Lecture Notes: Prediction

Generative and predictive models are integral in statistical modeling and machine learning, each fulfilling distinct objectives. Generative models aim to replicate the underlying probability distribution of data, offering insights into the data generation process. Predictive models, in contrast, focus on forecasting future outcomes from observed data patterns.

One notable feature of these models is the comparison of errors. Typically, the error in generative processes is smaller or equal to that in predictive processes. For example, consider a generative process where X_1 is a normally distributed variable ($X_1 \sim \mathcal{N}(0,1)$), and an associated variable Y is a function of X_1 with added noise

$$Y = X_1 + \epsilon_1, \qquad \epsilon_1 \sim \mathcal{N}(0,1). \tag{1}$$

Additionally, a second variable X_2 is derived from Y with further noise

$$X_2 = 3Y + \epsilon_2, \qquad \epsilon_2 \sim \mathcal{N}(0,1) \tag{2}$$

In predictive modeling, specifically in linear regression, the objective is to estimate a function that best describes the relationship between explanatory variables and the response variable. The equation

$$\hat{Y} = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \epsilon \tag{3}$$

exemplifies this, where the model aims to minimize the Mean Squared Error (MSE) for optimal coefficient determination. Key elements of this approach include:

- Finding coefficients (β_0, β_1, β_2) that minimize MSE.
- Analyzing the linear relationship between Y and predictors X_1, X_2.
- Employing the model to predict Y using X_1 and X_2.

The coefficients from our linear regression model, estimated using training data, are summarized in the table below:

Coefficient	**Value**
Intercept (Constant)	0.003203
Coefficient of X_1	0.099575
Coefficient of X_2	0.301211

Table 1: Summary of Linear Regression Model Coefficients

Error analysis in predictive modeling is vital for evaluating a model's performance. It involves calculating the discrepancy between observed and predicted values as a measure of accuracy.

To visually compare the error distributions of generative and predictive models, a violin plot is presented:

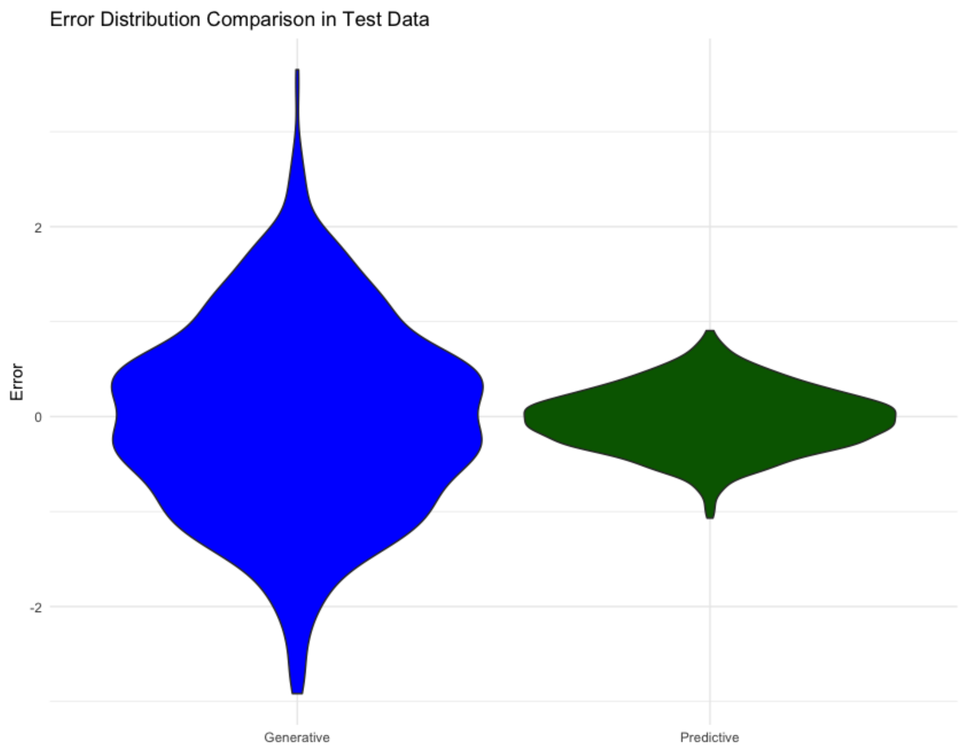

Figure 1: Comparison of Error Distributions in Predictive and Generative Models

Predicting binary variables

Consider the case where a random variable, Y, is binary. Typically, $Y = 1$ with probability π_i and $Y = 0$ with probability $1 - \pi_i$. The aim of logistic regression is to model the probability π_i as a function of predictor variables.

For the i-th observation, the probability π_i of observing $Y = 1$ is modeled using the logistic function:

$$\pi_i = \frac{1}{1 + e^{-z}}$$

where z represents a linear combination of predictor variables, given by:

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

Here, $x_1, x_2, \ldots, x_n$ are the predictor variables, and $\beta_0, \beta_1, \beta_2, \ldots, \beta_n$ are the coefficients to be estimated from the data.

The logistic function, $\sigma(z) = \frac{1}{1+e^{-z}}$, maps the linear combination z to a probability value between 0 and 1. This characteristic is crucial for modeling binary outcomes where the response variable can only take two distinct values.

Estimation Process

Logistic regression, often used for binary classification problems, can be estimated using various methods. One such approach, familiar from previous classes, is the Mean Squared Error (MSE).

In logistic regression, the MSE is calculated based on the difference between the observed binary outcomes and the predicted probabilities. The MSE is defined as:

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^{N} (p_i - y_i)^2 \tag{4}$$

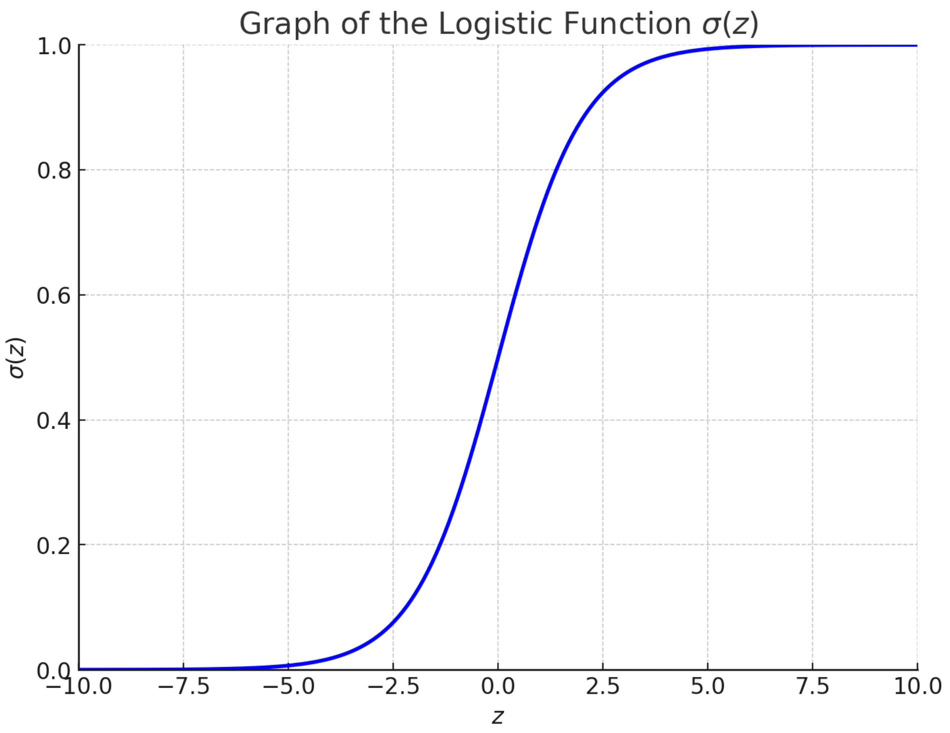

Figure 2: Graph of the logistic function $\sigma(z)$

where $p_i = \frac{1}{1+e^{-z_i}}$, $z_i = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_k x_{ik}$, and y_i are the observed binary outcomes.

To find the optimal parameters for the logistic regression model using MSE, we calculate the derivative of MSE with respect to the parameters and set it to zero. However, as we will see, this leads to a complex equation that is difficult to solve analytically.

