Labs
Machine Learning Course
Fall 2024

EPFL
School of Computer and Communication Sciences
Nicolas Flammarion & Martin Jaggi
www.epfl.ch/labs/mlo/machine-learning-cs-433

Problem Set 5, Oct 10, 2024 (Logistic Regression)

Goals. The goal of this exercise is to

- Do classification using linear regression.
- Implement and debug logistic regression using gradient descent.
- Compare it to linear regression.
- Implement Newton's method for logistic regression.

Setup, data and sample code. Obtain the folder `labs/ex05` of the course github repository

github.com/epfml/ML_course

We will use the dataset `height_weight_genders.csv` in this exercise, and we have provided sample code templates that already contain useful snippets of code required for this exercise.

Warm-up. For logistic regression, the following concepts are useful and provided here as a recap:

- The logistic function (a.k.a. sigmoid): $\sigma(x) := (1 + e^{-x})^{-1}$
- The derivative of the logistic function is: $\frac{d\sigma(x)}{dx} = \sigma(x)(1 - \sigma(x))$
- The logistic regression predicts $y \in \{0, 1\}$ with the following probability
$$p(y = 1|\boldsymbol{w}, \boldsymbol{x}) = \sigma(\boldsymbol{x}^\top \boldsymbol{w})$$
$$p(y = 0|\boldsymbol{w}, \boldsymbol{x}) = 1 - \sigma(\boldsymbol{x}^\top \boldsymbol{w}).$$
- The negative log-likelihood loss for the binary classification problem (a.k.a. cross entropy loss) is:
$$\mathcal{L}(\boldsymbol{w}) = -\frac{1}{N} \sum_{i=1}^{n} y_i \log \sigma(\boldsymbol{x}_i^\top \boldsymbol{w}) + (1 - y_i) \log(1 - \sigma(\boldsymbol{x}_i^\top \boldsymbol{w}))$$
where $y_i \in \{0, 1\}$ for all i.

1 Classification Using Linear Regression

We will try to use linear regression to do classification. Although this is not a good idea in general (as discussed in the class), it will work for simple data. Concretely, we will use the height-weight data from the first exercise to predict the binary valued gender (sex). For better visualization, we will randomly sample 200 data points from the data.

Exercise 1:

Classification using linear regression.

- Use `least_squares(y, tx)` from the previous exercise to compute the weights $\boldsymbol{w}$ on the height-weight data. Please COPY your previous implementation to the template file of this exercise `least_squares.py`.
- Visualize the data points and the decision boundary with `visualization()` as in Figure 1.

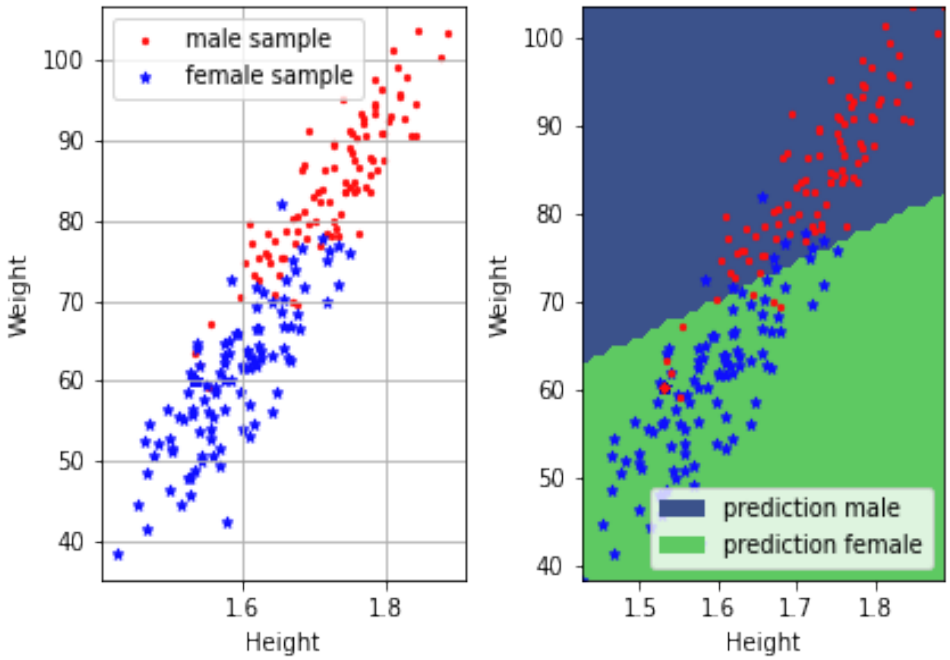

Figure 1: classification by least square.

2 Logistic Regression

Exercise 2:

Implement logistic regression using gradient descent.

- Fill in the notebook function `sigmoid()`.
- Fill in the two notebook functions `calculate_loss()` and `calculate_gradient()`. The first function should return the *negative* log-likelihood loss, while the second function should return the gradient of this loss w.r.t. the parameters $\boldsymbol{w}$.
- Implement gradient descent `learning_by_gradient_descent()` for logistic regression. You should calculate the loss and its gradient w.r.t. $\boldsymbol{w}$. Now, you can update the weights $\boldsymbol{w}$, and the function should return the loss and these updated weights $\boldsymbol{w}$.
- Plot the predictions to get a visualization similar to the right part of Figure 1. Check if you get similar or different results.

Now that we have gradient descent, we can easily implement Newton's method.

Exercise 3:

Newton's method for logistic regression

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \gamma \nabla^2 \mathcal{L}(\boldsymbol{w}_t)^{-1} \nabla \mathcal{L}(\boldsymbol{w}_t).$$

- Fill in the notebook function `calculate_hessian()`. Now, complete `logistic_regression()` by using `calculate_loss()`, `calculate_gradient()` and `calculate_hessian()`. This function `logistic_regression()` should return the loss, gradient, and Hessian. Note that it should not update nor return the parameters $\boldsymbol{w}$.
- Your gradient descent code can now be turned into a Newton's method algorithm to find the optimal parameters $\boldsymbol{w}$. Please fill in the notebook function `learning_by_newton_method()`, which does a single parameter update. The function should return the loss and the updated parameters. Check the convergence speed comparison with previous results.

  hint: Use np.linalg.solve instead of directly computing the inverse $\nabla^2 \mathcal{L}(\boldsymbol{w}_t)^{-1}$.

Exercise 4:

Penalized logistic regression.

- Fill in the notebook function `penalized_logistic_regression()`. This requires you to add the regularization term $\lambda\|\boldsymbol{w}\|^2$. As a sanity check, set λ to a very small value and see if it gives the same result as before. Once complete, please fill in the notebook function `learning_by_penalized_gradient()`, and increase the value of λ and check whether (the norm of) $\boldsymbol{w}$ is shrinking or not. (Note: you should **not** include the penalty term in your loss value calculation).
- Check if this gives the same answer as gradient descent. To debug, print the function value and the norm of the gradient in every iteration. All these values should decrease in every iteration.