Fundamentals of Machine Learning (Spring 2025)

Homework #2 (100 Pts, Due date: April 16)

Student ID 2021311716

Name 김서영

Files you need to submit

`ML_HW2_YourName_StudentID.zip`: All code in the directory and your report.

- `ML_HW2_YourName_StudentID.pdf`: Your report converted into a PDF file.
- `Code(folder)`

NOTE 1: Please write your code only in the 'EDIT HERE' section.

NOTE 2: You may need to install the NumPy and Matplotlib libraries.

NOTE 3: Please read the comments in the code carefully.

1. **[Implementation]** Implement the logistic regression and SoftMax classifier in your code. **Once you have completed your implementation, run the checker code ('1_LogisticRegression_Checker.py' or '1_SoftmaxClassifier_Checker.py') to validate if your code is executed correctly.**

(a) [10 pts] Implement the functions in 'models/LogisticRegression.py'. ('**forward**', '**compute_grad**', '**_sigmoid**', and '**eval**' respectively). Given a mini-batch data (X, Y), the error function for a mini-batch is defined as follows:

$$\boldsymbol{E(w) = -\frac{1}{|\mathcal{B}|}\sum_{(x_i,y_i)\in\mathcal{B}} y_i * \log(\hat{y}_i) + (1-y_i) * \log(1-\hat{y}_i),}$$

$$\textbf{where } \hat{y}_i = \text{sigmoid}(w^T x_i), |\mathcal{B}| \textbf{ is the number of the mini} - \textbf{batch samples.}$$

Fill in your code here. Please submit your code to i-campus.

Answer:

```
def forward:
# ========== EDIT HERE ==========
    z = np.dot(x, self.W)
    logits = self._sigmoid(z)
```

```
        loss = -np.mean(y * np.log(logits + eps) + (1 – y) * np.log(1 – logits + eps)
# ==================================


def compute_grad:
# ========== EDIT HERE ==========
        grad_weight = np.dot(x.T, (logit – y)) / num_data
# ==================================


def _sigmoid:
# ========== EDIT HERE ==========
        sigmoid = 1 / (1 + np.exp(-x))
# ==================================


def eval:
# ========== EDIT HERE ==========
        z = np.dot(x, self.W)
        prob = self._sigmoid(z)
        pred = (prob >= threshold).astype(int).flatten( )
# ==================================
```

(b) [10 pts] Implement functions in 'models/SoftmaxClassifier.py'. ('**forward**', '**compute_grad**', '**_softmax**', and '**eval**' respectively). Given a mini-batch data (X, Y), the error function for a mini-batch is defined as follows:

$$E(\boldsymbol{w}) = -\frac{1}{|\mathcal{B}|}\sum_{(\boldsymbol{x_i},\boldsymbol{y_i})\in\mathcal{B}} \boldsymbol{y_i} * \log(\hat{\boldsymbol{y}}_i),$$

$$\text{where } \hat{\boldsymbol{y}}_i = \text{softmax}\left(\boldsymbol{w}^T\boldsymbol{x_i}\right), |\mathcal{B}| \text{ is the number of the mini} - \text{batch samples}.$$

Fill in your code here. Please submit your code to i-campus.

Answer:

```
def forward:
# ========== EDIT HERE ==========
    logits = np.dot(x, self.W)
    prob = self._softmax(logits)
    y_onehot = np.zeros_like(prob)
    y_onehot = [np.arange(num_data), y] = 1
    eps = 1e – 10
    softmax_loss = -np.sum(y_onehot * np.log(prob + eps)) / num_data
# ================================


def compute_grad:
# ========== EDIT HERE ==========
    yy = np.zeros(num_classes)
    yy[y[j]] = 1
    pp = prob[j, :]
    grad_weight += np.outer(xx, (pp – yy))
# ================================


def _softmax:
# ========== EDIT HERE ==========
    x_shifted = x – np.max(x, axis – 1, keepdims = True)
    exp_x = np.exp(x_shifted)
    softmax = exp_x / np.sum(exp_x, axis = 1, keepdims = True)
# ================================


def eval:
# ========== EDIT HERE ==========
    logits = np.dot(x, self.W)
```

```
        prob = self._softmax(logits)

        pred = np.argmax(prob, axis = 1)

# ================================
```

Instructions: For problems 2, 3, and 4, we have prepared two classification datasets: Banknote Authentication and Iris. The **Banknote authentication dataset is used for binary classification**. It consists of **four features, such as variance, skewness, kurtosis, and entropy,** to predict authentication for Banknotes. On the other hand, the **Iris dataset is used for multi-class classification**, representing three Iris-types. It consists of **four features, such as sepal length and width, and petal length and width.** For detailed information on each dataset, please refer to the link below.

(Default hyperparameter settings for (b) and (c): Epoch = 100, Batch_size = 256, learning_rate = 0.1)

Dataset	# training data	# test data	# classes	Details
Banknote	1,029	343	2	link
Iris	125	25	3	link

2. **[Normalization] Z-score normalization** transforms data by subtracting the mean and dividing by the standard deviation:

$$Z = \frac{(X - \mu)}{\sigma}$$

Min-max normalization is a technique to rescale data by transforming it into a specified range, typically between 0 and 1, by subtracting the minimum value and dividing by the range of the data.

$$x_{norm} = \frac{x - X_{min}}{X_{max} - X_{min}}$$

(a) **[5 pts]** Implement the **Z-score Normalization and Min-Max Normalization** in **utils.py (z_score, min_max function)**

Fill in your code here. Please submit your code to i-campus.

Answer:

```
def z_score:
# ========== EDIT HERE ==========
    x_col = (x_col – x_col.mean( )) / x_col.std( )
# ================================


def min_max:
# ========== EDIT HERE ==========
    x_col = (x_col – x_col.min( )) / (x_col.max( ) – x_col.min( ))
# ================================
```

(b) **[15 pts]** For the Banknote and Iris datasets, compare the performance before and after normalization in default settings. Analyze why the performance differs as follows.

Run 2_LogisticRegression_main.py. and 2_SoftmaxClassifier_main.py

Banknote	Train	Test
Unnormalized	0.9825	0.9737
Z-Score Norm	0.9640	0.9708
Min-Max Norm	0.8327	0.8363

Iris	Train	Test
Unnormalized	0.7339	0.6667
Z-Score Norm	0.9032	0.8333
Min-Max Norm	0.7258	0.7917

Answer:

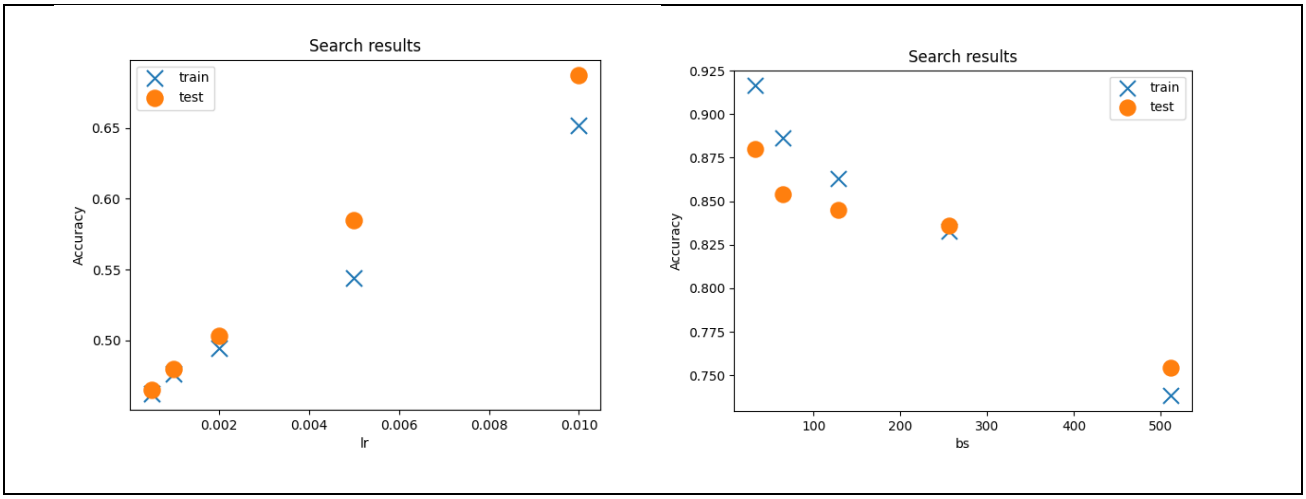
Banknote 데이터셋의 경우, train data와 test data모두 정규화하지 않은 상태에서 가장 높은 정확도를 보였다. 이는 데이터의 각 feature가 이미 적절한 스케일로 분포되어 있어 정규화 없이도 좋은 학습 결과를 보였음을 의미한다. Z-score 정규화를 적용한 경우에도 큰 성능 저하 없이, 높은 정확도를 보였다. 이는 Z-score가 스케일의 균형을 유지하면서도 데이터의 분포를 크게 왜곡하지 않았기 때문이다. 반면, Min-Max 정규화를 적용한 경우에는 성능이 크게 저하되었다. 이는 각 feature를 0~1 사이로 조정하는 과정에서 원래의 분포가 훼손되어 모델 학습에 방해되었다고 해석할 수 있다.

Iris 데이터셋의 경우, 정규화를 적용하지 않았을 때, 가장 낮은 정확도를 보였다. 이는 feature 간 스케일 차이로 모델이 class의 경계를 제대로 학습하지 못했기 때문이다. 반면, Z-score 정규화를 적용한 경우, train data와 test data 모두 가장 높은 정확도를 높였다. 이는 각 feature의 분포를 조정하면서 class 간 거리를 효과적으로 반영할 수 있었기 때문이다. Min-Max 정규화를 적용한 경우에는 test data의 정확도를 향상시켰지만, Z-score만큼 안정적인 성능을 보이지는 못했다. 이는 Min-Max 정규화가 이상치에 민감하며, class 간 상대적 거리를 왜곡할 가능성이 있기 때문이라고 해석할 수 있다.

3. **[Hyperparameter]** Draw the plot by adjusting the **learning rate and the batch size**. The x-axis represents the values of the hyperparameters being searched, and the y-axis represents the accuracy score. Try various values (i.e., 0.0005, 0.001, 0.002, 0.005, 0.01) for the learning rate, and (i.e., 32, 64, 128, 256, 512) for batch size. Analyze each of the two results and determine which parameter has the most significant impact on performance

(a) [10 pts] For the **Banknote dataset**, draw the plot by adjusting the parameters. (Fix the other hyperparameters as default settings.) **Run 2_LogisticRegression_main.py.**

Answer:



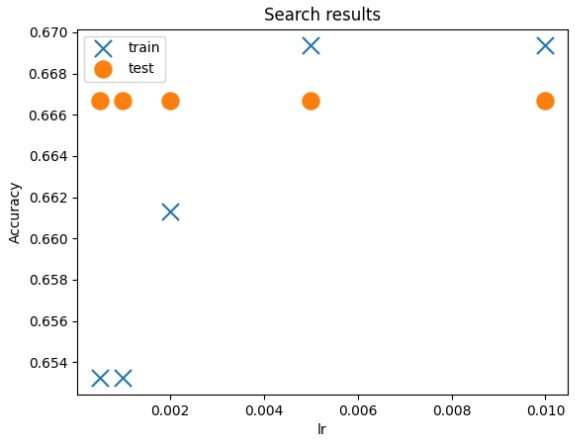
Learning rate를 0.0005, 0.001, 0.002, 0.005, 0.01로 실험해본 결과, train data와 test data 모두 learning rate가 커질수록 정확도가 증가하는 경향을 보였다. 특히, learning rate가 0.01일 때, 가장 높은 정확도를 보였다. 이는, learning rate가 너무 작으면 모델의 학습 속도가 느려서 충분히 수렴하지 못하기 때문이라고 해석할 수 있다.

