HOCHSCHULE
MITTWEIDA
University of Applied Sciences

MASTER THESIS

Mr.
MD RAJIBUL ISLAM, M.Sc.

Investigations of Counterfactuals and Counterfactual metrics in Learning Vector Quantization approaches

Mittweida, July 2025

Faculty of **Applied Computer Sciences and Biosciences**

MASTER THESIS

Investigations of Counterfactuals and Counterfactual metrics in Learning Vector Quantization approaches

Author:
MD RAJIBUL ISLAM

Course of Study:
Applied Mathematics For Network and Data Sciences

Seminar Group:
MA20w1-M

First Examiner:
Prof. Dr. rer. nat. Thomas Villmann

Second Examiner:
Dr. Marika Kaden

Submission:
Mittweida, 01.07.2025

Defense/Evaluation:
Mittweida, 2025

Bibliographic Description

ISLAM, MD RAJIBUL:
Investigations of Counterfactuals and Counterfactual metrics in Learning Vector Quantization approaches. – 2025. – 46 S.
Mittweida, Hochschule Mittweida – University of Applied Sciences, Faculty of Applied Computer Sciences and Biosciences, Master Thesis, 2025.

Abstract

In recent years, counterfactual explanations have emerged as a powerful tool for interpreting machine learning models by identifying minimal input changes that lead to different predictions. While considerable progress has been made for black-box models, limited attention has been given to prototype-based classifiers such as Generalized Learning Vector Quantization (GLVQ), which offer inherent interpretability through prototype representations.

This thesis investigates counterfactual generation and evaluation strategies tailored to GLVQ models. We explore distance-based formulations using weighted Manhattan, generalized Euclidean, and squared Euclidean metrics. To assess plausibility, we incorporate statistical density estimation methods, including Kernel Density Estimation (KDE) and Gaussian Mixture Models (GMM), ensuring that generated counterfactuals lie within high-density regions of the data distribution.

Our methods are evaluated on both 2D synthetic datasets and real-world data (e.g., subsets of MNIST), and are compared against the Counterfactual Explanations for Machine Learning (CEML) framework. Results show that integrating prototype margins into the optimization objective yields counterfactuals that are not only decision-boundary-faithful but also more plausible and interpretable. Visualizations of prototype positions, decision boundaries, and counterfactual trajectories further illustrate the geometric nature of our approach.

This work advances the understanding of counterfactuals in prototype-based models and demonstrates their potential for robust and transparent model auditing.

Contents

List of Figures

List of Tables

Acknowledgement

I would like to express my heartfelt gratitude to my supervisor, Prof. Dr. rer. nat. Thomas Villmann, for his exceptional guidance, constructive feedback, and unwavering support throughout the course of this thesis.

I am equally grateful to Dr. Marika Kaden for kindly agreeing to serve as the second examiner and for her valuable comments and suggestions, which significantly contributed to improving this work.

My deepest thanks go to my entire family and my closest loved ones, whose continuous encouragement, understanding, and belief in my abilities have been a constant source of strength throughout my studies.

I also wish to acknowledge the professors and fellow students of the Applied Mathematics for Network and Data Sciences program at Hochschule Mittweida for fostering an intellectually enriching and collaborative academic environment.

Abbreviations and Symbols

Table 1: List of Acronyms and Mathematical Symbols

Acronym / Symbol	**Description**
— Acronyms —	
NN	Neural Network
LVQ	Learning Vector Quantization
GLVQ	Generalized Learning Vector Quantization
ARS	Attraction-Repulsion Scheme
CF	Counterfactual
MAD	Median Absolute Deviation
CEML	Counterfactual Explanation in Machine Learning
KDE	Kernel Density Estimator
GMM	Gaussian Mixture Model
MNIST	Modified National Institute of Standards and Technology
BMU	Best Matching Unit
WTA	Winner Takes All
GLVQ-OptCF	GLVQ-based Optimized Counterfactuals
— Mathematical Symbols —	
C	Regularization strength
$\theta(\cdot)$	Penalty term or regularization function
ρ	Density threshold for plausibility
$\hat{p}$	Estimated probability density

1 Introduction

1.1 Background

In recent years, machine learning has become an integral component of decision-making systems across diverse domains, including finance to healthcare and beyond. However, the complexity of many machine learning models- particularly black box models like deep neural networks-makes it difficult to understand how they arrive at their predictions[18].
This lack of transparency hinders trust, especially in high-stakes applications such as loan approval, medical diagnosis, or hiring decisions. This is why Explainable Artificial Intelligence (XAI) has become important. XAI focuses on creating tools and methods that help people understand and explain the decisions made by machine learning models.
Among various explanation techniques, counterfactual explanations have emerged as particularly intuitive and actionable form of interpretability [27]. They provide users with the minimal changes required to flip a model's prediction, thereby answering what-if questions in a direct and meaningful way.
While most existing counterfactual methods focus on black-box models, prototype-based learning models, such as Generalized Learning Vector Quantization (GLVQ) [21], offer an inherently interpretable alternative. A new data point is classified by measuring its distance to these prototypes, making the decision process easy to follow and explain. This structure allows us for transparent, human-understandable decision boundaries, in contrast to more complex and unclear such as neural networks.
This thesis investigates counterfactual generation and evaluation within the GLVQ framework and compares it to the well-known counterfactual explanation method CEML [1].Our goals are threefold. First, we formulate and solve the counterfactual generation problem using GLVQ with the squared Euclidean distance, and derive geometric interpretations from both analytical equations and graphical representations (e.g., Figure 3.1). Second, we compare these findings with those from CEML, identifying key similarities and differences in decision boundary behavior. Third, we evaluate the quality of counterfactual explanations on several datasets, including the 2D Circles, Iris, and a subset of MNIST digits.
A core challenge in counterfactual generation is ensuring that the resulting examples are not only minimally perturbed but also realistic. To address this, we evaluate plausibility using Kernel Density Estimation (KDE) and Gaussian Mixture Models (GMM) [4]. These probabilistic models help determine whether the counterfactuals lie in high-density regions of the input space, making them more credible and interpretable.

1.2 The Layout of the Thesis

During the course of this thesis, we structured our work into five distinct phases. In Chapter 2, we presents the theoretical foundation of our study. It introduces key machine learning concepts and elaborates on the mathematical background necessary to understand Learning Vector Quantization(LVQ) and Generalized Learning Vector Quantization. This chapter laid the foundation for the development and implementation of our GLVQ model.

Chapter 3 focuses on the concept of counterfactual explanation. It details the methodology for generating counterfactuals, including illustrative examples, regularization techniques, geometric interpretations in 2D, and strategies for plausibility assessment. The chapter emphasizes a systematic pipeline for computing and evaluating counterfactuals.

In Chapter 4, we describe the practical tools and datasets utilized in our experiments. We worked with three datasets: 2D toy data, the Iris dataset and the MNIST digit dataset. For each dataset, we assessed model performance such as accuracy, and generated counterfactual explanations using the CEML and GLVQ-OptCF 1 frameworks. Visualizations and supporting analyses were included to facilitate interpretation.

Finally, in Chapter 5, we conclude the thesis by summarizing the essential findings and insights gained throughout the project. This chapter reflects on the effectiveness of our GLVQ-based approach for counterfactual generation, its interpretability, and potential applications in decision support systems.

2 Learning Vector Quantization

There are several variants of Learning Vector Quantization algorithms, but the fundamental concepts remain consistent: the use of prototypes and dissimilarities. In this section, we will define the notions of prototypes and their associated dissimilarities.

2.0.1 Distance Metrics in Learning Vector Quantization

We rely on the concept of distance extensively in our daily lives. For example, when trying to locate the nearest train station, we often consider the physical distance between our current location and the station. Similarly, in machine learning models, various mathematical distance measures—also known as metrics—are used to quantify the difference between data points. A solid understanding of both similarity and dissimilarity is fundamental to many machine learning approaches, particularly those involving clustering, classification, or prototype-based learning. In this section, we focus on the concept of dissimilarity metrics, which play a crucial role in comparing and analyzing data within a given feature space.

Definition 2.1 (Metric and Metric Space)
A metric space is a pair (X, d), where X is a non-empty set of objects and d is a function and $d : X \times X \to \mathbb{R}$, called distance metric. Which satisfies the following conditions for all $p, q, r \in X$.

- $d(p, q) \geq 0$, non-negativity.
- $d(p, q) = 0$ if and only if $p = q$ (Identity)
- $d(p, q) = d(q, p)$ (Symmetry)
- $d(p, q) \leq d(q, p) + d(q, r)$ for all $p, q, r \in X$(Triangle Inequality)

Definition 2.2 (Translation-invariant metric)
A metric space (X, d) consists of a set X and a metric d, which measures the distance between any two points in X. Suppose X is equipped with a group operation $+$,

$$+ : X \times X \to X.$$

that forms a group $(X, +)$ on X. The metric X is called translation-invariant if shifting both points p and q by the same amount r doesn't change the distance between points

$$d(p, q) = d(p + r, q + r) \tag{2.1}$$

for all $p, q, r \in X$.

Definition 2.3 (Squared Euclidean Distance)
The squared Euclidean distance between two vectors $p, q \in \mathbb{R}^n$ is defined as:

$$d_E^2(p,q) = \sum_{i=1}^{n} (p_i - q_i)^2 \tag{2.2}$$

or equivalently,

$$d_E^2(p,q) = (p-q)^\top (p-q)$$

It is commonly used in machine learning models, as it avoids the computational cost of the square root while preserving distance ordering.

Definition 2.4 (Euclidean Distance)
Let p and q be the vectors of n-dimensional real vector space $\mathbb{R}^n$. The Euclidean distance can be defined as

$$d_E(p,q) = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2} \tag{2.3}$$

and in terms of vector operation as:

$$d_E(p,q) = \sqrt{(a-b)^T (a-b)} \tag{2.4}$$

Definition 2.5 (Mahalanobis Distance)
Let the given two vectors $p, q \in \mathbb{R}^n$ and a full ranked covariance matrix Σ which must be symmetric positive definite. The Mahalanobis distance is defined as:

$$d_M(p,q) = \sqrt{(p-q)^T \Sigma^{-1} (p-q)} \tag{2.5}$$

- In general, if the covariance matrix $\Sigma = I($*Identity Matrix*$)$, then the Mahalanobis distance reduces to Euclidean distance.

2.0.2 Activation Functions

An activation function is a mathematical function utilized in machine learning models to determine a node's output based on its input. Here are a few commonly used activation functions defined below.[22]

Sigmoid
The sigmoid activation function is a mathematical function commonly used in NN, particularly for binary classification. It has S-shaped, and maps any real valued number to a value between 0 and 1.

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

Properties:

- Range $(0, 1)$
- Smooth and differentiable
- Derivative $\sigma'(x) = \sigma(x)(1 - \sigma(x))$

Binary Step Function:

A binary step function is an activation function commonly used in binary classifiers. It maps an input $x \in \mathbb{R}$ to either 1 or 0 depending in whether x is greater than or equal to a threshold (usually 0):

$$f(x) = \begin{cases} 1, & \text{if } x \geq 0 \\ 0, & \text{if } x < 0 \end{cases}$$

It is not suitable for gradient-based learning methods because its derivative is almost zero everywhere and undefined at $x = 0$.

Rectified Linear Unit(ReLU):

The Rectified Linear Unit (ReLU) is an activation function used in Machine Learning, defined as:

$$f(x) = max(0, x)$$

Where x is the input to the function.
ReLU activates a neuron by outputting the input itself it the input is positive; else, it outputs zero.

Softplus Activation Function:

The softplus activation function [24] is a smooth, differentiable approximation of the ReLU function.It is defined as:

$$f(x) = \ln\left(e^x + 1\right)$$

It is non-linear and continuously differentiable across its domain, providing a gradual transition between active and inactive states.

2.0.3 Learning Vector Quantization

T. Kohonen introduced Learning Vector Quantization (LVQ) [16], a supervised classification algorithm based on the concept of a nearest prototype classifier. The classification method is grounded in identifying the best matching unit (BMU) through a Winner-Takes-All strategy. Currently, several variants of LVQ have been developed, including LVQ1, LVQ2/LVQ2.1, and LVQ3. All these variants operate according to Hebbian Learning principles. To provide an overview, we will first discuss the standard concepts of LVQ before diving into Generalized Learning Vector Quantization (GLVQ). This variety in LVQ methods originates from the need to select a dissimilarity measure tailored to specific classification tasks.

2.0.4 Kohonen's Learning Vector Quantization Algorithms

LVQ, introduced by Kohonen, is one of the most intuitive nearest prototype classifiers [15]. We assume a dataset

$$X = \{\vec{x}_i \in \mathbb{R}^n \mid i = 1, 2, \ldots, N\}$$

where each data vector $\vec{x}$ is associated with a class label s.t. $c(x_i) \in C$ where C is the set of all class labels. Let the set of prototypes be

$$\mathcal{W} = \{\vec{w}_k \in \mathbb{R}^n \mid k = 1, \ldots, P\}$$

each associated with a class label $c(\vec{w}_k)$. The goal is to positions the prototypes $\{\vec{w}_1, \cdots \vec{w}_k\}$ throughout the feature space, so they effectively represent the training data [26]. Since LVQ is a prototype-based model, each class $c \in C$ must be represented at least one $\vec{w}_k$. A new input sample $\vec{x}$ classified using the winner-takes-all (WTA) strategy [15] , which selects the prototype closest to $\vec{x}$.

$$s(\vec{x}) = \arg \min_{k \in 1, \ldots, P} d(\vec{x}, \vec{w}_k) \tag{2.6}$$

Here, d(.,.) denotes a dissimilarity measure [13]. In LVQ, the Squared Euclidean distance define in 2.3 typically used to compute the distance between input $\vec{x}$ and prototype $\vec{w}_k$, .
The predicted class label for $\vec{x}$ is then determined by:

$$c(\vec{x}) = c(\vec{w}_s(\vec{x})) \tag{2.7}$$

Here $\vec{w}_s(\vec{x})$ is the closest(winning) prototype to $\vec{x}$ and its class label is assigned to the data point.

