Assignment 2: Regression and Classification

AIGS/CSED515 Machine Learning
Instructor: Jungseul Ok
jungseul@postech.ac.kr

Due: 20:00pm Oct 22, 2020

Remarks

- Group study and open discussion via LMS board are encouraged, however, assignment that your hand-in must be **of your own work**, and **hand-written**.
- Submit a scanned copy of your answer on LMS online in **a single PDF file**.
- Delayed submission may get some penalty in score: 5% off for delay of $0 \sim 4$ hours; 20% off for delay of $4 \sim 24$ hours; and delay longer than 24 hours will not be accepted.

1. [20pt; Linear Regression] We are given a dataset $\mathcal{D} = \{(1,1),(2,1)\}$ containing two pairs (x,y) with $x \in \mathbb{R}$ and $y \in \mathbb{R}$. We want to find the parameters $\boldsymbol{w} = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \in \mathbb{R}^2$ of a linear regression model $\boldsymbol{y} = w_1 x + w_2$ using

$$\min_{\boldsymbol{w}} \frac{1}{2} \sum_{(x,y)\in\mathcal{D}} \left(y - \boldsymbol{w}^\top \begin{bmatrix} x \\ 1 \end{bmatrix} \right)^2 . \tag{1}$$

(a) [2 pt] Plot the given dataset and find the optimal $\boldsymbol{w}^*$ by inspection.

(b) [4 pt] Write down $\boldsymbol{y} \in \mathbb{R}^2$ and $\boldsymbol{X} \in \mathbb{R}^{2\times}$ which makes the following optimization equivalent to (1):

$$\min_{\boldsymbol{w}} \frac{1}{2} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2 \tag{2}$$

(c) [3 pt] Derive the general analytical solution for (2). Also plug in the values for the given dataset $\mathcal{D}$ and compute the solution numerically.

(d) [3 pt] There are several ways to compute this solution via PyTorch. Read the docs for the functions `torch.lstsq`, `torch.solve`, `torch.inverse`. Use all three approaches when completing the file `LinearRegression.py` and verify your answer.

(e) [6 pt] We are now given a dataset $\mathcal{D}' = \{(0,0),(1,1),(2,1)\}$ of pairs (x,y) with $x \in \mathbb{R}$ and $y \in \mathbb{R}$. We want to fit a quadratic model $\hat{y} = w_1 x^2 + w_2 x + w_3$ using (2). Specify the dimensions of the matrix $\boldsymbol{X}$ and the vector $\boldsymbol{y}$. Also write down explicitly the matrix and vector using the values in $\mathcal{D}'$. Find the optimal solution $\boldsymbol{w}^*$ and draw it together with the dataset into a plot.

(f) [2 pt] Specify $\boldsymbol{y}$ and $\boldsymbol{X}$ in `LinearRegression.py` to verify your answer for Problem 1e.

```
X = torch.Tensor(...fill_this...)
y = torch.Tensor(...fill_this...)
```

2. [20pt; Binary Logistic Regression] We are given a dataset $\mathcal{D} = \{(-1,-1),(1,1),(2,1)\}$ containing three pairs (x, y), where each $x \in \mathbb{R}$ denotes a real-valued point and $y \in \{-1,+1\}$ is the point's class label.

   Assuming the samples in the dataset $\mathcal{D}$ to be i.i.d. and using maximum likelihood, we want to train a logistic regression model parameterized by $\boldsymbol{w} = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \in \mathbb{R}^2$:

$$p(y \mid x) = \frac{1}{1 + \exp\left(-y\boldsymbol{w}^\top \begin{bmatrix} x \\ 1 \end{bmatrix}\right)} \tag{3}$$

   (a) [1pt] Instead of maximizing the likelihood we commonly minimize the negative log-likelihood ($p(\mathcal{D} \mid \boldsymbol{w})$). Write the objective for the model given in (3) (don't plug in the instances of $\mathcal{D}$. In other words, write an optimization problem like (1)).

   (b) [3pt] Compute the derivative of the negative log-likelihood objective which you specified in Problem 2a (don't plug in the instances of $\mathcal{D}$). Sketch a simple gradient-descent algorithm using pseudo-code (use $\boldsymbol{w}$ for the parameters, α for the learning rate, f for the objective function, and $g = \nabla_{\boldsymbol{w}} f$ for the gradient).

   (c) [5pt] Implement the algorithm by completing `LogisticRegression.py`. State the code that you implemented. What is the optimal solution $\boldsymbol{w}^*$ that your program found?

   (d) [3pt] If the third datapoint (2,1) was instead (10; 1), would this influence the bias $\boldsymbol{w}_2$ much? How about if we had used linear regression to fit $\mathcal{D}$ as opposed to logistic regression? Provide a reason for your answer.

   (e) [3pt] Instead of manually deriving and implementing the gradient we now want to take advantage of PyTorch auto-differentiation. Investigate `LogisticRegression2.py` and complete the update step using the instance named `optimizer`. What code did you add? If you compare the result of `LogisticRegression.py` with that of `LogisticRegression2.py` after an equal number of iterations, what do you realize?

   (f) [5pt] Instead of manually implementing the cost function, we now want to take advantage of available functions in PyTorch, specifically `torch.nn.BCEWithLogitsLoss` which expects targets to be $y \in \{0,1\}$. Consequently, you need to translate dataset $\mathcal{D}$ with $y \in \{-1,1\}$ to dataset $\mathcal{D}'$ with $y \in \{0,1\}$. Write the probabilities $p(y = 1 \mid x), p(y = 0 \mid x)$ and $p(y \mid x)$. if we use `torch.nn.BCEWithLogitsLoss`. Complete `LogisticRegression3.py` and compare the obtained result after 100 iterations to the one obtained in previous functions.

3. [29 pt; Support Vector Machine]

We are given a dataset $\mathcal{D} = \{(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}) : i = 1, 2, 3, 4\}$ of $(\boldsymbol{x}^{(1)}, \boldsymbol{y}^{(1)}) = \left(\begin{bmatrix}1\\0\end{bmatrix}, 1\right), (\boldsymbol{x}^{(2)}, \boldsymbol{y}^{(2)}) = \left(\begin{bmatrix}0\\1\end{bmatrix}, 1\right), (\boldsymbol{x}^{(3)}, \boldsymbol{y}^{(3)}) = \left(\begin{bmatrix}0\\0\end{bmatrix}, -1\right), (\boldsymbol{x}^{(4)}, \boldsymbol{y}^{(4)}) = \left(\begin{bmatrix}-1\\-1\end{bmatrix}, -1\right)$. We want to train the parameters $\boldsymbol{w} = \begin{bmatrix}w_1\\w_2\end{bmatrix} \in \mathbb{R}^2$ and the bias $b \in \mathbb{R}$ of a max-margin support vector machine (SVM) using: (for hyperparameter $C > 0$)

$$\min_{\boldsymbol{w}, b} \quad \frac{C}{2}\|\boldsymbol{w}\|_2^2 \tag{4a}$$

$$\text{s.t.} \quad y^{(i)}(\boldsymbol{w}^\top \boldsymbol{x}^{(i)} + b) \geq 1 \quad \forall i = 1, 2, 3, 4\,. \tag{4b}$$

(a) [5 pt] For the given data $\mathcal{D}$, how many constraints are part of the program in (4)? Specify all of them explicitly.

(b) [8 pt] For $b = 0, b = -1$, and $b = -2$, respectively, find the corresponding the set of feasible $\boldsymbol{w}$ and optimal $\boldsymbol{w}^*$ (if exists). Given only the three options $b \in \{0, -1, -2\}$, what is the optimal solution? Discuss whether a better solution exists.

(c) [5 pt] Draw the dataset in (x_1, x_2)-space using crosses $(\times)$ for the points belonging to class 1 and circles $(\circ)$ for the points belonging to class -1. Using your drawing, find the support vectors. Noting that those points for which the constraints hold with equality at the optimal solution, solve the resulting linear system w.r.t. $\boldsymbol{w}$ and b and draw the solution into (x_1, x_2)-space.

(d) [1 pt] What conditions do the datapoints have to fulfill such that the program in (4) has a feasible solution?

(e) [6 pt] In practice, for large datasets, it is hard to find the support vectors by inspection. A gradient based method is applicable. Using general notation, i.e., no plugging in $\mathcal{D}$, and introducing slack variables $\boldsymbol{\zeta} = (\zeta_i)_{i=1,\ldots,4}$ into (4), state the soft-margin problem with L_1 penalty on $\boldsymbol{\zeta}$ (including all constraints). Subsequently, reformulate this program into an unconstrained program. Finally obtain the gradient of this unconstrained program w.r.t. w (use $\frac{\partial}{\partial x}\max\{0, x\} = \mathbb{1}[x > 0]$). Compute the gradient at $w_1 = 2$, $w_2 = 2$ and $b = -1$, and discuss the impact of C and the relation between the max-margin and soft-margin SVMs.

(f) [4 pt] Complete `SVM.py` with $C = 1$ and verify your reply for the previous answer. What is the optimal solution $(\boldsymbol{w}, b)$ that your program found and what is the corresponding loss? Explain the solution and what you observe when running the program, as well as how to fix this issue.