Assignment 3 - Machine Learning
Theoretical Questions

Mohammad Hossein Basouli

May 18, 2025

Question 1

Likelihood and probability both refer to the chance of seeing some outcome (outcomes means some data in this context) conditioned on some specific parameters that we consider for our probabilistic model, but with this difference, that, in probability, the data is variable but the parameters of the probabilistic model are fixed, whereas in likelihood, the data is fixed and the parameters of the probabilistic model are variables that we want to change in order to maximize the chance of seeing such an data (or outcome) for having a better modeling.

MLE for Logistic Regression

Notice: I have used AI to write down the detailed steps of showing how MLE is used for Logistic Regression instead of OLS, but I'm fully aware and understood of the details myself, thus there was no point not to use AI.

We want to model the probability that a binary outcome $y \in \{0, 1\}$ occurs, given input features $\mathbf{x} \in \mathbb{R}^d$, using logistic regression.

1. Model Definition

The logistic (sigmoid) function is defined as:

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

The hypothesis for logistic regression is:

$$P(y = 1 \mid \mathbf{x}; \boldsymbol{\theta}) = \sigma(\boldsymbol{\theta}^T \mathbf{x}) = \frac{1}{1 + e^{-\boldsymbol{\theta}^T \mathbf{x}}}$$

$$P(y = 0 \mid \mathbf{x}; \boldsymbol{\theta}) = 1 - \sigma(\boldsymbol{\theta}^T \mathbf{x})$$

2. Likelihood Function

Given a dataset $\{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^n$, where each $y^{(i)} \in \{0, 1\}$, the likelihood of the data is:

$$\mathcal{L}(\boldsymbol{\theta}) = \prod_{i=1}^{n} \left[\sigma(\boldsymbol{\theta}^T \mathbf{x}^{(i)})\right]^{y^{(i)}} \left[1 - \sigma(\boldsymbol{\theta}^T \mathbf{x}^{(i)})\right]^{1-y^{(i)}}$$

3. Log-Likelihood Function

Taking the logarithm of the likelihood function gives the log-likelihood:

$$\ell(\boldsymbol{\theta}) = \log \mathcal{L}(\boldsymbol{\theta}) = \sum_{i=1}^{n} \left[y^{(i)} \log \sigma(\boldsymbol{\theta}^T \mathbf{x}^{(i)}) + (1 - y^{(i)}) \log(1 - \sigma(\boldsymbol{\theta}^T \mathbf{x}^{(i)}))\right]$$

4. Gradient of the Log-Likelihood

To maximize the log-likelihood, we take its gradient with respect to $\boldsymbol{\theta}$:

$$\nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}) = \sum_{i=1}^{n} \left(y^{(i)} - \sigma(\boldsymbol{\theta}^T \mathbf{x}^{(i)})\right) \mathbf{x}^{(i)}$$

5. Optimization

We maximize the log-likelihood using optimization methods such as gradient ascent. The update rule for gradient ascent is:

$$\boldsymbol{\theta} := \boldsymbol{\theta} + \alpha \cdot \nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta})$$

where α is the learning rate.

Question 2

There are two main reasons that might make accuracy, an inappropriate metric for evaluating some classification problems:

1. It doesn't account for data imbalance.
2. It don't capture different errors that we might be curious about; e.g. we might care more about false negative more than false positives.

Alternative Metrics:

- Recall
  - Formula: Recall $= \frac{\text{True Positives}}{\text{True Positives}+\text{False Negatives}}$
  - What it captures: Measures the ability of the model to find the positive instaces.
  - Where it's good to use: When **missing a positive case is costly**, such as in **medical diagnosis**, **fraud detection**, or **security** applications.
- Precision
  - Formula: Precision $= \frac{\text{True Positives}}{\text{True Positives}+\text{False Positives}}$
  - What it captures: Measures how many of the predicted positives are actually positive.
  - Where it's good to use: When **false positives are costly**, such as in **spam detection**, **legal document classification**, or **alert systems**.
- F1-score
  - Formula: F1-score $= 2 \times \frac{\text{Precision}\times\text{Recall}}{\text{Precision}+\text{Recall}}$
  - What it captures: Harmonic mean of precision and recall. Balances both concerns, especially when you need a trade-off.
  - Where it's good to use: When you need a **balance between precision and recall**, especially in **imbalanced datasets** where both types of errors matter.

