

Cyclic Coordinate Descent for Logistic Regression with Lasso regularization

Patryk Prusak

supervisor

XYZ

Warsaw University of Technology

March 30, 2025

Advanced Machine Learning Course

Contents

1	Methodology	2
1.1	Selection and generation of datasets	2
1.2	Details about algorithm implementation and applied optimizations	3
2	Impact of dataset parameters: n, p, d, g on the performance of LogRegCCD algorithm	5
3	Benchmark of LogRegCCD with LogisticRegression algorithm	6
3.1	Performance of algorithms regarding different metrics	6
3.2	Values of coefficients obtained in these two methods	6
4	Discussion about correctness of the LogRegCCD algorithm	6
4.1	Performance of the algorithm at $\lambda=0$	6
4.2	Likelihood function values and coefficient values depending on iteration	6
4.3	Comparison with ready implementation of logistic regression with L1 penalty . . .	6

1 Methodology

1.1 Selection and generation of datasets

The implemented Cyclic Coordinate Descent for Logistic Regression with Lasso regularization (LogRegCCD) algorithm was tested on both real and synthetic datasets to evaluate its performance given various metrics such as accuracy, precision, recall, F1 score, balanced accuracy and ROC AUC [1, 2, 3] provided by the scikit-learn [4] Python [5] library.

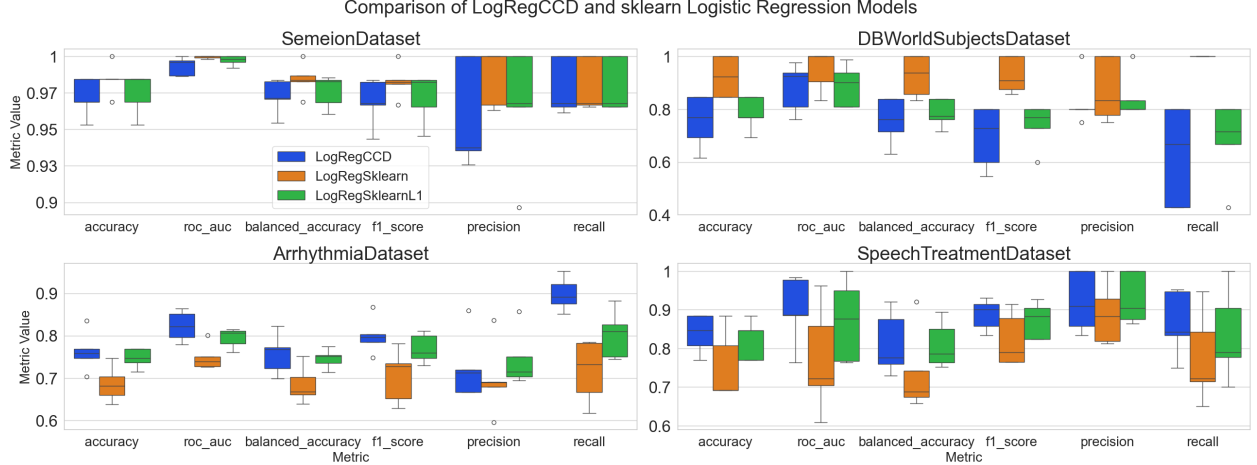


Figure 1: Boxplots of metrics for real datasets

The synthetic datasets have been created with various values of the following parameters: class prior probability p , number of samples n , number of features d , and the covariance matrix g parameter defined as $S[i, j] = g^{|i-j|}$. We have examined all unique combinations of the following parameter values: $p \in \{0.3, 0.4, 0.5, 0.6, 0.7\}$, $n \in \{1000, 1500, 2000\}$, $d \in \{2, 5, 10, 30\}$ and $g \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$. The synthetic datasets were generated according to the task description, that is:

- Generate binary class variable ($Y=0$ or $Y=1$) from Bernoulli distribution with class prior probability p .
- Generate feature vector X , such that for class $Y=0$, X follows d -dimensional multivariate normal distribution with mean vector $\mu_0 = (0, \dots, 0)$ and covariance matrix S where $S[i, j] = g^{|i-j|}$.
- For class $Y=1$, X follows d -dimensional multivariate normal distribution with mean vector $\mu_1 = (1, \frac{1}{2}, \frac{1}{3}, \dots, \frac{1}{d})$ and the same covariance matrix S .
- Generate n observations using the above steps.

1.2 Details about algorithm implementation and applied optimizations

Logistic Regression is a machine learning method capable of binary classification. It predicts the probability of an outcome by computing the linear combination of input features and weights (Formula 1), then passing it through the sigmoid function presented in Formula 2.

$$z = w_0 + w_1x_1 + w_2x_2 + \dots + w_nx_n = \mathbf{w}^T \mathbf{x} + b \quad (1)$$

Where x denotes input feature vector, while x_1, \dots, x_n are the elements of that vector, w denotes model weights vector and b is the bias term.

$$\sigma(z) = \frac{1}{1 + e^{-z}} \quad (2)$$

The output of the sigmoid function in range $[0, 1]$ denotes the probability that given feature vector x belongs to the positive class. What follows the prediction rule is based on the output of the sigmoid function, if it's larger than a set threshold, such as 0.5, we assign the sample to class 1, otherwise assign to class 0.

To fit the model to the training data one needs to minimize the loss function, in this case Binary Cross-Entropy defined in Formula 3.

$$\mathcal{L} = -\frac{1}{m} \sum_{i=1}^m [y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log (1 - \hat{y}^{(i)})] \quad (3)$$

Where m denotes the number of training examples, $y^{(i)}$ is the class label and $\hat{y}^{(i)}$ is the predicted probability. The weights of the model need to be optimized to find the proper fit, this can be achieved by standard gradient descent algorithm. The weights are updated according to the following formulas (Formula 4, Formula 5):

$$w_j := w_j - \alpha \frac{\partial \mathcal{L}}{\partial w_j} \quad (4)$$

$$b := b - \alpha \frac{\partial \mathcal{L}}{\partial b} \quad (5)$$

Where α is the learning rate, the higher the value the more aggressive weight updates and $\frac{\partial \mathcal{L}}{\partial w_j}$ is a gradient with respect to weight w_j .

One of the methods to prevent overfitting of the model to the training data is Lasso Regularization. Overfitting describes the situation when the trained model can predict samples from the training set very well but struggles on the test set. The loss function with Lasso regularization is defined in Formula 6.

$$\mathcal{L}_{\text{lasso}} = -\frac{1}{m} \sum_{i=1}^m [y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log (1 - \hat{y}^{(i)})] + \lambda \sum_{j=1}^n |w_j| \quad (6)$$

Where m is the number of training samples, $y^{(i)}$ is the class label, $\hat{y}^{(i)}$ is the predicted probability, λ denotes regularization strength.

In essence during the training process, the model will also minimize the absolute sum of the coefficients in addition to the loss function. This will result in some of the weights being set to zero,

effectively reducing the number of features the model is trained on. This can be useful in situations where the number of features is very large and some of them are irrelevant to the prediction task.

Now, to use the Cyclic Coordinate Descent instead of the standard Gradient Descent one needs to minimize the $\mathcal{L}_{\text{lasso}}$ using a different algorithm for updating model weights. However the authors of the 2010 publication entitled *Regularization Paths for Generalized Linear Models via Coordinate Descent* [6] present a more sophisticated approach with certain optimizations.

The logistic regression with lasso regularization log-likelihood function is approxiamted using a quadratic approximation presented in Formula 7. This converts the problem into a penalized weighted least squares. The authors also use a regularization path that starts from largest λ where $\beta = 0$ and decreases λ gradually, using previous solutions as warm starts. Instead of computing gradients from scratch with each iteration, the authors propose to use covariance updates (Formula 8). This in turn allows for a more efficient computation of the gradients. For each feature, the optimization problem simplifies to a minimization problem presented in Formula 9.

$$\ell_Q(\beta_0, \beta) = -\frac{1}{2N} \sum_{i=1}^N w_i (z_i - \beta_0 - x_i^T \beta)^2 + C \quad (7)$$

$$\sum_{i=1}^N x_{ij} r_i = \langle x_j, y \rangle - \sum_{k: \beta_k \neq 0} \langle x_j, x_k \rangle \beta_k \quad (8)$$

$$\min_{\beta_j} \left[\frac{1}{2} \sum_{i=1}^N w_i \left(z_i - \beta_0 - \sum_{k \neq j} x_{ik} \beta_k - x_{ij} \beta_j \right)^2 + \lambda |\beta_j| \right] \quad (9)$$

The algorithm for the cyclic coordinate descent for a given feature j is as follows:

1. Compute partial residuals (excluding β_j)

$$r_i = z_i - (\beta_0 + \sum_{k \neq j} x_{ik} \beta_k)$$

2. Compute the gradient component ρ_j

$$\rho_j = \sum_{i=1}^N w_i x_{ij} r_i$$

3. Apply soft-thresholding for L1 regularization

$$\beta_j = \frac{S(\rho_j, \lambda)}{\sum_{i=1}^N w_i x_{ij}^2}$$

$$S(z, \lambda) = \text{sign}(z) \cdot \max(|z| - \lambda, 0)$$

4. Update β_0 that is not regularized

$$\beta_0 = \frac{\sum_{i=1}^N w_i (z_i - x_i^T \beta)}{\sum_{i=1}^N w_i}$$

From a high level overview the presented algorithm consists of:

1. Outer Loop: Decrease λ along a regularization path.
2. Middle Loop: Update the quadratic approximation using the current (β_0, β) .
3. Inner Loop: Perform coordinate descent on the penalized weighted least squares problem.

2 Impact of dataset parameters: n,p,d,g on the performance of LogRegCCD algorithm

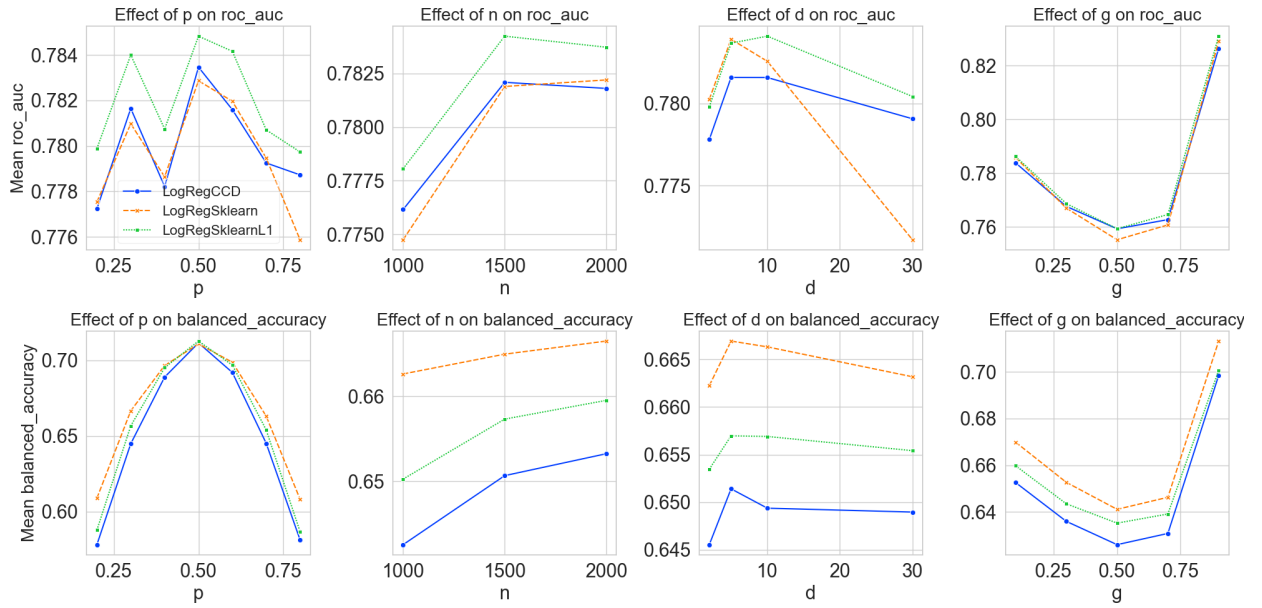


Figure 2: impact of synthetic dataset parameters on the performance of LogRegCCD algorithm

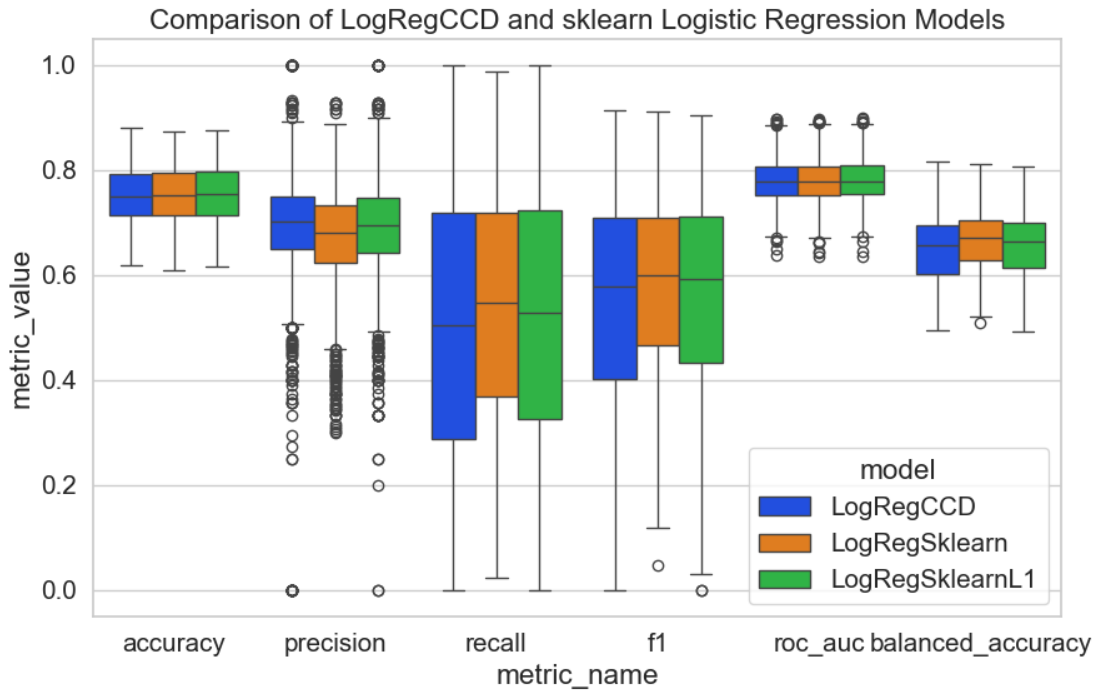


Figure 3: Comparison of LogRegCCD and LogisticRegression on synthetic dataset

3 Benchmark of LogRegCCD with LogisticRegression algorithm

3.1 Performance of algorithms regarding different metrics

3.2 Values of coefficients obtained in these two methods

4 Discussion about correctness of the LogRegCCD algorithm

4.1 Performance of the algorithm at lambda=0

4.2 Likelihood function values and coefficient values depending on iteration

4.3 Comparison with ready implementation of logistic regression with L1 penalty

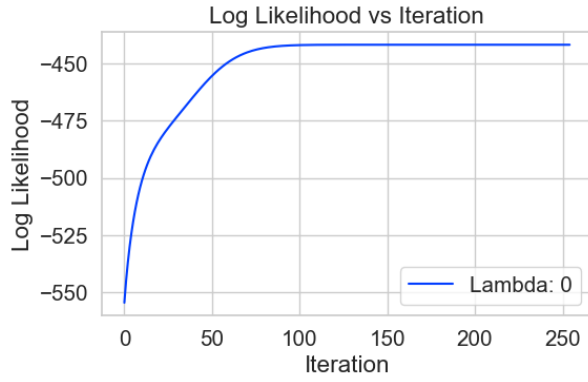


Figure 4: Log likelihood function values depending on iteration for synthetic dataset with $\lambda = 0$

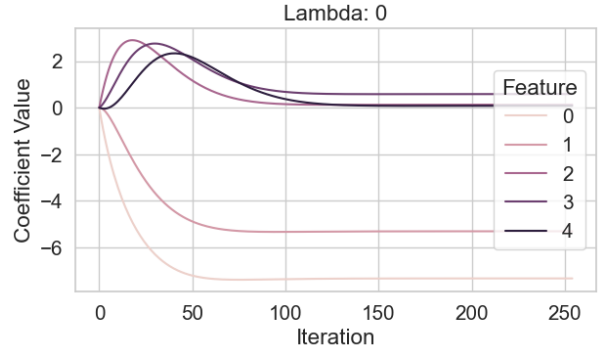


Figure 5: Coefficient values depending on iteration for synthetic dataset with $\lambda = 0$

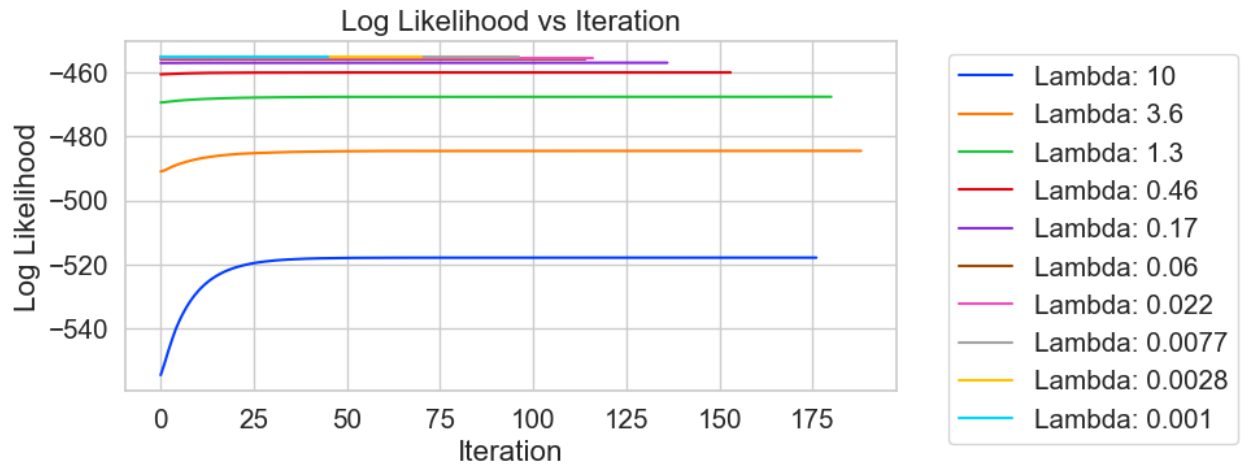


Figure 6: Log likelihood function values depending on iteration for synthetic dataset

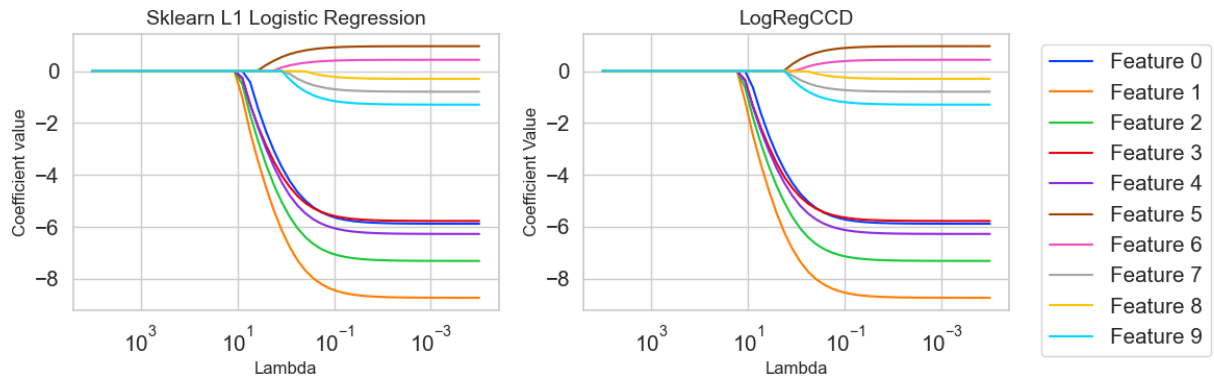


Figure 7: Comparison of LogRegCCD and LogisticRegression on synthetic dataset with redundant features

List of Figures

1	Boxplots of metrics for real datasets	2
2	impact of synthetic dataset parameters on the performance of LogRegCCD algorithm	5
3	Comparison of LogRegCCD and LogisticRegression on synthetic dataset	6
4	Log likelihood function values depending on iteration for synthetic dataset with $\lambda = 0$	7
5	Coefficient values depending on iteration for synthetic dataset with $\lambda = 0$	7
6	Log likelihood function values depending on iteration for synthetic dataset	7
7	Comparison of LogRegCCD and LogisticRegression on synthetic dataset with redundant features	8

List of Tables

References

- [1] David Martin Powers. “Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation”. In: *Journal of Machine Learning Technologies* 2.1 (2011), pp. 37–63.
- [2] Kay H Brodersen et al. “The balanced accuracy and its posterior distribution”. In: *20th International Conference on Pattern Recognition* (2010), pp. 3121–3124.
- [3] Tom Fawcett. *An introduction to ROC analysis*. Vol. 27. 8. Elsevier, 2006, pp. 861–874.
- [4] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.
- [5] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009. ISBN: 1441412697.
- [6] Jerome H. Friedman, Trevor Hastie, and Robert Tibshirani. “Regularization Paths for Generalized Linear Models via Coordinate Descent”. In: *Journal of Statistical Software* 33.1 (2010), pp. 1–22. DOI: 10.18637/jss.v033.i01. URL: <https://doi.org/10.18637/jss.v033.i01>.