Homework Assignment 1

COGS 181: Neural Networks and Deep Learning

Due: Oct. 6, 2017, 11:59pm
Instructions: Please answer the questions below, attach your code, and insert figures to create a pdf file; submit your file to TED (ted.ucsd.edu) by 11:59pm, 10/06/2017. You may search information online but you will need to write code/find solutions to answer the questions yourself.

Late Policy: %5 of the total points will be deducted on the first day past due. Every 10% of the total points will be deducted for every extra day past due.

System Setup: You are free to choose either pip or anaconda as the package installer. After the installation of one of the installer, type pip/conda install $PACKAGE_NAME in the terminal to install python packages. For more information, see Piazza system setup post.

Grade: ____ out of 100 points

1 (15 points)Basic Matrix Operations Using Numpy

Suppose $\mathbf{A} = \begin{bmatrix} 1 & 3 \\ 5 & 7 \\ 9 & 11 \end{bmatrix}$, $\mathbf{B} = \begin{bmatrix} 1 & -1 \\ -1 & 1 \\ -1 & 0 \end{bmatrix}$, we can convert $\mathbf{A}$ and $\mathbf{B}$ into numpy arrays by

```python
import numpy as np
A = np.array([[1,3],[5,7],[9,11]])
B = np.array([[1,-1],[-1,1],[-1,0]])
```

Numpy package uses np.dot(A,B) to compute the dot product between matrices A and B. Binary operators such as '*','/','+' and '-' compute the element-wise operations between two matrices. Please compute the following: (If the matrix multiplication is not possible, write 'impossible' in your answer)

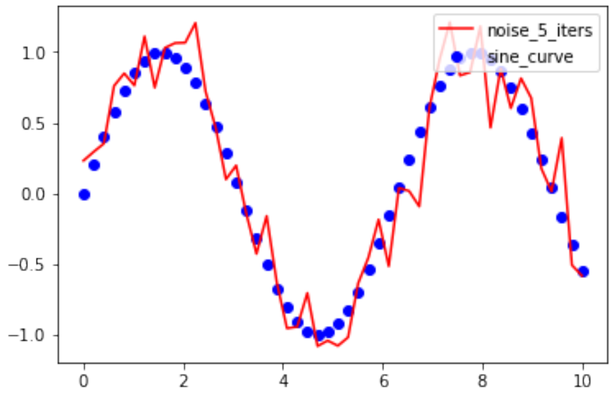
1. $A - B$ array([[0, 4], [6, 6], [10, 11]])
2. $A \circ B$ (element-wise product) array([[1, -3], [-5, 7], [-9, 0]])
3. $A^T B$ array([[-13, 4], [-15, 4]])
4. AB^T array([[-2, 2, -1], [-2, 2, -5], [-2, 2, -9]])
5. AB impossible

2 (20 points)Basic Plots Using Matplotlib

Matplotlib is a very useful library to plot graphs. In later course, you may need to plot your own graphs such as: losses v.s. iterations, accuracies v.s. categories and include those graphs in your report. We will start with a simple exercise. Copy and paste the following code into one notebook cell:

```
import matplotlib.pyplot as plt
%matplotlib inline
np.random.seed(0)
space = np.linspace(0,10,num=50)
sine = np.sin(space)
plt.scatter(space,sine,color='b',label = 'sine_curve')
sine_5 = sine
for i in range(5):
    sine_5 = sine_5 + np.random.normal(scale=0.1,size=50)
plt_sine_5, = plt.plot(space,sine_5,color='r',label = 'noise_5_iters')
plt.legend(loc='upper right')
plt.savefig('./cogs181_Q2.png')
```