The derivative of MSE with respect to a parameter β_j is given by:

$$\frac{\partial \text{MSE}}{\partial \beta_j} = \frac{2}{N} \sum_{i=1}^{N} (p_i - y_i) \frac{\partial p_i}{\partial \beta_j} \tag{5}$$

the derivative of p_i with respect to z_i is given by:

$$\frac{\partial p_i}{\partial z_i} = \frac{\partial}{\partial z_i} \left(\frac{1}{1+e^{-z_i}} \right)$$

Using the chain rule, this becomes:

$$\frac{\partial p_i}{\partial z_i} = \frac{e^{-z_i}}{(1+e^{-z_i})^2} = p_i(1-p_i)$$

The derivative of z_i with respect to a parameter β_j is:

$$\frac{\partial z_i}{\partial \beta_j} = x_{ij}$$

Thus, using the chain rule, the derivative of p_i with respect to β_j is:

$$\frac{\partial p_i}{\partial \beta_j} = \frac{\partial p_i}{\partial z_i} \cdot \frac{\partial z_i}{\partial \beta_j} = p_i(1-p_i)x_{ij}$$

Substituting and expanding, we get:

$$\frac{\partial \text{MSE}}{\partial \beta_j} = \frac{2}{N} \sum_{i=1}^{N} (p_i - y_i) p_i (1-p_i) x_{ij} \tag{6}$$

Setting this derivative to zero for optimization:

$$\frac{2}{N}\sum_{i=1}^{N}(p_i - y_i)p_i(1 - p_i)x_{ij} = 0 \tag{7}$$

However, solving this equation directly for β_j is not straightforward due to the non-linear nature of the logistic function embedded in p_i. This non-linearity introduces complexity, making it challenging to find a closed-form solution for the parameters β_j.

Gradient Descent

Gradient descent is an iterative optimization algorithm used for finding the minimum of a function. In the context of MSE in logistic regression, the gradient descent algorithm updates the model parameters by moving in the direction that reduces MSE.

Gradient descent is a first-order iterative optimization algorithm for finding the minimum of a function. In the context of machine learning, this function is often a loss function that we want to minimize.

Given a loss function $L(\beta)$, where β represents the parameters of our model, the idea of gradient descent is to update the parameters β iteratively in the direction of steepest descent. The update rule is given by:

$$\beta := \beta - \alpha \nabla L(\beta) \tag{8}$$

where:

- α is the learning rate, which determines the step size in the direction of the gradient.
- $\nabla L(\beta)$ is the gradient of the loss function, which gives the direction of steepest ascent.

By subtracting the gradient scaled by the learning rate from the current parameters, we move in the direction of steepest descent, thereby reducing the loss function until we hopefully reach a minimum.

The gradient descent algorithm iteratively updates the parameters until it converges to a minimum. Convergence is typically determined by either a small change in the loss function between iterations or reaching a predetermined number of iterations.

Example

This example demonstrates logistic regression with a dataset of 8 rows, using a single feature X and a binary outcome Y. We will illustrate the parameter update process using gradient descent.

Dataset: Consider the following dataset:

X	Y
1	0
2	0
3	0
4	1
5	1
6	1
7	1
8	1

Here, X is the feature, and Y is the binary outcome.

Model Setup: We form the parameter vector $\mathbf{w}' = [w, b]$ by combining the weight w and bias b. Initially, let's set $w = 0$ and $b = 0$. The augmented input data X' includes the original feature X and a constant 1 for the bias term.

Gradient Descent Optimization: We will apply one iteration of gradient descent with a learning rate $\eta = 0.01$. The update rule for the parameter vector $\mathbf{w}'$ is:

$$\mathbf{w}'_{\text{new}} = \mathbf{w}'_{\text{old}} - \eta \nabla \ell(\mathbf{w}'_{\text{old}}) \tag{9}$$

Calculations: First, calculate the predicted probabilities for each data point with the initial parameters:

$$P(Y = 1|X' = 1) = \frac{1}{1 + e^{-(0 \cdot 1 + 0)}} = 0.5$$

$$P(Y = 1|X' = 2) = \frac{1}{1 + e^{-(0 \cdot 2 + 0)}} = 0.5$$

... and so on for $X' = 3, 4, \ldots, 8$.

