MASTER 2 MICAS

Machine Learning Communications and Security

Course : MICAS931 - Introduction to Optimization.

School : Institut Polytechnique de Paris

Teacher : HADI GHAUCH

Subject : Homework 2 - Submit before 22/11/2020

Etudiant : Panongbene Jean Mouhamed Sawadogo.
Email : panongbene.sawadogo@telecom-paris.fr

SOMMAIRE :

Part I : Closed-form solutions vs iterative algorithms

Consider the following linear regression problem :

$$w^* = \underset{w \in \mathbb{R}^d}{min} \frac{1}{N} \sum_{i \in [N]} \|w^T x_i - y_i\|_2^2 + \lambda \|w\|_2^2$$

where $\{(x_i, y_i)\}_{i=1}^N$ is the training set

a) Derive a closed-form solution for $w^* \in (1)$.

Answer : Define, the f function by for all w :

$$f(w) = \frac{1}{N} \sum_{i \in [N]} \|w^T x_i - y_i\|_2^2 + \lambda \|w\|_2^2$$

$$\Longrightarrow f(w) = \frac{1}{N} \sum_{i \in [N]} \left(\|w^T x_i - y_i\|_2^2 + \lambda \|w\|_2^2 \right)$$

Let $g_i(w) = \|w^T x_i - y_i\|_2^2 + \lambda \|w\|_2^2$

then we have :

$$f(w) = \frac{1}{N} \sum_{i \in [N]} g_i(w)$$

NB : we consider the functions g_i thus defined in the other questions

For all $i \in [N]$, $g_i(w)$ is strictly convex(because, the function w $\rightarrow \lambda \|w\|_2^2$ and the function w $\rightarrow \|w^T x_i - y_i\|_2^2$ are strictly convex for $x_i \neq 0_{\mathbb{R}^d}$ and $\lambda \neq 0$).

So we have $w^* \in \{w, \nabla f(w) = 0\}$

We have :

$$\nabla f(w) = \nabla_w \left(\frac{1}{N} \sum_{i \in [N]} g_i(w) \right)$$

$$\Longrightarrow \nabla f(w) = \frac{1}{N} \sum_{i \in [N]} \nabla g_i(w)$$

$$\nabla g_i(w) = \nabla_w \left(\|w^T x_i - y_i\|_2^2 + \lambda \|w\|_2^2 \right)$$

$$\implies \nabla g_i(w) = 2x_i(w^T x_i - y_i) + 2\lambda w$$

$$\implies \nabla g_i(w) = 2(x_i x_i^T + \lambda I_d)w - 2x_i y_i$$

$$\implies \nabla f(w) = \frac{1}{N} \sum_{i \in [N]} \nabla g_i(w) = \frac{1}{N} \sum_{i \in [N]} \left(2(x_i x_i^T + \lambda I_d)w - 2x_i y_i \right)$$

$$\implies \nabla f(w) = \frac{2}{N} \sum_{i \in [N]} \left((x_i x_i^T + \lambda I_d) \right) w - \frac{2}{N} \sum_{i \in [N]} x_i y_i$$

$$\implies \nabla f(w) = 0 \iff \frac{2}{N} \sum_{i \in [N]} \left((x_i x_i^T + \lambda I_d) \right) w - \frac{2}{N} \sum_{i \in [N]} x_i y_i = 0$$

$$\implies \nabla f(w) = 0 \iff \frac{2}{N} \sum_{i \in [N]} \left(x_i x_i^T + \lambda I_d \right) w = \frac{2}{N} \sum_{i \in [N]} x_i y_i$$

$$\implies \nabla f(w) = 0 \iff w = \left(\sum_{i \in [N]} \left(x_i x_i^T + \lambda I_d \right) \right)^{-1} \left(\sum_{i \in [N]} x_i y_i \right)$$

$$\boxed{w^* = \left(\sum_{i \in [N]} \left(x_i x_i^T + \lambda I_d \right) \right)^{-1} \left(\sum_{i \in [N]} x_i y_i \right)}$$

Derive an approximation of the computational complexity (in terms of $\mathcal{O}(-)$ analysis) for this solution.

Answer :

To caculate the value of w^*, we will need to calculate the sum $B = \sum_{i \in [N]} x_i y_i$ and the matrix $A = \sum_{i \in [N]} \left(x_i x_i^T + \lambda I_d \right)$. We also need to calculate the inverse of A matrix. So we can say that the complexity of calculatind w^* is :

$$\mathcal{O}\left(d^3 + d^2 + N^2 \right)$$

where :

- d is the number of columns of dataset

- N is the number of items in the dataset

b) Consider "Communities and Crime" dataset (N = 1994, d = 128) and find the optimal linear regressor from the closed-form expression.

Answer : We have for $\lambda = 0.5$:

w^* = [-6.02231923e-04 -1.50167149e-04 -1.37859019e-07 -8.18314963e-06 -1.57346396e-03 1.13965333e-02 3.14958146e-02 1.84383001e-01 -3.41133211e-02 -1.60405777e-02 8.51617354e-02 8.92041327e-02 -1.75013051e-01 -1.00931819e-01 9.19171975e-02 -5.21123524e-02 4.41278161e-02 5.89882879e-03 -9.18369729e-02 3.82116458e-02 -1.24920407e-01 1.01895214e-01 2.22120080e-02 -6.49036991e-02 1.04600899e-01 3.34179572e-02 -2.17844194e-01 -2.66645770e-02 -2.85124015e-02 2.35119297e-02 4.17598658e-02 3.06897178e-02 -2.80944874e-02 -1.28366925e-01 -8.84634642e-02 6.58146250e-02 5.25991613e-02 -5.76285800e-03 2.19768468e-01 -5.90657949e-02 -1.10044077e-02 7.43840094e-02 8.24641935e-02 2.21515314e-01 1.86674902e-01 -1.10648838e-01 -5.23342363e-02 -1.10027567e-02 -5.32493981e-02 -1.67453729e-01 -2.94713882e-02 -4.66694282e-03 4.01817080e-02 -1.66537419e-01 -6.96338755e-02 1.51115096e-01 -1.65141065e-01 2.37491352e-02 -1.35079200e-02 -3.46203722e-02 2.64724272e-02 -3.40446148e-02 -2.82429388e-02 1.00298530e-01 4.17499819e-03 2.86529736e-02 -1.32738573e-01 -7.61519654e-02 -6.61230146e-02 2.94949076e-01 -9.06136177e-02 -4.65985651e-02 -1.76037529e-01 1.46524624e-01 9.32108706e-02 2.72410650e-02 1.40781169e-01 -5.04206918e-02 6.25515676e-02 6.05588730e-02 -5.81904686e-02 2.13235418e-03 3.84437264e-02 -1.09626451e-02 -1.78950477e-01 3.21895760e-02 6.15499106e-02 -1.98739477e-01 1.28773453e-02 -1.73132390e-02 2.20117799e-01 5.85224696e-02 -3.43346432e-02 -8.07973576e-02 1.19495604e-01 1.47819683e-01 1.19654077e-01 2.37508932e-02 1.91154929e-02 9.16087180e-03 5.23790813e-03 1.20621026e-01 1.58412509e-02 2.33364772e-01 1.28799217e-01 -6.22907222e-02 -4.21103245e-03 1.24943482e-01 1.93948514e-02 -8.68494037e-02 2.33912593e-02 5.18532743e-02 9.21820216e-02 8.25776104e-02 -2.21594318e-02 -2.94221796e-02 -1.42134566e-02 -1.89246197e-02 3.75092809e-02 1.19614554e-02 -4.11103057e-02 9.02887491e-02 1.23399861e-01 5.93705840e-03 2.85111631e-02 2.74732492e-03 -1.13282884e-01]

NB : Before performing the computation of w, we preprocessed the data. Indeed, for each column, we replaced the missing data by the median value

of the column, for column 4 where the data was qualitative data, we replaced this qualitative data by numerical data by associating each qualitative data with a number. . The first qualitative data takes the value 1, the next the value 2 ... We have considered here that the values to be predicted(the target) are the values in the last columns

What is the complexity in that case for this value of N ?

Answer : In this case, we have N=1994 and d = 128 so the complexity is given by :

$$\boxed{\mathcal{O}\left(128^3 + 128^2 + 1994^2\right) = \mathcal{O}\left(6089572\right)}$$

c) Repeat b) for "Individual household electric power consumption" dataset (N = 2075259, d = 9). Observe the scalability issue of the closed-form expression.

Answer : we have :

w^* =[-3.40895363e-07 3.29228908e-11 4.50806913e-07 5.99998847e-02 5.99998845e-02 5.99998896e-02 5.99998863e-02]

NB : Before performing the computation of w, we preprocessed the data. Indeed, for each column, we have replaced the missing data with the median value of the column. For the datatime column where the data was dates, we delete this columns. **We have considered here that the values to be predicted(the target) are the values in the *Global_active_power* columns**

SCALABILITY : We observe scalability of the parameter w.

The complexity in that case ?

Answer : In this case, we have N=2075259 and d = 9 so the complexity is given by :

$$\boxed{\mathcal{O}\left(9^3 + 9^2 + 2075259^2\right) = \mathcal{O}\left(4306699917891\right)}$$

d) Derive a simple gradient algorithm to solve (1), iteratively..

Answer : We can consider the following gration descent algorithm :

$$w_{k+1} = w_k - \alpha \nabla f(w_k), \alpha \geq 0 \quad where \quad \alpha > 0$$

$$\Longrightarrow w_{k+1} = w_k - \frac{2\alpha}{N} \sum_{i \in [N]} \left((x_i x_i^T + \lambda I_d) \right) w_k - \frac{2\alpha}{N} \sum_{i \in [N]} x_i y_i$$

$$\boxed{w_{k+1} = w_k - \frac{2\alpha}{N} \sum_{i \in [N]} \left((x_i x_i^T + \lambda I_d) \right) w_k - \frac{2\alpha}{N} \sum_{i \in [N]} x_i y_i}$$

Algorithm :

step 1- Choose $\epsilon > 0$, the stop condition and $\alpha > 0$

step 2- Initialize $w_k = w_0$

step 3- While $\|\nabla f(w_k)\|_2 \leqslant \epsilon$

$$w_k = w_k - \alpha \nabla f(w_k)$$

step 5- return w_k

How do you select the 'best' stepsize ? argue your choice.

Answer : we can select the step size $\alpha = \frac{2}{L+\mu}$ where L and μ is respectively the max and the min of eigenvalues of A=$\frac{2}{N} \sum_{i \in [N]} \left((x_i x_i^T + \lambda I_d) \right)$ matrix.

As we demonstrated in the hw1 this step is the best to have a good time of convergence.

Derive an approximation of the computational complexity (in terms of $\mathcal{O}(-)$ analysis) for this method. Compare it with that of the closed form solution.

Answer : For a given step, the calculation of w requires to calculate the matrix A=$\frac{2}{N} \sum_{i \in [N]} \left((x_i x_i^T + \lambda I_d) \right)$ and the sum B=$\sum_{i \in [N]} x_i y_i$ but unlike the previous one we do not need a matrix inversion. therefore, the complexity in

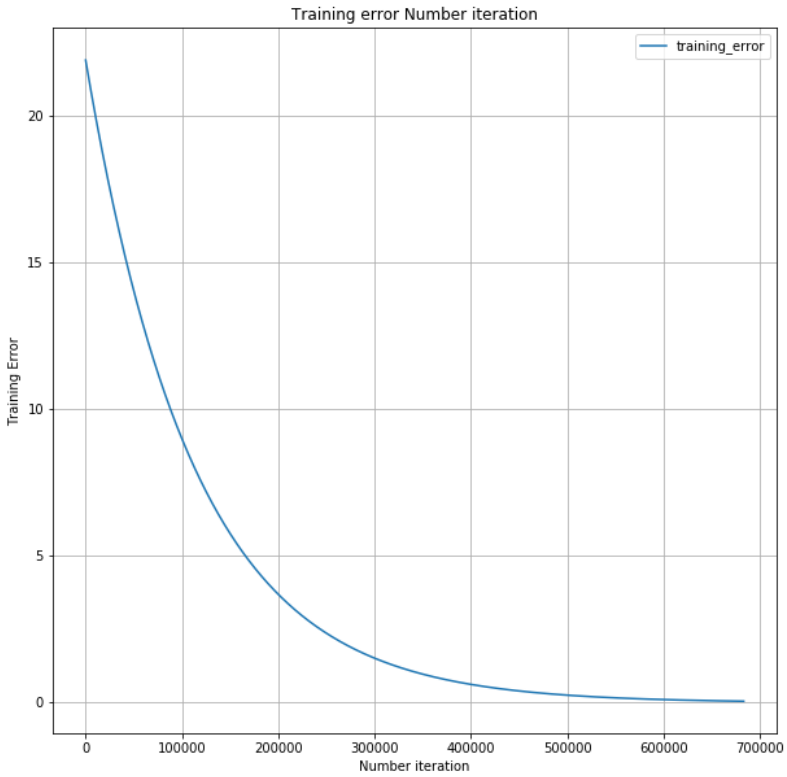
this case is equal to :

$$\mathcal{O}(N^2 + d^2)$$

This complexity is smaller than in the previous case.

Implement this algorithm using the "Individual household electric power consumption"
Plot the training error (using l_2 loss function) vs the number of iterations, against the optimal closed-form solution.

Answer : we have :



Verify that the gradient descent eventually converges to the optimal solution.

Argue the reason for that.

Answer : For an error less than or equal to 0.05, we have at the last itération :

$w^* = [\,0.02275059\ 0.02496082\ -0.04545081\ -0.04788073\ -0.04779283\ -0.04797129\ -0.0473603\,]$

which approaches the optimal solution, we can therefore say that the descent of the gradient converges towards the optimal solution

Part II :Deterministic vs Stochastic algorithms

Now consider logistic ridge regression

$$\underset{w\in\mathbb{R}^d}{arg\,min} = \frac{1}{N}\sum_{i\in[N]} f_i(w) + \lambda\|w\|_2^2,\ where\ f_i(w) = log\left(1 + exp\left\{-y_i w^T x_i\right\}\right)$$

for the "Individual household electric power consumption" dataset (N = 2075259, d = 9).
This is supervised learning problem, specifically, a classification problem, i.e., the training set is defined as :$\left\{(x_i, y_i)_{i=1}^N\right\}$, where $x_i \in \mathbb{R}^d$ is a feature vector, and $y_i \in \mathbb{B}$ is a binary label. Solve the optimization problem using GD, plain stochastic GD, and SVRG.

0-a) Is f Lipschitz continuous ? If so, find a small Lipschitz constant L ?

Answer : We have for all w :

$$f(w) = \frac{1}{N}\sum_{i\in[N]} f_i(w) + \lambda\|w\|_2^2$$

$$\Longrightarrow f(w) = \frac{1}{N}\sum_{i\in[N]} \left(f_i(w) + \lambda\|w\|_2^2\right)$$

Let $g_i(w) = f_i(w) + \lambda\|w\|_2^2$

then we have :

$$f(w) = \frac{1}{N} \sum_{i \in [N]} g_i(w)$$

NB : we consider the functions g_i thus defined in the other questions

Let B > 0 and define $\mathcal{H} = \left\{ w \in \mathbb{R}^d \, \|w\|_2 \leq B \right\}$

For all i $\in [N]$ and $w \in \mathcal{H}$ we have :

$g_i(w) = f_i(w) + \lambda \|w\|_2^2$

$$\implies g_i(w) = log\left(1 + exp(-y_i w^T x_i)\right) + \lambda \|w\|_2^2$$

$$\implies \nabla g_i(w) = \frac{-y_i x_i exp(-y_i w^T x_i)}{1 + exp(-y_i w^T x_i)} + 2\lambda w$$

we have :

$$exp(-y_i w^T x_i) \leqslant 1 + exp(-y_i w^T x_i)$$

$$\implies \frac{exp(-y_i w^T x_i)}{1 + exp(-y_i w^T x_i)} \leqslant 1$$

$$\implies \|\nabla g_i(w)\|_2 \leqslant \|y_i x_i\|_2 + 2\lambda \|w\|_2$$

$$\implies \|\nabla g_i(w)\|_2 \leqslant \|y_i x_i\|_2 + 2\lambda B$$

$$\implies \frac{1}{N} \sum_{i \in [N]} \|\nabla g_i(w)\|_2 \leqslant \frac{1}{N} \sum_{i \in [N]} \|y_i x_i\|_2 + 2\lambda B$$

$$\implies \|\nabla f(w)\|_2 \leqslant \frac{1}{N} \sum_{i \in [N]} \|y_i x_i\|_2 + 2\lambda B$$

So f is lipschtz continous and we can take :

$$\boxed{L = \frac{1}{N} \sum_{i \in [N]} \|y_i x_i\|_2 + 2\lambda B}$$

0-b) Is f strongly convex ? If so, find a large μ ?

Answer : Consider this function : $h(w) = \lambda \|w\|_2^2$ then :

$\nabla h(w) = 2\lambda w$

$\implies \nabla^2 h(w) = 2\lambda I_d$

$\implies \nabla^2 h(w) \succ 2\lambda I_d$

So, the h function is 2λ-strongly convex (**)

$\nabla f_i(w) = \nabla_w \left(log(1 + exp(-y_i w^T x_i)) \right)$

$$\implies \nabla f_i(w) = \frac{-y_i x_i exp(-y_i w^T x_i)}{1 + exp(-y_i w^T x_i)}$$

$$\implies \nabla^2 f_i(w) = \frac{y_i^2 x_i x_i^T exp(-y_i w^T x_i)}{(1 + exp(-y_i w^T x_i))^2}$$

$\implies \nabla^2 f_i(w) \succcurlyeq 0$

so we can say that the function f_i is convex (***)

(**) and (***) $\implies g_i(w) = f_i(w) + h(w)$ is 2λ-strongly convex

So, $f(w) = \frac{1}{N} \sum_{i \in [N]} g_i(w)$ is 2λ-strongly convex and we can choose

$$\boxed{\mu = 2\lambda}$$

0-c) Can you use the same "trick" as Part III of HW1 to find L and μ ?

Answer : we ca,=n determinate the value of L and μ using the same strick in Part III of HW1 i.e :

L= max $\{\lambda_i\}_{i \in [N]}$

μ= min $\{\lambda_i\}_{i \in [N]}$

where $\{\lambda_i\}$ is the eigenvalues of the matrix $\nabla^2 f(w)$

0-d) What can you say about the ratio $\frac{L}{\mu}$? is it a 'good' or 'bad' setup for a plain GD method, for the dataset considered here ?

Answer : we have :

$$\nabla^2 f(w) = \frac{1}{N} \sum_{i \in [N]} \left(\frac{y_i^2 x_i x_i^T exp(-y_i w^T x_i)}{(1 + exp(-y_i w^T x_i))^2} + 2\lambda I_d \right)$$

So we can see to compute $\nabla^2 f(w)$ depend of the size os dataset. So, determinated the eigenvalues of $\nabla^2 f(w)$ depends also of size of dataset : more the dataset is large, more the computational complexity for to determinate eigenvalues is difficults.

So for this case, the ratio $\frac{L}{\mu}$ is bad for a plain GD method, for the dataset considered here.

a) Write/Derive the update equations for each algorithm, and implement them

Answer : we have :

b) Tune the hyper-parameters for algorithm (including λ), using a held out cross validation test set.

Answer : to do :

c) Compare all these methods in terms complexity of hyper-parameter tunning, convergence time, convergence rate (in terms of # outer-loop iterations), and memory requirement.

Answer : to do :

d) Compare the training error (using l_2 loss function) of all these algorithms. Comment on these results : are they expected or is there any discrepancy ?

Answer : to do :

e) Compare the test error (using l_2 loss function) of all these algorithms. Comment on these results : are they expected or is there any discrepancy ?

Answer : to do :

Part III : Matrix Factorization for Recommendation System

Consider the low-rank matrix factorization (MF) for recommendation systems. Read carefully the problem setting in second lecture on non-convex optimisation (lecture 7)

- Data matrix $\mathbf{X}$ of size M $\times$ N . M = number of users, N = number of items

- $[X]_{u,i}$ is rating (score) that user u gave to item i

- In recommendation sys model rating as : $[X]_{u,i} = p_i^T q_u$,
$p_i \in \mathbb{R}^d$ vector of item characteristics ,$q_u \in \mathbb{R}^d$ vector of user preferences

- factorize $\mathbf{X}$ with low-rank matrices : $\mathbf{P} \in \mathbb{R}^{M\times d}, \mathbf{Q} \in \mathbb{R}^{N\times d}, d \ll (M, N)$ such that $\mathbf{X} = \mathbf{P}\mathbf{Q}^\mathbf{T}$

- every col. of $\mathbf{Q}^\mathbf{T}$ (resp. rom of $\mathbf{P}$) is user prefs (resp item characteristics)

We will build on that example by considering a realistic implementation using popular movie recommendation datasets, e.g., the MovieLens dataset. You will be solving the problem posed in Lecture 7, namely, the "Low-rank MF (supervised learning)" slide : please make sure you read the problem that is formulated in that slide.

We denote by $\mathcal{K}$ the set of **ratedentries** in $\mathbf{X}$ i.e, ntries of the rating matrix which have a score/rating by at least one user. Then the low-rank MF model is learned from the set of rated entries, $\mathcal{K}$, e.i.,

$$\underset{\{p_i,q_u\}_{(u,i)\in\mathcal{K}}}{argmin} \sum_{(u,i)\in\mathcal{K}} \Big[\Big([X]_{u,i} + \lambda_i\|p_i\|_2^2 + \mu_u\|q_u\|_2^2\Big)\Big]$$

In general, $\{\lambda_i > 0, \mu_u\}_{(u,i)\in\mathcal{K}}$ are regularization parameters to prevent overfitting, tuned via cross-validation. However, here we will focus on the training part (rather than the testing) and assume that all these regularization parameters are **strictly positive** and **given** to us. Due to the presence of coupling in the cost function, Block Coordinate Descent (BCD) methods are prevalent for solving (2). We thus propose to use the following standard methods to solve (2) : Block-Coordinate Descent (BCD) with closed-form solution, and BCD with Gradient Descent (GD).

0a) Starting from (2), apply the BCD method to derive the subproblem for each block of coordinates, In other words, derive the subproblem corresponding to each of the optimization variables in (2), $\{p_i, q_u\rangle_{(u,i)\in\mathcal{K}}$

Answer : to do :

0b) show that each subproblem is strongly convex.

Answer : to do :

1) **BCDwithclosed − formsolution** : also called alternating least-squares
- derive the update for each of the subproblems (part 0a)) in **closed − formsolution**
- show that algorithm which results from these updates **converges monotonically to a stationary point** of (2)

- derive an estimate of the **computational complexity** per BCD iteration (in $\mathcal{O}()$ notation)

2) **BCD with GD** : also called block-gradient descent
- In part 1) you derived a closed-form solution for each of the subproblems. For this method, do not solve each subproblem optimally, but rather take T **GD steps** (T is a small positive integer) with respect to the block of coordinates being optimized for that subproblem. Derive the update equations that result from this strategy, as a function T.
- what happens as $T \to \infty$?
- can you show the **convergence to a stationary point** of the resulting algorithm using the framework of BCD ?
- derive an estimate of the **computational complexity** per BCD iteration, as a function T (in $\mathcal{O}()$ notation)

3 Implement all your algorithms using the MovieLens 100K dataset.

- pick a value for model complexity, d, and plot the training error (empirical risk in (2)) using the ‘ 2 loss function, for each of the algorithm.

- plot the training error of BCD with GD for increasing value of T . what are the pros and cons of increasing T ?