AI6102: Machine Learning Methodologies & Applications

Rohin Kumar Maheswaran (G2303513K)

rohinkum001@e.ntu.edu.sg

Assignment 1

Question 1: Consider a multi-class classification problem of C classes. Based on the parametric forms of the conditional probabilities of each class introduced on the 39th Page ("Extension to Multiple Classes") of the lecture notes of L4, derive the learning procedure of regularized logistic regression for multi-class classification problems.

Solution:

$$\text{For } c > 0: \quad P(y = c|x) = \frac{\exp(-w^{(c)^T} x)}{1 + \sum_{c=1}^{C-1} \exp(-w^{(c)T} x)} = \hat{y}_c$$

$$\text{For } c = 0: \quad P(y = 0|x) = \frac{1}{1 + \sum_{c=1}^{C-1} \exp(-w^{(c)T} x)} = \hat{y}_0$$

This allows us to express the binary cross-entropy (BCE) loss as:

$$L(w) = \sum_{c=1}^{C-1} y_c \ln \hat{y}_c + y_0 \ln(1 - \hat{y}_c)$$

if we have a dataset consisting of N training examples, where each example contains input features represented as vectors x_i in $\mathbb{R}^M$, our goal is to minimize the overall loss function.

$$J(w) = -\sum_{i=1}^{N} \left[\sum_{c=1}^{C-1} y_{ci} \ln \hat{y}_{ci} + y_{0i} \ln \hat{y}_{0i} \right] + \frac{\lambda}{2} \|\mathbf{w}\|_2^2$$

$$w_{t+1} = w_t - \alpha \frac{\partial J(w)}{\partial w}$$

$$w_{t+1} = w_t - \alpha \left(-\sum_{i=1}^{N} \frac{\partial L(w)}{\partial w} + \frac{\partial \frac{\lambda}{2} \|\mathbf{w}\|_2^2}{\partial w} \right)$$

To compute the gradient $\frac{\partial L(w)}{\partial w}$, we can employ the Chain Rule. In this process, we introduce a variable $z^{(i)}$ defined as the negative dot product of the weight vector $w^{(i)}$ and the input vector x_i, i.e., $z^{(i)} = -w^{(i)^T} x_i$." $\frac{\partial L(w)}{\partial w} = \frac{\partial L(w)}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial z} \times \frac{\partial z}{\partial w}$

Initially, we calculate the derivative of $L(w)$ concerning $z^{(i)}$ while excluding the regularization component.

$$\begin{aligned}
\frac{\partial L(w)}{\partial z^{(i)}} &= \frac{\partial \left[\sum_{c=1}^{C-1} y_c \ln \hat{y}_c + y_0 \ln \hat{y}_0\right]}{\partial z^{(i)}} \\
&= \frac{\partial}{\partial z^{(i)}} \left[\sum_{c=1}^{C-1} y_c \ln \hat{y}_c\right] + \frac{\partial \left(y_0 \ln \hat{y}_0\right)}{\partial z^{(i)}} \\
&= \sum_{c=1}^{C-1} y_c \frac{\partial \ln \hat{y}_c}{\partial z^{(i)}} + y_0 \frac{\partial \ln \hat{y}_0}{\partial z^{(i)}} \\
&= \sum_{c=1}^{C-1} \frac{y_c}{\hat{y}_c} \frac{\partial \hat{y}_c}{\partial z^{(i)}} + \frac{y_0}{\hat{y}_0} \frac{\partial \hat{y}_0}{\partial z^{(i)}}
\end{aligned}$$

$$\frac{\partial \hat{y}_c}{\partial z^{(i)}} = \frac{\partial}{\partial z^{(i)}} \left(\frac{\exp(-w^{(c)^T} x)}{1 + \sum_{c=1}^{C-1} \exp(-w^{(c)^T} x)} \right)$$

