MSDM-5054 homework-2

TIANYAO TAN

October 10 2025

1 Problem 1: Basic Knowledge

1.1 Question:

Show that the log-likelihood function is concave in logistic regression.

1.1.1 Answer:

We shall analyse second derivatives in logistic regression is concave, also known as the Hessian matrix. If the Hessian matrix of a function is negative semidefinite, the function is deemed concave.

In logistic regression, the model predicts the probability of a binary outcome. For a single data point $(\mathbf{x}_i, y_i)$, the probability of the positive class $(y_i = 1)$ is given by the sigmoid function:

$$p_i = P(y_i = 1|\mathbf{x}_i, \beta) = \frac{1}{1 + e^{-\mathbf{x}_i^T \beta}}$$

For a dataset containing N observations, the log-likelihood function is the sum of the log-likelihoods for each observation.

$$\ell(\beta) = \sum_{i=1}^{N} [y_i \log(p_i) + (1 - y_i) \log(1 - p_i)]$$

To determine the concavity of this function, we compute its Hessian matrix $\mathbf{H}$. After differentiation, the Hessian matrix of the log-likelihood function can be expressed in compact matrix form:

$$\mathbf{H} = -\mathbf{X}^T \mathbf{W} \mathbf{X}$$

We know:

- $\mathbf{X}$ is an $N \times d$ design matrix, where each row represents an observation and each column represents a feature.
- $\mathbf{W}$ is an $N \times N$ diagonal matrix. The diagonal elements w_{ii} are calculated as $w_{ii} = p_i(1 - p_i)$. All off-diagonal elements are zero.

A matrix is negative semidefinite if, for any non-zero vector $\mathbf{v}$, its quadratic form $\mathbf{v}^T \mathbf{H} \mathbf{v}$ is less than or equal to zero. Consider the quadratic form of the Hessian matrix:

$$\begin{aligned} \mathbf{v}^T \mathbf{H} \mathbf{v} &= \mathbf{v}^T (-\mathbf{X}^T \mathbf{W} \mathbf{X}) \mathbf{v} \\ &= -(\mathbf{v}^T \mathbf{X}^T) \mathbf{W} (\mathbf{X} \mathbf{v}) \\ &= -(\mathbf{X} \mathbf{v})^T \mathbf{W} (\mathbf{X} \mathbf{v}) \end{aligned}$$

Let us define a new vector $\mathbf{z} = \mathbf{X}\mathbf{v}$. The expression then becomes:

$$-\mathbf{z}^T \mathbf{W} \mathbf{z}$$

Since the predicted probability p_i always lies between 0 and 1, the term $w_{ii} = p_i(1 - p_i)$ is always non-negative $(w_{ii} \geq 0)$. As $\mathbf{W}$ is a diagonal matrix with these non-negative values on its diagonal, it is a positive semidefinite matrix.

$\mathbf{z}^T\mathbf{W}\mathbf{z}$ is always greater than or equal to zero.

$$\mathbf{z}^T\mathbf{W}\mathbf{z} = \sum_{i=1}^{N} w_{ii} z_i^2 = \sum_{i=1}^{N} p_i(1-p_i) z_i^2 \geq 0$$

Therefore, owing to the presence of the leading negative sign, $-\mathbf{z}^T\mathbf{W}\mathbf{z}$ is always less than or equal to 0.

$$\mathbf{v}^T\mathbf{H}\mathbf{v} = -\mathbf{z}^T\mathbf{W}\mathbf{z} \leq 0$$

This indicates that the Hessian matrix of the log-likelihood function is negative semidefinite.

Since the Hessian matrix of the log-likelihood function in logistic regression is negative semidefinite, the function is concave.

1.2 Question:

Besides the gradient descent algorithm and Newton algorithm, state another algorithm which can be used to find MLE for logistic regression. Explain the details and its pros and cons compared to Newton algorithm.

1.2.1 Answer:

Alternative Algorithms: Quasi-Newton Methods (e.g., BFGS)

Quasi-Newton methods serve as alternatives to the Newton method. These algorithms capture second-order curvature information from the Newton method without incurring high computational costs. The most widely used quasi-Newton algorithm is the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm.