2.0.5 LVQ Learning Scheme

The LVQ learning process is also known as the heuristic LVQ scheme [19]. It follows an intuitive, rule-based approach updating, where updates are based on the proximity between data points and prototypes and along with their corresponding class labels.

The update procedure consists of the following steps:

- At each iteration, a data point $\vec{x} \in \mathbb{R}^n$ is randomly selected from the training set.
- The closest prototype $\vec{w}^*$ is determined using the dissimilarity measure (typically squared euclidean distance), as defined in definition 2.3.
- The update follows the Attraction-Repulsion-Scheme (ARS) [19], where the closest prototype is either attracted to or repelled from the input $\vec{x}$ depending on the correctness of its class label. If the prototype's label matches the data point, it is moved closer; otherwise, it is pushed away.

  The prototype update rule [15] is defined as:

$$\Delta w_{s(\vec{x})} = \eta.\, \psi(\vec{x}, s(\vec{x})) \cdot \left(\vec{x} - \vec{w}_{s(\vec{x})}\right) \tag{2.8}$$

- The attraction-repulsion factor $\psi(x, s(x))$ is defined as:

$$\psi(\vec{x}, s(\vec{x})) = \begin{cases} +1, & \text{if } c(\vec{w}_{s(\vec{x})}) = c(\vec{x}) \\ -1, & \text{otherwise} \end{cases}$$

- η is the learning rate, where $0 < \eta \ll 1$. It controls the magnitude of the prototype update.
- The function $\psi(\vec{x}, s(\vec{x}))$ serves as an indicator that determines whether the winning prototype should be attracted to or repelled from the input, depending on the correctness of its prediction.

While the basic heuristic update rule is straightforward- several cost-function-based variants of LVQ have been proposed to achieve better convergence and facilitate training with gradient descent or optimization strategies [8].

2.0.6 Generalized Learning Vector Quantization

Sato and Yamada introduced a new learning method that generalizes Kohonen's LVQ algorithms, proposing a differentiable cost function and margin-based learning, known as GLVQ loss [21].
We begin with a labeled dataset [15]

$$\mathcal{D} = \{(\vec{x}_1, c(\vec{x}_1)), \ldots, (\vec{x}_m, c(\vec{x}_m))\},$$

where each data point $\vec{x}_i \in \mathbb{R}^n$ has an associated class label $c(\vec{x}_i) \in \{1, \ldots, C\}$.

In addition, we define a set of labeled prototypes

$$\mathcal{W} = \{(\vec{w}_1, c(\vec{w}_1)), \ldots, (\vec{w}_K, c(\vec{w}_K))\},$$

where each prototype $\vec{w}_k \in \mathbb{R}^n$ also has a class label $c(\vec{w}_k) \in \{1, \ldots, C\}$.

Each prototype represents a class, and classification decisions for any input sample are made based on the distance between the sample and all prototypes. The most commonly used distance measure in GLVQ is the squared Euclidean distance, defined as:

$$d(\vec{x}, \vec{w}_k) = \|\vec{x} - \vec{w}_k\|^2$$

To determine the most relevant prototypes for a sample $\vec{x}$, GLVQ identifies:

- The closest correct prototype:

$$d^+(\vec{x}) = \min_{\vec{w}_k : c(\vec{w}_k) = c(\vec{x})} \|\vec{x} - \vec{w}_k\|^2$$

- The closest incorrect prototype:

$$d^-(\vec{x}) = \min_{\vec{w}_k : c(\vec{w}_k) \neq c(\vec{x})} \|\vec{x} - \vec{w}_k\|^2$$

These distances are used to define the margin function, also called the relative distance function [21]:

$$\mu(\vec{x}) = \frac{d^+(\vec{x}) - d^-(\vec{x})}{d^+(\vec{x}) + d^-(\vec{x})}$$

The value of $\mu(\vec{x}) \in [-1, 1]$ indicates the model's classification confidence, with lower values indicating better separation.

$\mu(\vec{x}) < 0$ implies correct classification.
and $\mu(\vec{x}) > 0$ implies misclassification.

To train the GLVQ model, a cost function [21] is minimized over all training samples. It is defined as the sum of a monotonically increasing function ϕ applied to the margin:

$$E_{\text{GLVQ}} = \sum_{i=1}^{N} \phi\left(\mu(\vec{x}_i)\right)$$

The update rule for the prototypes in GLVQ is derived via stochastic gradient descent [13]. For each training sample $\vec{x}$, only the closest correct prototype $\vec{w}^+$ and the closest incorrect prototype $\vec{w}^-$ are updated.

Let α be the learning rate. Then the gradient-based updates [21] are given by:

$$\vec{w}^+ \leftarrow \vec{w}^+ + \alpha \cdot \frac{\partial \phi}{\partial \mu} \cdot \frac{d^-(\vec{x})}{(d^+(\vec{x}) + d^-(\vec{x}))^2} \cdot (\vec{x} - \vec{w}^+)$$

$$\vec{w}^- \leftarrow \vec{w}^- - \alpha \cdot \frac{\partial \phi}{\partial \mu} \cdot \frac{d^+(\vec{x})}{(d^+(\vec{x}) + d^-(\vec{x}))^2} \cdot (\vec{x} - \vec{w}^-)$$

These update rules push the correct prototype $\vec{w}^+$ closer to the input and the incorrect prototype $\vec{w}^-$ further away, depending on the current classification margin $\mu(\vec{x})$ and the choice of ϕ.

3 Counterfactual

3.1 Counterfactual Explanation

A counterfactual explanation shows how a model's decision would change if certain input features were modified [27]. It helps answer the question: *"What would need to be different to get a different outcome?"* These explanations are useful because they offer direct and actionable changes that lead to a different decision.

For example, in a loan application scenario, suppose a person is denied a home loan. A counterfactual explanation might say:

> *"If your credit score had been 700 instead of 620, and your monthly income increased by €1,000, your application would have been approved."*

This type of explanation is intuitive and promotes transparency in automated systems, helping users understand what influenced the decision [27].

While counterfactuals offer simple and clear explanations, they sometimes present too many possible solutions. When several different changes can all flip a prediction, this is referred to as the Rashomon effect [10].

3.2 Counterfactual Vs Adversarial Perturbations

At first glance, counterfactual explanations and adversarial perturbations [27] appear very similar, as both aim to identify minimal changes that alter a model's prediction. However, their objectives differ fundamentally. The purpose of an adversarial perturbation is to *mislead* the model by introducing small, often imperceptible modifications that cause misclassification. In contrast, a counterfactual explanation aims to *clarify* a prediction by identifying the smallest and most interpretable set of feature changes that would result in a different, meaningful prediction.

