Logistic Regression

Group 1
Hoàng Long - 21520058
Lâm Thanh - 21520055
Bảo Trung - 21521598
Huỳnh Dương - 20520461
Phạm Kiên - 20521490

Bảng phân công công việc

MSSV	Công việc	Đánh giá
21520058	Thiết kế bài thuyết trình, chuẩn bị nội dung, thuyết trình	22%
21520055	Demo, thuyết trình	22%
21521598	Thiết kế bài thuyết trình, chuẩn bị nội dung, trả lời câu hỏi	20%
20520461	Thiết kế bài thuyết trình, chuẩn bị nội dung,	18%
20521490	Thiết kế bài thuyết trình, chuẩn bị nội dung,	18%

Outline

$$\begin{tikzcd}
\text{What is Logistic Regression?} \arrow[r] & \text{Types of Logistic Regression} \arrow[r] & \text{Sigmoid Function} \arrow[d] \\
\text{Regularization} & \text{Optimizing the Loss Function} \arrow[l] & \text{Loss Function - Cross-Entropy} \arrow[l]
\end{tikzcd}$$

What is Logistic Regression?

- Logistic Regression is a statistical model used for binary classification tasks, predicting the probability that a given input point belongs to a specific class.

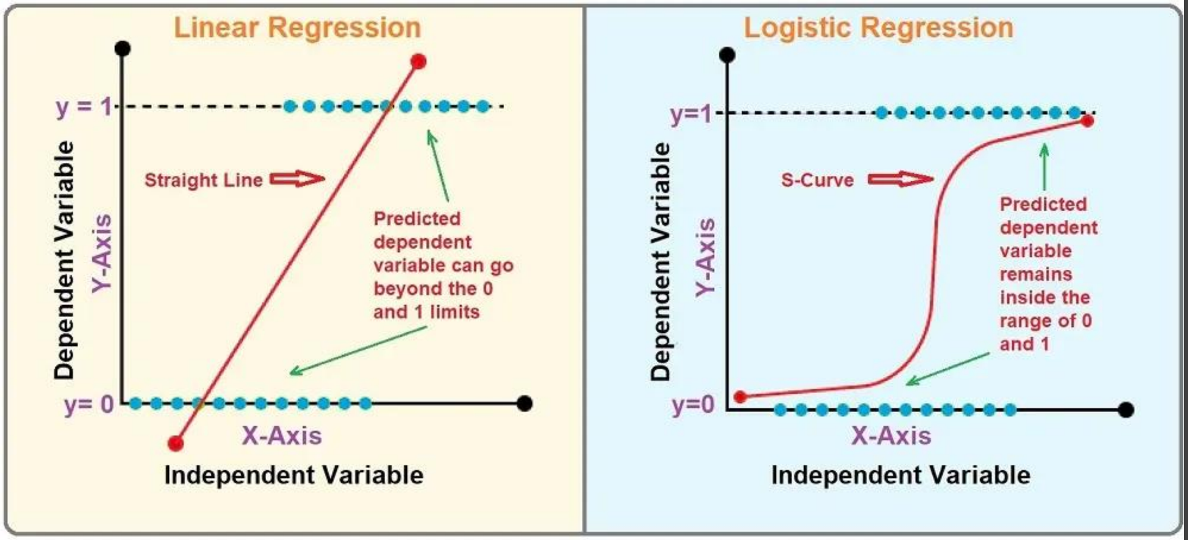
What is Logistic Regression?

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

in there:

$z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n$

$\sigma(z) is\ the\ output\ probability\ of\ a\ class$



What is Logistic Regression?

Types of Logistic Regression

- 1. Binary Logistic Regression: For two classes.
- 2. Multinomial Logistic Regression: For multiple classes.
- 3. Ordinal Logistic Regression: For ordered classes.

- Applications include spam detection, disease diagnosis, and more.

Sigmoid Function

- The Sigmoid function is defined as:
- $f(s) = \frac{1}{1 + e^{-s}}$
- It maps any real-valued number to a value between 0 and 1, making it ideal for binary classification.

Binary logistic regression aims to train a classifier that makes a binary decision for an observation, $y \in \{0,1\}$. The classifier uses a set of feature weights w_i and a bias term b to assess how likely x belongs to the positive class $P(y = 1|x)$.

For a given observation $x = [x_1, x_2, \ldots, x_n]$,the classifier's weighted sum of features plus the bias is represented by:

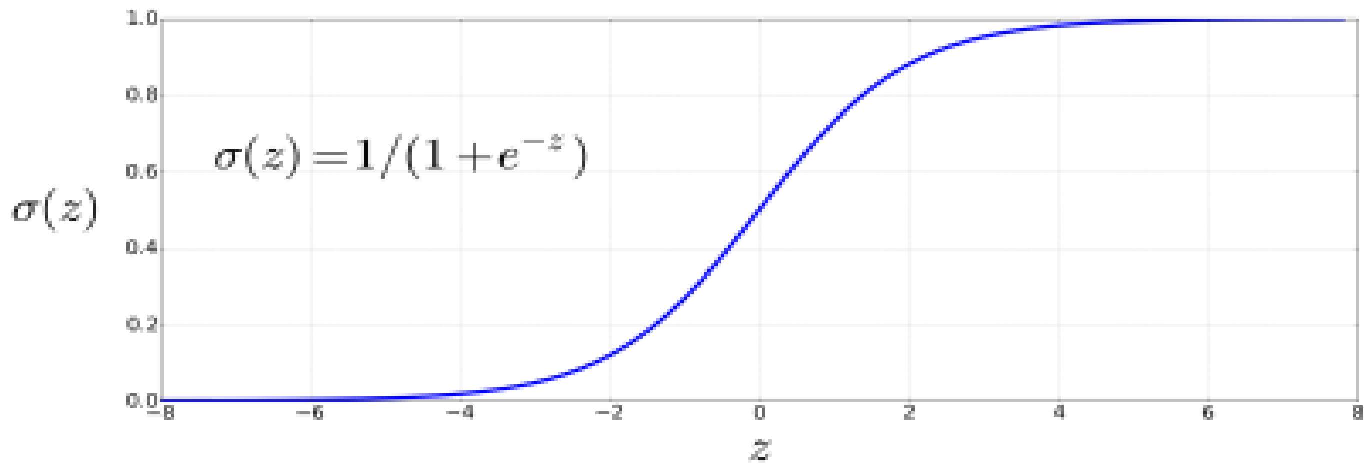
$$z = \left(\sum_{i=1}^{n} w_i x_i\right) + b \qquad (5.2)$$

Dot product notation from linear algebra. The dot product of two vectors a and b, written as a b, is the sum of the products of the corresponding elements of each vector. (Notice that we represent vectors using the boldface notation b).

