Multi-Class Classification

Shusen Wang

Multi-Class Classification

Tasks   **Methods**   **Algorithms**

Multi-Class Classification

Example 1: face recognition.
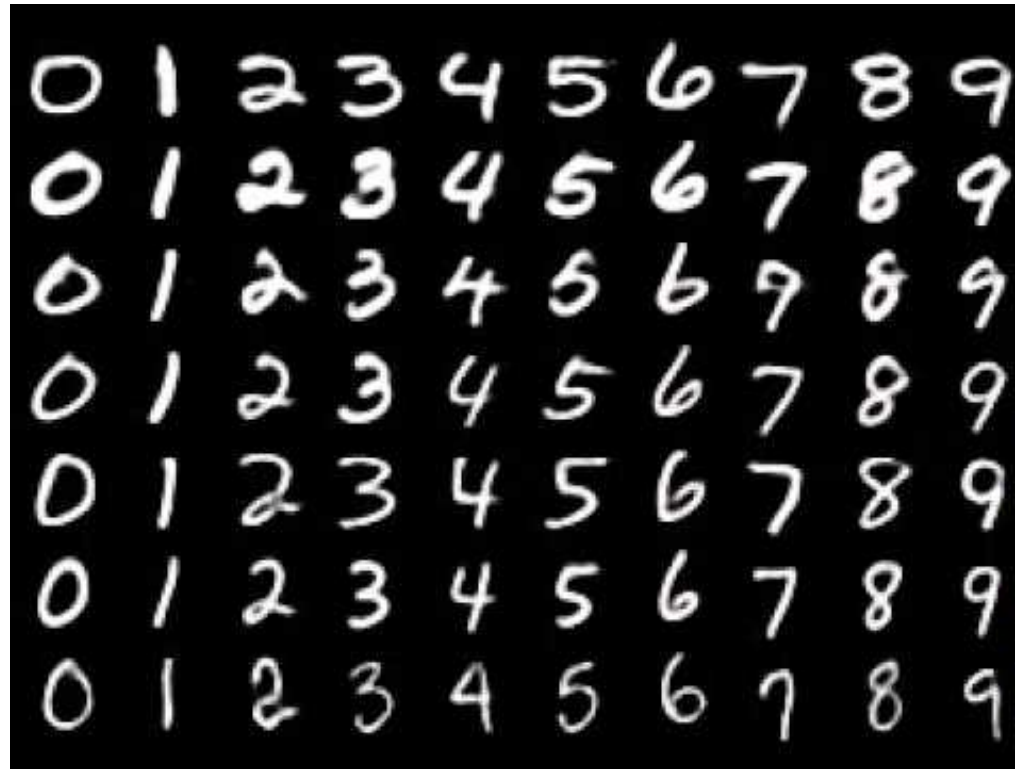
- #classes = #people

Multi-Class Classification

Example 2: hand-written digit recognition.
- #classes = 10

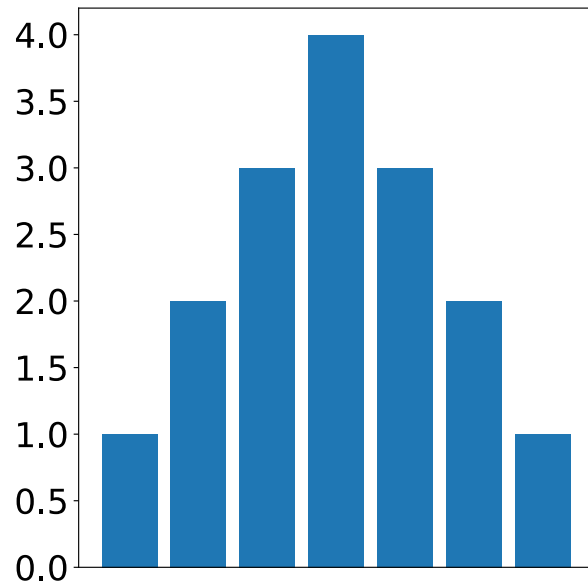# Softmax Classifier: Preliminaries

One-Hot Encoding

- #Class = 10 (e.g., in digit recognition).

- One-hot encode of $y = 3$:

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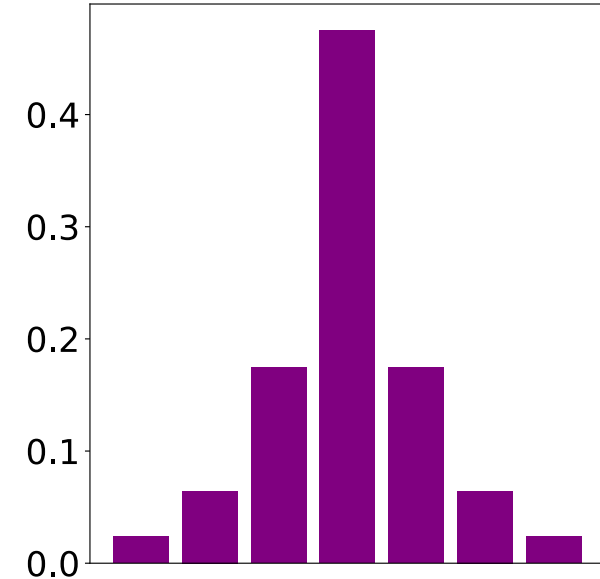
Softmax Function

- $\boldsymbol{\phi} \in \mathbb{R}^K$

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

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

Cross-Entropy

- The vectors **y** and **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 **y** and **p**:

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

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

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

Softmax Classifier

Tasks

Methods

Algorithms

Softmax Classifier

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 $\qquad$ d: #features $\qquad$ K: #classes

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

- $\phi_{i,k}$ (the k-th entry of $\boldsymbol{\phi}_i$) indicates how likely $\mathbf{x}_i$ is in the k-th class.

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

$$K{\times}1 \qquad K{\times}d \qquad d{\times}1$$

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　　　　　d: #features　　　　　K: #classes

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

- $\phi_{i,k}$ (the k-th entry of $\boldsymbol{\phi}_i$) indicates how likely $\mathbf{x}_i$ is in the k-th class.

- Softmax function: $p_{i,k} = \dfrac{\exp(\phi_{i,k})}{\sum_{j=1}^{K} \exp(\phi_{i,j})}$.

- $p_{i,1} + \cdots + p_{i,K} = 1$.
- Thus $\mathbf{p}_i = \left[p_{i,1}, \cdots, p_{i,K} \right] \in \mathbb{R}^K$ 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

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

- $\phi_{i,k}$ (the k-th entry of $\boldsymbol{\phi}_i$) indicates how likely $\mathbf{x}_i$ is in the k-th class.

- Softmax function: $p_{i,k} = \dfrac{\exp(\phi_{i,k})}{\sum_{j=1}^{K} \exp(\phi_{i,j})}$.

- Cross-entropy loss: $H(\mathbf{y}_i, \ \mathbf{p}_i) = -\sum_{k=1}^{K} y_{i,k} \log p_{i,k}$.

Softmax Classifier

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

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

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

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

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

- 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 $\left\| \mathbf{y}_i - \mathbf{p}_i \right\|_2^2$.

- In practice, cross-entropy works better.

$$H(\mathbf{y}, \mathbf{p}) \in \mathbb{R}_+$$

Loss function
(distance between y and p)

$$\mathbf{p} = \text{softmax}(\boldsymbol{\phi}) \in \mathbb{R}^K$$

Activation function "Softmax"
(any vector ==> prob distribution)

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

$$\mathbf{x} \in \mathbb{R}^d$$

$$\mathbf{W} \in \mathbb{R}^{K \times d}$$

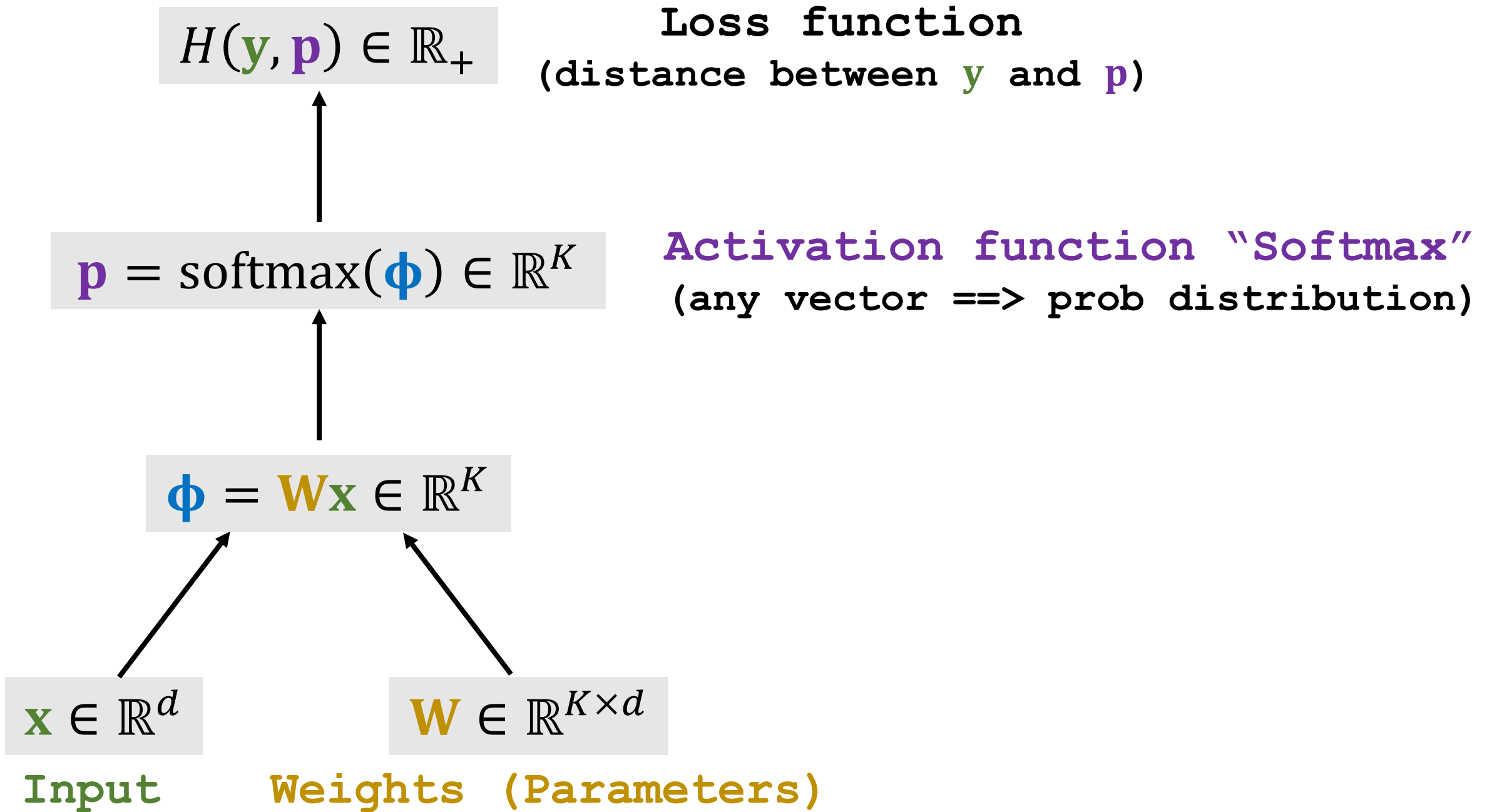**Input**   **Weights (Parameters)**

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

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

Tasks	**Methods**	**Algorithms**
	Softmax	Gradient Descent (GD)
Binary Classification	KNN	Accelerated GD
Multi-Class Classification	Neural Networks	Stochastic GD

$$H(\mathbf{y}, \mathbf{p}) \in \mathbb{R}_+$$

Loss

$$\mathbf{p} = \text{softmax}(\boldsymbol{\phi}) \in \mathbb{R}^K$$

Activation

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

$$\mathbf{x} \in \mathbb{R}^d$$

Input

$$\mathbf{W} \in \mathbb{R}^{K \times d}$$

Weights (Parameters)

$$H(\mathbf{y}, \mathbf{p}) \in \mathbb{R}_+$$

$$\mathbf{p} = \mathrm{softmax}(\boldsymbol{\phi}) \in \mathbb{R}^K$$

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

$$\mathbf{x} \in \mathbb{R}^d \qquad \mathbf{W} \in \mathbb{R}^{K \times d}$$

$$\frac{\partial H}{\partial \mathbf{p}} \in \mathbb{R}^K$$

$$\frac{\partial H}{\partial \boldsymbol{\phi}} = \frac{\partial \mathbf{p}}{\partial \boldsymbol{\phi}} \cdot \frac{\partial H}{\partial \mathbf{p}} \in \mathbb{R}^K$$

