

Lab Gradient Descent: Logistic Regression nhận diện chữ số viết tay (MNIST - Dataset)

Phạm Tiến Lâm, Đặng Văn Báu

Activity 1. Warmup

- Tạo một colab mới với tên “ho_va_ten_MSV_multiclass”
- Import thư viện numpy, matplotlib

Activity 2. Load data and preprocessing


- Load data MNIST

```
1 from keras.datasets import mnist
2
3 (X_train, y_train), (X_test, y_test) = mnist.load_data()
```

- Hiển thị dữ liệu

```
1 print(y_train[0])
2 plt.imshow(X_train[0], cmap = 'gray')
```

5
<matplotlib.image.AxesImage at 0x7fc06f17b8b0>



- Chuẩn hóa Input:

```
1 X_train_scaled = np.array([x.ravel()/255. for x in X_train])
2
3 X_test_scaled = np.array([x.ravel()/255. for x in X_test])
```

1 X_train_scaled.shape, X_test_scaled.shape

((60000, 784), (10000, 784))

- Chuẩn hóa output:

```
1 from sklearn.preprocessing import OneHotEncoder
2 enc = OneHotEncoder(handle_unknown='ignore')
3
4 y_train_onehot = enc.fit_transform(y_train.reshape(-1, 1)).toarray()
5 y_test_onehot = enc.fit_transform(y_test.reshape(-1, 1)).toarray()
```

```
1 print(y_train[0])
2 print(y_train_onehot[0])

5
[0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
```

Activity 3. Xây dựng và huấn luyện mô hình

- Mô hình dự báo và hàm loss:

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}}$$

```
1 def predict(X, w):
2     h = np.dot(X, w)
3     softmax = np.exp(h)
4     y_pred = softmax / np.sum(softmax, axis=1, keepdims=True)
5     return y_pred
6
7
8 def loss(X, w, y):
9     y_pred = predict(X, w)
10    return -np.sum(y * np.log(y_pred)) / y.shape[0]
```

- Hàm gradient:

```
def grad(X, w, y):
    y_pred = predict(X, w)
    delta = y_pred - y
    return np.dot(X.T, delta) / X.shape[0]
```

- Gradient Descent:

```
def train(X, w, y, lr = 0.01, n_epoch = 1000):  
    for i in range(n_epoch):  
        w = w - lr*grad(X, w, y)  
    print(f'Loss cross entropy: {loss(X, w, y)}')  
    return w
```

Bài tập.

1. Sử dụng các trọng số tìm được trong mục activity 3, dự đoán kết quả trong tập test, so sánh giá trị dự báo với giá trị thực tế thông qua các đại lượng ACCURACY, RECALL, PRECISION.
2. Viết lại python code cho mô hình hồi quy tuyến tính dưới dạng một class có tên là LogisticRegression.
3. Viết lại python code cho mô hình hồi quy tuyến tính bao gồm cả hệ số bias (intercept) và sử dụng để xây dựng mô hình dự báo kết quả. Đánh giá mô hình dựa trên tập dữ liệu test set.
4. Sử dụng thư viện sklearn sử dụng confusion_matrix để visualize kết quả trên tập test.

Như hình sau:

