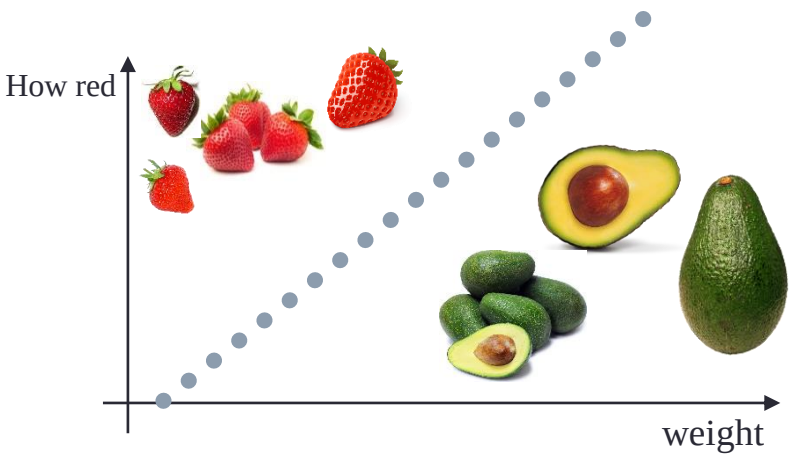
REGRESSION

CLASSIFICATION VS REGRESSION

Classification vs Regression

Classification

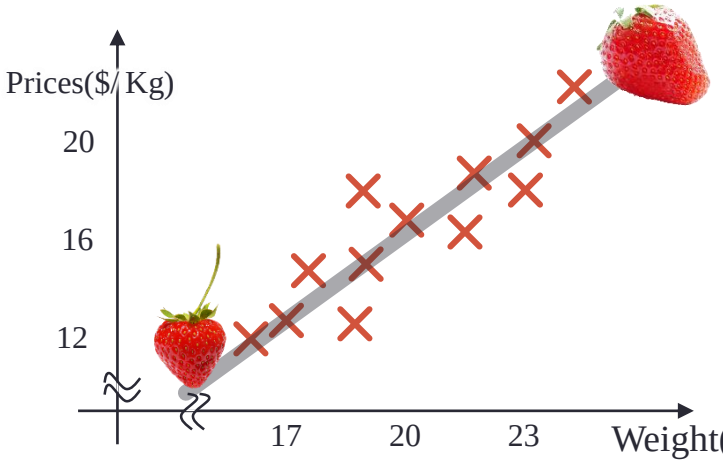


Berry?

Avocado?

The variable we are trying to predict is
DISCRETE

Regression



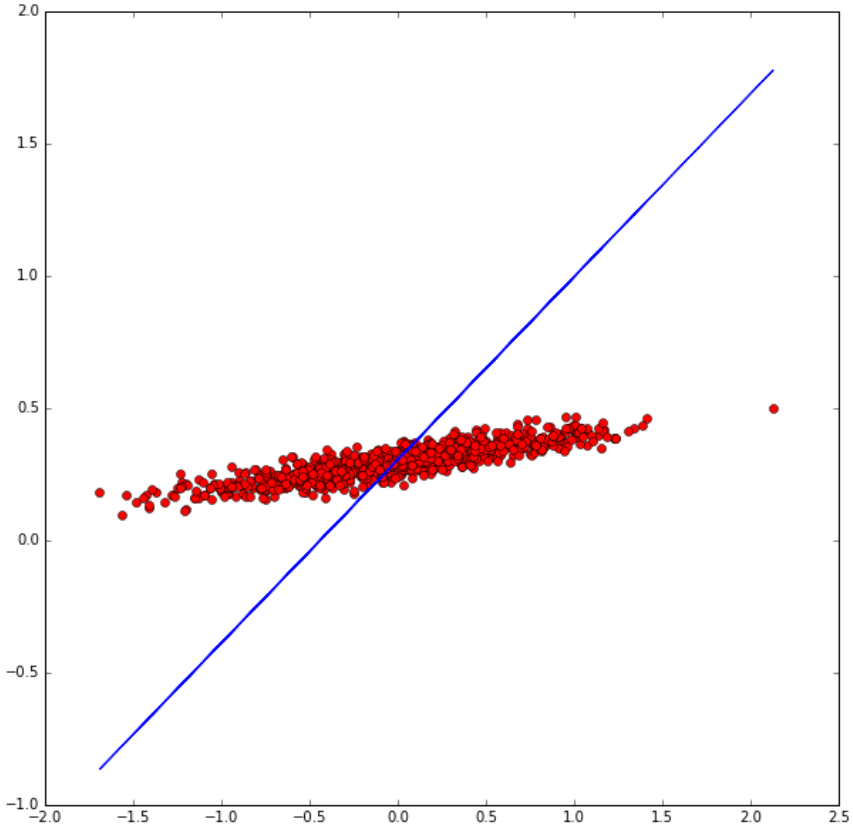
$price = f(weight)$

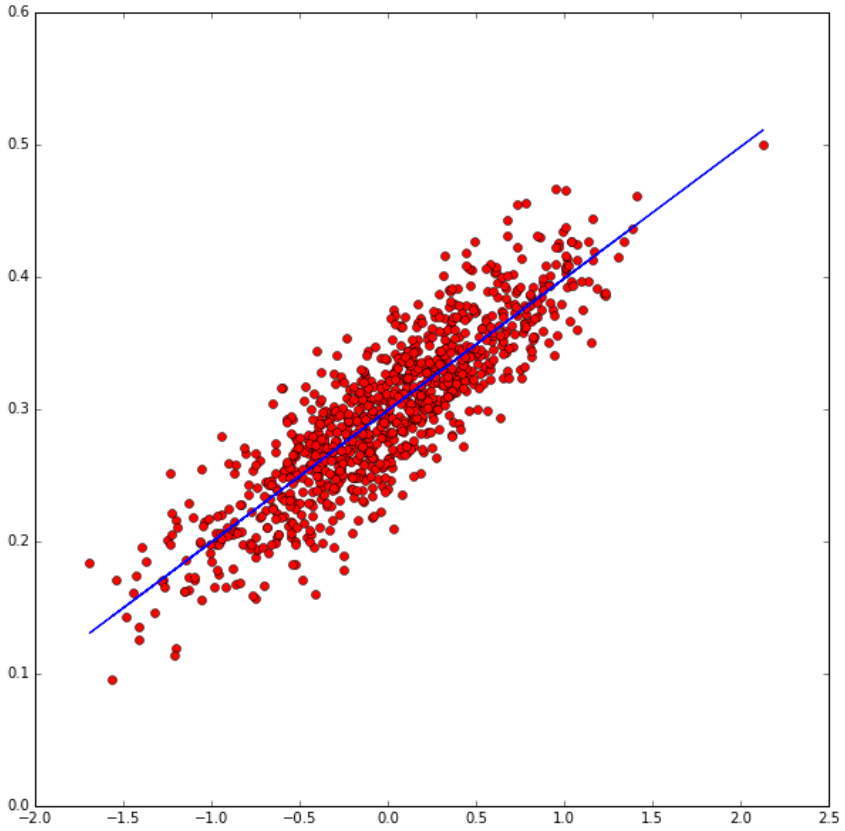
weight of a berry

The variable we are trying to predict is
CONTINUOUS

LINEAR REGRESSION

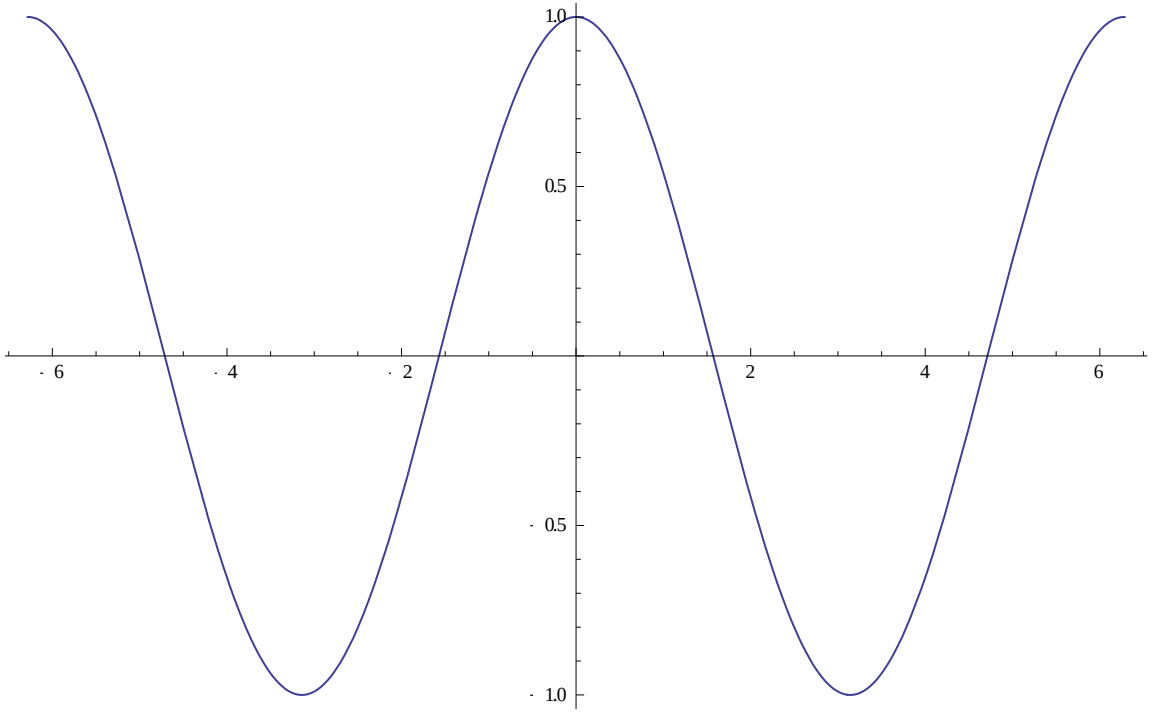
```
import tensorflow as tf
import numpy as np

# Create 100 phony x, y data points in NumPy, y = x * 0.1 + 0.3
x_data = np.random.rand(100).astype(np.float32)
y_data = x_data * 0.1 + 0.3

# Try to find values for W and b that compute y_data = W * x_data + b
# (We know that W should be 0.1 and b 0.3, but TensorFlow will
# figure that out for us.)
W = tf.Variable(tf.random_uniform([1], -1.0, 1.0))
b = tf.Variable(tf.zeros([1]))
y = W * x_data + b

# Minimize the mean squared errors.
loss = tf.reduce_mean(tf.square(y - y_data))
optimizer = tf.train.GradientDescentOptimizer(0.5)
train = optimizer.minimize(loss)

# Before starting, initialize the variables.  We will 'run' this first.
init = tf.initialize_all_variables()

# Launch the graph.
sess = tf.Session()
sess.run(init)

# Fit the line.
for step in range(201):
    sess.run(train)
    if step % 20 == 0:
        print(step, sess.run(W), sess.run(b))

# Learns best fit is W: [0.1], b: [0.3]
```


2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
2.5

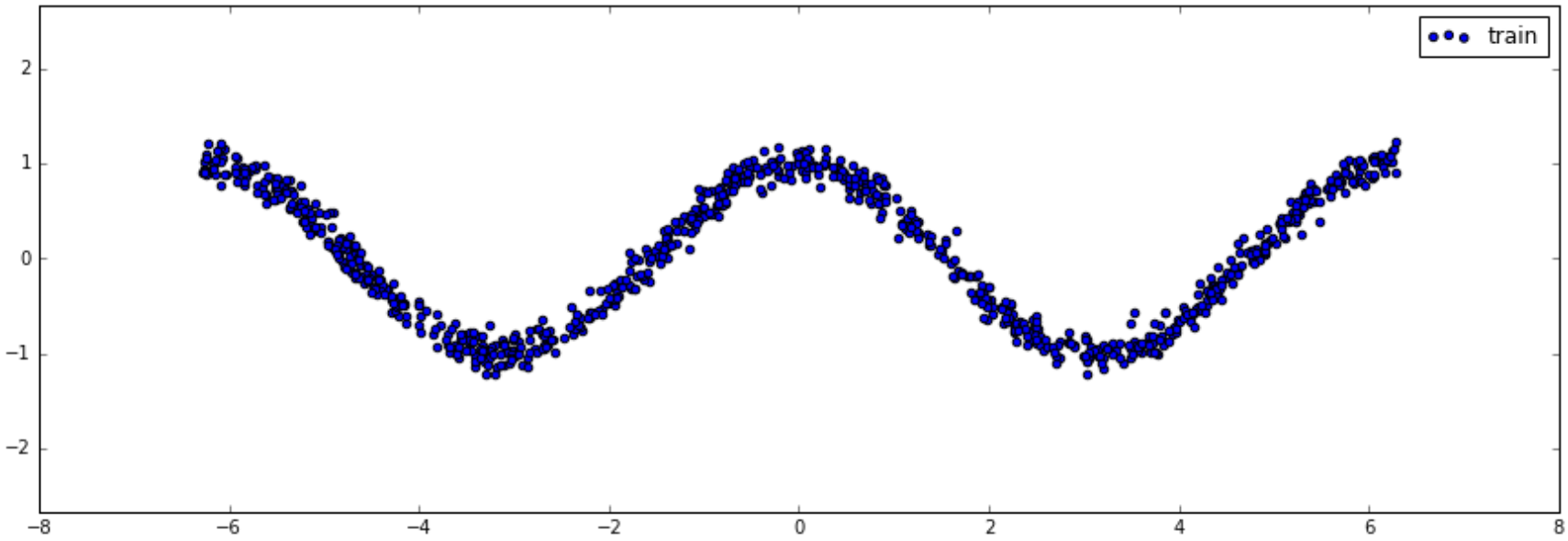

0.6
0.5
0.4
0.3
0.2
0.1
0.0
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
2.5

