Ideal and Nonideal Kinetic KNN Clustering Optimization

Zackary Hoyt

4 December 2022

Abstract

In stochastic clustering problem spaces where sampled observations of multiple classifications are linearly separable—allowing for some degree of error due to stochastic influence—the k-means clustering algorithm is traditionally used to approximate those classification definitions. The k-means clustering algorithm approximates those definitions over a number of computing the average definition of a cluster and then updating the classification definitions accordingly. Repeating until the definitions do not change from the previous generation. This has similarities to genetic algorithms training processes, by which multiple generations of solutions are derived by mutating the previous one to better approximate an optimal one. More advanced implementations detail these mutations as a function of some learned momentum, which it is observed that individual genes mutating in a consistent direction yields ever improving models. This paper takes that concept of momentum-based mutations and proposes the kinetic optimizers to shorten the clustering training times, in comparison to the k-means clustering. Multiple kinetic optimizer variants are explored and demonstrated to not only be capable shorten training times, but also consistently improve the overall testing results.

Table of Contents

Abstract 1
1 Introduction 2
2 Kinetic Optimizers 2
2.1 Ideal Kinetic Optimizer (IKO) 2
2.2 Nonideal Kinetic Optimizer (NKO) 3
3 KNN Experiments 4
4 Results 7
5 Conclusions 12
6 References 12
7 Proofs 13
7.1 Ideal Kinetic Update 13
7.2 Nonideal Kinetic Update 13

1 Introduction

Genetic algorithms represent a specialized domain in machine learning that comparatively excels against other domains where there is no deterministic algorithm or optimization process—or it is infeasible to develop one—to solve a given stochastic objective function. Simple implementations of genetic algorithms explore these problem spaces and stochastically approximate an optimal solution by unguided but statistically probabilistic—with a high enough number of generations—training process. However, stochastic objective functions typically detail an optimal solution that is part of a continuum from which it can be reached by monotonically adjusting a sub-optimal solution in the direction of the optimal solution. That is, there typically exists a local space such that a slope can be followed from a sub-optimal solution to the optimal solution. More advanced genetic algorithm implementations utilize a type of momentum-based mutation to account for this, yielding faster training times and/or better performing solutions (Temby and et al., 2005).

The kinetic optimizers proposed here take that concept and apply it to the k-nearest neighbors (KNN), clustering problem, which exists within the machine learning subfield of unsupervised learning, to similarly attempt to increase the speed at which the clustering converges and/or decrease the classification error with respect to the k-means clustering algorithm. It is demonstrated that for a range problem spaces the proposed kinetic optimizers can decrease training time and/or resulting classification error of the points by this momentum-based learning technique.

2 Kinetic Optimizers

The kinetic optimizers proposed here are developed to linearly update individual weights relative to two models and their respective evaluated losses given some objective function. These are the ideal kinetic optimizer (IKO) and nonideal kinetic optimizer (NKO) algorithms. Terminology here has been adopted from the mechanical physics terminologies of frameworks that ignore friction (ideal) and include friction (nonideal) due to the added difficulty they provide. In machine learning applications, the ideal framework then provides for unbounded models, which are then not constrained to exist within any intervals. Similarly, the nonideal framework then provides constraints that model weights must then exist within. Notably, for the NKO, the friction term provides increasing resistance the closer to the bounds the model reaches, but those bounds are ultimately inclusive of the NKO's domain.

2.1 Ideal Kinetic Optimizer (IKO)

$$f\left(X_1^{(i)}, X_2^{(i)}, \Delta\right) = Y^{(i)} = X_1^{(i)} + \Delta\left(X_1^{(i)} - X_2^{(i)}\right) \tag{2.1.1}$$

$$F(X_1, X_2, \Delta) = Y = X_1 + \Delta(X_1 - X_2) \tag{2.1.2}$$

$$\Delta \geq 0 \tag{2.1.3}$$

$$X_1^{(i)}, X_2^{(i)}, Y^{(i)} \in (-\infty, +\infty) \tag{2.1.4}$$

The feature-specific update function is given in equation (2.1.1) with the matrix form given in equation (2.1.2). The Δ term indicates the performance gradient between the two models X_1 and X_2, where X_1 is the better performing model. As such, then Δ must be ≥ 0. While a system where $\Delta < 0$ is possible, indicating that X_2 outperformed X_1, Y will simply converge to X_2 in such cases.

Notably, there is no hard definition for Δ. In effect, the Δ will be defined with respect to the individual problem space. Later in the experiments, the definition will be provided for both the IKO and NKO. A potential risk of the kinetic optimizers is that, depending on this definition of Δ, Y can oscillate between two or more points. Therefore, some knowledge of the problem space must be established such that it is guaranteed that the optimizers will converge.

Algorithm 2.1.1: Template Ideal Kinetic Optimizer
Inputs: A loss function f_l and a gradient function f_Δ that is a function of the losses of X_1 and X_2. Output: Model trained using the ideal kinetic optimizer.
(1) Randomly initialize models A and B. (2) Identify the initial X_1 and X_2 models. $\boldsymbol{X_1, X_2 = (A, B)}$ **sorted by** $\boldsymbol{\left(f_l(A), f_l(B)\right)}$ **in ascending order** (3) Train the model until it converges. **while training** (4) Calculate the gradient Δ. $\boldsymbol{\Delta = f_\Delta\left(f_l(X_1), f_l(X_2)\right)}$ (5) Calculate the updated X_2. $\boldsymbol{X_2 = X_1 + \Delta(X_1 - X_2)}$ (6) Potentially swap X_1 and X_2 if the loss of the updated X_2 is less than that of X_1. $\boldsymbol{X_1, X_2 = (X_1, X_2)}$ **sorted by** $\boldsymbol{\left(f_l(X_1), f_l(X_2)\right)}$ **in ascending order** **end while** (7) Return the optimized model. **Return** $\boldsymbol{X_1}$

The IKO algorithm provided in algorithm (2.1.1) is a generalized template that exemplifies the general process by which the IKO operates. Similar to the definition of Δ, this will vary somewhat on the problem space. The more exact algorithm implementation for the clustering problem is similarly given in the experiments section. Notably, this templated optimizer can be applicable for other non-clustering domains, such as a basis for mutation in genetic algorithms or means by which a problem space is explored in other reinforcement learning domains.

