Probabilistic Interpretation of Linear Models

In Machine Learning, we often accept loss functions (like "Least Squares" or "Log Loss") as given rules. The **Probabilistic Interpretation** explains *why* these rules exist. It proves that these loss functions are not arbitrary; they are mathematically derived from statistical assumptions about the data.

1. Linear Regression: The Gaussian Assumption

The Core Question

Why do we minimize the **Sum of Squared Errors (Least Squares)** when training a linear regression model?

The Assumption

We assume that the relationship between the input (x) and the output (y) is linear, but reality is imperfect. Therefore, we assume the target value is the model's prediction plus some random "noise" or error (ϵ).

$$y^{(i)} = \theta^T x^{(i)} + \epsilon^{(i)}$$

We assume this error term $\epsilon^{(i)}$ is distributed according to a **Gaussian (Normal) Distribution** (a Bell Curve) with a mean of zero and variance σ^2.

$$\epsilon^{(i)} \sim \mathcal{N}(0, \sigma^2)$$

The "Likelihood"

We want to find the parameters θ that make the observed data **most probable**. This is called **Maximum Likelihood Estimation (MLE)**.

Since the noise is Gaussian, the probability density of a single error is:

$$p(\epsilon^{(i)}) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(\epsilon^{(i)})^2}{2\sigma^2}\right)$$

Because $y^{(i)}$ depends on this error, maximizing the probability of the data is mathematically equivalent to minimizing the exponent term.

The Conclusion

When you do the math to maximize the Likelihood $L(\theta)$, the complicated constants drop out, and you are left with minimizing exactly the **Least Squares** term:

$$\text{Minimize} \sum_{i=1}^{n} (y^{(i)} - \theta^T x^{(i)})^2$$

Key Takeaway: Minimizing "Least Squares" is statistically the best way to fit a line if you assume the errors are normally distributed (Bell Curve).

2. Logistic Regression: The Bernoulli Assumption

The Core Question

Since Logistic Regression is for classification (outputs are 0 or 1), a Bell Curve (Gaussian) assumption doesn't make sense. Why do we use the **Log Loss** function instead?

The Assumption

The output y must be either 0 or 1. We assume the data follows a **Bernoulli Distribution** (like a weighted coin toss).

- The probability that $y = 1$ is the hypothesis $h_\theta(x)$.
- The probability that $y = 0$ is the remainder $1 - h_\theta(x)$.

The "Switch" Formula

To perform math on this, we combine these two probabilities into a single definition:

$$P(y|x;\theta) = (h_\theta(x))^y (1 - h_\theta(x))^{1-y}$$

- **If $y = 1$:** The equation becomes $(h)^1(1-h)^0 = h$ (Probability of success).
- **If $y = 0$:** The equation becomes $(h)^0(1-h)^1 = 1 - h$ (Probability of failure).

The "Likelihood"

We want to choose parameters θ that maximize the likelihood of guessing the correct label for *all* n training examples. Since examples are independent, we multiply their probabilities:

$$L(\theta) = \prod_{i=1}^{n} (h_\theta(x^{(i)}))^{y^{(i)}} (1 - h_\theta(x^{(i)}))^{1-y^{(i)}}$$

The Conclusion (Log Loss)

Multiplying probabilities results in tiny numbers. To make it easier, we take the **Log** of the likelihood. This turns the product ($\prod$) into a sum ($\sum$) and brings down the exponents:

$$l(\theta) = \sum_{i=1}^{n} \left[y^{(i)} \log(h_\theta(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)})) \right]$$

This is the **Log Likelihood**. To train the model, we want to maximize this likelihood, which is the same as minimizing the negative **Log Loss** (Cross-Entropy).

3. Calculation of Learning Algorithms (Gradient Descent)

This section details how we calculate the "update rules" (how we change the weights θ) by taking the derivative of the loss functions derived above.

A. Linear Regression Update Rule

Goal: Minimize the Cost Function $J(\theta)$ (Least Squares).