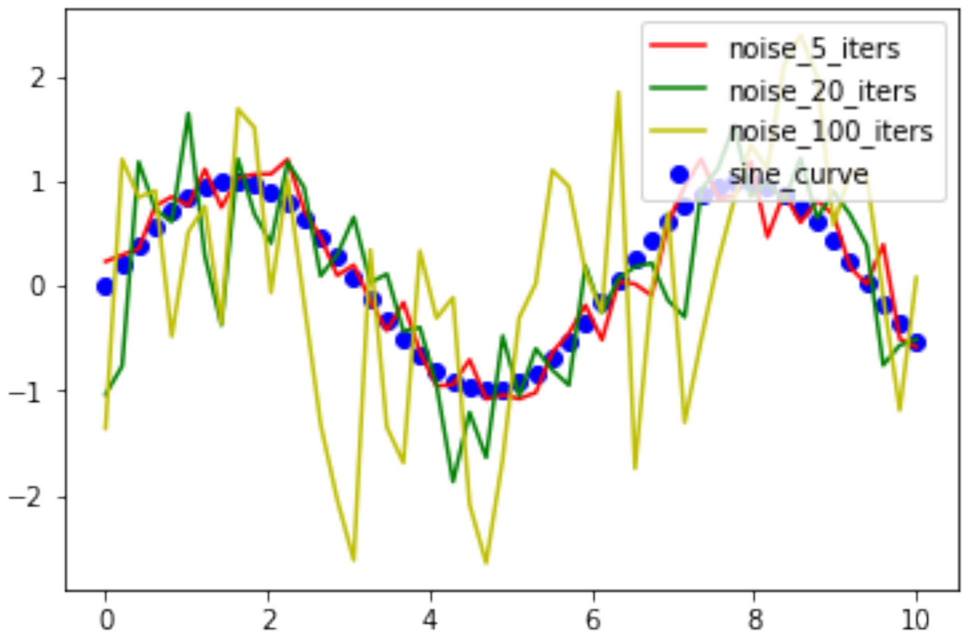
The above code creates a graph with the original sine curve and another curve which 5 iterations of Gaussian noise (0 mean, 0.1 standard deviation) are added to the original sine curve. You should get a graph similar like this:



Edit the code above and apply 20 and 100 iterations of Gaussian noise (0 mean, 0.1 standard deviation) on the original sine curve respectively. Plot all four curves in one graph (including the two curves from the original code) and add the legend of the additional curves and label them as 'noise_20_iters' and 'noise_100_iters'.

```
import matplotlib.pyplot as plt
%matplotlib inline
np.random.seed(0)
space = np.linspace(0,10,num=50)
sine = np.sin(space)
plt.scatter(space,sine,color='b',label = 'sine_curve')
sine_5 = sine
sine_20 = sine
sine_100 = sine
for i in range(5):
    sine_5 = sine_5 + np.random.normal(scale=0.1,size=50)
for i in range(20):
    sine_20 = sine_20 + np.random.normal(scale=0.1,size=50)
for i in range(100):
    sine_100 = sine_100 + np.random.normal(scale=0.1,size=50)
plt_sine_5, = plt.plot(space,sine_5,color='r',label = 'noise_5_iters')
plt_sine_20, = plt.plot(space,sine_20,color='g',label = 'noise_20_iters')
plt_sine_100, = plt.plot(space,sine_100,color='y',label = 'noise_100_iters')

plt.legend(loc='upper right')
plt.savefig('./cogs181_Q2.png')
```



3 (25 points)Basic Image Operations Using Scipy

Scipy library offers a variety of functions for common image processing. In this section you will load the image, resize image and rotate image. First you need to install scipy and download the image ’tabby.jpg’ from our course website.

```
from scipy.misc import imread
img = imread(/path/to/image)
```

Complete the following questions and paste your output in the homework:

1. What is the shape of the image array?(hint: 3 dimensions)
2. Resize the image to 300x300 using scipy.misc.imresize

Besides resize, we are also interested in rotating images and scipy.ndimage provides more image operations than scipy.misc.

```
from scipy.misc import imread
img = imread(/path/to/image)
from scipy.ndimage import rotate
rotated = rotate(img,30,reshape=False,mode='nearest',cval=0)
```

Complete the following questions and paste your output in the homework. All the operations should base on the original image (’img’ in the code):

1. Set the reshape option to True
2. Change the mode to ’constant’, and let ’cval’ be 0
3. Change the mode to ’constant’, and let ’cval’ be 255
4. Change the mode to ’reflect’

4 (15 points) Data and Visualization

The dataset description can be found ***here***.
https://archive.ics.uci.edu/ml/datasets/Iris.
Plot two figures to visualize the dataset using: (1) the first 2 feature dimensions, and (2) the first 3 feature dimensions. Some useful instructions are shown below:
Python:

(1) Import several useful package into Python:

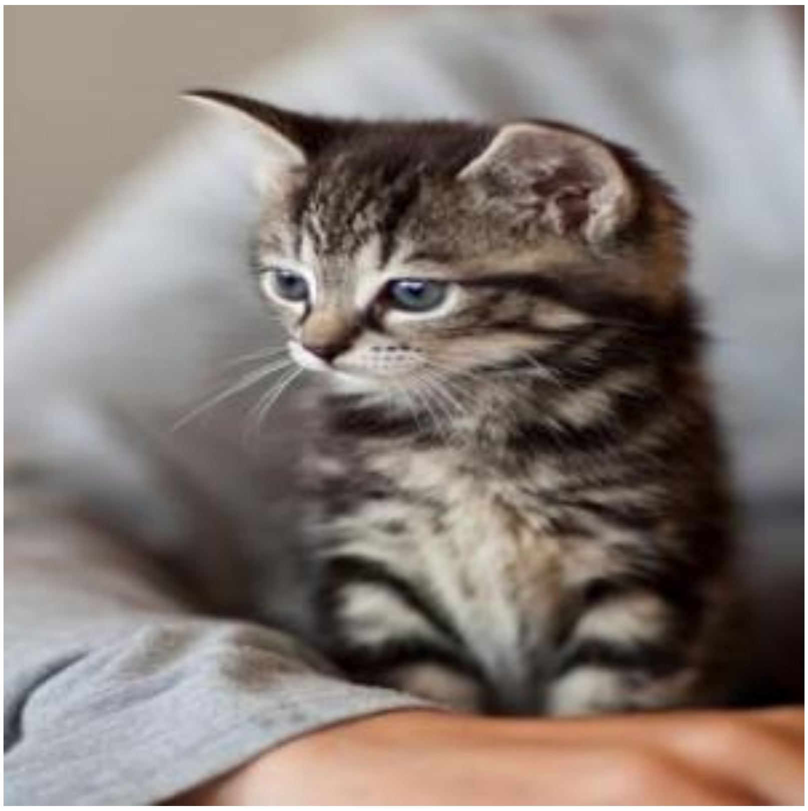
1. What is the shape of the image array?(hint: 3 dimensions)

```
from scipy.misc import imread
img = imread('tabby.jpg')
img.shape
```

Answer: (535, 356, 3)

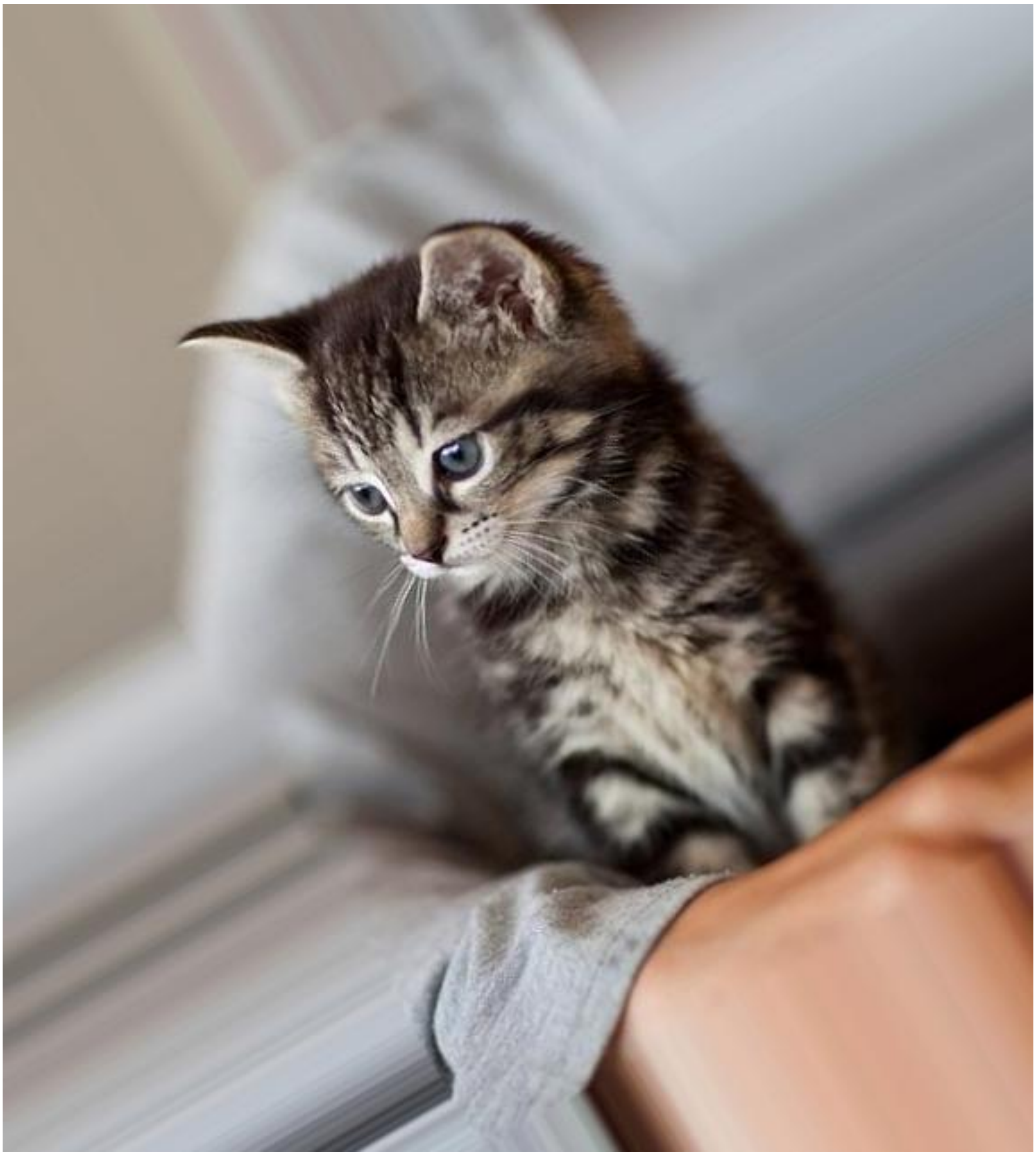
2. Resize the image to 300x300 using scipy.misc.imresize

```
from scipy.misc import imresize,imsave
img_resize = imresize(img,(300,300))
imsave('tabby_resize.jpg',img_resize)
```



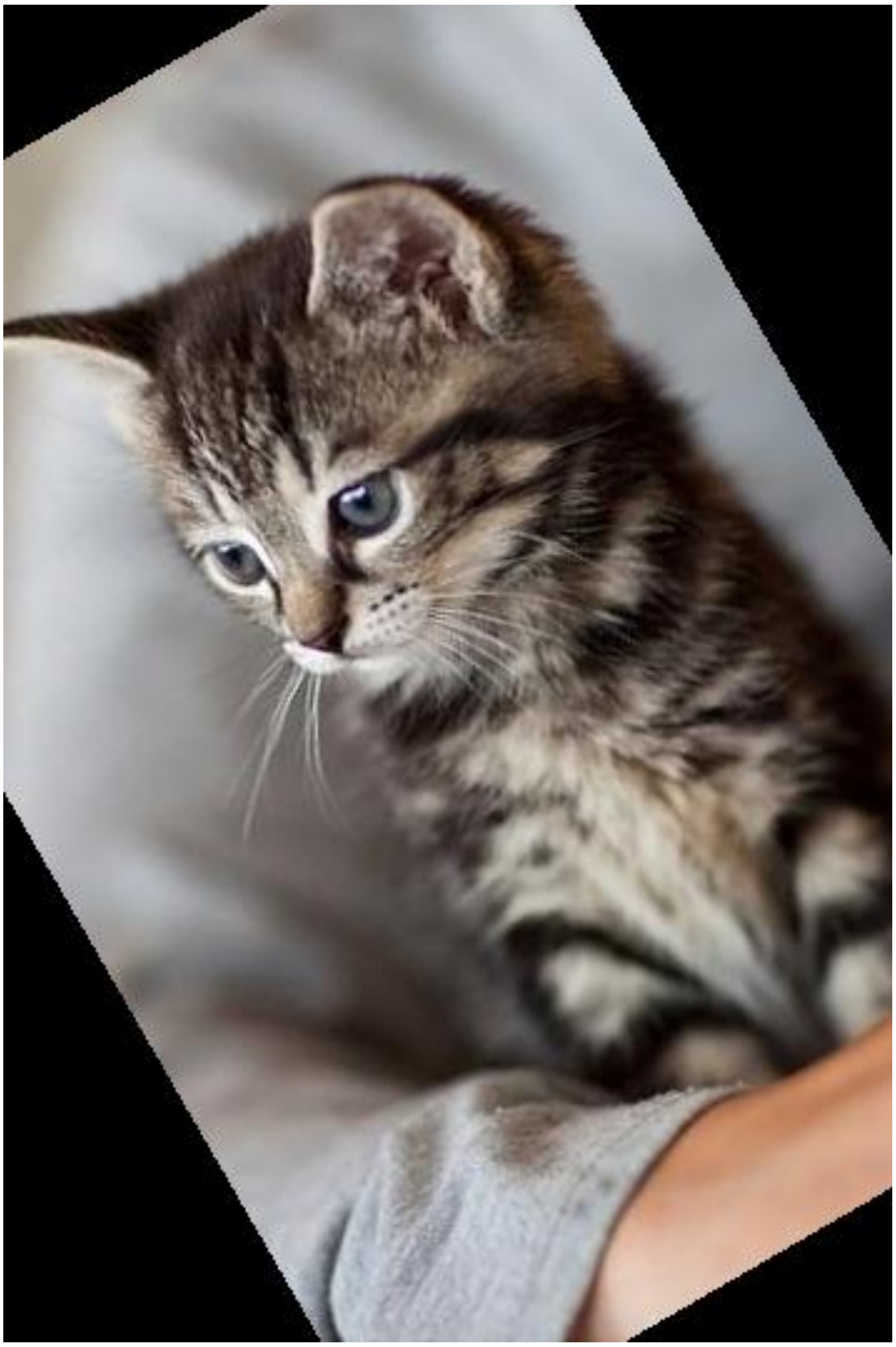
1. Set the reshape option to True

```
from scipy.misc import imread
img = imread('tabby.jpg')
from scipy.ndimage import rotate
rotated = rotate(img,30,reshape=True,mode='nearest',cval=0)
imsave('tabby_rotate.jpg',rotated)
```



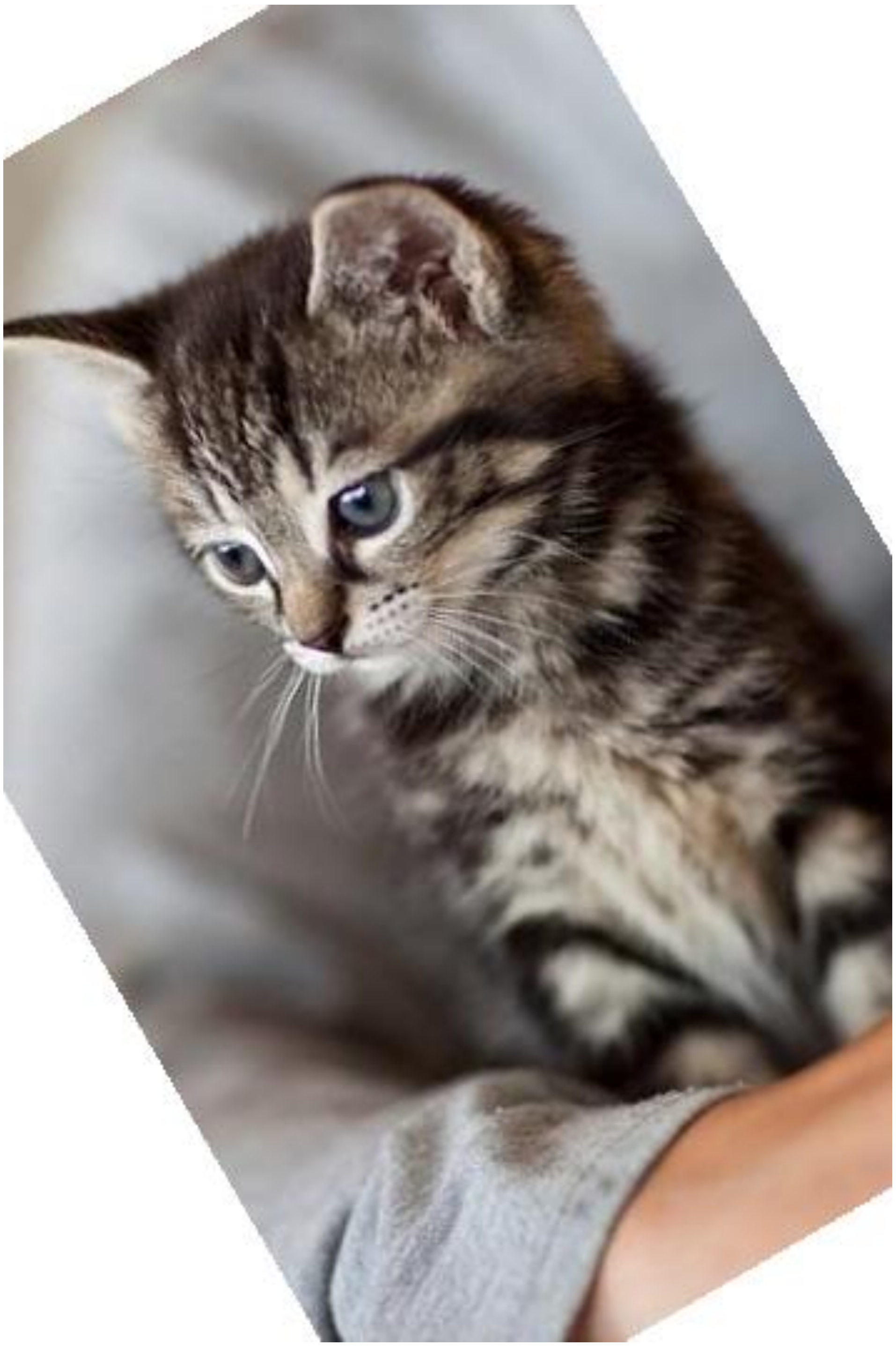
2. Change the mode to 'constant', and let 'cval' be 0

```
from scipy.misc import imread
img = imread('tabby.jpg')
from scipy.ndimage import rotate
rotated = rotate(img,30,reshape=False,mode='constant',cval=0)
imsave('tabby_rotate.jpg',rotated)
```