Next, compute the gradient of the loss function with respect to $\mathbf{w}'$ for each data point and sum them up to get the overall gradient. For simplicity, we'll show the calculation for just one data point here:

$$\frac{\partial}{\partial w} \ell(\mathbf{w}', 1) = (0.5 - 0) \cdot 1 = 0.5$$

$$\frac{\partial}{\partial b} \ell(\mathbf{w}', 1) = 0.5 - 0 = 0.5$$

Finally, update the parameters using the average gradient across all data points:

$$w_{\text{new}} = w_{\text{old}} - \eta \times \text{Average Gradient w.r.t. } w$$

$$b_{\text{new}} = b_{\text{old}} - \eta \times \text{Average Gradient w.r.t. } b$$

Conclusion: This example demonstrates the initial step of parameter optimization in logistic regression using gradient descent. In practice, multiple iterations over the entire dataset are necessary, and the parameters are updated iteratively until convergence.

Beyond the MSE

In logistic regression, we deal with a binary outcome variable, Y, which takes values 0 or 1. For each observation i, the model predicts a probability $\pi_i = P(Y_i = 1|X_i)$ based on input features X_i. The probability of observing the actual outcome Y_i given this predicted probability is expressed as:

$$P(Y_i|X_i) = \pi_i^{Y_i} \cdot (1 - \pi_i)^{1 - Y_i}$$

This formula accounts for both possible outcomes:

- If $Y_i = 1$, the probability is π_i.
- If $Y_i = 0$, the probability is $1 - \pi_i$.

The overall probability for all observations is the product of these individual probabilities:

$$L(\beta) = \prod_{i=1}^{n} P(Y_i|X_i)$$

Taking the logarithm of this probability function, we get the log-probability:

$$\log(L(\beta)) = \sum_{i=1}^{n} [Y_i \log(\pi_i) + (1 - Y_i) \log(1 - \pi_i)]$$

The negative log-probability, the function we aim to minimize, is thus:

$$-\log(L(\beta)) = -\sum_{i=1}^{n} [Y_i \log(\pi_i) + (1 - Y_i) \log(1 - \pi_i)]$$

In this formulation, a lower negative log-probability value indicates a better model fit. Accurate predictions lead to smaller values in the summation, reflecting lower prediction error, while inaccurate predictions result in larger values, indicating higher error. Therefore, minimizing this function aligns with reducing inaccuracies in the model's predictions.

The optimization of the logistic regression model is performed using gradient descent, focusing on the parameter vector $\mathbf{w}'$. The update rule for gradient descent is:

$$\mathbf{w}'_{\text{new}} = \mathbf{w}'_{\text{old}} - \eta \nabla \ell(\mathbf{w}'_{\text{old}}) \tag{10}$$

In this equation, η represents the learning rate and $\nabla \ell(\mathbf{w}'_{\text{old}})$ is the gradient of the loss function with respect to the parameter vector.

The gradient of the loss function with respect to the parameter vector $\mathbf{w}$ is obtained by taking partial derivatives:

$$\nabla \ell(\mathbf{w}) = \sum_{i=1}^{n} (p_i - y_i) X_i \tag{11}$$

Each term $(p_i - y_i)$ represents the error between the predicted probability and the actual outcome, and X_i is the feature vector for the i-th observation.

In a vectorized implementation, where X is the matrix of input features and Y is the vector of outcomes, the gradient can be expressed more compactly:

$$\nabla \ell(\mathbf{w}) = X^T (P - Y) \tag{12}$$

Here, P is the vector of predicted probabilities for all observations, and X^T is the transpose of the feature matrix.

Prediction and performance evaluation

Logistic regression leverages the logistic (or sigmoid) function, denoted as σ, to estimate the probability of a binary outcome based on predictor variables. The probability that the outcome y equals 1, given predictors $\mathbf{x}$, is given by the logistic function:

$$P(y = 1|\mathbf{x}) = \sigma(\mathbf{w}^T \mathbf{x} + b) = \frac{1}{1 + e^{-(\mathbf{w}^T \mathbf{x} + b)}}.$$

The model delineates a linear decision boundary in the feature space. A data point is classified based on which side of the boundary it falls on:

$$\hat{y} = \begin{cases} 1 & \text{if } \mathbf{w}^T \mathbf{x} + b \geq 0, \\ 0 & \text{otherwise.} \end{cases}$$

This mechanism allows logistic regression to make predictions on new, unseen data by applying the learned linear function. The model outputs the likelihood of each instance belonging to the positive class.

Extending logistic regression beyond binary classification, techniques like One-vs-Rest (OvR) or multinomial logistic regression enable it to handle multiclass classification scenarios.

In assessing feature importance, the model's coefficients, $\mathbf{w}$, reveal the influence of each predictor. Features with larger absolute values of coefficients have a greater impact on the prediction.

