sgd

April 14, 2019

```
In [1]: from matplotlib import pyplot as plt
        import math
        import numpy as np
```

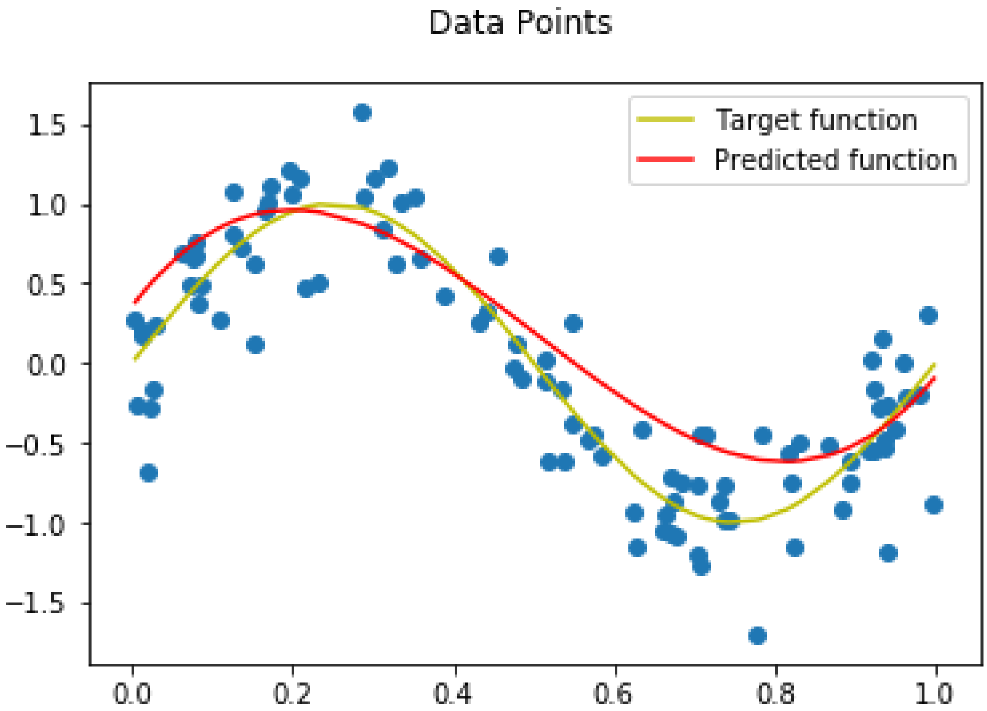
```
In [2]: # variables
        alpha = 0.25
        dataPointAmount = 100
        thetaAmount = 4
```

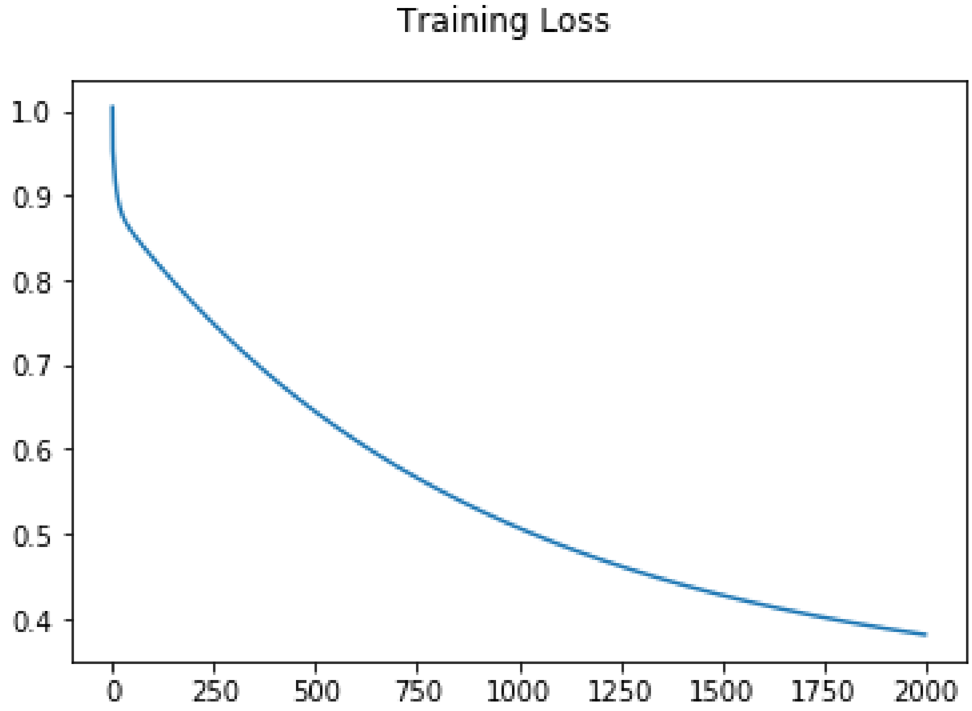
```
In [3]: def yFunction(x):
            '''Applies the function to be trained for'''
            return np.sin(2 * math.pi * x)

        def epsilon():
            '''Adds noise'''
            return np.random.normal(0, 0.3, dataPointAmount)

        def calculateDataPoints(x):
            '''Calculates noisy data points'''
            return yFunction(x) + epsilon()

        def hypothesis(x, theta):
            '''Predict value of x with the given thetas'''
            prediction = 0
            for i in range(thetaAmount):
                prediction += (x ** i) * thetas[i]
            return prediction
```

```
In [4]: # generate random x values for the data points
        X = np.random.random_sample(dataPointAmount)
        # calculate y values for the samples
        y = calculateDataPoints(X)

        # init weights randomly
        thetas = np.random.normal(0, 0.5, thetaAmount)
        # init loss history
        lossHistory = []
```

```
# train model
for epoch in range(2000):
    epochLoss = 0
    for i in range(dataPointAmount):
        # update weights
        for j in range(thetaAmount):
            loss = np.sum(y[i] - hypothesis(X[i], thetas))
            thetas[j] += alpha * loss * (X[i] ** j)
            epochLoss += loss ** 2

    # log current loss
    lossHistory.append(epochLoss / dataPointAmount)

print('Final theta values:')
print(thetas)

# sort for easier plotting
X.sort()
# recalculate y for sorted values
y = calculateDataPoints(X)
# make predictions with trained model
Y = hypothesis(X, thetas)

# plot data, expectation and prediction
fig = plt.figure()
plt.scatter(X, y)
plt.plot(X, yFunction(X), "y-", label='Target function')
plt.plot(X, Y, "r-", label='Predicted function')
plt.legend()
plt.suptitle("Data Points")

# plot loss history
fig = plt.figure()
plt.plot(np.arange(0, len(lossHistory)), lossHistory)
fig.suptitle("Training Loss")
plt.show()
```

```
Final theta values:
[  0.35397469   6.6480207  -20.78854865  13.70218289]
```


Data Points
Target function
Predicted function
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
0.0
0.2
0.4
0.6
0.8
1.0


Training Loss
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0
250
500
750
1000
1250
1500
1750
2000