Multi-class Classification (using Softmax)

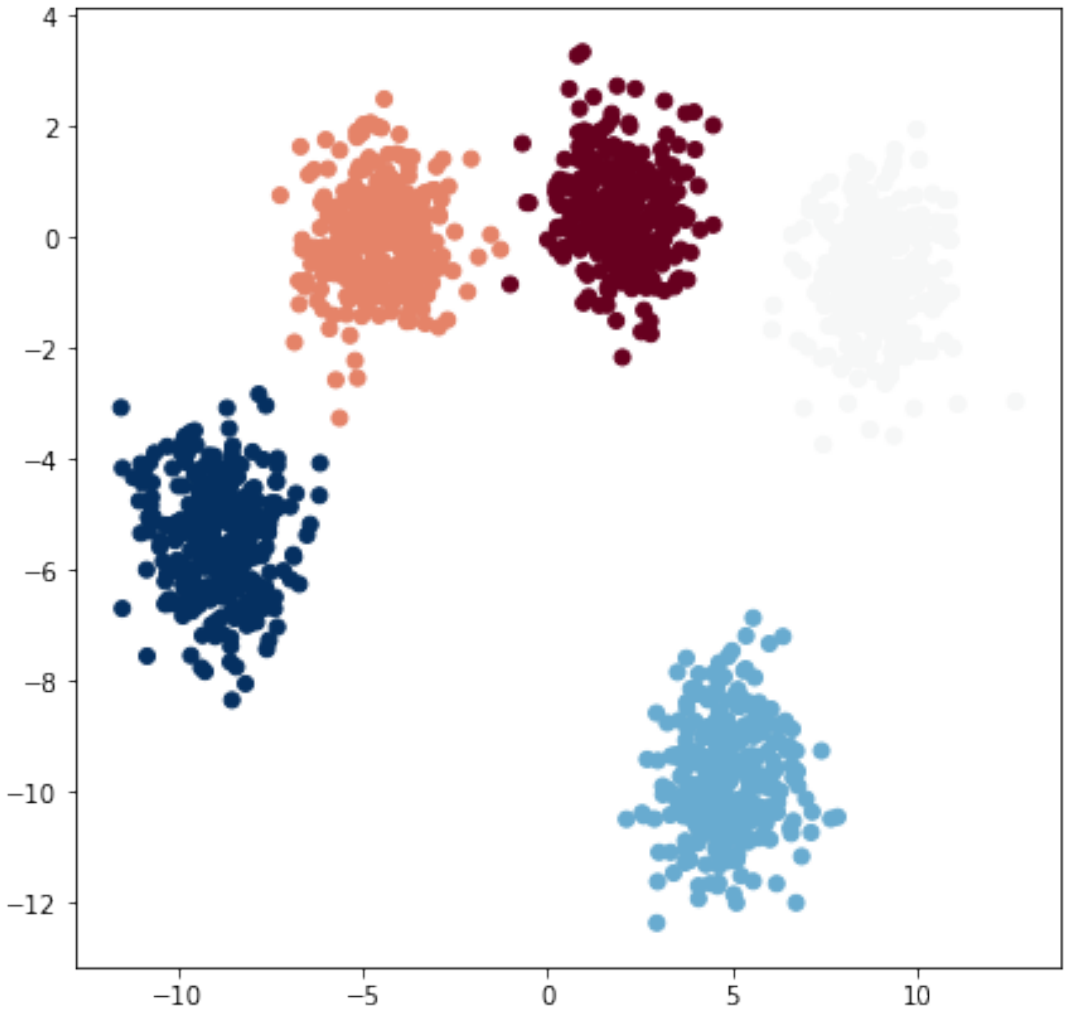
Computer Vision (CSCI 5520G)

Faisal Z. Qureshi

http://vclab.science.ontariotechu.ca

Softmax

Our goal is to extend ideas first explored in logistic regression, which is a binary classifier, to multi-class problems.



