CSE6740 CDA Homework 2

Name:
GTID:

Deadline: Sep 28 st 11:59 pm ET

1 Logistic Regression [10 points]

1.1 Log odds [5 points]

Logistic regression is named after the log-odds of success (the logit of the probability). Show that log odds of success for is a linear function of X where .

$$P[Y = 1 \mid X = x] = \frac{\exp(w_0 + w^T x)}{1 + \exp(w_0 + w^T x)}.$$

$$\text{To prove that } : \text{logit } P(Y = 1 \mid X = x) = w_0 + \mathbf{w}^\top \mathbf{x}$$

Solution:

Let
$$p(x) \;=\; P(Y = 1 \mid X = x) \;=\; \frac{\exp(w_0 + w^\top x)}{1 + \exp(w_0 + w^\top x)}.$$

Then
$$P(Y = 0 \mid X = x) = 1 - p(x) = \frac{1}{1 + \exp(w_0 + w^\top x)}.$$

The odds of success are

$$\begin{aligned} \frac{P(Y = 1 \mid X = x)}{P(Y = 0 \mid X = x)} &= \frac{\dfrac{\exp(w_0 + w^\top x)}{1 + \exp(w_0 + w^\top x)}}{\dfrac{1}{1 + \exp(w_0 + w^\top x)}} \\ &= \exp(w_0 + w^\top x). \end{aligned}$$

Taking natural logarithms yields the log-odds (logit):

$$\ln\left(\frac{P(Y = 1 \mid X = x)}{P(Y = 0 \mid X = x)}\right) = \ln\big(\exp(w_0 + w^\top x)\big) = w_0 + w^\top x.$$

Hence, the logistic regression model implies a *linear* log-odds:

$$\text{logit}\left(P(Y=1 \mid X=x)\right) = w_0 + w^\top x.$$

From the result above, $p(x) = \sigma(w_0 + w^\top x)$ where

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

Solution:
1. The log odds is linear in the original features. Hence, vanilla logistic regression can only learn linear decision boundaries and struggles when classes are nonlinearly separable in the data. [1 Mark]

2.
$$P[Y=1 \mid X=x] = \frac{\exp(w_0 + w^\top \phi(x))}{1+\exp(w_0 + w^\top \phi(x))}.$$

] [2 Mark]

1.2 Proof [5 points]

The logistic regression model for a single data point (x_i, y_i) is given by:

$$P(Y=1|X=x_i) = \frac{\exp(w_0 + \mathbf{w}^\top x_i)}{1+\exp(w_0 + \mathbf{w}^\top x_i)} = \sigma(z_i)$$

$$P(Y=0|X=x_i) = 1 - \sigma(z_i) = \frac{1}{1+\exp(w_0 + \mathbf{w}^\top x_i)} = \sigma(-z_i)$$

where $z_i = w_0 + \mathbf{w}^\top x_i$.

Derive the negative log-likelihood (NLL) loss function for a dataset and Show that this NLL is equivalent to the cross-entropy loss between true labels and predicted probabilities.

$$-\sum_{i=1}^{N}\Big[y_i \log \sigma(z_i) + (1-y_i)\log(1-\sigma(z_i))\Big] = -\sum_{i=1}^{N}\Big[y_i \log \hat{y}_i + (1-y_i)\log(1-\hat{y}_i)\Big], \quad \hat{y}_i = \sigma(z_i)$$

Solution:

The probability of a single data point (x_i, y_i) in a logistic regression model is given by:

$$P(Y = y_i | X = x_i) = [\sigma(z_i)]^{y_i} [1 - \sigma(z_i)]^{1-y_i}$$

where $z_i = w_0 + \mathbf{w}^\top x_i$.

The likelihood function for the entire dataset is the product of these probabilities:

$$L(\mathbf{w}, w_0) = \prod_{i=1}^{N} [\sigma(z_i)]^{y_i} [1 - \sigma(z_i)]^{1-y_i}$$

The log-likelihood is then:

$$\log L = \sum_{i=1}^{N} (y_i \log(\sigma(z_i)) + (1 - y_i) \log(1 - \sigma(z_i)))$$

The **negative log-likelihood (NLL)** loss function is the negative of the log-likelihood:

$$\text{NLL} = -\log L = -\sum_{i=1}^{N} (y_i \log(\sigma(z_i)) + (1 - y_i) \log(1 - \sigma(z_i)))$$

The binary cross-entropy loss for a single data point with true label y_i and predicted probability $\hat{y}_i = \sigma(z_i)$ is:

$$\text{CE}_i = -[y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]$$

The total cross-entropy loss for the dataset is the sum over all points:

$$\text{CE} = \sum_{i=1}^{N} \text{CE}_i = -\sum_{i=1}^{N} [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]$$

By substituting $\hat{y}_i = \sigma(z_i)$, we get:

$$\text{CE} = -\sum_{i=1}^{N} [y_i \log(\sigma(z_i)) + (1 - y_i) \log(1 - \sigma(z_i))]$$

Comparing this final expression with the derived NLL, it is clear that they are identical: NLL = CE. This proves their equivalence.

