognition).
  - $1K \times 1K = 1M$ parameters.
- Suppose $\#\text{classes} = 1M$ (e.g., face recognition).
  - $1K \times 1M = 1G$ parameters ➔ Heavy computation and memory costs.
- What if $\#\text{classes} = 1G$? (E.g., face recognition for all the Chinese citizens.) 例如，对所有中国公民的人脸识别，这是无法胜任的