$$\frac{\partial H}{\partial \mathbf{W}} = \frac{\partial \boldsymbol{\phi}}{\partial \mathbf{W}} \cdot \frac{\partial H}{\partial \boldsymbol{\phi}} \in \mathbb{R}^{K \times d}$$
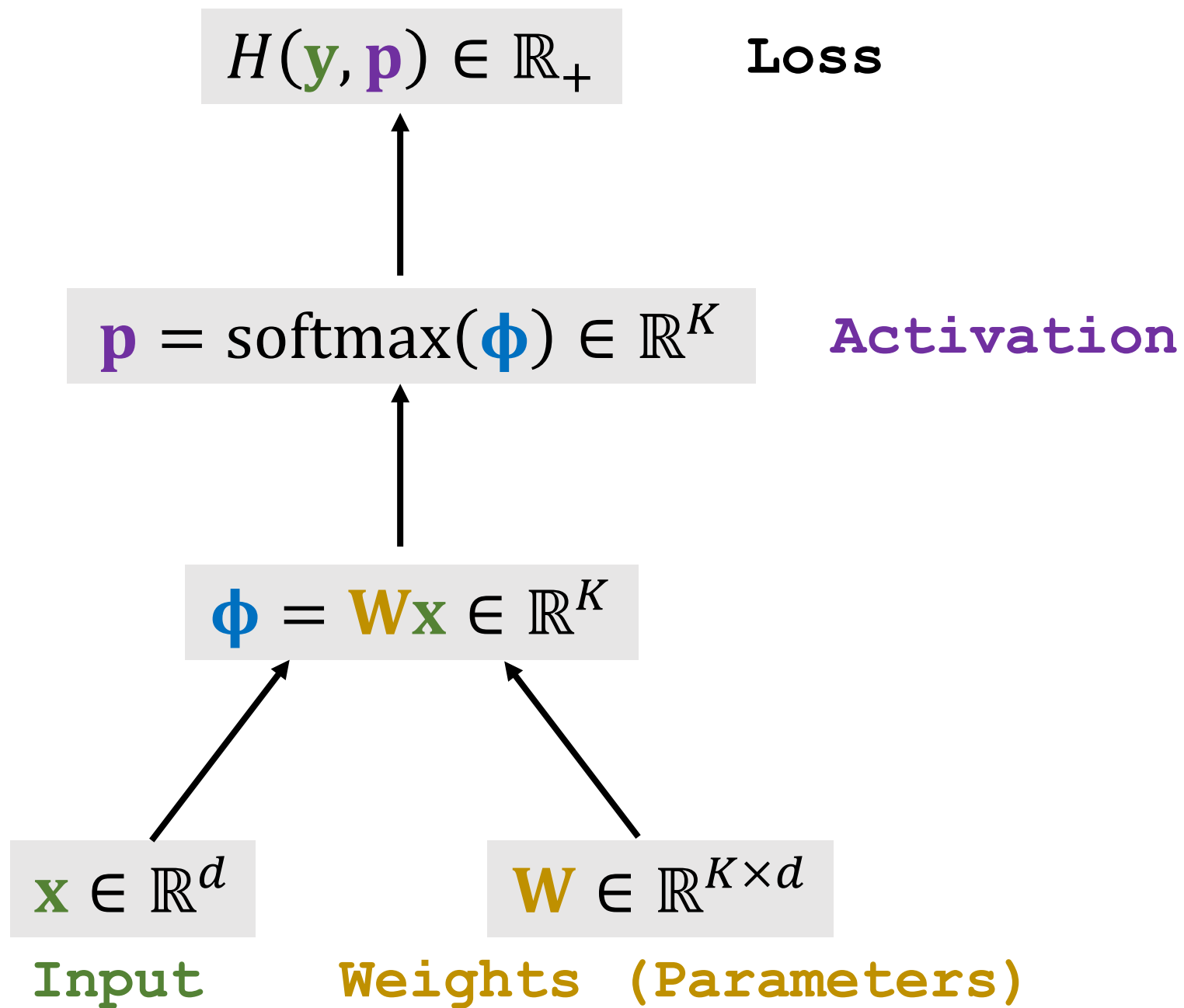