2 Feature Selection [20 points]

2.1 Information Theory [10 points]

For a pair of discrete random variables X and Y with the joint distribution $p(x, y)$, the **joint entropy** $H(X, Y)$ is defined as:

$$H(X,Y) = -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x,y)\log p(x,y),$$

which can also be expressed as

$$H(X,Y) = -\mathbb{E}[\log p(X,Y)].$$

Let X and Y take on values $x_1, x_2, \ldots, x_r$ and $y_1, y_2, \ldots, y_s$ respectively. Prove that

$$H(X,Y) \le H(X) + H(Y).$$

You may use the inequality

$$\ln x \le x - 1 \quad \text{for } x > 0.$$

Solution:

Solution:

$$H(X) + H(Y) = \sum_{i=1}^{r} p(x_i)\log\frac{1}{p(x_i)} + \sum_{j=1}^{s} p(y_j)\log\frac{1}{p(y_j)}$$

$$= \sum_{i=1}^{r}\left(\sum_{j=1}^{s} p(x_i,y_j)\right)\log\frac{1}{p(x_i)} + \sum_{j=1}^{s}\left(\sum_{i=1}^{r} p(x_i,y_j)\right)\log\frac{1}{p(y_j)}$$

$$= \sum_{i=1}^{r}\sum_{j=1}^{s} p(x_i,y_j)\log\frac{1}{p(x_i)} + \sum_{i=1}^{r}\sum_{j=1}^{s} p(x_i,y_j)\log\frac{1}{p(y_j)}$$

$$= \sum_{i=1}^{r}\sum_{j=1}^{s} p(x_i,y_j)\log\frac{1}{p(x_i)p(y_j)}$$

Now we will prove:

$$p(x_i,y_j)\log\frac{1}{p(x_i,y_j)} \le p(x_i,y_j)\log\frac{1}{p(x_i)p(y_j)} + p(x_i)\cdot p(y_j) - p(x_i,y_j)$$

Proof:

$$\begin{aligned}
p(x_i,y_j)\log\frac{1}{p(x_i,y_j)} &= p(x_i,y_j)\log\left(\frac{1}{p(x_i)p(y_j)}\cdot\frac{p(x_i)p(y_j)}{p(x_i,y_j)}\right)\\
&= p(x_i,y_j)\left(\log\frac{1}{p(x_i)p(y_j)} + \log\frac{p(x_i)p(y_j)}{p(x_i,y_j)}\right)\\
&\le p(x_i,y_j)\left(\log\frac{1}{p(x_i)p(y_j)} + \frac{p(x_i)p(y_j)}{p(x_i,y_j)} - 1\right)\\
&= p(x_i,y_j)\log\frac{1}{p(x_i)p(y_j)} + p(x_i)p(y_j) - p(x_i,y_j)
\end{aligned}$$

Taking the sum over i and j, we get:

$$\sum_{i=1}^{r}\sum_{j=1}^{s} p(x_i,y_j)\log\frac{1}{p(x_i,y_j)} \leq \sum_{i=1}^{r}\sum_{j=1}^{s} p(x_i,y_j)\log\frac{1}{p(x_i)p(y_j)} + \sum_{i=1}^{r}\sum_{j=1}^{s} p(x_i)\cdot p(y_j) - \sum_{i=1}^{r}\sum_{j=1}^{s} p(x_i,y_j)$$

Since $\sum_{i=1}^{r}\sum_{j=1}^{s} p(x_i)\cdot p(y_j) = \sum_{i=1}^{r}\sum_{j=1}^{s} p(x_i,y_j) = 1$, we obtain:

$$\sum_{i=1}^{r}\sum_{j=1}^{s} p(x_i,y_j)\log\frac{1}{p(x_i,y_j)} \leq \sum_{i=1}^{r}\sum_{j=1}^{s} p(x_i,y_j)\log\frac{1}{p(x_i)p(y_j)}$$