Its core idea avoids computing the true Hessian matrix ($\mathbf{H}$) and its inverse at each step. Instead, BFGS constructs and refines an approximation of the inverse Hessian matrix using only first-order gradient information.

Advantages and Disadvantages Compared to Newton's Method

(a) Advantages of BFGS (Pseudonewtonian Method)

(1) Computational efficiency: The primary advantage is avoiding direct computation and inversion of the Hessian matrix. Computing the Hessian matrix has a complexity of $O(d^2)$, while finding its inverse has a complexity of $O(d^3)$, where d is the number of features. BFGS uses simpler vector operations to update its approximations, making each iteration faster.

(2) Rapid Convergence Rate: BFGS achieves superlinear convergence speed. This is significantly faster than the linear speed of gradient descent. In practice, it strikes an excellent balance between speed and the number of iterations required.

(3) Robustness and Stability: The BFGS update formula ensures the Hessian approximation ($\mathbf{B}_k$) remains positive definite. This guarantees descent directions are always valid, making the algorithm more stable and reliable than Newton's method.

(b) Disadvantages of BFGS (Quasi-Newton Method)

(1) Memory Requirements: The standard BFGS algorithm must store a dense $d \times d$ matrix approximation ($\mathbf{B}_k$). This consumes substantial memory when the number of features (d) is very large.

(2) Slower convergence than Newton's method: Although it converges superlinearly at high speed, it is not as fast as Newton's method's quadratic convergence rate. Therefore, for problems where the Hessian matrix is easy to compute and invert, Newton's method may reach the optimal solution in fewer iterations.

(3) Lower accuracy than Newton's method: Since the Hessian matrix is only an approximation, it may converge less precisely than Newton's method.

1.3 Question:

In a logistic regression model, how would you interpret the coefficient β_i associated with a continuous predictor variable X_i?

1.3.1 Answer:

The logistic regression model is defined as:

$$\log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 X_1 + \cdots + \beta_i X_i + \cdots + \beta_n X_n$$

$\log\left(\frac{p}{1-p}\right)$, is the log-odds of the event occurring.

β_i is:

Holding all other predictor variables constant, a one-unit increase in X_i is associated with a change of β_i in the log-odds of the outcome.

We can exponentiate the coefficient, e^{β_i}:

Holding all other predictor variables constant, for a one-unit increase in X_i, the odds of the outcome occurring are multiplied by a factor of e^{β_i}.

This can be broken down based on the sign of β_i:

(1) If $\beta_i > 0$, then $e^{\beta_i} > 1$. A one-unit increase in X_i increases the odds of the outcome.

(2) If $\beta_i < 0$, then $0 < e^{\beta_i} < 1$. A one-unit increase in X_i decreases the odds of the outcome.

(3) If $\beta_i = 0$, then $e^{\beta_i} = 1$. The variable X_i has no effect on the odds of the outcome.

1.4 Question:

Suppose you can access the `LogisticRegression()` function in a Python library. Write the pseudocode to draw the ROC curve of logistic regression on the training data $\mathcal{D}_{\text{train}}$.

1.4.1 Answer:

```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_classification
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_curve, auc

X, y = make_classification(n_samples=1000, n_features=20, n_informative=2, n_redundant=10, random_state=42)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

model = LogisticRegression()
model.fit(X_train, y_train)

y_scores = model.predict_proba(X_train)[:, 1]

fpr, tpr, thresholds = roc_curve(y_train, y_scores)
roc_auc = auc(fpr, tpr)

plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (area = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--', label='Random Classifier')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate (FPR)')
plt.ylabel('True Positive Rate (TPR)')
plt.title('ROC Curve for Logistic Regression on Training Data')
plt.legend(loc="lower right")
plt.grid(True)
plt.savefig('ROC_Curve.png')
plt.show()
```

Listing 1: "Python code for generating an ROC curve."

