APM 598: Homework 1 (02/06)

Ex 1. [Overfitting]

Some data $\{x_i, y_i\}_{i=1...N}$ has been generated from an unknown polynomial $P_*(x)$ (see figure 1 and code page 3 to load the data *'data_HW1_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_* (i.e. $k_* = \deg(P_*)$).

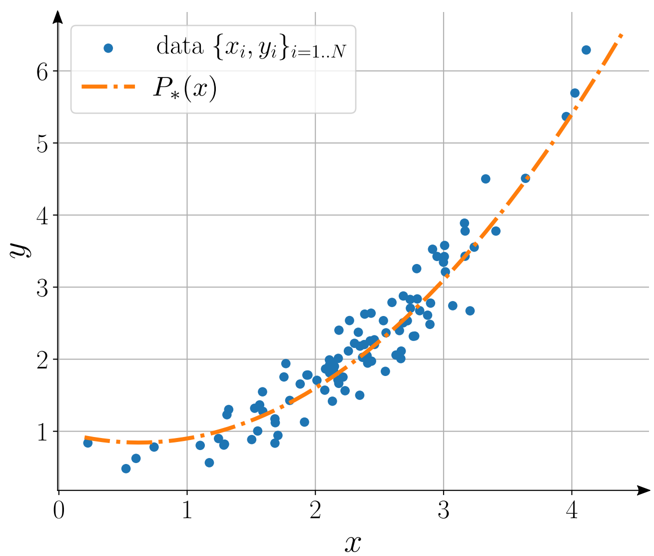

Figure 1: Data points $\{x_i, y_i\}_{i=1..N}$ generated from a polynomial P_*.

a) For each $k = 0 \ldots 12$, 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:

$$\hat{P}_k = \underset{P,\, deg(P) \leq k}{\text{argmin}} \; \ell(P).$$

Plot the loss $\ell(\hat{P}_k)$ as a function of k (i.e. plot the points $\{k, \ell(\hat{P}_k)\}_{k=0...12}$).

b) Split the data-set $\{x_i, y_i\}_{i=1...N}$ 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 $k = 0 \ldots 12$, estimate the polynomial $\widetilde{P}_k$ of order k that minimizes the mean-square-error over the training set:

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

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

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

Ex 2. [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 function:

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

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 $\|(a_n, b_n) - (a_*, b_*)\|$ converge 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 $\eta = .05$.

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

Ex 3. [Classification]

The goal is to implement a linear classifier for the data-set MNIST-fashion (see also code page 3).

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_HW1_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 3     (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()
```