MLP FUNCTION APPROXIMATION

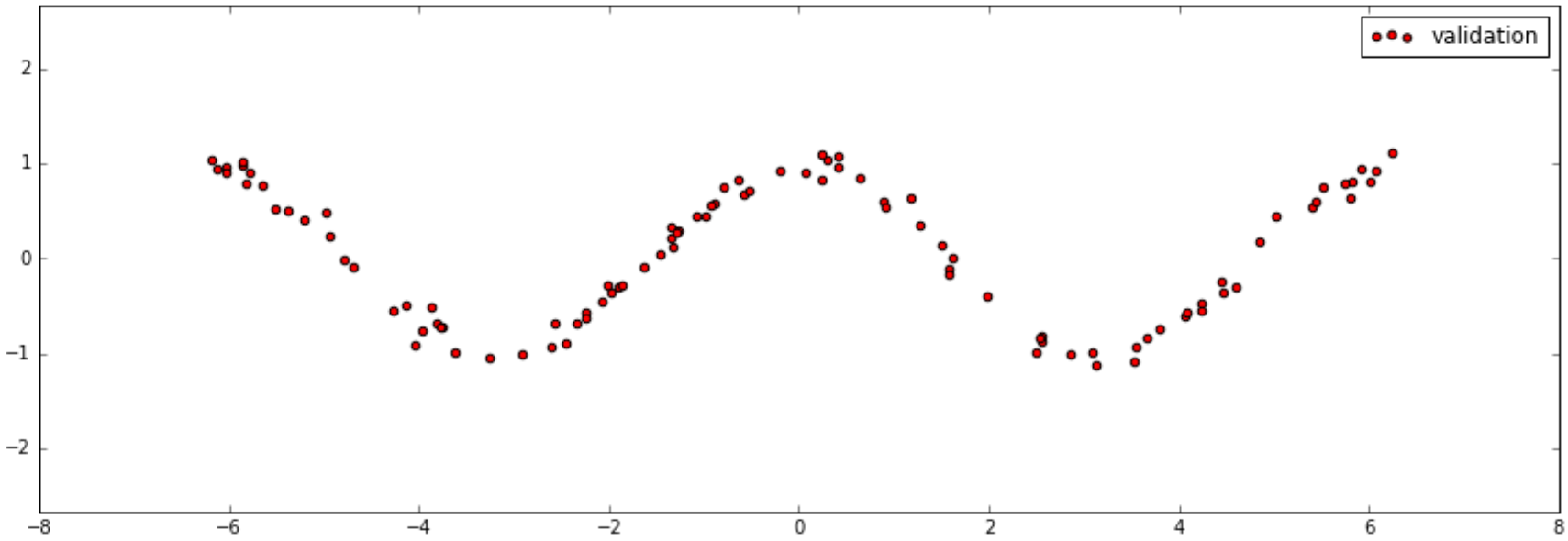
Function Apporoximation



Training samples



Test samples



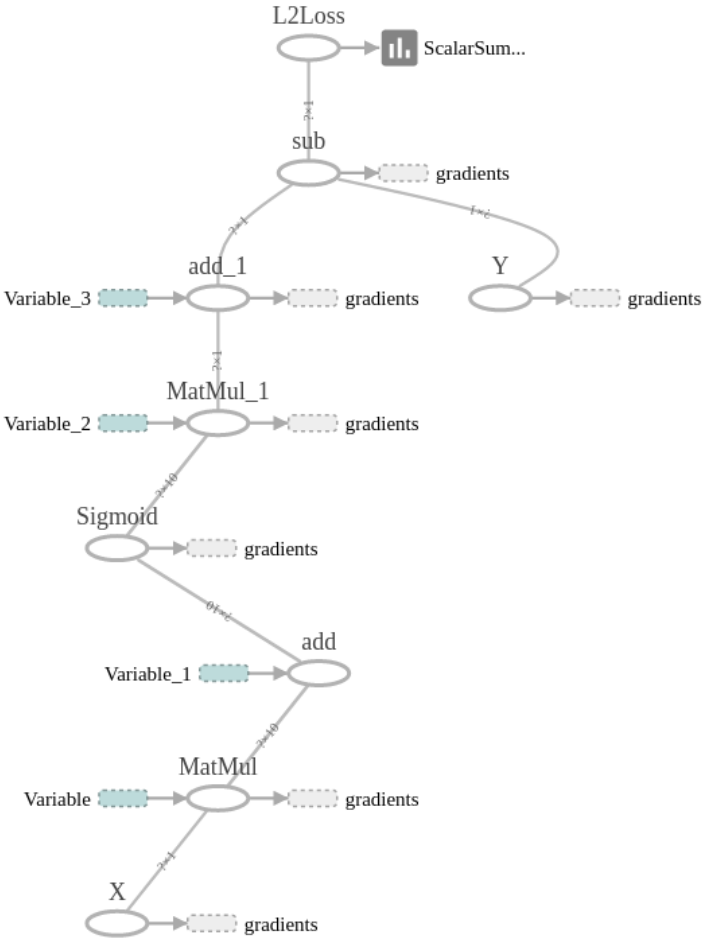
Tensorflow code

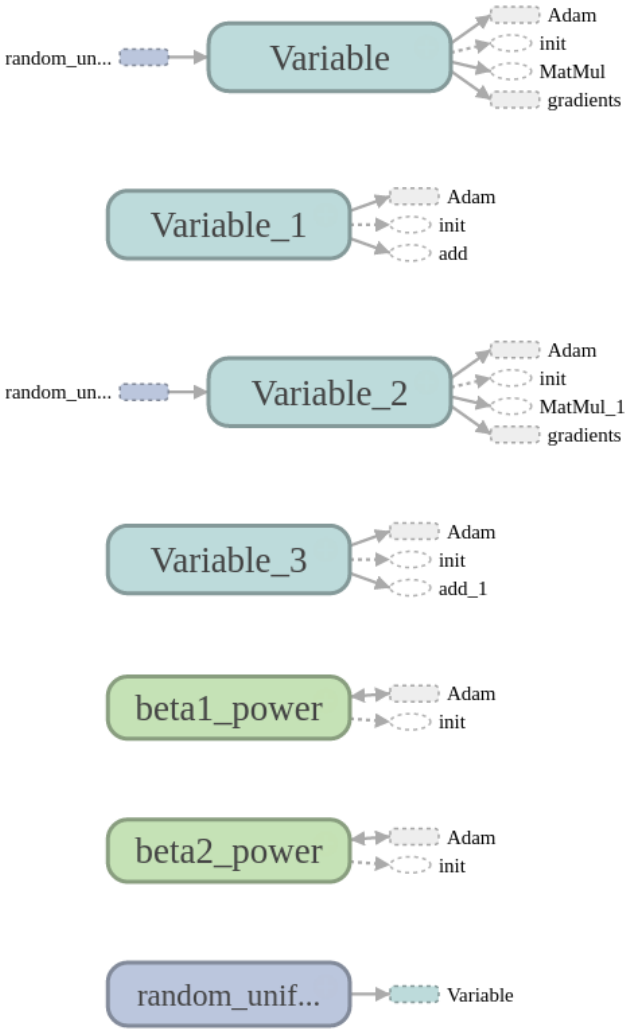
```
1  import tensorflow as tf
2  import numpy as np
3  import math, random
4  import matplotlib.pyplot as plt
5
6
7  NUM_points = 1000
8  np.random.seed(NUM_points)
9  function_to_learn = lambda x: np.cos(x) + 0.1*np.random.randn(*x.shape)
10
11 layer_1_neurons = 10
12
13 batch_size = 100
14 NUM_EPOCHS = 1500
15
16 all_x = np.float32(np.random.uniform(-2*math.pi,2*math.pi, (1, NUM_points))).T
17 np.random.shuffle(all_x)
18 train_size = int(900)
19
20
21 x_training = all_x[:train_size]
22 y_training = function_to_learn(x_training)
23
24 x_validation = all_x[train_size:]
25 y_validation = function_to_learn(x_validation)
26
27
28 X = tf.placeholder(tf.float32, [None,1], name="X")
29 Y = tf.placeholder(tf.float32, [None,1], name="Y")
```