Multinomial distribution

Multinomial distribution can be used to model a random variable X that takes values in $\{1, \cdots, k\}$.

$$\Pr(X = i) = \phi_i$$

Since probabilities are non-negative and all sum to 1, $\sum_{i=1}^{k} \phi_i = 1$. Therefore, $\phi_k = 1 - \sum_{i=1}^{k-1} \phi_i$.

Parameters of a multinomial distribution are $\phi_1, \cdots, \phi_{k-1}$.

Example

- Classification in 3 or more classes
- Which of the k diseases does a patient have?
- Does this image contains a horse, a zebra, or a giraffe?
- Modeling a dice throw

Multinomial distribution

Using indicators variables, we can write the the probability of a multinomial random variable X as follows:

$$\Pr(X) = \phi_1^{\mathbb{I}_1(x)} \phi_2^{\mathbb{I}_2(x)} \cdots \phi_k^{\mathbb{I}_k(x)} = \prod_{i=1}^{K} \phi_i^{\mathbb{I}_i(x)},$$

where the indicator variable $\mathbb{I}_i(x)$ is defined as

$$\mathbb{I}_i(x) = \begin{cases} 1 & \text{if } x = i \\ 0 & \text{otherwise} \end{cases}$$

Multinomial distribution

Using indicators variables, we can write the the probability of a multinomial random variable X as follows:

$$\Pr(X) = \phi_1^{\mathbb{I}_1(x)} \phi_2^{\mathbb{I}_2(x)} \cdots \phi_k^{\mathbb{I}_k(x)} = \prod_{i=1}^{K} \phi_i^{\mathbb{I}_i(x)},$$

where the indicator variable $\mathbb{I}_i(x)$ is defined as

$$\mathbb{I}_i(x) = \begin{cases} 1 & \text{if } x = i \\ 0 & \text{otherwise} \end{cases}$$

It follows

$$\phi_k = 1 - \sum_{i=1}^{k-1} \phi_i, \ \ \phi_i \geq 0 \text{ and}$$

$$\mathbb{I}_k(x) = \sum_{i=1}^{k-1} 1 - \mathbb{I}_i(x).$$

Bernoulli distribution (recap)

A Bernoulli random variable X takes values in $\{0, 1\}$

$$\begin{aligned}\Pr(X|\theta) &= \begin{cases}\theta & \text{if } X = 1\\ 1-\theta & \text{otherwise}\end{cases}\\ &= \theta^X(1-\theta)^{1-X}\end{aligned}$$

Multiclass classification

Change of notation: θ_i, where $i \in 1, \cdots, K$ now refers to an $(M+1)$-dimensional vector. Previously θ_i referred to the ith element of the $(M+1)$-dimensional vector θ.

The goal of multiclass classification is to learn $h_\theta(\mathbf{x})$, which can be used to assign a label $y \in \{1, \cdots, K\}$ to the input $\mathbf{x}^{(i)}$, $i \in [1, N]$. Label y takes values in $\{1, \cdots, K\}$, so we can use multinomial distribution to specify its probability distribution.

Under the assumption that data is i.i.d.

$$\Pr(y|\mathbf{X}, \theta) = \prod_{i=1}^{N} \left(\prod_{j=1}^{K} \left(h_{\theta_j}(\mathbf{x}^{(i)}) \right)^{\mathbb{I}_j(y^{(i)})} \right)$$

Multiclass classification

Change of notation: θ_i, where $i \in 1, \cdots, K$ now refers to an $(M+1)$-dimensional vector. Previously θ_i referred to the ith element of the $(M+1)$-dimensional vector θ.

The goal of multiclass classification is to learn $h_\theta(\mathbf{x})$, which can be used to assign a label $y \in \{1, \cdots, K\}$ to the input $\mathbf{x}^{(i)}$, $i \in [1, N]$. Label y takes values in $\{1, \cdots, K\}$, so we can use multinomial distribution to specify its probability distribution.