Before employing the Quotient Rule, we compute the individual partial derivatives for $\frac{g(x)}{h(x)}$.
Let $g(x) = \exp(z^{(c)}) = \exp(-w^{(c)^T} x_i)$.
Let $h(x) = 1 + \sum_{c=1}^{C-1} \exp(-w^{(c)^T} x_i)$.

$$g'(x) = \frac{\partial g(x)}{\partial z^{(i)}} = \frac{\partial \exp(-w^{(c)^T} x_i)}{\partial(-w^{(i)^T} x_i)} = \begin{cases} \exp(-w^{(i)^T} x_i) & \text{if } i = c \\ 0 & \text{if } i \neq c \end{cases}$$

$$h'(x) = \frac{\partial h(x)}{\partial z^{(i)}} = \frac{\partial \left[1 + \sum_{c=1}^{C-1} \exp(-w^{(c)^T} x_i)\right]}{\partial(-w^{(i)^T} x_i)} = \exp(-w^{(i)^T} x_i)$$

Case when $i = c$:

$$\begin{aligned}
\frac{\partial \hat{y}_c}{\partial z^{(i)}} &= \frac{\exp\left(-w^{(i)^T} x_i\right) \left[1 + \sum_{c=1}^{C-1} \exp\left(-w^{(c)^T} x_i\right)\right] - \exp\left(-w^{(i)^T} x_i\right) \exp\left(-w^{(c)^T} x_i\right)}{\left(1 + \sum_{c=1}^{C-1} \exp\left(-w^{(c)^T} x_i\right)\right)^2} \\
&= \frac{\exp\left(-w^{(i)^T} x_i\right) \left[1 + \sum_{c=1}^{C-1} \exp\left(-w^{(c)^T} x_i\right) - \exp\left(w^{(c)^T} x_i\right)\right]}{\left(1 + \sum_{c=1}^{C-1} \exp\left(-w^{(c)^T} x_i\right)\right)^2} \\
&= \frac{\exp\left(-w^{(i)^T} x_i\right)}{1 + \sum_{c=1}^{C-1} \exp\left(-w^{(c)^T} x_i\right)} \cdot \frac{1 + \sum_{c=1}^{C-1} \exp\left(-w^{(c)^T} x_i\right) - \exp\left(-w^{(c)^T} x_i\right)}{1 + \sum_{c=1}^{C-1} \exp\left(-w^{(c)^T} x_i\right)} \\
&= \frac{\exp\left(-w^{(i)^T} x_i\right)}{1 + \sum_{c=1}^{C-1} \exp\left(-w^{(c)^T} x_i\right)} \left(\frac{\exp\left(-w^{(c)^T} x_i\right)}{1 + \sum_{c=1}^{C-1} \exp\left(-w^{(c)^T} x_i\right)} \right) \\
&= \hat{y}_i \left(1 - \hat{y}_c\right) \\
&= \hat{y}_i \left(1 - \hat{y}_i\right)
\end{aligned}$$

Case when $i \neq c$:

$$\begin{aligned}
\frac{\partial \hat{y}_c}{\partial z^{(i)}} &= \frac{(0)\left(1+\sum_{c=1}^{C-1} \exp\left(-w^{(c)^T} x_i\right)\right) - \exp\left(-w^{(i)^T} x_i\right) \exp\left(-w^{(c)^T} x_i\right)}{\left(1+\sum_{c=1}^{C-1} \exp\left(-w^{(c)^T} x_i\right)\right)^2} \\
&= -\frac{\exp\left(-w^{(i)^T} x_i\right) \exp\left(-w^{(c)^T} x_i\right)}{\left(1+\sum_{c=1}^{C-1} \exp\left(-w^{(c)^T} x_i\right)\right)^2} \\
&= -\frac{\exp\left(-w^{(i)^T} x_i\right)}{1+\sum_{c=1}^{C-1} \exp\left(-w^{(c)^T} x_i\right)} \cdot \frac{\exp\left(-w^{(c)^T} x_i\right)}{1+\sum_{c=1}^{C-1} \exp\left(-w^{(c)^T} x_i\right)} \\
&= -\hat{y}_i \left(\hat{y}_c\right)
\end{aligned}$$

Therefore the derivatives of $\frac{\partial \hat{y}_c}{\partial z^{(i)}}$ are:

$$\frac{\partial \hat{y}_c}{\partial z^{(i)}} = \begin{cases} \hat{y}_i \left(1-\hat{y}_i\right) & \text{if } i = c \\ -\hat{y}_i \left(\hat{y}_c\right) & \text{if } i \neq c \end{cases}$$

Now we find the derivative for $\frac{\partial \hat{y}_0}{\partial z^{(i)}}$:

$$\begin{aligned}
\frac{\partial \hat{y}_0}{\partial z^{(i)}} &= \frac{\partial}{\partial z^{(i)}} \left(\frac{1}{1+\sum_{c=1}^{C-1} \exp\left(-w^{(c)^T} x\right)}\right) \\
&= (-1) \frac{\exp\left(-w^{(i)^T} x\right)}{\left(1+\sum_{c=1}^{C-1} \exp\left(-w^{(c)^T} x_i\right)\right)^2} \\
&= -\frac{1}{1+\sum_{c=1}^{C-1} \exp\left(-w^{(c)^T} x_i\right)} \cdot \frac{\exp\left(-w^{(i)^T} x_i\right)}{1+\sum_{c=1}^{C-1} \exp\left(-w^{(c)^T} x_i\right)} \\
&= -\hat{y}_0 \left(\hat{y}_i\right)
\end{aligned}$$

By Substituting the derivatives found :

$$\frac{\partial L(w)}{\partial z^{(i)}} = y_i - \hat{y}_i$$

The derivative of $z^{(i)}$ with respect to the weights $w^{(i)}$ is:

$$\frac{\partial z^{(i)}}{\partial w^{(i)}} = -x_i$$

Hence, the derivative of the loss function with respect to the weights $w^{(i)}$ is:

$$\frac{\partial L(w)}{\partial w^{(i)}} = \left(\hat{y}_i - y_i\right) x_i$$

Now, we find the derivative for the regularization term, $\frac{\lambda}{2}\|\mathbf{w}\|_2^2$:

$$\frac{\partial \frac{\lambda}{2}\|\mathbf{w}\|_2^2}{\partial w} = \lambda w$$

Substituting Equations into the Gradient Descent rule :

$$w_{t+1} = w_t + \alpha \left(\sum_{i=1}^{N} \left(\hat{y}_i - y_i\right) x_i - \lambda w\right)$$

Question 2

This is a hands-on exercise to use the SVC API of scikit-learn to train a SVM with the linear kernel and the rbf kernel, respectively, on a binary classification dataset.

Instructions:

1. Download the a9a dataset from the LIBSVM Dataset page. This is a preprocessed dataset of the Adult dataset in the UCI Irvine Machine Learning Repository, which consists of a training set and a test set.
2. Regarding the linear kernel, show 3-fold cross-validation results in terms of classification accuracy on the training set with different values of the parameter C in {0.01, 0.05, 0.1, 0.5, 1}, respectively, in the following table. Note that for all the other parameters, you can simply use the default values or specify the specific values you used in your submitted PDF file.
3. Regarding the rbf kernel, show 3-fold cross-validation results in terms of classification accuracy on the training set with different values of the parameter gamma (i.e., σ^2 in the lecture notes) in {0.01, 0.05, 0.1, 0.5, 1} and different values of the parameter C in {0.01, 0.05, 0.1, 0.5, 1}, respectively, in the following table. Note that for all the other parameters, you can simply use the default values or specify the specific values you used in your submitted PDF file.
4. Based on the results shown in Tables 1-2, determine the best kernel and the best parameter setting. Use the best kernel with the best parameter setting to train a SVM using the whole training set and make predictions on the test set to generate the following table:

Solution:

(2.1) - Loading the Dataset as a Sparse Matrix

Here are some Python libraries in use:

```
import os
import numpy as np
import sklearn
import matplotlib.pyplot as plt
from sklearn.datasets import load_svmlight_file
import seaborn as sns
from sklearn import svm
from sklearn.metrics import accuracy_score
from sklearn.metrics import accuracy_score, precision_score, recall_score
```

Load Dataset Into Sparse Matrix:

```
d_direc = '/Users/rohinkumar/Desktop/ML_Assignment_1/Dataset/'
train = os.path.join(d_direc, 'a9a.txt')
test = os.path.join(d_direc, 'a9a.t')
x_train, y_train = load_svmlight_file(train)
x_test, y_test = load_svmlight_file(test)
```

Characteristics of the dataset:

Dataset shapes after loading:

- Training set shape: (32,561 rows, 123 columns)
- Test set shape: (16,281 rows, 122 columns)

Ensuring that the training and test datasets have the same feature dimensions is crucial for the proper functioning of our model. When the training and test datasets have inconsistent feature dimensions, it means they contain different types or amounts of information for each data point. This mismatch can lead to problems with our model's performance and its ability to make accurate predictions.

To avoid these issues, we adjust the test dataset so that it has the same feature dimensions as the training dataset. This way, our model can work effectively and provide accurate results because it's trained on data with consistent features.

As a result, we proceed with the reshaping of the test data.

Shape of the Dataset after Reshaping:

- Training set (after reshape): (32,561 rows, 123 columns)
- Test set (after reshape): (16,281 rows, 123 columns)