2.2 Nonideal Kinetic Optimizer (NKO)

$$f\left(X_1^{(i)}, X_2^{(i)}, \Delta; \beta, \alpha\right) = Y^{(i)} = X_1^{(i)} + \frac{\Delta}{\beta - \alpha}\left(\Gamma_i - X_1^{(i)}\right)\left(X_1^{(i)} - X_2^{(i)}\right) \tag{2.2.1}$$

$$F(X_1, X_2, \Delta; \beta, \alpha) = Y = X_1^{(i)} + \frac{\Delta}{\beta - \alpha}(\Gamma - X_1) \cdot (X_1 - X_2) \tag{2.2.2}$$

$$0 \leq \Delta \leq 1 \tag{2.2.3}$$

$$X_1^{(i)}, X_2^{(i)}, Y^{(i)} \in [\alpha, \beta] \tag{2.2.4}$$

$$\left(\Gamma_i = \alpha \cap X_1^{(i)} \leq X_2^{(i)}\right) \cup (\Gamma_i = \beta) \tag{2.2.5}$$

The nonideal variant of the ideal kinetic update rule takes two additional arguments, specifying the bounds of the space: α, β. Furthermore, Δ is constrained to be ≤ 1 in addition to the other ideal constraints of Δ given in equation (2.1.3). Therefore, for cases where it is desirable or necessary to constrain the model to specific domains, the NKO can be used in place of the IKO to achieve similar results. However, for the algorithmic similarities, the demonstrated performance of solutions derived using NKO are typically worse than the those using IKO, as will be further explored in the experiments.

Algorithm 2.2.2: Example Nonideal Kinetic Optimizer

Inputs: A loss function f_l and a gradient function f_Δ that is a function of the losses of X_1 and X_2. Additionally, a domain upper and lower bound, respectively denoted as β and α.
Output: Model trained using the ideal kinetic optimizer.

(1) Randomly initialize models A and B.
(8) Identify the initial X_1 and X_2 models.
$\boldsymbol{X_1, X_2 = (A, B)}$ **sorted by** $\boldsymbol{\left(f_l(A), f_l(B)\right)}$ **in ascending order**
(2) Train the model until it converges.
while training
(3) Calculate the gradient Δ.
$$\boldsymbol{\Delta = f_\Delta\left(f_l(X_1), f_l(X_2)\right)}$$
(4) Calculate the bounding Γ to impose domain constraints on the updated state.
$$\boldsymbol{\forall i \in \{1, 2, \dots, |X_1|\} \rightarrow \left(\Gamma_i = \alpha \cap X_1^{(i)} \leq X_2^{(i)}\right) \cup \Gamma_i = \beta}$$
(5) Calculate the updated X_2.
$$\boldsymbol{X_2 = X_1 + \frac{\Delta}{\beta - \alpha}(\Gamma - X_1) \cdot (X_1 - X_2)}$$
(9) Potentially swap X_1 and X_2 if the loss of the updated X_2 is less than that of X_1.
$\boldsymbol{X_1, X_2 = (X_1, X_2)}$ **sorted by** $\boldsymbol{\left(f_l(X_1), f_l(X_2)\right)}$ **in ascending order**
end while
(6) Return the optimized model.
Return $\boldsymbol{X_1}$

The templated NKO algorithm, defined in algorithm (2.2.2) is similar to the IKO. It should be similarly modified according the problem space, and again the version used in the experiments will also be detailed later on.

3 KNN Experiments

In addition to the two types of kinetic optimizers (IKO and NKO), two types of kinetic clustering algorithms were designed: loss-timeout and cluster-timeout. The cluster-timeout design is functionally similar to how the traditional k-means clustering algorithm's stopping condition. The k-means clustering algorithm stops once the averaged clusters—for each cluster—are equal

to the current set of classifications. The cluster-timeout for the kinetic optimizer then similar details that the algorithm is halted once the clusters do not change from the previous set of classifications. At which point, the center of the clusters is evaluated and finalized as the solution. Loss-timeout functions differently, in that a loss threshold is predetermined and the optimization algorithm continues until the loss differences between a generation and the one immediately prior is less-than-or-equal, per each cluster, than the loss-timeout threshold.

A number of environmental parameters were selected and each set repeatedly experimented ($n_{\text{trials}} = 30$) to derive sufficiently high confidence in the results of that parameter configuration. Classifications and classifications are randomly generated for each trial given a set of environmental parameters, but remain consistent for the tested algorithms: k-means clustering, IKO with loss-timeout (IKO-loss), IKO with cluster-timeout (IKO-cluster), NKO with loss timeout (NKO-loss), and NKO with cluster-timeout (NKO-cluster). The range of explored parameters include the count of cluster centers k being learned, classes c, dimensions d, and sampled observations n_{samples}. The standard deviations of how classifications are defined, as well as samples thereof, are also explored parameters, given respectively as σ_{classes} and σ_{samples}. Notably, classes are normally distributed with a zero mean, and then samples are distributed with their respective class as their distribution's mean. The cartesian product of these sets of parameters is taken to be the set of environmental parameters to be tested with each algorithm.

$$k, c, \sigma_{\text{classes}}, \sigma_{\text{samples}}, d \in \{2,4, \dots ,10\} \tag{3.1}$$

$$n_{\text{samples}} \in \{m_{\text{samples}}, \dots ,55\} \tag{3.2}$$

$$m_{\text{samples}} = \begin{cases} \left\lceil \frac{k+1}{c} \right\rceil \text{ rounded up to nearest increment of } 5 & k > c \\ 5 \quad \text{else} \end{cases} \tag{3.3}$$

Important Note: Due to data loss, the parameters tested for the cluster-timeout kinetic optimizer variants is $\boldsymbol{k, c, \sigma_{\text{classes}}, \sigma_{\text{samples}}, d \in \{2, 6, 10\}}$, which was done to quickly regenerate the largely representative datapoints. However, this did significantly reduce the number of datapoints in the resulting plots and averaged analysis.

The set of parameters here was done to explore the more minimal cases in which clustering could be needed to situations where there is significant noise or other environmental complexities. These complexities typically entail the imbalances between the number of classifications c and the number of clusters k used to identify those classifications and large between datapoints given a higher amount of sample standard deviations with reduced dimensionality to provide sufficient distances between the points to easily separate the datapoints. The number of samples is made to be symmetric of $n_{\text{samples}} = 30$, which is the point at which per the Central Limit Theorem that the population's mean and standard deviation approximates a normal distribution. Which is a notable property, as a normal distribution is symmetrical. Therefore, going from $n_{\text{samples}} = 5$ to $n_{\text{samples}} = 55$ evaluates a balanced distribution amongst low confidence and high confidence that the distribution means of the sampled items for each classification then accurately approximate that classification.

$$\Delta = \frac{\text{loss}(X_1)}{\text{loss}(X_1) + \text{prev\_loss}(X_1)} \tag{3.4}$$

For all kinetic optimizers, the same definition of Δ is used, given in equation (3.4). Tangentially noting that other variants were provided, but this was the best one discovered in terms of the reliability at which the kinetic optimizers performed better, in terms of classification error, than the traditional k-means clustering algorithm. Though it was observed some variants of Δ could result in the kinetic optimizer converging faster or being less likely to be worse than the k-means clustering algorithm at the cost of less likely to actually be any better (equivalent losses).

