Homework Assignment 1

COGS 118A: Introduction to Machine Learning I

Due: Oct. 7, 2018, 11:59pm (Pacific Time)

Instructions: Please answer the questions below, attach your code in the document, and insert figures to create **a single PDF file**. Submit it to TritonEd by 11:59pm, 10/7/2018. 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: For this class, please use **Python 3.6** for homework. We highly recommend you to use the Jupyter Notebook http://jupyter.org/ and Anaconda. You can install Anaconda to setup the Jupyter Notebook environment. Most packages are already installed in Anaconda. You can refer to this guide for installation (start reading from `00-Introduction.ipynb` to `04-DataSciencePython.ipynb` will give you great foundation on preparing necessary development environment for this class):
`https://github.com/UCSD-COGS118A/Tutorials`

Grade: ____ out of 100 points

1 (10 points) Matrix Calculus

Several particular derivatives are useful for the course. For matrix $\mathbf{A}$, vector $\mathbf{x}$ and $\mathbf{a}$, we have

- $\dfrac{\partial \mathbf{a}^T\mathbf{x}}{\partial \mathbf{x}} = \dfrac{\partial \mathbf{x}^T\mathbf{a}}{\partial \mathbf{x}} = \mathbf{a}$,
- $\dfrac{\partial \mathbf{x}^T\mathbf{A}\mathbf{x}}{\partial \mathbf{x}} = (\mathbf{A}+\mathbf{A}^T)\mathbf{x}$. If $\mathbf{A}$ is symmetric, $\dfrac{\partial \mathbf{x}^T\mathbf{A}\mathbf{x}}{\partial \mathbf{x}} = 2\mathbf{A}\mathbf{x}$.

Applying the above rules, for

1. $f(\mathbf{x}) = \lambda(1-\mathbf{a}^T\mathbf{x})$, where λ is a constant, derive $\dfrac{\partial f(\mathbf{x})}{\partial \mathbf{x}}$.
2. $f(\mathbf{x}) = \lambda(1-\mathbf{x}^T\mathbf{A}\mathbf{x})$ where $\mathbf{A}$ is a symmetric matrix and λ is a constant, derive $\dfrac{\partial f(\mathbf{x})}{\partial \mathbf{x}}$.

Answer:

1. $\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}}$ for $f(\mathbf{x}) = \lambda(1 - \mathbf{a}^T\mathbf{x})$

$$\begin{aligned}
\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} &= \frac{\partial \lambda(1 - \mathbf{a}^T\mathbf{x})}{\partial \mathbf{x}} \\
&= \frac{\partial \lambda - \lambda \mathbf{a}^T\mathbf{x}}{\partial \mathbf{x}} \\
&= \frac{\partial \lambda}{\partial \mathbf{x}} - \frac{\partial \lambda \mathbf{a}^T\mathbf{x}}{\partial \mathbf{x}} && \text{Watch out for constant term } \lambda \\
&= 0 - \frac{\partial \lambda \mathbf{a}^T\mathbf{x}}{\partial \mathbf{x}} && \text{Watch out the sign} \\
&= -\lambda a^T
\end{aligned}$$

2. $\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}}$ for $f(\mathbf{x}) = \lambda(1 - \mathbf{x}^T\mathbf{A}\mathbf{x})$

$$\begin{aligned}
\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} &= \frac{\partial}{\partial \mathbf{x}} \left(\lambda(1 - \mathbf{x}^T\mathbf{A}\mathbf{x})\right) \\
&= -\lambda \frac{\partial}{\partial \mathbf{x}} (\mathbf{x}^T\mathbf{A}\mathbf{x}) && \text{Note A is a symmetric matrix} \\
&= -2\lambda \mathbf{A}\mathbf{x}
\end{aligned}$$

2 (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)	Make	Color
$\mathbf{x}_1$	183	62	Toyota	Blue
$\mathbf{x}_2$	181	65	BMW	Silver
$\mathbf{x}_3$	182	59	BMW	Red
$\mathbf{x}_4$	179	68	Ford	Blue
$\mathbf{x}_5$	182	53	Toyota	Black

Represent this dataset S using a matrix and show how your one-hot encoding is derived. **Hint**: (1) For the categorical features, you may use the one-hot encoding strategy. (2) You may choose either a row vector or a column vector to represent each data sample in your result. If you use a row vector to represent each data sample, the shape of the result matrix should be 5×9.

Answer:

In one-hot encoding, use the 3rd, 4th, and 5th columns to represent Toyota, BMW, and Ford respectively, and use the 6th, 7th, 8th, and 9th represent Blue, Silver, Red, and Black respectively.

$$\begin{bmatrix}
183 & 62 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
181 & 65 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\
182 & 59 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\
179 & 68 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
182 & 53 & 1 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}$$

3 (25 points) Basic Matrix Operations Using NumPy

Suppose $A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$, $B = \begin{bmatrix} 1 & -1 \\ -1 & 1 \\ 1 & -1 \end{bmatrix}$. We can convert A and B into NumPy arrays by

```
import numpy as np
A = np.array([[1,2], [3,4], [5,6]])
B = np.array([[1,-1], [-1,1], [1,-1]])
```

`np.dot(A,B)` or `A.dot(B)` in NumPy package compute the dot product between matrices A and B (Strictly speaking, it is equivalent to matrix multiplication for 2-D arrays, and inner product of vectors for 1-D arrays). 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' as your answer)

1.
   ```
   print(A+B)
   ```

   $$A + B = \begin{bmatrix} 2 & 1 \\ 2 & 5 \\ 6 & 5 \end{bmatrix}$$

2.
   ```
   print(np.multiple(A,B))
   ```

   $$A \circ B = \begin{bmatrix} 1 & -2 \\ -3 & 4 \\ 5 & -6 \end{bmatrix}$$ ('$\circ$' operator means element-wise product or Hadamard product)

3.
   ```
   print(np.dot(np.transpose(A),B))
   ```

   $$A^T B = \begin{bmatrix} 3 & -3 \\ 4 & -4 \end{bmatrix}$$

4.
   ```
   print(np.dot(A, np.transpose(B)))
   ```

   $$AB^T = \begin{bmatrix} -1 & 1 & -1 \\ -1 & 1 & -1 \\ -1 & 1 & -1 \end{bmatrix}$$

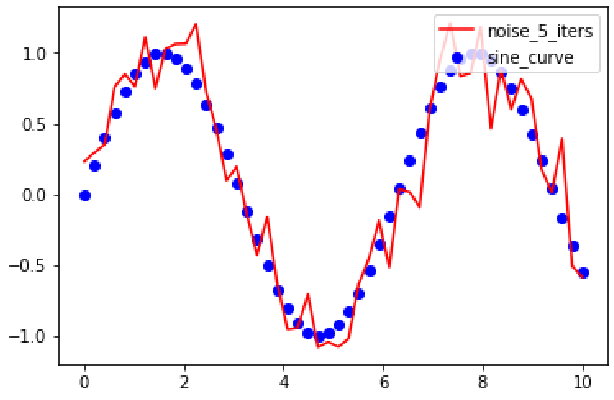
5. AB impossible

4 (10 points) Basic Plots Using Matplotlib

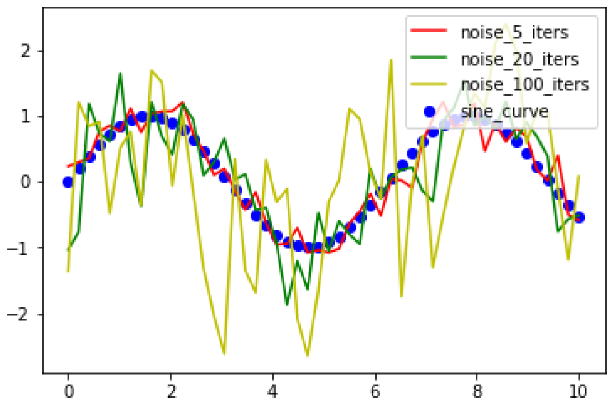
Matplotlib is a very useful library to plot graphs. Later in the course, you may need to plot your own graphs in your report such as: loss vs. iteration, accuracy vs. category. We will start with a simple exercise. Copy and paste the following code into one notebook cell:

```python
import numpy as np
import matplotlib.pyplot as plt
np.random.seed(0)
space = np.linspace(0, 10, num = 50)
sine = np.sin(space)
sine_5 = sine
for i in range(5):
    sine_5 = sine_5 + np.random.normal(scale = 0.1, size = 50)
plt.scatter(space, sine, color = 'b', label = 'sine_curve')
plt.plot(space, sine_5, color = 'r', label = 'noise_5_iters')
plt.legend(loc = 'upper right')
plt.savefig('./Q4.png')
plt.show()
```

The above code creates a graph with the original sine curve and another curve where 5 iterations of Gaussian noise (0 mean, 0.1 standard deviation) are added to the original sine curve. You should get a graph similar to 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. Plot the two new curves you obtained as well as the two curves from the original code. Add the legend of the additional curves and label them as `noise_20_iters` and `noise_100_iters`.



5 (10 points) Basic Image Operations Using Matplotlib

Matplotlib library also offers a variety of functions for common image visualzation. In this problem, you will need to load the image into python and display it. Download the image `cat.jpg` from the course website, and write code to load and display the image.

```python
import matplotlib.pyplot as plt
img = plt.imread("/path/to/image")     # The img here is a NumPy array.
plt.imshow(img)
plt.show()
```

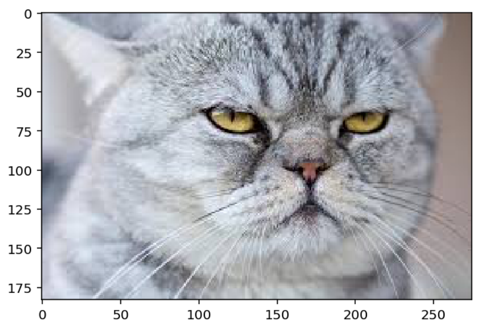
Complete the following questions and paste your output in the homework:

1. Display the `cat.jpg` using the code you write.

2. What is the shape of the image array?
   (hint: it is 3-dimensional and you can use `A.shape` to get the shape of array A)

Answer:

1. Load and show the image:



2. Shape: (183, 275, 3).

6 (10 points) Data and Visualization

Download the Iris Data Set from UCI Machine Learning Repository (click ***here***). The description of the dataset can be found at `https://archive.ics.uci.edu/ml/datasets/Iris`. Plot two figures to visualize the following of the dataset:

1. First 2 feature dimensions.

2. First 3 feature dimensions.

Some useful instructions are shown below:

- Import several useful packages into Python:

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

- Load Iris dataset into Python:

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

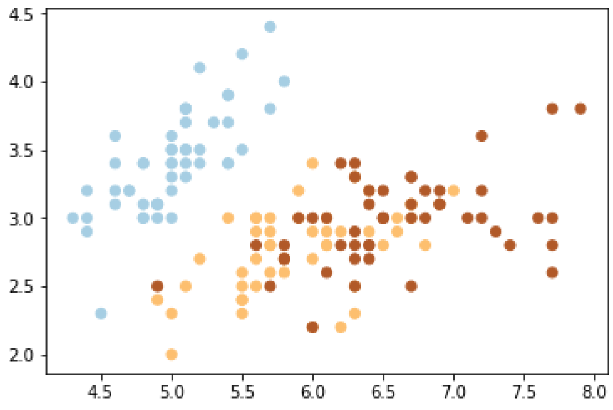


Figure 1: First 2 features.

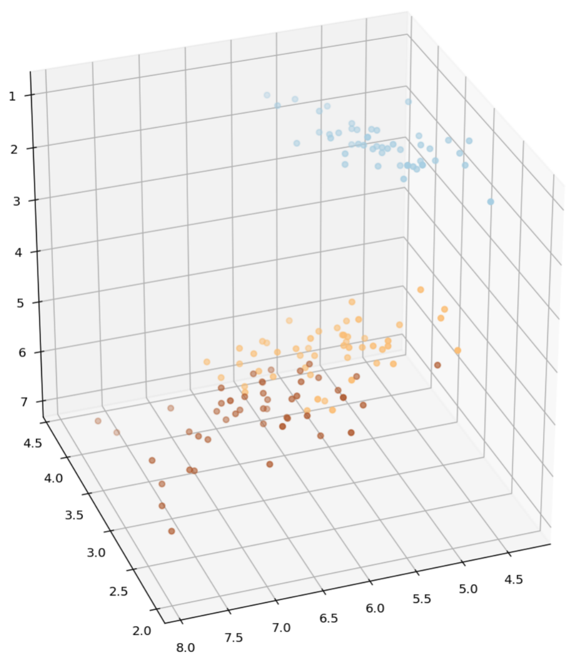


Figure 2: First 3 features.

7 (25 points) Data Manipulation

We have already had a glimpse of the Iris dataset in Question 6. In this question, we will still use the Iris dataset. In fact, you can see the shape of array X is (150, 4) by running `X.shape`, which means it contains 150 data points and 4 features per data point. Here, we will calculate some measures of the array X and perform some basic data manipulation:

1. Show the first 3 features of the first 5 data points (i.e. first 3 columns and first 5 rows) of array X. (You can print the 5×3 array).

2. Calculate the mean and the variance of the 3rd feature (the 3rd column) of array X.

3. Perform a linear projection on the 4 features of all data points using the weight vector $\mathbf{w}$, where $\mathbf{w} = (1, 2, 3, 4)$. You can do so by calculating a dot product between the 4 features of all data points and the weight vector $\mathbf{w}$. The shape of projected data points should be (150, 1) or (150,), depending on the weight vector is regarded as a matrix or a vector. Calculate the mean of the projected data points.
   (hint: You can calculate the projection with a single line code using `np.dot`, but you should be careful with the dimension matching for the dot product)

4. Randomly sample 4 data points (rows) of array X by randomly choosing the row indices. Show the indices and the sampled data points.

5. Add one more feature (one more column) to the array X after the last feature. The values of the added feature for all data points are constant 1. Show the first data point (first row) of the new array.

6. Add one more data point (one more row) to the array X after the last data point. The value of the added data point is the same as the first data point. Show the first feature (first column) of the new array.

Some useful instructions are shown below:

- Get a row or a column of the array X:

```
print X[0]          # Print the first row of array X.
print X[:, 0]       # Print the first column of array X.
                    # ':' here means all rows and '0' means column 0.
```

- Get part of the array:

```
print X[3:5, 1:3] # Print 4th and 5th rows, 2nd and 3rd columns.
print X[:3, :2]   # Print first 3 rows, first 2 columns.
```

Answer:

1.
```
    import matplotlib.pyplot as plt
    from sklearn import datasets
    from mpl_toolkits.mplot3d import Axes3D
    iris = datasets.load_iris()
    X = iris.data
    Y = iris.target
    print(X[:5, :3])
```

```
[[ 5.1  3.5  1.4]
 [ 4.9  3.   1.4]
 [ 4.7  3.2  1.3]
 [ 4.6  3.1  1.5]
 [ 5.   3.6  1.4]]
```

2.
```
    mean = np.mean(X[:,2])
    var = np.var(X[:,2])
```

Mean = 3.75866666667, Variance = 3.09242488889

3.
```
    w = [1,2,3,4]
    print(np.mean(np.dot(X, np.transpose(w))))
```

Mean = 28.022

4. The answer is not unique, but should match the index. For example:

```
rand = np.random.randint(0,150, size=4)
print(X[rand, :])
```

```
[141  13  58 123]
[[ 6.9  3.1  5.1  2.3]
 [ 4.3  3.   1.1  0.1]
 [ 6.6  2.9  4.6  1.3]
 [ 6.3  2.7  4.9  1.8]]
```

5.
```
ones = np.ones(50,1)
new_X = np.hstack((X, ones))
print(new_X[0,:])
```

```
[ 5.1  3.5  1.4  0.2  1. ]
```

6.
```
new_X = np.vstack((X, X[0,:]))
print(new_X[:,0])
```

Output omitted.