

Multi-Class Classification

Shusen Wang

Multi-Class Classification

Tasks

Methods

Algorithms

Multi-Class Classification

Example 1: face recognition.

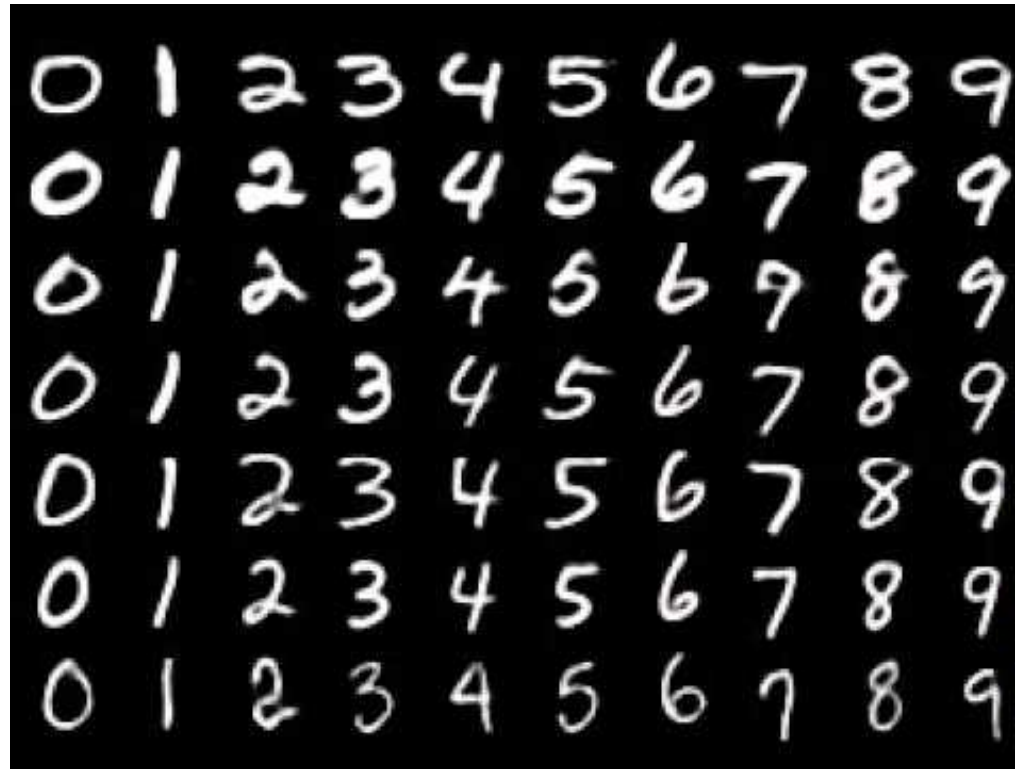
- #classes = #people



Multi-Class Classification

Example 2: hand-written digit recognition.

- #classes = 10



Softmax Classifier: Preliminaries

One-Hot Encoding

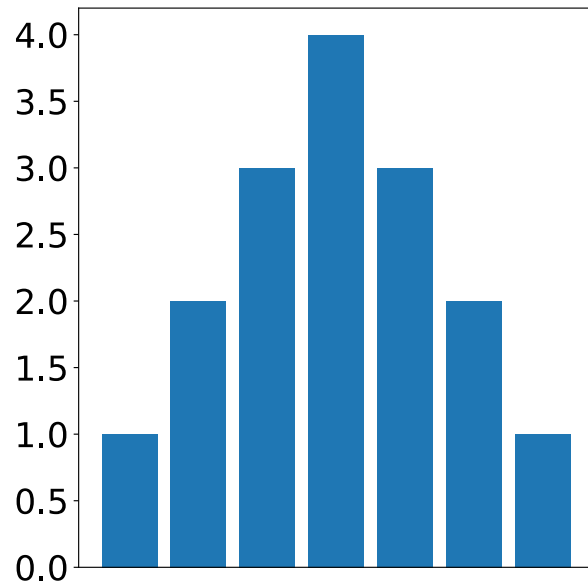
- #Class = 10 (e.g., in digit recognition).
- One-hot encode of $y = 3$:

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

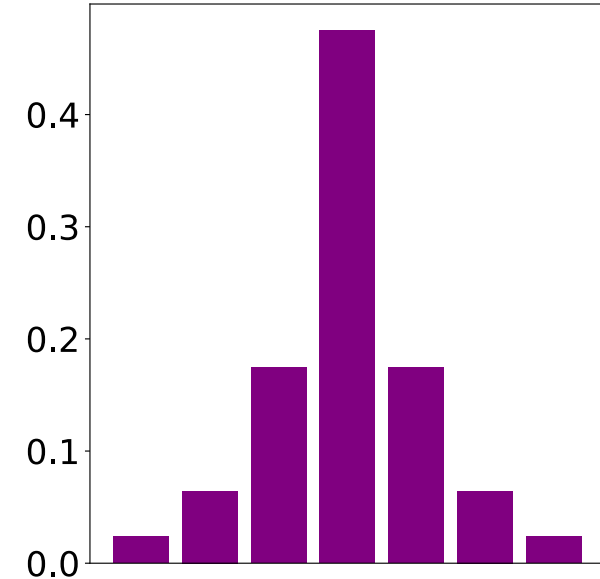
Softmax Function

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

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



SoftMax



Cross-Entropy

- The vectors \mathbf{y} and \mathbf{p} are both K -dim.

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

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

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

Softmax Classifier

Tasks

Methods

Algorithms

Softmax Classifier

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

n : #samples

d : #features

K : #classes

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

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

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

Softmax Classifier

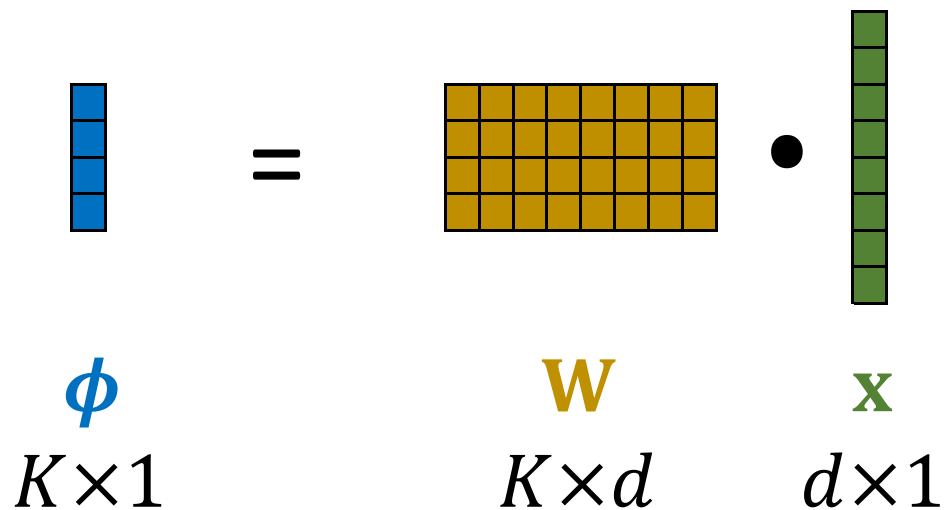
Input: feature vectors $\mathbf{x}_1, \dots, \mathbf{x}_n \in \mathbb{R}^d$ and labels $\mathbf{y}_1, \dots, \mathbf{y}_n \in \mathbb{R}^K$.

n : #samples

d : #features

K : #classes

- $\phi_i = \mathbf{W}\mathbf{x}_i \in \mathbb{R}^K$
- $\phi_{i,k}$ (the k -th entry of ϕ_i) indicates how likely \mathbf{x}_i is in the k -th class.


$$\begin{matrix} \begin{matrix} \text{blue column} \\ \phi \\ K \times 1 \end{matrix} & = & \begin{matrix} \text{yellow grid} \\ \mathbf{W} \\ K \times d \end{matrix} & \bullet & \begin{matrix} \text{green column} \\ \mathbf{x} \\ d \times 1 \end{matrix} \end{matrix}$$

Softmax Classifier

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

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

- $p_{i,1} + \dots + p_{i,K} = 1$.
- Thus $\mathbf{p}_i = [p_{i,1}, \dots, p_{i,K}] \in \mathbb{R}^K$ is a distribution.