For the kinetic optimizer algorithm implementations, the better performing model, as demonstrated by the monotonically decreasing algorithm, is the averaged center point of the items that are grouped together in a single generation. This is then the definition of X_1, where then X_2 is the previous cluster center. Using these definitions of X_1 and X_2, the Δ used by the kinetic optimizers is given in equation (5.4). For the nonideal kinetic optimizer, the α and β terms were set to be equal to the minimum and maximum observation, respectively.

Algorithm 3.1: Kinetic Optimization Clustering Algorithm

Inputs: The kinetic optimization function f_o to be either the clustering IKO or NKO. If it is NKO, parameters α, β are also required. Additionally, the items to be clustered and the k initial cluster centers. Note the gradient function f_Δ is given in equation (3.4) and the loss function f_l is simply the aggregated mean absolute error across all clusters.
Output: Model trained using the ideal kinetic optimizer.

(1) Initialize variables.
centers_next = centers = initial centers
clusters_prev = empty list
cluster_losses_prev = $[\infty_1, \infty_2, \ldots, \infty_k]$
(2) Train model.
while true:
(3) Calculate clusters and cluster losses. Note the cluster losses are given to be weighted proportional to the number of items in each cluster.
clusters,cluster_losses = generate_clusters(centers,items)
(4) Calculate the next cluster centers. X_1 is the list of averaged clusters and X_2 the current cluster centers.
X_1 = [average(cluster,axis = 0) for cluster in clusters]
X_2 = centers
$\Delta = f_\Delta(X_1, X_2)$
centers_next = $f_o(X_1, X_2, \Delta; \alpha, \beta)$
(5) Evaluate stopping criteria.
(a) Cluster-Timeout
if clusters_prev = clusters:
return X_1
(b) Loss-Timeout
if all([$|f_l(X_1) - f_l(X_2)| \leq$ loss_threshold]):
return X_1 if $f_l(X_1) \leq f_l(X_2)$ else X_2
cluster_losses_prev = cluster_losses

For the sake of brevity, it is excluded from algorithm (3.1) that for the f_0 if $\Delta < 0$ then X_1 and X_2 are swapped and Δ is reassigned to be $\Delta = -\Delta$. As mentioned earlier, depending on the definition of Δ it is possible for the output—given as the next set of centers here—to oscillate between a number of values without converging. Similarly, it is possible for the kinetic optimizer to overshoot such that the next set of centers is worse than the previous one. Therefore, some care needs to be in place such that the updated states are with respect to the current best model. Finally, note the loss threshold used for the timeout is 0.0000001.

4 Results

Experiment result are summarized in figure (4-1) (right). In general, the IKO algorithms yielded a performance gain over the k-means clustering algorithm. Additionally, the cluster-timeout variant on average reduced the number of steps required to converge by 5%. While the cluster-timeout variants either performed better or slightly worse than the k-means algorithm, again averaging for all experimented datapoints, the loss-timeout variants took substantially longer to converge. Notably, a smaller loss threshold would significantly reduce the time cost, but at a small hit to the performance gain.

Optimizer	Average Relative Improvement (%)	
	Loss	Steps
iko-clusters	0.1870%	4.9780%
nko-clusters	-0.0301%	-1.0782%
iko-loss	0.2011%	-54.9904%
nko-loss	0.0269%	-44.9382%

Figure 4-1 - Experiment Results Summary

A linear regression was performed to identify the key contributing factors to both the time and performances outcomes given a set of environmental parameters, which was broken down into simpler more interpretable heuristics.

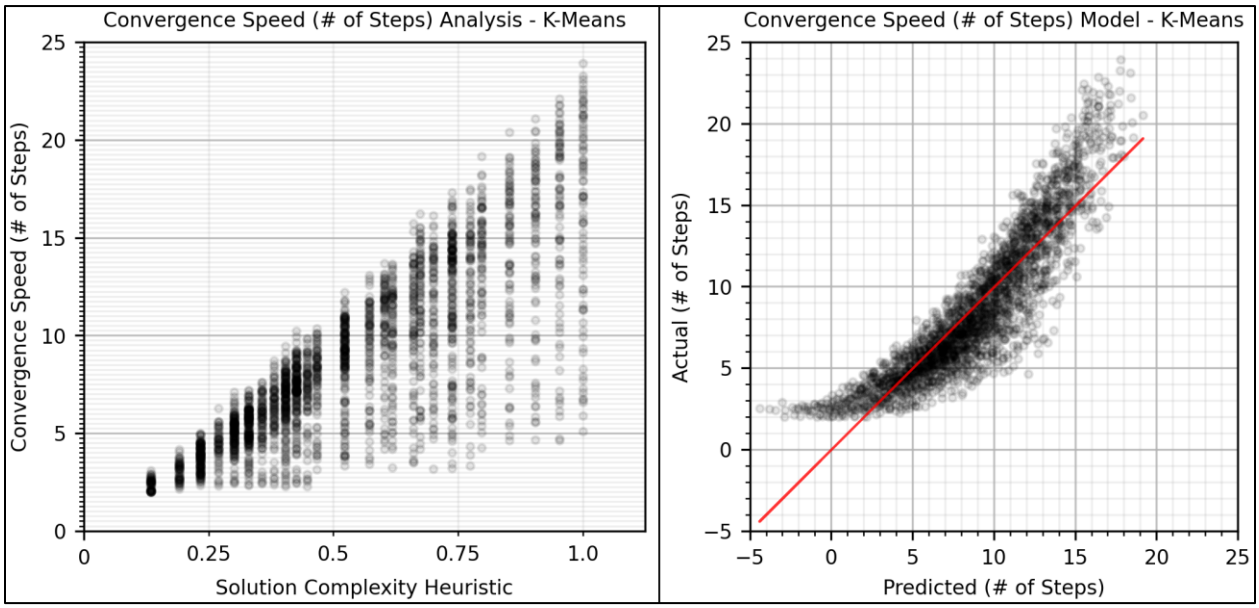

Figure 4-2 – K-Means Converge Speed (# Steps) Heuristic Analysis (Left) and Linear Regression Model (Right)

Where the linear regression model for the k-means clustering algorithm to estimate the convergence speed—the number of steps it takes to converge—with a model prediction error (RMSE) of 0.438179.

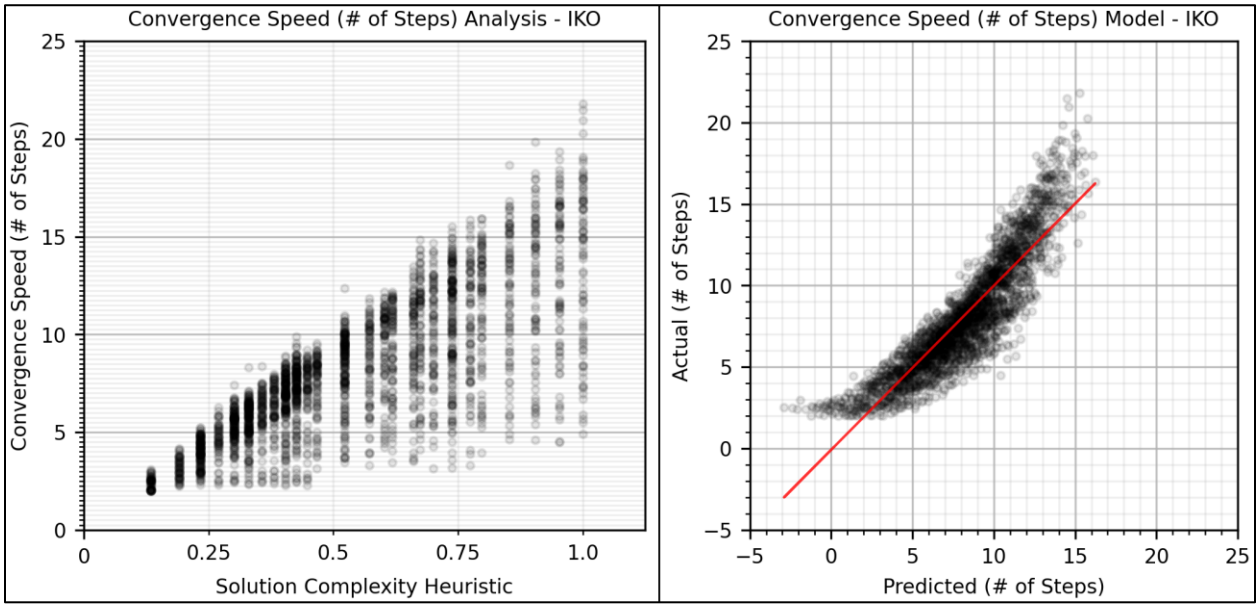

Figure 4-3 - IKO-Cluster Converge Speed (# Steps) Heuristic Analysis (Left) and Linear Regression Model (Right)

Where the linear regression model for the IKO-cluster clustering algorithm to estimate the convergence speed—the number of steps it takes to converge— with a model prediction error (RMSE) of 0.451330.

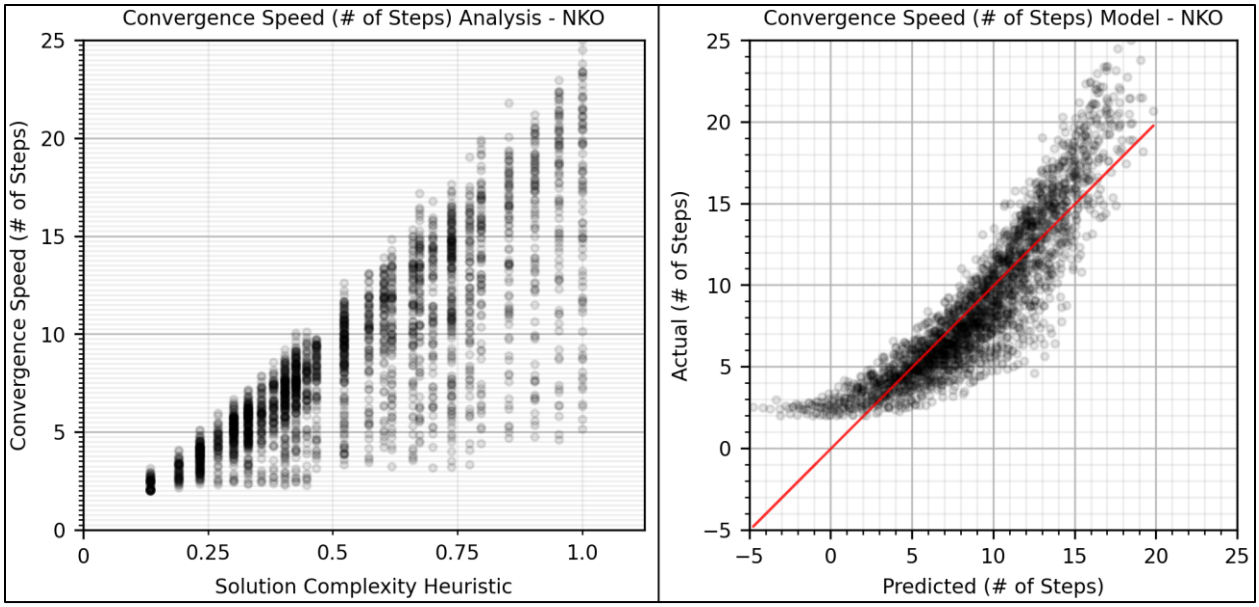

Figure 4-4 – NKO-Cluster Converge Speed (# Steps) Heuristic Analysis (Left) and Linear Regression Model (Right)

Where the linear regression model for the NKO-cluster clustering algorithm to estimate the convergence speed—the number of steps it takes to converge—with a model prediction error (RMSE) of 0.446305.

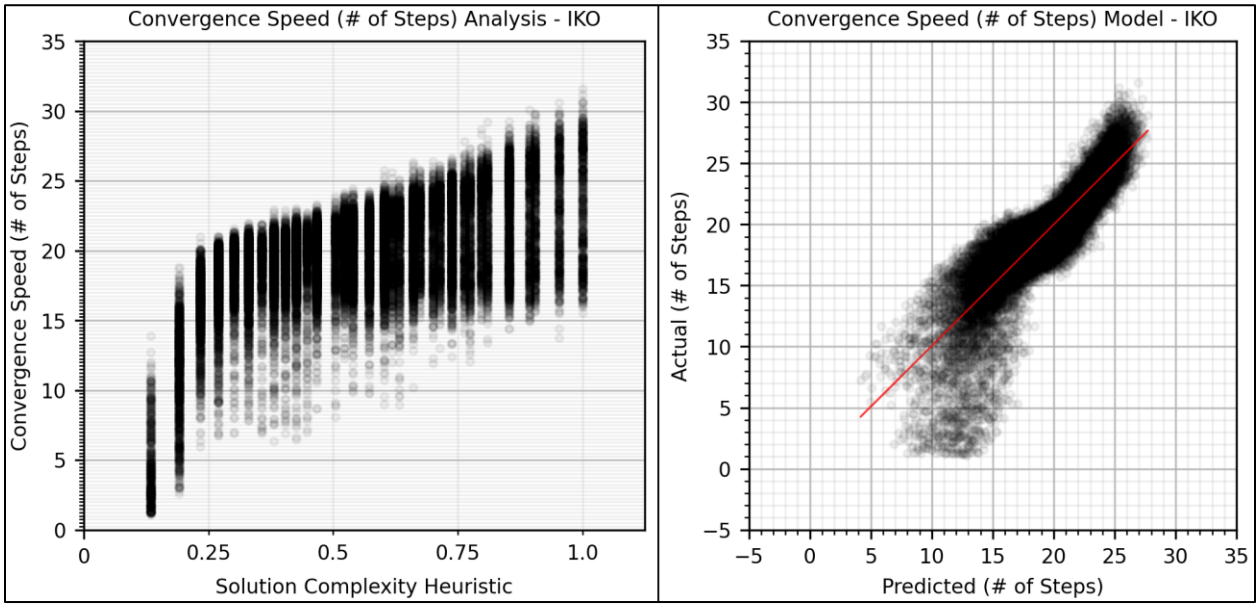

Figure 4-5 - IKO-Loss Convergence Speed (# Steps) Heuristic Analysis (Left) and Linear Regression Model (Right)

Where the linear regression model for the IKO-loss clustering algorithm to estimate the convergence speed—the number of steps it takes to converge—with a model prediction error (RMSE) of 0.471936.

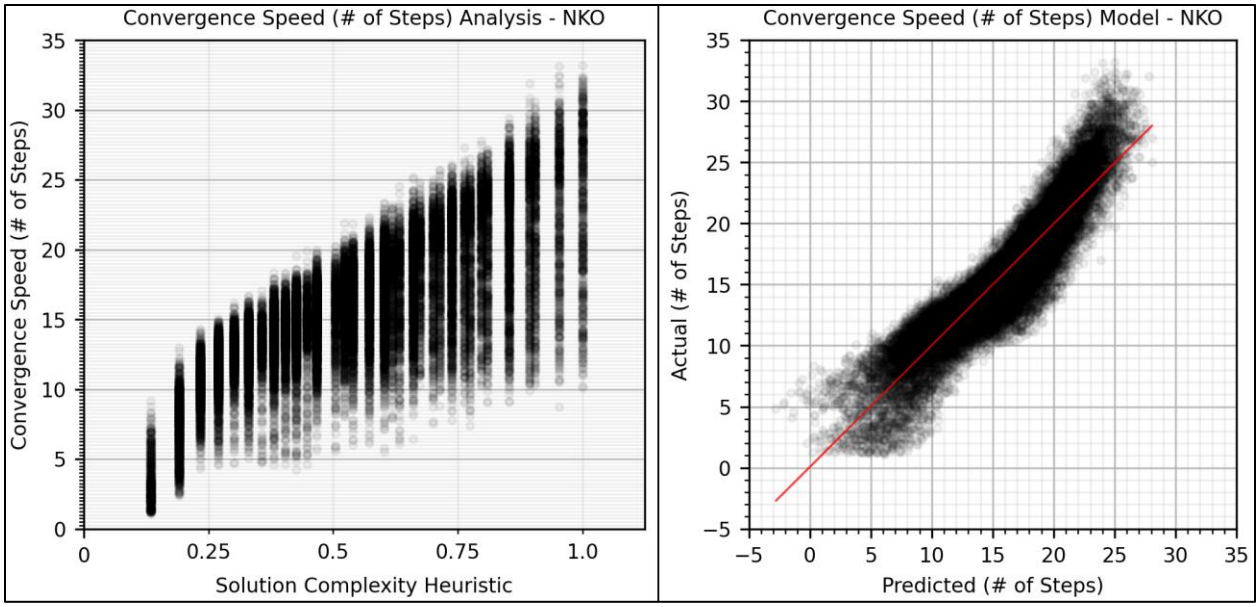

Figure 4-6 - NKO-Loss Convergence Speed (# Steps) Heuristic Analysis (Left) and Linear Regression Model (Right)

Where the linear regression model for the NKO-cluster clustering algorithm to estimate the convergence speed—the number of steps it takes to converge—with a model prediction error (RMSE) of 0.380761.

The solution complexity heuristic estimates how many steps it will take for a model to converge, and is given relative to the number of classes and the number of samples in a space.

$$\text{solution\_complexity\_heuristic} = \text{normalized}\left(\sqrt{c * n_{\text{samples}}}\right) \tag{4.1}$$

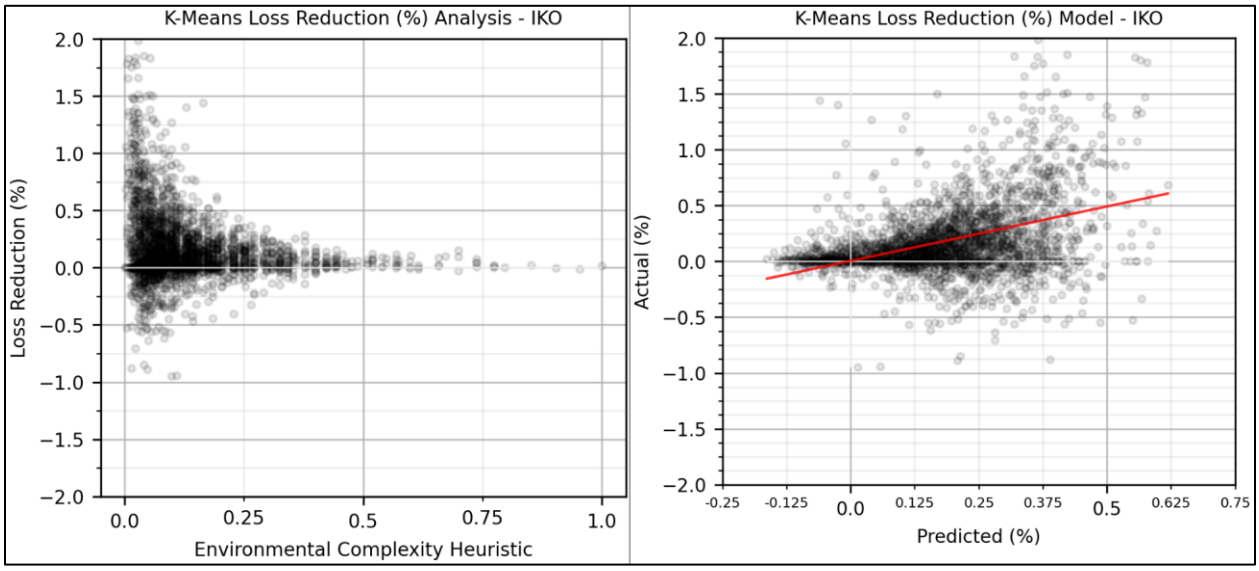

Figure 4-7 - IKO-Cluster Loss Reduction (MAE) Heuristic Analysis (Left) and Linear Regression Model (Right)

Where the linear regression model for the NKO-cluster clustering algorithm to estimate the loss reduction—the percentage that the kinetic optimizer reduced the error by with respect to the k-means clustering algorithm—with a model prediction error (RMSE) of 0.885325.

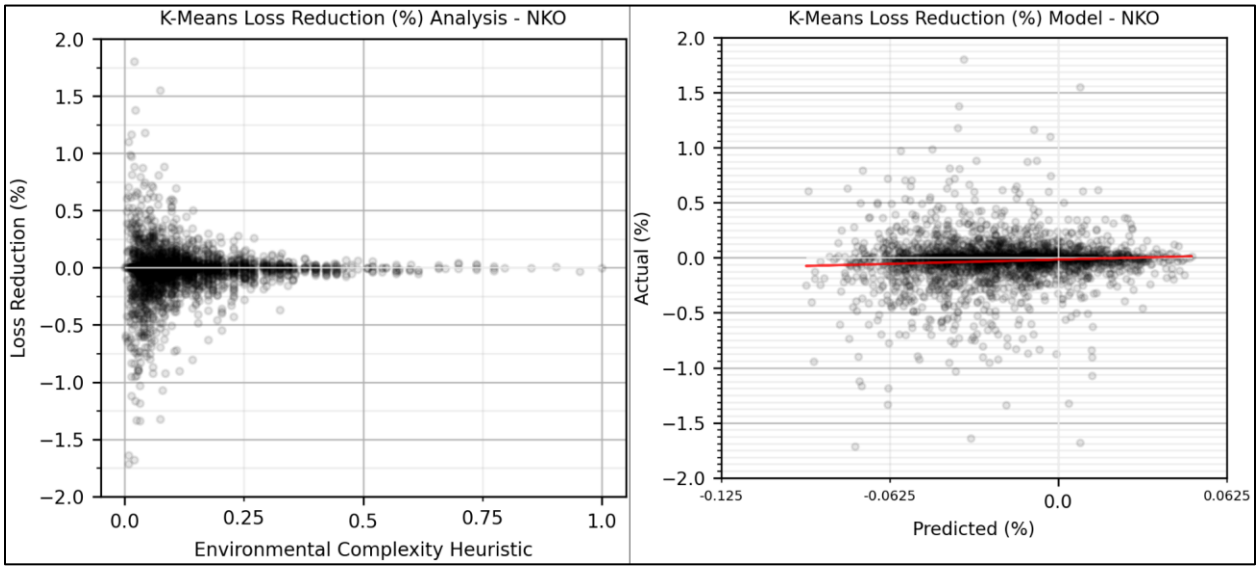

Figure 4-8 - NKO-Cluster Loss Reduction (MAE) Heuristic Analysis (Left) and Linear Regression Model (Right)

Where the linear regression model for the NKO-cluster clustering algorithm to estimate the loss reduction—the percentage that the kinetic optimizer reduced the error by with respect to the k-means clustering algorithm—with a model prediction error (RMSE) of 0.916308.

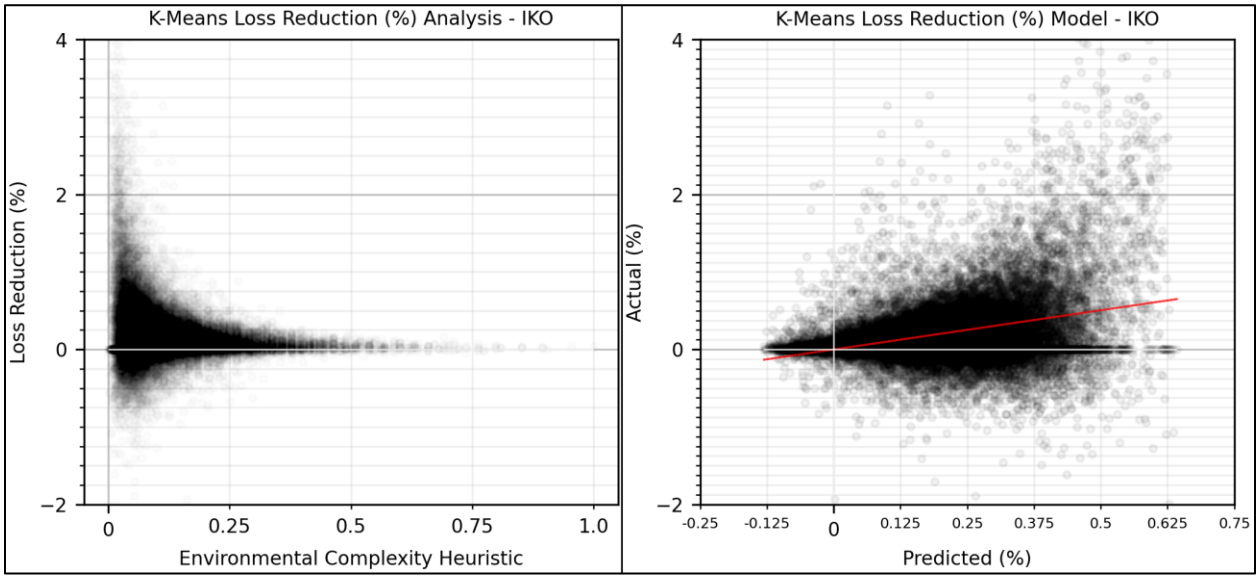

Figure 4-9 - IKO-Loss Loss Reduction (MAE) Heuristic Analysis (Left) and Linear Regression Model (Right)

Where the linear regression model for the NKO-cluster clustering algorithm to estimate the loss reduction—the percentage that the kinetic optimizer reduced the error by with respect to the k-means clustering algorithm—with a model prediction error (RMSE) of 0.958036.

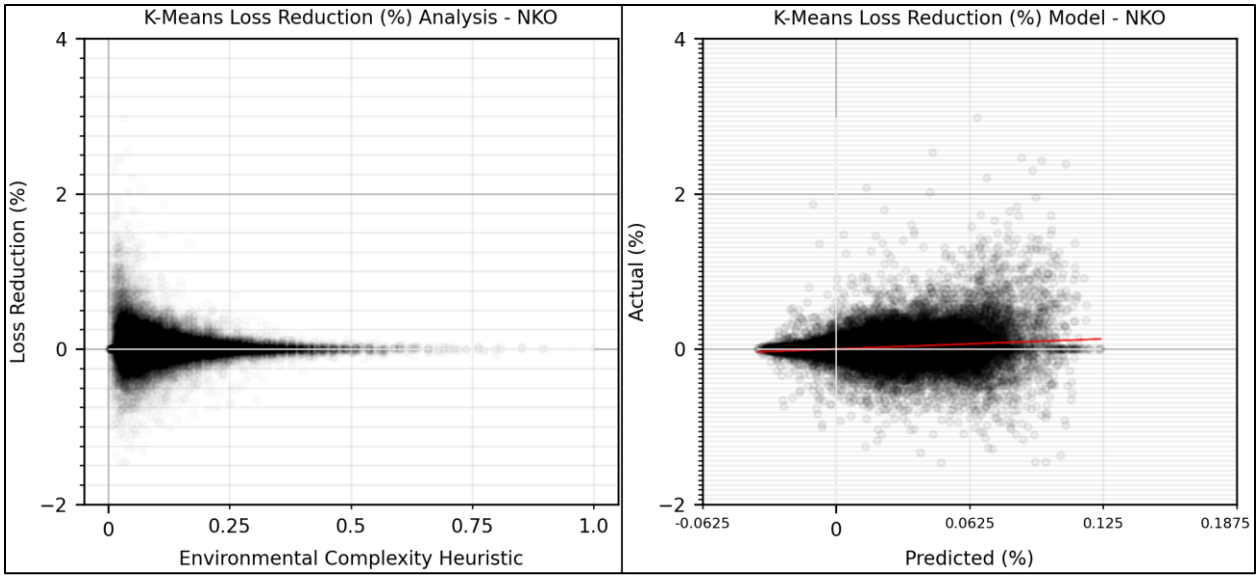

Figure 4-10 - NKO-Loss Loss Reduction (MAE) Heuristic Analysis (Left) and Linear Regression Model (Right)

Where the linear regression model for the NKO-cluster clustering algorithm to estimate the loss reduction—the percentage that the kinetic optimizer reduced the error by with respect to the k-means clustering algorithm—with a model prediction error (RMSE) of 0.974309.

The environmental complexity heuristic estimates how lossy a model converge will be. It is proportional to the number of classes, samples, and dimensions and the standard deviation of samples. Additionally, it is inversely proportional to the number of clusters and standard deviation of classes.

$$\text{environmental\_complexity\_heuristic} = \text{normalized}\left(\sqrt{\frac{c * n_{\text{samples}} * \sigma_{\text{samples}} * d}{k * \sigma_{\text{classes}}}}\right) \quad (4.1)$$

5 Conclusions

Multiple kinetic optimizers are developed and explored where it is demonstrated that given certain environmental parameter configurations of a clustering problem space, the kinetic optimizers can potentially outperform the traditional k-means clustering algorithm with respect to final loss and speed at which the models converge. Specifically, the IKO algorithm using the clustering-timeout protocol superseded the k-means algorithm both in loss and speed in the tests. However, the IKO algorithm using the loss-timeout protocol, while significantly worse in terms of speed, added further marginal loss reduction over the cluster-timeout variant. The NKO algorithms typically perform worse than both the IKO variants, but provide a technique by which the bounds of a model can be constrained given further external requirements.

In addition to these findings, the kinetic optimizers can likely be implemented to other unsupervised and reinforcement learning problems, but as tangentially discovered—though only lightly explored—the definition of Δ is critical. Furthermore, the definition of Δ requires problem space knowledge to understand how the model would update and will likely be inflexible across multiple domains if they are not sufficiently similar. This is the key controlling factor of whether or not the kinetic optimizers can be a likely better choice than other implementations, and may require a system to estimate a sufficiently optimal function, such as a parameter sweeping model.

6 References

[1] Temby, Luke & Vamplew, Peter & Berry, Adam. (2005). Accelerating Real-Valued Genetic Algorithms Using Mutation-with-Momentum. 3809. 1108-1111. 10.1007/11589990_149.

GitHub repo with experiment source code: https://github.com/ZackaryHoyt/kinetic_optimizers

7 Proofs

7.1 Ideal Kinetic Update

$$f\left(X_1^{(i)}, X_2^{(i)}, \Delta\right) = X_1^{(i)} + \Delta\left(X_1^{(i)} - X_2^{(i)}\right) \tag{7.1.1}$$

For the ideal kinematic update rule from equation (2.1.1), restated in (7.1.1) for convenience, prove that $f \in (-\infty, +\infty)$.

Step 1. Solving the partial derivative with respect to $X_1^{(i)}$.

$$\frac{\partial f\left(X_1^{(i)}, X_2^{(i)}, \Delta\right)}{\partial X_1^{(i)}} = 1 + \Delta \tag{7.1.2}$$

This shows that Δ is always the gradient by which the weight $X_1^{(i)}$ is updated. Given the constraint $\Delta \geq 0$, also detailed in equation (2.1.3), then the bounds of f are therefore given to be in $(-\infty, +\infty)$.

Q.E.D.

7.2 Nonideal Kinetic Update

$$f\left(X_1^{(i)}, X_2^{(i)}, \Delta; \beta, \alpha\right) = X_1^{(i)} + \frac{\Delta}{\beta - \alpha}\left(\Gamma_{\mathrm{i}} - X_1^{(i)}\right)\left(X_1^{(i)} - X_2^{(i)}\right) \tag{7.2.1}$$

For the variable nonideal kinetic update rule from equation (2.2.1), restated in equation (7.2.1), prove that $f \in [\alpha, \beta]$. Note the variable nonideal kinematic update rule is also applicable to the standard nonideal kinematic update rule, as $\alpha = -1, \beta = 1$ in that variant. The bounds of the function are proven to hold when the maximized value of f is equal to β when $X_1^{(i)} \geq X_2^{(i)}$, else the minimal value of f is equal to α when the complementary conditional is true: $X_1^{(i)} < X_2^{(i)}$.

To prove this, the critical parameters of f—the parameters that the result of f is its critical values—are calculated by solving the partial derivatives. This will identify a critical $X_1^{(i)}$ and $X_2^{(i)}$, denoted as $\hat{X}_1^{(i)}$ and $\hat{X}_2^{(i)}$ respectively and show the two critical points of f yield the values α and β. Thus, proving that $f \in [\alpha, \beta]$.

Let $\delta = \beta - \alpha$

$$f\left(X_1^{(i)}, X_2^{(i)}, \Delta; \beta, \alpha\right) = X_1^{(i)} + \frac{\Delta}{\delta}\left(\Gamma_{\mathrm{i}} - X_1^{(i)}\right)\left(X_1^{(i)} - X_2^{(i)}\right) \tag{7.2.2}$$

Compute the critical parameters that identify the critical points of f: $\hat{X}_1^{(i)}, \hat{X}_2^{(i)}$.

Step 1. Expanding equation (7.2.2).

$$f\left(X_1^{(i)}, X_2^{(i)}, \Delta; \beta, \alpha\right) = X_1^{(i)} + \frac{\Delta}{\delta}\left(\Gamma_{\mathrm{i}} X_1^{(i)} - \Gamma_{\mathrm{i}} X_2^{(i)} - \left(X_1^{(i)}\right)^2 + X_1^{(i)} X_2^{(i)}\right) \tag{7.2.3}$$

Step 2. Solving the partial derivative with respect to $X_1^{(i)}$.

$$\frac{\partial f\left(X_1^{(i)}, X_2^{(i)}, \Delta; \beta, \alpha\right)}{\partial X_1^{(i)}} = 1 + \frac{\Delta}{\delta}\left(\Gamma_{\mathrm{i}} - 2X_1^{(i)} + X_2^{(i)}\right) \tag{7.2.4}$$

Step 3. Solving (7.2.4) for $\hat{X}_1^{(i)}$.

$$\frac{\partial f\left(\hat{X}_1^{(i)}, X_2^{(i)}, \Delta; \beta, \alpha\right)}{\partial X_1^{(i)}} = 0 \to 1 + \frac{\Delta}{\delta}\left(\Gamma_{\mathrm{i}} - 2\hat{X}_1^{(i)} + X_2^{(i)}\right) = 0 \tag{7.2.5}$$

$$\to \hat{X}_1^{(i)} = \frac{\delta + \Delta\Gamma_{\mathrm{i}}}{2\Delta} + \frac{X_2^{(i)}}{2} \tag{7.2.6}$$

Step 4. Substituting back into equation (7.2.2) with the critical parameter $\hat{X}_1^{(i)}$.

Let $\gamma_i = \frac{\Delta\Gamma_{\mathrm{i}} + \delta}{\Delta}$.

$$f\left(\hat{X}_1^{(i)}, X_2^{(i)}, \Delta; \beta, \alpha\right) = \left(\frac{\gamma_i}{2} + \frac{X_2^{(i)}}{2}\right) + \frac{\Delta}{\delta}\left(\Gamma_{\mathrm{i}} - \left(\frac{\gamma_i}{2} + \frac{X_2^{(i)}}{2}\right)\right)\left(\left(\frac{\gamma_i}{2} + \frac{X_2^{(i)}}{2}\right) - X_2^{(i)}\right) \tag{7.2.7}$$

Step 5. Expanding equation (7.2.7).

$$f\left(\hat{X}_1^{(i)}, X_2^{(i)}, \Delta; \beta, \alpha\right) = \frac{\gamma_i}{2} + \frac{X_2^{(i)}}{2} + \frac{\Delta}{\delta}\left(\Gamma_{\mathrm{i}} - \frac{\gamma_i}{2} - \frac{X_2^{(i)}}{2}\right)\left(\frac{\gamma_i}{2} - \frac{X_2^{(i)}}{2}\right) \tag{7.2.8}$$

$$= \frac{\gamma_i}{2} + \frac{X_2^{(i)}}{2} + \frac{1}{4}\frac{\Delta}{\delta}\left(2\Gamma_{\mathrm{i}} - \gamma_i - X_2^{(i)}\right)\left(\gamma_i - X_2^{(i)}\right) \tag{7.2.9}$$

$$= \frac{\gamma_i}{2} + \frac{X_2^{(i)}}{2} + \frac{1}{4}\frac{\Delta}{\delta}\left(2\Gamma\gamma_i - 2\Gamma_{\mathrm{i}}X_2^{(i)} - \gamma_i^2 + \gamma_i X_2^{(i)} - \gamma_i X_2^{(i)} + \left(X_2^{(i)}\right)^2\right) \tag{7.2.10}$$

Step 6. Solving the partial derivative with respect to $x_2^{(i)}$.

$$\frac{\partial f\left(\hat{X}_1^{(i)}, X_2^{(i)}, \Delta; \beta, \alpha\right)}{\partial X_2^{(i)}} = \frac{1}{2} + \frac{1}{4}\frac{\Delta}{\delta}\left(-2\Gamma_{\mathrm{i}} + 2X_2^{(i)}\right) \tag{7.2.11}$$

Step 7. Solving (7.2.11) for $X_2^{(i)}$.

$$\frac{\partial f\left(\hat{X}_1^{(i)}, \hat{X}_2^{(i)}, \Delta; \beta, \alpha\right)}{\partial X_2^{(i)}} = 0 \to \frac{1}{2} + \frac{1}{4}\frac{\Delta}{\delta}\left(-2\Gamma_{\mathrm{i}} + 2\hat{X}_2^{(i)}\right) = 0 \tag{7.2.12}$$

$$\to \hat{X}_2^{(i)} = \frac{\Delta\Gamma_i - \delta}{\Delta} \tag{7.2.13}$$

Step 8. Substituting back into equation (7.2.8) with the critical parameter $\hat{X}_2^{(i)}$.

Note that the equality $\frac{\gamma_i}{2} + \frac{\hat{X}_2^{(i)}}{2} = \Gamma_i$.

$$f\left(\hat{X}_1^{(i)}, \hat{X}_2^{(i)}, \Delta; \beta, \alpha\right) = (\Gamma_i) + \frac{\Delta}{\delta}\left(\Gamma_i - (\Gamma_i)\right)\left((\Gamma) - \left(\frac{\Delta\Gamma_i - \delta}{\Delta}\right)\right) \tag{7.2.14}$$

Step 9. Simplifying equation (7.2.14).

$$f\left(\hat{X}_1^{(i)}, \hat{X}_2^{(i)}, \Delta; \beta, \alpha\right) = \Gamma_i \tag{7.2.15}$$

This verifies that at $X_1^{(i)} = \hat{X}_1^{(i)}$ and $X_2^{(i)} = \hat{X}_2^{(i)}$ the function's critical points are defined by Γ_i. In the case $X_1^{(i)} \geq X_2^{(i)} \rightarrow f = \beta$. Else, in the case $X_1^{(i)} < X_2^{(i)} \rightarrow f = \alpha$. Given the bounds of Δ, $X_1^{(i)}$, and $X_2^{(i)}$, these critical points are then also the minimum and maximum values of f.

$$\therefore f\left(X_1^{(i)}, X_2^{(i)}, \Delta; \beta, \alpha\right) \in [\alpha, \beta]$$

Q.E.D.