APM 598 (Intro. Deep learning): Homework 1 (01/28)

Ex 1. [overfitting]

Some data $\{x_i, y_i\}_{i=1\dots N}$ has been generated from an unknown polynomial $P_*(x)$ (see figure 1 and code page 5 to load the data *'data_ex1.csv'* in Python), i.e.

$$y_i = P_*(x_i) + \varepsilon_i,$$

where ε_i is some 'white noise' or more precisely $\varepsilon \sim \mathcal{N}(0, \sigma^2)$ with σ^2 is the intensity of the noise (also unknown). The goal is to estimate the polynomial P_* and in particular its degree denoted k_* ($\deg(P_*) = k_*$)

a) For each $k = 0 \dots 12$, estimate the polynomial $\hat{P}_k$ of order k that minimizes the mean-square-error (use *polyfit*):

$$\ell(P) = \frac{1}{N} \sum_{i=1}^{N} |y_i - P(x_i)|^2,$$

in other words find the polynomial $\hat{P}_k$ satisfying:

$$\hat{P}_k = \underset{P,\, deg(P) \le k}{\operatorname{argmin}} \ \ell(P).$$

Plot the loss $\ell(\hat{P}_k)$ as a function of k.

b) Split the data-set $\{x_i, y_i\}_{i=1\dots N}$ into training and test sets using (resp.) 80% and 20% of the data. We define two loss functions:

$$\ell_{\text{train}}(P) = \frac{1}{N_{\text{train}}} \sum_{i \text{ in train set}} |y_i - P(x_i)|^2 \quad (1)$$

$$\ell_{\text{test}}(P) = \frac{1}{N_{\text{test}}} \sum_{i \text{ in test set}} |y_i - P(x_i)|^2 \quad (2)$$

where N_{train} and N_{text} denote the number of elements in (resp.) the train and test sets ($N_{\text{train}} + N_{\text{text}} = N$).
Similarly, for each $k = 0 \dots 12$, estimate the polynomial $\widetilde{P}_k$ of order k that minimizes the mean-square-error over the training set:

$$\widetilde{P}_k = \underset{P,\, deg(P) \le k}{\operatorname{argmin}} \ \ell_{\text{train}}(P).$$

and plot the values of the loss functions ℓ_{train} and ℓ_{test} at $\widetilde{P}_k$ for all k.

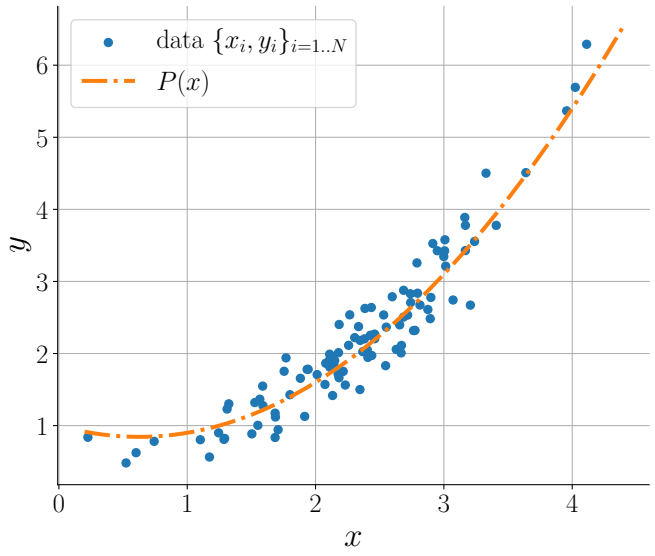

Figure 1: Data points $\{x_i, y_i\}_{i=1..N}$ generated from a polynomial P.

c) Guess what is the order of the polynomial P and give your estimate of its coefficients.

Ex 2. [curse of dimensionality]

The goal of this exercise is to show that in large dimension space (i.e. $\mathbb{R}^d$ with $d \gg 1$) two points are usually *far away*. A collection of points $\{x_i\}_{i=1...N}$ will be sparse in $\mathbb{R}^d$ even if N is large.

Consider the unit cube in $\mathbb{R}^d$:

$$\mathcal{C}_d = \{\mathbf{x} \in \mathbb{R}^d \, : \, 0 \leq x_i \leq 1\} = \underbrace{[0,1] \times \cdots \times [0,1]}_{d \text{ times}}$$

a) Assume the dimension $d = 1$ (i.e. $\mathcal{C}_1 = [0,1]$). Estimate the average square distance between two points, i.e. let X, Y two independent variables distributed uniformly on $\mathcal{C}_1$, compute $\mathbb{E}[|X-Y|^2]$.

b) Generalize and estimate the average square distance between two points in any dimension d.
Plot the evolution of this distance as a function of d.
Hint: if you cannot compute analytically the average square distance $\mathbb{E}[|X-Y|^2]$, *use Monte-Carlo simulations to find a numerical estimate.*

c) Take (any) two similar images of the same size and compute their square Euclidean norm. An example is given in figure 2.

Figure 2: Even-though the two images, denoted x_1 and x_2, are *similar*, the square Euclidean distance (see code page 5) is: $\|x_1 - x_2\|_2^2 = 1117.48$.

Ex 3. [gradient descent]

The goal is to implement and test gradient descent methods. We use linear regression as a toy problem using the data set $\{x_i, y_i\}_{i=1..N}$ from **Ex 1** and we would like to minimize the following loss:

$$\ell(a, b) = \frac{1}{N} \sum_{i=1}^{N} |y_i - (a + bx_i)|^2.$$

a) Compute the gradient of the loss function $\nabla \ell = (\partial_a \ell, \partial_b \ell)$.
Deduce (numerically) the minimum (a_*, b_*) of ℓ.

b) Starting from $(a_0, b_0) = (1.8, 1.4)$ and using the constant learning rate $\eta = .05$, implement the gradient descent algorithm:

$$\begin{pmatrix} a_{n+1} \\ b_{n+1} \end{pmatrix} = \begin{pmatrix} a_n \\ b_n \end{pmatrix} - \eta \nabla \ell(a_n, b_n).$$

Estimate the rate of convergence (i.e. how fast does $\|(a_n, b_n) - (a_*, b_*)\|$ go to zero?).

c) Implement the momentum and Nesterov algorithm, denoting $\theta_n = (a_n, b_n)$,

$$\textbf{momentum} \quad \begin{cases} v_{n+1} &= \gamma v_n + \eta \nabla \ell(\theta_n) \\ \theta_{n+1} &= \theta_n - v_{n+1} \end{cases}$$

$$\textbf{Nesterov} \quad \begin{cases} v_{n+1} &= \gamma v_n + \eta \nabla \ell(\theta_n - \gamma v_n) \\ \theta_{n+1} &= \theta_n - v_{n+1} \end{cases}$$

Estimate the convergence rate for $\gamma = .9$ and $\eta = .05$.

d) Test other gradient descent methods (e.g. Adam, AdaGrad, RMSProp) using class defined in either Pytorch or TensorFlow (but preferably Pytorch...).

Ex 4. [classification]

The goal is to implement a linear classifier for the data-set MNIST-fashion (see also code page 5).

a) Adapt the linear classifier for the MNIST-data set to the new data-base.

b) Train the model using 40 epochs with learning rate $\eta = 10^{-4}$ and the gradient descent method of your choices.
Plot the evolution of the loss at each epoch.

c) Draw the 'template' of each class.

```python
##############################################
##        Exercise 1    (Python)            ##
##############################################
import matplotlib.pyplot as plt
import numpy as np
# get the data
data = np.loadtxt('data_ex1.csv',delimiter=',')
x,y = data[:,0],data[:,1]
# plot
plt.figure(1)
plt.plot(x,y,'o')
plt.xlabel(r'$x$')
plt.ylabel(r'$y$')
plt.show()

##############################################
##        Exercise 2    (Python)            ##
##############################################
import numpy as np
from PIL import Image
# load image
image1 = np.array( Image.open('mona_lisa_1.jpg') )
image2 = np.array( Image.open('mona_lisa_2.jpg') )
# normalize
x1 = image1/255
x2 = image2/255
# square Euclidean distance
np.sum((x1-x2)**2)

##############################################
##        Exercise 4    (Python)            ##
##############################################
from torchvision import datasets
import matplotlib.pyplot as plt
# Download and load
data_collection = datasets.FashionMNIST('data_fashionMNIST', train=True, download=True)
# Illustration
label_fashion = dict([(0,'T-shirt'),(1,'trouser'),(2,'pullover'),(3,'dress'),(4,'coat'),
                      (5,'sandal'),(6,'shirt'),(7,'sneaker'),(8,'bag'),(9,'boot')])
X,y = data_collection.__getitem__(123)
plt.figure(1);plt.clf()
plt.imshow(X)
plt.title("Example of image with label "+label_fashion[y.item()])
plt.show()
```