PerceptronAdeline

April 2, 2023

1 ANN

1.1 Adeline class yang dibuat tanpa library

[1]:
```
class CustomAdaline(object):

    def __init__(self, n_iterations=100, random_state=1, learning_rate=0.01):
        self.n_iterations = n_iterations
        self.random_state = random_state
        self.learning_rate = learning_rate

    '''
    Batch Gradient Descent

    1. Weights are updated considering all training examples.
    2. Learning of weights can continue for multiple iterations
    3. Learning rate needs to be defined
    '''
    def fit(self, X, y):
        rgen = np.random.RandomState(self.random_state)
        self.coef_ = rgen.normal(loc=0.0, scale=0.01, size=1 + X.shape[1])
        for _ in range(self.n_iterations):
              activation_function_output = self.activation_function(self.
↪net_input(X))
              errors = y - activation_function_output
              self.coef_[1:] = self.coef_[1:] + self.learning_rate*X.T.
↪dot(errors)
              self.coef_[0] = self.coef_[0] + self.learning_rate*errors.sum()

    '''
    Net Input is sum of weighted input signals
    '''
    def net_input(self, X):
            weighted_sum = np.dot(X, self.coef_[1:]) + self.coef_[0]
            return weighted_sum

    '''
```

```
    Activation function is fed the net input. As the activation function is
    an identity function, the output from activation function is same as the
    input to the function.
    '''
    def activation_function(self, X):
            return X

    '''
    Prediction is made on the basis of output of activation function
    '''
    def predict(self, X):
        return np.where(self.activation_function(self.net_input(X)) >= 0.0, 1, 0)

    '''
    Model score is calculated based on comparison of
    expected value and predicted value
    '''
    def score(self, X, y):
        misclassified_data_count = 0
        for xi, target in zip(X, y):
            output = self.predict(xi)
            if(target != output):
                misclassified_data_count += 1
        total_data_count = len(X)
        self.score_ = (total_data_count - misclassified_data_count)/
↪total_data_count
        return self.score_
```

1.2 Perceptron class yang dibuat tanpa library

[2]:
```
import numpy as np
#
# Perceptron implementation
#
class CustomPerceptron(object):

    def __init__(self, n_iterations=100, random_state=1, learning_rate=0.01):
        self.n_iterations = n_iterations
        self.random_state = random_state
        self.learning_rate = learning_rate

    '''
    Stochastic Gradient Descent

    1. Weights are updated based on each training examples.
    2. Learning of weights can continue for multiple iterations
    3. Learning rate needs to be defined
```

```
    '''
    def fit(self, X, y):
        rgen = np.random.RandomState(self.random_state)
        self.coef_ = rgen.normal(loc=0.0, scale=0.01, size=1 + X.shape[1])
        self.errors_ = []
        for _ in range(self.n_iterations):
            errors = 0
            for xi, expected_value in zip(X, y):
                predicted_value = self.predict(xi)
                self.coef_[1:] = self.coef_[1:] + self.learning_rate *␣
↪(expected_value - predicted_value) * xi
                self.coef_[0] = self.coef_[0] + self.learning_rate *␣
↪(expected_value - predicted_value) * 1
                update = self.learning_rate * (expected_value - predicted_value)
                errors += int(update != 0.0)
            self.errors_.append(errors)
    '''
    Net Input is sum of weighted input signals
    '''
    def net_input(self, X):
            weighted_sum = np.dot(X, self.coef_[1:]) + self.coef_[0]
            return weighted_sum

    '''
    Activation function is fed the net input and the unit step function
    is executed to determine the output.
    '''
    def activation_function(self, X):
            weighted_sum = self.net_input(X)
            return np.where(weighted_sum >= 0.0, 1, 0)

    '''
    Prediction is made on the basis of output of activation function
    '''
    def predict(self, X):
        return self.activation_function(X)

    '''
    Model score is calculated based on comparison of
    expected value and predicted value
    '''
    def score(self, X, y):
        misclassified_data_count = 0
        for xi, target in zip(X, y):
            output = self.predict(xi)
            if(target != output):
                misclassified_data_count += 1
```

```
        total_data_count = len(X)
        self.score_ = (total_data_count - misclassified_data_count)/
 ↪total_data_count
        return self.score_
```

1.3 Implementasi code untuk Perceptron dan Adeline

Pada implementasinya, kelas custom model perceptron dan adeline akan dipanggil

[3]:
```
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
import numpy as np

data = load_breast_cancer()
X = data.data
y = data.target
#
# Create training and test split
#
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
 ↪random_state=42, stratify=y)
#
# Instantiate CustomPerceptron
#
adaline = CustomAdaline(n_iterations = 10)
#
# Fit the model
#
adaline.fit(X_train, y_train)
#
# Score the model
#
adaline.score(X_test, y_test),adaline.score(X_train, y_train)
```

[3]:
```
(0.6257309941520468, 0.628140703517588)
```

[4]:
```
import numpy as np
from sklearn import datasets
from sklearn.model_selection import train_test_split
#
# Load the data set
#
bc = datasets.load_breast_cancer()
X = bc.data
y = bc.target
#
# Create training and test split
```

```
#
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
 ↪random_state=42, stratify=y)
#
# Instantiate CustomPerceptron
#
prcptrn = CustomPerceptron(n_iterations=10)
#
# Fit the model
#
prcptrn.fit(X_train, y_train)
#
# Score the model
#
prcptrn.score(X_test, y_test), prcptrn.score(X_train, y_train)
```

[4]: (0.8888888888888888, 0.9120603015075377)

1.4 SKLEARN Implementation

1.4.1 Implementasi code untuk Perceptron dan Adeline dengan menggunakan Library

Perceptron Import fungsi Percepron yang bereda di Library sklearn,kemudian deklarasikan variable model perceptron lalu melakukan process training.

[5]:
```
from sklearn.linear_model import Perceptron
p = Perceptron()
p.fit(X_train, y_train)
```

[5]: Perceptron()

[6]:
```
from sklearn.metrics import accuracy_score
predictions_train = p.predict(X_train)
predictions_test = p.predict(X_test)
train_score = accuracy_score(predictions_train, y_train)
print("score on train data: ", train_score)
test_score = accuracy_score(predictions_test, y_test)
print("score on test data: ", test_score)
```

```
score on train data:  0.9271356783919598
score on test data:  0.8830409356725146
```

MLP Implementation

[7]:
```
from sklearn.neural_network import MLPClassifier
clf = MLPClassifier()
clf.fit(X_train, y_train)
```

```
/Users/istiqomah/opt/anaconda3/lib/python3.8/site-
packages/sklearn/neural_network/_multilayer_perceptron.py:686:
```

```
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and
the optimization hasn't converged yet.
  warnings.warn(
```

[7]:
```
MLPClassifier()
```

[8]:
```
from sklearn.metrics import accuracy_score
predictions_train = clf.predict(X_train)
predictions_test = clf.predict(X_test)
train_score = accuracy_score(predictions_train, y_train)
print("score on train data: ", train_score)
test_score = accuracy_score(predictions_test, y_test)
print("score on test data: ", test_score)
```

```
score on train data:  0.9396984924623115
score on test data:  0.9064327485380117
```

[9]:
```
from sklearn.neural_network import MLPClassifier
clf = MLPClassifier(hidden_layer_sizes=(150,100))
clf.fit(X_train, y_train)
```

[9]:
```
MLPClassifier(hidden_layer_sizes=(150, 100))
```

[10]:
```
from sklearn.metrics import accuracy_score
predictions_train = clf.predict(X_train)
predictions_test = clf.predict(X_test)
train_score = accuracy_score(predictions_train, y_train)
print("score on train data: ", train_score)
test_score = accuracy_score(predictions_test, y_test)
print("score on test data: ", test_score)
```

```
score on train data:  0.9296482412060302
score on test data:  0.9122807017543859
```

[ ]:
```

```