ART.T458 Advanced Machine Learning Midterm Assignment (Final report assigned by Shimosaka)

Masamichi Shimosaka
Department of Computer Science
Tokyo Institute of Technology

Instruction

- Choose the problems from your preference and solve the chosen problems.
- Use Tex / MS word.
- Submit the A4 sized / letter sized report (pdf format file) by **30th 17:00 July 2021 via t2schola**.
- The maximum points you can earn in each problem is shown in each section title.
- Let $s_i \geq 0$ be the score you earned in i-th problem, the final score q used in the evaluation is processed by

$$q = \min(30, \sum_i s_i).$$

- In addition to the above q pts, you could also earn 2 pts (that is independent of q), if you show which topics of machine learning you want to take in the 1st half of this course as well as linear models, sparse learning, optimization, and the contents of 2nd half (deep learning). I hope to take some keywords of your preference such as *ensemble models such as boosting with decision or regression tress and bagging, Bayesian methods such as nonparametric Bayes, Markov chain Monte Carlo techniques, and variational Bayes, sequence modeling such as Kalman filters, Markov models, and CRFs, some application perspective issues such as recommender systems (learning to ranking, factorization machines), numerical optimization such as interior point algorithms, augmented Lagrangian, ADMM, sequential quadratic programming*, and so on. Please do not refer current contents of 1st and 2nd half lectures. If you find some typos / mistakes in the slides, your suggestion x 回目の講義のスライド y ページ目の z の式に誤りがあり，w のように修正すべきです can be added as this bonus 2 pts. Personally, suggestions to improve the quality of the course, of course, are welcome.
- You do not need to include the source code of your implementation in your report. However, you are welcome to show the URL of your codes via github or some other publicly available repository services.
- IMPORTANT: In this report, you are not allowed to use high level machine learning libraries, such as SciPy, scikit-learn, (Py)-Torch, Tensor Flow, and Neural network toolbox in Matlab, but are allowed to use basic linear algebra libraries such as NumPy, and basic Matlab language functionalities. It should be noted that the main objective of this report is to understand mathematical perspective of ML with implementations instead of how to use (black box) ML libraries.
- You can use **Japanese** as well as English.
- Some reference code in Jupyter notebook in `https://bit.ly/32gbpRt` might be helpful to promote this assignment and might also be updated frequently. You are allowed not to use this script but create your own code from the start. Of course, you could use Matlab as well as Python.

NOTE: *, **, *** indicate the levels of difficulty of each problem, respectively. I hope you could solve most of the problems labeled as *. I encourage students choose **, and *** if they want to enhance mathematical aspects of machine learning. Note that you could earn 30 pts without any implementation of ML software :-).

Problem 1 (10 pts)*

We consider a binary classification with a linear logistic regression. Let $\boldsymbol{x} \in \mathbb{R}^d$, and $\boldsymbol{w} \in \mathbb{R}^d$ be an d-dimensional input vector, and a parameter of the model, respectively. The classifier is represented by $f(\boldsymbol{x}) = 2[\![\boldsymbol{w}^\top \boldsymbol{x} > 0]\!] - 1$, where $[\![c]\!]$ denotes an indicator function that returns 1 if c is true, otherwise returns 0. With the supervised dataset $\{\boldsymbol{x}_i, y_i\}_{i=1}^n$, we consider an optimization problem for the logistic regression. The optimization problem can be written as

$$\hat{\boldsymbol{w}} = \operatorname*{argmin}_{\boldsymbol{w}} J(\boldsymbol{w})$$
$$J(\boldsymbol{w}) := \sum_{i=1}^{n} \left(\ln\left(1 + \exp(-y_i \boldsymbol{w}^\top \boldsymbol{x}_i)\right)\right) + \lambda \boldsymbol{w}^\top \boldsymbol{w}.$$

Here we assume that $\boldsymbol{x}_i$ contains constant value 1 to make the classifier adaptable to offset in $d-1$ dimensional feature space. With some artificial dataset (see Toy Dataset section, Dataset IV), we consider to implement some optimization methods in the following way.

1. Implement batch steepest gradient method[1].
2. Implement Newton based method.
3. Compare the performance of the above two optimization methods by showing $|J(\boldsymbol{w}^{(t)}) - J(\hat{\boldsymbol{w}})|$ w.r.t. t, where $\boldsymbol{w}^{(t)}$ represents the parameter at t-th iteration, and $\hat{\boldsymbol{w}}$ represent optimal parameter that reaches minimum of J obtained by (either of) the two methods, in semi-log plot.
4. Implement Newton method and simple steepest gradient method for multiclass version of logistic regression (use Toy dataset V) and run the same experiment as binary logistic regression mentioned above.

Problem 2 (5 pts)*

We consider *lasso*, where the square loss, and the L1 regularization are employed for linear regression. In this problem, we employ proximal gradient method (PG). So as to make the discussion simple, we use the following objective:

$$\hat{\boldsymbol{w}} = \operatorname*{argmin}_{\boldsymbol{w}} \left((\boldsymbol{w} - \boldsymbol{\mu})^\top \boldsymbol{A} (\boldsymbol{w} - \boldsymbol{\mu}) + \lambda ||\boldsymbol{w}||_1 \right).$$

Implement PG for lasso and show the results in a couple of conditions. In this question, use the same learning rate $\eta_t = L^{-1}$, where L depicts the Lipsitz constant of the gradient of the objective, which is derived from the Hessian matrix $2\boldsymbol{A}$ (i.e. use the maximum eigen value of $2\boldsymbol{A}$ as the inverse of the learning rate: η_t^{-1}).

1. Show the result of PG in terms of $||\boldsymbol{w}^{(t)} - \hat{\boldsymbol{w}}||$ w.r.t. the number of iteration. Use semi log plot. Use the following condition:

   $$\boldsymbol{A} = \begin{pmatrix} 3 & 0.5 \\ 0.5 & 1 \end{pmatrix}, \boldsymbol{\mu}^\top = \begin{pmatrix} 1 & 2 \end{pmatrix}.$$

   To verify the property of L1 regularization, run the experiment with $\lambda = 2, 4, 6$. Recall that the numerical result can be found in the slide used in the lecture. You can also verify the result by using cvx (matlab) / cvxopt (python).

[1]We assume here the learning rate is constant for simplicity. Consider the upper bound of Lipsitz constant of the gradient of this objective.

Problem 3 (15 pts)*

We consider the dual of the support vector machine (L2-regularized hinge loss based binary classifier). The original optimization problem of this classification can be represented as

$$\hat{\boldsymbol{w}} = \underset{\boldsymbol{w}\in\mathbb{R}^d}{\operatorname{argmin}} \left(\sum_{i=1}^{n} \max(0, 1 - y_i \boldsymbol{w}^\top \boldsymbol{x}_i) + \lambda ||\boldsymbol{w}||_2^2 \right), \tag{1}$$

where $\boldsymbol{x}_i \in \mathbb{R}^d$, $\boldsymbol{w} \in \mathbb{R}^d$, $y_i \in \{\pm 1\}$, and $\lambda > 0$ denotes i-th input variable, the parameter vector, and the label for i-th input data, and a coefficient of the regularization term, respectively.

1. Verify that the dual Lagrange function of this optimization can be written as

$$\begin{array}{rl} \text{maximize}_{\boldsymbol{\alpha}\in\mathbb{R}^n} & -\frac{1}{4\lambda}\boldsymbol{\alpha}^\top \boldsymbol{K}\boldsymbol{\alpha} + \boldsymbol{\alpha}^\top \mathbf{1} \\ \text{subject to} & \mathbf{0} \le \boldsymbol{\alpha} \le \mathbf{1} \end{array}, \tag{2}$$

   where $\mathbf{1}$ and $\mathbf{0}$ denote a n dimensional vector whose elements are all 1 and 0, respectively, and $\boldsymbol{K} \in \mathbb{R}^{n\times n}$ denotes a symmetric square matrix, and its i-th row and j-th column element can be represented by $y_i y_j \boldsymbol{x}_i^\top \boldsymbol{x}_j$.

2. From the KKT condition, verify the optimal weight parameter $\boldsymbol{w}$ given by $\boldsymbol{\alpha}$ can be written as

$$\hat{\boldsymbol{w}} = \frac{1}{2\lambda}\sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x}_i \tag{3}$$

3. Implement minimization for the negative dual Lagrange function using projected gradient (for simplicity). For the validity, show the score of the dual Lagrange function (use (2)), and sum of hinge loss function and the regularization, w.r.t. number of iteration (use (1) and (3)), respectively. Confirm that the duality gap reaches 0 for the convergence. Here we assume projected gradient just computes

$$\boldsymbol{\alpha}^{(t)} = P_{[0,1]^n}\left(\boldsymbol{\alpha}^{(t-1)} - \eta_t \left(\frac{1}{2\lambda}\boldsymbol{K}\boldsymbol{\alpha}^{(t-1)} - \mathbf{1} \right) \right),$$

   where η_t represents learning rate at t-th iteration, and $P_{[0,1]^n}$ depicts a projection operator that each of the input cast into $[0, 1]$.

Problem 4 (10 pts)**

We consider a binary classification problem, where the hinge loss function, and L1 regularization are leveraged. Let $\boldsymbol{x} \in \mathbb{R}^d$ be an input, and $\boldsymbol{w} \in \mathbb{R}^d$ be a parameter of the model, respectively. We consider a binary discriminant function with linear regression as $f(\boldsymbol{x}) = 2[\![\boldsymbol{w}^\top \boldsymbol{x} \ge 0]\!] - 1$, where $[\![c]\!]$ returns 1 when c is true, otherwise it returns 0. With a supervised dataset $\{\boldsymbol{x}_i, y_i\}_{i=1}^n$, the learning problem can be formalized as

$$\hat{\boldsymbol{w}} = \underset{\boldsymbol{w}\in\mathbb{R}^d}{\operatorname{argmin}} \left(\sum_{i=1}^{n} \max(0, 1 - y_i \boldsymbol{w}^\top \boldsymbol{x}_i) + \lambda ||\boldsymbol{w}||_1 \right), \tag{4}$$

where $y_i \in \{\pm 1\}$ denotes a binary label for i-th training example.

1. Derive a linear program from (4) (with auxiliary variable $\xi_i \ge \max(0, 1 - y_i \boldsymbol{w}^\top \boldsymbol{x}_i) \ge 0$, and $e_i \ge |w_i| \ge 0$). Recall that linear program can be written as

$$\begin{array}{rl} \text{minimize}_{\boldsymbol{z}} & \boldsymbol{c}^\top \boldsymbol{z} \\ \text{subject to} & \boldsymbol{A}\boldsymbol{z} \le \boldsymbol{b} \end{array}$$

   (See *LpBoost* that deals with LP from L1-regularized hinge loss model)

2. By using some artificial dataset (see Toy Dataset section, Dataset IV), implement this problem via cvx (in Matlab) / cvxopt (in python) (just for reference) and a (batch) proximal subgradient method. Then confirm that the parameter properly converges. The specification of the dataset should be described in the report.

Problem 5 (10 pts + optional 5 pts)**

We consider a matrix optimization problem with the following objective:

$$\underset{\boldsymbol{Z}\in\mathbb{R}^{m\times n}}{\operatorname{argmin}} \left(\sum_{i,j\notin Q} |A_{i,j} - Z_{i,j}|^2 + \lambda ||\boldsymbol{Z}||_* \right)$$

where $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ represents data matrix (user vs. movie rating data in recommendation systems), and also contains null value on Q denotes. $||\cdot||_*$ represents a nuclear norm of the matrix, $\lambda > 0$ denotes a hyper parameter for the regularization. In this optimization problem, $\boldsymbol{Z}$ corresponds to the recovered data from the incomplete data matrix $\boldsymbol{A}$. In the scenario of the recommendation systems, the inferred $\boldsymbol{Z}$ at location Q corresponds to the inferred score of movie rating.

1. Describe the definition of the nuclear norm of a matrix by investigating it from www. In addition to its definition, define the proximal operation with the nuclear norm. (Hint: Use singular value decomposition.)
2. With some dataset (see Toy Dataset III), implement proximal gradient method for this machine learning problem and shows the recovered data $\boldsymbol{Z}$ by using surface plotting.
3. (Option: You can earn additional 5 pts) Implement non-negative matrix factorization as alternative approach to recover $\boldsymbol{A}$ and compare the performance by choosing the hyper parameters. Discuss the advantages and disadvantages of the two methods you implement here.

Problem 6 (10 pts)***

We consider a gradient descent algorithm for minimizing L-Lipschitz convex function f. From the starting point $\boldsymbol{w}^{(0)} \in \mathbb{R}^d$, and at each iteration we update the parameter $\boldsymbol{w}$ as $\boldsymbol{w}^{(t+1)} \leftarrow \boldsymbol{w}^{(t)} - \eta \left.\nabla f\right|_{\boldsymbol{w}=\boldsymbol{w}^{(t)}}$ (i.e. fixed step size gradient descent update). Finally we choose the optimum value

$$\hat{\boldsymbol{w}} = \underset{\boldsymbol{w}^{(0)},\boldsymbol{w}^{(1)},\ldots,\boldsymbol{w}^{(T)}}{\operatorname{argmin}} \{f(\boldsymbol{w}^{(0)}), f(\boldsymbol{w}^{(1)}), \ldots, f(\boldsymbol{w}^{(T)})\}$$

. We want to estimate the minimum number of iterations T^*, where $f(\hat{\boldsymbol{w}}) - f(\boldsymbol{w}^*) \leq \epsilon$ holds when we set $\eta = \epsilon/L^2$. Here we assume $\boldsymbol{w}^*$ be the optimum value to minimize function f. Please prove that such T^* can be written as $O(1/\epsilon^2)$.

Problem 7 (10 pts)***

Similar to Problem 6, we consider a gradient descent algorithm for minimizing γ-smooth convex function f. From the starting point $\boldsymbol{w}^{(0)} \in \mathbb{R}^d$, and at each iteration we update the parameter $\boldsymbol{w}$ as $\boldsymbol{w}^{(t+1)} \leftarrow \boldsymbol{w}^{(t)} - \eta \left.\nabla f\right|_{\boldsymbol{w}=\boldsymbol{w}^{(t)}}$ (i.e. fixed step size gradient descent update). Finally we choose the optimum value

$$\hat{\boldsymbol{w}} = \underset{\boldsymbol{w}^{(0)},\boldsymbol{w}^{(1)},\ldots,\boldsymbol{w}^{(T)}}{\operatorname{argmin}} \{f(\boldsymbol{w}^{(0)}), f(\boldsymbol{w}^{(1)}), \ldots, f(\boldsymbol{w}^{(T)})\}$$

. We want to estimate the minimum number of iterations T^*, where $f(\hat{\boldsymbol{w}}) - f(\boldsymbol{w}^*) \leq \epsilon$ holds when we set $\eta = 1/\gamma$. Here we assume $\boldsymbol{w}^*$ be the optimum value to minimize function f. Please prove that such T^* can be written as $O(1/\epsilon)$.

Problem 8 (10pts)*

We consider linear regression and the effect of regularization. As shown in the lecture, the optimization of the linear regression also known as least square problem is defined as the following optimization problem:

$$\hat{\boldsymbol{w}}_{\text{LS}} = \underset{\boldsymbol{w}}{\operatorname{argmin}} \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2,$$

where the design matrix, the response vector, and the parameter is represented by $\boldsymbol{X} \in \mathbb{R}^{n\times d}$, $\boldsymbol{y} \in \mathbb{R}^n$, and $\boldsymbol{w} \in \mathbb{R}^d$, respectively (Review lecture slides and check the video if necessary). Though the regression through this optimization may work properly under some conditions; it is also known that this model is prone to overfitting. As a simple approach to tackle the overfitting issue, Ridge regularization is frequently employed in machine learning community. The resultant optimization problem is defined as follows:

$$\hat{\boldsymbol{w}}_{\text{ridge}} = \underset{\boldsymbol{w}}{\operatorname{argmin}} \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2 + \lambda\|\boldsymbol{w}\|_2^2.$$

1. Obtain analytical solutions of $\hat{\boldsymbol{w}}_{\text{LS}}$, and $\hat{\boldsymbol{w}}_{\text{ridge}}$, respectively. Here we assume $\boldsymbol{X}^\top\boldsymbol{X}$ is regular, i.e. $(\boldsymbol{X}^\top\boldsymbol{X})^{-1}$ exists.

2. Even if $\boldsymbol{X}^\top\boldsymbol{X}$ is not regular, prove that $\boldsymbol{X}^\top\boldsymbol{X} + \lambda\boldsymbol{I}$ is regular. This means that the optimal parameter $\hat{\boldsymbol{w}}_{\text{ridge}}$ is always available whether $\boldsymbol{X}^\top\boldsymbol{X}$ is regular or not. Here $\boldsymbol{I} \in \mathbb{R}^{d\times d}$ represents an identity matrix and $\lambda > 0$ denotes hyper parameter of the regularization.

3. Here we assume that $\boldsymbol{X}^\top\boldsymbol{X}$ is regular, prove that $\|\boldsymbol{X}\hat{\boldsymbol{w}}_{\text{LS}}\|_2^2 \geq \|\boldsymbol{X}\hat{\boldsymbol{w}}_{\text{ridge}}\|_2^2$. This result is also known as shrinkage in machine learning. Explain situation(s) where the shrinkage works effectively in machine learning.

Problem 9 (10 pts) **

1. We consider to minimize the maximum distance from (x, y) to points $\{x_i, y_i\}_{i=1}^n$, i.e.

$$\text{minimize}_{x\in\mathbb{R}, y\in\mathbb{R}} \quad \max_i \sqrt{(x_i - x)^2 + (y_i - y)^2}$$

Convert this optimization problem into quadratic programming problem where quadratic function is used as objective function and linear equalities and inequalities are used in the constraints.

2. Convert the following optimization as convex optimization. Here we assume variable x_1, x_2, x_3 is real number, respectively. Hint: Please focus on the 4-th constraints and use other variables to represent x_1, x_2, x_3, respectively.

$$\begin{array}{rl} \text{minimize}_{x_1,x_2,x_3} & x2/x1 \\ \text{subject to} & x_1^2 + x2/x3 \leq \sqrt{x_2}, \\ & x_1/x2 = x_3^2, \\ & 2 \leq x_1 \leq 3, \\ & x_1, x_2, x_3 > 0 \end{array}$$

Toy Datasets

Use the following datasets for some problems, if necessary. It might be better to give seed to a random number generator in the initialization step. Though the following codes are described in Matlab, MS hopes students do not have difficulty in generating the similar dataset in python or other programming language implementation.

Dataset I

```
n = 100;
x = 3 * (rand(n, 2) - 0.5);
radius = [x(:, 1).^2 + x(:, 2).^2];
y = (radius > 0.7 + 0.1 * randn(n, 1)) & (radius < 2.2 + 0.1 * randn(n, 1));
y = 2 * y -1;
```

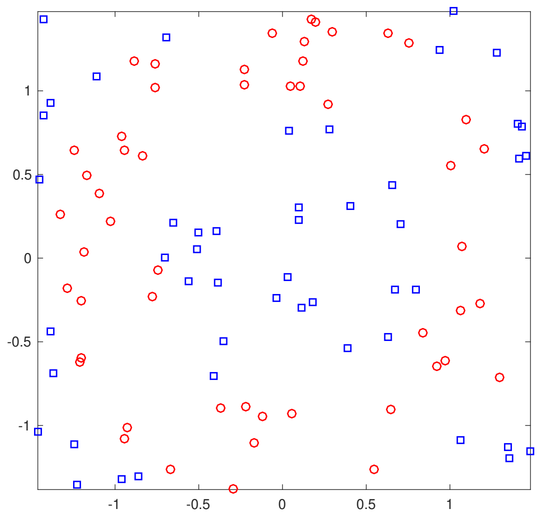

Figure 1: Dataset I

Dataset II

```
n = 40;
omega = randn(1, 1);
noise = 0.8 * randn(n, 1);

x = randn(n, 2);
y = 2 * (omega * x(:, 1) + x(:, 2) + noise > 0) - 1;
```

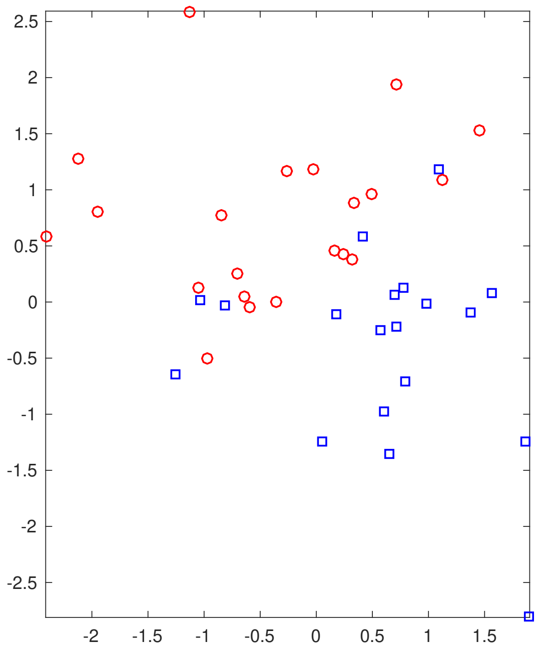

Figure 2: Dataset II

Dataset III

```
m = 20;
n = 40;

r = 2;
```

```
A = rand(m, r) * rand(r, n);

ninc = 100;

Q = randperm(m * n, ninc);

A(Q) = NaN;
```

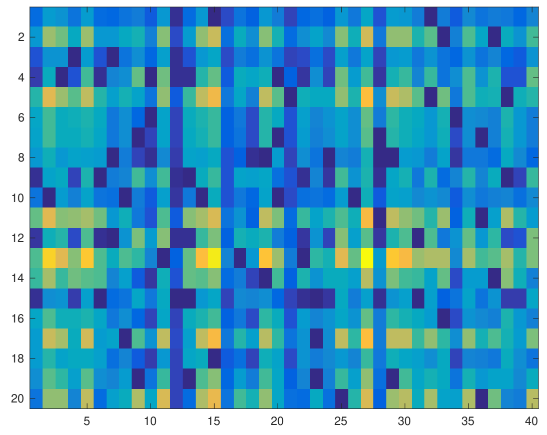

Figure 3: Dataset III

Dataset IV

```
n = 200;
x = 3 * (rand(n, 4) - 0.5);
y = (2 * x(:, 1) - 1 * x(:,2) + 0.5 + 0.5 * randn(n, 1)) > 0;
y = 2 * y -1;
```

Dataset V

```
n = 200;
x = 3 * (rand(n, 4) - 0.5);
W = [2, -1, 0.5;
    -3, 2, 1;
     1, 2, 3];

[maxlogit, y] =max( [x(:, 1:2), ones(n, 1)] * W' + 0.5 * randn(n, 3), [], 2);
```