A Scenario Where The Precision is Significantly More Important Than Recall: Assume that we want to detect *spam emails*, and %99 of the emails are *non-spam* and only %1 is *spam*. Now if we build a model that predicts all of the emails as *spam*, we get a really high **Recall**, since we will identify true positive correctly, but here underlies a problem; we classify all of %99 of the *non-spam* emails as *spam* which is really bad. this kind of error is captured by **Precision**

Question 3

LDA as a Classifier: It tries to fit some parameters to the data for each class in order to understand the distribution of the data within that class.

LDA as a Dimesion Reduction Technique: It looks for an space to project the data into, such that it maximizes the difference between means of the two classes, normalized by sum of their variances.
Naive Bayes as a Special Case of LDA: Well, actually **Naive Bayes** is an special case of **Linear Discriminant Analysis** since it has all of the assumptions that **LDA** has, but in addition, it assumes that all of the features are independent of each other.

Question 4

Situations Where LDA Outperforms Logistic Regression:

- When the data actually satisfies the assumptions of LDA.
- When the data size is relatively small.
- When the number of features is low.

Situations Where Logistic Regression Outperforms LDA:

- When we are not sure whether the data satisfies the assumptions of LDA.
- When the data size is relatively large.
- When the number of features is large.

Why These Assumptions Critical to Their Performance: Because they have been developed and proven to work well, only under these assumptions :/ .

Question 5

1. **Resampling Techniques**:
    - **Oversampling**: Adds new instances of the minority class to balance it with the majority class. SMOTE (Synthetic Minority Over-sampling Technique) generates synthetic examples by interpolating between existing minority samples.
        - When to use:
            (a) When losing instances of the majority class is not desirable (e.g., due to limited data).

    (b) When the dataset is small and additional minority samples are needed for effective learning.
  - Why to use:
    (a) To help the model learn the patterns of the minority class more effectively.
    (b) To reduce overfitting that may result from simply duplicating examples.
- **Undersampling**: Removes some instances of the majority class to balance the class distribution.
  - When to use:
    (a) When the majority class has abundant data and can be reduced without losing important information.
    (b) When reducing dataset size can help in speeding up training and lowering computational cost.
  - Why to use:
    (a) To prevent the model from being biased toward the majority class.
    (b) To simplify the learning process and focus more on the minority class.

2. **Using Appropriate Metrics For Evaluation of The Model**: Instead of accuracy, which can be misleading, use metrics that better reflect performance on imbalanced datasets.
   - **Precision**, **Recall**, **F1-score**, **ROC-AUC**, and **PR-AUC** are commonly used.
   - When to use:
     (a) Always, when dealing with imbalanced data, especially when the minority class is the class of interest.
     (b) When accurate performance evaluation of both classes is important.
   - Why to use:
     (a) To avoid misleading conclusions that can occur with accuracy.
     (b) To ensure the model performs well on the minority class, which is often more critical (e.g., fraud detection, disease diagnosis).
3. **Cost-Sensitive Learning**: Assigns higher misclassification costs to the minority class during training.

- Adjusts the learning algorithm itself to pay more attention to minority class errors, often via class weights or custom loss functions.
- When to use:
  (a) When resampling is not feasible or desired.
  (b) When different types of misclassification errors have different real-world consequences.
- Why to use:
  (a) To make the model more sensitive to the minority class without altering the data distribution.
  (b) To incorporate the real-world cost of misclassifications into model training.

4. **Ensemble Learning**: Combines multiple models to improve performance, particularly on difficult cases such as those from the minority class.
   - Includes techniques like:
     - **Boosting**: Focuses on examples the model previously misclassified.
     - **Bagging with Balanced Subsets**: Trains multiple models on balanced datasets (e.g., Balanced Random Forest, EasyEnsemble).
   - When to use:
     (a) When single models underperform due to class imbalance or data complexity.
     (b) When robustness and improved generalization are needed.
   - Why to use:
     (a) To focus model training on hard-to-classify examples, including those from the minority class.
     (b) To reduce bias and variance, and enhance model performance across all classes.

Question 6

Assuming that we have K classes that we would want to classify, **OvA** comes up with a separate classifier for each of the K classes; each classifier could say whether an instance belongs to that class or it belongs to the other classes.

At the prediction time, each classifier yields a score or probability, and the instance is assigned to the class that has the highest probability. Whereas **OvO** build a separate classifier for each pair of the K classes; each classifier votes to the class that is more likely to contain the instance. At the prediction time, the class that has most votes, is the most likely to contain the instance.
When to Use OvA:

1. When the number of classes is relatively large.
2. When we would like to avoid overfitting.

When to Use OvO:

1. When we care more about accurary of the prediction more than it's speed.
2. When we have enough data in each of the classes.