2021/2022

SD-TSIA 211 - TP1

Submitted in fulfillment of the requirements for the

Computer Lab 1 grade

Major: Data Science

Prepared By:

ZEINEDINE Dani
JABER Hadi

Natural Language Processing

Presented to:

Profs. : Anass Aghbalou, Tamim El Ahmad, Olivier Fercoq, Ekhine Irurozki

Table of contents

Abstract

The database consists of propositions on the ecological transition written by French citizens during a consultation led by the government in 2019.The code that we used to pre-process the data is given in Preprocess NLP.ipynb. The main steps of this preprocessing are stemming and computation of the TF-IDF vector representation of the answers. For each word, we compute its frequency over the whole database: its document frequency. Then for each user, we compare the frequency of each term in his vocabulary to the document frequency.

In this computer lab, the objective of the model is to predict the answer to the question “Diriez-vous que votre vie quotidienne est aujourd’hui touchée par le changement climatique ?” using the vocabulary used in the answers to the other questions.

Data is loaded using the given function:

load_data(file_name_matrix='FILE_PATH',file_name_feature_names='FILE_PATH',file_name_labels='FILE_PATH',samples_in_train_set=10000,samples_in_test_set=137562) by returning 4 values eX, y, eX_test and y_test.

Tikhonov regularization

We would like to solve the following logistic regression problem with l_2 regularization:

$$f_1 : (w_0, w) \mapsto \frac{1}{n}\sum_{i=1}^{n} log(1 + e^{-y_i(x_i^T w + w_0)}) + \frac{\rho}{2}\|w\|_2^2$$

Question 3.1

Let us calculate the gradient of f_1 and its Hessian matrix :

$$\frac{\partial f_1}{\partial w_0}(w_0, w) = \frac{1}{n}\sum_{i=1}^{n} \frac{-y_i e^{-y_i(x_i^T w + w_0)}}{1+e^{-y_i(x_i^T w + w_0)}}$$

$$\frac{\partial f_1}{\partial w}(w_0, w) = \frac{1}{n}\sum_{i=1}^{n} \frac{-y_i e^{-y_i(x_i^T w + w_0)}}{1+e^{-y_i(x_i^T w + w_0)}} x_i + \rho w$$

$$\nabla f_1(w_0, w) = \begin{pmatrix} \frac{1}{n}\sum_{i=1}^{n} \frac{-y_i e^{-y_i(x_i^T w + w_0)}}{1+e^{-y_i(x_i^T w + w_0)}} \\ \frac{1}{n}\sum_{i=1}^{n} \frac{-y_i e^{-y_i(x_i^T w + w_0)}}{1+e^{-y_i(x_i^T w + w_0)}} x_i + \rho w \end{pmatrix}$$

$$\frac{\partial^2 f_1}{\partial w_0^2}(w_0, w) = \frac{1}{n}\sum_{i=1}^{n} y_i^2 \frac{e^{y_i(x_i^T w + w_0)}}{(1+e^{y_i(x_i^T w + w_0)})^2}$$

$$\frac{\partial^2 f_1}{\partial w^2}(w_0, w) = \frac{1}{n}\sum_{i=1}^{n} y_i^2 \frac{e^{y_i(x_i^T w + w_0)}}{(1+e^{y_i(x_i^T w + w_0)})^2} x_i x_i^T + \rho I_p$$

$$\frac{\partial^2 f_1}{\partial w_0 \partial w}(w_0, w) = \frac{1}{n}\sum_{i=1}^{n} y_i^2 \frac{e^{y_i(x_i^T w + w_0)}}{(1+e^{y_i(x_i^T w + w_0)})^2} x_i^T$$

$$\frac{\partial^2 f_1}{\partial w \partial w_0}(w_0, w) = \frac{1}{n}\sum_{i=1}^{n} y_i^2 \frac{e^{y_i(x_i^T w + w_0)}}{(1+e^{y_i(x_i^T w + w_0)})^2} x_i$$

Let H be hessian matrix of f_1 : $H(f_1) = \begin{bmatrix} \frac{\partial^2 f_1}{\partial w_0^2} & \frac{\partial^2 f_1}{\partial w_0 \partial w} \\ \frac{\partial^2 f_1}{\partial w \partial w_0} & \frac{\partial^2 f_1}{\partial w^2} \end{bmatrix}$

$$= \begin{bmatrix} \frac{1}{n}\sum_{i=1}^{n} y_i^2 \frac{e^{y_i(x_i^T w + w_0)}}{(1+e^{y_i(x_i^T w + w_0)})^2} & \frac{1}{n}\sum_{i=1}^{n} y_i^2 \frac{e^{y_i(x_i^T w + w_0)}}{(1+e^{y_i(x_i^T w + w_0)})^2} x_i^T \\ \frac{1}{n}\sum_{i=1}^{n} y_i^2 \frac{e^{y_i(x_i^T w + w_0)}}{(1+e^{y_i(x_i^T w + w_0)})^2} x_i & \frac{1}{n}\sum_{i=1}^{n} y_i^2 \frac{e^{y_i(x_i^T w + w_0)}}{(1+e^{y_i(x_i^T w + w_0)})^2} x_i x_i^T + \rho I_p \end{bmatrix}$$

Convexity of f_1 :

$\|.\|_2^2$ is convex.

Let $f : x \mapsto log(1 + e^x)$; $\frac{\partial^2 f}{\partial x^2}(x) = \frac{e^x}{(1+e^x)^2} > 0$ thus f is convex.

In addition, $(w_0, w) \mapsto x_i^T w + w_0$ is an affine function thus convex.

Conclusion, f_1 is convex.

Question 3.2 : Implementation

We start to implement a function returning the value of f_1, its gradient and its hessian matrix at a given input.

```
import numpy as np
import scipy.sparse as sp
from scipy.linalg import norm, solve,inv
from scipy.optimize import check_grad
import matplotlib.pyplot as plt
import time
import math
import seaborn as sns
```

```
#exponential function code:
def my_exp(y,x,w):
    try:
        ans=math.exp(-y*(np.transpose(x).dot(w[1:])+w[0]))
    except OverflowError:
        ans = float('inf')
    return ans
#Our function f_1 code (value, gradient and hessian matrix):
def my_func(w):
    if y.shape[0]!=eX.shape[0]:
        print("Size mismatch!!")
        return -1
    else:
        n=y.shape[0]
        sumf=0
        sumgo=0
        sumgi=[0 for j in range(eX.shape[1])]
        hes11=0
        hes12=[0 for j in range(eX.shape[1])]
        hes22= [[0 for i in range(eX.shape[1])] for j in
range(eX.shape[1])]
```

```
        for i in range(n):
            sumf+= math.log(1+my_exp(y[i],eX[i],w))
            sumgo+= -y[i] * my_exp(y[i],eX[i],w) /
(1+my_exp(y[i],eX[i],w))
            sumgi=np.add(sumgi,(-y[i] * (my_exp(y[i],eX[i],w) /
(1+my_exp(y[i],eX[i],w)))) * eX[i])
            hes11+= (y[i]*y[i]) * ( my_exp(y[i],eX[i],w)) /
((1+my_exp(y[i],eX[i],w))*(1+my_exp(y[i],eX[i],w)))
            hes12=np.add(hes12,((y[i]*y[i]) * (
my_exp(y[i],eX[i],w)) /
((1+my_exp(y[i],eX[i],w))*(1+my_exp(y[i],eX[i],w))) )*eX[i])
            hes22=np.add(hes22,((y[i]*y[i]) * (
my_exp(y[i],eX[i],w)) /
((1+my_exp(y[i],eX[i],w))*(1+my_exp(y[i],eX[i],w)))
)*np.outer(np.transpose(eX[i]),eX[i]))
        sumf = (1/n)*sumf + (1/(2*n))*(norm(w[1:])**2)
        sumgo = (1/n)*sumgo
        sumgi=np.add((1/n)*sumgi,(1/n)*w[1:])
        sumgi = np.concatenate(([sumgo],sumgi))
        hes11=(1/n)*hes11
        hes12=(1/n)*hes12
        hes21=np.transpose(hes12)
        hes22=(1/n)*hes22 + (1/n)*np.eye(eX.shape[1])

        hesu=np.concatenate(([hes11], (hes12)))
        hesl=np.c_[hes21,hes22]
        hes=np.r_[[hesu],hesl]

        return sumf,sumgi,hes
```

Testing our computations using the function check_grad:

```
#code:
n, k = eX.shape
W = np.ones(k+1)
print(check_grad(lambda W: my_func(W)[0], lambda W:
my_func(W)[1], W))
```

By executing the ***check_grad*** function in python we found out that the output result is 1.07731188 × 10^(−5) which is negligible, hence our code is correct.

```
print(check_grad(lambda W: my_func(W)[0], lambda W: my_func(W)[1], W))
```

```
1.0773118895902403e-05
```

Question 3.3: Newton's method

Coding the Newton's method:

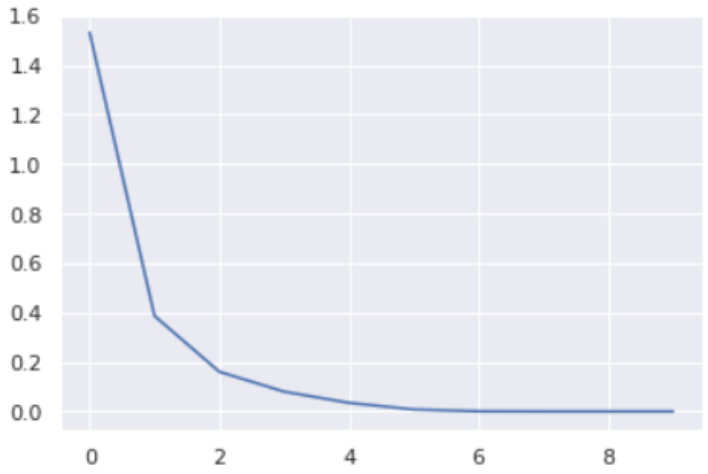
```
def newton(func, start, thresh):
    W = start
    norms=[]
    gradient=func(W)[1]
    current_norm = norm(gradient)
    norms.append(current_norm)
    while current_norm >= thresh:
        a,b,c=func(W)
        W = W - gradient.dot(inv(c))
        gradient = func(W)[1]
        current_norm = norm(gradient)
        norms.append(current_norm)

    sns.set()
    plt.plot(norms)
    plt.show()

    return W
```

Plotting the norm of the gradient as a function of iterations in logarithmic scale with threshold=10^(-10):

```
print(newton(my_func,np.zeros(577),10**-10))
```



Question 3.4

Running Newton's method with initial points as ones instead of zeros, we get infinite values and the method diverges:

```
print(newton(my_func,np.ones(577),10**-10))
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:
---------------------------------------------------------------
ValueError                                Traceback (most rec
<ipython-input-34-8821655997b6> in <module>()
----> 1 print(newton(my_func,np.ones(577),10**-10))

 ——————————————————————————— 2 frames ———————————————————————————

/usr/local/lib/python3.7/dist-packages/numpy/lib/function_bas
    484     if a.dtype.char in typecodes['AllFloat'] and not
    485         raise ValueError(
--> 486             "array must not contain infs or NaNs")
    487     return a
    488

ValueError: array must not contain infs or NaNs
```

Question 3.5

Armijo's line search:

Armijo's line search. This line search is the most famous one. Given $a \in (0,1)$, $b > 0$ and $\beta \in (0,1)$, determine the first integer l such that

$$f(x^+(ba^l)) \leq f(x_k) + \beta \langle \nabla f(x_k), x^+(ba^l) - x_k \rangle$$

Tried to code the algorithm `def armijo(f, start, thresh, a, b, beta)` using 2 nested loops (the first for checking if `gradient_norm >= thresh` and the second for checking the inequality of the algorithm for a specified integer l) but stuck with an infinite loop without getting an output.

Regularization for a sparse model

We are still interested in the logistic regression problem but we change the regularizer by using the norme 1 instead of 2.

$$\min_{w_0 \in \mathbb{R}, w \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} \log\left(1 + \exp(-y_i(x_i^\top w + w_0))\right) + \rho \|w\|_1$$

Question 4.1

The norm 1 (N_1 (x) = $\sum$ | x_i |) is not differentiable everywhere and the singular points are not the same ; on the other hand, the norm 2 (or the norm p for 1 <p <∞) is differentiable everywhere except 0. And since in this part we are using the norme 1 which is not of class C^2 , we can not use Newton's method.

Question 4.2

Writing the objective function as $F_2 = f_2 + g_2$ where f_2 is differentiable and the proximal operator of g_2 is easy to compute:

$F_2 = f_2 + g_2$

$f_2 : (w_0, w) \mapsto \frac{1}{n} \sum_{i=1}^{n} log(1 + e^{-y_i(x_i^T w + w_0)})$

$g_2 : (w_0, w) \mapsto \frac{\rho}{2} \|w\|_1$

$prox_{g_2} : (w_0, w) \mapsto (L_\rho(w_0), L_\rho(w_1), \ldots, L_\rho(w_p))$

where L_ρ is the soft threshold :

$$L_\rho : x \mapsto \begin{cases} x - \rho & \text{if } x > \rho \\ x + \rho & \text{if } x < \rho \\ 0 & \text{if } x \in [-\rho, \rho] \end{cases}$$

We can see that f_2 is the same term of question 1 but without the regularization term, so the gradient will be:

$$\frac{\partial f_2}{\partial w_0}(w_0, w) = \frac{1}{n} \sum_{i=1}^{n} \frac{-y_i e^{-y_i(x_i^T w + w_0)}}{1 + e^{-y_i(x_i^T w + w_0)}}$$

$$\frac{\partial f_2}{\partial w}(w_0, w) = \frac{1}{n} \sum_{i=1}^{n} \frac{-y_i e^{-y_i(x_i^T w + w_0)}}{1 + e^{-y_i(x_i^T w + w_0)}} x_i$$

$$\nabla f_2(w_0, w) = \begin{pmatrix} \frac{1}{n} \sum_{i=1}^{n} \frac{-y_i e^{-y_i(x_i^T w + w_0)}}{1 + e^{-y_i(x_i^T w + w_0)}} \\ \frac{1}{n} \sum_{i=1}^{n} \frac{-y_i e^{-y_i(x_i^T w + w_0)}}{1 + e^{-y_i(x_i^T w + w_0)}} x_i \end{pmatrix}$$