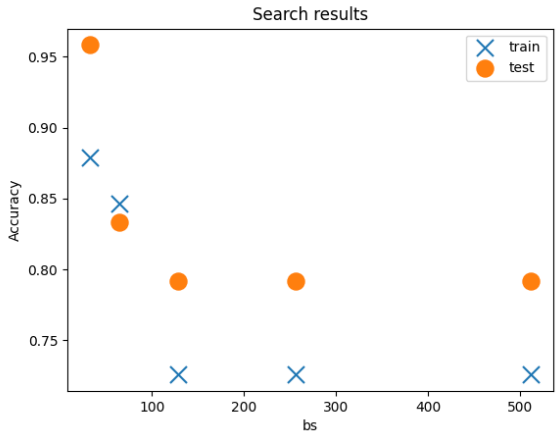
Batch size를 32, 64, 128, 256, 512로 실험해본 결과, train data와 test data 모두 batch size가 커질수록 정확도가 감소하는 경향을 보였다. 특히, batch size가 512일 때, 가장 낮은 정확도를 보였고, batch size가 32일 때, 가장 높은 정확도를 보였다. 이는 batch size가 작을수록 모델이 더 자주 업데이트되며 성능 향상으로 이어지기 때문이라고 해석할 수 있다.

Learning rate를 변화시키며 정확도를 확인했을 때, 가장 높은 정확도와 가장 낮은 정확도의 차이는 약 0.2정도이다. Batch size를 변화시키며 정확도를 확인했을 때, 가장 높은 정확도와 가장 낮은 정확도의 차이는 약 0.125정도이다. 따라서, learning rate가 batch size보다 모델의 성능에 더 큰 영향을 미친다고 볼 수 있다.

(b) [10 pts] For the **Iris dataset**, draw the plot by adjusting the parameters. (Fix the other hyperparameters as default settings.) **Run 2_SoftmaxClassifier_main.py**

Answer:





Learning rate를 0.0005, 0.001, 0.002, 0.005, 0.01로 실험해본 결과, train data는 learning rate가 커질수록 정확도가 증가하는 경향을 보였지만, test data는 learning rate와 관계없이 정확도가 일정하게 유지되는 경향을 보였다. 이는, learning rate가 커지더라도 overfitting이나 급격한 변동 없이 안정적으로 학습이 진행되기 때문이라고 해석할 수 있다.

Batch size를 32, 64, 128, 256, 512로 실험해본 결과, train data와 test data 모두 batch size가 커질수

록 정확도가 감소하는 경향을 보였다. 특히, batch size가 128 이상일 때, train data와 test data 모두에서 정확도가 낮아졌다. 이는, batch size가 작을수록 모델이 더 자주 업데이트되며 성능 향상으로 이어지기 때문이라고 해석할 수 있다.

따라서, batch size가 learning rate보다 모델의 성능에 더 큰 영향을 미친다고 볼 수 있다.

4. **[40 pts] (Kaggle challenge)** Perform a Binary Classification Task using the given **imbalanced "Back Order Prediction" Dataset**. You can improve performance by modifying the data features, and you can further enhance it through a more effective model. You need to sign up for Kaggle, the site where you can find various competitions. Observe the site below and follow the instructions. You should submit your answer on the site.

Kaggle Site: https://www.kaggle.com/t/23ea2f302e5245ae950cd183b9a6c8fa

Note that you are free to use additional libraries. Feel free to edit any part of the code while working on this task.

You can run "baseline.ipynb" or rewrite the code and report the results.

Please explain your method and provide the final accuracy.

Answer: