Logistic Regression

Evelyn Yosiana / 13522083

1. Secara garis besar, ada dua proses yang dilakukan oleh logistic regression, yaitu pembuatan linear model yang mirip dengan linear regression dan pengubahan nilai menjadi probabilitas dengan sigmoid function.

   Proses train: untuk proses train, pada dasarnya class ini (dalam kode saya diberi nama Logistic_Regression_Selfmade) bertujuan untuk membentuk model linear dengan persamaan sebagai berikut

   $$y = w1.x1 + w2.x2 + \dots + wnxn + bias$$

   atau dengan kata lain, tujuan dari proses train ini yaitu mencari nilai w dan bias. Perhitungan manual dengan contoh data sederhana dapat dilihat pada lampiran.

   Proses predict: untuk proses ini, tiap data yang akan diprediksi (X_test) dihitung nilai y nya menggunakan formula pada penjelasan proses train dengan menggunakan nilai w dan bias yang sudah didapatkan pada proses train. Kemudian, nilai y yang telah didapatkan, dimasukkan ke dalam sigmoid function yang bertujuan untuk mengubah nilai y tersebut dalam range 0 sampai 1 (probabilitas). Jika hasil sigmoid function dari y tersebut kurang dari threshold (dalam kode saya threshold yang digunakan adalah 0.5 karena merupakan binary classification) maka label prediksinya akan bernilai 0, dan jika sebaliknya akan bernilai 1. Proses tersebut diulangi untuk seluruh data di X_test.

   Parameter:

   - learning rate: laju belajar untuk mengontroll seberapa besar perubahan weight dan bias.
   - n_iteration: jumlah iterasi.
   - regularization: untuk menambah penalti yang bertujuan untuk mengurangi overfitting.
   - reg_lamda: koefisien penalti untuk mengontrol seberapa besar penalti pada weight pada proses regularization.
   - loss function: untuk menghitung kesalahan antara y_pred dan y_test (dalam kode saya terdapat cross entropy, focal, dan logit)

4. Pada model ini, hasil perbandingan model yang saya buat dan model dari library kurang sama. Hal ini dapat disebabkan oleh beberapa faktor, antara lain:
   - **Perbedaan inisialisasi**: kode yang saya buat menginisialisasi weight dengan 0, sedangkan kode dalam library (misalnya scikit) menggunakan random initialization.

- **Perbedaan learning rate**: kode yang saya buat menggunakan learning rate secara manual (di parameter atau by default = 0.01), sedangkan library (misalnya scikit) menggunakan optimization yang lebih advanced yang dapat mengatur learning rate untuk hasil yang optimal.
- **Clipping dan penggunaan epsilon**: dalam kode yang saya buat, terdapat proses clipping dan penambahan epsilon (angka yang sangat kecil) untuk menghindari error (misalnya log(0)), sedangkan kode dalam library sudah dioptimasi sedemikian rupa sehingga dapat menghasilkan perhitungan yang lebih presisi.
- **Stopping criteria**: kode yang saya buat menggunakan iteration sebagai batas iterasi, sedangkan kode dalam library menggunakan batas iterasi berdasarkan konvergensi.

5. Improvement yang dapat saya lakukan antara lain:
    - **Mencoba optimizer lain**, misalnya adam untuk menghindari false local minima.
    - Mengombinasikan L1 dan L2 menjadi **elastic net** untuk mendapatkan keuntungan dari kedua metode tersebut.
    - Mengubah adjustment learning rate manual menjadi **adaptive learning rate** sehingga seiring dengan bertambahnya iterasi bisa membantu konvergensi yang lebih stabil.
    - Menerapkan **early stopping** untuk mengurangi overfitting berdasarkan perubahan loss (berhenti saat loss sudah tidak turun secara signifikan).
    - **Scaling data** karena logistic regression sangat sensitif terhadap skala fitur.
    - **Hyperparameter tuning** untuk mendapatkan kombinasi parameter yang optimal (salah satunya dengan menggunakan metode grid search atau library optuna).
    - Menggunakan **confusion metrics** untuk model evaluation.

Lampiran

Contoh perhitungan manual: