Problems3and4

October 23, 2024

- This code can be found at: https://github.com/trentbellinger/Math-156/tree/main/Homework%203

0.1 Problem 3

[1]:
```
import numpy as np
import pandas as pd

# Sigmoid function
def compute_sigmoid(z):
    return 1 / (1 + np.exp(-z))

# Loss function
def compute_loss(X, y, w):
    return -np.sum((1 - y) * np.log(compute_sigmoid(X @ w)) + y * np.log(1 -␣
↪compute_sigmoid(X @ w)))

# Mini-Batch SGD
def get_weights(X, y, batch_size, fixed_learning_rate, max_iterations):
    num_samples, num_features = X.shape
    weights = np.random.randn(num_features)  # initialize weights with Gaussian␣
↪distribution

    # Training with mini-batch SGD
    for iteration in range(max_iterations):

        for start in range(0, num_samples, batch_size):
            end = min(start + batch_size, num_samples)
            X_batch = X[start:end]
            y_batch = y[start:end]

            gradient = (compute_sigmoid(X_batch @ weights) - y_batch) @ X_batch

            weights -= fixed_learning_rate * gradient

    return weights

def predict_new(X, weights):
    return (compute_sigmoid(X @ weights) >= 0.5).astype(int)
```

0.2 Problem 4

0.2.1 Problem 4(a)

[2]:
```
from ucimlrepo import fetch_ucirepo
from sklearn.preprocessing import StandardScaler

breast_cancer_wisconsin_diagnostic = fetch_ucirepo(id=17)
X = StandardScaler().fit_transform(breast_cancer_wisconsin_diagnostic.data.
 ↪features)
y = np.array(breast_cancer_wisconsin_diagnostic.data.targets["Diagnosis"])
y[y == "B"] = "0"
y[y == "M"] = "1"
y = y.astype(int)
```

0.2.2 Problem 4(b)

[3]:
```
from sklearn.model_selection import train_test_split

X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.3,␣
 ↪random_state=42, stratify=y)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=2/3,␣
 ↪random_state=42, stratify=y_temp)
```

0.2.3 Problem 4(c)

[4]:
```
print("Number of benign in training:", np.sum(y_train == 0))
print("Number of malignant in training:", np.sum(y_train == 1))
print("Number of benign in validation:", np.sum(y_val == 0))
print("Number of malignant in validation:", np.sum(y_val == 1))
```

```
Number of benign in training: 250
Number of malignant in training: 148
Number of benign in validation: 36
Number of malignant in validation: 21
```

0.2.4 Problem 4(d)

[5]:
```
weights = get_weights(X_train, y_train, fixed_learning_rate=0.0001,␣
 ↪batch_size=10, max_iterations=100000)
```

0.2.5 Problem 4(e)

```
[6]: from sklearn.metrics import accuracy_score, precision_score, recall_score,␣
     ↪f1_score

     test_preds = predict_new(X_test, weights)

     print("Accuracy:", accuracy_score(y_test, test_preds))
     print("Precision:", precision_score(y_test, test_preds))
     print("Recall:", recall_score(y_test, test_preds))
     print("F1-Score:", f1_score(y_test, test_preds))
```

```
Accuracy: 0.956140350877193
Precision: 0.975
Recall: 0.9069767441860465
F1-Score: 0.9397590361445783
```

0.2.6 Problem 4(f)

The model has an accuracy of 0.956, so it correctly classifies 95.6% of the breast cancer. The recall is relatively low in comparison to the other measures, which indicates that the model is not as good at correctly predicting malignant breast cancer. This is not good, because the model should not miss potentially deadly predictions of malignant cancer. Overall, the model performs very well, even with the mini-batch gradient descent, which was used to reduce computational intensiveness.