Regularization and SVM

부산대학교
Information Security & IoT Lab
조동근
2017.11.08

목 차

1. Review

1-1 Hypothesis, Cost function, Gradient Descent
1-2 Variable, Placeholder, Session
1-3 Example

2. Regularization

2-1 Overfitting Problem
2-2 Regularization
2-3 Example

3. Support Vector Machine(SVM)

3-1 SVM
3-2 Example

1. Review

1-1 Hypothesis, Cost function, Gradient Descent
1-2 Variable, Placeholder, Session
1-3 Example

Neural Network 정리

- **Hypothesis** : 임의의 입력에 대해 예측 및 분류를 수행하는 가설 함수
- **Cost function** : 이 함수는 선택된 w, b가 적절한 가설인지 여부를 결정
- **Gradient Descent** : 최적의 가설 함수를 찾기 위해 cost function 최소값을 찾는 방법

➢ 그 외 : activation function(Softmax, ReLu, sigmoid, tahn), mini batch, epoch …

$$h_w(x) = w \cdot x + b$$ ..... hypothesis

$$J(w,b) = \frac{1}{2n}\sum_x (h(x) - y)^2$$

$$J(w,b) = -\frac{1}{n}\sum_x [h(x)\ln a + (1-y)\ln(1-h(x))]$$ ..... Cost Function

$$w_j := w_j - \alpha\frac{\partial}{\partial w_j}J(\theta)$$ ..... Gradient Descent

Tensorflow 정리

- **Tf.Variable** : weight 와 bias 변수 선언 시 사용

```
W = tf.Variable(tf.random_normal(shape=[784, 10], stddev=0.01), name="weights")

b = tf.Variable(tf.zeros([1, 10]), name="bias")
```

- **Tf.placeholder** : feed_dict와 함께 사용되며 세션을 수행할 때 데이터를 입력함

```
tf.placeholder(dtype, shape=None, name=None)

sess.run(optimizer, feed_dict={X: x, Y: y})
```

- **Tf.Session** : tensorflow 그래프를 생성하여 코드를 수행

```
print (sess.run(x))
```

➢ 그 외 : tensor, node, edge, operation, tensorboard…

예제 소스(Linear Regression을 Tensorflow로 구현)

In [1]:
```
import tensorflow as tf
```

In [2]:
```
x_data = [2, 3, 4, 6, 9, 10]          #time
y_data = [40, 40, 45, 80, 100, 100] #score
```

In [3]:
```
# shape : 1 array, value: random(between -1.0 to 1.0)
W = tf.Variable(tf.random_uniform([1], -1.0, 1.0))
b = tf.Variable(tf.random_uniform([1], -1.0, 1.0))
```

In [4]:
```
# append placeholder
X = tf.placeholder(tf.float32)
Y = tf.placeholder(tf.float32)
```

In [5]:
```
# H(x) = Wx + b
hypothesis = W * X + b
```

In [6]:
```
# mean
cost = tf.reduce_mean(tf.square(hypothesis - Y))
```

In [7]:
```
# a is learning_rate
a = tf.Variable(0.001)
```

In [8]:
```
# minimize cost
optimizer = tf.train.GradientDescentOptimizer(a)
train = optimizer.minimize(cost)
```

예제 소스(Linear Regression을 Tensorflow로 구현)

In [9]:
```
# for init tensor
init = tf.global_variables_initializer()
```

In [10]:
```
# session start
with tf.Session() as sess:
    sess.run(init)

    for step in range(10001):
        sess.run(train, feed_dict={X:x_data, Y:y_data})
        if step % 200 == 0:
            cost_value = sess.run(cost, feed_dict={X:x_data, Y:y_data})
            print (step, cost_value, sess.run(W), sess.run(b))

    print (sess.run(hypothesis, feed_dict={X:[[1], [5], [8]]}))
```

출력 결과 :

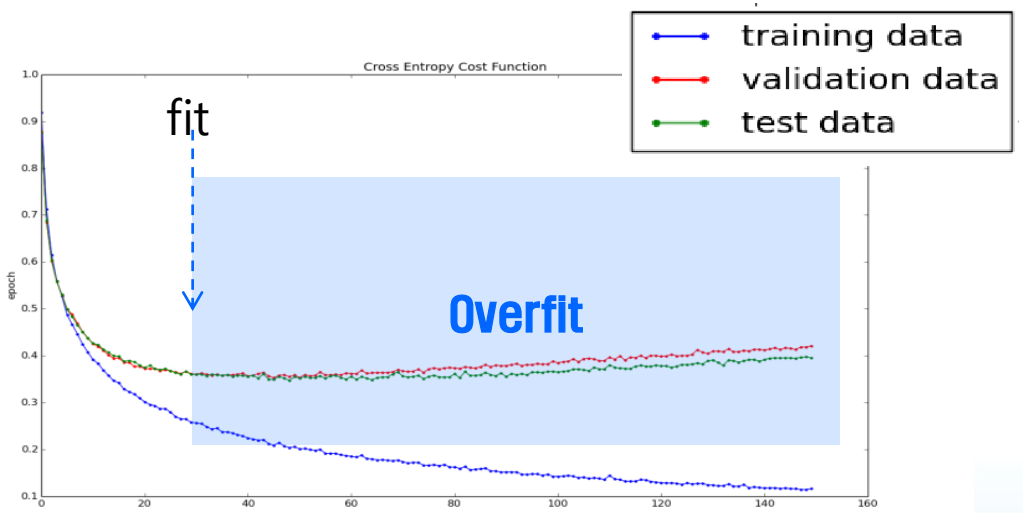
```
[[ 26.6099987 ]
 [ 61.61569214]
 [ 87.86995697]]
```

2. Regularization

2-1 Overfitting Problem
2-2 Regularization
2-3 Example

Overfitting Problem

- **regularization**
  - Layer 개수와 node의 개수가 많은 Neural network 아키텍처는 잠재적으로 overfitting 문제를 갖고 있음
  - Regularization은 overfitting 문제를 완화
  - validation data를 이용
- **Overfitting check**
  - cross validation check 를 사용
  - Training, Validation, Test data 비율을 6:2:2로 나눔



Regularization

Formula

➢ L2 regularization

$$C = C_0 + \frac{\lambda}{2n}\sum_w w^2$$

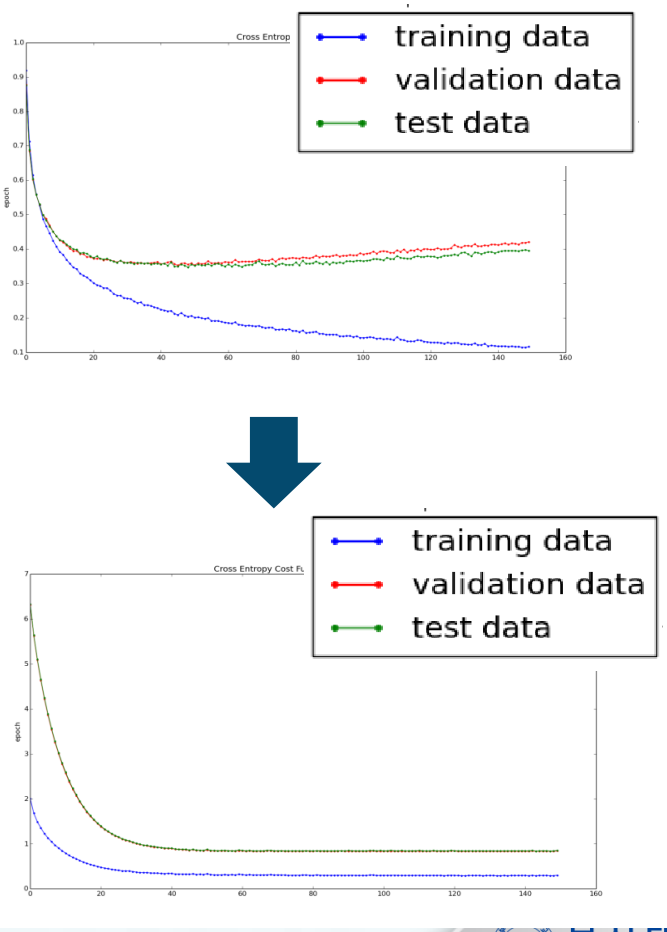
$$\frac{\partial C}{\partial w_j} = \frac{\partial C_0}{\partial w_j} + \frac{\lambda}{n} w_j$$

$$w_j := w_j - \eta \frac{\partial C}{\partial w_j}$$

$$w_j := w_j - \eta \left(\frac{\partial C_0}{\partial w_j} + \frac{\lambda}{n} w_j \right)$$

$$w_j := \left(1 - \eta \frac{\lambda}{n}\right) w_j - \eta \frac{\partial C_0}{\partial w_j}$$

적용 결과



Regularization Parameter 계산 방법

- $C = \frac{1}{2n}\sum_x (h(x) - y)^2 + \frac{\lambda}{2n}\sum_w w^2$ 에서 실험적으로 λ를 계산

Choosing the regularization parameter λ

$$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2m}\sum_{j=1}^{m}\theta_j^2$$

$$J_{train}(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

$J(\theta)$

$$J_{cv}(\theta) = \frac{1}{2m_{cv}}\sum_{i=1}^{m_{cv}}(h_\theta(x_{cv}^{(i)}) - y_{cv}^{(i)})^2$$

$$J_{test}(\theta) = \frac{1}{2m_{test}}\sum_{i=1}^{m_{test}}(h_\theta(x_{test}^{(i)}) - y_{test}^{(i)})^2$$

Regularization Parameter 계산 방법

Choosing the regularization parameter λ

Model: $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2m}\sum_{j=1}^{m}\theta_j^2$$

1. Try $\lambda = 0$ → $\min_\theta J(\theta)$ → $\theta^{(1)}$ → $J_{cv}(\theta^{(1)})$
2. Try $\lambda = 0.01$ → $\min_\theta J(\theta)$ → $\theta^{(2)}$ → $J_{cv}(\theta^{(2)})$
3. Try $\lambda = 0.02$ → $\theta^{(3)}$ → $J_{cv}(\theta^{(3)})$
4. Try $\lambda = 0.04$
5. Try $\lambda = 0.08$ → $\theta^{(5)}$ $J_{cv}(\theta^{(5)})$

⋮

12. Try $\lambda = 10$ → $\theta^{(12)}$ → $J_{cv}(\theta^{(12)})$

↑ 10.24

Pick (say) $\theta^{(5)}$. Test error: $J_{test}(\theta^{(5)})$

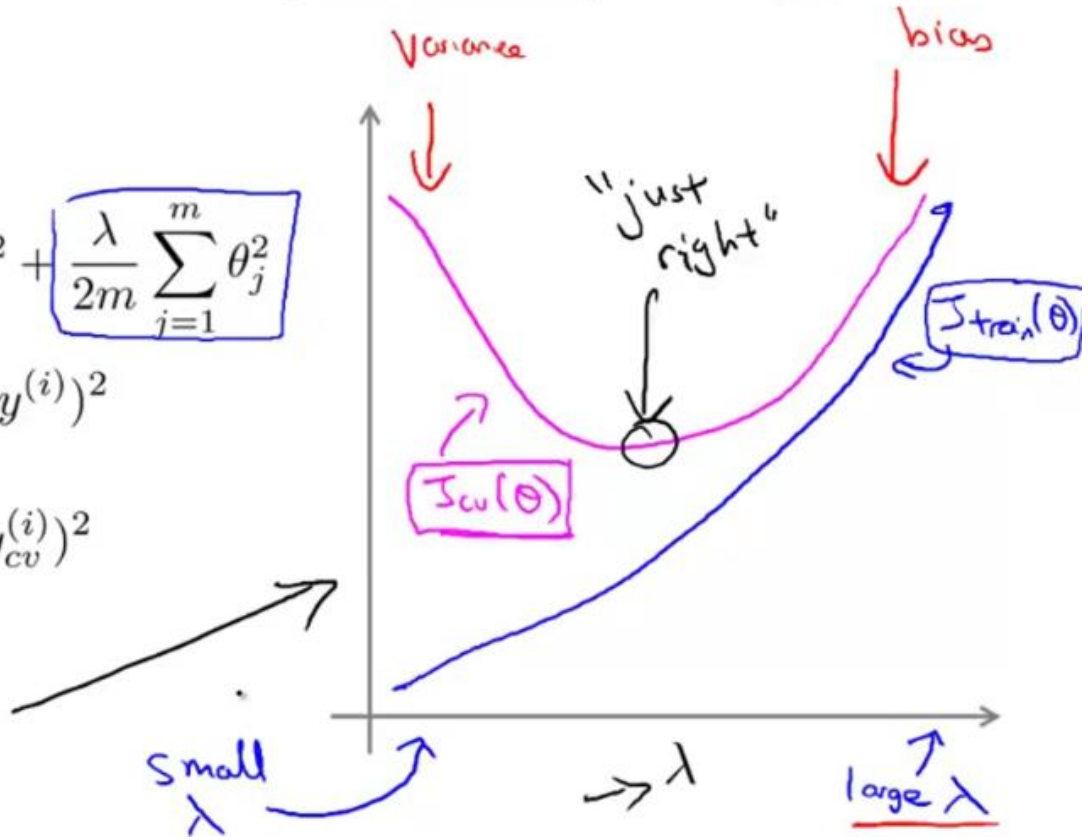
Regularization Parameter 계산 방법

Bias/variance as a function of the regularization parameter λ

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2m}\sum_{j=1}^{m}\theta_j^2$$

$$J_{train}(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

$$J_{cv}(\theta) = \frac{1}{2m_{cv}}\sum_{i=1}^{m_{cv}}(h_\theta(x_{cv}^{(i)}) - y_{cv}^{(i)})^2$$



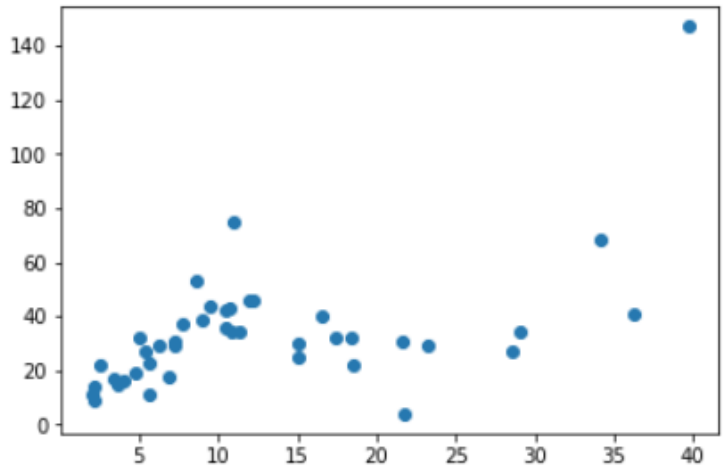
예제 소스

In [1]:
```
import tensorflow as tf
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:
```
#read data
data = pd.read_csv("./sample_data.txt", delimiter="₩t")
input_data = data["x"].values
output_data = data["y"].values
#input_data = [[1],[2],[3],[4]]
#output_data = [[10],[23],[45],[67]]
```

In [3]:
```
plt.scatter(data["x"], data["y"])
```

Out[3]:
```
<matplotlib.collections.PathCollection at 0x1e0ecc62160>
```



예제 소스

In [4]:
```
#parameter
input_size = 1
hidden_size = 20
hidden_size2 = 20
hidden_size3 = 20
output_size = 1
learning_rate = 0.01
```

In [5]:
```
#input, output placeholer
input_data = np.reshape(input_data, [-1, 1])
output_data = np.reshape(output_data, [-1, 1])

X = tf.placeholder(dtype=tf.float32, shape=[None, 1])
Y = tf.placeholder(dtype=tf.float32, shape=[None,1])
```

In [6]:
```
#weight and bias
W1 = tf.Variable(tf.random_uniform([input_size, hidden_size], -1.0, 1.0))
b1 = tf.Variable(tf.zeros([hidden_size]))

W2 = tf.Variable(tf.random_uniform([hidden_size, hidden_size2], -1.0, 1.0))
b2 = tf.Variable(tf.zeros([hidden_size2]))

W3 = tf.Variable(tf.random_uniform([hidden_size2, hidden_size3], -1.0, 1.0))
b3 = tf.Variable(tf.zeros([hidden_size3]))

W4 = tf.Variable(tf.random_uniform([hidden_size3, output_size], -1.0, 1.0))
b4 = tf.Variable(tf.zeros([output_size]))
```

예제 소스

In [7]:
```
#nerual network
z1 = tf.matmul(X, W1) + b1

a1 = tf.nn.relu(z1)

z2 = tf.matmul(a1,W2)+b2

a2 = tf.nn.relu(z2)

z3 = tf.matmul(a2, W3)+b3

a3 = tf.nn.relu(z3)

h = tf.matmul(a3, W4)+b4
```

In [8]:
```
#cost function with Regularization
lamb = 1.0
regularizer = tf.nn.l2_loss(W1) + tf.nn.l2_loss(W2) + tf.nn.l2_loss(W3) + tf.nn.l2_loss(W4)
cost = tf.reduce_mean(tf.square(h - Y)) + lamb*regularizer
```

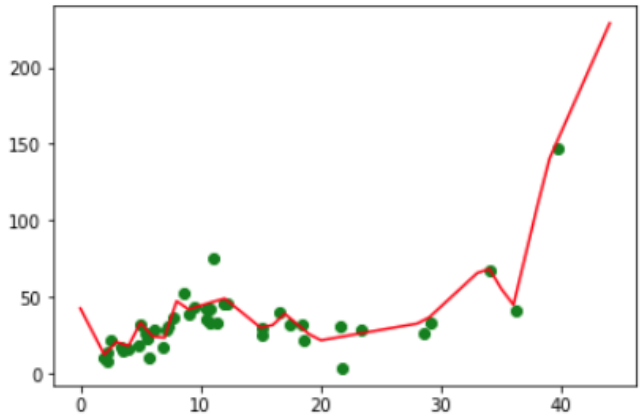
Regularization 적용

In [9]:
```
#gradient descent
opt = tf.train.AdamOptimizer(learning_rate = learning_rate).minimize(cost)
```

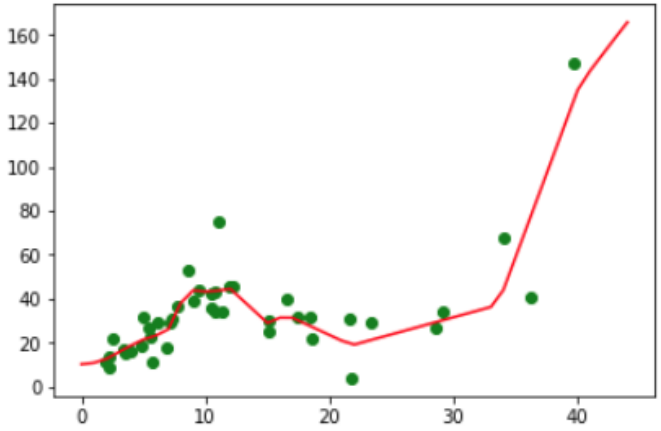
In [10]:
```
#init
sess = tf.Session()
sess.run(tf.global_variables_initializer())
```

In [11]:
```
#train
for i in range(50001):
    sess.run(opt, feed_dict={X:input_data, Y:output_data})
    if i % 10000== 0:
        loss = sess.run(cost, feed_dict={X:input_data, Y:output_data})
        print("epch: %s loss: %s "%(i, loss))
```

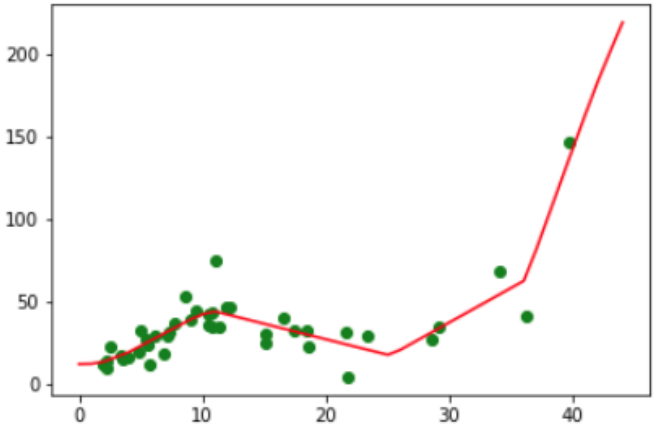
예제 소스 결과



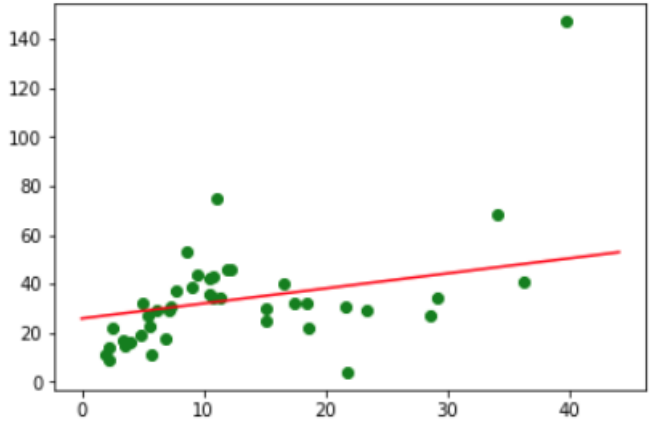
$\lambda = 0$ (variance)



$\lambda = 1$



$\lambda = 10$



$\lambda = 100$ (bias)

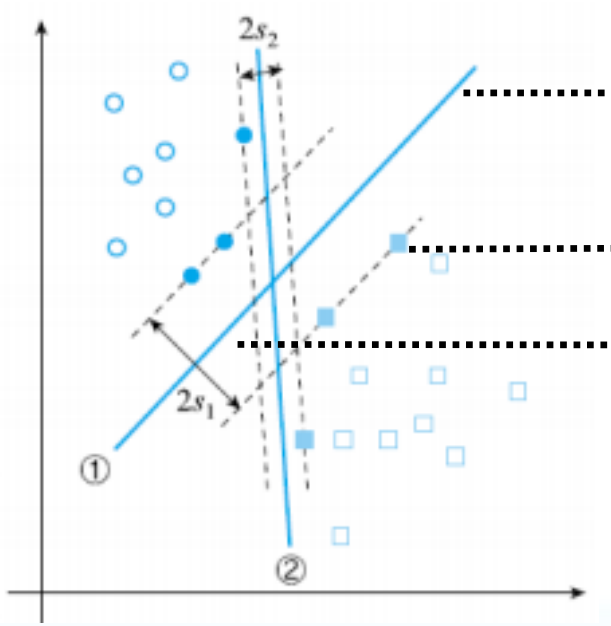
3. SVM

3-1 SVM
3-2 Example

SVM

- SVM은 Margin을 최대화하여 일반화 능력의 극대화를 꾀함
- Margin은 $\frac{2}{||w||}$으로 표현되며 이를 최대화 함
- $J(w) = \frac{||w||}{2}$ 형태로 변환하고, $t_i(w^T x_i + b) - 1 \geq 0$ 의 제한 조건을 가짐
- 라그랑제 승수법을 도입하여 w 대신 라그랑제 승수를 구하는 문제로 전환
-

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{N} \alpha_i (t_i(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + b) - 1)$$



Hyper Plane

Support Vector

Margin

- 예제 소스 https://github.com/nfmcclure/tensorflow_cookbook

In [1]:
```
import matplotlib.pyplot as plt
import numpy as np
import tensorflow as tf
from sklearn import datasets
from tensorflow.python.framework import ops
ops.reset_default_graph()
```

Start a computational graph.

In [2]:
```
sess = tf.Session()
```

Load the data

In [3]:
```
# iris.data = [(Sepal Length, Sepal Width, Petal Length, Petal Width)]
iris = datasets.load_iris()
x_vals = np.array([[x[0], x[3]] for x in iris.data])
y_vals = np.array([1 if y == 0 else -1 for y in iris.target])
```

예제 소스

Split data into train/test sets

In [4]:
```
train_indices = np.random.choice(len(x_vals),
                                 round(len(x_vals)*0.8),
                                 replace=False)
test_indices = np.array(list(set(range(len(x_vals))) - set(train_indices)))
x_vals_train = x_vals[train_indices]
x_vals_test = x_vals[test_indices]
y_vals_train = y_vals[train_indices]
y_vals_test = y_vals[test_indices]
```

Set model parameters, placeholders, and coefficients.

In [5]:
```
# Declare batch size
batch_size = 100