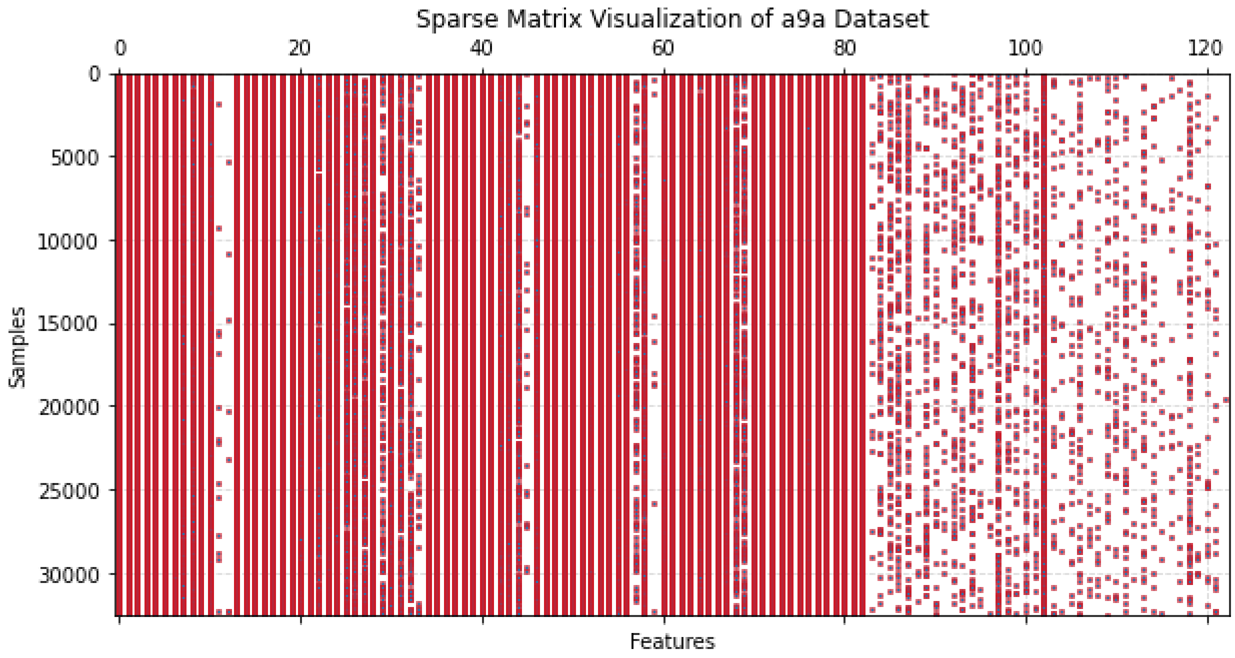

Figure 1: Visualisation of Sparse CSR Matrix of Training Dataset

(2.2) - Finding the Optimal Value of Parameter C with 3-Fold Cross-Validation using Linear Kernel

When looping through the given values $C = \{0.01, 0.05, 0.1, 0.5, 1\}$, we obtain the accuracies as follows:

Code Block:

```
C_vals = [0.01, 0.05, 0.1, 0.5, 1]
accuracies = {}

for i in C_vals:
    svm_classify = svm.SVC(kernel='linear', C=i)
    accuracy = cross_val_score(svm_classify, x_train, y_train, cv=3)
    mean_accuracy = accuracy.mean()
    accuracies[i] = mean_accuracy
for i, accuracy in accuracies.items():
    print(f'C_vals={i}: Accuracy = {accuracy:.5f}')
```

$C = 0.01$	$C = 0.05$	$C = 0.1$	$C = 0.5$	$C = 1$
0.84405	0.84610	0.84641	0.84706	**0.84724**

Table 1: Accuracy Scores for Different Values of C.

(2.3) - Performance on the Training Dataset with Various Combinations of C and γ Values

	$g = 0.01$	$g = 0.05$	$g = 0.1$	$g = 0.5$	$g = 1$
$C = 0.01$	0.75919	0.81991	0.81985	0.75919	0.75919
$C = 0.05$	0.83121	0.83575	0.83425	0.78916	0.75919
$C = 0.1$	0.83772	0.83965	0.83876	0.80612	0.76199
$C = 0.5$	0.84297	0.84577	0.84681	0.83216	0.78975
$C = 1$	0.84442	0.84675	**0.84742**	0.83661	0.79829

Table 2: Accuracy on the training set with different values of C and γ.

Code Block:

```
Accuracy on the training set with different values of C and gamma

param_grid = {
    'gamma': [0.01, 0.05, 0.1, 0.5, 1],
    'C': [0.01, 0.05, 0.1, 0.5, 1]
}

clf_rbf = svm.SVC(kernel='rbf')
grid_search = GridSearchCV(estimator=clf_rbf, param_grid=param_grid, cv=3,
              scoring='accuracy')
grid_search.fit(x_train, y_train)
results = grid_search.cv_results_

for gamma in [0.01, 0.05, 0.1, 0.5, 1]:
    print(f'g={gamma}', end=' ')
    for C in [0.01, 0.05, 0.1, 0.5, 1]:
        index = np.where((results['param_gamma'] == gamma) & (
                results['param_C'] == C))[0][0]
        mean_test_score = results['mean_test_score'][index]
        print(f'C={C}: {mean_test_score:.5f}', end=' ')
    print()
```

(2.4) - Model Evaluation with Optimal Hyperparameters

Considering that the radial basis function (RBF) kernel produces slightly better results compared to the linear kernel when hyperparameters are configured as $C = 1$ and $\gamma = 0.1$ during the model training phase, we proceed to use these specific hyperparameters to evaluate the model's performance on the test dataset.

Code Block:

```
clf_rbf = svm.SVC(kernel='rbf', gamma=0.1, C=1)
clf_rbf.fit(x_train, y_train)
y_pred = clf_rbf.predict(x_test)

accuracy = accuracy_score(y_test, y_pred)

# Print the accuracy
print("Accuracy:", accuracy)
```

Results from Testing :

Kernel	**Parameter Setting**
RBF	$C = 1$, $\gamma = 0.1$
Accuracy	
0.850316319636386	

Table 3: Results from using the $c = 1$ and $\gamma = 0.1$ in Test Dataset

Question 3 : The optimization problem of linear soft-margin SVMs can be re-formulated as an instance of empirical structural risk minimization (refer to Page 37 on L5 notes). Show how to reformulate it.

Solution:
Objective Function:
The objective function in optimization problem is given by:

$\hat{\theta} = \arg\min_\theta \sum_{i=1}^{N} \ell(f(x_i;\theta), y_i) + \lambda\Omega(\theta)$

In this context, $\ell(f(x_i;\theta), y_i)$ corresponds to the loss term, which quantifies the difference between the predicted values and the real labels.

Reformulation as Empirical SRM:

In empirical SRM, the objective is to minimize the empirical risk, which is the average loss over the training data. With N training examples, you can express the empirical risk as:

Empirical Risk $= \frac{1}{N}\sum_{i=1}^{N} \ell(f(\mathbf{x}_i;\boldsymbol{\theta}), y_i)$

Now, let's incorporate the regularization term into the empirical risk:

Empirical Risk $= \frac{1}{N}\sum_{i=1}^{N} \ell(f(\mathbf{x}_i;\boldsymbol{\theta}), y_i) + \frac{\lambda}{N}\Omega(\boldsymbol{\theta})$

The optimization problem for empirical SRM is to minimize the empirical risk, and you can formulate it as:

Minimize: $\frac{1}{N}\sum_{i=1}^{N} \ell(f(\mathbf{x}_i;\boldsymbol{\theta}), y_i) + \frac{\lambda}{N}\Omega(\boldsymbol{\theta})$

Question 4 : Using the kernel trick introduced in L5 to extend the regularized linear regression model (L3) to solve nonlinear regression problems. Derive a closed-form solution (i.e., to derive a kernelized version of the closed-form solution on Page 50 of L3).

Solution:

The regularized linear regression formula is as follows:

$$\hat{\mathbf{w}} = \arg\min_{w} \frac{1}{2}\sum_{i=1}^{N}\left(y_i - \mathbf{w}\cdot x_i\right)^2 + \frac{\lambda}{2}\|\mathbf{w}\|_2^2$$

We know that for any solution $\hat{\mathbf{w}}$, it can be written as a linear combination of features x_i :

$$\hat{w} = \sum_{i=1}^{N}\mathbf{w}\cdot x_i = \sum_{i=1}^{N}\beta_i x_i$$

Therefore using Kernel trick,

$$\begin{aligned}\hat{\beta} &= \arg\min_{\beta} \frac{1}{2}\sum_{i=1}^{N}\left(y_i - \sum_{j=1}^{N}\beta_j K\left(x_i, x_j\right)\right)^2 + \frac{\lambda}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\beta_i\beta_j K\left(x_i, x_j\right)\\ &= \arg\min_{\beta} \frac{1}{2}\left(\beta^T K^T K\beta - 2\beta^T K^T y + y^T y\right) + \frac{\lambda}{2}\beta^T K\beta\end{aligned}$$

Finding the closed form solution by changing the gradient equal to zero:

$$\begin{aligned}\frac{\partial\frac{1}{2}\left(\beta^T K^T K\beta - 2\beta^T K^T y + y^T y\right) + \frac{\lambda}{2}\beta^T K\beta}{\partial\beta} &= 0\\ \frac{\partial\frac{1}{2}\left(\beta^T K^T K\beta - 2\beta^T K^T y + y^T y\right)}{\partial\beta} + \frac{\partial\frac{\lambda}{2}\beta^T K\beta}{\partial\beta} &= 0\\ K^T K\beta - K^T y + \lambda K^T I\beta &= 0\\ K^T((K + \lambda I)\beta - y) &= 0\\ \hat{\beta} &= (K + \lambda I)^{-1}y\end{aligned}$$