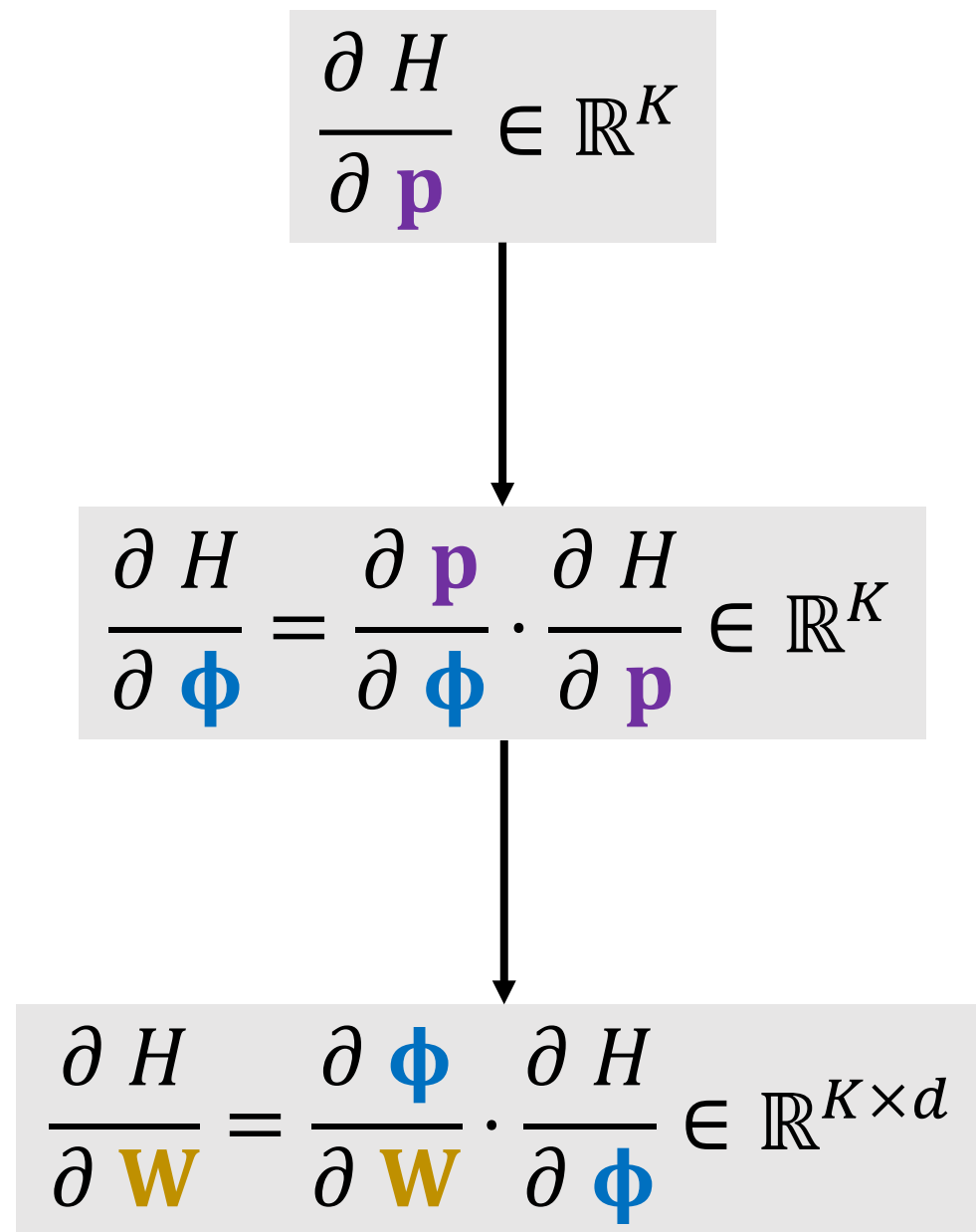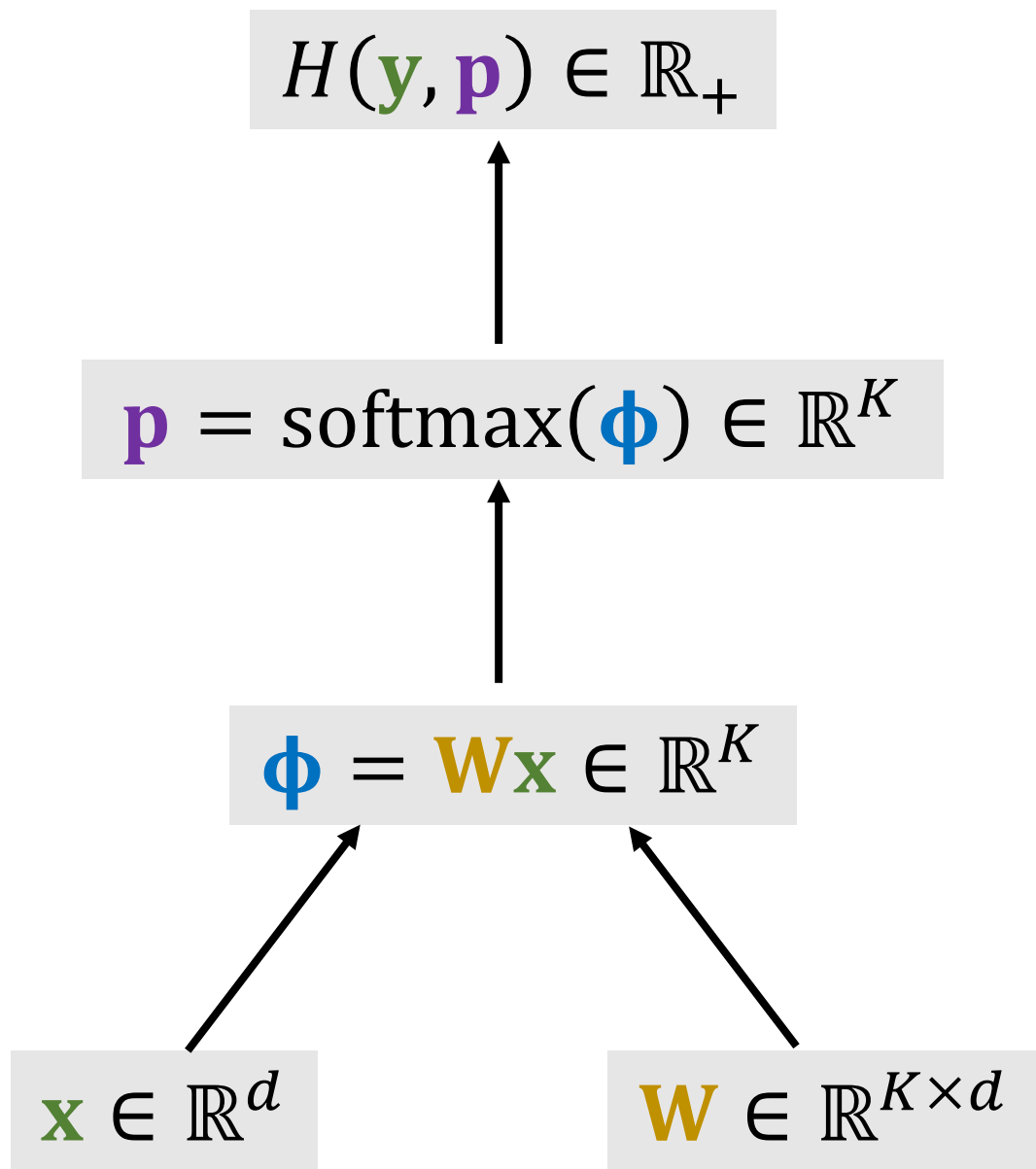
Gradient of the Objective Function

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

- Leave out the subscript i (which indexes a sample.)

Gradient of the Objective Function

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

- $H(\mathbf{y}, \mathbf{p}) = -\sum_{k=1}^{K} y_k \phi_k \; + \; \log\left(\sum_{j=1}^{K} e^{\phi_j}\right) \cdot \sum_{k=1}^{K} y_k.$

Gradient of the Objective Function

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

- $\dfrac{\partial H}{\partial \phi_k} = -y_k + \dfrac{1}{\sum_{j=1}^{K} e^{\phi_j}} \cdot e^{\phi_k} \cdot \sum_{k=1}^{K} y_k$

Gradient of the Objective Function

$$\boldsymbol{\phi} = \mathbf{Wx} \in \mathbb{R}^K \quad \text{and} \quad H(\mathbf{y}, \mathbf{p}) = -\sum_{k=1}^K y_k \phi_k + \log\left(\sum_{j=1}^K e^{\phi_j}\right) \cdot \sum_{k=1}^K y_k.$$

- $\dfrac{\partial H}{\partial \phi_k} = -y_k + \boxed{\dfrac{1}{\sum_{j=1}^K e^{\phi_j}} \cdot e^{\phi_k}} \cdot \boxed{\sum_{k=1}^K y_k} = -y_k + p_k.$

$$= p_k \qquad\qquad = 1$$

Gradient of the Objective Function

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

- $\dfrac{\partial H}{\partial \phi_k} = -y_k + \dfrac{1}{\sum_{j=1}^{K} e^{\phi_j}} \cdot e^{\phi_k} \cdot \sum_{k=1}^{K} y_k = -y_k + p_k.$

- $\dfrac{\partial \phi_k}{\partial \mathbf{w}_{j:}} = \dfrac{\partial \mathbf{w}_{k:}^T \mathbf{x}}{\partial \mathbf{w}_{j:}} = \begin{cases} \mathbf{x}, & \text{if } k = j; \\ \mathbf{0}, & \text{otherwise.} \end{cases}$ (Here $\mathbf{w}_{j:}$ is the j-th row of $\mathbf{W}$.)

- ➜ $\quad \dfrac{\partial H}{\partial \mathbf{w}_{j:}} = \sum_{k=1}^{K} \dfrac{\partial \phi_k}{\partial \mathbf{w}_{j:}} \cdot \dfrac{\partial H}{\partial \phi_k}$

Gradient of the Objective Function

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

- $\boxed{\dfrac{\partial H}{\partial \phi_k}} = -y_k + \dfrac{1}{\sum_{j=1}^{K} e^{\phi_j}} \cdot e^{\phi_k} \cdot \sum_{k=1}^{K} y_k = -y_k + p_k.$

- $\boxed{\dfrac{\partial \phi_k}{\partial \mathbf{w}_{j:}}} = \dfrac{\partial \mathbf{w}_{k:}^T \mathbf{x}}{\partial \mathbf{w}_{j:}} = \begin{cases} \mathbf{x}, & \text{if } k = j; \\ \mathbf{0}, & \text{otherwise.} \end{cases}$ (Here $\mathbf{w}_{j:}$ is the j-th row of $\mathbf{W}$.)

- ➔ $\dfrac{\partial H}{\partial \mathbf{w}_{j:}} = \sum_{k=1}^{K} \boxed{\dfrac{\partial \phi_k}{\partial \mathbf{w}_{j:}}} \cdot \boxed{\dfrac{\partial H}{\partial \phi_k}} = \left(p_j - y_j\right) \mathbf{x} \in \mathbb{R}^d.$

- ➔ $\dfrac{\partial H}{\partial \mathbf{W}} = (\mathbf{p} - \mathbf{y}) \cdot \mathbf{x}^T \in \mathbb{R}^{K \times d}.$

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: $\min\limits_{\mathbf{W}} \sum_{i=1}^{n} H(\mathbf{y}_i, \mathbf{p}_i)$

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

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

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

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

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

Algorithm: Iteratively update $\mathbf{W}$ using the (stochastic) gradient.

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 some algorithm, e.g., AGD, SGD, etc.

- 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 some algorithm, e.g., AGD, SGD, etc.

- 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 $\mathbf{W}^\star = \underset{\mathbf{W}}{\text{argmin}} \sum_{i=1}^{n} H(\mathbf{y}_i, \mathbf{p}_i)$ by some algorithm, e.g., AGD, SGD, etc.

- 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

Trainable parameters

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

$\boldsymbol{\phi}$
$K \times 1$

$\mathbf{W}$
$K \times d$

$\mathbf{x}$
$d \times 1$

#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.

#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.)