Università degli Studi di Padova

Dipartimento Matematica
Dipartimento di Matematica "Tullio Levi-Civita"

Report Homework 1

Optimization for Data Science

Students:
Daniele Barolo
Cesare Bidini
Erica Marras

2022/2023
Second Semester

1 Problem description

We are confronted with a semi-supervised learning scenario, in which a binary classification problem is to be addressed. Specifically, a limited number of labeled samples and a larger number of unlabeled points are utilized to develop an accurate model for classifying the correct points labels. To achieve this, we have formulated a loss function that comprises two terms. The first term focuses on the proximity between an unlabeled point and a labeled point, while the second term focuses on the proximity between two unlabeled points. Both terms have been weighted by a similarity measure, which is based on the Euclidean distance between the points. By minimizing this loss function, we aim to accurately classify as many labeled points as possible, given our assumption of proximity (i.e., if the points are indeed clustered). This approach is expected to effectively solve the classification problem.

2 Data set

Initially, synthetic data was generated to investigate the models from a theoretical standpoint. Subsequently, the model has been implemented on real-world datasets of increasing dimensions to ensure the model's efficacy under realistic scenarios.

2.1 Synthetic data generation

The synthetic dataset was created by generating 10000 points from a bimodal distribution, with each mode representing a separate cluster in the dataset. Furthermore, we evaluated the performance of the model on the challenging "two moons" configuration, in order to optimize the choice of similarity metric and associated parameters.

Finally, for both dataset, we have dealt the labeleing problem as follows. Let X be the dataset consisting of $n = 10,000$ points, divided into two distinct clusters, and Y be the set of binary labels corresponding to each point in X. A random subset of 3% of the points was selected with their true labels, while the remaining points were considered unlabeled. Subsequently, the labels for the unlabeled points in Y were randomly initialized with values ranging continuously from 0 to 1.

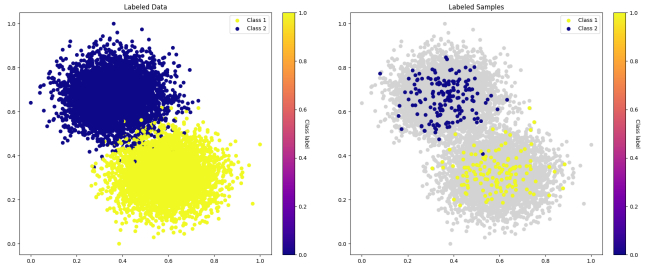

Figure 1: Blob clusters

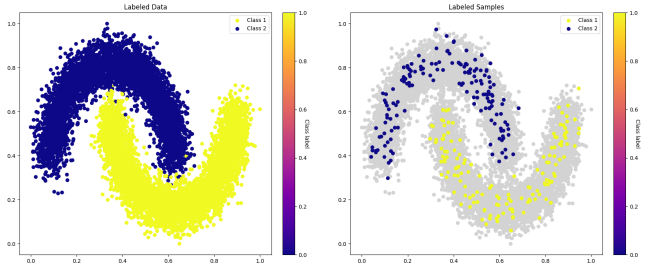

Figure 2: Moon clusters

2.2 Real Dataset: Occupancy of an Office

This dataset, collected by Luis Candanedo (Université de Mons), comprises 6 features (*Time*, *Temperature*, *Humidity*, *HumidityRatio*, *Light*, CO_2) and a binary target indicating the occupancy status of the office. The dataset consists of approximately 8000 data points. We have chosen not to include the *Time* related data cause dates are hard to embed and could have created confusion during the similarity weights computation. Moreover, we decided to drop *Humidity* feature in our analysis, as it is already well expressed by the *HumidityRatio* values.

Given the dimensionality of data we decided to reduce it using PCA (Principal Component Analysis) and get a 3D

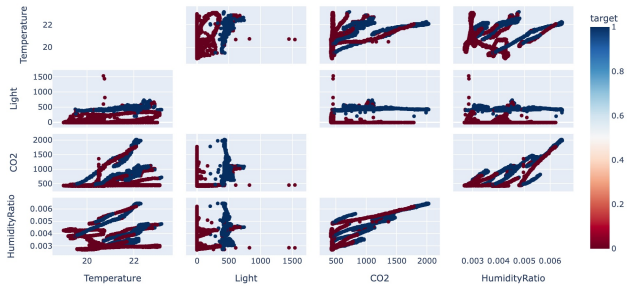

FIGURE 3: Scatter Matrix - Occupancy Dataset

plot.
PCA (Principal Component Analysis) identifies the direction in the feature space where data point exhibite maximum variance as the first principal component of the PCA. The successive principal componets are choosen again accordingly to the direction with maximum variance.

Furthermore, since the algorithm is implemented on a similarity measure based on distances, the features of the dataset points were scaled.

Similar to the approach taken with the synthetic dataset, only a fraction of the labels were retained while the remaining were randomly initialized. Specifically, a random subset consisting of 5% of the datapoints was chosen, which revealed an uneven distribution of the samples across the two clusters. In order to prevent any significant bias in the model, we opted to maintain an equal representation of samples from each cluster by disregarding the majority class labels.

3 Theoretical details

3.1 Loss Function

$$f(y) = \sum_{i=1}^{l}\sum_{j=1}^{u} w_{ij}(y^j - \bar{y}^i)^2 + \frac{1}{2}\sum_{i=1}^{u}\sum_{j=1}^{u} \bar{w}_{ij}(y^i - y^j)^2 \quad (1)$$

Here, w_{ij} and $\overline{w}_{ij}$ represent the weights associated with the pairwise terms in the loss function. The variable $y \in \mathbb{R}^u$ represents the vector of variables we aim to optimize, with y^j denoting the j-th component of the vector.

The loss function $f(y)$ is twice continuously differentiable and also convex.
$f : \mathbb{R}^u \to \mathbb{R}$ has Lipshitz continuous gradient: there exists $L > 0$ such that
$\|\nabla f(y_1) - \nabla f(y_2)\| \leq L\|y_1 - y_2\|, \quad \forall y_1, y_2 \in \mathbb{R}^u.$

3.2 Gradient

The gradient of the Loss function with respect to y_j is:

$$\nabla_{y_j} f(y) = 2\sum_{i=1}^{l} w_{ij}(y^i - \bar{y}^j) + 2\sum_{i=1}^{u} \bar{w}_{ij}(\bar{y}^i - \bar{y}^j) \quad (2)$$

The Hessian matrix of the Loss Function is:

$$H = \nabla^2_{y_i y_j} f(y) = \begin{cases} -2\bar{w}_{ij} & \text{if } i \neq j \\ 2(\sum_{i=k}^{l} w_{ki} + \sum_{i=k}^{u} \bar{w}_{ki}) & \text{if } i = j \end{cases} \quad (3)$$

3.3 Measure of Similarity

The Gaussian function is employed as the preferred metric for evaluating the pairwise weights among the points in the dataset, providing a measure of their similarity.

$$w_{ij} = \exp\left(-\frac{\|\mathbf{x^i} - \hat{\mathbf{x}}^\mathbf{j}\|_2^2}{\text{temperature}}\right) \quad (4)$$

$$\hat{w}_{ij} = exp\left(-\frac{\|\overline{\mathbf{x}}^\mathbf{i} - \overline{\mathbf{x}}^\mathbf{j}\|_2^2}{\text{temperature}}\right) \quad (5)$$

Where $\mathbf{x}$ are the l labeled points and $\hat{\mathbf{x}}$ the u unlabeled ones.

To choice the value of the temperature we have done a grid search based on the efficiency of the model.

3.4 Accuracy

We selected accuracy as the metric to evaluate the implemented methods, which measures the number of correctly classified labels out of the total points to be classified.

4 Gradient Descent

4.1 Gradient Descent Method

A general scheme for the gradient descent algorithm used in our model is given below

1. Choose a starting point $\mathbf{y_1} \in \mathbb{R}^n$, and set $k = 1$.
2. Evaluate the current loss function $f(\mathbf{y_k})$.
3. If the loss is less then a threshold, i.e., $f(\mathbf{y_k}) < \epsilon$, then **STOP** and output $\mathbf{y_k}$ as the approximate solution.
4. Choose a step size $\alpha_k > 0$
5. Update the estimate of the solution as $\mathbf{y_{k+1}} = \mathbf{y_k} - \alpha_k \nabla f(\mathbf{y_k})$.
6. Set $k = k + 1$ and go back to step (a).

To set the step size, we choose two methods:

(a) Armijo rule line search

(b) Fixed step size

4.2 Armijo Rule

The Armijo rule in gradient descent is a line search method that adjusts the step size based on the slope of the objective function to ensure convergence.

4.2.1 Armijo Line Search Algorithm

1. Fix two parameters: $\delta \in (0, 1)$ and $\gamma \in (0, 1/2)$ and a starting step size Δ_k.
2. Starting from this step size, try steps of the form $\alpha = \delta^m \Delta_k$, where $m = 0, 1, 2, \ldots$, until the inequality
$$f(\mathbf{y}_k + \alpha \mathbf{d}_k) \leq f(\mathbf{y}_k) + \gamma\alpha \nabla f(\mathbf{y}_k)^\top \mathbf{d}_k \quad (6)$$
is satisfied. Here, $\mathbf{d}_k = -\nabla f(\mathbf{y}_k)$ is the descent direction.
3. Set $\alpha_k = \alpha$.

4.2.2 Grid Search

In our implementation, we performed a grid search over different Δ_k, δ, and γ values. Specifically, we considered starting step sizes of 0.5, 1, 5, and 10, δ values of 0.3, 0.3, 0.4, and 0.5, and γ values of 0.1, 0.2, 0.3 and 0.4.
We evaluated the performance of each combination of parameters in terms of the speed of convergence and accuracy of the resulting model. Based on our experiments, we found that the best parameters were $\Delta_k = 1$, $\delta = 0.5$, and $\gamma = 0.4$.

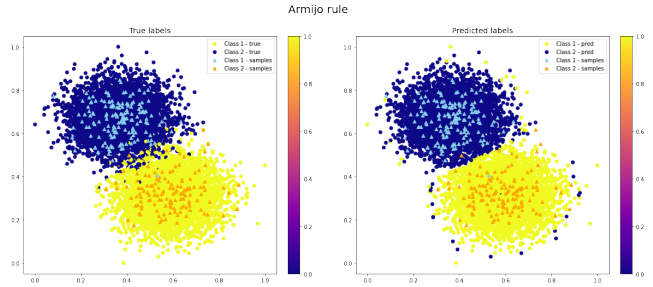

FIGURE 4: Armijo - True labels vs Predicted labels

4.3 Fixed stepsize with 1/L

The loss function $f(y)$ is twice continuously differentiable, so L can be set as the maximum eigenvalue of the Hessian matrix of the loss function. (3)
The gradient method has been applied with fixed stepsize $\alpha_k = \frac{1}{L}$

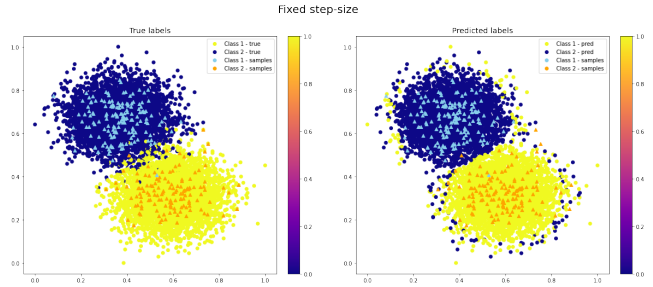

FIGURE 5: Fixed step-size - True labels vs Predicted labels

4.4 Results and comments

The ϵ threshold we used in the stopping condition of both the Armijo and fixed step size implementations of gradient descent has been set to 0.06% of the loss function evaluated in the starting point $\mathbf{y}_1$.

For the regularity of our problem, we expected the fixed step size to be better in terms of efficiency compared to the Armijo rule. However, in order to compare the two step size methods, we have chosen to be consistent in defining the stopping condition. Therefore, in the fixed step size approach, we calculate the loss function even though it may not be strictly necessary for optimization purposes. In our opinion, it is only because of this choice that the improvement of the fixed step size over the Armijo rule is not evident.

However, it is important to acknowledge that the Armijo Rule is particularly advantageous when dealing with

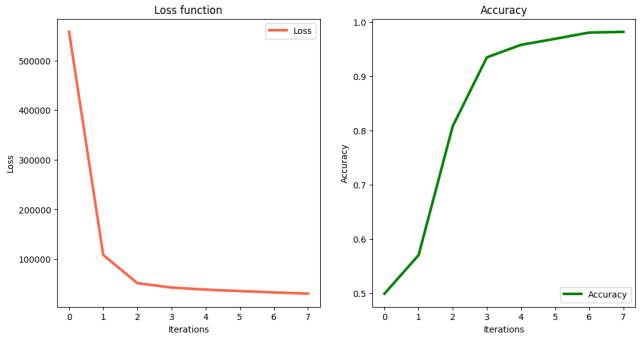

FIGURE 6: Armijo Loss and Accuracy

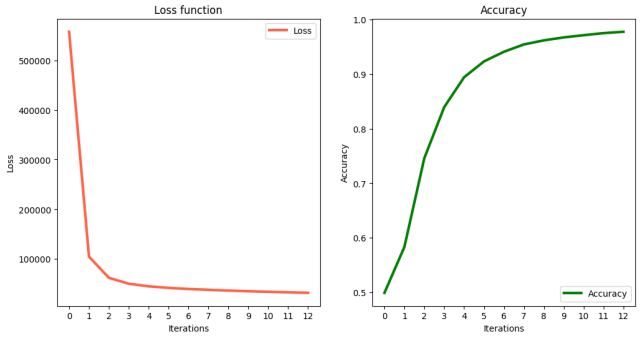

FIGURE 7: Fixed step-size Loss and Accuracy

functions that exhibit varying curvatures. Its ability to dynamically select and adjust the step size based on the slope of the objective function allows it to adapt more efficiently, ultimately leading to faster convergence compared to alternative line search approaches.

	Armijo Rule	Fixed step size
Accuracy	0.986	0.977
Num of Iterations	7	12
Time	86.27 s	62.24 s

TABLE 1: Gradient Descent comparison

5 BCGD

5.1 Random Rule

The BCGD - Random Rule approach randomly chooses the search direction during optimization.

The Algorithm

1. Choose a starting point $\mathbf{y}_1 \in \mathbb{R}^u$.
2. For $k = 1, 2, 3, \ldots$:
   (a) Pick at random $i_k \in 1, \ldots, b$ ($b = u$ because we are considering blocks of one dimension)
   (b) If $|\mathbf{y}_{k+1}[j] - \mathbf{y}_k[j]| < \epsilon$ is satisfied for *tolerance* times, then **STOP**.
   (c) Set $\mathbf{y}_k = \mathbf{y}_{k-1} - \alpha_k \mathbf{U}_{i_k} \nabla_{i_k} f(\mathbf{y}_k)$.

Here, $\mathbf{U}_{i_k}$ is a matrix that selects the i_k-th block of the coordinates of $\mathbf{y}$, and $\nabla_{i_k} f(\mathbf{y}_k)$ is the gradient of the objective function f with respect to the i_k-th coordinates

block (1d) evaluated at $\mathbf{y}_k$. The step size α_k is fixed and set equal to $1/L$.

The threshold we used in the stopping condition has been set to $\epsilon = \frac{1}{2}10^{-4}$. And the *tolerance* is 3% of the dimension of $\mathbf{y}$ (e.g. u).

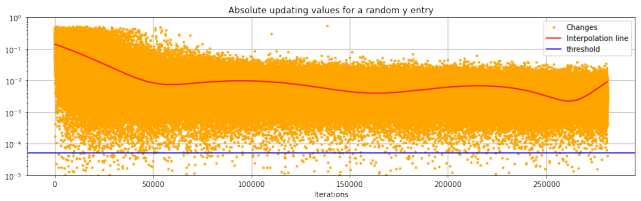

FIGURE 8: CBGD-Random update values

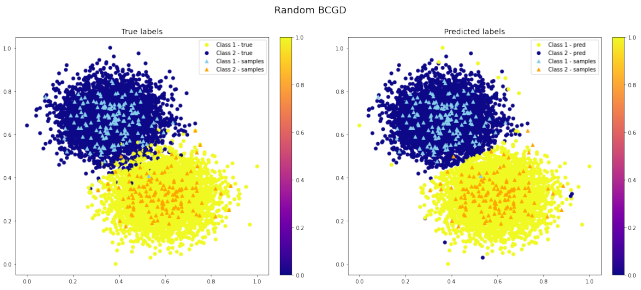

FIGURE 9: CBGD-Random - True labels vs Predicted labels

5.2 Gauss-Southwell

The BCGD - Gauss-Southwell approach systematically selects the search direction during optimization.

The Algorithm

1. Choose a starting point $\mathbf{y}_1 \in \mathbb{R}^u$.
2. For $k = 1, 2, 3, \ldots$:
   (a) If $||\nabla f(\mathbf{y}_k)|| < \epsilon$ (a given threshold), then **STOP**.
   (b) Choose block i_k such that $i_k = \underset{j \in 1,\ldots,b}{\operatorname{argmax}} ||\nabla_j f(\mathbf{y}_k)||$ where $b = u$ (blocks of one dimension)
   (c) Set $\mathbf{y}_{k+1} = \mathbf{y}_k - \frac{1}{L}\mathbf{U}_{i_k} \nabla_{i_k} f(\mathbf{y}_k)$.

Here, $||\cdot||$ denotes the Euclidean norm, and $\nabla_j f(\mathbf{y}_k)$ denotes the jth block of the gradient of f at $\mathbf{y}_k$. The algorithm stops when the norm of the gradient at the current point $\mathbf{y}_k$ is below the given threshold ϵ.

The threshold we used in the stopping condition $||\nabla f(\mathbf{y}_k)|| < \epsilon$ has been set at $10^{-2} * ||\nabla f(\mathbf{y}_1)||$

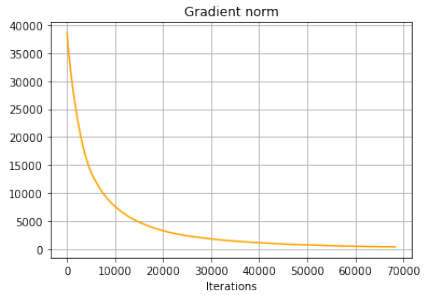

FIGURE 10: Norm of the gradient at each iteration

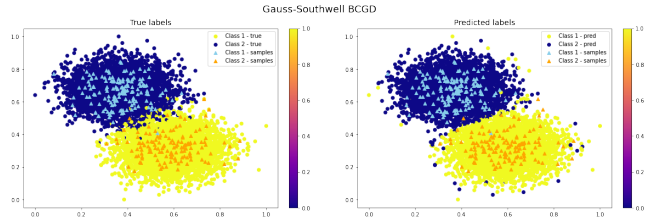

FIGURE 11: CBGD-GS - True labels vs Predicted labels

5.3 Results and comments

	Random	Gauss-Southwell
Accuracy	0.989	0.987
Num of Iterations	280000	68300
Time	102.77 s	30.44 s

TABLE 2: BCGD methods comparison

In terms of efficiency, the two BCGD methods share remarkable similarities. However, the Gauss-Southwell method excels in efficiency, thanks to the smoothness of the problem it addresses. In general, the Random method is favored for exploring complex problems because it enables diverse exploration and the possibility of uncovering improved solutions. However, the random nature of this approach can also introduce unpredictability and potentially lead to slower optimization. On the other hand, when dealing with smoother problems, the Gauss-Southwell method offers clear advantages.

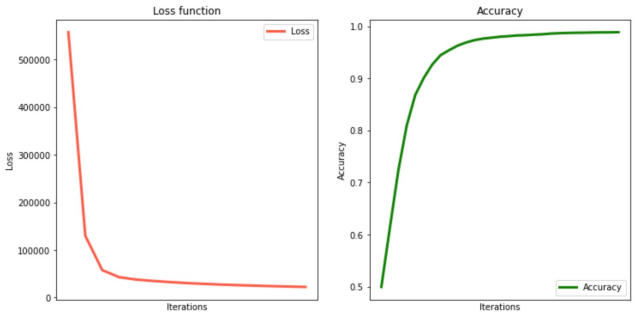

FIGURE 12: CBGD-Random Loss and Accuracy - Synthetic data

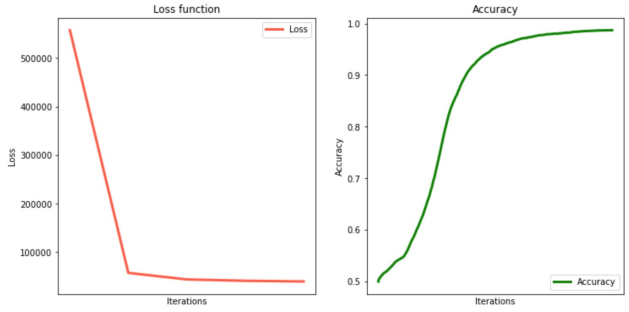

FIGURE 13: CBGD-GS Loss and Accuracy - Synthetic data

6 Methods in Real Dataset

6.1 Gradient Descent - Armijo Rule and Fixed stepsize

The ϵ threshold we used in the stopping condition of both the Armijo and fixed step size implementations of gradient descent has been set to 0.006% of the loss function evaluated in the starting point $\mathbf{y}_1$.
We have determined the value for ϵ by considering the minimization of the loss. Through experimentation, we have observed that at this particular point, a satisfying level of accuracy is achieved.

	Armijo Rule	**Fixed step size**
Accuracy	0.982	0.958
Num of Iterations	37	148
Time	112.75	98.8

TABLE 3: Gradient Descent comparison

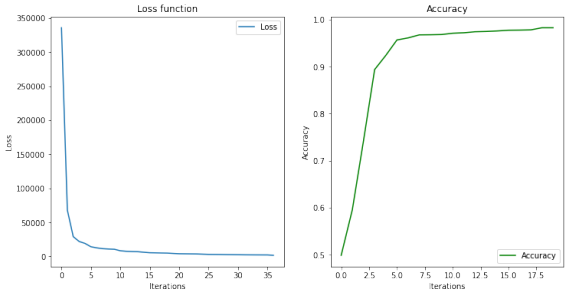

FIGURE 14: Armijo Loss and Accuracy

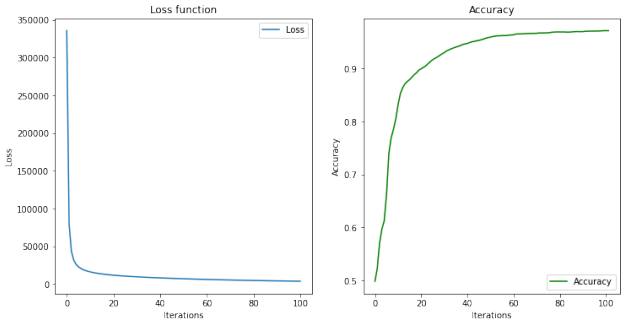

FIGURE 15: Fixed Loss adn Accuracy

6.2 BCGD - Random Rule and Gauss-Southwell

The threshold we used in the stopping condition for the Random Rule BCGD has been set to $\epsilon = \frac{1}{2}10^{-5}$. And the *tolerance* is 3% of the dimension of $\mathbf{y}$.

The threshold we used in the stopping condition for the Gauss-Southwell $||\nabla f(\mathbf{y}_k)|| < \epsilon$ has been set at $10^{-3} * ||\nabla f(\mathbf{y}_1)||$

	Random	**Gauss-Southwell**
Accuracy	0.972	0.982
Num of Iterations	775537	572838
Time	93.477	60.35

TABLE 4: CBGD comparison

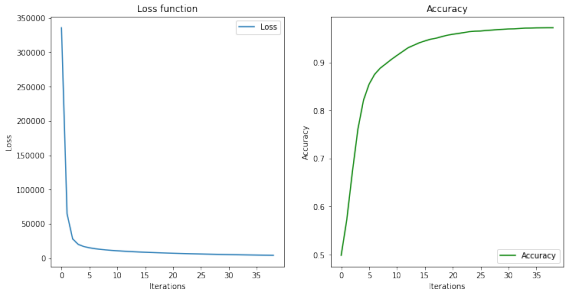

FIGURE 16: Random Loss and Accuracy

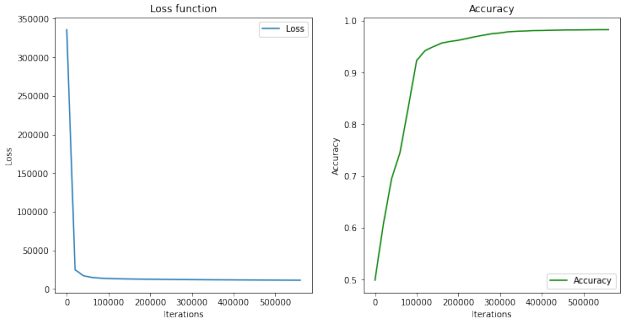

FIGURE 17: Gauss-Southwell Loss and Accuracy

7 Conclusions and Plots

By comparing the methods in terms of accuracy and computational efficiency, for both the synthetic and real datasets, we can conclude that the BCGD Gauss-Southwell method outperforms the others. This conclusion is based on the regularity of the loss function that describes the minimization problem. The smoothness and well-behaved nature of the loss function contribute to the superior performance of the BCGD Gauss-Southwell method across both datasets.

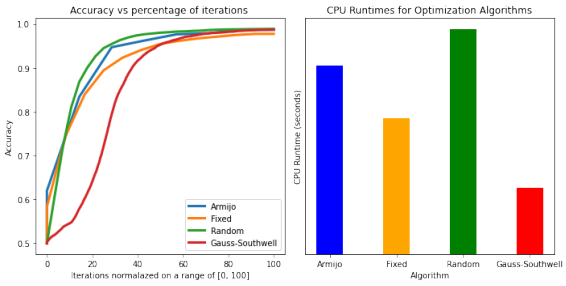

FIGURE 18: Synthetic Data Summary

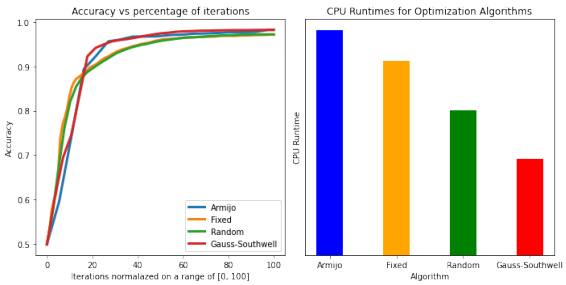

FIGURE 19: Occupancy Dataset Summary