Multi-Objective Semi-Supervised Learning

Selena Chen
Department of Computer Science
North Carolina State University
Raleigh, NC 27606
schen53@ncsu.edu
Arun Kumar Ramesh
Department of Computer Science
North Carolina State University
Raleigh, NC 27606
arames25@ncsu.edu
Andrew Sauerbrei
Department of Computer Science
North Carolina State University
Raleigh, NC 27606
amsauerb@ncsu.edu

***Abstract*—The objective of this paper is to implement a multi-objective semi-supervised explanation system. The creation of such a model comes with its own constraints and challenges. For instance, our access to the ground-truth Y values is restricted, meaning there is only a limited number of times we could access the Y values. Similarly, there are multiple Y values that are required to be optimized. Our proposed model optimizes multiple Y values while having restrictive access to them.**

I. Introduction

In the real world, accessing Y values is difficult. This is because labeled data is scarce or expensive to obtain. An effective semi-supervised learning algorithm leverages the minimum access to Y values. Multi-objective optimization refers to the process of simultaneously optimizing multiple objectives or goals. For example, consider the scenario of buying a car. There are multiple objectives that we could consider and optimize. We would choose to maximize mileage while considering minimizing the price. This report discusses approaches to implementing an effective multi-objective semi-supervised algorithm taking inspiration from the concepts discussed in class. The goal is to move as close to the Pareto Frontier as possible.

While we understand what multi-objective semi-supervised learning is, it is challenging to implement such an algorithm due to the following reasons. Firstly, how can we effectively balance multiple objectives in semi-supervised learning? To add to the first question, how can we handle both discrete and continuous objectives in multi-objective problems? Often, objectives may be conflicting or have different levels of importance. How do we prioritize one goal over the other in such a situation? Perhaps the question might draw parallels to the philosophical Trolley problem [1]. There can be many ”right” solutions but not a ”correct” solution. Questions like how to handle missing labels and imbalanced datasets are also worth pondering upon. In the subsequent paragraphs, we try to answer some of these questions.

The multi-objective semi-supervised algorithm discussed in class, in a nutshell, summarizes the columns, recursively clusters the data, and chooses the better half (done by ’SWAY’). The implementation of ’SWAY’ uses a recursive Fast-Map to split the data into two halves and uses Zitler’s continuous domination predicate [2] to choose the better half.

A potential algorithm improvement could be carried out by either a better implementation of ’SWAY’ or better hyper-parameter optimization. We modified the functionality of SWAY by implementing the ’half’ function inspired by K-Means clustering [3]. We also changed the Zitler domination in ’Better’ to a modified version of Boolean Domination. Finally, we also perform hyperparameter tuning. It is our view that, by tuning the hyperparameters, we can better understand how each hyperparameter impacts the performance of the model. This knowledge can then be used to make more informed decisions when changing the model architecture subsequently.

II. Related Work

As mentioned in the introduction, Y values are difficult to obtain. Olliver Chapelle et al. discuss in [4] that the main challenges in semi-supervised learning are that the amount of labeled data is often limited. The authors have also discussed various methods to perform effective semi-supervised learning using both labeled and unlabeled data.

A few popular techniques include graph-based methods, co-training, and self-training. Graph-based methods involve constructing a graph based on the data, where nodes represent data points and edges represent relationships between data points. Graph-based methods use the structure of the graph to transfer information between labeled and unlabeled data points. Self-training is one of the earliest approaches to utilizing unlabeled data. In this approach, the semi-supervised model learns from the labeled data, and then the most confident data from the unlabeled data are then added to the labeled data. In co-training, two or more models are made to learn on different subsets of the data, and the models learn from each other by exchanging information about the unlabeled data points.

We have already emphasized the importance of hyper-parameter tuning earlier. Hyperparameter tuning is the process of finding the best set of values for the model’s hyper-parameters. Hyperparameters are the parameters that are set before the learning process begins. Hyper-parameters are not learned from the data, as opposed to the model parameters, which are derived from training data. The ’number of neighbors (K)’ in KNN [5] is a hyper-parameter, while the weights in an Artificial Neural Network are an example of a model parameter.

Often, learning algorithms get stuck at local minima. Local minima refer to a point where the cost function has a lower value than at all nearby points, but is not necessarily the global minimum. Approaches such as Simulated Annealing [6] and Genetic Algorithms [7] can be used to navigate across the local minima. Both these approaches have been discussed in detail in the course.

Learning algorithms traversing through higher dimensions can encounter the curse of dimensionality. They can also encounter noise and redundancy affecting their learning. Dimensionality reduction techniques can be used to reduce the number of features in the data while retaining the most important information. Projection is a commonly used dimensionality reduction technique in semi-supervised learning. In projection, the high-dimensional data is mapped onto a lower-dimensional space while preserving the pairwise distances between the data points. Principal Components Analysis (PCA) [8] is one such traditional approach for projecting multidimensional data onto lower-dimensional subspaces with minimal variance loss.

Many research works have been focused on semi-supervised learning and Multi-objective optimization problems. D.-H. Lee [9] formulated a simple approach to utilize unlabeled data by using pseudo-labels. In this method, he trained a neural network on labeled data (in a supervised manner) and unlabeled data simultaneously. The model is made to predict the unlabeled data samples, and the prediction with the highest confidence is given as a ’pseudo-label’ to the originally unlabeled data. Grandvalet et al. [10] use a regularization technique called Entropy Minimization (EntMin) to perform semi-supervised learning among labeled and unlabeled data. The regularization encourages the model to make predictions with lower entropy (entropy is a measure of uncertainty, minimizing entropy leads to more confident predictions). This helps to prevent the decision boundary from passing through a high-density region and ensures there is a clear separation between multiple classes. Deb and Saxena [11] tackled Multi-Objective Optimization by incorporating PCA with Non-Dominated Sorting Genetic Algorithm II (NSGA-II) [12], eliminating redundant, conflicting objectives and reducing the problem's dimensionality to progress toward the Pareto-optimal region. The approach assumes that if two objectives are negatively correlated, they conflict with each other.

We have discussed how researchers have tackled both multi-objective and semi-supervised problems. In the following paragraphs, we will explain in detail about our approach which was briefly mentioned in the earlier paragraphs.

III. Methods

A. Sway

Sway [13] is the default clustering method that we used throughout the instructor-driven portion of the implementation. The method would take in the entire data as input and would use a recursive sampling algorithm to determine which dimension of the data gives the best model. This algorithm is a Recursive Fastmap, which divides the data into two halves by using two distance points in the data and then projecting the rest of the points to the line. These projections are then clustered and split into two halves. We then use a Zitzler Continuous Domination Predicate to determine which half is better than the other. This half is then recursed on until the best cluster is of a certain size, at which point we return best and rest to the be further examined.

B. DBScan

DBScan [14] is a clustering algorithm that we felt might be able to replace the original Sway algorithm. This algorithm works off the idea that clusters are more dense parts of the data set, and we can find them through the fact that they are more dense. This idea allows DBScan to be robust against outliers and to also not be told how many clusters to look for beforehand like other algorithms do. All it needs is an epsilon value, which defines the radius of a circle around a data point to look for neighboring points, and a minimum number of points that must be within that radius to be considered a cluster. The algorithm uses Euclidean distance metrics to determine how close points are to each other, although our attempt used the dist function that we already had defined. This method also has the upside of only needing to scan through the data once. It is particularly useful when the data is not well-separated, and when the number of clusters is unknown or difficult to determine beforehand. Our implementation would go through each a list of the data points and check how much density surrounded it. If the point is suitably dense, it is flagged as a seed point and then grown into a cluster. Within this growth cycle, we check all the surrounding data points to see if they are dense enough to be a branch point or if they are just a leaf point. If they are a branch point, we check if its surrounding points are branch or leaf points in the same manner. The goal is to get all leaf points and then the cluster is completed. This algorithm goes until each data point is either labeled as belonging to a cluster or is labeled noise.

C. Agglomerative Hierarchal Clustering

Agglomerative Hierarchal Clustering is another clustering algorithm that we tired to use in place of Sway. This algorithm works differently from DBScan: Instead of trying to find clusters by locating the densest data points, it instead repeatedly steps through the data points, combining them one at a time, to create the clusters. AHC takes in only one parameter, k, which defines the number of clusters the algorithm is supposed to condense the graph to. At the start, each data point is treated as its own cluster. The algorithm then loops thorough all the clusters and finds the two that have the lowest distance between the two of them. There are a few different ways to measure this distance. When finding the distance between the clusters, we have to look at the distance between each data point in the cluster, which can be a lot. To this end, we then either take the minimum distance, maximum distance, or an average of all the determined distances; any of these methods can work to determine the two closest clusters. Once we have found them, we then condense the list of clusters to be all the preexisting clusters, plus those two clusters combined into one. This process repeats until there are the right number of clusters according to the input parameter k.

D. Zitzler

The Zitzler Domination Predicate is a commonly used method in multi-objective optimization that compares two solutions based on their objectives. It was implemented in the instructor's version of the 'Better' function. Zitzler's function considers the values of each individual's objectives, as well as the weights assigned to each objective (to indicate whether it should be maximized or minimized). The domination status of one solution over another is determined by comparing the losses incurred when jumping to and from each individual. A solution is preferred over the other if the loss incurred when jumping to it is less than the loss incurred when jumping to the other solution. Also, since the jumps involve exponential calculation, the values are normalized before feeding them to Zitzler's function.

E. Kmeans

We have implemented a K-means-style clustering approach to split data into two clusters. Similar to the original 'SWAY', the code splits the data into left and right halves (two clusters). The algorithm initializes two random rows as centroids, assigns the other rows to their closest centroid, and updates the new centroid based on the cluster. These steps are performed iteratively until there is no change in centroid or if the code reaches a certain number of iterations. The farthest row is calculated as the row furthest to the randomly chosen 'above' row and within the same cluster.

F. Boolean Domination

Boolean domination is a form of dominance observed in multi-objective optimization. According to this type of dominance, solution A is considered to dominate another solution B if A is superior in at least one objective and not inferior to B in any other objective. We have replaced a Zitzler Domination with a version of Boolean Domination. Boolean Domination is not expected to perform well in higher dimensions. This is because there are many objectives to consider, and it is difficult to find a single solution that can excel in all of them simultaneously. However, we wanted to explore Boolean Domination since ultimately the performance of both dominance relations can depend on the specific problem and the characteristics of the Pareto front. The code implementation calculates the sum of the square of the distance between the data points for each objective and then uses this to calculate a weighted sum for each row, comparing them to determine which one dominates the other.

G. Bootstrap

Bootstrap is one of the statistical methods we used on our data to determine if our changes to the base implementation actually made a statistical difference. The method takes the two output sets from sway and kmeans and then summarizes each into a NUM structure.

We also combine the two into a third NUM structure for future comparing. After this, we create a list of values taken from each NUM structure that we adjust to have the same mean in each list. Finally, take a sample of each list and determine the delta value for that sample. If this value exceeds the delta value of the two whole lists, then we increment a variable. If that variable is greater than our confidence metric, we know that the two algorithms returned data that is statistically similar.

H. Cliffs Delta

Cliff's Delta is another statistical method we used on our data. This metric is a non-parametric effect size measurement that determines the amount of difference between our two groups beyond p-value interpretations. For our implementation, we used the global Cliff threshold of .147. We input the two lists of data from our sway and kmeans implementation. This maxes out at 256, so we use a helper method to truncate the list to a 256 length random sample. We then check if the length of either set is greater than 10 times the length of the other. If so, we again get a random sample from that list in order to make sure they are both the same length. The algorithm loops through each list, determining which value is greater than the other at each index, and incrementing a counter accordingly. If our the absolute value of the difference of these two counters divided by the number of value sets we compared is less greater than our threshold of .147, then that means that the two sets are statistically similar.