Performance metrics are calculated for each resample based on the model's predictions and the true labels of the resampled data.

1. **Accuracy:** The ratio of correct predictions to the total number of predictions.

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$$

2. **Precision:** The ratio of true positive predictions to the total number of true positive and false positive predictions.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

3. **Recall:** The ratio of true positive predictions to the total number of true positive and false negative predictions.

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

4. **F1 Score:** The harmonic mean of precision and recall.

$$\text{F1 Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

5. **Kappa:** A statistic that measures inter-rater agreement for categorical items, adjusting for chance agreement.

$$\text{Kappa} = \frac{P_o - P_e}{1 - P_e}$$

where P_o is the observed agreement, and P_e is the expected agreement by chance.

Loan Prediction Example

We have a dataset representing loan applicants, comprising various features that are typically considered in loan approval decisions. Each row represents an individual applicant, with the following columns:

- **AnnualIncome**: The applicant's annual income in dollars.
- **EmploymentStatus**: Binary indicator of employment status (1 for employed, 0 for unemployed).
- **OpenCreditLines**: Number of open credit lines.
- **CreditScore**: Credit score of the applicant.
- **DebtToIncomeRatio**: Ratio of debt to income.
- **LoanApproved**: Binary outcome of the loan application (1 for approved, 0 for denied).

AnnualIncome	EmployStatus	OpenCreditLines	CreditScore	DebtToIncomeRatio	LoanApproved
76460	1	2	628	0.20	1
56002	1	9	675	0.37	1
64681	1	5	672	0.20	0
...	...	...	...	...	...
56029	0	7	747	0.26	1

Table 2: Sample data from the loan application dataset.

Below is a sample from the dataset, providing a glimpse into the nature of the data used for logistic regression analysis.

The estimated logistic regression model for predicting loan approval is given by the following equation:

$$\begin{aligned}\log\left(\frac{P(\text{Loan Approved}|\mathbf{x})}{1-P(\text{Loan Approved}|\mathbf{x})}\right) &= 3.644\cdot(\text{Intercept})\\ &- 3.058\times10^{-5}\cdot(\text{AnnualIncome})\\ &+ 0.4219\cdot(\text{EmploymentStatus})\\ &+ 0.03151\cdot(\text{OpenCreditLines})\\ &- 0.003686\cdot(\text{CreditScore})\\ &- 0.3325\cdot(\text{DebtToIncomeRatio})\end{aligned}$$

Parameter Estimation

The logistic regression model parameters and their statistical significance are presented in the table below:

Parameter	**Estimate**	**Std. Error**	**z value**	**Pr(>|z|)**
Intercept	3.644e+00	3.524e+00	1.034	0.3011
AnnualIncome	-3.058e-05	1.805e-05	-1.694	0.0902
EmploymentStatus	4.219e-01	5.022e-01	0.840	0.4009
OpenCreditLines	3.151e-02	7.770e-02	0.405	0.6851
CreditScore	-3.686e-03	4.544e-03	-0.811	0.4172
DebtToIncomeRatio	-3.325e-01	2.015e+00	-0.165	0.8689

Table 3: Estimated coefficients of the logistic regression model.

The coefficients represent the change in the log odds of loan approval for a one-unit change in each predictor variable. The p-values indicate the statistical significance of each coefficient.

Once the logistic regression model was evaluated on the test dataset to determine its predictive performance. The following metrics were calculated:

- **Accuracy:** The ratio of correct predictions to the total number of predictions.

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}} = 55.5\%$$

- **Precision:** The ratio of true positive predictions to the total number of true positive and false positive predictions.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} = 55.07\%$$

- **Recall:** The ratio of true positive predictions to the total number of true positive and false negative predictions.

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} = 39.58\%$$

- **F1 Score:** The harmonic mean of precision and recall.

$$\text{F1 Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} = 46.06\%$$

- **Kappa:** A statistic that measures inter-rater agreement for categorical items, adjusting for chance agreement.

$$\text{Kappa} = \frac{P_o - P_e}{1 - P_e} = 0.0988$$

  where P_o is the observed agreement, and P_e is the expected agreement by chance.

The model demonstrates moderate predictive ability, with a balanced performance across accuracy, precision, and recall. However, the low Cohen's Kappa score indicates the need for further model improvement.