The Chinese University of Hong Kong, Shenzhen

DDA3020

Machine Learning

Assignment1 Report

Author:
Shi Wenlan

Student Number:
119010265

October 2, 2022

DDA3020: Assignment I

September 29, 2022

Homework due: 11:59 pm, Oct 17, 2022. The weight of this assignment among all assignments is 13/60.

1 Written Problems (50 points)

1.1 Given the following denominator layout derivatives, (15 points)

- **Differentiation of a scalar function *w.r.t.* a vector**: If $f(\mathbf{w})$ is a scalar function of d variables, $\mathbf{w}$ is a $d \times 1$ vector, then differentiation of $f(\mathbf{w})$ *w.r.t.* $\mathbf{w}$ results in a $d \times 1$ vector

$$\frac{df(\mathbf{w})}{d\mathbf{w}} = \begin{bmatrix} \frac{\partial f}{\partial w_1} \\ \vdots \\ \frac{\partial f}{\partial w_d} \end{bmatrix}$$

- **Differentiation of a vector function *w.r.t.* a vector**: If $\mathbf{f}(\mathbf{w})$ is a vector function of size $h \times 1$ and $\mathbf{w}$ is a $d \times 1$ vector, then differentiation of $\mathbf{f}(\mathbf{w})$ *w.r.t.* $\mathbf{w}$ results in a $d \times h$ vector

$$\frac{d\mathbf{f}(\mathbf{w})}{d\mathbf{w}} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_h}{\partial w_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_1}{\partial w_d} & \cdots & \frac{\partial f_h}{\partial w_d} \end{bmatrix}$$

Please prove the following derivatives, and $\mathbf{X}$ and $\mathbf{y}$ are not functions of $\mathbf{w}$:

$$\frac{d(\mathbf{X}^\top \mathbf{w})}{d\mathbf{w}} = \mathbf{X},$$
$$\frac{d(\mathbf{y}^\top \mathbf{X}\mathbf{w})}{d\mathbf{w}} = \mathbf{X}^\top \mathbf{y}$$
$$\frac{d(\mathbf{w}^\top \mathbf{X}\mathbf{w})}{d\mathbf{w}} = (\mathbf{X} + \mathbf{X}^\top)\mathbf{w}$$

Solution:

Assume $X^T = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,d} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,d} \\ x_{h,1} & x_{h,2} & \cdots & x_{h,d} \end{bmatrix} = \begin{bmatrix} \bar{x}_1 \\ \bar{x}_2 \\ \bar{x}_h \end{bmatrix} = [\bar{x}_1, \bar{x}_2 \cdots \bar{x}_d]$ $\quad y^T = [y_1\ y_2\ \cdots\ y_d]$ $\quad w^T = [w_1\ w_2\ \cdots\ w_d]$

① $\frac{d(X^Tw)}{d(w)} = \begin{bmatrix} \frac{\partial(\bar{x}_1 w)}{\partial w_1} & \frac{\partial(\bar{x}_2 w)}{\partial w_1} & \cdots & \frac{\partial(\bar{x}_h w)}{\partial w_1} \\ \frac{\partial(\bar{x}_1 w)}{\partial w_2} & \frac{\partial(\bar{x}_2 w)}{\partial w_2} & \cdots & \frac{\partial(\bar{x}_h w)}{\partial w_2} \\ \frac{\partial(\bar{x}_1 w)}{\partial w_d} & \frac{\partial(\bar{x}_2 w)}{\partial w_d} & \cdots & \frac{\partial(\bar{x}_h w)}{\partial w_d} \end{bmatrix} = \begin{bmatrix} x_{1,1} & x_{2,1} & \cdots & x_{h,1} \\ x_{1,2} & x_{2,2} & \cdots & x_{h,2} \\ x_{1,d} & x_{2,d} & \cdots & x_{h,d} \end{bmatrix} = X$

② According to ①

$$\frac{d(y^TXw)}{d(w)} = \frac{d((y^TX)w)}{d(w)} = (y^TX)^T = X^Ty$$

③ In this case X must be a square matrix, where $h=d$.

$w^TXw = w^T\bar{x}_1 w_1 + w^T\bar{x}_2 w_2 + \cdots + w^T\bar{x}_d w_d$

$w^T\bar{x}_i = w_1 x_{i,1} + w_2 x_{i,2} + \cdots + w_d x_{i,d}$

$$\frac{d(w^TXw)}{dw} = \begin{bmatrix} \frac{\partial(w^TXw)}{\partial w_1} \\ \frac{\partial(w^TXw)}{\partial w_2} \\ \vdots \\ \frac{\partial(w^TXw)}{\partial w_d} \end{bmatrix} = \begin{bmatrix} 2w_1x_{1,1} + w_2x_{1,2} + \cdots + w_dx_{1,d} + x_{1,1}w_2 + \cdots + x_{d,1}w_d \\ x_{1,2}w_1 + w_1x_{2,1} + 2w_2x_{2,2} + w_2x_{2,3} + \cdots + w_dx_{i,d} + x_{3,2}w_3 + \cdots + x_{d,2}w_d \\ \vdots \\ x_{1,d}w_1 + \cdots + x_{d-1,d}w_{d-1} + w_1x_{d,1} + \cdots + w_{d-1}x_{d,d-1} + 2w_dx_{d,d} \end{bmatrix}$$

$$= \begin{bmatrix} \bar{x}_1 w + \bar{x}_1 w \\ \bar{x}_2 w + \bar{x}_2 w \\ \vdots \\ \bar{x}_d w + \bar{x}_d w \end{bmatrix} = X^Tw + Xw = (X + X^T)w$$

1.2 Suppose we have training data $\{(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), \ldots, (\mathbf{x}_N, \mathbf{y}_N)\}$, where $\mathbf{x}_i \in \mathbb{R}^d$ and $\mathbf{y}_i \in \mathbb{R}^k$, $i = 1, 2, \ldots, N$. (1) Find the closed-form solution of the following problem

$$\min_{\mathbf{W},\mathbf{b}} \sum_{i=1}^{N} \alpha_i \|\mathbf{y}_i - \mathbf{W}\mathbf{x}_i - \mathbf{b}\|^2,$$

where $\alpha_1, \ldots, \alpha_n$ are weights for different samples; (2) Show how to use gradient descent to solve the problem. (12 points)

Solution: $N\times(d+1)$ $(d+1)\times k$ $N\times k$

(1) Assume $X = \begin{bmatrix} \sqrt{\alpha_1}\mathbf{x}_1^T & \sqrt{\alpha_1} \\ \sqrt{\alpha_2}\mathbf{x}_2^T & \sqrt{\alpha_2} \\ \vdots & \vdots \\ \sqrt{\alpha_n}\mathbf{x}_N^T & \sqrt{\alpha_n} \end{bmatrix}$ $A = \begin{bmatrix} W \\ b \end{bmatrix}$ $Y = \begin{bmatrix} \sqrt{\alpha_1}\mathbf{y}_1^T \\ \sqrt{\alpha_2}\mathbf{y}_2^T \\ \vdots \\ \sqrt{\alpha_n}\mathbf{y}_N^T \end{bmatrix}$ and $(X^TX)^{-1}$ exists, and $d=k$ or $k=1$

$\mathbf{x}_1$ $\mathbf{x}_2$ $\mathbf{x}_n$

1 1 1

$$\therefore \min_{W,b} \sum_{i=1}^{N} \alpha_i \|\mathbf{y}_i - W\mathbf{x}_i - b\|^2 = \min_A \operatorname{trace}((XA-Y)^T(XA-Y))$$
$$= \min_A \operatorname{trace}((A^TX^T - Y^T)(XA-Y))$$
$$= \min_A \operatorname{trace}(A^TX^TXA - A^TX^TY - Y^TXA + Y^TY)$$
$$= \min_A \operatorname{trace}(A^TX^TXA - 2Y^TXA + Y^TY)$$

$$\frac{\partial(A^TX^TXA - 2Y^TXA + Y^TY)}{\partial A} = \frac{\partial(A^TX^TXA)}{\partial A} - 2\frac{\partial(Y^TXA)}{\partial A} + \frac{\partial(Y^TY)}{\partial A}$$
$$= (X^TX + X^TX)A - 2X^TY$$
$$= 2X^TXA - 2X^TY = 0$$

$\therefore X^TXA = X^TY$ $\therefore \begin{bmatrix} W \\ b \end{bmatrix} = A = (X^TX)^{-1}X^TY = \begin{bmatrix} \sum_{i=1}^{N}\alpha_i\mathbf{x}_i\mathbf{x}_i^T & \sum_{i=1}^{N}\sqrt{\alpha_i}\mathbf{x}_i^T \\ \sum_{i=1}^{N}\sqrt{\alpha_i}\mathbf{x}_i^T & \sum_{i=1}^{N}\alpha_i \end{bmatrix}^{-1} \begin{bmatrix} \sum_{i=1}^{N}\alpha_i\mathbf{x}_i\mathbf{y}_i^T \\ \sum_{i=1}^{N}\sqrt{\alpha_i}\mathbf{y}_i^T \end{bmatrix}$

$$\min_{W,b} \sum_{i=1}^{N} \alpha_i \|\mathbf{y}_i - W\mathbf{x}_i - b\|^2$$
$$= \operatorname{trace}((X^TX)^{-1}X^TY)^TX^TX(X^TX)^{-1}X^TY - 2Y^TX(X^TX)^{-1}X^TY + Y^TY)$$
$$= \operatorname{trace}(Y^TX(X^TX)^{-1}X^TY - 2Y^TX(X^TX)^{-1}X^TY + Y^TY)$$
$$= \operatorname{trace}(Y^TY - Y^TX(X^TX)^{-1}X^TY)$$
$$= \operatorname{trace}\left(\sum_{i=1}^{N}\alpha_i\mathbf{y}_i\mathbf{y}_i - \left[\sum_{i=1}^{N}\alpha_i\mathbf{y}_i\mathbf{x}_i^T \quad \sum_{i=1}^{N}\sqrt{\alpha_i}\mathbf{y}_i\right]\begin{bmatrix} \sum_{i=1}^{N}\alpha_i\mathbf{x}_i\mathbf{x}_i^T & \sum_{i=1}^{N}\sqrt{\alpha_i}\mathbf{x}_i^T \\ \sum_{i=1}^{N}\sqrt{\alpha_i}\mathbf{x}_i^T & \sum_{i=1}^{N}\alpha_i \end{bmatrix}^{-1}\begin{bmatrix} \sum_{i=1}^{N}\alpha_i\mathbf{x}_i\mathbf{y}_i^T \\ \sum_{i=1}^{N}\sqrt{\alpha_i}\mathbf{y}_i^T \end{bmatrix}\right)$$

(2) given a start point X in the domain of A

repeat

1. $\Delta X := -X^T(XA-Y)$
2. Choose the step size $t = \operatorname{argmin}_{t>0} X(X + t\Delta X)$
3. Update $X = X + t\Delta X$

until stopping criterion (e.g. $\|X^T(XA-Y)\|_2 \le \epsilon$) is satisfied.

$\begin{bmatrix} W \\ b \end{bmatrix} = X$

1.3 Prove that: (1) $f(x) = x^4$ is convex; (2) $f(x) = |x|$ is convex; (3) $f(\mathbf{x}) = \|\mathbf{Ax} - \mathbf{b}\|^2$ is convex, where $\mathbf{A}$ is a matrix. (12 points)

Solution:

(1) ∵ Domain of $f(x)$ is $\mathbb{R}$, which is convex

∵ $f''(x) = 12x^2 \geq 0$

∴ According to SOC, $f(x)$ is convex

(2) ∵ Domain of $f(x)$ is $\mathbb{R}$, which is convex

∵ $|\theta x + (1-\theta)y| \leq \theta|x| + (1-\theta)|y|$

∴ According to definition, $f(x)$ is convex

(3) ∵ Domain of $f(x)$ is $\mathbb{R}^n$, which is convex

$$f(x) = (Ax-b)^T(Ax-b)$$
$$= (x^TA^T - b^T)(Ax-b)$$
$$= x^TA^TAx - 2b^TAx + b^Tb$$
$$\nabla f(x) = \frac{\partial(x^TA^TAx - 2b^TAx + b^Tb)}{\partial x} = 2A^TAx - 2A^Tb$$

∵ $\nabla^2 f(x) = \frac{\partial(2A^TAx - 2A^Tb)}{\partial x} = 2A^TA \geq 0$

∴ According to SOC, $f(x)$ is convex.

1.4 Suppose $x_1, x_2, \ldots, x_N$ are drawn from $\mathcal{N}(\mu, \sigma^2)$. Show that the MLE (maximum likelihood estimation) of σ^2 is $\sigma^2_{ML} = \frac{1}{N}\sum_{n=1}^{N}(x_n - \mu_{ML})^2$. (11 points)

Solution:

Assume $X = \{x_i\}_{i=1}^N$

$$\sigma^2_{ML} = \arg\max_{\sigma^2} \log L(\sigma^2, \mu | X)$$
$$f(\sigma^2) = \log L(\sigma^2 | X) = \log\left(\prod_{i=1}^{N} p(x_i | \mu, \sigma^2)\right)$$
$$= \sum_{i=1}^{N} \log N(\mu, \sigma^2)$$
$$= \sum_{i=1}^{N} \log \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x_i-\mu)^2}{2\sigma^2}\right)$$
$$= -N\log\sqrt{2\pi\sigma^2} - \frac{1}{2\sigma^2}\sum_{i=1}^{N}(x_i-\mu)^2$$
$$f'(\sigma^2) = -N\frac{1}{\sqrt{2\pi\sigma^2}}\cdot\sqrt{2\pi}\cdot\frac{1}{2\sqrt{\sigma^2}} + \sum_{i=1}^{N}(x_i-\mu)^2\frac{1}{2\sigma^4}$$
$$= -\frac{N}{2\sigma^2} + \sum_{i=1}^{N}(x_i-\mu)^2\frac{1}{2\sigma^4} = 0$$
$$\therefore \sum_{i=1}^{N}(x_i-\mu)^2\frac{1}{\sigma^2} = N$$
$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i-\mu)^2$$
$$\therefore \sigma^2_{ML} = \frac{1}{N}\sum_{i=1}^{N}(x_i-\mu_{ML})^2$$

2.1. The CSV or XLS file contains a dataset for regression. There are 7750 samples with 25 features (described in the doc file). This data is for the purpose of bias correction of next-day maximum and minimum air temperatures forecast of the LDAPS model operated by the Korea Meteorological Administration over Seoul, South Korea. This data consists of summer data from 2013 to 2017. The input data is largely composed of the LDAPS model's next-day forecast data, in-situ maximum and minimum temperatures of present-day, and geographic auxiliary variables. There are two outputs (i.e. next-day maximum and minimum air temperatures) in this data. Hindcast validation was conducted for the period from 2015 to 2017.

You need to delete the first two attributes (station and date), and use attributes 3-23 to predict attributes 24 and 25. Randomly split the data into two parts, one contains 80% of the samples and the other contains 20% of the samples. Use the first part as training data and train a linear regression model and make prediction on the second part. Report the training error and testing error in terms of RMSE.

Repeat the splitting, training, and testing for 10 times. Use a loop and print the RMSEs in each trial.

Solution:

Read in data : np.loadtxt(file name)

Clean data : np. delete(rows with NaN)

Shuffle data : np.random.shuffle (data)

Splite data : train data : test data = 4 : 1

$X = [data[2:23]\ \mathbb{1}]$ $W = [w, b]$

$Y = [data[23:]]$

because the whole training process is too slow

Train : Initial value of W : get from a previous training process (initial : np.ones() , after 295000 iterations)

Objective function : $RMSE = \sqrt{\frac{1}{n} trace((XW-Y)^T(XW-Y))}$

Gradient : $X^T(XW-Y)$ and take negative

Stepsize : get from Golden Section Method iteration range from (0,2)

Stop condition : norm (Gradient) < 0.001

or

decrease of objective value < 0.001

because the whole training process is too slow

Test : Calculate RMSE on test data set

Result snapshoot :

```
Trail 0
train RMSE = 6.574515200326087
test RMSE = 7.289944883147615
Trail 1
train RMSE = 6.540215519510085
test RMSE = 7.412679830282912
Trail 2
train RMSE = 6.752142677709038
test RMSE = 6.608645543493965
Trail 3
train RMSE = 6.64360268964265
test RMSE = 7.02734407988645
Trail 4
train RMSE = 6.6777866935260155
test RMSE = 6.9043236270052954
Trail 5
train RMSE = 6.699952345247394
test RMSE = 6.451579486781242
Trail 6
train RMSE = 6.611091036144075
test RMSE = 6.899438956434362
Trail 7
train RMSE = 6.499482804076262
test RMSE = 7.537555680283795
Trail 8
train RMSE = 6.770978396204615
test RMSE = 6.531154433978679
Trail 9
train RMSE = 6.779375874598276
test RMSE = 6.496120347208102
```

2.2. The classification data file contains the iris dataset. This is perhaps the best known database to be found in the pattern recognition literature. Fisher's paper is a classic in the field and is referenced frequently to this day. (See Duda Hart, for example.) The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant. One class is linearly separable from the other 2; the latter are NOT linearly separable from each other. There are fiver attributes: 1. sepal length in cm; 2. sepal width in cm; 3. petal length in cm; 4. petal width in cm; 5. class: – Iris Setosa – Iris Versicolour – Iris Virginica. You need to use the first attributes to predict the last attribute, namely, classifying the data into three classes. Randomly split the data into two parts, one contains 80% of the samples and the other contains 20% of the samples. Use the first part as training data and train a linear model and make classification on the second part. Report the training error and testing error in terms of classification error rate (number of miss-classified samples divided by number of all samples)

Repeat the splitting, training, and testing for 10 times. Use a loop and print the classification errors in each trial.

Solution:

Read in data : xlrd.open_workbook(file name).sheets()[0]

Clean data: replace string tag to array like [1 0 0]

Shuffle data: np.random.shuffle(data)

Splite data: train data : test data = 4:1

X = [data[:4] 1] W = [w, b]

Y = [data[4:]]

because the whole training process is too slow

Train: Initial value of W: get from a previous training process (initial: np.ones() , after 5700 iterations)

Objective function: $RMSE = \sqrt{\frac{1}{n} trace((XW-Y)^T(XW-Y))}$

Gradient : $X^T(XW-Y)$ and take negative

Stepsize: get from Golden Section Method iteration range from (0,2)

Stop condioin: norm(Gradient) < 0.000001

or

decrease of objective value < 0.000001

Test: Classify the sample to the class with the highest value : e.g. XW = [0.1 0.7 0.2] → [0 1 0]

Calculate the error rate on both data sets

Result snapshoot:

```
Trail 0
train classification error rate = 0.125
test classification error rate = 0.23333333333333328
Trail 1
train classification error rate = 0.15833333333333333
test classification error rate = 0.09999999999999998
Trail 2
train classification error rate = 0.1333333333333333
test classification error rate = 0.19999999999999996
Trail 3
train classification error rate = 0.15000000000000002
test classification error rate = 0.16666666666666663
Trail 4
train classification error rate = 0.17500000000000004
test classification error rate = 0.09999999999999998
Trail 5
train classification error rate = 0.14166666666666672
test classification error rate = 0.19999999999999996
Trail 6
train classification error rate = 0.14166666666666672
test classification error rate = 0.19999999999999996
Trail 7
train classification error rate = 0.15833333333333333
test classification error rate = 0.06666666666666665
Trail 8
train classification error rate = 0.14166666666666672
test classification error rate = 0.09999999999999998
Trail 9
train classification error rate = 0.14166666666666672
test classification error rate = 0.19999999999999996
```