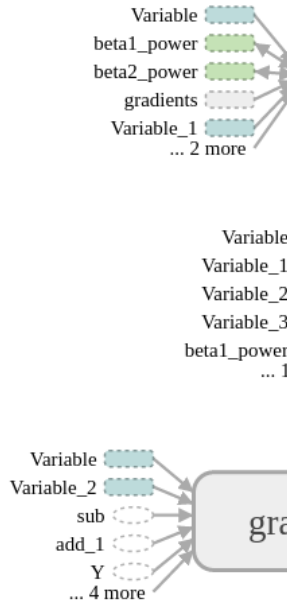
Tensorflow code

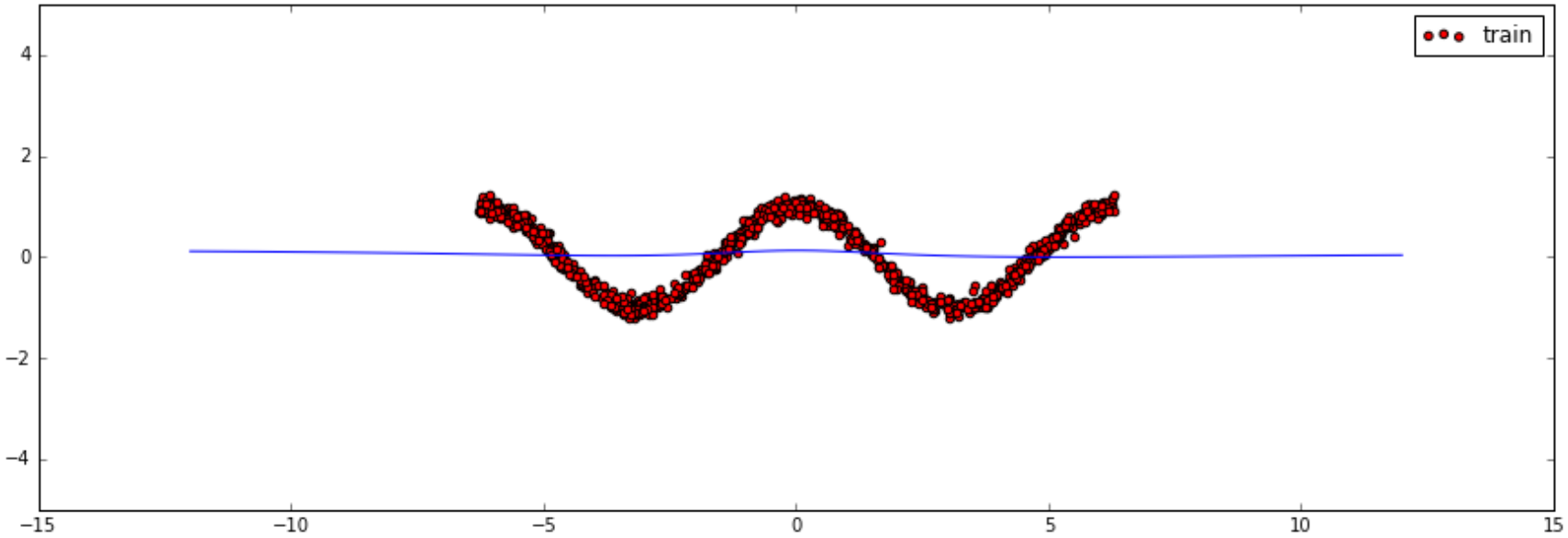
```
30
31 w_h = tf.Variable(tf.random_uniform([1, layer_1_neurons], minval=-1,maxval=1,dtype=tf.float32))
32 b_h = tf.Variable(tf.zeros([1, layer_1_neurons], dtype=tf.float32))
33
34 h = tf.nn.sigmoid(tf.matmul(X,w_h)+b_h)
35
36 w_o = tf.Variable(tf.random_uniform([layer_1_neurons,1],minval=-1,maxval=1,dtype=tf.float32))
37 b_o = tf.Variable(tf.zeros([1,1],dtype=tf.float32))
38
39 model = tf.matmul(h, w_o) + b_o
40
41 train_op = tf.train.GradientDescentOptimizer(0.003).minimize(tf.nn.l2_loss(model-Y))
42
43 init = tf.initialize_all_variables()
44 sess = tf.Session()
45 sess.run(init)
46
47 train_writer = tf.train.SummaryWriter("./curvefitlog", sess.graph)
48
49 errors = []
50 for i in range(NUM_EPOCHS):
51     for start, end in zip( range(0,len(x_training), batch_size), range(batch_size,len(x_training),batch_size)):
52         sess.run(train_op, feed_dict={X: x_training[start:end],Y:y_training[start:end]})
53     cost = sess.run(tf.nn.l2_loss(model-y_validation),feed_dict = {X:x_validation})
54     errors.append(cost)
55     if i%100 == 0: print "epoch %d, cost = %g" %(i,cost)
```

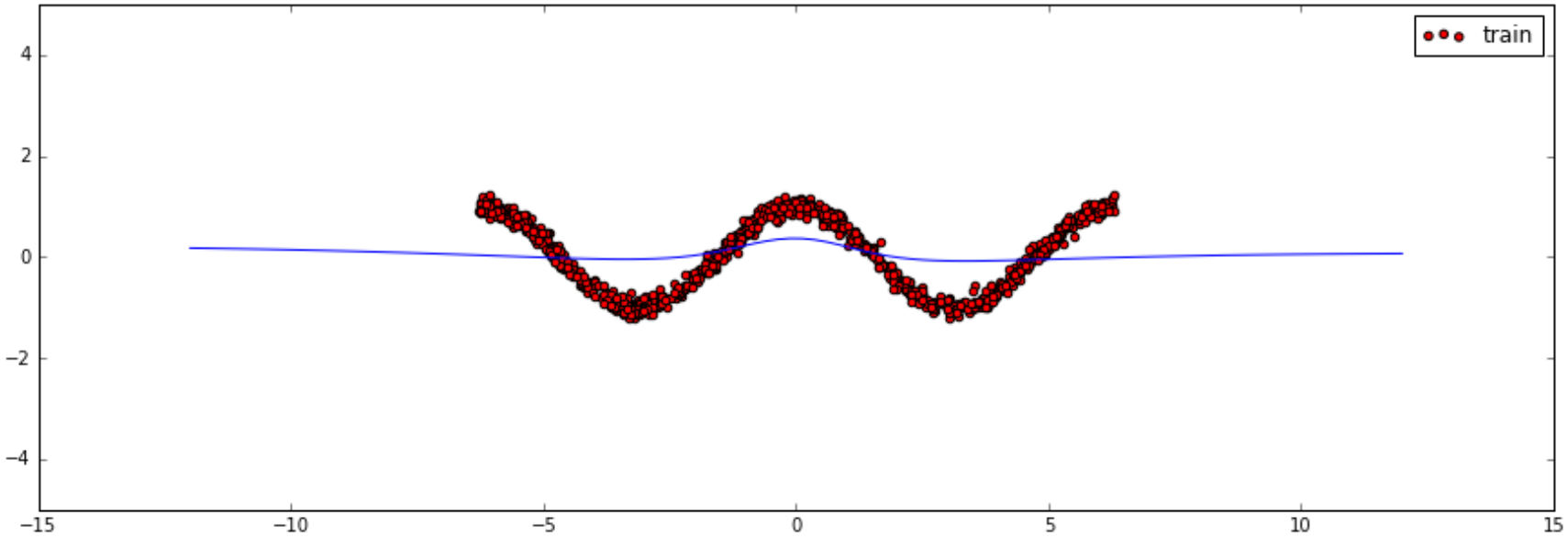
Graph

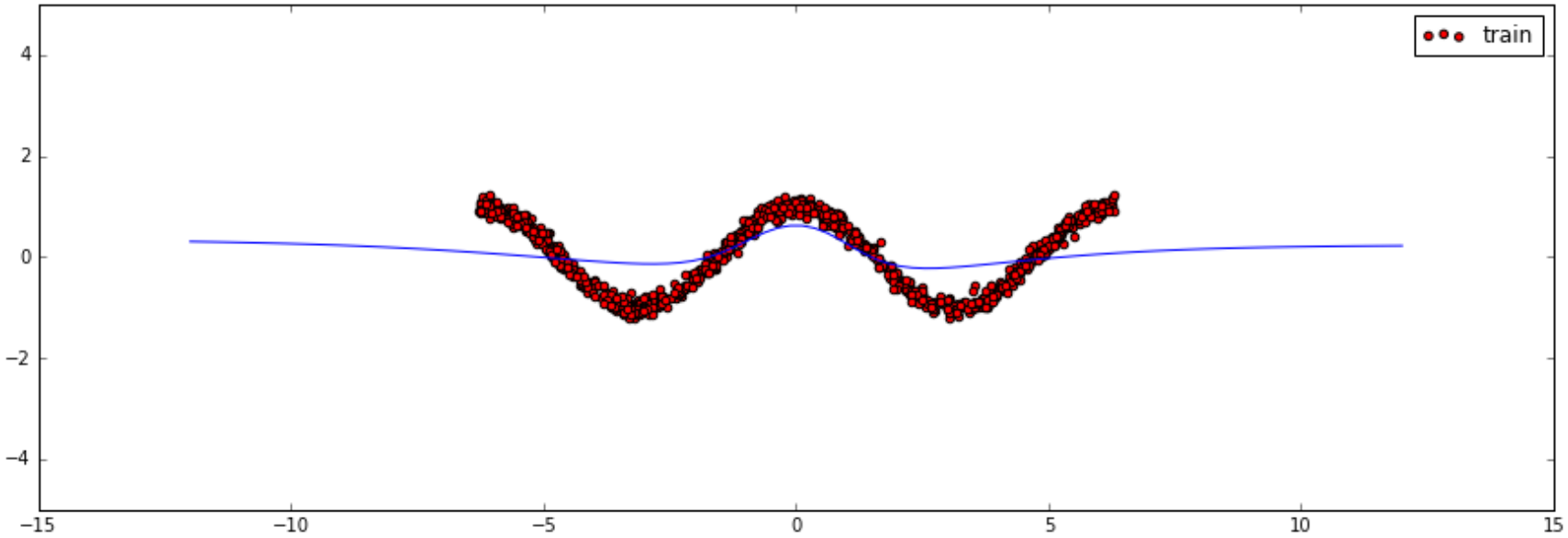

L2Loss
ScalarSum...
sub
gradients
add_1
Y
Variable_3
gradients
gradients
MatMul_1
Variable_2
gradients
Sigmoid
gradients
add
Variable_1
MatMul
Variable
gradients
X
gradients

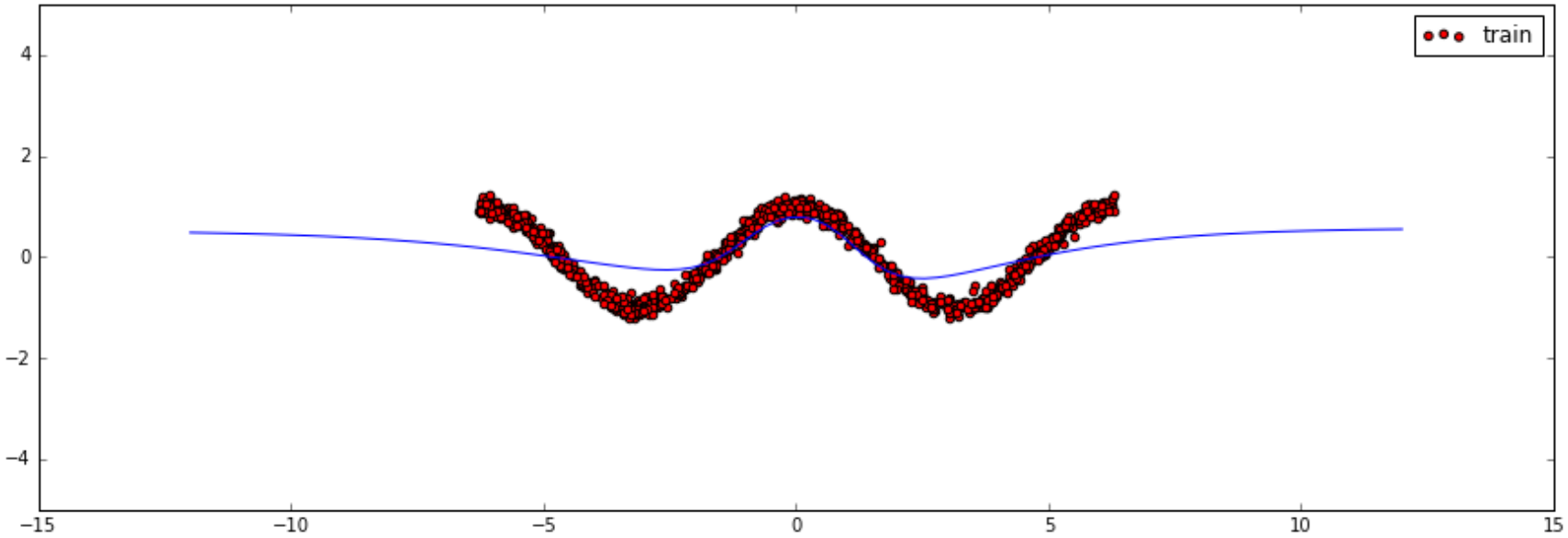

random_un...
Variable
Adam
init
MatMul
gradients
Variable_1
Adam
init
add
random_un...
Variable_2
Adam
init
MatMul_1
gradients
Variable_3
Adam
init
add_1
beta1_power
Adam
init
beta2_power
Adam
init
random_unif...
Variable

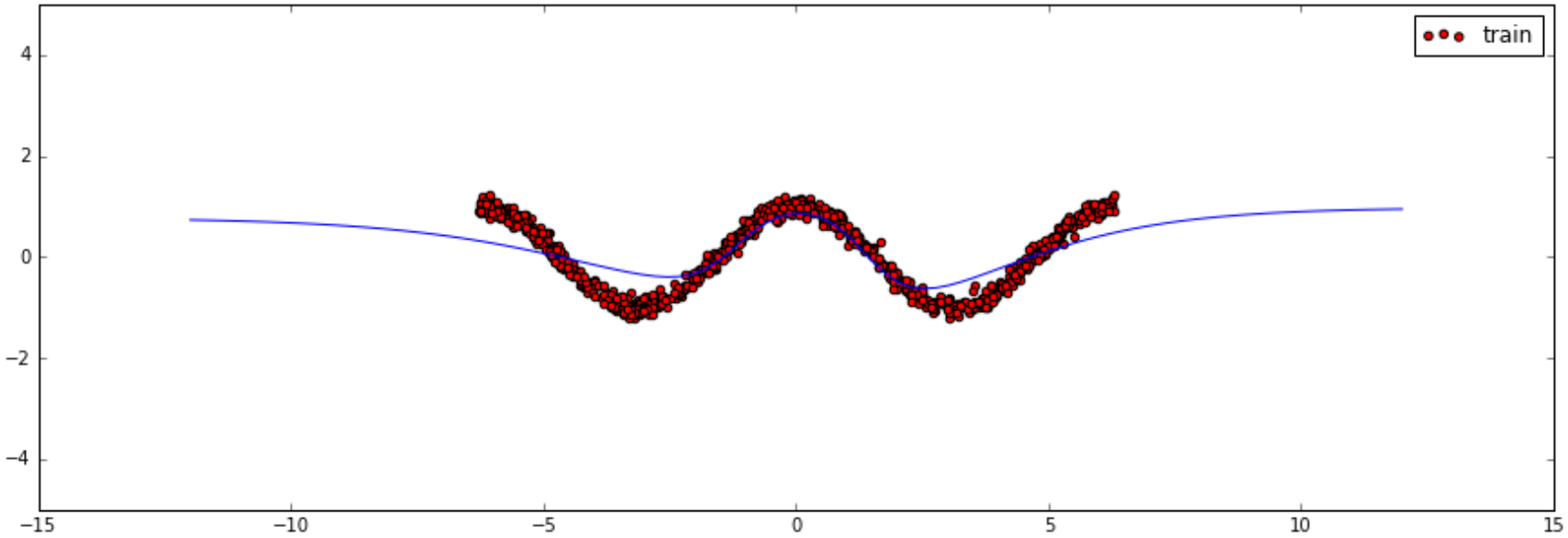

Variable
beta1_power
beta2_power
gradients
Variable_1
... 2 more
Variable
Variable_1
Variable_2
Variable_3
beta1_power
... 1
Variable
Variable_2
sub
add_1
Y
... 4 more
gra

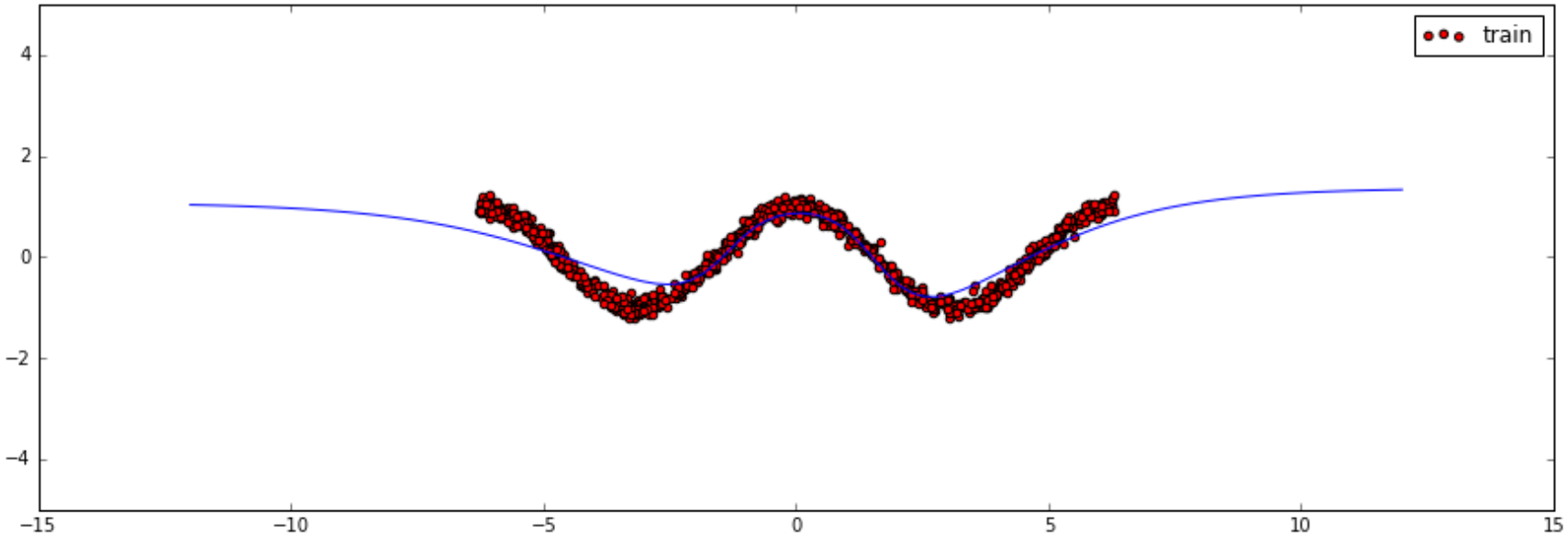

train
4
2
0
−2
−4
−15
−10
−5
0
5
10
15

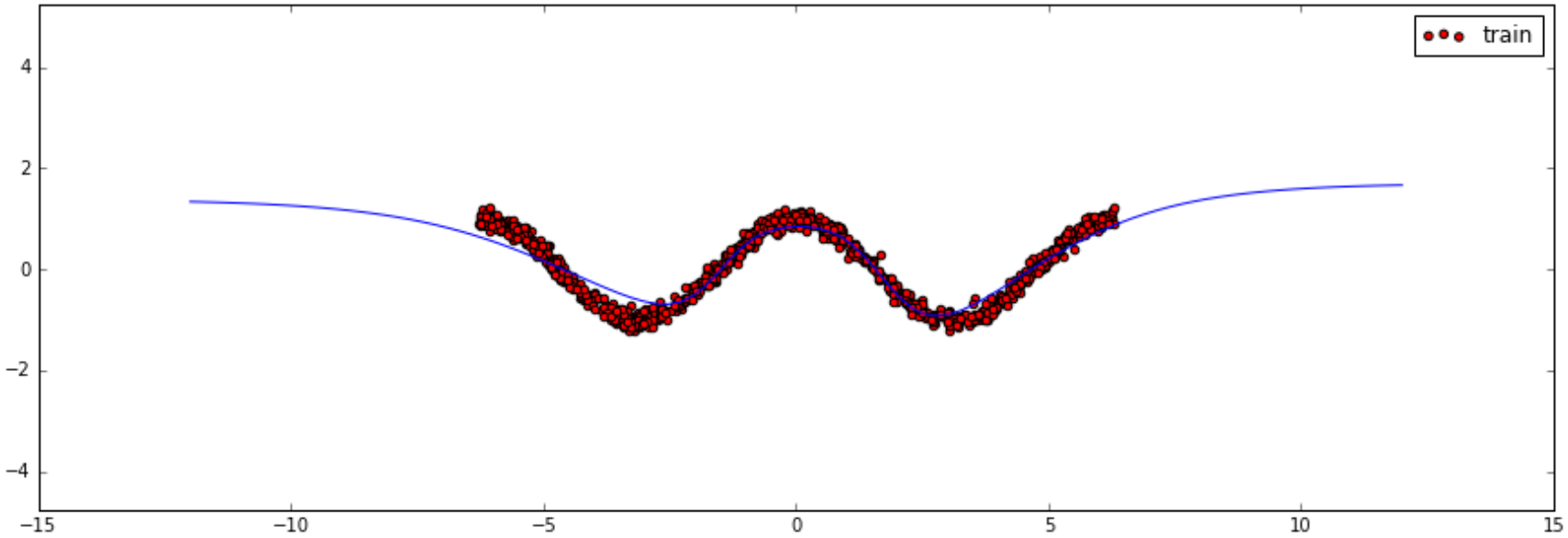

train
4
2
0
−2
−4
−15
−10
−5
0
5
10
15

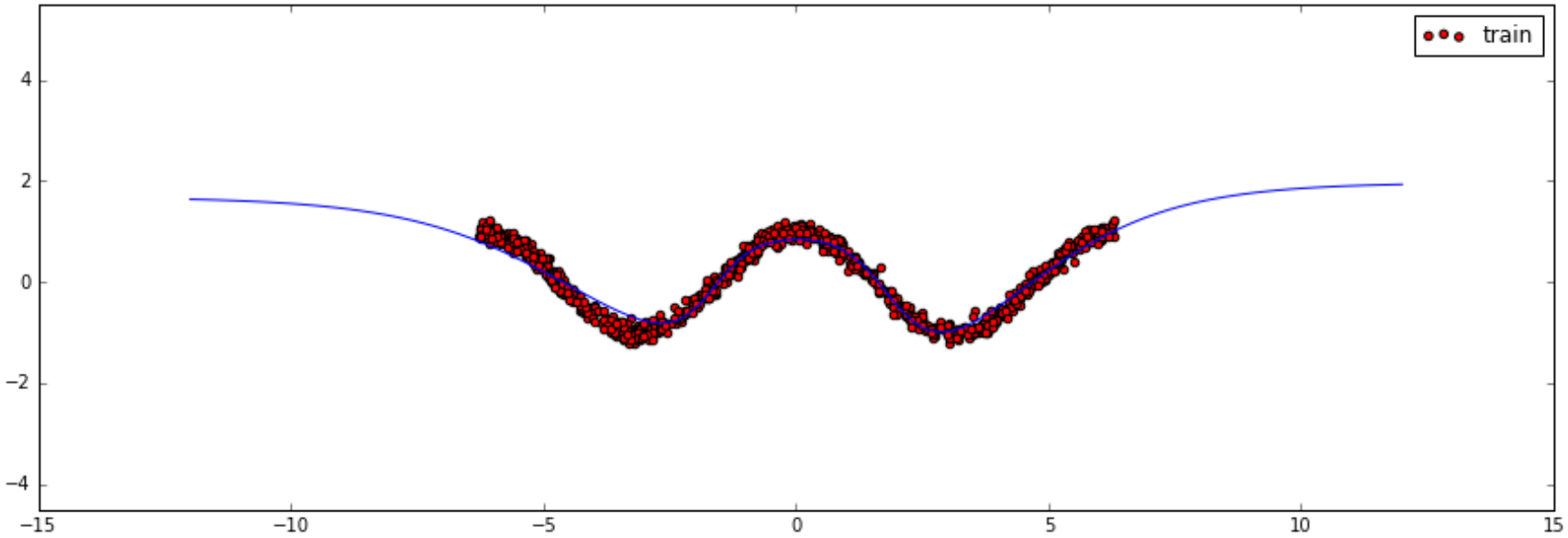

train
4
2
0
−2
−4
−15
−10
−5
0
5
10
15

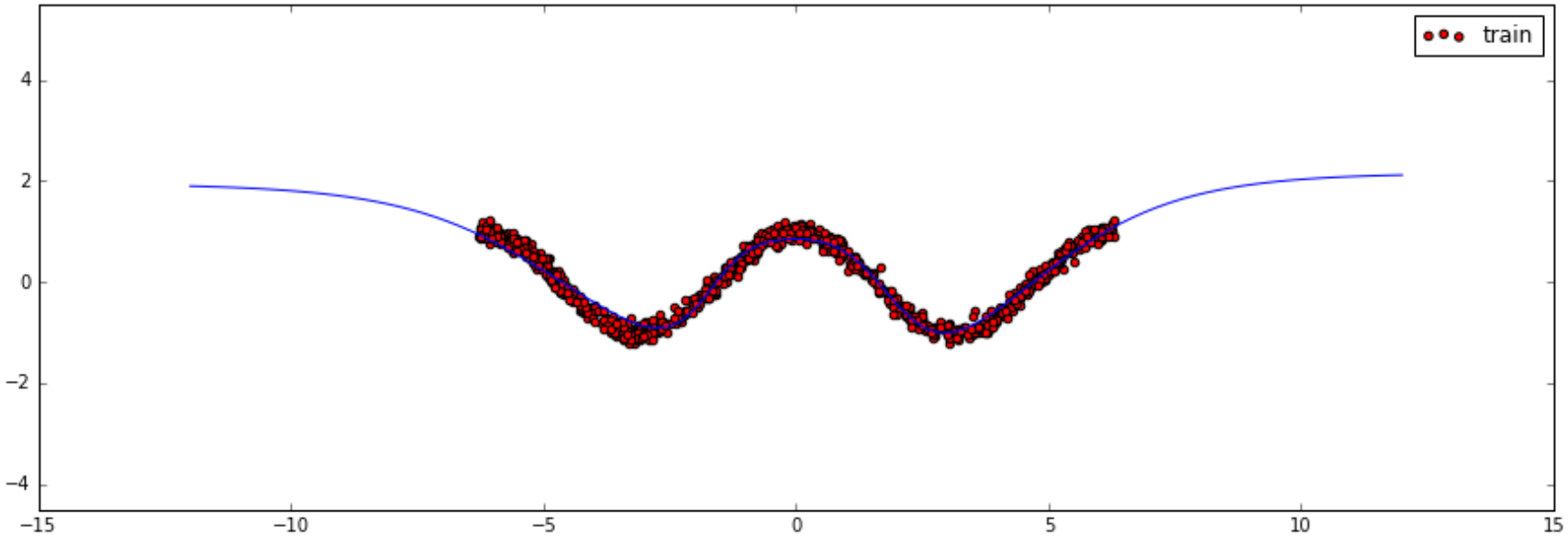

train
4
2
0
−2
−4
−15
−10
−5
0
5
10
15

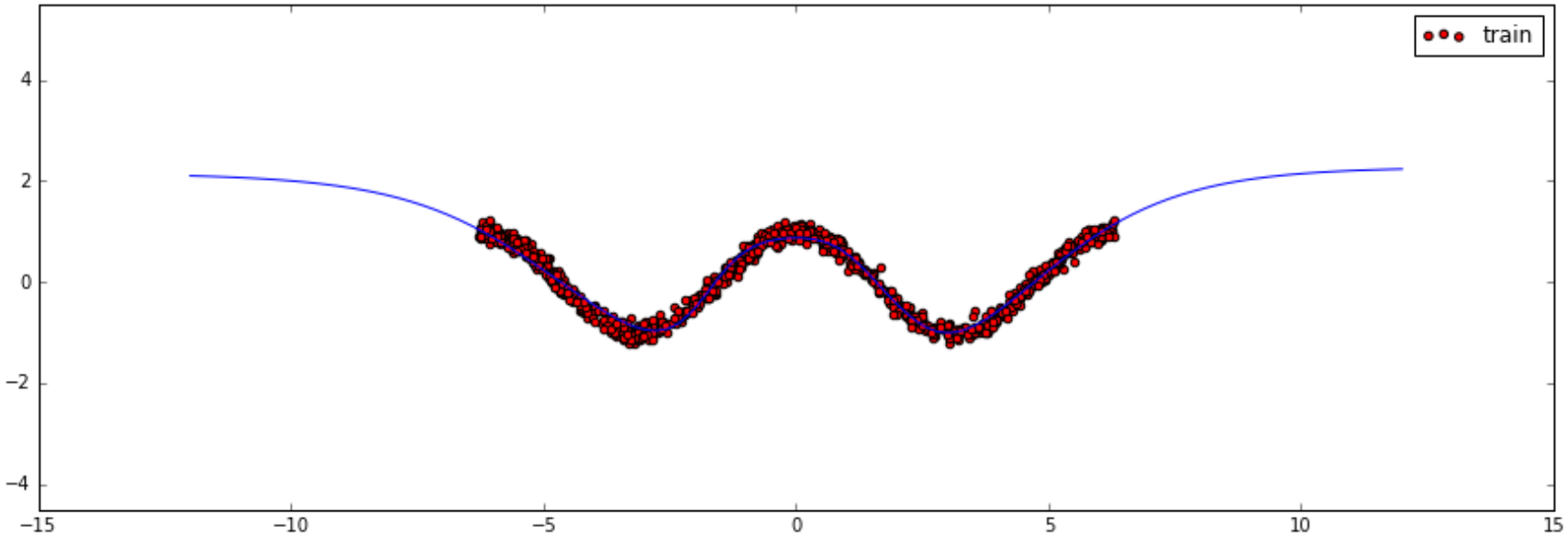

train
4
2
0
−2
−4
