Generalized Linear Models (GLMs) - Complete Breakdown

---

1. In-Depth and Specific Intuitive Understanding

What are Generalized Linear Models (GLMs)?

Generalized Linear Models (GLMs) extend **Linear Regression** by allowing:

1. **Non-normal distributions** for the target variable (y).
2. **A non-linear link function** that relates the linear predictor to the target variable.

GLMs unify **Linear Regression**, **Logistic Regression**, and **Poisson Regression** into a **single mathematical framework**.

Key Idea

GLMs have three main components:

1. **Random Component**: Defines the probability distribution of y (e.g., Normal, Bernoulli, Poisson).
2. **Systematic Component**: Defines the linear predictor η:

$$\eta = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n$$

3. **Link Function**: Transforms the linear predictor into a suitable range for the response variable:

$$g(\mathbb{E}[y]) = \eta$$

This allows us to model different types of response variables.

---

2. When GLMs are Used and When They Should Be Avoided

✅ When to Use GLMs

- When the **target variable is non-Gaussian** (binary, count data, etc.).
- When a **linear model** is needed but **not limited to normal errors**.
- When **interpretable coefficients** are required.

❌ When to Avoid GLMs

- If the relationship between features and target is **highly non-linear**, requiring more complex models.
- If the dataset is **very large**, some GLM variants may be computationally slow.
- If the target variable is **hierarchical or has dependencies**, GLMs may not capture structure well (use hierarchical models instead).

3. When It Fails to Converge and How to Avoid That

When GLMs Fail

- **Multicollinearity in features** → Causes instability in estimation.
- **Perfect separation in logistic regression** → Infinite coefficients.
- **Outliers in Poisson Regression** → Large influence on estimated parameters.
- **Incorrect choice of link function** → Poor convergence.

How to Ensure Convergence

✅ **Feature scaling** (standardization helps numerical stability).
✅ **Use regularization (Ridge/Lasso)** for multicollinearity.
✅ **Choose an appropriate link function** based on data distribution.
✅ **Apply robust estimation techniques** if outliers are present.

When GLMs Always Converge

- When **data meets assumptions** (correct distribution and no perfect separation).
- When **regularization is applied** to handle numerical instability.
- When the **iteratively reweighted least squares (IRLS) algorithm** is used instead of basic gradient descent.

---

4. Advantages and Disadvantages

Advantages

✅ **Unifies multiple models** under one framework (Linear, Logistic, Poisson, etc.).

✅ **Interpretable coefficients.**

✅ **Can model many types of response variables** beyond just continuous numbers.

✅ **Can be extended with regularization (Elastic Net, Ridge, Lasso).**

Disadvantages

❌ **Choosing the correct link function is critical** (wrong choice leads to poor predictions).

❌ **Computationally expensive for large datasets.**

❌ **Sensitive to outliers in some distributions (e.g., Poisson regression).**

5. Intuitive Algorithm / Pseudo Code

1. **Initialize weights** w and bias w_0.
2. **For each training epoch**:
   - Compute linear predictor η:

   $$\eta = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n$$

   - Apply **link function**:

   $$g(\mathbb{E}[y]) = \eta$$

   - Compute **loss** (depends on the GLM type).
   - Compute **gradients**:

   $$w_j := w_j - \alpha \frac{\partial J}{\partial w_j}$$

   - Update weights and bias.
3. **Repeat until convergence.**

6. Mathematical and Logical Breakdown

General Form of GLMs

GLMs assume:

$$y \sim \text{Exponential Family Distribution}$$

with:

1. **Linear predictor**:

$$\eta = Xw$$

2. **Link function**:

$$g(\mathbb{E}[y]) = \eta$$

3. **Loss function** (varies based on the model):
   - **Linear Regression (Identity Link, Gaussian Response):**

$$J(w) = \sum (y - Xw)^2$$

   - **Logistic Regression (Logit Link, Bernoulli Response):**

$$J(w) = -\sum y \log p + (1 - y)\log(1 - p)$$

   - **Poisson Regression (Log Link, Poisson Response):**

$$J(w) = \sum y \log \lambda - \lambda$$

How to Determine the Link Function $g(y)$ in Generalized Linear Models (GLMs)

In **Generalized Linear Models (GLMs)**, the link function $g(y)$ transforms the **expected value** $\mathbb{E}[y]$ into a linear function of the predictors:

$$g(\mathbb{E}[y]) = \eta = Xw$$

where Xw is the linear predictor.

Choosing the Right Link Function

To determine the **appropriate link function**, follow these **three key steps**:

1. Identify the Distribution of the Response Variable

GLMs assume that the response variable y follows an **exponential family distribution**:

$$p(y|\theta) = \exp\left(\frac{y\theta - b(\theta)}{a(\phi)} + c(y, \phi)\right)$$

where:

- θ is the **natural parameter** (dependent on Xw).
- $b(\theta)$ defines the **mean function** of y.
- $a(\phi)$ is a scaling function.
- $c(y, \phi)$ normalizes the distribution.

The most common **exponential family distributions** used in GLMs are:

Response Type	Distribution	Mean $\mathbb{E}[y]$
Continuous	Normal (Gaussian)	μ
Binary (0/1)	Bernoulli	p
Count Data	Poisson	λ

2. Use the Canonical Link Function

A **canonical link function** is the **natural choice** derived from the probability distribution of y. It makes **maximum likelihood estimation** (MLE) easier and ensures **convergence**.

The canonical link function is given by:

$$g(\mathbb{E}[y]) = \theta$$

where θ is the **natural parameter** of the exponential family distribution.

Canonical Link Functions for Common GLMs

Response Type	Distribution	$g(y)$ (Canonical Link)	Interpretation
Continuous	Normal	$g(y) = y$ (Identity)	No transformation (Linear Regression)
Binary (0/1)	Bernoulli	$g(y) = \log \frac{p}{1-p}$ (Logit)	Converts probabilities into log-odds (Logistic Regression)
Count Data	Poisson	$g(y) = \log(y)$ (Log)	Ensures positive count values (Poisson Regression)

3. Consider Interpretability and Stability

While the **canonical link function** is often the best choice, there are cases where other link functions are preferable for **better interpretability or numerical stability**.

Alternative Link Functions and When to Use Them

Response Type	Canonical Link	Alternative Link	When to Use Alternative
Binary (0/1)	Logit $\log \frac{p}{1-p}$	Probit $\Phi^{-1}(p)$	When data is normally distributed in probability space
Binary (0/1)	Logit $\log \frac{p}{1-p}$	Complementary log-log $\log(-\log(1-p))$	When extreme probabilities (near 0 or 1) are common
Count Data	Log $\log(y)$	Identity y	When the response has small counts, including zeros
Continuous	Identity y	Log $\log(y)$	When variance is proportional to the mean (Gamma Regression)

Mathematical Derivation of the Link Function

1. **Start with the probability distribution of y:**

   - For **Poisson Regression**:

   $$P(y) = \frac{e^{-\lambda}\lambda^y}{y!}$$

   - For **Logistic Regression**:

   $$P(y=1) = p, \quad P(y=0) = 1-p$$

2. **Find the mean function $\mathbb{E}[y]$:**

   - Poisson: $\mathbb{E}[y] = \lambda$.
   - Logistic: $\mathbb{E}[y] = p$.

3. **Determine the canonical parameter θ:**

   - Poisson: $\theta = \log(\lambda)$.
   - Logistic: $\theta = \log \frac{p}{1-p}$.

4. **Set the canonical link function:**

   $$g(\mathbb{E}[y]) = \theta$$

   - Poisson: $g(y) = \log(y)$.
   - Logistic: $g(y) = \log \frac{p}{1-p}$.

Example: Determining the Link Function

Scenario 1: Predicting House Prices

- The response variable is **continuous**.
- **Distribution:** Normal (Gaussian).
- **Canonical Link:** Identity $g(y) = y$.
- **Conclusion:** Use **Linear Regression**.

Scenario 2: Predicting Whether a Customer Buys a Product (Yes/No)

- The response variable is **binary** (0/1).
- **Distribution:** Bernoulli.
- **Canonical Link:** Logit $g(y) = \log \frac{p}{1-p}$.
- **Alternative:** Probit (if data follows a normal distribution).
- **Conclusion:** Use **Logistic Regression**.

Scenario 3: Predicting the Number of Website Clicks per Day

- The response variable is a **count** (0, 1, 2, ...).
- **Distribution:** Poisson.
- **Canonical Link:** Log $g(y) = \log(y)$.
- **Conclusion:** Use **Poisson Regression**.

Gradient Descent Update Rules

For a GLM, we use:

$$\frac{\partial J}{\partial w_j} = X^T(g^{-1}(\eta) - y)$$

$$w_j := w_j - \alpha \frac{\partial J}{\partial w_j}$$

where g^{-1} is the **inverse of the link function.**

---

7. Manual Implementation in Python

Copy Edit

```python
import numpy as np

class GeneralizedLinearModel:
    def __init__(self, link_function="logit", learning_rate=0.01, epochs=1000):
        self.link_function = link_function
        self.learning_rate = learning_rate
        self.epochs = epochs
        self.weights = None

    def _link(self, z):
        """Apply the link function based on the GLM type."""
        if self.link_function == "identity":  # Linear Regression
            return z
        elif self.link_function == "logit":  # Logistic Regression
            return 1 / (1 + np.exp(-z))
        elif self.link_function == "log":  # Poisson Regression
            return np.exp(z)
        else:
            raise ValueError("Unsupported link function")

    def fit(self, X, y):
        """Train the model using gradient descent."""
        m, n = X.shape
        self.weights = np.zeros(n)

        for _ in range(self.epochs):
            z = np.dot(X, self.weights)
            y_pred = self._link(z)
            gradient = (1/m) * np.dot(X.T, (y_pred - y))
            self.weights -= self.learning_rate * gradient

    def predict(self, X):
        """Make predictions using the trained model."""
        z = np.dot(X, self.weights)
        return self._link(z)
```

8. Scikit-Learn Implementation (Fully Commented)

python

Copy Edit

```python
from sklearn.linear_model import LinearRegression, LogisticRegression, PoissonRegres
from sklearn.model_selection import train_test_split
from sklearn.datasets import make_regression, make_classification
import numpy as np

# Example 1: Linear Regression (Identity Link)
X, y = make_regression(n_samples=100, n_features=2, noise=10, random_state=42)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_stat
model = LinearRegression()
model.fit(X_train, y_train)
print("Linear Regression Coefficients:", model.coef_)

# Example 2: Logistic Regression (Logit Link)
X, y = make_classification(n_samples=100, n_features=2, n_classes=2, random_state=42
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_stat
model = LogisticRegression()
model.fit(X_train, y_train)
print("Logistic Regression Coefficients:", model.coef_)

# Example 3: Poisson Regression (Log Link)
y = np.random.poisson(lam=3, size=100)
model = PoissonRegressor()
model.fit(X_train, y)
print("Poisson Regression Coefficients:", model.coef_)
```

Final Summary

- GLMs unify different regression models under a common framework.
- The choice of link function determines the model type.
- GLMs work for binary, count, and continuous data.
- Choosing the right link function is crucial for performance.