Consider an input $\vec{x} \in \mathbb{R}^n$, and let $\{\vec{w_k}\} \subset \mathbb{R}^n$ denote the set of class prototypes. In Generalized Learning Vector Quantization, the classifier assigns $\vec{x}$ to the class of the nearest prototype:

$$s(\vec{x}) = \arg\min_k d(\vec{x}, \vec{w_k}),$$

where $d(\vec{x}, \vec{w_k})$ is typically the squared Euclidean distance:

$$d(\vec{x}, \vec{w_k}) = \|\vec{x} - \vec{w_k}\|^2.$$

and the function $s(\vec{x})$ represents the class prediction for a given input $\vec{x}$.

Let y be the true class label of $\vec{x}$. An adversarial example is formed by adding a small perturbation δ to $\vec{x}$, yielding:

$$\vec{x}_{\text{adv}} = \vec{x} + \delta,$$

such that the classifier's prediction changes:

$$s(\vec{x}_{\text{adv}}) \neq y.$$

In this context, the goal of the adversarial perturbation is to produce a minimally altered input that fools the classifier into making an incorrect prediction, often without perceptible change to a human observer. In contrast, counterfactual explanations seek meaningful, sparse modifications to $\vec{x}$ that shift the classification outcome while remaining interpretable and actionable.

3.3 Counterfactual Definition

A counterfactual explanation [27] seeks the smallest possible changes to an input instance $\vec{x}$ that would result in a different (desired) prediction y_{cf} from the model. Suppose $g : \mathbb{R}^d \to Y$ is a prediction function. The goal is to compute a counterfactual $\vec{x}_{\text{cf}} \in \mathbb{R}^d$ for a given input $\vec{x} \in \mathbb{R}^d$.

The optimization formulation [4] is:

$$\arg\min_{\vec{x}_{\text{cf}} \in \mathbb{R}^d} \ell(g(\vec{x}_{\text{cf}}), y_{\text{cf}}) + C \cdot \theta(\vec{x}_{\text{cf}}, \vec{x}_{\text{orig}}) \tag{3.1}$$

Here, $\ell(\cdot)$ represents the loss function, y_{cf} is the desired target prediction, $\theta(\cdot)$ is the penalty term that measures the dissimilarity between $\vec{x}_{\text{cf}}$ and $\vec{x}_{\text{orig}}$, and $C > 0$ controls the strength of the regularization.

The term $\theta(\cdot)$ serves as a regularization (penalty) term. It penalizes large changes, encouraging the counterfactual to stay close to the original input $\vec{x}_{\text{orig}}$, and quantifies how different $\vec{x}_{\text{cf}}$ is from $\vec{x}_{\text{orig}}$. The parameter $C > 0$ controls the strength of the regularization, determining how strongly we prioritize proximity to the original input. The actual counterfactual explanation includes recommendations for actions that modify certain features in specific ways. Additionally, $\theta(\cdot)$ can be interpreted as a measure of the complexity of the counterfactual $\vec{x}_{\text{cf}}$, indicating how challenging it is to implement the suggested changes θ. Since changing some features may be easier or more difficult than modifying others, both the amount and the nature of change contribute to the complexity and feasibility of the explanation [14].

The counterfactual $\vec{x}_{\text{cf}}$ that stays as close as possible to the original input $\vec{x}_{\text{orig}}$ by following Eq. 3.1 is referred to as the *closest counterfactual*. However, the closest counterfactual does not necessarily satisfy the conditions of plausibility and actionability that a useful counterfactual explanation must fulfill.

Counterfactuals are a model-agnostic method [27]; we do not require access to the internal definition or implementation of $g(\cdot)$. Access to an interface for passing inputs to the system and receiving predictions is sufficient.

Furthermore, the goal is to stay as close as possible to the original input $\vec{x}_{\mathrm{orig}}$ without making assumptions about the structure of the data space. Therefore, continuous optimization is possible. In the general context of real-valued vector spaces, two common choices for the regularization term $\theta(\cdot)$ are the weighted Manhattan distance (L_1) and the generalized Euclidean (L_2) distance used in CEML [5].

3.4 Regularization in Counterfactual Generation

To control how much the counterfactual diverges from the original input, we encourage solutions that are realistic, interpretable, and involve minimal change. Without regularization, a model may significantly alter numerous features to adjust its prediction. In this section, we will introduce three common regularization [3] techniques: the weighted Manhattan distance and the generalized Euclidean distance, which optimizes the computations of counterfactuals in CEML. Furthermore, we implement the squared Euclidean distance and compare it with these common regularization methods later.

The Weighted Manhattan distance (L_1)
The distance function $d(.,.)$ is a core feature in counterfactual explainer. The weighted Manhattan distance is also called as L_1 distance is defined as:

$$\theta(\vec{x}_{\mathit{cf}}, \vec{x}_{\mathit{orig}}) = \sum_k a_k \cdot |(\vec{x}_{\mathit{orig}})_k - (\vec{x}_{\mathit{cf}})_k| \tag{3.2}$$

Here, $\alpha_k > 0$ denotes the weight assigned to feature k.
How to chose weights α_k:
One common strategy is :

$$\alpha_k = \frac{1}{\mathrm{MAD}_k}$$

where

$$\mathrm{MAD}_k = \mathrm{median}_{\vec{x}\in\mathcal{D}}\, |(\vec{x})_k - \mathrm{median}_{\vec{x}\in\mathcal{D}}(\vec{x})_k|$$

To avoid local minima, in [27], Wachter et al. use the Manhattan distance weighted by the inverse of each feature's median absolute deviation MAD_k in training dataset $\mathcal{D}$. This offers increased flexibility in adjusting features while considering them 'close' under the distance. Median absolute distance (MAD) is less sensitive to outliers, which leads to more stable computations of distance. The L_1 norm promotes sparse solutions, such as modifying most features to zero. This sparsity reduces the number of features that need to change, making counterfactuals easier to interpret and communicate.

The generalized L_2 distance:
The generalized Euclidean distance L_2 is the extension of the standard Euclidean distance that introduces a positive semi-definite(psd) matrix to Ω to weight and correlate feature dimensions.

$$\theta(\vec{x}_{cf}, \vec{x}_{orig}) = \|\vec{x}_{orig} - \vec{x}_{cf}\|_{\Omega}^2 = (\vec{x}_{orig} - \vec{x}_{cf})^{\top}\Omega(\vec{x}_{orig} - \vec{x}_{cf}) \tag{3.3}$$

The term Ω denotes a symmetric positive semi-definite (s.psd) matrix. When $\Omega = \mathbf{I}$, the L_2 distance reduces to the standard Euclidean distance. The generalized L_2 distance can promote conditional sparsity and may be suitable for counterfactual generation, depending on the choice of Ω.

Squared Euclidean Distance(squared L_2 norm):
The squared euclidean distance is denoted as squared L_2 norm, and defined as:

$$\|\vec{x}_{orig} - \vec{x}_{cf}\|_2^2 = \sum_{k=1}^{p} (x_{orig,k} - x_{cf,k})^2$$

Equivalently using matrix notation:

$$\|\vec{x}_{orig} - \vec{x}_{cf}\|_2^2 = (\vec{x}_{orig} - \vec{x}_{cf})^\top (\vec{x}_{orig} - \vec{x}_{cf})$$

This tends to prefer changes that are small in magnitude and spread uniformly across variables, rather than large changes in a single variable [27]. It does not promote sparsity, as changes are typically distributed across many features.

The choice of optimization method for generating counterfactuals depends on both the machine learning model and the properties of the loss function $\ell(\cdot)$ and the regularization term $\theta(\cdot)$ [27]. If the overall objective is differentiable, gradient-based optimization techniques such as gradient descent, conjugate gradient, or L-BFGS can be applied efficiently. However, when differentiability is not guaranteed, one must rely on black-box optimization methods such as the Downhill-Simplex method (also known as Nelder-Mead) or Powell's method. Although optimization strategies tailored to specific models and regularization terms are often more efficient, such specialized approaches are currently limited to a small number of models in the domain of counterfactual explanation research.

3.5 Counterfactual Explanations of GLVQ

We discussed GLVQ method in section 2.0.6 where the classifier is based on a margin function defined as:

$$\mu(\vec{x}) = \frac{d^+(\vec{x}) - d^-(\vec{x})}{d^+(\vec{x}) + d^-(\vec{x})}$$

Here, $d^+(\vec{x})$ denotes the squared distance from $\vec{x}$ to the closest prototype of the correct class, and $d^-(\vec{x})$ is the distance to the closest prototype of an incorrect class. This margin-based formulation allows GLVQ to not only classify based on proximity but also to encourage better class separation by maximizing the relative distance margin between correct and incorrect prototypes.

To compute a counterfactual $\vec{x}_{cf}$ (as described in 3.1) [5], the goal is to find a modified input that is classified into a desired target class y_{cf} while remaining as close as possible to the original input $\vec{x}_{orig}$. The objective can be expressed as:

$$\arg\min_{\vec{x}_{cf} \in \mathbb{R}^d} \theta(\vec{x}_{cf}, \vec{x}_{orig})$$

Subject to

$$d(\vec{x}_{cf}, \vec{w}_i) + \epsilon \leq d(\vec{x}_{cf}, \vec{w}_j)$$

Here,

$$\theta(\vec{x}_{cf}, \vec{x}_{orig}) = \|\vec{x}_{cf} - \vec{x}_{orig}\|^2$$

the squared Euclidean distance between $\vec{x}_{cf}$ and $\vec{x}_{orig}$.
A constraint is applied to ensure that the margin is negative.
The margin constraint is:

$$\mu(\vec{x}_{cf}) = \frac{d^+(\vec{x}_{cf}) - d^-(\vec{x}_{cf})}{d^+(\vec{x}_{cf}) + d^-(\vec{x}_{cf})} < 0$$

Where:

$$d^+(\vec{x}_{cf}) = \text{min}\|\vec{x}_{cf} - \vec{w}_i\|^2$$

$d^+(\vec{x}_{cf})$ is the distance between $\vec{x}_{cf}$ and the prototype $\vec{w}_i$ of the target class.

$$d^-(\vec{x}_{cf}) = \text{min}\|\vec{x}_{cf} - \vec{w}_j\|^2$$

$d^-(\vec{x}_{cf})$ is the squared euclidean distance between $\vec{x}_{cf}$ and the prototype $\vec{w}_j$ of the incorrect class.

In Convex programming, as discussed in Section 3.5.1, strict inequalities such as $\mu(\vec{x}_{cf}) < 0$ are generally avoided [6]. This is because the corresponding feasible region $\{\vec{x} \mid \mu(\vec{x}) < 0\}$ forms an open set, which complicates feasibility and stability in numerical optimization.

To address this, a small positive value $\epsilon > 0$ is introduced as a countermeasure to ensure that the counterfactual does not lie exactly on the decision boundary. Moreover, optimal solutions in convex programs often lie on the boundary of the feasible region, i.e., where $\mu(\vec{x}) = 0$.

Thus, the constraint can be reformulated as:

$$\mu(\vec{x}_{cf}) + \epsilon \leq 0$$

This modification ensures the the counterfactual lies safely on the correct side of the decision boundary but not exactly on it.

This is equivalent to:

$$d^+(\vec{x}_{cf}) + \epsilon \leq d^-(\vec{x}_{cf}) \tag{3.4}$$

To maintain convexity during optimization in stead of optimizing overall combinations of prototypes. we fix one prototype pair $\vec{w}_i$ target class y_c and $\vec{w}_j \notin y_c$ For every such fixed pair, the constraint becomes:

$$\|\vec{x}_{cf} - \vec{w}_i\|^2 + \epsilon \leq \|\vec{x}_{cf} - \vec{w}_j\|^2 \tag{3.5}$$

Expanding both sides of this inequality and rearranging leads to a linear constraint explain in 3.5.2:

$$(\vec{w}_j - \vec{w}_i)^\top \vec{x}_{cf} + b_{ij} + \epsilon \leq 0$$

where the bias term is defined as:

$$b_{ij} = \frac{1}{2}\left(\|\vec{w}_i\|^2 - \|\vec{w}_j\|^2\right)$$

depends on the prototype positions and does not involve the optimization variable.

The objective function which minimizes the squared euclidean distance to the original input can be expanded as:

$$\|\vec{x}_{cf} - \vec{x}_{orig}\|^2 = \vec{x}_{cf}^\top \vec{x}_{cf} - 2\vec{x}_{orig}^\top \vec{x}_{cf} + \vec{x}_{orig}^\top \vec{x}_{orig}$$

Since the constant term $\vec{x}_{orig}^\top \vec{x}_{orig}$ does not affect the optimization, we can ignore it. The objective becomes :

$$\min_{\vec{x}_{cf}} \ \frac{1}{2}\vec{x}_{cf}^\top \vec{x}_{cf} - \vec{x}_{orig}^\top \vec{x}_{cf}$$

This formulation provide a convex quadratic program (QP)- the objective is quadratic and constraints are linear.

