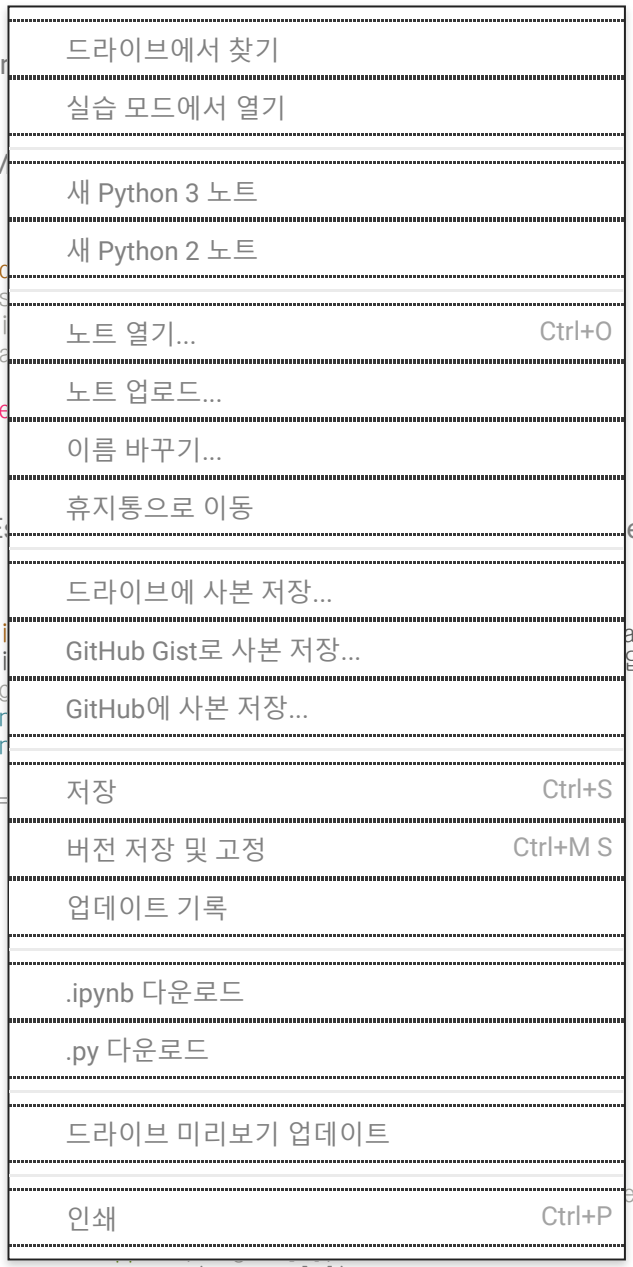
Perceptrons - Training



Note for

- M

```
def prec
    bias
    acti
    if a

    else
```

- E ent descent

```
def trai                                                    aset이 아니다
    #wei                                                    없음
    weig
    prin
    prin

    vb =
    vw0
    vw1

    for
```

```
                                                            e * error * row[i] #에러가 0이면 값 유지 #ro
            vw0.append(weights[1])
            vw1.append(weights[2])

        print('epoch={}, error={}'.format(epoch, sum_error))
    return weights,vb,vw0,vw1
```

```
#training set for AND gate
dataset = [[0,0,0],
           [0,1,0],
           [1,0,0],
           [1,1,1]]
```

- Hyperparameters

```
l_rate = 0.1
n_epoch = 5
```

```
weights, vb,vw0,vw1 = train_weights(dataset, l_rate, n_epoch)
```

```
print(weights)#에러빼기=미분
```

```
pred = predict([0.51,1], weights)
print(pred)
```

```
pred = predict([1,0.0000000000000001], weights)
print(pred)
```

```
import matplotlib.pyplot as plt

plt.plot(pred)
```

```
plt.plot(vb)
```

```
plt.plot(vw0)
```



```
plt.plot(vw1)
```



```
plt.plot(prediction)
```


- Why ?

partial derivative with respect to m

#derivative=미분

$$\frac{\partial J(m,b)}{\partial m} = \frac{1}{n}\sum_{i=1}^{n} -2x^{(i)}(y_i - (mx^{(i)} + b))$$
$$= \frac{2}{n}\sum_{i=1}^{n} x^{(i)}((mx^{(i)} + b) - y^{(i)})$$
$$= \frac{2}{n}\sum_{i=1}^{n} x^{(i)}(\hat{y}^{(i)} - y^{(i)})$$

partial derivative with respect to b

$$\frac{\partial J(m,b)}{\partial b} = \frac{1}{n}\sum_{i=1}^{n} -2(y^{(i)} - (mx^{(i)} + b))$$
$$= \frac{-2}{n}\sum_{i=1}^{n}(y^{(i)} - (mx^{(i)} + b))$$
$$= \frac{2}{n}\sum_{i=1}^{n}(\hat{y}^{(i)} - y^{(i)})$$

m은 x가 붙어있음 바이오스 = 맨아래부분 차이점 x붙어있음

Partial derivatives : https://www.mathsisfun.com/calculus/derivatives-partial.html

- References

https://machinelearningmastery.com/implement-perceptron-algorithm-scratch-python/