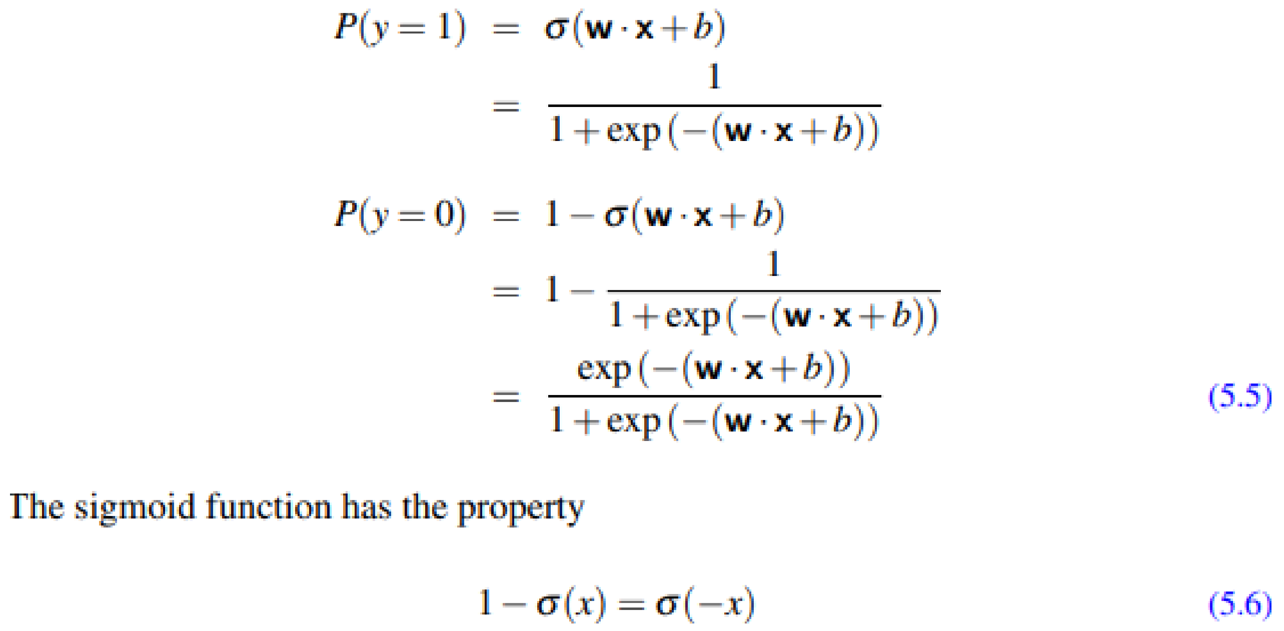
$$z = w.x + \mathrm{b}$$

However, z can range from $-\infty$ *to* ∞, so it's not a probability. To map z to $(0,1)$, we apply the sigmoid function:

$$\sigma(z) = \frac{1}{1+e^{-z}} = \frac{1}{1+\exp(-z)} \tag{5.4}$$



This squashes z to fall between 0 and 1, useful for interpreting z as a probability:



$$\begin{aligned} P(y=1) &= \sigma(\mathbf{w}\cdot\mathbf{x}+b) \\ &= \frac{1}{1+\exp(-(\mathbf{w}\cdot\mathbf{x}+b))} \\ P(y=0) &= 1-\sigma(\mathbf{w}\cdot\mathbf{x}+b) \\ &= 1-\frac{1}{1+\exp(-(\mathbf{w}\cdot\mathbf{x}+b))} \\ &= \frac{\exp(-(\mathbf{w}\cdot\mathbf{x}+b))}{1+\exp(-(\mathbf{w}\cdot\mathbf{x}+b))} \end{aligned} \tag{5.5}$$

The sigmoid function has the property

$$1-\sigma(x)=\sigma(-x) \tag{5.6}$$

Finally, one terminological point. The input to the sigmoid function, the score $z = wx + b$ from, is often called the logit. This is because the logit function is the inverse of the sigmoid. The logit function is the log of the odds ratio $\frac{\mathrm{p}}{1-\mathrm{p}}$:

$$\operatorname{logit}(p) = \sigma^{-1}(p) = \ln \frac{p}{1-p}$$

Loss Function - Cross-Entropy

- The cross-entropy loss measures the performance of a classification model whose output is a probability value between 0 and 1. It is defined as:
- $L(y, \hat{y}) = -ylog(\hat{y}) + (1 - \mathrm{y})\log(1 - \hat{y})$

Conditional Probability: In a logistic model, the likelihood of the label y given input x is defined as:

$$p(y|x) = \hat{y}^y (1-\hat{y})^{1-y} \tag{5.21}$$

Taking the Log of the Likelihood: To simplify calculations, we take the log of the above likelihood function, leading to:

$$\begin{aligned} \log p(y|x) &= \log \left[\hat{y}^y (1-\hat{y})^{1-y}\right] \\ &= y \log \hat{y} + (1-y) \log(1-\hat{y}) \end{aligned} \tag{5.22}$$

Defining the Loss Function (Cross-Entropy Loss): To convert the log-likelihood function into a loss function for minimization, we flip the sign of Equation 5.22, resulting in the cross-entropy loss L_{CE}:

$$L_{\mathrm{CE}}(\hat{y}, y) = -\log p(y|x) = -[y \log \hat{y} + (1-y) \log(1-\hat{y})] \tag{5.23}$$

Substituting with Sigmoid: When substituting the predicted value $\hat{y} = \sigma(w.x + b)$into the loss function:

$$L_{\mathrm{CE}}(\hat{y}, y) = -[y \log \sigma(\mathbf{w} \cdot \mathbf{x} + b) + (1-y) \log(1 - \sigma(\mathbf{w} \cdot \mathbf{x} + b))] \qquad (5.24)$$

Optimizing the Loss Function

- To find the optimal weights that minimize the loss, gradient descent is applied. The update rule for weights is:

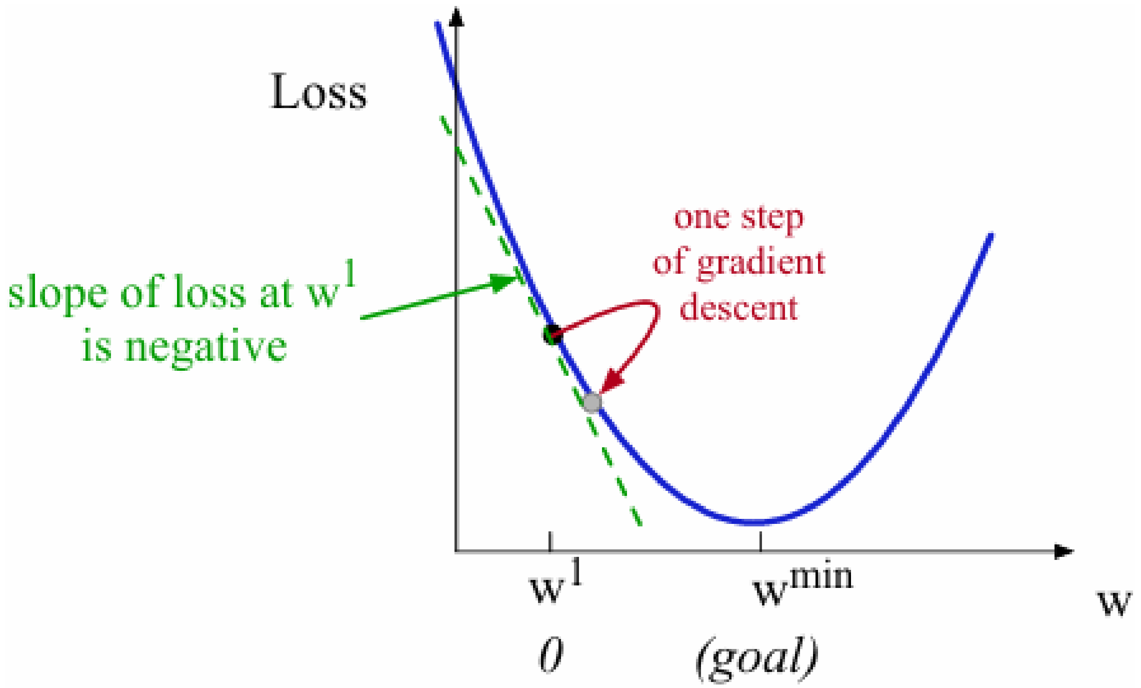
Gradient Descent

- Our goal with gradient descent is to find the optimal weights: minimize the loss function we've defined for the model.

In Eq. below, we'll explicitly represent the fact that the cross-entropy loss function LCE is parameterized by the weights. In machine learning in general we refer to the parameters being learned as ; in the case of logistic regression $\theta = \{w, b\}$. So the goal is to find the set of weights which minimizes the loss function, averaged over all examples

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \frac{1}{m} \sum_{i=1}^{m} L_{\mathrm{CE}}(f(x^{(i)}; \theta), y^{(i)}) \tag{5.25}$$