# Initialize placeholders
x_data = tf.placeholder(shape=[None, 2], dtype=tf.float32)
y_target = tf.placeholder(shape=[None, 1], dtype=tf.float32)

# Create variables for SVM
A = tf.Variable(tf.random_normal(shape=[2, 1]))
b = tf.Variable(tf.random_normal(shape=[1, 1]))
```

예제 소스

Declare our model and L2 Norm

SVM linear model is given by the equation:

$$\left[\frac{1}{n}\sum_{i=1}^{n}\max(0, 1 - A \cdot x - b)\right] + \alpha \cdot \|A\|^2$$

Our loss function will be the above quantity and we will tell TensorFlow to minimize it. Note that n is the number of points (in a batch), A is the hyperplane-normal vector (to solve for), b is the hyperplane-offset (to solve for), and α is the soft-margin parameter.

In [6]:
```
# Declare model operations
model_output = tf.subtract(tf.matmul(x_data, A), b)

# Declare vector L2 'norm' function squared
l2_norm = tf.reduce_sum(tf.square(A))
```

예제 소스

In [7]:
```
# Declare loss function
# Loss = max(0, 1-pred*actual) + alpha * L2_norm(A)^2
# L2 regularization parameter, alpha
alpha = tf.constant([0.01])
# Margin term in loss
classification_term = tf.reduce_mean(tf.maximum(0., tf.subtract(1., tf.multiply(model_output, y_target))))
# Put terms together
loss = tf.add(classification_term, tf.multiply(alpha, l2_norm))
```

Creat the prediction function, optimization algorithm, and initialize the variables.

In [8]:
```
# Declare prediction function
prediction = tf.sign(model_output)
accuracy = tf.reduce_mean(tf.cast(tf.equal(prediction, y_target), tf.float32))

# Declare optimizer
my_opt = tf.train.GradientDescentOptimizer(0.01)
train_step = my_opt.minimize(loss)

# Initialize variables
init = tf.global_variables_initializer()
sess.run(init)
```

예제 소스

In [9]:
```
# Training loop
loss_vec = []
train_accuracy = []
test_accuracy = []
for i in range(500):
    rand_index = np.random.choice(len(x_vals_train), size=batch_size)
    rand_x = x_vals_train[rand_index]
    rand_y = np.transpose([y_vals_train[rand_index]])
    sess.run(train_step, feed_dict={x_data: rand_x, y_target: rand_y})

    temp_loss = sess.run(loss, feed_dict={x_data: rand_x, y_target: rand_y})
    loss_vec.append(temp_loss)

    train_acc_temp = sess.run(accuracy, feed_dict={
        x_data: x_vals_train,
        y_target: np.transpose([y_vals_train])})
    train_accuracy.append(train_acc_temp)

    test_acc_temp = sess.run(accuracy, feed_dict={
        x_data: x_vals_test,
        y_target: np.transpose([y_vals_test])})
    test_accuracy.append(test_acc_temp)

    if (i + 1) % 100 == 0:
        print('Step #{} A = {}, b = {}'.format(
            str(i+1),
            str(sess.run(A)),
            str(sess.run(b))
        ))
        print('Loss = ' + str(temp_loss))
```

예제 소스

Now we extract the linear coefficients and get the SVM boundary line.

In [10]:

```
# Extract coefficients
[[a1], [a2]] = sess.run(A)
[[b]] = sess.run(b)
slope = -a2/a1
y_intercept = b/a1