This proves that:

$$H(X,Y) \leq H(X) + H(Y)$$

2.2 Entropy [10 points]

Let X and Y be discrete random variables. Show that the mutual information can be expressed as

$$I(X;Y) = H(X) - H(X \mid Y) = H(Y) - H(Y \mid X).$$

Solution:

The mutual information is defined as

$$I(X;Y) = \sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x,y)\log\frac{p(x,y)}{p(x)p(y)}.$$

Expanding the logarithm:

$$I(X;Y) = \sum_{x,y} p(x,y)\log p(x,y) - \sum_{x,y} p(x,y)\log p(x) - \sum_{x,y} p(x,y)\log p(y).$$

Now observe:

$$\sum_{x,y} p(x,y)\log p(x) = \sum_{x} p(x)\log p(x),$$

and similarly

$$\sum_{x,y} p(x,y)\log p(y) = \sum_{y} p(y)\log p(y).$$

Thus:

$$I(X;Y) = \sum_{x,y} p(x,y)\log p(x,y) - \sum_{x} p(x)\log p(x) - \sum_{y} p(y)\log p(y).$$

Recognizing entropy terms:

$$H(X) = -\sum_{x} p(x)\log p(x), \quad H(Y) = -\sum_{y} p(y)\log p(y), \quad H(X,Y) = -\sum_{x,y} p(x,y)\log p(x,y),$$

we can rewrite:

$$I(X;Y) = H(X) + H(Y) - H(X,Y).$$

Using the chain rule of entropy:

$$H(X,Y) = H(X) + H(Y \mid X) = H(Y) + H(X \mid Y).$$

Therefore:

$$I(X;Y) = H(X) - H(X \mid Y) = H(Y) - H(Y \mid X).$$

3 Naive Bayes [10 points]

3.1 Conditionally Independent [10 points]

Suppose features $X = (X_1, X_2, \ldots, X_n)$ are conditionally independent Gaussians given $Y \in \{0, 1\}$, with parameters μ_{ik} and shared variance σ_i^2 for each feature i and class k. Prove that the Naive Bayes classifier in this case is a linear classifier in the feature space. Derive explicitly the form of the weight vector w and bias b.

$$\log \frac{P(Y=0 \mid X)}{P(Y=1 \mid X)} = w^\top X + b, \quad w_i = \frac{\mu_{i1} - \mu_{i0}}{\sigma_i^2}, \quad b = \log \frac{P(Y=1)}{P(Y=0)} - \sum_{i=1}^{n} \frac{\mu_{i1}^2 - \mu_{i0}^2}{2\sigma_i^2}.$$

Solution:

For a Naive Bayes classifier, we classify X according to:

$$\hat{Y} = \arg\max_{k \in \{0,1\}} P(Y = k \mid X) = \arg\max_{k} P(X \mid Y = k) P(Y = k).$$

Since the features are conditionally independent given Y, we have

$$P(X \mid Y = k) = \prod_{i=1}^{n} P(X_i \mid Y = k),$$

and for Gaussian likelihoods with shared variance σ_i^2:

$$P(X_i \mid Y = k) = \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left(-\frac{(X_i - \mu_{ik})^2}{2\sigma_i^2}\right).$$

The decision boundary is determined by:

$$\log \frac{P(Y=1 \mid X)}{P(Y=0 \mid X)} \gtrless 0$$

$$\log \frac{P(X \mid Y=1)P(Y=1)}{P(X \mid Y=0)P(Y=0)} \gtrless 0$$

$$\log \frac{P(Y=1)}{P(Y=0)} + \sum_{i=1}^{n} \log \frac{P(X_i \mid Y=1)}{P(X_i \mid Y=0)} \gtrless 0.$$

$$\log \frac{P(X_i \mid Y=1)}{P(X_i \mid Y=0)} = \log \frac{\exp\left(-\frac{(X_i-\mu_{i1})^2}{2\sigma_i^2}\right)}{\exp\left(-\frac{(X_i-\mu_{i0})^2}{2\sigma_i^2}\right)} = -\frac{(X_i-\mu_{i1})^2-(X_i-\mu_{i0})^2}{2\sigma_i^2}.$$

Expanding the squares:

$$(X_i-\mu_{i1})^2-(X_i-\mu_{i0})^2 = X_i^2-2X_i\mu_{i1}+\mu_{i1}^2-\left(X_i^2-2X_i\mu_{i0}+\mu_{i0}^2\right) = -2X_i(\mu_{i1}-\mu_{i0})+(\mu_{i1}^2-\mu_{i0}^2).$$

Thus:

$$\log \frac{P(X_i \mid Y=1)}{P(X_i \mid Y=0)} = \frac{X_i(\mu_{i1}-\mu_{i0})}{\sigma_i^2} - \frac{\mu_{i1}^2-\mu_{i0}^2}{2\sigma_i^2}.$$

Summing over i and adding the prior term gives the linear discriminant function:

$$\sum_{i=1}^{n} \frac{X_i(\mu_{i1}-\mu_{i0})}{\sigma_i^2} + \log \frac{P(Y=1)}{P(Y=0)} - \sum_{i=1}^{n} \frac{\mu_{i1}^2-\mu_{i0}^2}{2\sigma_i^2} \gtrless 0.$$

Define the weight vector w and bias b as:

$$w_i = \frac{\mu_{i1}-\mu_{i0}}{\sigma_i^2}, \quad b = \log \frac{P(Y=1)}{P(Y=0)} - \sum_{i=1}^{n} \frac{\mu_{i1}^2-\mu_{i0}^2}{2\sigma_i^2}.$$

Then the classifier can be written as a linear function of X:

$$w^T X + b >= 0.$$

4 K Nearest Neighbors [10 points]

4.1 kNN Algorithm [5 points]

Given a training dataset $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ where:

- $x_i \in \mathbb{R}^d$ are the feature vectors in d-dimensional space
- $y_i \in \{1, 2, \ldots, K\}$ are the class labels from K possible classes
- n is the total number of training samples

Objective: For a new test point $x \in \mathbb{R}^d$, predict its class label $\hat{y}$ using the k-nearest neighbor algorithm and discus it's time complexity.

Solution:
Given a test point x, the naive k-NN classifier works as follows:

1. Compute the distance from x to every point in the training set X_{train}. Typically, Euclidean distance is used:
$$d(x, x_i) = \sqrt{\sum_{j=1}^{d} (x_j - x_{i,j})^2}, \quad i = 1, \ldots, n$$
2. Sort the distances and select the k points with the smallest distances.
3. Take a majority vote among the labels of these k nearest neighbors.
4. Assign the label with the highest vote to x.

Time Complexity:

- Distance computation: $O(n \cdot d)$ for n training points in d dimensions.
- Sorting distances: $O(n \log n)$.
- Overall complexity per query: $O(n \cdot d + n \log n)$.

4.2 kNN Limitation [5 points]

The naive k-NN search can be slow for large datasets. Implement a data structure that can improve the query speed of k-NN. In your answer, explain the principle behind the structure, why it improves performance, and discuss its advantages and disadvantages, particularly in high-dimensional spaces.

Solution:
The naive k-NN algorithm requires computing distances from the query point to all training samples, which is inefficient for large datasets. To improve query speed, we can use a KD-Tree.
Implementation of KD-Tree (Python Example):

```
import numpy as np

class KDNode:
    def __init__(self, point, left=None, right=None):
        self.point = point      # Data point
        self.left = left        # Left subtree
        self.right = right      # Right subtree

def build_kdtree(points, depth=0):
```

```
    if len(points) == 0:
        return None

    k = points.shape[1]  # dimensionality
    axis = depth % k     # select splitting axis

    # sort points and choose median as pivot
    points = points[points[:, axis].argsort()]
    median = len(points) // 2

    return KDNode(
        point=points[median],
        left=build_kdtree(points[:median], depth + 1),
        right=build_kdtree(points[median + 1:], depth + 1)
    )
```

This code recursively builds a KD-Tree by splitting data along alternating dimensions, using the median point at each step.

Principle: A KD-Tree is a binary tree that partitions space along feature axes. Each internal node corresponds to a splitting hyperplane that divides the dataset into two halves. When performing a nearest neighbor search, we:

1. Traverse the tree to reach the leaf node that would contain the query point.
2. Backtrack and explore only those branches that could contain closer points than the current best.
3. This pruning avoids checking all data points, reducing query time.

Advantages:

- Construction takes $O(n \log n)$ time, which is efficient compared to storing all distances.
- Querying a single point can often be done in $O(\log n)$ on average in low dimensions, much faster than the naive $O(n)$.
- Saves computation by pruning large portions of the dataset.
- Well-suited for static datasets where queries are frequent.

Disadvantages in High Dimensions:

- In high-dimensional spaces, most points become nearly equidistant from the query point (curse of dimensionality).
- KD-Tree pruning becomes ineffective, and the query time approaches the naive $O(n)$ complexity.

- Tree balance and efficiency deteriorate as d increases, since splitting along dimensions does not meaningfully separate points.
- For $d > 30$, KD-Trees often perform no better than brute-force search.

Conclusion: KD-Trees are highly effective for accelerating k-NN searches in low-to-moderate dimensional data. However, in very high-dimensional settings, their advantages diminish, and approximate nearest neighbor methods or dimensionality reduction techniques are often preferred.

5 Bonus Q

5.1 MultiVariate Gaussian [10 points]

Let the target variable $Y \sim \text{Bernoulli}(\pi)$, so that

$$P(Y = 1) = \pi, \quad P(Y = 0) = 1 - \pi.$$

Let the features

$$\mathbf{X} = \langle X_1, X_2, \dots, X_d \rangle \in \mathbb{R}^d$$

have a shared covariance matrix Σ and class-specific mean vectors $\boldsymbol{\mu}_0, \boldsymbol{\mu}_1$:

$$\mathbf{X} \mid Y = k \sim \mathcal{N}(\boldsymbol{\mu}_k, \Sigma), \quad k \in \{0, 1\}.$$

The class-conditional density is:

$$P(\mathbf{X} \mid Y = k) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{X} - \boldsymbol{\mu}_k)^\top \Sigma^{-1} (\mathbf{X} - \boldsymbol{\mu}_k) \right).$$

Derive the posterior probability $P(Y = 1 \mid \mathbf{X})$ in the form:

$$P(Y = 1 \mid \mathbf{X}) = \frac{1}{1 + \exp(w_0 + \mathbf{w}^\top \mathbf{X})},$$

and explicitly show that

$$w_0 = \ln \frac{1 - \pi}{\pi} + \frac{1}{2}\boldsymbol{\mu}_1^\top \Sigma^{-1} \boldsymbol{\mu}_1 - \frac{1}{2}\boldsymbol{\mu}_0^\top \Sigma^{-1} \boldsymbol{\mu}_0, \quad \mathbf{w} = \Sigma^{-1}(\boldsymbol{\mu}_0 - \boldsymbol{\mu}_1).$$

Hint: Use the log-odds formula

$$\ln \frac{P(Y = 1 \mid \mathbf{X})}{P(Y = 0 \mid \mathbf{X})} = \ln \frac{P(Y = 1)P(\mathbf{X} \mid Y = 1)}{P(Y = 0)P(\mathbf{X} \mid Y = 0)}.$$

Solution:

Target: Derive $P(Y=1 \mid X)$ in vector/matrix form and show it is a logistic function.

We start with Bayes' theorem:

$$P(Y=1 \mid X) = \frac{P(Y=1)P(X \mid Y=1)}{P(Y=1)P(X \mid Y=1) + P(Y=0)P(X \mid Y=0)} = \frac{1}{1 + \frac{P(Y=0)P(X|Y=0)}{P(Y=1)P(X|Y=1)}} = \frac{1}{1 + \exp\Big(\ln \frac{P(Y=0)}{P(Y=1)}}$$

Focus on the term $\ln \frac{P(X|Y=0)}{P(X|Y=1)}$:

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

Then

$$\ln \frac{P(X \mid Y=0)}{P(X \mid Y=1)} = -\frac{1}{2}(X-\mu_0)^T\Sigma^{-1}(X-\mu_0) + \frac{1}{2}(X-\mu_1)^T\Sigma^{-1}(X-\mu_1) = (\mu_0-\mu_1)^T\Sigma^{-1}X + \frac{1}{2}\mu_1^T\Sigma^{-1}\mu_1 - \frac{1}{2}\mu_0^T\Sigma^{-1}\mu$$

Substitute back into the logistic form:

$$P(Y=1 \mid X) = \frac{1}{1 + \exp(w_0 + w^T X)}$$

where

$$w_0 = \ln \frac{1-\pi}{\pi} + \frac{1}{2}\mu_1^T\Sigma^{-1}\mu_1 - \frac{1}{2}\mu_0^T\Sigma^{-1}\mu_0, \quad w = \Sigma^{-1}(\mu_0 - \mu_1)$$

6 Programming

Please use this link to download all the required files. This homework contains only a ipynb, which you can make a copy and run on Google Colab.

Deliverables

For the programming part, please submit your `.ipynb` file to the programming autograder. Then, use `File` (top-left corner) → `Print` to generate and submit a PDF. The PDF is for future manual grading despite it is not used for this homework. For HW0, as long as you submit, you will receive 100%.

Expected files

- `HW2.pdf`
- `HW2.ipynb`
- `hw0.ipynb - Colab.pdf`