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{n}(h_\theta(x^{(i)}) - y^{(i)})^2$$

To find the minimum, we take the partial derivative with respect to a weight θ_j:

$$\frac{\partial}{\partial\theta_j}J(\theta) = (h_\theta(x) - y)x_j$$

Update Rule (LMS Algorithm):

$$\theta_j := \theta_j + \alpha\sum_{i=1}^{n}(y^{(i)} - h_\theta(x^{(i)}))x_j^{(i)}$$

(Note: We add the term because we are moving towards the target y)

B. Logistic Regression Update Rule

Goal: Maximize the Log Likelihood $l(\theta)$.

$$l(\theta) = \sum_{i=1}^{n}[y^{(i)}\log(h(x^{(i)})) + (1 - y^{(i)})\log(1 - h(x^{(i)}))]$$

To find the maximum, we take the derivative. This requires the Chain Rule because $h(x)$ is inside the log, and $h(x) = g(\theta^T x)$ (the sigmoid function).

Step-by-Step Derivation:

1. Apply derivative to the log terms:

$$\frac{\partial}{\partial\theta_j}l(\theta) = \left(y\frac{1}{g(\theta^T x)} - (1 - y)\frac{1}{1 - g(\theta^T x)}\right)\frac{\partial}{\partial\theta_j}g(\theta^T x)$$

2. The derivative of the sigmoid function $g(z)$ is $g(z)(1 - g(z))$:

$$= \left(y\frac{1}{g} - (1 - y)\frac{1}{1 - g}\right)g(1 - g)\frac{\partial}{\partial\theta_j}(\theta^T x)$$

3. Simplify the terms:

$$= (y(1 - g) - (1 - y)g)x_j$$

$$= (y - yg - g + yg)x_j$$

$$= (y - g(\theta^T x))x_j$$

Final Update Rule:

$$\theta_j := \theta_j + \alpha \sum_{i=1}^{n} (y^{(i)} - h_\theta(x^{(i)}))x_j^{(i)}$$

> **Crucial Observation:** The update rule for Logistic Regression looks **identical** to Linear Regression. However, they are different algorithms because the definition of $h_\theta(x)$ has changed (from linear to sigmoid).

4. Gist of Lecture Note (L7: Linear Models & Perceptron)

Here is a summary of the key concepts covered in the entire PDF document:

1. Basics & Notations

- **Supervised Learning:** The goal is to learn a function (hypothesis h) that maps inputs X to outputs Y.
- **Data:** We use a training set of n examples: $\{(x^{(i)}, y^{(i)}); i = 1...n\}$.
- **Types:**
  - **Regression:** Target y is continuous (e.g., price).
  - **Classification:** Target y is discrete (e.g., 0 or 1).

2. Linear Regression

- **Hypothesis:** $h_\theta(x) = \theta_0 + \theta_1 x_1 + \cdots + \theta_n x_n = \theta^T x$.
- **Cost Function:** We measure error using the "Least Squares" method ($J(\theta)$).
- **Algorithm:** We use **Gradient Descent** (specifically the LMS or Widrow-Hoff learning rule) to iteratively update weights θ to minimize error.

3. Probabilistic Interpretations

- Explains the statistical origin of the cost functions (as detailed in Sections 1 & 2 above).
  - **Linear Regression** $\rightarrow$ assumes Gaussian Noise $\rightarrow$ yields Least Squares.
  - **Logistic Regression** $\rightarrow$ assumes Bernoulli Distribution $\rightarrow$ yields Log Loss.

4. Logistic Regression

- Used for **Classification** ($y \in \{0, 1\}$).
- Uses the **Sigmoid Function** (Logistic function) to squash outputs between 0 and 1:

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

- The hypothesis $h_\theta(x)$ is interpreted as the **probability** that $y = 1$.

5. Perceptron

- Briefly introduced as another type of learning algorithm.

- Similar structure to the others but changes the definition of $g(z)$ to a hard step function (output is exactly 0 or 1, not a probability).