Thomas Whitcomb
thomaswhitcomb@yahoo.com
March 4th, 2019

Lesson 7

Lesson 7 **1**
Problem 1 2
Code 2
Results 5
Problem 2 9
Code 9
Results 10
Problem 3 13
Code 13
Results 14
Lady before filter 14
Lady after Filter#1+Filter#2 15
Lady After 5x5 16
Lady after 15x15 17
Lady after 29x29 18
Lady after 39x39 19

Problem 1

Code

```
##############################################
# MNIST Image Classification Using Linear Regression #
###############################################
# 1.1 Load the libraries
#
import sys
import matplotlib.pyplot as plt
import tensorflow as tf
import numpy as np
from sklearn.metrics import confusion_matrix
from tensorflow.examples.tutorials.mnist import input_data

def optimize(optimizer,num_iterations,learning_rate,batch_size):
    for i in range(num_iterations):
        x_batch, y_true_batch = data.train.next_batch(batch_size= batch_size)
        feed_dict_train = {x : x_batch,
                           lr: learning_rate,
                           y_true : y_true_batch}
        session.run(optimizer, feed_dict = feed_dict_train)

def print_confusion_matrix():
    cls_true = [np.argmax(label) for label in data.test.labels]
    cls_pred = session.run(y_pred_cls, feed_dict = feed_dict_test)
    cm = confusion_matrix(y_true = cls_true, y_pred = cls_pred)
    print(cm)

def print_accuracy(iterations,learning_rate,batch_size):
    # Use TensorFlow to compute the accuracy.
    acc = session.run(accuracy , feed_dict= feed_dict_test)
    # Print the accuracy.
    print('Accuracy : {:2.1f}% with {:d} iterations, {:1.2f} learning rate and
{:d} batch size'.format((acc*100),iterations,learning_rate,batch_size))
###############################################
# 1.2 Download and read MNIST data
#
```

```
old_v = tf.logging.get_verbosity()
tf.logging.set_verbosity(tf.logging.ERROR)
data = input_data.read_data_sets("MNIST_data/", one_hot = True)
tf.logging.set_verbosity(old_v)
#######################################################
# the images are stored in one-dimensional arrays of this length. #
img_size_flat = data.train.images[0].shape[0]
# Tuple with height and width of images used to reshape arrays.

img_shape = (28,28)
# Number of classes, one class for each of 10 digits.
num_classes = 10

data.test.cls = np.array([label.argmax() for label in data.test.labels])

###########################################
# 1.5 Plot a few images
# Get the first images from the Test-set. #
images = data.test.images[0:9]
# Get the true classes for those images.
cls_true = [np.argmax(oh) for oh in data.test.labels[0:9] ]

############################################## # 2.1 Placeholder variables
#
lr = tf.placeholder(tf.float32)
x = tf.placeholder( tf.float32, [None, img_size_flat])
y_true = tf.placeholder( tf.float32, [None, num_classes])
y_true_cls = tf.placeholder( tf.int64, [None])

############################################## # 2.2 Variables
#
weights = tf.Variable(tf.zeros([img_size_flat, num_classes]))
bias = tf.Variable(tf.zeros([num_classes]))

############################################### # 2.3 Model
#
logits = tf.matmul(x, weights) + bias
y_pred = tf.nn.softmax(logits)
y_pred_cls = tf.argmax(y_pred, axis=1)

# 2.4 Cost Function
#
cross_entropy = tf.nn.softmax_cross_entropy_with_logits_v2( logits= logits,
labels = y_true)
cost = tf.reduce_mean(cross_entropy)

################################################ # 2.5 Optimization Function
#
```

```
gradient_descent_optimizer =
tf.train.GradientDescentOptimizer(lr).minimize(cost)
adagrad_optimizer = tf.train.AdagradOptimizer(lr).minimize(cost)

# 2.6 Performance measures #
correct_prediction = tf.equal( y_pred_cls , y_true_cls)
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))

############################################## # 3.1 Create TensorFlow Session
#
session = tf.Session()

############################################# # 3.2 Initialize Variables
#

###################################################
# 3.4 Optimization Iteration
#
feed_dict_test = {
    x : data.test.images,
    y_true : data.test.labels,
    y_true_cls : [np.argmax(label) for label in data.test.labels]
}

#############################################
# 4.2 Performance Iteration#1
#
# Number of iteration means how many of batchs are iterated #
print("Gradient decent optimizer")
for lrx in [x/10 for x in range(5,0,-1)]:
    session.run(tf.global_variables_initializer())
    for i in [1,9,990]:
        optimize(gradient_descent_optimizer,num_iterations= i,learning_rate =
lrx,batch_size=100)
        print_accuracy(i,lrx,100)

#print_confusion_matrix()

print("Adagra optimizer ")
for lrx in [x/10 for x in range(5,0,-1)]:
    session.run(tf.global_variables_initializer())
    for i in [1,9,990]:
        optimize(adagrad_optimizer,num_iterations= i,learning_rate =
lrx,batch_size=100)
        print_accuracy(i,lrx,100)

#print_confusion_matrix()
print("Adagra optimizer with incremental batch size ")
```

```
session.run(tf.global_variables_initializer())
for lrx in [x/10 for x in range(5,0,-1)]:
    for b in range(1,1000,100):
        session.run(tf.global_variables_initializer())
        for i in [1,9,990]:
            optimize(adagrad_optimizer,num_iterations= i,learning_rate =
lrx,batch_size=100)
            print_accuracy(i,lrx,b)

print_confusion_matrix()
```

Results

Run > python lesson7.1.py

```
Gradient descent optimizer

Accuracy : 32.9% with 1 iterations, 0.50 learning rate and 100 batch size
Accuracy : 75.4% with 9 iterations, 0.50 learning rate and 100 batch size
Accuracy : 91.5% with 990 iterations, 0.50 learning rate and 100 batch size
Accuracy : 17.6% with 1 iterations, 0.40 learning rate and 100 batch size
Accuracy : 77.6% with 9 iterations, 0.40 learning rate and 100 batch size
Accuracy : 91.7% with 990 iterations, 0.40 learning rate and 100 batch size
Accuracy : 33.4% with 1 iterations, 0.30 learning rate and 100 batch size
Accuracy : 79.0% with 9 iterations, 0.30 learning rate and 100 batch size
Accuracy : 91.8% with 990 iterations, 0.30 learning rate and 100 batch size
Accuracy : 31.4% with 1 iterations, 0.20 learning rate and 100 batch size
Accuracy : 76.9% with 9 iterations, 0.20 learning rate and 100 batch size
Accuracy : 91.6% with 990 iterations, 0.20 learning rate and 100 batch size
Accuracy : 46.8% with 1 iterations, 0.10 learning rate and 100 batch size
Accuracy : 74.2% with 9 iterations, 0.10 learning rate and 100 batch size
Accuracy : 91.0% with 990 iterations, 0.10 learning rate and 100 batch size

Adagra optimizer

Accuracy : 30.0% with 1 iterations, 0.50 learning rate and 100 batch size
Accuracy : 49.3% with 9 iterations, 0.50 learning rate and 100 batch size
Accuracy : 91.3% with 990 iterations, 0.50 learning rate and 100 batch size
Accuracy : 33.6% with 1 iterations, 0.40 learning rate and 100 batch size
Accuracy : 78.1% with 9 iterations, 0.40 learning rate and 100 batch size
Accuracy : 91.4% with 990 iterations, 0.40 learning rate and 100 batch size
Accuracy : 23.8% with 1 iterations, 0.30 learning rate and 100 batch size
Accuracy : 48.8% with 9 iterations, 0.30 learning rate and 100 batch size
Accuracy : 91.7% with 990 iterations, 0.30 learning rate and 100 batch size
```

```
Accuracy : 59.0% with 1 iterations, 0.20 learning rate and 100 batch size
Accuracy : 74.6% with 9 iterations, 0.20 learning rate and 100 batch size
Accuracy : 92.1% with 990 iterations, 0.20 learning rate and 100 batch size
Accuracy : 33.8% with 1 iterations, 0.10 learning rate and 100 batch size
Accuracy : 74.3% with 9 iterations, 0.10 learning rate and 100 batch size
Accuracy : 91.8% with 990 iterations, 0.10 learning rate and 100 batch size
```

Adagra optimizer with incremental batch size

```
Accuracy : 36.9% with 1 iterations, 0.50 learning rate and 1 batch size
Accuracy : 46.8% with 9 iterations, 0.50 learning rate and 1 batch size
Accuracy : 91.9% with 990 iterations, 0.50 learning rate and 1 batch size
Accuracy : 52.7% with 1 iterations, 0.50 learning rate and 101 batch size
Accuracy : 69.7% with 9 iterations, 0.50 learning rate and 101 batch size
Accuracy : 91.5% with 990 iterations, 0.50 learning rate and 101 batch size
Accuracy : 29.0% with 1 iterations, 0.50 learning rate and 201 batch size
Accuracy : 67.9% with 9 iterations, 0.50 learning rate and 201 batch size
Accuracy : 92.0% with 990 iterations, 0.50 learning rate and 201 batch size
Accuracy : 17.0% with 1 iterations, 0.50 learning rate and 301 batch size
Accuracy : 60.2% with 9 iterations, 0.50 learning rate and 301 batch size
Accuracy : 91.0% with 990 iterations, 0.50 learning rate and 301 batch size
Accuracy : 22.4% with 1 iterations, 0.50 learning rate and 401 batch size
Accuracy : 66.7% with 9 iterations, 0.50 learning rate and 401 batch size
Accuracy : 91.4% with 990 iterations, 0.50 learning rate and 401 batch size
Accuracy : 32.2% with 1 iterations, 0.50 learning rate and 501 batch size
Accuracy : 74.0% with 9 iterations, 0.50 learning rate and 501 batch size
Accuracy : 91.2% with 990 iterations, 0.50 learning rate and 501 batch size
Accuracy : 11.8% with 1 iterations, 0.50 learning rate and 601 batch size
Accuracy : 60.3% with 9 iterations, 0.50 learning rate and 601 batch size
Accuracy : 91.6% with 990 iterations, 0.50 learning rate and 601 batch size
Accuracy : 29.9% with 1 iterations, 0.50 learning rate and 701 batch size
Accuracy : 64.6% with 9 iterations, 0.50 learning rate and 701 batch size
Accuracy : 92.2% with 990 iterations, 0.50 learning rate and 701 batch size
Accuracy : 30.4% with 1 iterations, 0.50 learning rate and 801 batch size
Accuracy : 67.4% with 9 iterations, 0.50 learning rate and 801 batch size
Accuracy : 91.0% with 990 iterations, 0.50 learning rate and 801 batch size
Accuracy : 51.8% with 1 iterations, 0.50 learning rate and 901 batch size
Accuracy : 63.7% with 9 iterations, 0.50 learning rate and 901 batch size
Accuracy : 91.4% with 990 iterations, 0.50 learning rate and 901 batch size
Accuracy : 15.8% with 1 iterations, 0.40 learning rate and 1 batch size
Accuracy : 63.2% with 9 iterations, 0.40 learning rate and 1 batch size
Accuracy : 91.8% with 990 iterations, 0.40 learning rate and 1 batch size
Accuracy : 37.6% with 1 iterations, 0.40 learning rate and 101 batch size
Accuracy : 64.1% with 9 iterations, 0.40 learning rate and 101 batch size
Accuracy : 92.1% with 990 iterations, 0.40 learning rate and 101 batch size
Accuracy : 39.3% with 1 iterations, 0.40 learning rate and 201 batch size
Accuracy : 62.9% with 9 iterations, 0.40 learning rate and 201 batch size
Accuracy : 91.9% with 990 iterations, 0.40 learning rate and 201 batch size
```

```
Accuracy : 41.3% with 1 iterations, 0.40 learning rate and 301 batch size
Accuracy : 67.6% with 9 iterations, 0.40 learning rate and 301 batch size
Accuracy : 91.2% with 990 iterations, 0.40 learning rate and 301 batch size
Accuracy : 37.0% with 1 iterations, 0.40 learning rate and 401 batch size
Accuracy : 80.3% with 9 iterations, 0.40 learning rate and 401 batch size
Accuracy : 91.5% with 990 iterations, 0.40 learning rate and 401 batch size
Accuracy : 24.2% with 1 iterations, 0.40 learning rate and 501 batch size
Accuracy : 71.3% with 9 iterations, 0.40 learning rate and 501 batch size
Accuracy : 92.2% with 990 iterations, 0.40 learning rate and 501 batch size
Accuracy : 15.1% with 1 iterations, 0.40 learning rate and 601 batch size
Accuracy : 69.1% with 9 iterations, 0.40 learning rate and 601 batch size
Accuracy : 92.1% with 990 iterations, 0.40 learning rate and 601 batch size
Accuracy : 11.0% with 1 iterations, 0.40 learning rate and 701 batch size
Accuracy : 79.1% with 9 iterations, 0.40 learning rate and 701 batch size
Accuracy : 92.0% with 990 iterations, 0.40 learning rate and 701 batch size
Accuracy : 40.1% with 1 iterations, 0.40 learning rate and 801 batch size
Accuracy : 64.2% with 9 iterations, 0.40 learning rate and 801 batch size
Accuracy : 91.4% with 990 iterations, 0.40 learning rate and 801 batch size
Accuracy : 35.9% with 1 iterations, 0.40 learning rate and 901 batch size
Accuracy : 71.1% with 9 iterations, 0.40 learning rate and 901 batch size
Accuracy : 92.0% with 990 iterations, 0.40 learning rate and 901 batch size
Accuracy : 29.2% with 1 iterations, 0.30 learning rate and 1 batch size
Accuracy : 71.6% with 9 iterations, 0.30 learning rate and 1 batch size
Accuracy : 91.9% with 990 iterations, 0.30 learning rate and 1 batch size
Accuracy : 29.4% with 1 iterations, 0.30 learning rate and 101 batch size
Accuracy : 66.5% with 9 iterations, 0.30 learning rate and 101 batch size
Accuracy : 91.9% with 990 iterations, 0.30 learning rate and 101 batch size
Accuracy : 16.0% with 1 iterations, 0.30 learning rate and 201 batch size
Accuracy : 56.7% with 9 iterations, 0.30 learning rate and 201 batch size
Accuracy : 92.0% with 990 iterations, 0.30 learning rate and 201 batch size
Accuracy : 36.1% with 1 iterations, 0.30 learning rate and 301 batch size
Accuracy : 79.0% with 9 iterations, 0.30 learning rate and 301 batch size
Accuracy : 91.7% with 990 iterations, 0.30 learning rate and 301 batch size
Accuracy : 34.9% with 1 iterations, 0.30 learning rate and 401 batch size
Accuracy : 62.2% with 9 iterations, 0.30 learning rate and 401 batch size
Accuracy : 91.5% with 990 iterations, 0.30 learning rate and 401 batch size
Accuracy : 29.1% with 1 iterations, 0.30 learning rate and 501 batch size
Accuracy : 67.8% with 9 iterations, 0.30 learning rate and 501 batch size
Accuracy : 92.3% with 990 iterations, 0.30 learning rate and 501 batch size
Accuracy : 22.0% with 1 iterations, 0.30 learning rate and 601 batch size
Accuracy : 64.4% with 9 iterations, 0.30 learning rate and 601 batch size
Accuracy : 91.9% with 990 iterations, 0.30 learning rate and 601 batch size
Accuracy : 20.6% with 1 iterations, 0.30 learning rate and 701 batch size
Accuracy : 72.3% with 9 iterations, 0.30 learning rate and 701 batch size
Accuracy : 91.9% with 990 iterations, 0.30 learning rate and 701 batch size
Accuracy : 31.7% with 1 iterations, 0.30 learning rate and 801 batch size
Accuracy : 67.4% with 9 iterations, 0.30 learning rate and 801 batch size
Accuracy : 92.2% with 990 iterations, 0.30 learning rate and 801 batch size
```

```
Accuracy : 24.1% with 1 iterations, 0.30 learning rate and 901 batch size
Accuracy : 74.4% with 9 iterations, 0.30 learning rate and 901 batch size
Accuracy : 91.6% with 990 iterations, 0.30 learning rate and 901 batch size
Accuracy : 34.1% with 1 iterations, 0.20 learning rate and 1 batch size
Accuracy : 71.9% with 9 iterations, 0.20 learning rate and 1 batch size
Accuracy : 92.1% with 990 iterations, 0.20 learning rate and 1 batch size
Accuracy : 36.5% with 1 iterations, 0.20 learning rate and 101 batch size
Accuracy : 78.7% with 9 iterations, 0.20 learning rate and 101 batch size
Accuracy : 91.9% with 990 iterations, 0.20 learning rate and 101 batch size
Accuracy : 35.7% with 1 iterations, 0.20 learning rate and 201 batch size
Accuracy : 71.5% with 9 iterations, 0.20 learning rate and 201 batch size
Accuracy : 92.1% with 990 iterations, 0.20 learning rate and 201 batch size
Accuracy : 43.5% with 1 iterations, 0.20 learning rate and 301 batch size
Accuracy : 78.9% with 9 iterations, 0.20 learning rate and 301 batch size
Accuracy : 92.0% with 990 iterations, 0.20 learning rate and 301 batch size
Accuracy : 24.1% with 1 iterations, 0.20 learning rate and 401 batch size
Accuracy : 74.6% with 9 iterations, 0.20 learning rate and 401 batch size
Accuracy : 92.0% with 990 iterations, 0.20 learning rate and 401 batch size
Accuracy : 43.0% with 1 iterations, 0.20 learning rate and 501 batch size
Accuracy : 68.8% with 9 iterations, 0.20 learning rate and 501 batch size
Accuracy : 91.9% with 990 iterations, 0.20 learning rate and 501 batch size
Accuracy : 28.0% with 1 iterations, 0.20 learning rate and 601 batch size
Accuracy : 68.8% with 9 iterations, 0.20 learning rate and 601 batch size
Accuracy : 91.9% with 990 iterations, 0.20 learning rate and 601 batch size
Accuracy : 42.3% with 1 iterations, 0.20 learning rate and 701 batch size
Accuracy : 78.1% with 9 iterations, 0.20 learning rate and 701 batch size
Accuracy : 92.2% with 990 iterations, 0.20 learning rate and 701 batch size
Accuracy : 30.9% with 1 iterations, 0.20 learning rate and 801 batch size
Accuracy : 68.5% with 9 iterations, 0.20 learning rate and 801 batch size
Accuracy : 92.0% with 990 iterations, 0.20 learning rate and 801 batch size
Accuracy : 37.4% with 1 iterations, 0.20 learning rate and 901 batch size
Accuracy : 74.8% with 9 iterations, 0.20 learning rate and 901 batch size
Accuracy : 92.1% with 990 iterations, 0.20 learning rate and 901 batch size
Accuracy : 26.0% with 1 iterations, 0.10 learning rate and 1 batch size
Accuracy : 70.8% with 9 iterations, 0.10 learning rate and 1 batch size
Accuracy : 91.7% with 990 iterations, 0.10 learning rate and 1 batch size
Accuracy : 22.3% with 1 iterations, 0.10 learning rate and 101 batch size
Accuracy : 79.9% with 9 iterations, 0.10 learning rate and 101 batch size
Accuracy : 91.6% with 990 iterations, 0.10 learning rate and 101 batch size
Accuracy : 33.5% with 1 iterations, 0.10 learning rate and 201 batch size
Accuracy : 77.1% with 9 iterations, 0.10 learning rate and 201 batch size
Accuracy : 91.3% with 990 iterations, 0.10 learning rate and 201 batch size
Accuracy : 36.0% with 1 iterations, 0.10 learning rate and 301 batch size
Accuracy : 75.8% with 9 iterations, 0.10 learning rate and 301 batch size
Accuracy : 91.6% with 990 iterations, 0.10 learning rate and 301 batch size
Accuracy : 31.9% with 1 iterations, 0.10 learning rate and 401 batch size
Accuracy : 78.6% with 9 iterations, 0.10 learning rate and 401 batch size
Accuracy : 91.7% with 990 iterations, 0.10 learning rate and 401 batch size
```

```
Accuracy : 40.8% with 1 iterations, 0.10 learning rate and 501 batch size
Accuracy : 78.7% with 9 iterations, 0.10 learning rate and 501 batch size
Accuracy : 91.6% with 990 iterations, 0.10 learning rate and 501 batch size
Accuracy : 29.1% with 1 iterations, 0.10 learning rate and 601 batch size
Accuracy : 79.6% with 9 iterations, 0.10 learning rate and 601 batch size
Accuracy : 91.6% with 990 iterations, 0.10 learning rate and 601 batch size
Accuracy : 38.7% with 1 iterations, 0.10 learning rate and 701 batch size
Accuracy : 80.3% with 9 iterations, 0.10 learning rate and 701 batch size
Accuracy : 91.8% with 990 iterations, 0.10 learning rate and 701 batch size
Accuracy : 25.3% with 1 iterations, 0.10 learning rate and 801 batch size
Accuracy : 80.4% with 9 iterations, 0.10 learning rate and 801 batch size
Accuracy : 91.6% with 990 iterations, 0.10 learning rate and 801 batch size
Accuracy : 34.6% with 1 iterations, 0.10 learning rate and 901 batch size
Accuracy : 78.0% with 9 iterations, 0.10 learning rate and 901 batch size
Accuracy : 91.6% with 990 iterations, 0.10 learning rate and 901 batch size

Confusion Matrix

[[ 958    0    2    1    0    4   12    1    2    0]
 [   0 1112    2    2    0    3    4    1   11    0]
 [  11    7  911   17   11    1   18   13   33   10]
 [   3    1   23  920    0   24    4    8   13   14]
 [   1    3    3    1  895    0   16    2    7   54]
 [  11    4    2   40    8  758   25    5   29   10]
 [   9    3    3    2   10    8  921    1    1    0]
 [   2   11   24    7    7    0    0  928    1   48]
 [   9   10    8   29    9   32   13   11  832   21]
 [  11    7    1   12   27    8    1   11    5  926]]

The batch size did not seem to have much effect on the accuracy.
```

Problem 2

Code

```
import numpy as np
from scipy import signal as sg
import tensorflow as tf

image = np.genfromtxt('Problem#2-Image.csv', delimiter=',')
filtr = np.genfromtxt('Problem#2-Filter Gaussian.csv',delimiter=",")

sg_full = sg.convolve(image,filtr,"full")
sg_valid = sg.convolve(image,filtr,"valid")

print("SCIPY FULL")
```

```
print(sg_full)
print("SCIPY VALID")
print(sg_valid)

image_list = tf.constant(image)
image = tf.reshape(image_list,[1,image_list.shape[0],image_list.shape[1],1])

filtr_list = tf.constant(filtr)
filtr = tf.reshape(filtr_list,[filtr_list.shape[0],filtr_list.shape[1],1,1])

tf_valid = tf.nn.conv2d(image,filtr,strides=[1,1,1,1], padding='VALID')
tf_same = tf.nn.conv2d(image,filtr,strides=[1,1,1,1], padding='SAME')

with tf.Session() as sess:
    sess.run(image)
    sess.run(filtr)
    print("TF SAME")
    print(sess.run(tf_same))
    print("TF VALID")
    print(sess.run(tf_valid))
```

Results

Run > python lesson7.2.py

SCIPY FULL

```
[[  97.  246.  300.  312.  292.  245.  229.  185.   70.]
 [ 293.  704.  787.  808.  818.  733.  644.  459.  162.]
 [ 354.  860.  951.  914.  936.  993.  903.  519.  142.]
 [ 245.  656.  832.  839.  805.  944. 1045.  643.  155.]
 [ 189.  472.  575.  761.  867.  823.  948.  748.  225.]
 [ 262.  631.  587.  641.  871.  726.  629.  560.  209.]
 [ 300.  787.  779.  613.  687.  584.  436.  368.  150.]
 [ 194.  489.  534.  476.  430.  349.  294.  214.   76.]
 [  54.  115.  135.  164.  142.  107.   96.   56.   15.]]
```

SCIPY VALID

```
[[ 951.  914.  936.  993.  903.]
 [ 832.  839.  805.  944. 1045.]
 [ 575.  761.  867.  823.  948.]
 [ 587.  641.  871.  726.  629.]
 [ 779.  613.  687.  584.  436.]]
```

TF SAME

```
[[[[ 704.]
   [ 787.]
   [ 808.]
   [ 818.]
   [ 733.]
   [ 644.]
   [ 459.]]

  [[ 860.]
   [ 951.]
   [ 914.]
   [ 936.]
   [ 993.]
   [ 903.]
   [ 519.]]
  [[ 656.]
   [ 832.]
   [ 839.]
   [ 805.]
   [ 944.]
   [1045.]
   [ 643.]]

  [[ 472.]
   [ 575.]
   [ 761.]
   [ 867.]
   [ 823.]
   [ 948.]
   [ 748.]]

  [[ 631.]
   [ 587.]
   [ 641.]
   [ 871.]
   [ 726.]
   [ 629.]
   [ 560.]]
  [[ 787.]
   [ 779.]
```

```
  [ 613.]
  [ 687.]
  [ 584.]
  [ 436.]
  [ 368.]]

 [[ 489.]
  [ 534.]
  [ 476.]
  [ 430.]
  [ 349.]
  [ 294.]
  [ 214.]]]]

TF VALID

[[[[ 951.]
  [ 914.]
  [ 936.]
  [ 993.]
  [ 903.]]

 [[ 832.]
  [ 839.]
  [ 805.]
  [ 944.]
  [1045.]]
[[ 575.]
  [ 761.]
  [ 867.]
  [ 823.]
  [ 948.]]

 [[ 587.]
  [ 641.]
  [ 871.]
  [ 726.]
  [ 629.]]

 [[ 779.]
  [ 613.]
  [ 687.]
  [ 584.]
```

[436.]]]]

Problem 3

Code

```
import numpy as np
from scipy import signal
from scipy import misc
import matplotlib.pyplot as plt
from PIL import Image
from scipy import ndimage
import sys
import math

im = Image.open('01 Lady.png')
misc.imsave("filtered_lady_start.png",im)

image_gr = im.convert("L")
image_array = np.asarray(image_gr)

filtr1 = np.genfromtxt('filter1.csv',delimiter=",")
grad = signal.convolve2d(image_array,filtr1, mode='same', boundary='symm')
misc.imsave("filtered_lady_1.png",np.absolute(grad))

filtr2 = np.genfromtxt('filter2.csv',delimiter=",")
grad = signal.convolve2d(image_array,filtr2, mode='same', boundary='symm')
misc.imsave("filtered_lady_2.png",np.absolute(grad))

filtr1plus2 = np.genfromtxt('filter2.csv',delimiter=",")
grad = signal.convolve2d(image_array,filtr1+filtr2, mode='same', boundary='symm')
misc.imsave("filtered_lady_1plus2.png",np.absolute(grad))

def make_filter(n):

  filtr = [[1 for i in range(n)] for j in range(n)]
  filtr[math.ceil(n/2)][math.ceil(n/2)] = 1-(n ** 2)
  return filtr

def filter_image(image,filtr_size,name):
    filtr = make_filter(filtr_size)
```