Under the assumption that data is i.i.d.

$$\Pr(y|\mathbf{X}, \theta) = \prod_{i=1}^{N} \left(\prod_{j=1}^{K} \left(h_{\theta_j}(\mathbf{x}^{(i)}) \right)^{\mathbb{I}_j(y^{(i)})} \right)$$

where

$$\mathbb{I}_c(y^{(i)}) = \begin{cases} 1 & \text{if } y^{(i)} = c \\ 0 & \text{otherwise} \end{cases}$$

Multiclass classification

Change of notation: θ_i, where $i \in 1, \cdots, K$ now refers to an $(M+1)$-dimensional vector. Previously θ_i referred to the ith element of the $(M+1)$-dimensional vector θ.

The goal of multiclass classification is to learn $h_\theta(\mathbf{x})$, which can be used to assign a label $y \in \{1, \cdots, K\}$ to the input $\mathbf{x}^{(i)}$, $i \in [1, N]$. Label y takes values in $\{1, \cdots, K\}$, so we can use multinomial distribution to specify its probability distribution.

Under the assumption that data is i.i.d.

$$\Pr(y|\mathbf{X}, \theta) = \prod_{i=1}^{N} \left(\prod_{j=1}^{K} \left(h_{\theta_j}(\mathbf{x}^{(i)}) \right)^{\mathbb{I}_j(y^{(i)})} \right)$$

where

$$\mathbb{I}_c(y^{(i)}) = \begin{cases} 1 & \text{if } y^{(i)} = c \\ 0 & \text{otherwise} \end{cases}$$

Likelihood for ith example

$$
\begin{aligned}
L(\theta)^{(i)} &= \Pr(y^{(i)}|\mathbf{X}, \theta) \\
&= \prod_{j=1}^{K} \left(h_{\theta_j}(\mathbf{x}^{(i)}) \right)^{\mathbb{I}_j(y^{(i)})}
\end{aligned}
$$

Likelihood for ith example

$$
\begin{aligned}
L(\theta)^{(i)} &= \Pr(y^{(i)}|\mathbf{X}, \theta) \\
&= \prod_{j=1}^{K} \left(h_{\theta_j}(\mathbf{x}^{(i)}) \right)^{\mathbb{I}_j(y^{(i)})}
\end{aligned}
$$

Negative log-likelihood for ith example

$$
l(\theta)^{(i)} = -\sum_{j=1}^{K} \mathbb{I}_j(y^{(i)}) \log h_{\theta_j}(\mathbf{x}^{(i)})
$$

Negative log-likelihood for ith example

Define $\mathbf{y}^{(i)}$, a K-dimensional vector as follows:

$$\mathbf{y}_j^{(i)} = \begin{cases} 1 & \text{if } \mathbb{I}_j(y^{(i)}) \\ 0 & \text{otherwise} \end{cases}$$

Here $i \in [1, N]$ and $j \in [1, K]$. $\mathbf{y}^{(i)}$ is often referred to as one-hot encoded vector.

We can now write negative log-likelihood as follows

$$l(\theta)^{(i)} = -\sum_{j=1}^{K} \mathbf{y}_j^{(i)} \log h_{\theta_j}(\mathbf{x}^{(i)})$$

Negative log-likelihood for multi-class classification

Putting it all together, we can write the likelihood of data for multicass classification as

$$l(\theta) = \sum_{i=1}^{N} \left(- \sum_{j=1}^{K} \mathbf{y}_j^{(i)} \log h_{\theta_j}(\mathbf{x}^{(i)}) \right)$$

In order to estimate the parameters θ (i.e., learn the multi-class classifier), we need to minimize negative log-likelihood.

Negative log-likelihood for multi-class classification

Putting it all together, we can write the likelihood of data for multicass classification as

$$l(\theta) = \sum_{i=1}^{N} \left(\underbrace{-\sum_{j=1}^{K} \mathbf{y}_j^{(i)} \log h_{\theta_j}(\mathbf{x}^{(i)})}_{\text{Cross-entropy for sample } i} \right)$$

In order to estimate the parameters θ (i.e., learn the multi-class classifier), minimize the cross-entropy between the groud truth and predicted distributions.

Softmax function

Softmax function or *normalized exponential function* "squashes" a K-dimensional vector $\mathbf{z}$ of arbitrary real values to a K-dimensional vector $\mathrm{S}(\mathbf{z})$ of real values in the range $[0, 1]$ that add up to 1.

$$\mathrm{S}(\mathbf{z})_i = \frac{e^{z_i}}{\sum_k e^{z_k}}$$

Softmax function is often used to highlight the largest values and suppress values which are significantly below the maximum value.

Softmax function

Code example (from Wikipedia)

```
>>> import math
>>> z = [1.0, 2.0, 3.0, 4.0, 1.0, 2.0, 3.0]
>>> z_exp = [math.exp(i) for i in z]
>>> print([round(i, 2) for i in z_exp])
[2.72, 7.39, 20.09, 54.6, 2.72, 7.39, 20.09]
>>> sum_z_exp = sum(z_exp)
>>> print(round(sum_z_exp, 2))
114.98
>>> softmax = [round(i / sum_z_exp, 3) for i in z_exp]
>>> print(softmax)
[0.024, 0.064, 0.175, 0.475, 0.024, 0.064, 0.175]
```

Derivative of softmax function (Case 1)

$$\frac{\partial}{\partial z_i}\mathrm{S}(\mathbf{z})_i = \frac{\partial}{\partial z_i}\left[\frac{e^{z_i}}{\sum_k e^{z_k}}\right]$$

Derivative of softmax function (Case 1)

$$\begin{aligned}\frac{\partial}{\partial z_i}\mathrm{S}(\mathbf{z})_i &= \frac{\partial}{\partial z_i}\left[\frac{e^{z_i}}{\sum_k e^{z_k}}\right] \\ &= \left(\frac{1}{\sum_k e^{z_k}}\right)\frac{\partial}{\partial z_i}e^{z_i} + e^{z_i}\frac{\partial}{\partial z_i}\left[\frac{1}{\sum_k e^{z_k}}\right]\end{aligned}$$

Derivative of softmax function (Case 1)

$$
\begin{aligned}
\frac{\partial}{\partial z_i}\mathrm{S}(\mathbf{z})_i &= \frac{\partial}{\partial z_i}\left[\frac{e^{z_i}}{\sum_k e^{z_k}}\right] \\
&= \left(\frac{1}{\sum_k e^{z_k}}\right)\frac{\partial}{\partial z_i}e^{z_i} + e^{z_i}\frac{\partial}{\partial z_i}\left[\frac{1}{\sum_k e^{z_k}}\right] \\
&= \frac{e^{z_i}}{\sum_k e^{z_k}} - \frac{e^{z_i}e^{z_i}}{(\sum_k e^{z_k})^2}
\end{aligned}
$$

Derivative of softmax function (Case 1)

$$
\begin{aligned}
\frac{\partial}{\partial z_i}\mathrm{S}(\mathbf{z})_i &= \frac{\partial}{\partial z_i}\left[\frac{e^{z_i}}{\sum_k e^{z_k}}\right] \\
&= \left(\frac{1}{\sum_k e^{z_k}}\right)\frac{\partial}{\partial z_i}e^{z_i} + e^{z_i}\frac{\partial}{\partial z_i}\left[\frac{1}{\sum_k e^{z_k}}\right] \\
&= \frac{e^{z_i}}{\sum_k e^{z_k}} - \frac{e^{z_i}e^{z_i}}{(\sum_k e^{z_k})^2} \\
&= \frac{e^{z_i}(\sum_k e^{z_k}) - e^{z_i}e^{z_i}}{(\sum_k e^{z_k})^2}
\end{aligned}
$$

Derivative of softmax function (Case 1)

$$
\begin{aligned}
\frac{\partial}{\partial z_i}\mathrm{S}(\mathbf{z})_i &= \frac{\partial}{\partial z_i}\left[\frac{e^{z_i}}{\sum_k e^{z_k}}\right] \\
&= \left(\frac{1}{\sum_k e^{z_k}}\right)\frac{\partial}{\partial z_i}e^{z_i} + e^{z_i}\frac{\partial}{\partial z_i}\left[\frac{1}{\sum_k e^{z_k}}\right] \\
&= \frac{e^{z_i}}{\sum_k e^{z_k}} - \frac{e^{z_i}e^{z_i}}{(\sum_k e^{z_k})^2} \\
&= \frac{e^{z_i}(\sum_k e^{z_k}) - e^{z_i}e^{z_i}}{(\sum_k e^{z_k})^2} \\
&= \left(\frac{e^{z_i}}{\sum_k e^{z_k}}\right)\left(\frac{\sum_k e^{z_k} - e^{z_i}}{\sum_k e^{z_k}}\right)
\end{aligned}
$$

Derivative of softmax function (Case 1)

$$
\begin{aligned}
\frac{\partial}{\partial z_i}\mathrm{S}(\mathbf{z})_i &= \frac{\partial}{\partial z_i}\left[\frac{e^{z_i}}{\sum_k e^{z_k}}\right] \\
&= \left(\frac{1}{\sum_k e^{z_k}}\right)\frac{\partial}{\partial z_i}e^{z_i} + e^{z_i}\frac{\partial}{\partial z_i}\left[\frac{1}{\sum_k e^{z_k}}\right] \\
&= \frac{e^{z_i}}{\sum_k e^{z_k}} - \frac{e^{z_i}e^{z_i}}{(\sum_k e^{z_k})^2} \\
&= \frac{e^{z_i}(\sum_k e^{z_k}) - e^{z_i}e^{z_i}}{(\sum_k e^{z_k})^2} \\
&= \left(\frac{e^{z_i}}{\sum_k e^{z_k}}\right)\left(\frac{\sum_k e^{z_k} - e^{z_i}}{\sum_k e^{z_k}}\right) \\
&= \left(\frac{e^{z_i}}{\sum_k e^{z_k}}\right)\left(1 - \frac{e^{z_i}}{\sum_k e^{z_k}}\right)
\end{aligned}
$$

Derivative of softmax function (Case 1)

$$
\begin{aligned}
\frac{\partial}{\partial z_i}\mathrm{S}(\mathbf{z})_i &= \frac{\partial}{\partial z_i}\left[\frac{e^{z_i}}{\sum_k e^{z_k}}\right] \\
&= \left(\frac{1}{\sum_k e^{z_k}}\right)\frac{\partial}{\partial z_i}e^{z_i} + e^{z_i}\frac{\partial}{\partial z_i}\left[\frac{1}{\sum_k e^{z_k}}\right] \\
&= \frac{e^{z_i}}{\sum_k e^{z_k}} - \frac{e^{z_i}e^{z_i}}{(\sum_k e^{z_k})^2} \\
&= \frac{e^{z_i}(\sum_k e^{z_k}) - e^{z_i}e^{z_i}}{(\sum_k e^{z_k})^2} \\
&= \left(\frac{e^{z_i}}{\sum_k e^{z_k}}\right)\left(\frac{\sum_k e^{z_k} - e^{z_i}}{\sum_k e^{z_k}}\right) \\
&= \left(\frac{e^{z_i}}{\sum_k e^{z_k}}\right)\left(1 - \frac{e^{z_i}}{\sum_k e^{z_k}}\right) \\
&= \mathrm{S}(\mathbf{z})_i(1 - \mathrm{S}(\mathbf{z})_i)
\end{aligned}
$$

Derivative of softmax function (Case 2)

$$\frac{\partial}{\partial z_j}\mathrm{S}(\mathbf{z})_i = \frac{\partial}{\partial z_j}\left[\frac{e^{z_i}}{\sum_k e^{z_k}}\right]$$

Derivative of softmax function (Case 2)

$$\begin{aligned}\frac{\partial}{\partial z_j}\mathrm{S}(\mathbf{z})_i &= \frac{\partial}{\partial z_j}\left[\frac{e^{z_i}}{\sum_k e^{z_k}}\right] \\ &= e^{z_i}\frac{\partial}{\partial z_j}\left[\frac{1}{\sum_k e^{z_k}}\right]\end{aligned}$$

Derivative of softmax function (Case 2)

$$
\begin{aligned}
\frac{\partial}{\partial z_j}\mathrm{S}(\mathbf{z})_i &= \frac{\partial}{\partial z_j}\left[\frac{e^{z_i}}{\sum_k e^{z_k}}\right] \\
&= e^{z_i}\frac{\partial}{\partial z_j}\left[\frac{1}{\sum_k e^{z_k}}\right] \\
&= \frac{-e^{z_i}e^{z_j}}{(\sum_k e^{z_k})^2}
\end{aligned}
$$

Derivative of softmax function (Case 2)

$$
\begin{aligned}
\frac{\partial}{\partial z_j}\mathrm{S}(\mathbf{z})_i &= \frac{\partial}{\partial z_j}\left[\frac{e^{z_i}}{\sum_k e^{z_k}}\right] \\
&= e^{z_i}\frac{\partial}{\partial z_j}\left[\frac{1}{\sum_k e^{z_k}}\right] \\
&= \frac{-e^{z_i}e^{z_j}}{(\sum_k e^{z_k})^2} \\
&= -\left(\frac{e^{z_i}}{\sum_k e^{z_k}}\right)\left(\frac{e^{z_j}}{\sum_k e^{z_k}}\right)
\end{aligned}
$$

Derivative of softmax function (Case 2)

$$
\begin{aligned}
\frac{\partial}{\partial z_j}\mathrm{S}(\mathbf{z})_i &= \frac{\partial}{\partial z_j}\left[\frac{e^{z_i}}{\sum_k e^{z_k}}\right] \\
&= e^{z_i}\frac{\partial}{\partial z_j}\left[\frac{1}{\sum_k e^{z_k}}\right] \\
&= \frac{-e^{z_i}e^{z_j}}{(\sum_k e^{z_k})^2} \\
&= -\left(\frac{e^{z_i}}{\sum_k e^{z_k}}\right)\left(\frac{e^{z_j}}{\sum_k e^{z_k}}\right) \\
&= -\left(\frac{e^{z_i}}{\sum_k e^{z_k}}\right)\left(\frac{e^{z_i}}{\sum_k e^{z_k}}\right)
\end{aligned}
$$

Derivative of softmax function (Case 2)

$$
\begin{aligned}
\frac{\partial}{\partial z_j}\mathrm{S}(\mathbf{z})_i &= \frac{\partial}{\partial z_j}\left[\frac{e^{z_i}}{\sum_k e^{z_k}}\right] \\
&= e^{z_i}\frac{\partial}{\partial z_j}\left[\frac{1}{\sum_k e^{z_k}}\right] \\
&= \frac{-e^{z_i}e^{z_j}}{(\sum_k e^{z_k})^2} \\
&= -\left(\frac{e^{z_i}}{\sum_k e^{z_k}}\right)\left(\frac{e^{z_j}}{\sum_k e^{z_k}}\right) \\
&= -\left(\frac{e^{z_i}}{\sum_k e^{z_k}}\right)\left(\frac{e^{z_i}}{\sum_k e^{z_k}}\right) \\
&= -\mathrm{S}(\mathbf{z})_i\mathrm{S}(\mathbf{z})_j
\end{aligned}
$$

Derivative of softmax function

We can use Kronecker's delta function δ_{ij} to represent the derivative of a softmax function in terms of itself as follows

$$\frac{\partial}{\partial z_j}\mathrm{S}(\mathbf{z})_i = \mathrm{S}(\mathbf{z})_i(\delta_{ij} - \mathrm{S}(\mathbf{z})_j)$$

Here

$$\delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

Softmax classifier

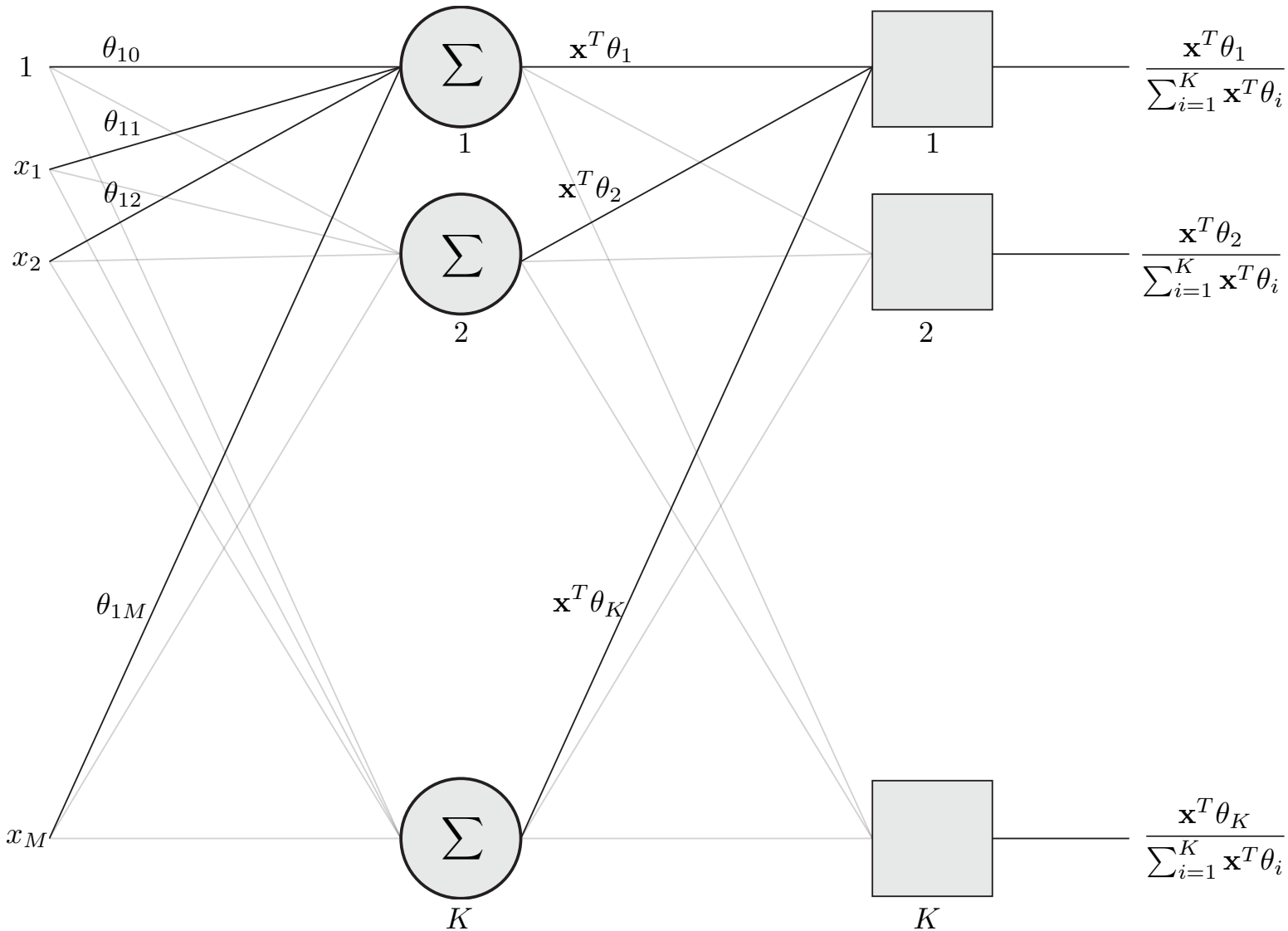
Probability distribution of label y is given by softmax function.

$$h_{\theta_i}(\mathbf{x}) = S(\mathbf{x}^{(i)})_j$$
$$= \frac{e^{\mathbf{x}^T\theta_i}}{\sum_{j=1}^{K} e^{\mathbf{x}^T\theta_j}}$$

Negative log likelihood for softmax classifier

$$l(\theta) = -\sum_{j=1}^{K} \mathbf{y}_j^{(i)} \log \mathrm{S}(\mathbf{x}^{(i)})_j \quad \text{(For } i\text{th example)}$$