−15
−10
−5
0
5
10
15

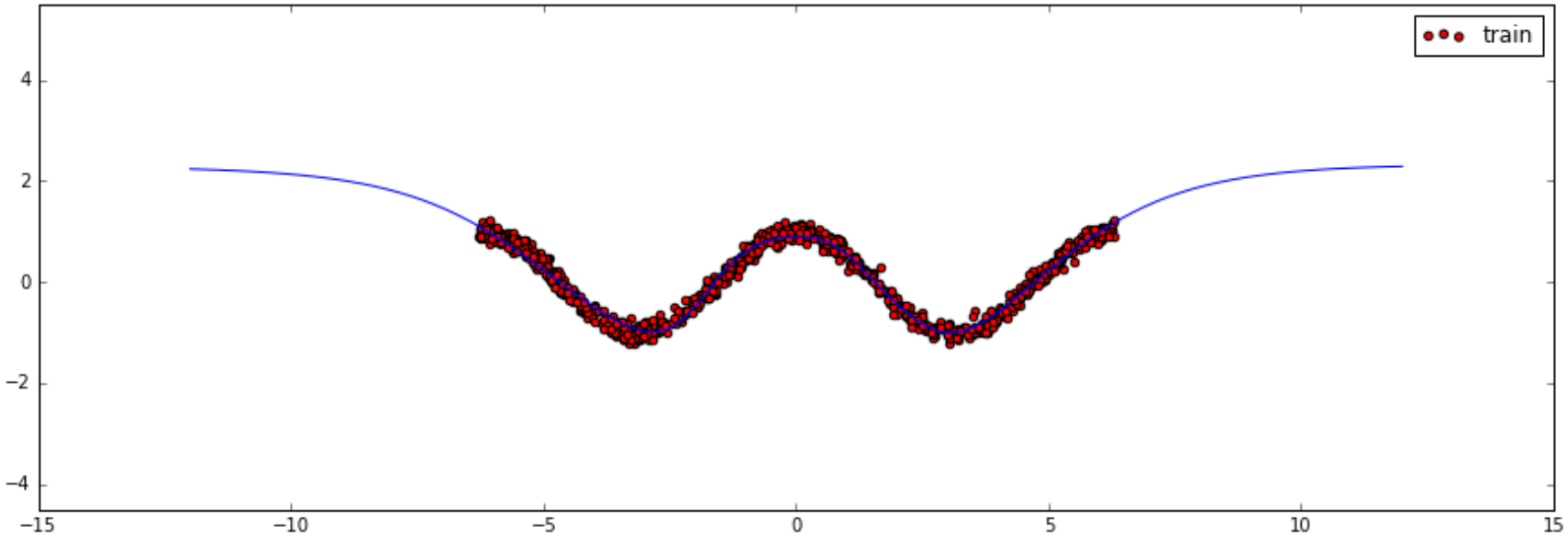

train
4
2
0
−2
−4
−15
−10
−5
0
5
10
15

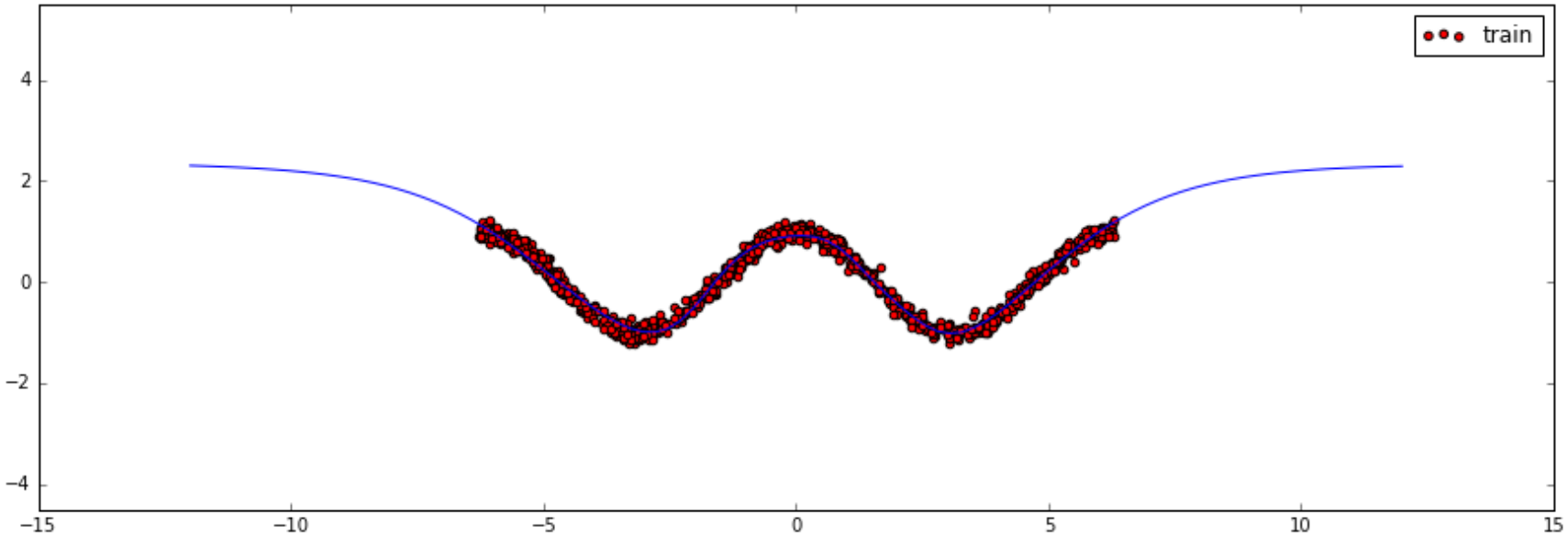

train
4
2
0
−2
−4
−15
−10
−5
0
5
10
15


train
4
2
0
−2
−4
−15
−10
−5
0
5
10
15


train
4
2
0
−2
−4
−15
−10
−5
0
5
10
15


train
4
2
0
−2
−4
−15
−10
−5
0
5
10
15


train
4
2
0
−2
−4
−15
−10
−5
0
5
10
15


train
4
2
0
−2
−4
−15
−10
−5
0
5
10
15