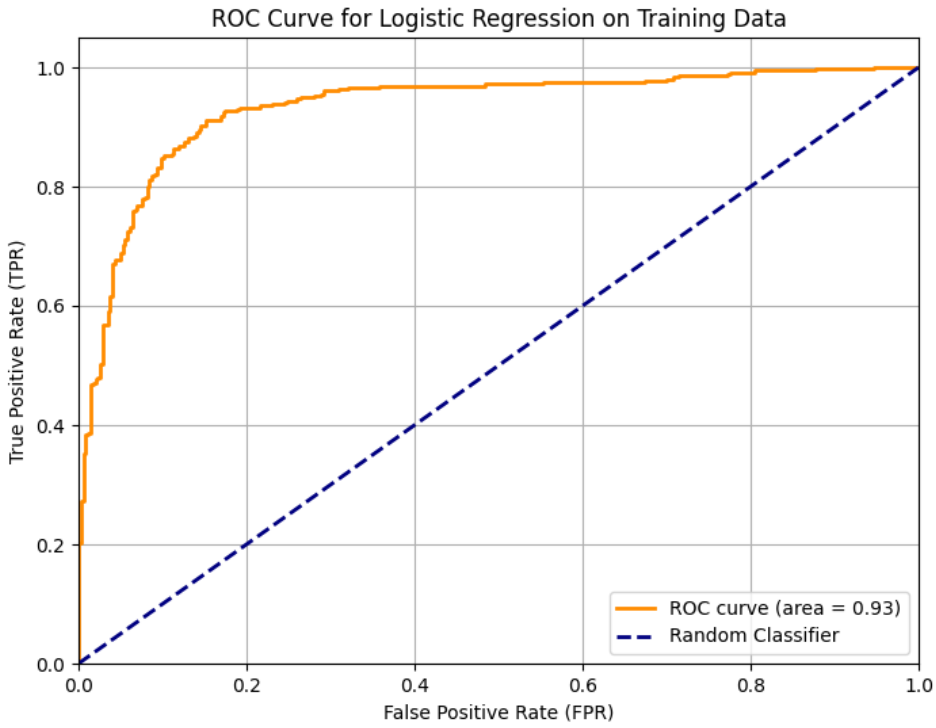

Figure 1: ROC Curve.png

1.5 Question:

Explain the primary difference between LDA and Logistic Regression in the context of classification tasks.

1.5.1 Answer:

The primary difference is that Linear Discriminant Analysis (LDA) is a generative model, while Logistic Regression is a discriminative model.

(a) Logistic Regression: The Discriminative Approach

Logistic Regression models the conditional probability of the class given the input features, $P(Y|X)$, directly.

Assumptions: It makes very few assumptions about the underlying distribution of the features (X). This makes it more flexible and robust, especially if the data isn't normally distributed.

Performance: More robust if assumptions are violated. Better on large datasets.

(b) Linear Discriminant Analysis (LDA): The Generative Approach

LDA works by modeling the distribution of the features (X) for each class (Y) separately. It models the class-conditional probability, $P(X|Y)$, along with the prior probability of each class, $P(Y)$. It then uses Bayes' theorem to compute the posterior probability $P(Y|X)$ that we actually want.

Assumptions: LDA has strong assumptions:

(1) The features (X) within each class follow a multivariate Gaussian distribution.
(2) Each class has an identical covariance matrix ($\Sigma_1 = \Sigma_2 = \cdots = \Sigma_k$).

Performance: More stable and efficient on small datasets if assumptions hold.

1.6 Question:

Derive the discrimination function $\delta_k(x)$ in LDA using Bayes' Theorem.

1.6.1 Answer:

We begin with Bayes' theorem:

$$P(Y = k \mid x) = \frac{P(x \mid Y = k)P(Y = k)}{P(x)}$$

Since we are performing classification, we are interested in which category has the highest posterior probability. Therefore, we can ignore the denominator $P(x)$ and focus on:

$$\delta_k(x) = \log P(x \mid Y = k) + \log P(Y = k)$$

Assuming the conditional density of each class follows a Gaussian distribution:

$$x \mid Y = k \sim \mathcal{N}(\mu_k, \Sigma)$$

Then:

$$P(x \mid Y = k) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x - \mu_k)^T \Sigma^{-1}(x - \mu_k)\right)$$

Taking the logarithm yields:

$$\log P(x \mid Y = k) = -\frac{1}{2}(x - \mu_k)^T \Sigma^{-1}(x - \mu_k) - \frac{1}{2}\log|\Sigma| - \frac{d}{2}\log(2\pi)$$

Let $\pi_k = P(Y = k)$, then:

$$\delta_k(x) = \log P(x \mid Y = k) + \log \pi_k$$

Neglecting the constant term, we obtain:

$$\delta_k(x) = -\frac{1}{2}(x - \mu_k)^T \Sigma^{-1}(x - \mu_k) + \log \pi_k$$

Expanding the quadratic term:

$$(x - \mu_k)^T \Sigma^{-1}(x - \mu_k) = x^T \Sigma^{-1} x - 2x^T \Sigma^{-1} \mu_k + \mu_k^T \Sigma^{-1} \mu_k$$

Therefore:

$$\delta_k(x) = x^T \Sigma^{-1} \mu_k - \frac{1}{2}\mu_k^T \Sigma^{-1} \mu_k + \log \pi_k$$

This is a linear function of x — hence the name Linear Discriminant Analysis.

Ultimately, we obtain:

$$\delta_k(x) = x^T \Sigma^{-1} \mu_k - \frac{1}{2}\mu_k^T \Sigma^{-1} \mu_k + \log \pi_k$$

1.7 Question:

Show that the expected pooled sample covariance matrix

$$\hat{\Sigma} = \frac{1}{n - K} \sum_{k=1}^{K} \sum_{i:y_i=k} (X_i - \hat{\mu}_k)(X_i - \hat{\mu}_k)^T$$

in LDA is an unbiased estimator for the population covariance matrix Σ.

1.7.1 Answer:

Let's define:

(1) n_k: number of samples in class k

(2) $n = \sum_{k=1}^{K} n_k$: total number of samples

(3) $\hat{\mu}_k = \frac{1}{n_k} \sum_{i:y_i=k} X_i$: sample mean of class k

The sample covariance matrix for class k is:

$$S_k = \frac{1}{n_k - 1} \sum_{i:y_i=k} (X_i - \hat{\mu}_k)(X_i - \hat{\mu}_k)^T$$

Then thepooled sample covariance matrix is:

$$\hat{\Sigma} = \frac{1}{n - K} \sum_{k=1}^{K} (n_k - 1) S_k$$

Unbiased:

Each S_k is an unbiased estimator of Σ:

$$\mathbb{E}[S_k] = \Sigma$$

So:

$$\mathbb{E}[\hat{\Sigma}] = \frac{1}{n-K}\sum_{k=1}^{K}(n_k - 1)\mathbb{E}[S_k] = \frac{1}{n-K}\sum_{k=1}^{K}(n_k - 1)\Sigma$$

Factor out Σ:

$$\mathbb{E}[\hat{\Sigma}] = \Sigma \cdot \frac{1}{n-K}\sum_{k=1}^{K}(n_k - 1)$$

But:

$$\sum_{k=1}^{K}(n_k - 1) = n - K$$

So:

$$\mathbb{E}[\hat{\Sigma}] = \Sigma$$

Therefore, $\hat{\Sigma}$ is an unbiased estimator of Σ.

1.8 Question:

Derive the discrimination function $\delta_k(x)$ in QDA using Bayes' Theorem.

1.8.1 Answer:

Start from Bayes' Theorem:

$$P(Y = k \mid x) = \frac{P(x \mid Y = k)P(Y = k)}{P(x)}$$

We define the discriminant function:

$$\delta_k(x) = \log P(x \mid Y = k) + \log P(Y = k)$$

Assume:

$$x \mid Y = k \sim \mathcal{N}(\mu_k, \Sigma_k)$$

Then:

$$P(x \mid Y = k) = \frac{1}{(2\pi)^{d/2}|\Sigma_k|^{1/2}} \exp\left(-\frac{1}{2}(x - \mu_k)^T \Sigma_k^{-1}(x - \mu_k)\right)$$

Taking the logarithm yields:

$$\log P(x \mid Y = k) = -\frac{1}{2}(x - \mu_k)^T \Sigma_k^{-1}(x - \mu_k) - \frac{1}{2}\log|\Sigma_k| - \frac{d}{2}\log(2\pi)$$

Let $\pi_k = P(Y = k)$, then:

$$\delta_k(x) = -\frac{1}{2}(x - \mu_k)^T \Sigma_k^{-1}(x - \mu_k) - \frac{1}{2}\log|\Sigma_k| + \log \pi_k$$

We can ignore the constant $-\frac{d}{2}\log(2\pi)$ since it's the same for all classes.

Finally we get discriminant function for QDA:

$$\delta_k(x) = -\frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k) - \frac{1}{2}\log|\Sigma_k| + \log\pi_k$$

This is a quadratic function in x because of the term $(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k)$, which expands to include x^TAx terms.

1.9 Question:

Explain how confusion matrix can be extended to multiclass classification problem? How about ROC?

1.9.1 Answer:

Extending the Confusion Matrix

Extending the confusion matrix to a multiclass problem is straightforward. If you have K classes, the confusion matrix simply becomes a K x K matrix.

(a) Structure: The rows represent the Actual Class and the columns represent the Predicted Class.

(b) Interpretation:

(1) The diagonal elements of the matrix show the number of correct predictions for each class. For example, the cell at '(row=i, col=i)' contains the count of samples that were actually class 'i' and were correctly predicted as class 'i'.

(2) The off-diagonal elements show the errors. The cell at '(row=i, col=j)' where 'i is not equal to j' contains the count of samples that were actually class 'i' but were incorrectly predicted as class 'j'.

ROC Curve for Multi-Class Problems

Extending the ROC curve to multi-class settings, the core concepts of true positive rate (TPR) and false positive rate (FPR) are inherently binary. They require a "positive" class and a "negative" class, which do not exist in problems involving three or more categories.

The standard solution is to binarize the problem by decomposing a single multi-class task into multiple binary classification tasks (the OvR strategy).

For a classification problem with K categories, the OvR approach generates K independent ROC curves.

(1) Iterate over each class: For each class 'i' (from 1 to K), temporarily treat it as the positive class.

(2) Define the negative class: Combine all remaining K-1 classes into a single negative class.

(3) Generate a binary ROC curve: Using the model's probability scores for class 'i', generate a standard binary ROC curve.

Repeat this process for all K classes, ultimately generating K independent ROC curves.

1.10 Question:

Explain how the number of folds K affects the variance and bias of cross validation error.

1.10.1 Answer:

The number of folds, K, in cross-validation controls the trade-off between the bias and variance of the estimated test error. As K increases, the bias of the error estimate decreases, but its variance increases.

Bias of the Error Estimate

The bias in this context refers to how much the cross-validation error estimate differs from the true test error of a model trained on the entire dataset.

(1) Small K: When K is small, the size of the training set in each fold is significantly smaller than the full dataset. Models trained on less data are typically less performant and have a higher error rate.This means a small K leads to a high-bias error estimate.

(2) Large K: When K is large, the training set in each fold is nearly the entire dataset. The models trained in each fold are almost identical to the final model trained on all 'n' samples. As a result, the average error from these folds is a very accurate, nearly unbiased estimate of the true test error. This means a large K leads to a low-bias error estimate.

Variance of the Error Estimate

The variance of the estimate refers to how much the cross-validation error would fluctuate if we could repeat the procedure on different samples of data.

(1) Small K: When K is small, there is little to no overlap between the training sets of each fold.

Averaging the results from these less-correlated models produces a more stable estimate with low variance.

(2) Large K : When K is large, the training sets are almost identical to one another. This causes the models trained in each fold to be highly correlated. The average of many highly correlated quantities has high variance.

Therefore, the resulting error estimate is highly sensitive to the specific composition of the dataset and has high variance.

Specifically, K values like 5 or 10 are used to balance bias and variance effectively.