3. Change the mode to ’constant’, and let ’cval’ be 255

```
from scipy.misc import imread
img = imread('tabby.jpg')
from scipy.ndimage import rotate
rotated = rotate(img,30,reshape=False,mode='constant',cval=255)
imsave('tabby_rotate.jpg',rotated)
```



4. Change the mode to 'reflect'

```
from scipy.misc import imread
img = imread('tabby.jpg')
from scipy.ndimage import rotate
rotated = rotate(img,30,reshape=False,mode='reflect',cval=0)
imsave('tabby_rotate.jpg',rotated)
```

```
import matplotlib.pyplot as plt
from sklearn import datasets
from mpl_toolkits.mplot3d import Axes3D
```

(2) Load Iris dataset into Python:

```
iris = datasets.load_iris()
X = iris.data
Y = iris.target
```

(3) Visualize first 2 feature dimensions:

```
plt.scatter(X[:,0],X[:,1], c=Y, cmap=plt.cm.Paired)
```

(4) Visualize first 3 feature dimensions:

```
fig = plt.figure(1, figsize=(8, 8))
ax = Axes3D(fig, elev=-150, azim=110)
ax.scatter(X[:,0],X[:,1],X[:,2],c=Y,cmap=plt.cm.Paired)
```

5 (10 points) One-hot Encoding

A dataset S is denoted as $S = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_5\}$, where each sample refers to the specification of a car.

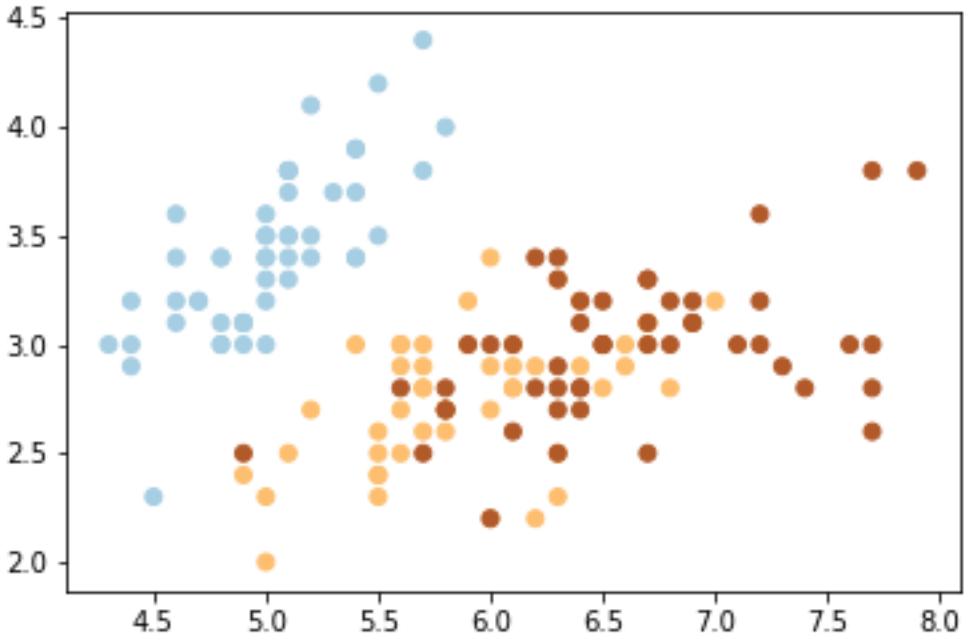
	Length (inch)	Height (inch)	Color
$\mathbf{x}_1$	182.3	62	Silver
$\mathbf{x}_2$	181	66	Blue
$\mathbf{x}_3$	186	56	Red
$\mathbf{x}_4$	179	59	Blue
$\mathbf{x}_5$	182	50	Black

Please represent this dataset S using a matrix and show it below. **Hint**: for the categorical feature, you may use the one-hot encoding strategy; you can choose either a row vector or a column vector to represent each data sample, but please use one consistently for the entire dataset.

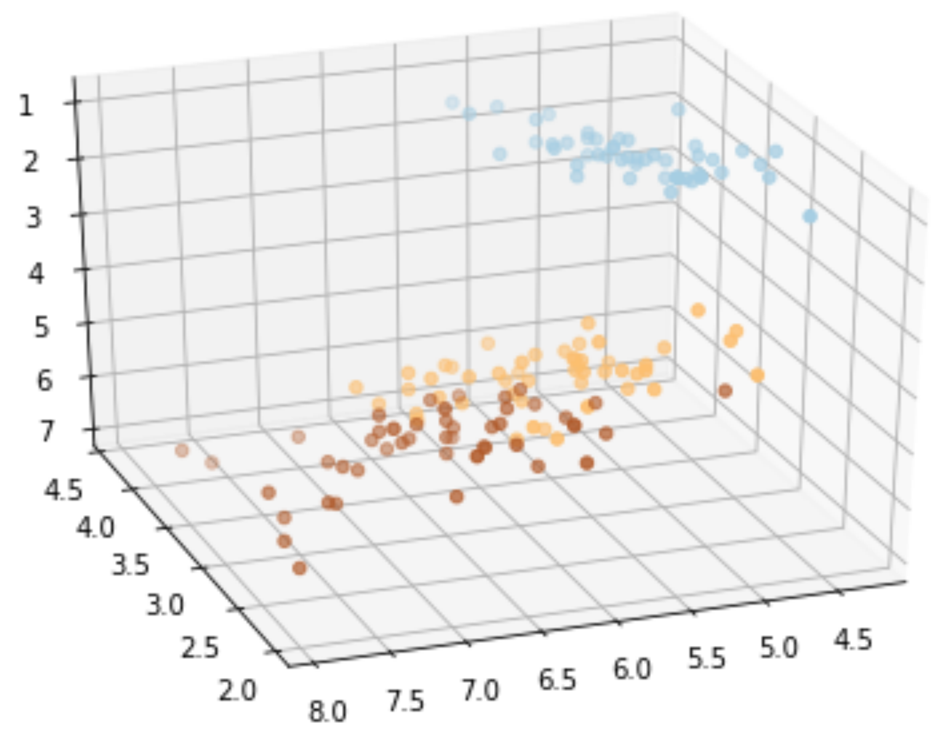
Answer:

```
182.3  62 0 0 0 1
181    66 0 0 1 0
186    56 0 1 0 0
179    59 0 0 1 0
182    50 1 0 0 0
```

Visualize first 2 feature dimensions:



Visualize first 3 feature dimensions:



6 (15 points) Linear regression

Download the data file **data.txt** from the course website: data.txt
and load the data into your environment. The data samples are represented as row vectors; the first column refers to the input (x-axis) and the second column refers to the output (y-axis). We assume that this dataset was generated from a linear model: $y = ax + b$ plus noise, and you need to find the optimal a^* and b^* to fit the data. Now follow the instructions below using the least squre estimation and show the result in your report.

HINT: The closed form solution for the least square estimation is $W = (X^T X)^{-1} X^T Y$.
Python:

(1) Import *matplotlib* and *numpy* package into Python.

```
import matplotlib.pyplot as plt
import numpy as np
```

(2) Load data.

```
data = np.loadtxt('data.txt',dtype='float')
x = data[:,0].reshape(len(data),1)
y = data[:,1].reshape(len(data),1)
```

(3) Plot the data.

```
plt.plot(x,y)
plt.grid()
```

(4) Create the matrix

```
X = np.hstack((np.ones((len(x),1)),np.power(x,1)))
```

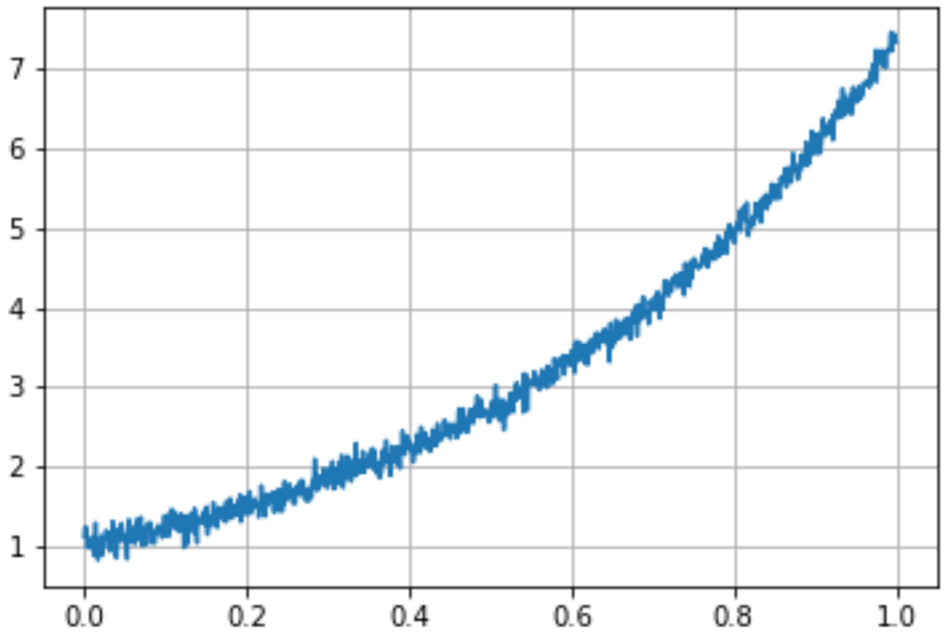
(5) Compute the least squares line over the given data by

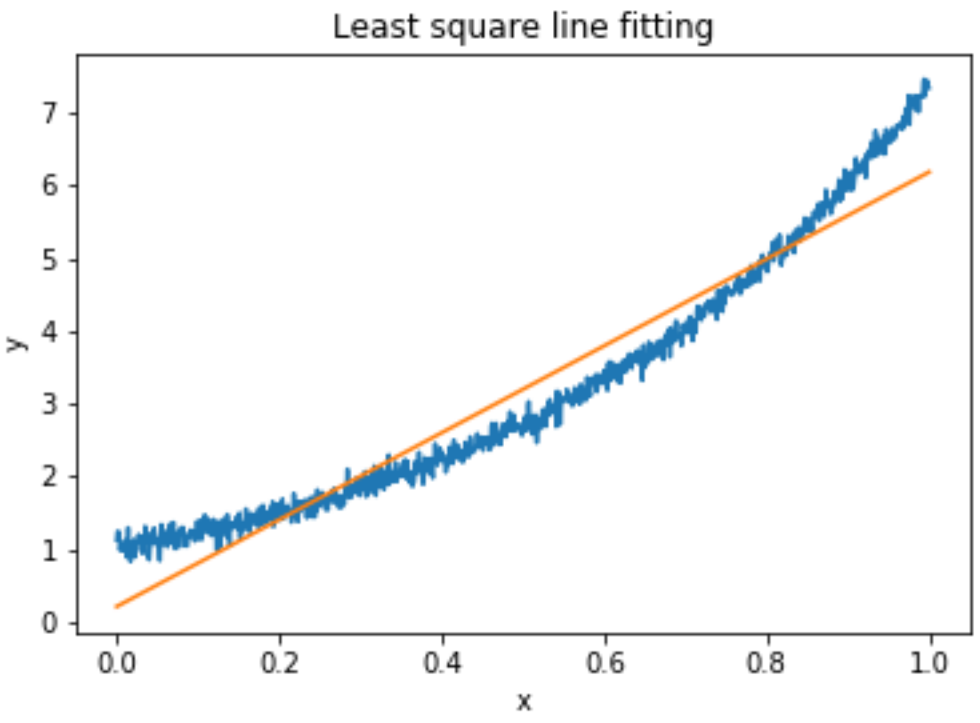
```
X_t = X.transpose((1,0))
sol = np.dot(np.linalg.inv(np.dot(X_t,X)),np.dot(X_t,y))
```

(6) Plot the least square line, and assign title to the figure

```
plt.plot(x,y)
plt.hold(True)
plt.plot(x,sol[0]+sol[1]*x)
plt.title('Least square line fitting')
plt.xlabel('x')
plt.ylabel('y')
```

```
import matplotlib.pyplot as plt
import numpy as np
data = np.loadtxt('data.txt',dtype='float')
x = data[:,0].reshape(len(data),1)
y = data[:,1].reshape(len(data),1)
plt.plot(x,y)
plt.grid()
X = np.hstack((np.ones((len(x),1)),np.power(x,1)))
X_t = X.transpose((1,0))
sol = np.dot(np.linalg.inv(np.dot(X_t,X)),np.dot(X_t,y))
plt.plot(x,y)
plt.plot(x,sol[0]+sol[1]*x)
plt.title('Least square line fitting')
plt.xlabel('x')
plt.ylabel('y')
plt.savefig('./Q6.png')
```




Least square line fitting
y
x
0
1
2
3
4
5
6
7
0.0
0.2
0.4
0.6
0.8
1.0