Lecture 19

Logistic Regression

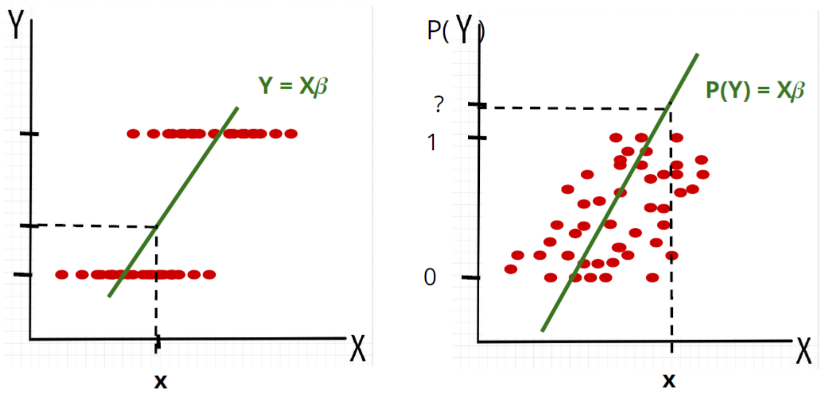
Logistic Regression is a supervised classification algorithm used to predict the probability that a given input belongs to a certain class. Despite its name, it is primarily used for binary classification (two classes), though extensions exist for multiclass problems.

- *Input:* Feature vector $x \in \mathbb{R}^n$
- *Output:* Probability of a class label y being 1 (or true)

Unlike linear regression which predicts continuous values, logistic regression outputs probabilities that lie between 0 and 1.

Why not use Linear Regression

Linear regression predicts outputs on the entire real line $(-\infty, +\infty)$, which cannot properly represent probabilities. It could give values less than 0 or greater than 1, which makes no sense for probabilities. Output of Linear regression on such data below:



Logistic regression instead models the log-odds of the probability through a logistic function which outputs values bounded between 0 and 1.

The Logistic Function (Sigmoid Function)

The core of logistic regression is the logistic (sigmoid) function, defined as:

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

This function maps real value $z \in (-\infty, +\infty)$ into range $(0,1)$ which can be interpreted as a probability.

Logistic regression models the probability that the dependent variable $y = 1$ given the input features X:

$$p(y = 1|X) = \sigma(W^T X + b) = \frac{1}{1 + e^{-(W\^T X+b)}}$$

Where:

- $W \in \mathbb{R}^n$ are model weights
- b is bias term intercept
- $W^T X + b$ is linear combination of inputs

We can interpret $W^T X + b$ as the log – odds:

$$\log\left(\frac{p(y = 1|x)}{1 - p(y = 1|x)}\right) = W^T X + b$$

Decision Rule:
to classify a new input:

$$\hat{y} = \begin{cases} 1, & if\ p(y = 1|x) \geq 0.5 \\ 0, & otherwiswe \end{cases}$$

The threshold 0.5 can be adjusted depending on the problem's needs.

Loss Function and Parameter Estimation

We estimate W and b by maximizing the likelihood of the observed data.

Given the training set $\{(x_i, y_i)\}_{i=1}^{m}$ where $y_i \in \{0,1\}$, the likelihood is:

$$L(W, b) = \prod_{i=1}^{m} p(y_i|X_i; W, b) = \prod_{i=1}^{m} [\sigma(W^T X_i + b)]^{y_i} [1 - \sigma(W^T X_i + b)]^{1-y_i}$$

Log – Likelihood

Taking log:

$$l(W, b) = \sum_{i=1}^{m} [y_i \log \sigma(W^T X_i + b) + (1 - y_i) \log(1 - \sigma(W^T X_i + b))]$$

The optimization problem becomes minimizing the negative log-likelihood:

$$J(W, b) = -l(W, b) = -\sum_{i=1}^{m} [y_i \log \sigma(W^T X_i + b) + (1 - y_i) \log(1 - \sigma(W^T X_i + b))]$$

Gradient Descent

Minimizing $J(W, b)$ numerically using gradient – based methods, such as Gradient Descent or variants like Stochastic Gradient Descent.

Gradients:

$$\frac{\partial J}{\partial w_j} = \sum_{i=1}^{m} (\hat{y}_i - y_i) x_{ij}$$

$$\frac{\partial J}{\partial b} = \sum_{i=1}^{m} (\hat{y}_i - y_i)$$

Update rules:

$$w_j \leftarrow w_j - \alpha \frac{\partial J}{\partial w_j}$$

$$b \leftarrow b - \alpha \frac{\partial J}{\partial b}$$

- *Multinomial Logistic Regression*: For multiclass classification (more than two classes).
- *Regularized Logistic Regression*: Adds penalty terms (L1 or L2) to the loss function to prevent overfitting.
- *Weighted Logistic Regression*: Incorporates weights for imbalanced classes.

Notes about $\boldsymbol{\alpha}$:

- If $\boldsymbol{\alpha}$ is too large, GD may overshoot the maximum, take a long time to or never be able to converge
- If $\boldsymbol{\alpha}$ is too small, GD may take too long to converge

What if the data is not linearly separable

Logistic Regression is a linear classifier, meaning it tries to find a linear decision boundary (a hyperplane) that separates the classes by modeling the log-odds as a linear function of the inputs.

If the classes are not linearly separable, that is, no straight line (or hyperplane) can perfectly separate the classes, then:

- The model will not be able to perfectly classify the data.
- Logistic regression will try to find the best linear boundary that minimizes the classification error (or cross-entropy loss), but there will inevitably be some misclassifications.
- The predicted probabilities will reflect uncertainty near areas where classes overlap.

Ways to handle non-linearly separable data:

1. *Feature Engineering / Transformation*
    a. Transform the original features x into a higher-dimensional space where the classes become (approximately) linearly separable.
    b. Add polynomial features: squares, cross-products, etc.
    c. Use basis functions like radial basis functions, splines, or other nonlinear transformations.
2. *Use Kernel Methods (Kernel Logistic Regression)*
    a. Like Support Vector Machines (SVMs), you can apply the kernel trick to implicitly map data into a higher dimensional space.
    b. Kernels allow non-linear decision boundaries without explicitly computing those transformations.
    c. Kernel logistic regression models the probability with kernels, but it is computationally more expensive.

Multiclass Logistic Regression

When you have more than two classes, logistic regression can be extended from binary classification to multiclass classification. The two main approaches are:

1. *One-vs-Rest (OvR) / One-vs-All (OvA)*
    a. You train one binary logistic regression model per class.
    b. For each model, treat the current class as positive (label 1) and all other classes combined as negative (label 0).
    c. At prediction time, run all models on the input and pick the class with the highest predicted probability.
    d. Pros:
        i. Simple to implement.
        ii. Uses binary logistic regression as is.
    e. Cons:
        i. Independent models may have inconsistent probability estimates.
        ii. Can be suboptimal if classes overlap.
2. Softmax Regression
    a. This is a direct generalization of logistic regression for multiple classes.
    b. Instead of modeling probability with a sigmoid for one class, model the probability distribution over all classes using the softmax function.
    c. If there are K classes ($y \in \{1,2,\ldots,K\}$), the model assigns:
    $$p(y = k|X) = \frac{\exp(W_k^T X + b_k)}{\sum_{j=1}^{K} \exp(W_j^T X + b_j)}$$
    where each class k has its own parameter vector W_k and bias b_k.
    d. The denominator ensures the probabilities sum to 1 across all K classes.

e. The model outputs a probability distribution over the classes.

f. The predicted class is:

$$\hat{y} = \arg\max_k p(y = k|x)$$

g. The loss to minimize is the multiclass cross-entropy loss (extension of binary cross-entropy):

$$J = -\sum_{i=1}^{m}\sum_{k=1}^{K} y_{i,k} \log p(y = k|x_i)$$

where $y_{i,k}$ if example i is of class k, else 0.

Some Plots