Softmax Classifier

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

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: $p_{i,k} = \frac{\exp(\phi_{i,k})}{\sum_{j=1}^K \exp(\phi_{i,j})}$.
- 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 $\mathbf{x}_1, \dots, \mathbf{x}_n \in \mathbb{R}^d$ and labels $\mathbf{y}_1, \dots, \mathbf{y}_n \in \mathbb{R}^K$.

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

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

Softmax function



Cross-entropy loss



Softmax Classifier

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

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

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

Tasks

Binary Classification

Multi-Class Classification

Methods

Softmax

KNN

Neural Networks

Algorithms

Gradient Descent (GD)

Accelerated GD

Stochastic GD

Gradient of the Objective Function

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

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

Gradient of the Objective Function

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

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

Gradient of the Objective Function

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

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

Gradient of the Objective Function

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

$$\begin{aligned} \bullet \quad \frac{\partial H}{\partial \phi_k} &= -y_k + \frac{1}{\sum_{j=1}^K e^{\phi_j}} \cdot e^{\phi_k} \cdot \sum_{k=1}^K y_k = -y_k + p_k. \\ &\qquad\qquad\qquad = p_k \qquad\qquad\qquad = 1 \end{aligned}$$

Gradient of the Objective Function

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

- $\frac{\partial H}{\partial \phi_k} = -y_k + \frac{1}{\sum_{j=1}^K e^{\phi_j}} \cdot e^{\phi_k} \cdot \sum_{k=1}^K y_k = -y_k + p_k.$
- $\frac{\partial \phi_k}{\partial \mathbf{w}_{j:}} = \frac{\partial \mathbf{w}_{k:}^T \mathbf{x}}{\partial \mathbf{w}_{j:}} = \begin{cases} \mathbf{x}, & \text{if } k = j; \\ \mathbf{0}, & \text{otherwise.} \end{cases} \quad (\text{Here } \mathbf{w}_{j:} \text{ is the } j\text{-th row of } \mathbf{W}.)$
- $\rightarrow \frac{\partial H}{\partial \mathbf{w}_{j:}} = \sum_{k=1}^K \frac{\partial \phi_k}{\partial \mathbf{w}_{j:}} \cdot \frac{\partial H}{\partial \phi_k}$

Gradient of the Objective Function

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

- $\frac{\partial H}{\partial \phi_k} = -y_k + \frac{1}{\sum_{j=1}^K e^{\phi_j}} \cdot e^{\phi_k} \cdot \sum_{k=1}^K y_k = -y_k + p_k.$
- $\frac{\partial \phi_k}{\partial \mathbf{w}_{j:}} = \frac{\partial \mathbf{w}_{k:}^T \mathbf{x}}{\partial \mathbf{w}_{j:}} = \begin{cases} \mathbf{x}, & \text{if } k = j; \\ \mathbf{0}, & \text{otherwise.} \end{cases} \quad (\text{Here } \mathbf{w}_{j:} \text{ is the } j\text{-th row of } \mathbf{W}.)$
- $\rightarrow \frac{\partial H}{\partial \mathbf{w}_{j:}} = \sum_{k=1}^K \frac{\partial \phi_k}{\partial \mathbf{w}_{j:}} \cdot \frac{\partial H}{\partial \phi_k} = (p_j - y_j) \mathbf{x} \in \mathbb{R}^d.$
- $\rightarrow \frac{\partial H}{\partial \mathbf{W}} = (\mathbf{p} - \mathbf{y}) \cdot \mathbf{x}^T \in \mathbb{R}^{K \times d}.$

Softmax Classifier

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

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

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

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

Softmax Classifier

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

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

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

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

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

Softmax Classifier: Train and Test

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

Limitations of Softmax Classifier

#Parameter v.s. #Classes

- Suppose #features = $1K$.
- Suppose #classes = 10 (e.g., digit recognition).
 - $1K \times 10 = 10K$ parameters.
- Suppose #classes = $1K$ (e.g., ImageNet image recognition).
 - $1K \times 1K = 1M$ parameters.
- Suppose #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 \rightarrow Heavy computation and memory costs.
- What if $\text{\#classes} = 1G$? (E.g., face recognition for all the Chinese citizens.)