Exercício prático 07_ Regressão Logística

October 29, 2025

1 Exercício de Programação: Regressão Logística e Softmax com MNIST

Neste exercício, você aplicará os conceitos de Regressão Logística para classificação binária e Regressão Softmax para classificação multiclasse. Usaremos o famoso dataset MNIST, que consiste em imagens de dígitos manuscritos.

Objetivos: 1. Treinar um classificador binário para identificar se um dígito é '5' ou 'não-5'. 2. Treinar um classificador multiclasse para identificar os dígitos de 0 a 9. 3. Avaliar a performance de ambos os modelos.

1.1 1. Preparação do Ambiente e Carregamento dos Dados

Primeiro, vamos importar as bibliotecas necessárias e carregar o dataset MNIST.

Também vamos pré-processar os dados: - Dividir em conjuntos de treino e teste. - Escalar os valores dos pixels para melhorar a performance do gradiente descendente. - Remodelar as imagens de 28x28 para vetores de 784 dimensões, que é o formato esperado por um classificador como `LogisticRegression`.

[24]:
```
import numpy as np
from sklearn.datasets import load_digits
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import StandardScaler

# Carregar o dataset MNIST
digits = load_digits()
X, y = digits.data, digits.target

# Dividir em conjuntos de treino e teste
X_train, X_test = X[:1600], X[1600:]
y_train, y_test = y[:1600], y[1600:]

# Escalar os pixels
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```
# Remodelar as imagens para vetores 1D (28*28 = 784)
X_train_flat = X_train.reshape(X_train.shape[0], -1)
X_test_flat = X_test.reshape(X_test.shape[0], -1)

print(f"Formato dos dados de treino: {X_train_flat.shape}")
print(f"Formato dos dados de teste: {X_test_flat.shape}")
```

```
Formato dos dados de treino: (1600, 64)
Formato dos dados de teste: (197, 64)
```

1.2 2. Treinando um Classificador Binário (5 ou não-5)

Agora, vamos criar um classificador para uma tarefa binária: detectar se um dígito é o número 5 ou não. Para isso, precisamos ajustar nossos rótulos (`y_train` e `y_test`).

[25]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.datasets import fetch_openml
from sklearn.model_selection import train_test_split

# Carregar o dataset MNIST
mnist = fetch_openml('mnist_784', version=1, as_frame=False)
X, y = mnist["data"], mnist["target"].astype(int)

# Dividir em treino e teste
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)

# Criar os rótulos binários: True para 5, False para outros
y_train_5 = (y_train == 5)
y_test_5 = (y_test == 5)

# Inicializar e treinar o modelo de Regressão Logística
log_reg = LogisticRegression(max_iter=10)
log_reg.fit(X_train, y_train_5)

print(" Modelo de Regressão Logística treinado!")
```

```
 Modelo de Regressão Logística treinado!
/home/tailan/Documentos/6 Semestre/Aprendizado de Maquina/resolucao dos
colabs/venv/lib/python3.12/site-packages/sklearn/linear_model/_logistic.py:473:
ConvergenceWarning: lbfgs failed to converge after 10 iteration(s) (status=1):
STOP: TOTAL NO. OF ITERATIONS REACHED LIMIT

Increase the number of iterations to improve the convergence (max_iter=10).
You might also want to scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
```

```
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

1.2.1 Avaliação do Classificador Binário

Com o modelo treinado, vamos avaliar sua acurácia no conjunto de teste.

[26]:
```
from sklearn.metrics import accuracy_score

y_pred_5 = log_reg.predict(X_test)

# Ou usar o método .score() diretamente
acc_score = log_reg.score(X_test, y_test_5)
print(f"Acurácia usando .score(): {acc_score:.4f}")
```

```
Acurácia usando .score(): 0.9629
```

1.3 3. Treinando um Classificador Multiclasse (Regressão Softmax)

O `LogisticRegression` do Scikit-Learn automaticamente lida com a classificação multiclasse usando a estratégia "um-contra-o-resto" (OvR) por padrão. Para usar a Regressão Softmax (também chamada de Regressão Logística Multinomial), podemos definir o argumento `multi_class='multinomial'`.

Vamos treinar um novo modelo para classificar todos os 10 dígitos (0 a 9).

[27]:
```
# SEU CÓDIGO AQUI: Treine o modelo softmax usando o conjunto de treinamento com␣
↪todas as classes (X_train_flat, y_train).

# Inicializar o modelo de Regressão Softmax
# Usamos max_iter=1000 para garantir a convergência.

# ----------------
from sklearn.linear_model import LogisticRegression

# Inicializar o modelo de Regressão Softmax
softmax_reg = LogisticRegression(multi_class='multinomial', solver='lbfgs',␣
↪max_iter=10)

# Treinar o modelo com todas as classes
softmax_reg.fit(X_train, y_train)

print("Modelo de Regressão Softmax treinado!")
```

```
/home/tailan/Documentos/6 Semestre/Aprendizado de Maquina/resolucao dos
colabs/venv/lib/python3.12/site-packages/sklearn/linear_model/_logistic.py:1272:
FutureWarning: 'multi_class' was deprecated in version 1.5 and will be removed
in 1.8. From then on, it will always use 'multinomial'. Leave it to its default
```

```
value to avoid this warning.
  warnings.warn(

Modelo de Regressão Softmax treinado!

/home/tailan/Documentos/6 Semestre/Aprendizado de Maquina/resolucao dos
colabs/venv/lib/python3.12/site-packages/sklearn/linear_model/_logistic.py:473:
ConvergenceWarning: lbfgs failed to converge after 10 iteration(s) (status=1):
STOP: TOTAL NO. OF ITERATIONS REACHED LIMIT

Increase the number of iterations to improve the convergence (max_iter=10).
You might also want to scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

1.3.1 Avaliação do Classificador Multiclasse

Por fim, avaliamos o modelo multiclasse no conjunto de teste.

[28]:
```python
# Calcular e imprimir a acurácia no conjunto de teste
accuracy = softmax_reg.score(X_test, y_test)
print(f"Acurácia do modelo multiclasse: {accuracy:.4f}")
```

```
Acurácia do modelo multiclasse: 0.8826
```

1.4 Conclusão

Parabéns! Você implementou e avaliou com sucesso dois tipos de classificadores lineares: 1. Um classificador de Regressão Logística para uma tarefa binária. 2. Um classificador de Regressão Softmax para uma tarefa multiclasse.

Você deve notar que, embora simples, esses modelos já alcançam uma acurácia razoavelmente alta para o reconhecimento de dígitos.