Interpretable Machine Learning, Assignment 1

Chuan Lu

February 5, 2020

Code for this Assignment are contained in `optimizer.py` (algorithms), and `Assignment1.ipynb` (main process).

1. Problem 1.a and Problem 2.(a)

   Let $x_i, w \in \mathbb{R}^d$, and $\hat{x}_i = (1, x_i^\top)^\top, \hat{w} = (w_0, w^\top)^\top \in \mathbb{R}^{d+1}$. Denote

   $$p(y = 1 \mid \hat{x}_i) = \sigma(\hat{x}_i) \tag{1}$$

   then the loss function of logistic regression is

   $$\begin{aligned}\mathcal{L} &= -\sum_{i=1}^{n} \Big(y_i \log p(y=1 \mid \hat{x}_i) + (1-y_i)\log(1 - p(y=1 \mid \hat{x}_i)\Big) \\ &= -\sum_{i=1}^{n} \Big(y_i \log \sigma(\hat{x}_i) + (1-y_i)\log(1-\sigma(\hat{x}_i))\Big),\end{aligned} \tag{2}$$

   and the gradient is

   $$\begin{aligned}\frac{\partial}{\partial \hat{w}}\mathcal{L} &= -\sum_{i=1}^{n} \Big(\frac{y_i}{\sigma(\hat{x}_i)} \frac{\partial \sigma}{\partial \hat{w}}(\hat{x}_i) - \frac{1-y_i}{1-\sigma(\hat{x}_i)} \frac{\partial \sigma}{\partial \hat{w}}(\hat{x}_i)\Big) \\ &= \sum_{i=1}^{n} \frac{\sigma(\hat{x}_i) - y_i}{\sigma(\hat{x}_i)(1-\sigma(\hat{x}_i))} \frac{\partial \sigma}{\partial \hat{w}}(\hat{x}_i).\end{aligned} \tag{3}$$

   Notice that

   $$\frac{\partial \sigma}{\partial \hat{w}}(\hat{x}_i) = \frac{\partial}{\partial \hat{w}} \frac{1}{1+e^{-\hat{x}_i^\top \hat{w}}} = \frac{\hat{x}_i e^{-\hat{x}_i^\top \hat{w}}}{(1+e^{-\hat{x}_i^\top \hat{w}})^2} = \sigma(\hat{x}_i)(1-\sigma(\hat{x}_i))\hat{x}_i, \tag{4}$$

   then

   $$\frac{\partial}{\partial \hat{w}}\mathcal{L} = \sum_{i=1}^{n} (\sigma(\hat{x}_i) - y_i)\hat{x}_i. \tag{5}$$

   The code for the gradient descent algorithm is in `optimizer.py`. In the implementations, we use the mean of losses for each sample instead of the sum, for a fair comparison between the training loss and test loss. The coefficients $\hat{w}$ are initialized as zeros, and we use learning rate $lr = 0.01$ in training. The convergence plot is shown in Figure 1.

   The ROC-AUC and precision-recall curve on the test set is reported in Figure 2. The F1-score is 0.6426.

2. Problem 1.b and Problem 2.(b)

   The code for simulated annealing is in `optimizer.py`. For this problem, we use the same preprocessing process with Problem 1.a. We use a linearly decreasing scheduler for temperature, and the weights $\hat{w}$ are initialized with zeros. The convergence plot is shown in Figure 3.

   The ROC-AUC and precision-recall curve on the test set is reported in Figure 2. The F1-score is 0.6408.

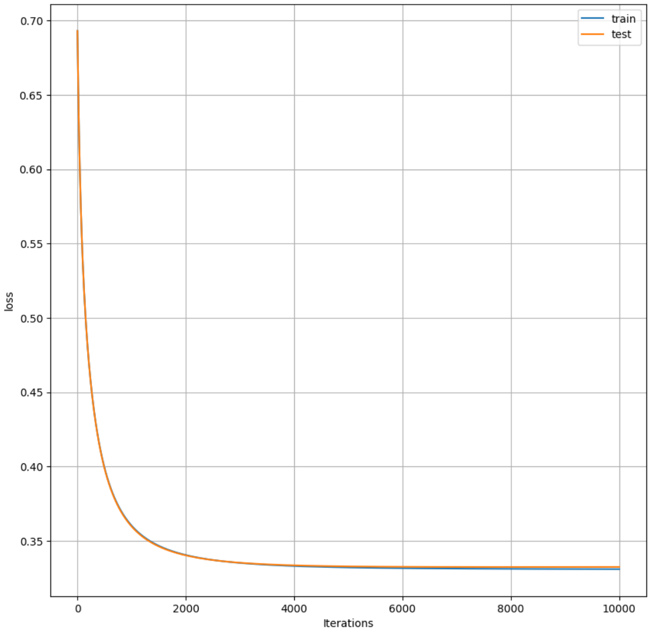

Figure 1: Convergence of loss v.s. number of iterations for Problem 1a, using gradient descent. The orange line shows the test loss, and the blue line represents the training loss.

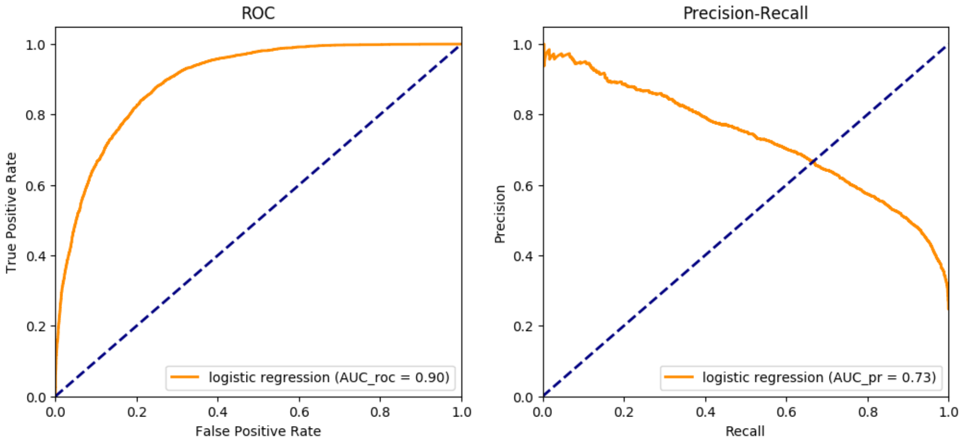

Figure 2: ROC, precision-recall curve, AUC for Problem 1.a (GD-solved logistic regression).

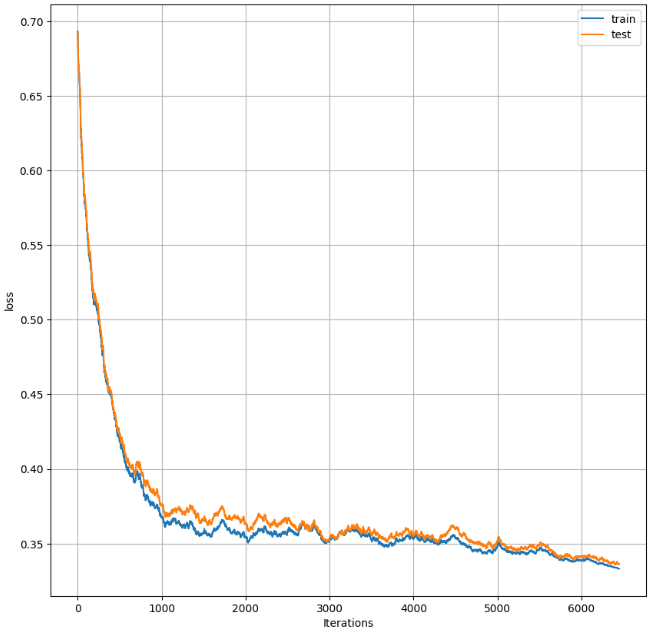

Figure 3: Convergence of loss v.s. number of iterations for Problem 1b, using simulated annealing. The orange line shows the test loss, and the blue line represents the training loss.

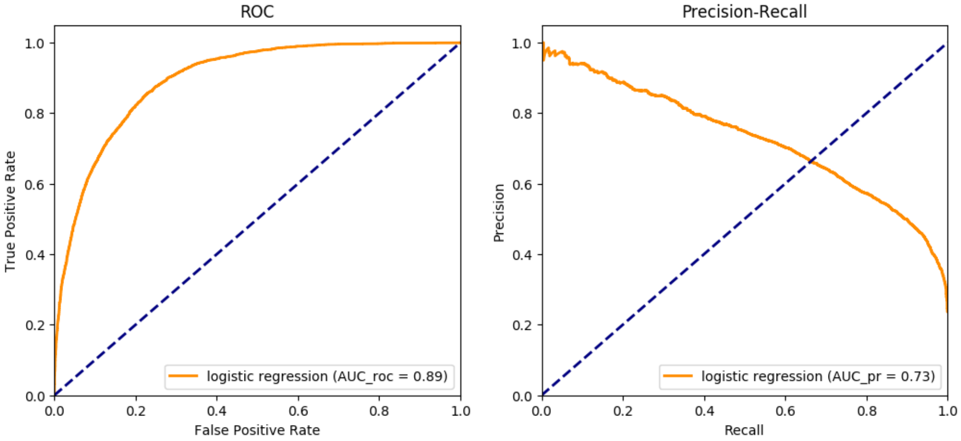

Figure 4: ROC, precision-recall curve, AUC for Problem 1.b (SA-solved logistic regression).

3. Problem 3

   We use the same preprocessing process as in Problem 1.a.

   The ROC and AUC are shown in Figure 5. Although ROC, AUC of Lasso are very similar with logistic regression, the number of non-zero coefficients is 17 instead of 89 for both solutions of logistic regression. This comes from the L_1 penalty of lasso, while logistic regression does not have any penalty on the number of non-zero coefficients.

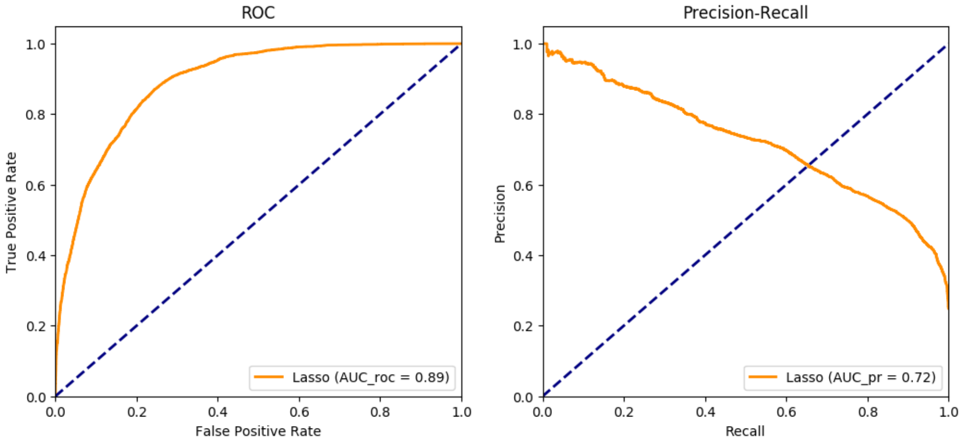

Figure 5: ROC and AUC for Problem 3 (Lasso).

4. Problem 4

   For this problem, we still use the same preprocessing process as in Problem 1.a.

   The feature importance plot for the 20 most important features is shown in Figure 6.

   The absolute value of weights (and corresponding rank) of each features are mostly consistent with the importance in Random Forest, with an exception `fnlwgt`, which is the most important in Random Forest but has less weights in logistic regression.

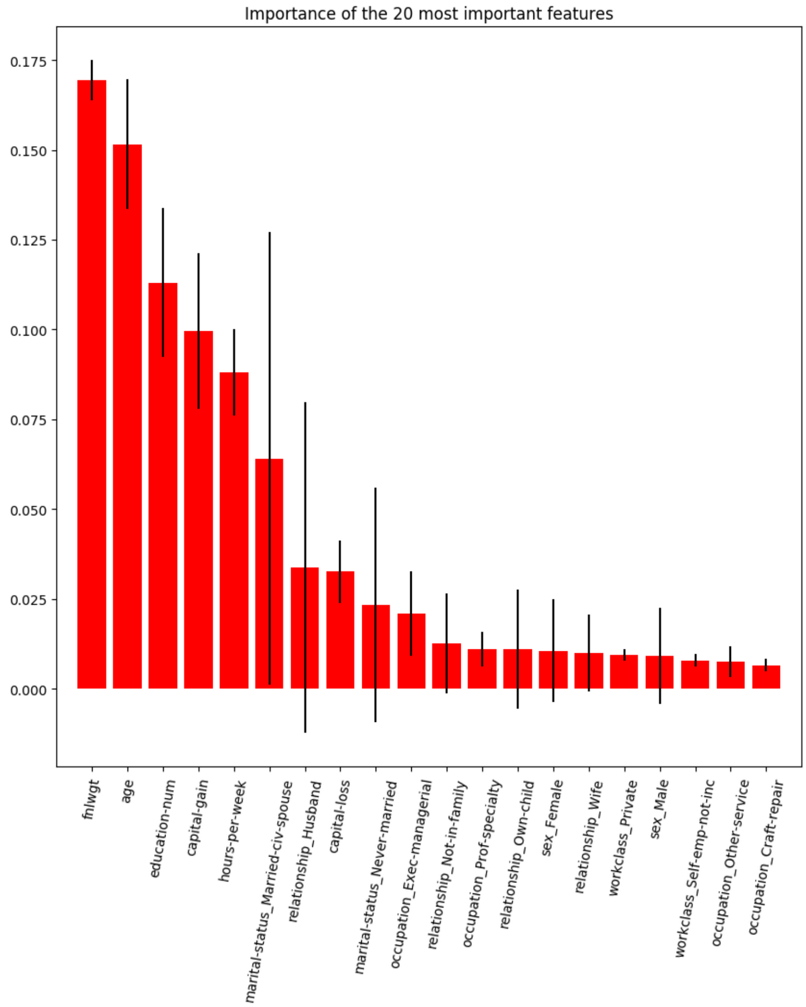

Figure 6: Feature importance for Problem 4 (Random Forest).