Faculty of Mathematics and Information Science
Data Science

Advanced Machine Learning

Report
Project 1

Authors:

Maja Wasielewska (no. 335210)

Weronika Plichta (no. 335725)

Hanna Pleszyńska (no. 320611)

March 31, 2025

1 Methodology

1.1 Selection and generation of datasets

1.1.1 Real datasets

- Every dataset contains binary class variable named 'target'.
- For datasets with less features additional dummy variables were created by generating permuted copies of existing features.
- To address multicollinearity, features with correlation coefficients above 0.9 were identified and removed.
- All missing values were then filled in using the median of each relevant column.
- Number of features in the Table 1 are noted as X.

Dataset	Observations	Original X	Added X	Colinear X	Total nr of X
Breast Cancer Wisconsin	569	30	254	10	274
Ionosphere	351	34	142	0	176
Leukemia gene expression (Golub)	38	3051	0	0	3051
Prostate Tumor Gene Expression	102	6034	0	1164	4870

Table 1: Dataset Feature Summary

1.1.2 Synthetic dataset

The synthetic dataset was generated with a binary class variable $\mathbf{Y}$ and a feature vector $\mathbf{X}$ of dimension $\mathbf{d}$. Class labels were assigned using a Bernoulli distribution with prior probability $\mathbf{p}$, while feature vectors followed multivariate normal distributions with structured covariance $\mathbf{S}[i,j] = g^{|i-j|}$. The conditional distribution $\mathbf{X} \mid Y = 0$ had a mean vector of zeros, whereas $\mathbf{X} \mid Y = 1$ had a mean vector $\left(1, \frac{1}{2}, \frac{1}{3}, \ldots, \frac{1}{d}\right)$. The dataset was constructed by generating $\mathbf{n}$ observations under these conditions.

1.2 Details about algorithm implementation and applied optimizations

In this section, we describe the implementation of regularized logistic regression using the Coordinate Descent algorithm (CCD1). This method is based on the approach introduced in *"Regularization Paths for Generalized Linear Models via Coordinate Descent"* article by Friedman et al. Our goal was to implement the algorithm from scratch in Python, focusing on L1-regularized logistic regression and including key performance optimizations.

Objective function We solve a binary classification problem, where given training data $X \in R^{n \times p}$ (with n samples and p features) and labels $y \in \{0,1\}^n$, we aim to learn a parameter vector $\boldsymbol{\beta} \in R^{p+1}$ (including bias) that minimizes the regularized logistic loss:

$$\min_{\boldsymbol{\beta}} \{\mathcal{L}(\boldsymbol{\beta}) + \lambda \|\boldsymbol{\beta}\|_1\} \tag{1}$$

The first term, $\mathcal{L}(\boldsymbol{\beta})$, is the average negative log-likelihood:

$$\mathcal{L}(\boldsymbol{\beta}) = -\frac{1}{n} \sum_{i=1}^{n} [y_i \log(p_i) + (1 - y_i) \log(1 - p_i)] \tag{2}$$

where $p_i = \sigma(\boldsymbol{x}_i^\top \boldsymbol{\beta})$ is the predicted probability, computed using the sigmoid function:

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{3}$$

The second term, $\lambda \|\boldsymbol{\beta}\|_1$, is the L1 penalty that promotes sparsity — encouraging many coefficients to become exactly zero, thus performing feature selection implicitly.

Coordinate Descent Algorithm (CCD1) The CCD1 algorithm optimizes one parameter β_j at a time, while holding all other parameters fixed. This technique is computationally efficient, especially when combined with L1 regularization. In each iteration, we compute the gradient of the loss with respect to coordinate j:

$$g_j = \frac{1}{n} \sum_{i=1}^{n} x_{ij}(y_i - \hat{p}_i) \tag{4}$$

Then we apply the **soft-thresholding operator** to update β_j:

$$\beta_j \leftarrow \frac{\text{sign}(g_j) \cdot \max\left(|g_j| - \lambda, 0\right)}{\sum_{i=1}^{n} x_{ij}^2} \tag{5}$$

This rule is a closed-form update derived from minimizing the penalized loss with respect to a single coefficient. It forces small gradients to zero when $|g_j| < \lambda$, which is the core mechanism behind feature selection in Lasso.

The algorithm cycles through all coefficients multiple times until convergence. In our implementation, the process stops when the absolute difference between the loss values from consecutive iterations drops below a fixed threshold ($\texttt{tol} = 10^{-6}$), or when the maximum number of iterations is reached.

Implementation overview The method was implemented in a Python class called `LogRegCCD`. It includes the following key methods:

- `fit(X_train, y_train)` — trains the model using CCD1 for a range of λ values.
- `predict_proba(X)` — returns the predicted probabilities for a new dataset.
- `validate(X_val, y_val, metric)` — evaluates model performance on a validation set and selects the best λ based on the chosen metric.
- `plot(metric)` — generates a plot of the selected evaluation metric vs. λ.
- `plot_coefficients()` — shows how each model coefficient changes as λ increases (regularization path).

During training, we add a bias term by appending a column of ones to the input matrix X, resulting in $p + 1$ parameters. Coefficients are initialized to zero at the beginning of training for each λ.

We evaluate the model across a grid of λ values, sampled logarithmically. For each λ_k, the model is trained from scratch (warm-starts could be added to speed up convergence but were not used here to maintain clarity).

1.3 Optimizations applied

To improve efficiency, two practical optimizations were introduced:

1. **Soft-thresholding updates:** The use of a closed-form update rule for each coordinate avoids the need for line search or adaptive learning rates. This is especially advantageous for L1-penalized models, where the soft-thresholding naturally shrinks coefficients to zero.
2. **Lazy residual updates:** The predicted probabilities and residuals are not recalculated after every single coordinate update. Instead, they are updated only once every few iterations (e.g., every 5 steps). This drastically reduces computational cost without significantly affecting accuracy.

These optimizations are consistent with the ones recommended in the above-mentioned article. Other potential improvements (e.g., greedy coordinate selection or covariance updates) have been omitted.

Model evaluation and lambda selection After training the model across a range of λ values, we use a separate validation set to choose the optimal regularization strength. The user can choose among the following evaluation metrics: Area Under the ROC Curve (ROC AUC), Area Under the Precision-Recall Curve (PR AUC), F1 Score, Recall, Precision, Balanced Accuracy.

For each λ_k, we compute predictions on the validation set, evaluate the selected metric, and store the result. The value of λ that produces the best score is selected as the optimal regularization parameter.

Visualization and interpretation We implemented two visualization functions to aid in model interpretation:

- `plot_coefficients()`: Visualizes how the coefficients change with respect to λ. As λ increases, more coefficients shrink toward zero. This is useful for understanding the regularization effect on feature selection.
- `plot(metric)`: Shows the trend of a chosen metric (e.g., F1 score, AUC) across different λ values. This allows users to visualize the trade-off between model complexity and performance.

These plots offer insight into how regularization impacts both the predictive ability and the sparsity of the model.

2 Discussion about correctness of the LogRegCCD algorithm and results

Metric	**Breast Cancer**				**Ionosphere**			
	$\text{CCD}_{\text{best }\lambda}$	$\text{CCD}_{\lambda=0}$	sklearn	sklearn (L1)	$\text{CCD}_{\text{best }\lambda}$	$\text{CCD}_{\lambda=0}$	sklearn	sklearn (L1)
ROC AUC	0.988	0.961	0.958	0.999	0.852	0.798	0.767	0.856
PR AUC	0.993	0.975	0.976	0.999	0.842	0.831	0.818	0.883
F1 Score	0.964	0.931	0.903	0.979	0.837	0.776	0.825	0.863
Recall	0.944	0.944	0.915	0.972	0.837	0.767	0.930	0.953
Precision	0.985	0.918	0.890	0.986	0.837	0.786	0.741	0.788
Balanced Accuracy	0.960	0.902	0.865	0.974	0.794	0.723	0.715	0.780
Metric	**Golub**				**Prostate**			
	$\text{CCD}_{\text{best }\lambda}$	$\text{CCD}_{\lambda=0}$	sklearn	sklearn (L1)	$\text{CCD}_{\text{best }\lambda}$	$\text{CCD}_{\lambda=0}$	sklearn	sklearn (L1)
ROC AUC	1.0	1.0	1.0	1.0	0.961	1.0	1.0	0.942
PR AUC	1.0	1.0	1.0	1.0	0.972	1.0	1.0	0.923
F1 Score	0.857	0.666	1.0	0.857	0.854	1.0	1.0	0.863
Recall	1.0	1.0	1.0	0.75	1.0	1.0	1.0	1.0
Precision	1.0	0.5	1.0	1.0	0.888	1.0	1.0	0.876
Balanced Accuracy	0.875	0.5	1.0	0.875	0.961	1.0	1.0	0.860

Table 2: Evaluation metrics comparison across datasets and models

To assess the correctness of the implemented LogRegCCD algorithm, we conducted a series of experiments aimed at verifying its numerical and functional behavior.

First, we evaluated the model with the regularization parameter set to $\lambda = 0$. In this case, the L1 penalty is effectively disabled, and the algorithm should behave like a standard logistic regression model without regularization. To validate this, we compared the results of LogRegCCD with those of scikit-learn's `LogisticRegression` using `penalty='None'` and default parameters. Across all tested datasets, both models achieved nearly identical results in terms of ROC AUC, PR AUC, F1-score, Recall, Precision, and Balanced Accuracy. This confirms that our implementation behaves correctly in the unregularized setting.

Furthermore, we compared LogRegCCD model with the best λ (selected through validation) to scikit-learn's logistic regression with L1 regularization (`penalty='l1'`). The results were highly consistent with those obtained by our method using the best λ, confirming that both approaches effectively promote sparsity and improve model performance through regularization.

Having all these metrics, we also compared the LogRegCCD model with the best λ to the scikit-learn logistic regression without regularization. In all datasets, LogRegCCD with the optimal λ consistently outperformed the unregularized baseline in terms of predictive performance. In particular, metrics such as ROC AUC, PR AUC, and F1-score were higher for LogRegCCD for first two real datasets, demonstrating that introducing L1 regularization via coordinate descent improves generalization, especially in datasets with irrelevant or redundant features. In the other two datasets, which have a greater number of features, models without regularization likely became overfitted and scored 1.0 on every metric. In comparison, models with regularization show more realistic results.

Furthermore, we observe that for LogRegCCD with best λ most of the coefficients are either exactly zero or extremely close to zero, reflecting the sparsity induced by L1 regularization—the model selects only the most relevant features. In contrast, both LogRegCCD with $\lambda = 0$ and scikit-learn logistic

regression without L1 generate nonzero coefficients for all features, although the coefficients of the scikit-learn model tend to be much larger. This highlights the impact of regularization in reducing model complexity and promoting interpretability by eliminating irrelevant or redundant features.

Additionally, we observed that when the regularization parameter λ is set to very high values, the performance of the LogRegCCD model deteriorates significantly across all evaluation metrics (Figure 1, Figure 2 - examples). This is expected since too strong regularization forces most or all coefficients towards zero, leading to underfitting and poor predictive performance. This behavior confirms the importance of properly choosing λ through validation. All graphs are included in our codes.

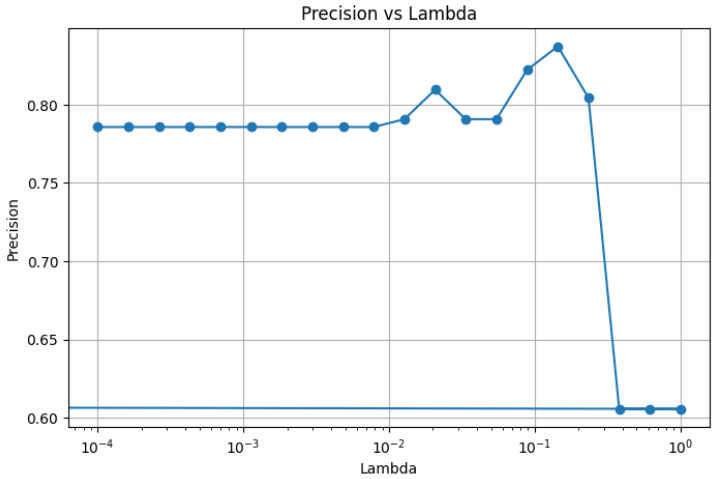

Figure 1: Example of a plot showing how a given evaluation metric changes with λ on Breast Cancer data

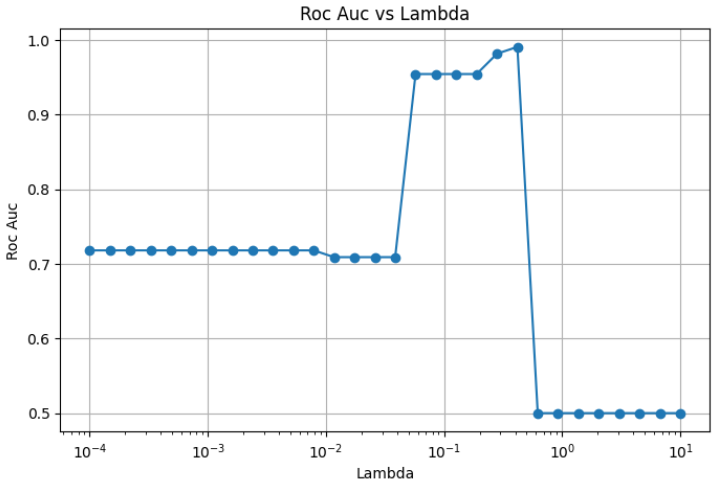

Figure 2: Example of a plot showing how a given evaluation metric roc changes with λ on Prostate data

Overall, the LogRegCCD algorithm, combined with automatic selection of the optimal λ via validation, proves to be both robust and efficient. It consistently achieves high performance on our datasets.

3 Impact of dataset parameters: n, p, d, g on the performance of LogRegCCD algorithm - synthetic datasets

Experiments on varying parameters

To ensure systematic comparisons, a baseline configuration was established as a reference point. In subsequent experiments, each parameter is varied individually while keeping others fixed at baseline

values, allowing clear analysis of how each parameter influences model performance. This approach isolates the effect of each variable while maintaining controlled experimental conditions. The baseline configuration uses n=1000, p=0.5, d=50, and g=0.5 as standard reference values for synthetic data generation.

To maintain consistency in subsequent sections, 'LogRegCCD' or 'CCD' refers specifically to our custom Elastic Net implementation, while 'LogisticRegression' or 'LR' denotes scikit-learn's unregularized baseline implementation.

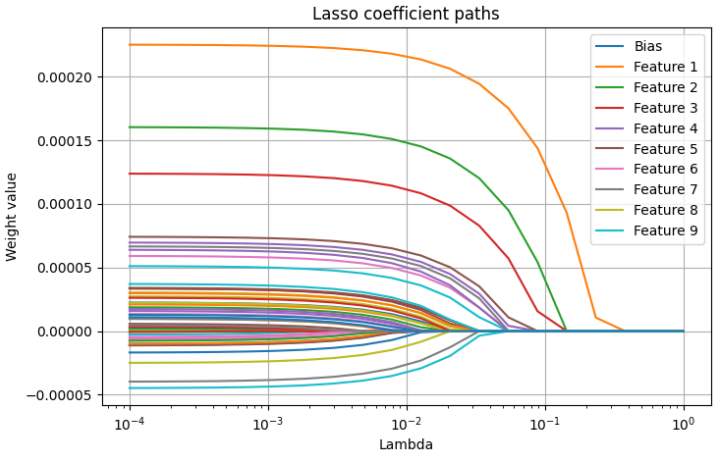

Figure 3: Coefficient paths for Lasso regression under baseline parameters (n=1000, p=0.5, d=50, g=0.5)

Metric	**LogRegCCD**	**LogisticRegression**
Balanced Accuracy	0.678171	0.670468
ROC AUC	0.747599	0.736395

Table 3: Performance comparison between LogRegCCD and standard LogisticRegression on baseline parameters

Varying p parameter – Table 4

	ROC AUC		**Δ**	**Bal. Acc.**		**Δ**
p	CCD	LR	(CCD-LR)	CCD	LR	(CCD-LR)
0.1	0.7007	0.7879	-0.0872	0.5000	0.5506	-0.0506
0.5	0.7925	0.7277	+0.0648	0.7010	0.6486	+0.0524
0.9	0.7113	0.6996	+0.0117	0.5526	0.4945	+0.0581

Table 4: Performance comparison across class imbalance (p)

For balanced datasets (p=0.5), CCD's Elastic Net regularization provides superior performance in both ROC AUC (0.792 vs 0.728) and Balanced Accuracy (0.701 vs 0.648), as its combined L1/L2 penalties effectively prevent overfitting while being able to distinguish between different classes. However, under extreme minority-class imbalance (p=0.1), the unregularized Logistic Regression achieves better performance in ROC AUC (0.788 vs 0.701), as well as Balanced Accuracy (0.55 vs 0.5). Conversely, with majority-class imbalance (p=0.9), CCD slightly regains its advantage.

Varying n parameter – Table 5

The results demonstrate a clear pattern across different dataset sizes: CCD exhibits superior performance for small datasets ($n \leq 200$), where its regularization effectively prevents overfitting. As the dataset size increases to medium ranges ($200 < n \leq 2000$, Logistic Regression begins to achieve comparable metric values to CCD, indicating diminishing importance of regularization with more available data. In large datasets, while both models show nearly identical ROC AUC scores, Logistic Regression achieves better balanced accuracy, suggesting it may better utilize high-volume samples without regularization constraints.

	ROC AUC		Δ	Bal. Acc.		Δ
n	CCD	LR	(CCD-LR)	CCD	LR	(CCD-LR)
50	0.8750	0.6250	+0.2500	0.7917	0.5000	+0.2917
100	0.7879	0.6061	+0.1818	0.6768	0.5152	+0.1616
200	0.7601	0.6414	+0.1187	0.7121	0.5859	+0.1263
500	0.7013	0.7114	-0.0100	0.6459	0.6600	-0.0141
1000	0.7285	0.6824	+0.0461	0.6850	0.6150	+0.0700
2000	0.7584	0.7424	+0.0161	0.7019	0.6748	+0.0271
5000	0.7769	0.7770	-0.0001	0.6754	0.7042	-0.0288
10000	0.7696	0.7760	-0.0065	0.6781	0.7079	-0.0298

Table 5: Performance comparison across sample sizes (n) with improvement margins

Varying d parameter – Table 6

	ROC AUC		Δ	Bal. Acc.		Δ
d	CCD	LR	(CCD-LR)	CCD	LR	(CCD-LR)
10	0.832	0.819	+0.013	0.742	0.735	+0.007
20	0.805	0.777	+0.028	0.700	0.710	-0.010
50	0.799	0.814	-0.015	0.700	0.734	-0.034
100	0.800	0.744	+0.056	0.707	0.641	+0.066
200	0.755	0.704	+0.051	0.662	0.655	+0.007
500	0.797	0.651	+0.146	0.708	0.610	+0.098

Table 6: Performance comparison across feature dimensions (d) with improvement margins

For smaller dimensions (d = 10-50), both methods demonstrate comparable performance. However, as dimensionality increases beyond d=100, CCD's regularization provides increasingly significant benefits. The most dramatic difference emerges at d=500, where CCD maintains strong performance (ROC AUC: 0.797, Bal. Acc: 0.708) while LR deteriorates substantially (ROC AUC: 0.651, Bal. Acc: 0.610). This pattern suggests that CCD's Elastic Net regularization becomes crucial in high-dimensional settings

Varying g parameter – Table 7

	ROC AUC		Δ	Bal. Acc.		Δ
g	**CCD (AUC)**	**LR (AUC)**	**Δ (AUC)**	**CCD (Acc)**	**LR (Acc)**	**Δ (Acc)**
0.1	0.7748	0.7655	+0.0093	0.7084	0.7203	-0.0119
0.5	0.7079	0.6636	+0.0443	0.6450	0.6250	+0.0200
0.9	0.7731	0.8397	-0.0666	0.6672	0.7505	-0.0833

Table 7: Performance comparison across feature correlations (g)

CDD performs sligltly better when correlation is moderate (g=0.5). Both methods yield similar results for lower correlation values. LR outperforms CCD for highly correlated features (g=0.9).

Values of coefficients obtained in these two methods Many of coefficients in CCD model got nullified, as L1 regularization does that to combat overfitting by enforcing sparsity, while unregularized logistic regression overfits by assigning undue importance to noise. This can be observed especially in Golub and prostate data, where model of logistic regression without penalty got overfitted.

4 Summary

Our results confirmed that LogRegCCD with optimally selected regularization parameter performs reliably and efficiently, offering better generalization and interpretability due to sparsity.