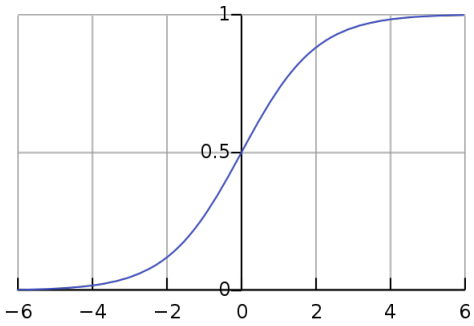
1 Sigmoid Function and Logistic Regression

In lecture, we discussed sigmoid functions - a class of functions characterized by an "S" shaped curve. This "S" shape is especially useful in machine learning, where we are constantly finding gradients and relying on our graphs being differentiable! Often times, as discussed in class, our choice of the sigmoid function is the logistic function:

$$g(z) = \frac{1}{1 + e^{-z}} \tag{1}$$



where $z = mx + b$

(a) Let's do a quick concept check. In what situation would we use logistic regression instead of linear regression?

(b) We see that $g(z)$ falls strictly between $(0, 1)$. Given what we have learned in class and discussed so far, what probability distribution does this graph represent?

(c) Now, let's consider a $\mathbb{R}^3$ space. For weight vector $\boldsymbol{\theta} = \begin{bmatrix} 1 \\ 4 \\ 3 \end{bmatrix}$, define (i) some $\boldsymbol{x}$ such that $\boldsymbol{\theta}^T \boldsymbol{x} > 0$. What is the resulting $g(z)$? Now, (ii) some $\boldsymbol{x}$ such that $\boldsymbol{\theta}^T \boldsymbol{x} = 0$. What is the resulting $g(z)$? Explain the overall relationship between $g(z)$ and $\boldsymbol{\theta}^T \boldsymbol{x}$.

2 Multinomial Logistic Regression: Multi-class Classification

So far, we have seen how to use logistic regression to model a binary variable, such as pass/fail, healthy/sick, etc. Now, what if we have k classes? How can we learn such a classifier?

In a K-class classification setting, we have training set $\mathcal{D} = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}) \mid i = 1, \ldots, n\}$ where $\mathbf{x}^{(i)} \in \mathbb{R}^M$ is a feature vector and $\mathbf{y}^{(i)} \in \{1, 2, ..., K\}$.

2.1 One-versus-All

Using a *one-vs-all* classifier (with logistic regression) to predict the label for a new data includes the following two steps:

Note: this method can be used with any binary classifier (including binary logistic regression, binary SVM classifier, etc).

2.2 Generalization of Logistic Regression: Softmax Regression

The Softmax Function

For a vector $\mathbf{z} = [z_1, z_2, ..., z_K]^T \in R^K$, the *softmax* function outputs a vector of the same dimension, $softmax(\mathbf{z}) \in R^K$, where each of its entries is defined as:

$$softmax(\mathbf{z})_k = \frac{e^{z_k}}{\sum_{c=1}^{K} e^{z_c}}, \text{ for all } k = 1, 2, ..., K$$

which guarantees two things:

- Each entry of the resulting vector $softmax(\mathbf{z})$ is a value in the range $(0, 1)$
- $\sum_{k=1}^{K} softmax(\mathbf{z})_k = 1$

Therefore, the *softmax* function is useful for converting a vector of arbitrary real numbers into a discrete probability density consisting of K probabilities proportional to the exponentials of the input vector components. Note that, for example, the larger input components correspond to larger probabilities.

Softmax is often used as the last layer of a neural networks, to map the non-normalized output of a network to a probability distribution over predicted output classes.

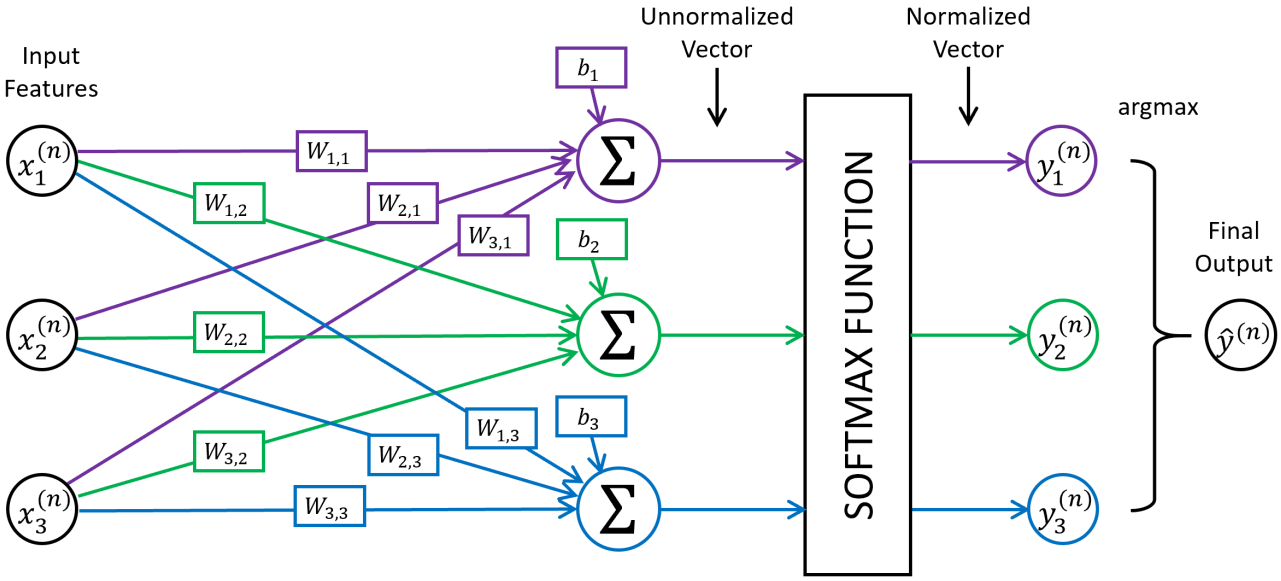


Figure 1: Procedure of Softmax Regression with 3-dimensional Features

Softmax Regression

For K-class classification, Softmax Regression has a parametric model of the form:

$$p(y^{(i)} = k \mid \mathbf{x}^{(i)}; \mathbf{W}, \mathbf{b}) = \frac{\exp(\mathbf{w}_k^T \mathbf{x}^{(i)} + b_k)}{\sum_{c=1}^{K} \exp(\mathbf{w}_c^T \mathbf{x}^{(i)} + b_c)}. \tag{2}$$

Therefore, the output of the softmax model looks like: $\hat{y} = \text{argmax}_k \; p(y^{(i)} = k \mid \mathbf{x}^{(i)}; \mathbf{W}, \mathbf{b})$
The intermediate result (a vector) outputted by the softmax function is:

$$\begin{bmatrix} p(y^{(i)} = 1 \mid \mathbf{x}^{(i)}; \mathbf{W}) \\ p(y^{(i)} = 2 \mid \mathbf{x}^{(i)}; \mathbf{W}) \\ \vdots \\ p(y^{(i)} = K \mid \mathbf{x}^{(i)}; \mathbf{W}) \end{bmatrix} = \frac{1}{\sum_{k=1}^{K} \exp(\mathbf{w}_k^T \mathbf{x}^{(i)} + b_k)} \begin{bmatrix} \exp(\mathbf{w}_1^T \mathbf{x}^{(i)} + b_1) \\ \exp(\mathbf{w}_2^T \mathbf{x}^{(i)} + b_2) \\ \vdots \\ \exp(\mathbf{w}_K^T \mathbf{x}^{(i)} + b_k) \end{bmatrix}$$

Note: now $\mathbf{W}$ is a matrix! Let $\mathbf{W}$ be the $M \times K$ matrix obtained by concatenating $\mathbf{w}_1$, $\mathbf{w}_2$,...,$\mathbf{w}_K$, where each $\mathbf{w}_i \in \mathbb{R}^M$.

1. **Exercise: Relationship to Logistic Regression**
   In the special case where $K = 2$, one can show that softmax regression reduces to logistic regression. This shows that softmax regression is a generalization of logistic regression.
   Specifically, show the equivalence between the two equations (2) and (3).

$$p(y^{(i)} = k \mid \mathbf{x}^{(i)}; \mathbf{W}, \mathbf{b}) = \frac{\exp(\mathbf{w}_k^T \mathbf{x}^{(i)} + b_k)}{\sum_{c=1}^{2} \exp(\mathbf{w}_c^T \mathbf{x}^{(i)} + b_k)}. \tag{3}$$

$$= \begin{cases} \frac{1}{1+\exp(-(\mathbf{w}_1^T \mathbf{x}^{(i)} + b_1))}, & \text{if } k = 1 \\ 1 - \frac{1}{1+\exp(-(\mathbf{w}_1^T \mathbf{x}^{(i)} + b_1))} = \frac{\exp(-(\mathbf{w}_1^T \mathbf{x}^{(i)} + b_1))}{1+\exp(-(\mathbf{w}_1^T \mathbf{x}^{(i)} + b_1))}, & \text{if } k = 2 \end{cases} \tag{4}$$

   Note that the softmax model contains two "sets" of weights, whereas the logistic regression output only contains one "set" of weight. Therefore, this simple example with $K = 2$ not only shows that softmax regression is a generalization of logistic regression, but also shows that softmax regression has a "redundant" set of parameters.