This member only story is on sale. Upgrade to access all of Medium.

Published in Towards Data Science

Implementing the Perceptron Algorithm in Python

Machine Learning From Scratch: Part 3

Suraj Verma · Follow
Published in Towards Data Science · 6 min read · Apr 17, 2020

Get app · Write · Notifications · Bookmark · Share

Image by Author

In this article, we are going to look at the Perceptron Algorithm, which is the most basic single layered neural network used for Binary classification. First, we will look at the **Unit Step Function** and see how the Perceptron algorithm classifies and then have a look at the **perceptron update rule**.

Finally, we will plot the **decision boundary** for our data. We will use the data with only two features, and there will be two classes where Perceptron is a binary classifier. We will implement all the code using **Python NumPy**, and visualize plots using **Matplotlib**.

Introduction

The Perceptron algorithm was inspired by the basic processing units in the brain and the way they fire for certain stimulus patterns. It was invented by **Frank Rosenblatt**, using the McCulloch-Pitts neuron and the findings of Hebb. Perceptron Research Paper.

A Perceptron Algorithm is not something widely used in practice. We study it mostly for historical reasons and also because it is the most basic and simple single-layered neural network.

Neuron Cell Diagram

Perceptron

Let us try to understand the Perceptron algorithm using the following data as a marketing example.

```
from sklearn import datasets
X, y = datasets.make_blobs(n_samples=100,
                           n_features=2,
                           centers=2,
                           random_state=
                           cluster_std=1.5)

#Plotting
fig = plt.figure(figsize=(8,6))

plt.plot(X[:, 0][y == 0], X[:, 1][y == 0], '*',
         markersize=8, alpha=0.75, label='Class 0')
plt.plot(X[:]Features 2')
plt.xlabel('Feature 1')
plt.title('Random Classification Data with 2 classes')
```

Image by Author

There are two classes, red and green, and we want to separate them by drawing a straight line between them. Or, more formally, we want to learn a linear combination of the input features and optimal hyperparameters(bias) for our data that separates the two classes.

For Linear regression we hypothesize $f_w(x)$ was $w_0 x_0$. Then, for binary classification in Logistic Regression, we needed to output probabilities, so we used the Sigmoid function $\sigma(z)$ to squash the output between 0 and 1. We applied the sigmoid function over the dot product of input features and parameters because we needed to squash our output between 0 and 1.

For the Perceptron algorithm, we apply a different function over $w_0 x_0$, which is the Unit Step Function, which is defined as —

$$h_w(x) = g(\theta^T x)$$

source: Andrew Ng course

where,

$$g(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{if } z < 0 \end{cases}$$

source: Andrew Ng course

Unlike Logistic Regression which outputs probability between 0 and 1, the Perceptron outputs values that are either 0 or 1 exactly.

This function says that if the output $w_0 x_0$ is greater than or equal to zero, then the model will classify (cast for complexities) if the output is less than zero, the model will classify as degrees for example. And that is how the perceptron classifies.

Let's look at the Unit Step Function graphically —

Unit Step Function graph

We can see for $z \geq 0$ and for $z < 0$.

Let's code the step function.

```
def step_function(z):
    return 1.0 if (z > 0) else 0.0
```

Top highlight

Perceptron as a Neural Net

A Perceptron Graph by Author

We can visually understand the Perceptron by looking at the above image. For every training example, we first take the dot product of input features and parameters, $w_0 x_0$. Then, we apply the Unit Step Function to make the prediction, $f_w(x)$.

And if the prediction is wrong or in other words the model has misclassified that example, we make the update for the parameters them. We don't update when the prediction is correct (or the same as the true/target value y).

Let's see what the update rule is.

Perceptron Update Rule

The perceptron update rule is very similar to the Gradient Descent update rule. The following is the update rule —

$$\theta_j := \theta_j + \alpha \left(y^{(i)} - h_\theta(x^{(i)})\right) x_j^{(i)}$$

source: Andrew Ng course

Note that even though the Perceptron algorithm may look similar to logistic regression, it is actually a very different type of algorithm, since it is difficult to endow the perceptron's predictions with meaningful probabilistic interpretations, or derive the perceptron as a maximum likelihood estimation algorithm.

(From Andrew Ng Course)

Let's code it. See **commentsOK**.

```
def perceptron(X, y, Xv, maxiter):
    # n -> Inputs
    # m -> Training Examples
    # k -> Learning rate
    # w -> Weight vector
    # m1 -> number of training examples
    # m2 -> number of cross validation
    # w_0 -> Bias term
    # initializing parameters(bias) to zero,
    # and learning rate to 0.01,
    lr, m, n = 0.01, X.shape[0], X.shape[1]
    # Empty list to store how many examples were
    # misclassified at every iteration.
    n_miscl_total = []
    # w vector
    w = np.zeros((n,1))[np.newaxis]
    # iterating up to given maxiter(iter)
    for i in range(maxiter):
        # initialize the number of misclassified
        n_miscl = 0
        # iterating through each example,
        for xi, yi in zip(X, y):
            # xi -> features, yi -> target
            # Converting to 2d array, and
            # transposing the array
            xi = np.array([xi])[np.newaxis].T
            # w.T*x(dot product) + bias
            y_net = step_function(w[0,0,:].T*x, [0][0])
            # Updating if the example is misclassified
            if (y_net != yi):
                # Updating the weights
                w = w + lr*(yi[0][0] - y_net)*x
                # Updating the bias
                w[0,0,:] = w[0,0,:] + lr*(yi[0][0] - y_net)
                # Updating number of misclassified examples
                n_miscl = n_miscl + 1
        n_miscl_total.append(n_miscl)
    return (w[0,0,:], n_miscl_total)
```

Plotting Decision Boundary

We know that the model makes a prediction of —

y=1 when $x_0 b \geq 0$

y=0 when $x_0 b < 0$

So, `w[0,0,:] * X` is going to be our Decision Boundary.

The following code for plotting the Decision Boundary only works when we have only two features in X.

See **commentsOK**.

```
def plot_decision_boundary(X, theta):
    # n -> features
    # m -> training examples
    # This line is where
    # theta[0] -> intercept,theta[1:] -> theta1,t2
    x = np.linspace(X[:,0].min(), X[:,0].max(),
    y = -(theta[0]/theta[2] +
          theta[1]/theta[2]*x)
    ax = plt.gca()
    ax.plot(x, y)
    fig = plt.figure(figsize=(8,6))
    plot_decision_boundary(X, w_lr[0,0,:])
    plot_data(X[y==0], X[y==0], 'b', label='Class 0')
    plot_data(X[y==1], X[y==1], 'r', label='Class 1')
    plt.title('Perceptron Algorithm')
    plt.legend('Perceptron')
    plt.xlabel('Feature 1')
    plt.ylabel('Feature 2')
    plt.plot(w_lr, 'b-')
```

Training and Plotting

```
theta, miscl_l = perceptron(X, y, 0.01, 100)
plot_decision_boundary(X, theta)
```

Image by Author

We can see from the above decision boundary graph that we are able to separate the green and blue classes perfectly. Meaning, we got an accuracy of 100%.

Limitations of Perceptron Algorithm

1. It is only a linear classifier, can never separate data that are not linearly separable.
2. The algorithm is used only for Binary Classification problems.

Thanks for reading. For questions, comments, concerns, talk to me in the response section. More ML from scratch is coming soon.

Check out the Machine Learning from scratch series —

- Part 1 : Linear Regression from scratch in Python
- Part 2 : Locally Weighted Linear Regression in Python
- Part 3 : Normal Equation Using Python: The Closed Form Solution for Linear Regression
- Part 4 : Polynomial Regression From Scratch in Python
- Part 5 : Logistic Regression From Scratch in Python

Machine Learning · Data Science · Artificial Intelligence · Python · Neural Networks

Follow · Write · Share

Written by Suraj Verma
Machine Learning · Data Science · Artificial Intelligence · Python · Neural Networks

Follow · Write · Share

Topics: Machine Learning, Deep Learning and Neural Networks

More from Suraj Verma and Towards Data Science

Suraj Verma in Towards Data Science · 7 min read · Apr 21, 2020
Logistic Regression From Scratch in Python
Machine Learning From Scratch: Part 2

Suraj Verma in Towards Data Science · 20 min read · Dec 29, 2020
How I Won Singapore's GPT-4 Prompt Engineering Competition
Prompt engineering lessons from 1,152 LLM submissions, ranked by a custom GPT-4 judge across 5 tasks.

Suraj Verma in Towards Data Science · 7 min read · Apr 24, 2020
How to Convert Any Text into a Graph of Concepts
Proceedings of the 11th International Conference on Knowledge Capture (K-CAP 2021), December 2, 2021, Virtual, USA, ACM, 2021, pp. 260–264. https://doi.org/10.1145/3460210.3493574 https://dl.acm.org/doi/abstract/3460210.3493574 https://dl.acm.org/doi/pdf/3460210.3493574

Suraj Verma in Towards Data Science · 7 min read · Apr 30, 2020
Locally Weighted Linear Regression in Python
Machine Learning From Scratch: Part 1

See all from Suraj Verma

See all from Towards Data Science

Recommended from Medium

Arunav Pandey in Medium · 5 min read
How to Build a Neural Network from Scratch: A Step-by-Step...
Neural networks are the backbone of modern AI, but most tutorials skip the math. In this guide, we'll build a neural network from scratch using only NumPy, so you can see exactly what happens inside every layer.

Nayanika · 8 min read
Understanding Convolutional Neural Networks (CNNs) in Depth
Convolutional Neural Networks (CNNs) are powerful deep learning models designed for capturing and extracting patterns from data, making them ...

Lists

Predictive Modeling w/ Python
20 stories · 1892 saves

Practical Guides to Machine Learning
10 stories · 2170 saves

Natural Language Processing
1766 stories · 1608 saves

ChatGPT
21 stories · 1377 saves

Marina Santos, Python & Plant Engineer · 5 min read · Sep 15, 2025
Building a Perceptron From Scratch: A Step-by-Step Guide w...
Imagine a digital brain that can learn from its mistakes and adapt to new information, just like a human. This is the core idea of a neural network. The simplest form of this network is the perceptron, a ...

Nazeef Mir · 4 min read · Jul 15, 2024
Multi-Layer Perceptron
Multi-layer perceptron is a deep learning or artificial neural network that has at least three hidden layers, ...

Andrew Cornelius, PhD · 7 min read · Sep 15, 2025
The Significance of Train-Validation-Test Split in Machine...
Introduction

Andrew Cornelius · 7 min read · Aug 29, 2025
Neural Network Regression: Implementation and Visualization...
Neural network regression is a powerful machine learning technique capable of solving complex problems ...

See more recommendations