I. Datasets

We have a variety of datasets that we used to test our base implementation and our adjusted implementation. Each objective has a '+' or a '-' sign at the end. The '+' sign implies that the objective has to be maximized whereas the '-' sign implies that the objective has to be minimized. We have two datasets that are working with car data. The first is trying to maximize the gas mileage for both Highway and City driving while minimizing the weight of the car and the class of the car. The second set is trying to maximize the car's acceleration and miles per gallon while minimizing weight. There are two datasets that use Constructive Cost Modeling to examine the software being used. The data tries to maximize the lines of code while minimizing the amount of effort, risk, application experience, and platform experience needed. There are two sets of data that examine the time it takes to close issues. We do this by maximizing ACC, minimizing MRE, and maximizing PRED40+. The pom dataset has to do with examining agile development and its challenges. We see how projects are doing by maximizing completion while minimizing time spent idle and the cost of the development. The last two datasets are the longest, and they have to do with computing complex physics. SSM is about trimeshes and the configuration space for them, and the goal is to minimize both the number of iterations and the time spent to get to a solution. SSM has to do with the amount of space needed to configure a video encoder, and it is trying to minimize the amount of energy needed to encode, as well as minimize the PSNR of the encoder, which is the ratio of the maximum possible power of the signal and the peak signal of the noise.

IV. Results

TABLE I
RESULTS FOR `auto2.csv`

	CityMPG+	Class-	HighwayMPG+	Weight-
all	21.0	17.7	28.0	3040.0
sway1	30.7	8.81	35.6	2134.75
xpln1	30.05	9.29	34.3	2197.25
sway2	22.85	15.96	28.9	2891.75
xpln2	22.35	16.34	28.95	2900.5
top	33.0	8.82	39.2	2050.5
	CityMPG+	**Class-**	**HighwayMPG+**	**Weight-**
all to all	=	=	=	=
all to sway1	≠	≠	=	=
all to sway2	≠	≠	≠	≠
sway1 to sway2	≠	≠	≠	≠
sway1 to xpln1	≠	≠	≠	≠
sway2 to xpln2	≠	≠	≠	≠
sway1 to top	≠	≠	≠	≠

TABLE II
RESULTS FOR `auto93.csv`

	Acc+	Lbs-	Mpg+
all	15.5	2800.0	20.0
sway1	16.34	2152.75	31.0
xpln1	16.07	2277.35	29.5
sway2	14.57	3109.95	21.5
xpln2	15.5	3117.5	23.0
top	18.78	1985.0	40.0
	Acc+	**Lbs-**	**Mpg+**
all to all	=	=	=
all to sway1	≠	≠	≠
all to sway2	≠	≠	≠
sway1 to sway2	≠	≠	≠
sway1 to xpln1	≠	≠	≠
sway2 to xpln2	≠	≠	≠
sway1 to top	≠	≠	≠

TABLE III
RESULTS FOR `china.csv`

	N_effort-
all	2098.0
sway1	877.2
xpln1	1460.95
sway2	11463.3
xpln2	5832.7
top	146.0
	N_effort-
all to all	=
all to sway1	≠
all to sway2	≠
sway1 to sway2	≠
sway1 to xpln1	≠
sway2 to xpln2	≠
sway1 to top	≠

TABLE IV
RESULTS FOR `coc1000.csv`

	AEXP-	EFFORT-	LOC+	PLEX-	RISK-
all	3.0	19548.0	1013.0	3.0	5.0
sway1	2.8	18347.65	1069.0	3.1	4.75
xpln1	2.85	21012.7	1088.25	2.85	5.05
sway2	3.0	21102.75	948.35	3.05	5.95
xpln2	3.0	19289.25	1026.65	3.05	5.75
top	2.0	29972.0	1529.0	1.0	3.0
	AEXP-	**EFFORT-**	**LOC+**	**PLEX-**	**RISK-**
all to all	=	=	=	=	=
all to sway1	≠	≠	≠	≠	≠
all to sway2	≠	≠	≠	≠	≠
sway1 to sway2	≠	≠	≠	≠	≠
sway1 to xpln1	≠	≠	≠	≠	≠
sway2 to xpln2	≠	≠	≠	≠	≠
sway1 to top	≠	=	=	≠	≠

TABLE V
RESULTS FOR `coc10000.csv`

	EFFORT-	LOC+	RISK-
all	21427.0	931.0	5.0
sway1	18167.55	1030.95	3.85
xpln1	24372.25	1213.85	5.0
sway2	19410.3	1037.2	5.25
xpln2	22576.85	1017.7	5.0
top	17752.0	1959.0	0.0
	EFFORT-	**LOC+**	**RISK-**
all to all	=	=	=
all to sway1	=	=	≠
all to sway2	≠	≠	≠
sway1 to sway2	≠	≠	≠
sway1 to xpln1	≠	≠	≠
sway2 to xpln2	≠	≠	≠
sway1 to top	=	=	≠

TABLE VI
RESULTS FOR healthCloseIsses12mths0001-hard.csv

	ACC+	MRE-	PRED40+
all	7.16	75.34	25.0
sway1	7.68	73.91	25.0
xpln1	7.55	73.79	25.0
sway2	7.1	75.13	25.0
xpln2	7.1	75.13	25.0
top	11.33	64.91	25.0
	ACC+	**MRE-**	**PRED40+**
all to all	=	=	=
all to sway1	=	=	≠
all to sway2	≠	≠	=
sway1 to sway2	=	=	≠
sway1 to xpln1	≠	=	=
sway2 to xpln2	=	=	=
sway1 to top	=	=	=

TABLE VII
RESULTS FOR healthCloseIsses12mths0011-easy.csv

	ACC+	MRE-	PRED40+
all	-12.24	119.28	0.0
sway1	-0.8	41.18	50.0
xpln1	-0.68	16.05	70.83
sway2	-12.15	119.02	0.0
xpln2	-12.13	119.06	0.0
top	0.0	0.0	83.33
	ACC+	**MRE-**	**PRED40+**
all to all	=	=	=
all to sway1	=	=	=
all to sway2	≠	≠	=
sway1 to sway2	=	=	=
sway1 to xpln1	=	≠	≠
sway2 to xpln2	≠	≠	=
sway1 to top	=	=	=

TABLE VIII
RESULTS FOR pom.csv

	Completion+	Cost-	Idle-
all	0.9	313.79	0.25
sway1	0.89	210.86	0.23
xpln1	0.9	324.03	0.24
sway2	0.91	380.53	0.25
xpln2	0.91	368.32	0.23
top	1.0	138.75	0.0
	Completion+	**Cost-**	**Idle-**
all to all	≠	=	≠
all to sway1	=	≠	≠
all to sway2	=	=	=
sway1 to sway2	≠	≠	≠
sway1 to xpln1	≠	≠	≠
sway2 to xpln2	≠	=	≠
sway1 to top	=	=	=

TABLE IX
RESULTS FOR SSM.csv

	NUMBERITERATIONS-	TIMETOSOLUTION-
all	6.0	133.54
sway1	5.7	130.45
xpln1	5.7	124.11
sway2	8.3	142.11
xpln2	9.0	164.32
top	4.0	59.99
	NUMBERITERATIONS-	**TIMETOSOLUTION-**
all to all	=	=
all to sway1	≠	≠
all to sway2	≠	≠
sway1 to sway2	≠	=
sway1 to xpln1	≠	≠
sway2 to xpln2	≠	≠
sway1 to top	=	=

TABLE X
RESULTS FOR SSN.csv

	Energy-	PSNR-
all	1237.73	45.67
sway1	955.28	45.61
xpln1	959.31	44.92
sway2	1818.81	43.38
xpln2	1709.71	44.16
top	466.17	26.71
	Energy-	**PSNR-**
all to all	=	=
all to sway1	≠	=
all to sway2	≠	≠
sway1 to sway2	≠	=
sway1 to xpln1	=	≠
sway2 to xpln2	≠	≠
sway1 to top	=	≠

Upon making the changes to the SWAY, we ran both the instructor's implementation of the SWAY and our implementation and generated the results. The Tables I - X juxtaposes the performance of the instructor's SWAY, which is referred to as sway1, and our implementation of SWAY, which is called sway2. Additionally, the sway2 uses the earlier mentioned K-means approach. The results obtained using DBScan and Agglomerative Hierarchical Clustering was not satisfactory and hence are not mentioned (and discussed) in our report. We also want to acknowledge that we did not incorporate Boolean Domination in the results for sway2, which was an oversight on our part. Unfortunately, due to time constraints, we were unable to rerun the experiment and revise the report to include Boolean Domination. The code implementation of these techniques is present in our GitHub repository (The Boolean Domination is implemented in the 'better2' function of 'query.py', while the DBScan and Agglomerative Hierarchical Clustering is implemented and commented in the 'optimization.py').

Tables I - X contain the aggregated results obtained from all the datasets. The columns for each set of data are the attributes of the dataset that we were either trying to maximize or minimize. The rows all detail what values were determined for each of these columns using the indicated algorithm. As previously mentioned, this means that the sway1 and xpln1 rows are what the base implementation would find for that dataset, whereas sway2 and xpln2 are what our KMeans implementation finds. The "all" row is just raw values for all the rows in the data, so this remains constant for each iteration of the dataset. The "top" row is the values found in the top subset of rows and shows what would be found if you looked at all the values, as opposed to the sway algorithms, which only look at the better parts of the dataset.

By comparing the values of sway1 and sway2 of the first table across Tables I - IX, we notice that the instructor's sway1 outperformed our sway2 for most of the objectives. Although, our sway2 performed better than sway1 in specific objectives or columns. For instance, our sway1 performed better for the 'Completion' objective in 'pom.csv', 'PSNR' objective in 'SSN.csv', and 'AEXP' objective in 'coc1000.csv'. Our sway2 also obtained the same result or obtained almost as good results as the instructor's sway1 in a few objectives like 'PRED40' in 'healthCloseIsses12mths0001-hard.csv'.

The bottom table of each report shows a conjunction of the effect size test (cliff's delta) and significance test (bootstrap) that we implemented during the semester. This section compares the values from the "all" row to the results from the other rows in order to see if there are any statistically significant differences between all of the data and the subsets that are taken in the other methods. If an equals sign is seen, that means that there wasn't enough difference seen by our tests. If there is a not equals sign, then we did find something that is statistically significant. This could either mean that our new sway2 algorithm is doing much better or much worse in comparison to the original sway. For example, in Table X, which has to do with the SSM.csv data, we see that the new sway is performing noticeably worse in terms of minimizing the Energy used, and this is reflected in the second part of the report when we compare sway1 to sway2. We also see in that same row that both sway algorithms found PSNR values that were about the same, and this is reflected with an equal sign for that part of the sway comparison row. We see a not equals sign in that column for that comparison. The opposite is seen in Table VIII, which relates to the pom.csv file. We see that our sway2 algorithm found a higher completion percentage value than the original sway, which was an improvement for that part of the data. We see this reflected in the bottom section where there is a not equals sign in that column for that row.

V. Conclusion

We implemented DBSCAN and KMeans-inspired algorithm to cluster the data and also used an implementation of Boolean Domination to choose the better cluster. The results provided in this report are obtained from the KMeans-inspired implementation of the SWAY function ('half' function, to be precise). The results from the instructor's SWAY algorithm are juxtaposed with those from our SWAY algorithm. Although our approach didn't yield better results, experimenting with different methods and modifying different regions of the code helped us better understand the multi-objective semi-supervised problem. We were also able to devise the limitations and come up with approaches that could potentially improve the results. They are discussed in the following paragraphs.

VI. Discussion and Future work

Although we attempted to develop an improved implementation of the SWAY algorithm, our results didn't perform better than our instructor's implementation in most scenarios tested. While our approach was inspired by popular existing techniques, our results suggest that our algorithm might not be optimal for the datasets used in this study. Nevertheless, we acknowledge several limitations in our implementation that may have contributed to these results and propose a few methods or approaches that we intend to perform in future work that could potentially improve the results.

One of the areas of improvement in our K-Means style clustering implementation is choosing better initial clusters. In our current implementation, we are choosing random rows (or data points) as initial clusters. We could consider using other approaches that could help us identify better initial clusters. For instance, we could use an approach similar to the KMeans++ [15] algorithm, where we could choose rows (or data points) that are distant from one another. This helps to ensure that the clusters are well-separated and improves the chances of finding the optimal solution.

In the future, we also intend to perform hyperparameter tuning since we believe that this aspect of our approach hasn't been used to its full potential. We anticipate that tuning and further optimizing the hyperparameters will significantly improve our results. We have a similar intuition about both DBScan and Agglomerative Hierarchical Clustering and their potential, and we would like to focus our future works on DBScan as well. We also will incorporate boolean domination into the SWAY algorithm to potentially improve its decision-making capabilities.

References

[1] B. Duignan, "Trolley problem," Encyclopædia Britannica, 10-Sep-2021. [Online]. Available: https://www.britannica.com/topic/trolley-problem. [Accessed: 20-Apr-2023].

[2] Zitzler, Eckart. Evolutionary algorithms for multiobjective optimization: Methods and applications. Vol. 63. Ithaca: Shaker, 1999.

[3] Lloyd, Stuart. "Least squares quantization in PCM." IEEE transactions on information theory 28, no. 2 (1982): 129-137.

[4] O. Chapelle, B. Schölkopf, and A. Zien, Semi-supervised learning. Cambridge, MA: MIT Press, 2006.

[5] Cover, Thomas, and Peter Hart. "Nearest neighbor pattern classification." IEEE transactions on information theory 13, no. 1 (1967): 21-27.

[6] Kirkpatrick, Scott, C. Daniel Gelatt Jr, and Mario P. Vecchi. "Optimization by simulated annealing." science 220, no. 4598 (1983): 671-680.

[7] Jh, Holland. "Adaptation in natural and artificial systems." Ann Arbor (1975).

[8] Jolliffe, Ian T. Principal component analysis for special types of data. Springer New York, 2002.

[9] Lee, Dong-Hyun. "Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks." In Workshop on challenges in representation learning, ICML, vol. 3, no. 2, p. 896. 2013.

[10] Grandvalet, Yves, and Yoshua Bengio. "Semi-supervised learning by entropy minimization." Advances in neural information processing systems 17 (2004).

[11] Deb, Kalyanmoy, and Dhish Saxena. "Searching for Pareto-optimal solutions through dimensionality reduction for certain large-dimensional multi-objective optimization problems." In Proceedings of the world congress on computational intelligence (WCCI-2006), pp. 3352-3360. 2006.

[12] Deb, Kalyanmoy, Amrit Pratap, Sameer Agarwal, and T. A. M. T. Meyarivan. "A fast and elitist multiobjective genetic algorithm: NSGA-II." IEEE transactions on evolutionary computation 6, no. 2 (2002): 182-197.

[13] https://ieeexplore.ieee.org/abstract/document/8249828

[14] He, Zengyou, Xiaofei Xu, and Shengchun Deng. "Discovering cluster-based local outliers." Pattern recognition letters 24, no. 9-10 (2003): 1641-1650.

[15] David, Arthur. "k-means++: The advantages of careful seeding." In Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms, 2007, pp. 1027-1035. 2007.