# Extract x1 and x2 vals
x1_vals = [d[1] for d in x_vals]

# Get best fit line
best_fit = []
for i in x1_vals:
    best_fit.append(slope*i+y_intercept)

# Separate I. setosa
setosa_x = [d[1] for i, d in enumerate(x_vals) if y_vals[i] == 1]
setosa_y = [d[0] for i, d in enumerate(x_vals) if y_vals[i] == 1]
not_setosa_x = [d[1] for i, d in enumerate(x_vals) if y_vals[i] == -1]
not_setosa_y = [d[0] for i, d in enumerate(x_vals) if y_vals[i] == -1]
```

예제 소스

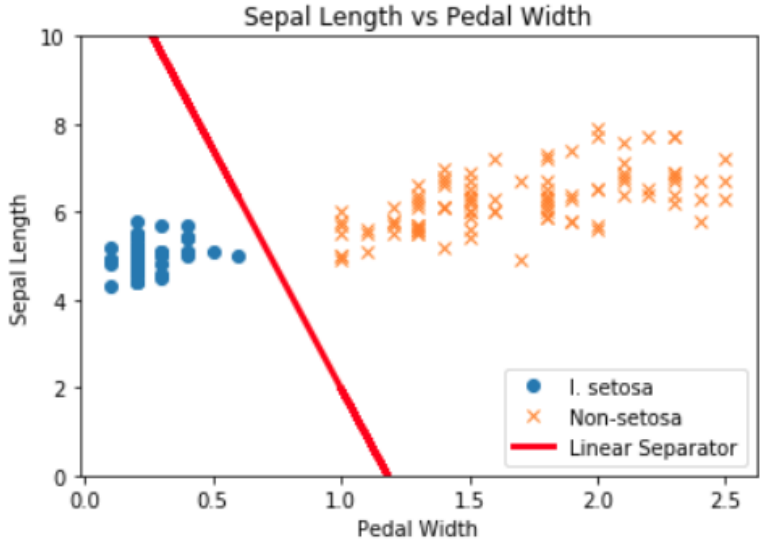
Matplotlib code for plotting

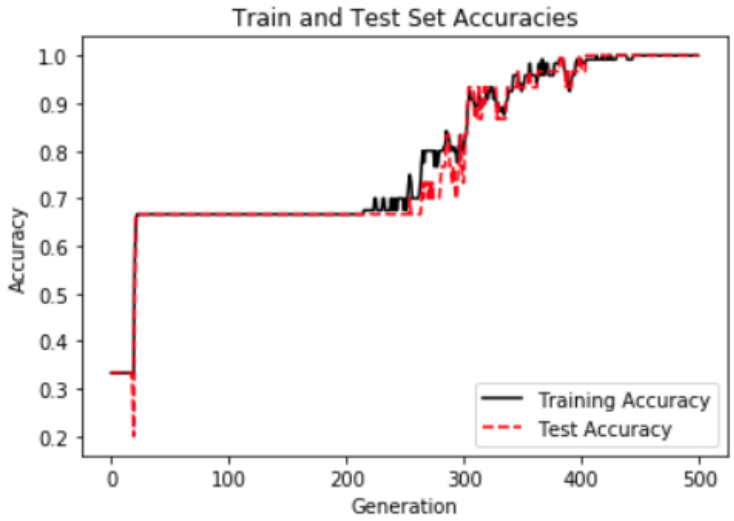
In [11]:

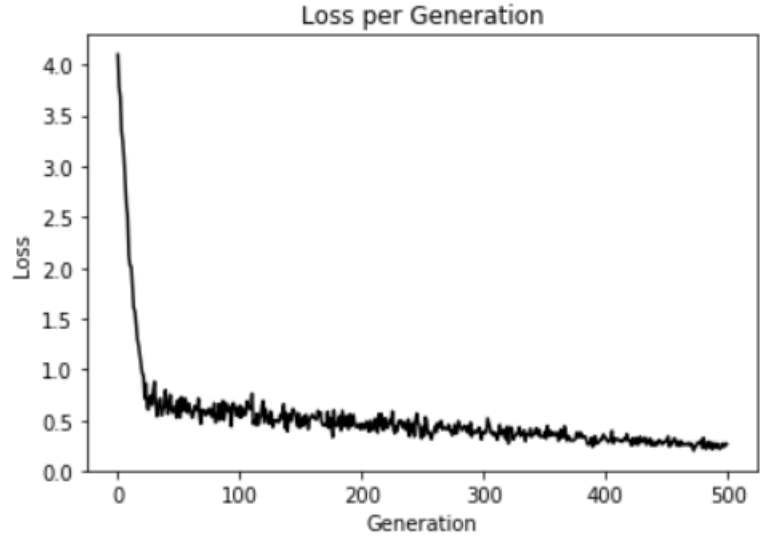
```
%matplotlib inline
# Plot data and line
plt.plot(setosa_x, setosa_y, 'o', label='I. setosa')
plt.plot(not_setosa_x, not_setosa_y, 'x', label='Non-setosa')
plt.plot(x1_vals, best_fit, 'r-', label='Linear Separator', linewidth=3)
plt.ylim([0, 10])
plt.legend(loc='lower right')
plt.title('Sepal Length vs Pedal Width')
plt.xlabel('Pedal Width')
plt.ylabel('Sepal Length')
plt.show()

# Plot train/test accuracies
plt.plot(train_accuracy, 'k-', label='Training Accuracy')
plt.plot(test_accuracy, 'r--', label='Test Accuracy')
plt.title('Train and Test Set Accuracies')
plt.xlabel('Generation')
plt.ylabel('Accuracy')
plt.legend(loc='lower right')
plt.show()

# Plot loss over time
plt.plot(loss_vec, 'k-')
plt.title('Loss per Generation')
plt.xlabel('Generation')
plt.ylabel('Loss')
plt.show()
```

예제 소스 결과

```
Step #100 A = [[-0.33496073]
 [-0.03939554]], b = [[-1.0989387]]
Loss = [ 0.5366677]
Step #200 A = [[-0.27491847]
 [-0.33351016]], b = [[-1.1501385]]
Loss = [ 0.41086033]
Step #300 A = [[-0.22121902]
 [-0.60417664]], b = [[-1.20213807]]
Loss = [ 0.38407847]
Step #400 A = [[-0.15989591]
 [-0.86473054]], b = [[-1.25803792]]
Loss = [ 0.33860666]
Step #500 A = [[-0.1022057 ]
 [-1.11257243]], b = [[-1.31473744]]
Loss = [ 0.26571095]
```







Thank you!