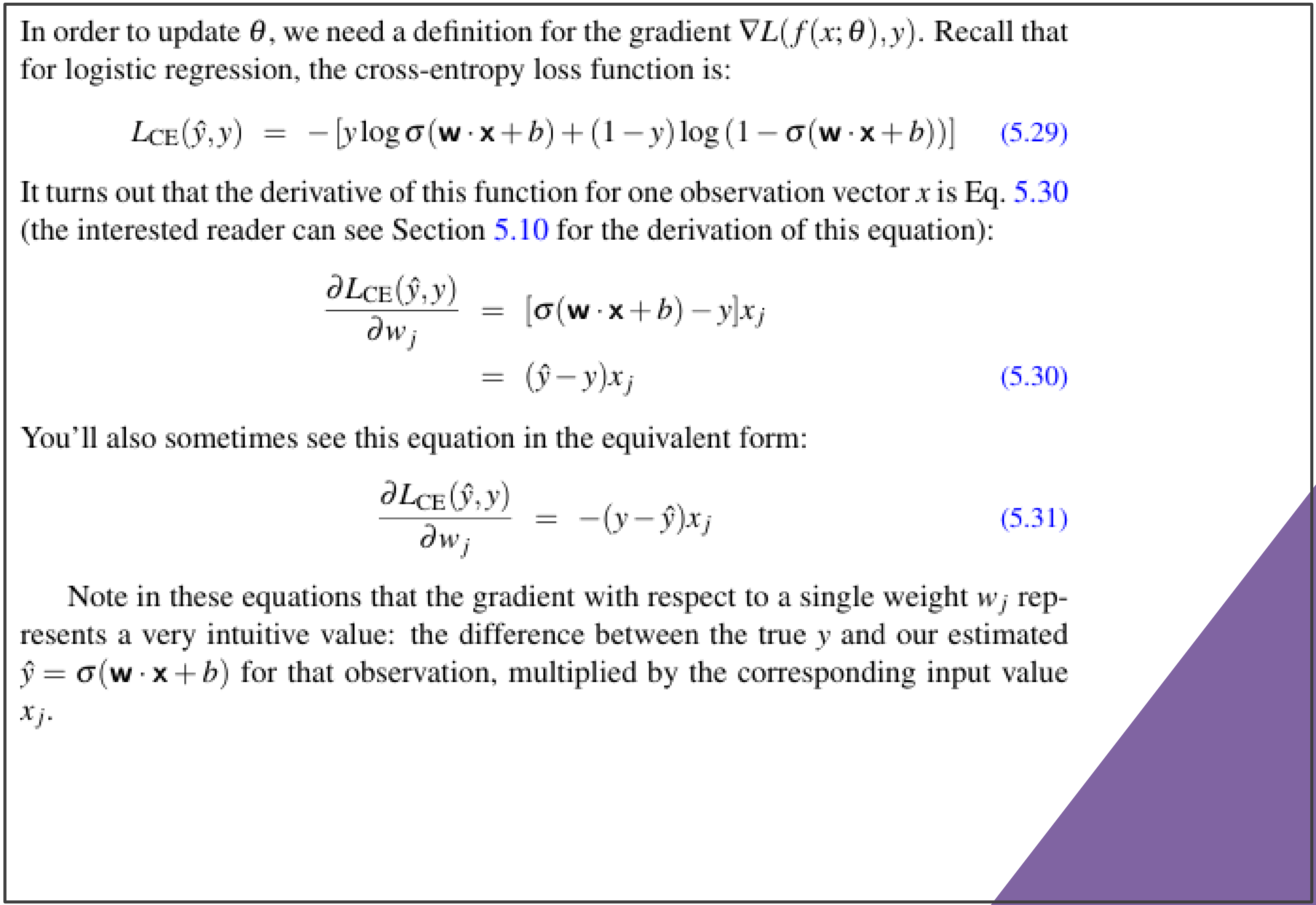
Gradient descent finds the minimum of a loss function by moving in the direction opposite to the steepest slope. In logistic regression, the convex loss function ensures a single minimum, so gradient descent reliably reaches it. For a single parameter w^2, the algorithm simply adjusts w^1 left or right to minimize the loss.


Loss
slope of loss at w^1 is negative
one step of gradient descent
w^1
w^{min}
w
0
(goal)

The gradient descent algorithm answers this question by finding the gradient of the loss function at the current point and moving in the opposite direction. The gradient of a function of many variables is a vector pointing in the direction of the greatest increase in a function. The gradient is a multi-variable generalization of the slope, so for a function of one variable like the one in Fig. 5.4, we can informally think of the gradient as the slope. The dotted line in Fig. 5.4 shows the slope of this hypothetical loss function at point w = w1. You can see that the slope of this dotted line is negative. Thus to find the minimum, gradient descent tells us to go in the opposite direction: moving w in a positive direction.

$$w^{t+1} = w^t - \eta \frac{d}{dw} L(f(x;w), y) \tag{5.26}$$

The Gradient for Logistic Regression



In order to update θ, we need a definition for the gradient $\nabla L(f(x;\theta),y)$. Recall that for logistic regression, the cross-entropy loss function is:

$$L_{\text{CE}}(\hat{y},y) = -[y\log\sigma(\mathbf{w}\cdot\mathbf{x}+b)+(1-y)\log(1-\sigma(\mathbf{w}\cdot\mathbf{x}+b))] \tag{5.29}$$

It turns out that the derivative of this function for one observation vector x is Eq. 5.30 (the interested reader can see Section 5.10 for the derivation of this equation):

$$\begin{aligned}\frac{\partial L_{\text{CE}}(\hat{y},y)}{\partial w_j} &= [\sigma(\mathbf{w}\cdot\mathbf{x}+b)-y]x_j \\ &= (\hat{y}-y)x_j \end{aligned} \tag{5.30}$$

You'll also sometimes see this equation in the equivalent form:

$$\frac{\partial L_{\text{CE}}(\hat{y},y)}{\partial w_j} = -(y-\hat{y})x_j \tag{5.31}$$

Note in these equations that the gradient with respect to a single weight w_j represents a very intuitive value: the difference between the true y and our estimated $\hat{y}=\sigma(\mathbf{w}\cdot\mathbf{x}+b)$ for that observation, multiplied by the corresponding input value x_j.