Softmax classifier (K-classes)



Softmax classifier derivative

Notation change: drop superscript (i) and let $\mathrm{S}(\mathbf{x}^{(i)})_j = \pi_j$ for simplicity. Recall that $l \in \{1, \cdots, K\}$.

$$\begin{aligned}
\frac{\partial}{\partial \theta_l} l(\theta) &= \frac{\partial}{\partial \theta_l}\left[-\sum_{j=1}^{K} \mathbf{y}_j \log \pi_j\right] = -\sum_{j=1}^{K} \mathbf{y}_j \frac{1}{\pi_j} \frac{\partial}{\partial \theta_l} \pi_j \\
&= -\frac{\mathbf{y}_l \left(\pi_l(1-\pi_l)\mathbf{x}\right)}{\pi_l} - \sum_{j \neq l} \frac{\mathbf{y}_j \left(-\pi_l \pi_j \mathbf{x}\right)}{\pi_j} \\
&= \left(-\mathbf{y}_l + \mathbf{y}_l \pi_l + \sum_{j \neq l} \mathbf{y}_j \pi_l\right) \mathbf{x} \\
&= \left(-\mathbf{y}_l + \pi_l \underbrace{\sum_{j=1}^{K} \mathbf{y}_j}_{=1}\right) \mathbf{x} \\
&= \left(-\mathbf{y}_l + \pi_l\right) \mathbf{x}
\end{aligned}$$

Softmax classifier gradient descent

Notation: k here refers to the iteration number for gradient descent. η is the learning rate. $l \in \{1, \cdots, K\}$, where K is the number of classes or distinct values labels can take.

Stochatic gradient descent

$$\begin{aligned}\theta_l^{(k+1)} &= \theta_l^{(k)} - \eta \nabla_\theta l(\theta) \\ &= \theta_l^{(k)} + \eta \left(\frac{e^{\mathbf{x}^T \theta_l}}{\sum_{j=1}^{K} e^{\mathbf{x}^T \theta_j}} - \mathbf{y}_l \right) \mathbf{x}\end{aligned}$$

Cross-entropy

How do we compare the output of a softmax $\hat{\mathbf{y}}^{(i)} \in \mathbb{R}^K$ for the current example $\mathbf{x}^{(i)}$ with the *true* class label of the current example?

1. Use one-hot encoding $\mathbf{y}^{(i)}$ for ground truth labels. If the true class label of $\mathbf{x}^{(i)}$ is j then the one hot encoding contains all 0 except at index location j where it stores a 1.

2. Compare $\hat{\mathbf{y}}^{(i)}$ and $\mathbf{y}^{(i)}$ using cross-entropy:

$$\text{Cross-entropy}(\mathbf{y}^{(i)}, \hat{\mathbf{y}}^{(i)}) = -\sum_{j=1}^{K} \underbrace{\mathbf{y}_j^{(i)}}_{\text{true}} \log \underbrace{\hat{\mathbf{y}}_j^{(i)}}_{\text{predicted}}$$

Cross-entropy

How do we compare the output of a softmax $\hat{\mathbf{y}}^{(i)} \in \mathbb{R}^K$ for the current example $\mathbf{x}^{(i)}$ with the *true* class label of the current example?

1. Use one-hot encoding $\mathbf{y}^{(i)}$ for ground truth labels. If the true class label of $\mathbf{x}^{(i)}$ is j then the one hot encoding contains all 0 except at index location j where it stores a 1.

2. Compare $\hat{\mathbf{y}}^{(i)}$ and $\mathbf{y}^{(i)}$ using cross-entropy:

$$\text{Cross-entropy}(\mathbf{y}^{(i)}, \hat{\mathbf{y}}^{(i)}) = -\sum_{j=1}^{K} \underbrace{\mathbf{y}_j^{(i)}}_{\text{true}} \log \underbrace{\hat{\mathbf{y}}_j^{(i)}}_{\text{predicted}}$$

Summary

- Multiclass classification
- Multinomial distribution
- Softmax classifier

Copyright and License