Assignment 1

Configuration: environment (Datahub or Anaconda)? Runs the practice assignment first?

To set up the environment locally

- install Anaconda on your laptop (see their official website)
- download/pull everything from the assignment repository ([here][https://github.com/COGS118A/AssignmentNotebooks_SP23]). Make sure you have the yml file `cogs118a_environment.yml` .
- open your terminal and go to where the directory that contains the assignment.
- run `conda env create -f cogs118a cogs118a_environment.yml` in the terminal

To open and run a notebook

- open the terminal, go to your workspace for the assignment
- run `conda activate cogs118a`
- run `jupyter notebook`. A browser should prompt up
- after you've finished, close the browser, go to the terminal and hit the key command+ctrl to end the session

Q1: Latex? Derivative of matrices?

Q2: Give all possible solutions.

Q3:

e.g. a houses dataset -- to predict the **price** based on a set of features...

	Sizes (sq ft)	District	Property type	#Bedrooms	w/wo a basement
1	1076.4	Mira Mesa	Condo	2	wo
2	2368.1	La Jolla	House	3	w
3	1291.7	Ocean Beach	Townhouse	3	wo
4	3229.2	Ocean Beach	Townhouse	4	wo
5	753.4	Mira Mesa	Condo	1	wo
6	5382.1	La Jolla	House	6	w

Or you are writing a calorie calculator for your cats?

	Breed	Weight (kg)	Age	Allergy
Luna	Egyptian Mau	6	12	Alcohol, Milk, Egg
Chessur	Cheshire	30	158	Milk
Snowball	American shorthair	7.5	5	Alcohol

you don't need to provide implementation, but you may find the follows helpful in the future...

`sklearn.preprocessing.OneHotEncoder` (link)

Example (from their documentations): what is this code snippet doing?

```
>>> from sklearn.preprocessing import OneHotEncoder
>>> enc = OneHotEncoder(handle_unknown='ignore')
>>> X = [['red', 1], ['green', 3], ['green', 2]] # how many features?
>>> enc.fit(X)
OneHotEncoder(handle_unknown='ignore')
>>> enc.categories_
[array(['green', 'red'], dtype=object), array([1, 2, 3], dtype=object)]
>>> enc.transform([['green', 1], ['red', 4]]).toarray()
array([[1., 0., 1., 0., 0.],
       [0., 1., 0., 0., 0.]])
>>> enc.inverse_transform([[0, 1, 1, 0, 0], [0, 0, 0, 1, 0]])
array([['red', 1],
       [None, 2]], dtype=object)
>>> enc.get_feature_names_out(['color', 'group'])
array(['color_green', 'color_red', 'group_1', 'group_2', 'group_3'], ...)
```

- Q: what is `handle_unknown`? (`error`, `ignore`, `infrequent_if_exist`)
- Q: explore `OrdinalEncoder`

Q4-Q5: Maths recap. (Why do we care about 'min' or 'max' or 'differentiable'?)

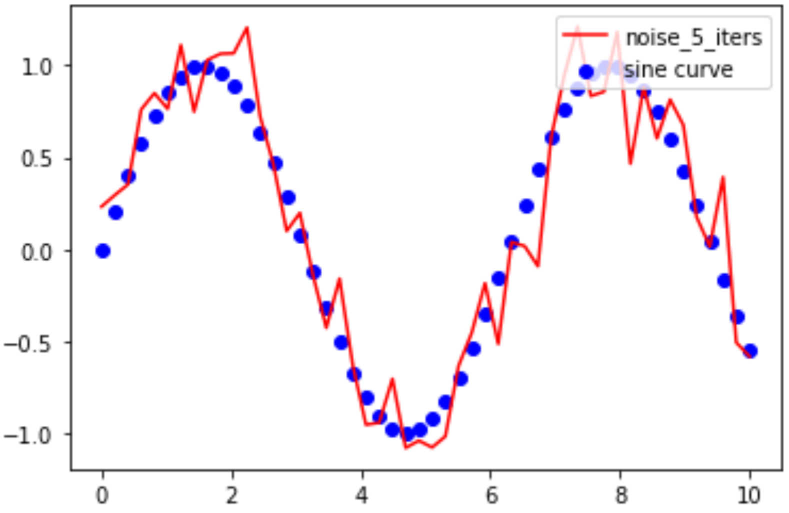
Q6: understand the code

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

# make the random generanator pseudo-random so that the output is reproducible.
np.random.seed(0)
# discretize the range of x ([0, 10]) into 50 data points, with both ends included
space = np.linspace(0, 10, num = 50)
sine = np.sin(space)
sine_5 = sine
```

```
for i in range(5):
    # the noise is of normal distribution
    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')
# generate legends (annotation for plots)
# and place it at the top-right corner of the diagram
plt.legend(loc = 'upper right')
plt.show()
```



Q7:

1. random sampling

```
np.random.randint(0, 5, size=3) # without replacement
np.random.choice(5, 3, replace=True) # with replacement
```

```


np.random.uniform(0, 5, 3) # uniform distribution
np.random.normal(loc=2.5, scale=1, size=3) # normal distribution

# you can also use matrices to define mean (M) and variance (V)
# for output O(i, j), it's sampled from a normal distribution of
# mean M(i,j) and variance V(i, j)
np.random.normal(
    loc=np.arange(6).reshape((3, 2)),
    scale=np.array([1,2,1])[:, None])
```

2. Is it 'safe' to use `vstack` or `hstack`?

```
>>> a = np.random.rand(4, 3)
>>> print(a)
[[0.02960751 0.80757664 0.0461697 ]
 [0.77171261 0.71336406 0.67932393]
 [0.54450577 0.34732819 0.97259812]
 [0.91073756 0.47431493 0.17541997]
>>> b = a[0]
>>> c = np.vstack([a, b])
>>> a[0, 1] = 20
>>> print(c[0, 1])
# 0.80757664 or 20?
>>> b[1] = 30
>>> print(c[0, 1])
# 0.80757664 or 30?
```

Compare to...

```
>>> a = a.tolist()
>>> b = a[0]
>>> c = a+b
>>> a[0][1] = 20
>>> print(c[0][1])
# 0.80757664 or 20?
>>> b[1] = 30
>>> print(c[0][1])
# 0.80757664 or 30?
```

Notes: always check whether an operations is **in-place**

3. matrix multiplication: `np.matmul`

Q: Read the following code snippet, can you 'translate' the code into math?

```
>>> a = np.reshape(np.arange(16), [4, 4])
>>> weight = np.array([1, 3, 4, 2])
>>> print(a)
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]
 [12 13 14 15]]
>>> print(a * weight)
[[ 0  3  8  6]
 [ 4 15 24 14]
 [ 8 27 40 22]
 [12 39 56 30]]
```

```
>>> print(a * weight[:, None])
[[ 0  1  2  3]
 [12 15 18 21]
 [32 36 40 44]
 [24 26 28 30]]
>>> print(np.matmul(a, weight))
[ 17  57  97 137]
>>> print(np.matmul(a, weight[:, None]))
[[ 17]
 [ 57]
 [ 97]
 [137]]
>>> print(np.matmul(weight, a))
[68 78 88 98]
```