Regularization

- Regularization techniques like L1 and L2 prevent overfitting by adding a penalty to the loss function:

Overfitting happens when a model is too closely fitted to small training data, leading to poor test performance. Regularization techniques are used to adjust the model slightly to prevent overfitting while retaining its key properties.

The way to do this is to add a penalty component R(θ) to the loss function:

$$\hat{\theta} = \arg\max_{\theta} \sum_{i=1}^{m} \log P(y^{(i)} | x^{(i)}) - \alpha R(\theta)$$

Idea: choose a number R(θ) to penalize large weights, this number is to evaluate the complexity of the model. Theretwo common ways to calculate this regularization component R(θ):

L1 Regularization (Lasso regression)

Named after the L1 norm ||W||1, = sum of absolute values of weights = Manhattan distance.

$$R(\theta) \;=\; ||\theta||_1 = \sum_{i=1}^{n} |\theta_i|$$

The L1 regularized loss function becomes:

$$\hat{\theta} \;=\; \underset{\theta}{\operatorname{argmax}} \left[\sum_{i=1}^{m} \log P(y^{(i)}|x^{(i)}) \right] - \alpha \sum_{j=1}^{n} |\theta_j|$$

L2 Regularization (Ridge Regularization)

The name comes from the fact that this is the (square of) the L2 norm ||θ||2, which is equivalent to the Euclidean distance of θ to the origin.

$$R(\theta) = ||\theta||_2^2 = \sum_{j=1}^{n} \theta_j^2$$

The L2 regularized loss function becomes:

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}} \left[\sum_{i=1}^{m} \log P(y^{(i)}|x^{(i)}) \right] - \alpha \sum_{j=1}^{n} \theta_j^2$$

Multinomial Logistic Regression

- For multiple classes, logistic regression uses the softmax function:
- $softmax(z_i) = \frac{\exp(z_i)}{\sum_{j=1}^{k} \exp(z_j)}\ 1 \le i \le k$
- This allows for classification across more than two classes.

Often we have more than 2 classes:

- Positive/Negative/Neutral
- Parts of speech (nouns, verbs, adjectives, adverbs, prepositions, etc.)

If we have more than 2 classes, we use Multinomial Logistic Regression. The probabilities of all of them must sum to 1.We need a general sigmoid function, called a softmax function:

- Take a vector $z = [z_1, z_2, \dots, z_k]\ of\ k$ arbitrary values.
- Output a probability distribution.
- Each value lies in the interval $[0, 1]$.
- All values sum to 1.

$$\text{softmax}(z_i) = \frac{\exp(z_i)}{\sum_{j=1}^{k} \exp(z_j)} \quad 1 \le i \le k$$

$$\text{softmax}(z) = \left[\frac{\exp(z_1)}{\sum_{i=1}^{k} \exp(z_i)}, \frac{\exp(z_2)}{\sum_{i=1}^{k} \exp(z_i)}, \dots, \frac{\exp(z_k)}{\sum_{i=1}^{k} \exp(z_i)} \right]$$

Example: Convert a vector $z = [z_1, z_2, \ldots, z_k]\ of\ k$ arbitrary values into probabilities.

z = [0.6, 1.1, -1.5, 1.7, 2.5, -0.8]
Kq:[0.079, 0.13, 0.0096, 0.236, 0.526, 0.019]

$Softmax$ in multivariate logistic regression:

$$p(y=c|x) = \frac{\exp(w_c \cdot x + b_c)}{\sum_{j=1}^{k} \exp(w_j \cdot x + b_j)}$$

The input is still the dot product between the weight vector w and the input vector x. But now we will need separate weight vectors for each of the K layers.

Demo

Summary

- Logistic Regression is a powerful model for classification tasks, with applications in binary and multiclass scenarios. Key aspects include the sigmoid function, cross-entropy loss, and regularization to improve generalization.