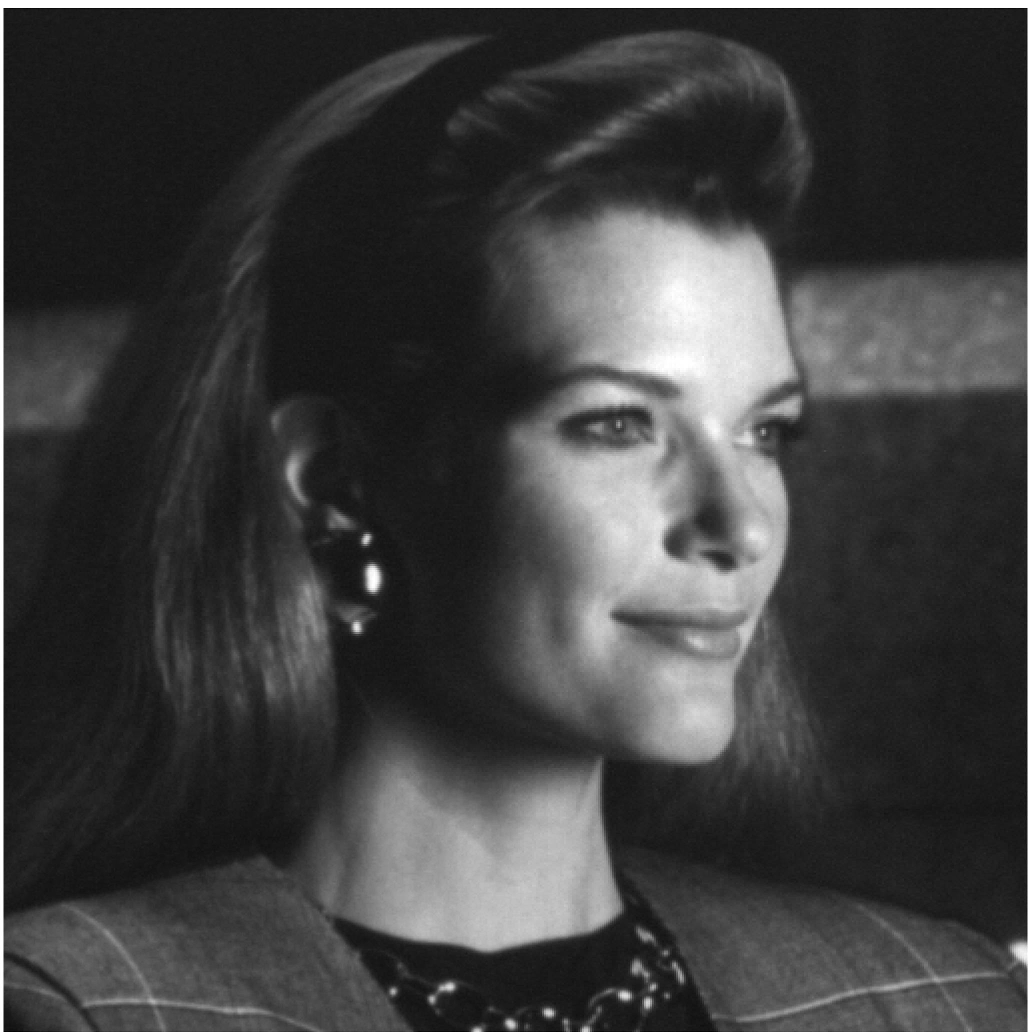
```
    grad = signal.convolve2d(image,filtr, mode='same', boundary='symm')
    misc.imsave(name,np.absolute(grad))

filter_image(image_array,7,"auto_5x5.png")
filter_image(image_array,11,"auto_11x11.png")
filter_image(image_array,15,"auto_15x15.png")
filter_image(image_array,29,"auto_29x29.png")
filter_image(image_array,39,"auto_39x39.png")
```

Results

`Run > python lesson7.3.py`

Lady before filter



Lady after Filter#1+Filter#2

Lady After 5x5

Lady after 15x15

Lady after 29x29

Lady after 39x39