Algorithm 1 GLVQ Counterfactual Generation (GLVQ-OptCF) [5]

```
Require: Original input $\vec{x}_{orig}$, desired target class $y_{cf}$, trained GLVQ model with prototypes $\mathcal{W}$ and
    labels
Ensure: Closest valid counterfactual $\vec{x}_{cf}$
 1: $\vec{x}_{cf} \leftarrow$ null
 2: $\theta_{\min} \leftarrow \infty$
 3: for all $\vec{w}_i \in \mathcal{W}$ such that label$(\vec{w}_i) = y_{cf}$ do
 4:     for all $\vec{w}_j \in \mathcal{W}$ such that label$(\vec{w}_j) \neq y_{cf}$ do
 5:         Define linear constraint:
 6:     $(\vec{w}_j - \vec{w}_i)^\top \vec{x}_{cf} + b_{ij} + \epsilon \leq 0,$
 7:     where $b_{ij} = \frac{1}{2}\left(\|\vec{w}_i\|^2 - \|\vec{w}_j\|^2\right)$
 8:         Solve the QP:
 9:     $\min_{\vec{x}_{cf}} \ \frac{1}{2}\vec{x}_{cf}^\top \vec{x}_{cf} - \vec{x}_{orig}^\top \vec{x}_{cf}$
10:     subject to the constraint above
11:         $\vec{x}_{cf-candidate} \leftarrow$ solution of QP
12:         $y_{pred} \leftarrow$ GLVQ.predict$(\vec{x}_{cf-candidate})$
13:         if $y_{pred} = y_{cf}$ and $\|\vec{x}_{cf-candidate} - \vec{x}_{orig}\|^2 < \theta_{\min}$ then
14:             $\theta_{\min} \leftarrow \|\vec{x}_{cf-candidate} - \vec{x}_{orig}\|^2$
15:             $\vec{x}_{cf} \leftarrow \vec{x}_{cf-candidate}$
16:         end if
17:     end for
18: end for
19: return $\vec{x}_{cf}$
```

3.5.1 Convex Optimization

General form of convex optimization [9] problem is defined as:

$$\begin{aligned} \min_{\vec{x}} \quad & f_0(\vec{x}) \\ \text{subject to} \quad & g_i(\vec{x}) \leq 0, \quad i = 1, \dots, m \\ & h_j(\vec{x}) = 0, \quad j = 1, \dots, p \end{aligned} \tag{3.6}$$

This formulation seeks the vector $\vec{x}$ that minimizes the convex objective function $f_0(\vec{x})$, while satisfying a set of inequality constraints $g_i(\vec{x}) \leq 0 \quad i = 1, \cdots, m$ and equality constraint $h_j(\vec{x}) = 0, \quad j = 1, \dots, p$. The functions g_i are required to be convex, and the equality constraints h_j must be affine. Strict inequalities such as $g_i(\vec{x}) < 0$ from Eq. 3.6 are generally avoided in convex optimization problems for the following reasons:

- **One Feasible Reason:** The set $\{\vec{x} \mid g_i(\vec{x}) < 0\}$ is open, excluded its boundary. This complicates the use of numerical solvers, which rely on well-defined (closed) constraint sets to verify feasibility.
- **Optimization Boundary:** In many cases, the optimal solution lies exactly on the boundary, where $g_i(\vec{x}) = 0$. Excluding the boundary can eliminate the valid optimal points from the search space.

Quadratic programming:
A quadratic programming (QP) [9] problem is a special case of convex optimization where the objective function is quadratic and the constraints are linear.

The standard form of a quadratic program is :

$$\begin{aligned} \min_{\vec{x} \in \mathbb{R}^n} \quad & \frac{1}{2} \vec{x}^\top Q \vec{x} + \vec{c}^\top \vec{x} \\ \text{subject to} \quad & A\vec{x} \leq \vec{b}, \\ & E\vec{x} = \vec{d} \end{aligned} \tag{3.7}$$

Here:

$Q \in \mathbb{R}^{n \times n}$	is a symmetric matrix,
$Q \succeq 0$	(positive semidefinite) $\rightarrow$ ensures the objective is convex,
$\vec{c} \in \mathbb{R}^n$	is a vector,
$A \in \mathbb{R}^{m \times n}, \ \vec{b} \in \mathbb{R}^m$	inequality constraint matrix and vector,
$E \in \mathbb{R}^{p \times n}, \ \vec{d} \in \mathbb{R}^p$	equality constraint matrix and vector.

The term $\frac{1}{2}\vec{x}^\top Q \vec{x}$ defines the quadratic part of the objective function. Its convexity is guaranteed when the matrix Q is positive semidefinite, which is essential for ensuring that the problem has a global minimum.

3.5.2 Geometric Interpretation-Find Decision Boundary

We will determine the decision boundary from Figure 3.1, where it is equidistant from prototype $\vec{w}^{(1)}$ and $\vec{w}^{(2)}$.

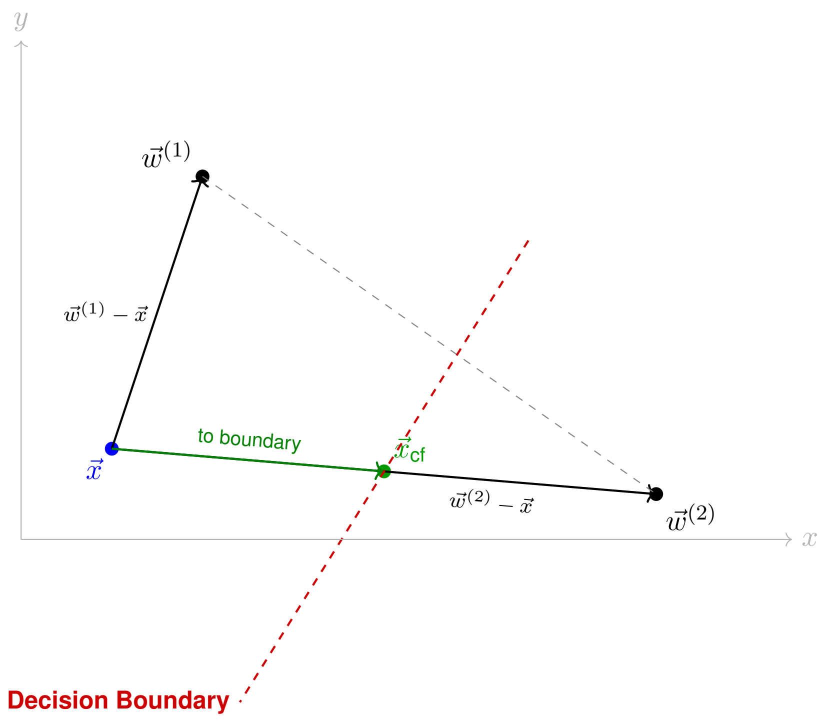


Figure 3.1: 2D Geometric Visualization with Decision Boundary

Here,

- $\vec{w}^{(1)}$- winning prototype for $\vec{x}$
- $\vec{w}^{(2)}$- winning prototype of target class for counterfactual $\vec{x}_{cf}$

One possible counterfactual for $\vec{x}$ is $\vec{x}_{cf}$ which fulfil the following equation

$$d^2(\vec{w}^{(1)}, \delta(\vec{w}^{(2)} - \vec{x})) = d^2(\vec{w}^{(2)}, \delta(\vec{w}^{(2)} - \vec{x})) \tag{3.8}$$

Goal: Find a solution for δ

Geometric Solution

- $\vec{x}$ — the original input
- $\vec{x}_{cf}$ is the counterfactual

The counterfactual $\vec{x}_{cf}$ is assumed to lie on the line connecting the original input $\vec{x}$ and the target prototype $\vec{w}^{(2)}$, its moved from original input $\vec{x}$ toward $\vec{w}^{(2)}$. A parametric vector equation [25] for generating a counterfactual in prototype-based model is defined as:

$$\vec{x}_{cf} = \vec{x} + \delta(\vec{w}^{(2)} - \vec{x}); \quad \delta \in \mathbb{R} \tag{3.9}$$

Let the direction vector be:

$$\vec{v} = \vec{w}^{(2)} - \vec{x}$$

Then the counterfactual can be rewritten as:

$$\vec{x}_{cf} = \vec{x} + \delta\vec{v}$$

Since the counterfactual lies on the decision boundary, it must be equidistant from both prototypes:

$$\left\|\vec{w}^{(1)} - \vec{x}_{cf}\right\|^2 = \left\|\vec{w}^{(2)} - \vec{x}_{cf}\right\|^2$$

Substituting $\vec{x}_{cf} = \vec{x} + \delta\vec{v}$ into the equation:

$$\left\|\vec{w}^{(1)} - (\vec{x} + \delta\vec{v})\right\|^2 = \left\|\vec{w}^{(2)} - (\vec{x} + \delta\vec{v})\right\|^2$$

$$\left\|\vec{w}^{(1)} - \vec{x} - \delta\vec{v}\right\|^2 = \left\|\vec{w}^{(2)} - \vec{x} - \delta\vec{v}\right\|^2$$

Now we expand both sides

$$\|\vec{w}^{(1)} - \vec{x}\|^2 - 2\delta(\vec{w}^{(1)} - \vec{x})^T\vec{v} + \delta^2\|\vec{v}\|^2 = \|\vec{w}^{(2)} - \vec{x}\|^2 - 2\delta(\vec{w}^{(2)} - \vec{x})^T\vec{v} + \delta^2\|\vec{v}\|^2$$

Canceling $\delta^2\|\vec{v}\|^2$ from both sides:

$$\|\vec{w}^{(1)} - \vec{x}\|^2 - \|\vec{w}^{(2)} - \vec{x}\|^2 = 2\delta\left((\vec{w}^{(1)} - \vec{w}^{(2)})^T\vec{v}\right)$$

Solving for δ, we get:

$$\delta = \frac{\|\vec{w}^{(1)} - \vec{x}\|^2 - \|\vec{w}^{(2)} - \vec{x}\|^2}{2(\vec{w}^{(1)} - \vec{w}^{(2)})^T(\vec{w}^{(2)} - \vec{x})} \tag{3.10}$$

This is the unique scalar δ such that the point $\vec{x}_{cf} = \vec{x} + \delta(\vec{w}^{(2)} - \vec{x})$ lies exactly on the decision boundary where the squared distances to the two prototypes $\vec{w}^{(1)}$ and $\vec{w}^{(2)}$ are equal.

3.5.3 Geometric Interpretation used in CEML

Now, we will compare this geometric interpretation with the one used in CEML, and identify their similarities and dissimilarities.

In Generalized Learning Vector Quantization with a squared Euclidean distance [5], the decision boundary between two prototypes $\vec{w}^{(i)}$ and $\vec{w}^{(j)}$ is determined as :

$$d(\vec{x}_{\text{cf}}, \vec{w}^{(i)}) = d(\vec{x}_{\text{cf}}, \vec{w}^{(j)})$$
$$\text{where} \quad d(\vec{x}_{\text{cf}}, \vec{w}) = (\vec{x}_{\text{cf}} - \vec{w})^\top (\vec{x}_{\text{cf}} - \vec{w})$$

Expanding the boundary condition:

$$\vec{x}_{\text{cf}}^{\top}\vec{x}_{\text{cf}} - 2\vec{w}^{(i)\top}\vec{x}_{\text{cf}} + \vec{w}^{(i)\top}\vec{w}^{(i)} = \vec{x}_{\text{cf}}^{\top}\vec{x}_{\text{cf}} - 2\vec{w}^{(j)\top}\vec{x}_{\text{cf}} + \vec{w}^{(j)\top}\vec{w}^{(j)}$$

Canceling $\vec{x}_{cf}^{\top}\vec{x}_{cf}$, we get:

$$-2\vec{w}^{(i)\top}\vec{x}_{\text{cf}} + \|\vec{w}^{(i)}\|^2 = -2\vec{w}^{(j)\top}\vec{x}_{\text{cf}} + \|\vec{w}^{(j)}\|^2$$

Move all terms to one side:

$$2(\vec{w}^{(j)} - \vec{w}^{(i)})^{\top}\vec{x}_{cf} + \|\vec{w}^{(i)}\|^2 - \|\vec{w}^{(j)}\|^2 = 0$$

Finally,

$$(\vec{w}^{(j)} - \vec{w}^{(i)})^{\top}\vec{x}_{cf} + \frac{1}{2}(\|\vec{w}^{(i)}\|^2 - \|\vec{w}^{(j)}\|^2) = 0$$

The geometric equation is :

$$(\vec{w}^{(j)} - \vec{w}^{(i)})^{\top}\vec{x}_{cf} + b_{ij} = 0 \tag{3.11}$$

Where:
bias

$$b_{ij} = \frac{1}{2}(\|\vec{w}^{(i)}\|^2 - \|\vec{w}^{(j)}\|^2) \tag{3.12}$$

The position of the hyperplane is offset [5] by b_{ij}. This equation defines a hyperplane in $\mathbb{R}^n$, containing all points $\vec{x}_{cf}$ that are equidistant from $\vec{w}^{(i)}$ and $\vec{w}^{(j)}$.

In GLVQ models using weighted Manhattan [5] [27] distance, the decision boundary between two prototypes $\vec{w}^{(i)}$ and $\vec{w}^{(j)}$ is defined by the set of points $\vec{x}_{cf}$ for which the weighted L_1 distances to both prototypes are equal:

$$\sum_k \alpha_k \left| x_{\text{cf},k} - w_k^{(i)} \right| = \sum_k \alpha_k \left| x_{\text{cf},k} - w_k^{(j)} \right|$$

This is the exact condition where the decision boundary lies where both prototypes are equally close in L_1 distance. It is a piecewise-linear and polyhedral. The surface is made up of several linear segments (or facets), depending on the relationship between each feature of $\vec{x}_{cf}$ and the corresponding prototype coordinate. The absolute value expands as:

$$\left| x_{\text{cf},k} - w_k^{(i)} \right| = \begin{cases} x_{\text{cf},k} - w_k^{(i)} & \text{if } x_{\text{cf},k} \geq w_k^{(i)} \\ -(x_{\text{cf},k} - w_k^{(i)}) & \text{if } x_{\text{cf},k} < w_k^{(i)} \end{cases}$$

This formulation results in a polyhedral decision boundary formed by the intersection of multiples half spaces. The specific configuration of the polyhedron depends on which regions of the feature space the input falls into.
To ensure a class flip during counterfactual generation, a small $\epsilon > 0$ is introduced, modifying the constraint as:

$$\sum_k \alpha_k \left| x_{\text{cf},k} - w_k^{(i)} \right| + \epsilon = \sum_k \alpha_k \left| x_{\text{cf},k} - w_k^{(j)} \right| \tag{3.13}$$

This defines a convex [4] feasible region for counterfactual- guaranteeing that $\vec{x}_{cf}$ is strictly closer to the target prototype $\vec{w}^{(i)}$ than to $\vec{w}^{(j)}$.

Similarities:
Despite using different distance functions, the squared Euclidean, weighted Manhattan, and parametric δ methods 3.10 exhibit several key similarities. All aim to produce counterfactuals that satisfy distance equality conditions with respect to two prototypes, thereby ensuring a class change at minimal cost. The approaches we have discuss in the Eq. 3.10, Eq. 3.12 and Eq. 3.13, the counterfactual satisfies the condition of equal distance to two prototypes.

$$d\left(\vec{x}_{\mathrm{cf}}, \vec{w}^{(i)}\right) = d\left(\vec{x}_{\mathrm{cf}}, \vec{w}^{(j)}\right)$$

- The counterfactual lies on the decision boundary between two classes.
- All conditions define a convex feasible boundary or region. The parametric point δ lies on a convex boundary (hyperplane). Equation 3.12 provides a linear hyperplane as the boundary, while the weighted Manhattan distance (L_1) defines a convex polyhedral boundary.
- Each method defines shapes where all points are equally far from the original input. For Euclidean distance, these shapes are spheres; for L_1 distance, they are polyhedra like diamonds (in two dimensions); and in the parametric δ case, they are spherical shapes considered only along a line towards the target prototype.

Dissimilarities:
Despite sharing the common goal of generating counterfactuals on the decision boundary between classes, the squared Euclidean, weighted Manhattan, and parametric δ methods 3.10 exhibit distinct characteristics in terms of their mathematical expressions, the geometry of their decision boundaries, and the nature of their solutions. The following outlines the key differences among these approaches.

The shape of the decision boundary is quite different for the squared Euclidean, weighted Manhattan, and parametric δ methods. For squared Euclidean distance, the decision boundary is a flat hyperplane, given by the equation

$$(\vec{w}^{(j)} - \vec{w}^{(i)})^\top \vec{x}_{\mathrm{cf}} + b_{ij} = 0,$$

where

$$b_{ij} = \frac{1}{2}\left(\|\vec{w}^{(i)}\|^2 - \|\vec{w}^{(j)}\|^2\right).$$

This hyperplane separates points that are equally far (in squared Euclidean terms) from the two prototypes.

With weighted Manhattan distance, the decision boundary is made up of flat pieces that together form a polyhedral shape. The condition for the boundary is

$$\sum_k \alpha_k \left|x_{\mathrm{cf},k} - w_k^{(i)}\right| = \sum_k \alpha_k \left|x_{\mathrm{cf},k} - w_k^{(j)}\right|,$$

where α_k are feature weights. The pieces of the boundary change depending on whether the coordinates of $\vec{x}_{\mathrm{cf}}$ are greater than or less than the prototype values, due to the absolute value terms in the L_1 distance.

The parametric δ method 3.10 is different because it does not form a surface at all. Instead, it looks for one specific point on the line between the original input $\vec{x}$ and the target prototype $\vec{w}^{(2)}$. This point is given by

$$\vec{x}_{\text{cf}} = \vec{x} + \delta(\vec{w}^{(2)} - \vec{x}),$$

where δ is chosen so that the squared distances to the two prototypes are equal:

$$\|\vec{w}^{(1)} - \vec{x}_{\text{cf}}\|^2 = \|\vec{w}^{(2)} - \vec{x}_{\text{cf}}\|^2.$$

This method finds a unique counterfactual point that lies exactly on the decision boundary along that line.

3.6 Plausibility

Counterfactual explanations, as introduced in 3.1, may fail to produce realistic examples. In many cases, the generated counterfactuals are similar to adversarial examples [27] —they involve very small changes that are hard to interpret and may not make sense in the real world. To be practically useful, counterfactuals must not only achieve a class change but also remain plausible— meaning they should resemble data that could have realistically occurred.

Density-Based Plausibility Constraint
In this thesis, we explore a density-based method that helps generate plausible counterfactuals by making sure they fall in high-density regions of the data. This method also generalizes to contrastive explanations [7], which aim to answer the question: why a certain prediction y was made instead of an alternative y'. It highlights the minimal and most relevant changes to the input $\vec{x}$ that would have resulted in a different output, thus answering the question: *"Why outcome y rather than y'?"* .

To evaluate the plausibility of a counterfactual explanation, we follow the method described by [4]. Let $\vec{x}_{cf}$ be the counterfactual point for a given input $\vec{x}_{orig}$. We want to find a counterfactual that changes the model's prediction to a desired class y_{cf}, while keeping the change as small as possible. This can be written as the following optimization problem:

$$\min_{\vec{x}_{cf} \in \mathbb{R}^d} \|\vec{x}_{cf} - \vec{x}_{orig}\|_2^2$$

subject to the condition:

$$g(\vec{x}_{cf}) = y_{cf}$$

Here, $g(\cdot)$ denotes the model's prediction function. The objective ensures the proximity while the constraint ensures correct classification into the target class.

However, not all counterfactuals are realistic. To ensure plausibility, we impose a density constraint requiring that the counterfactual lies in a region of high probability under the target class distribution:

$$\hat{p}_{y_{cf}}(\vec{x}_{cf}) \geq \rho$$

In this equation, $\hat{p}_{y_{cf}}(\cdot)$ is a density function that measures how typical a point is for the target class. The threshold ρ controls how strict this requirement is. If a counterfactual satisfies the prediction constraint but fails the density condition:

$$\hat{p}_{y_{cf}}(\vec{x}_{cf}) < \rho$$

It is considered implausible, even if it is close to the original and has the correct prediction. This means it may not represent a realistic data point.

The choice of ρ critically influences the trade-off between plausibility and proximity:

- A higher ρ ensures the counterfactual lies in a more realistic region, but this makes the search harder.
- A lower ρ makes it easier to find a counterfactual, but it may not be very realistic.

To choose ρ in practice, we look at the density values of the training data from the target class and pick a certain percentile. For example:

$$\rho = \text{Percentile}_q \left(\left\{ \hat{p}_{y_{cf}}(\vec{x}) \mid \vec{x} \in \mathcal{D}_{y_{cf}} \right\} \right), \quad q \in [0, 100]$$

where $\mathcal{D}_{y_{cf}}$ is the set of all training examples from the target class. This percentile-based threshold allows flexible control over the realism requirement for counterfactuals.

Definition 3.1 (ρ-plausible Counterfactual[2])
Let $g : X \to Y$ be a classifier. A counterfactual explanation $(\vec{x}_{cf}, y_{cf})$ for an input $\vec{x} \in X$ as ρ-plausible if it meets the following condition.

$$\vec{x}_{cf} = \arg\min_{\vec{x}_{cf}} \theta(\vec{x}_{cf}, \vec{x}_{orig}) \quad \text{s.t.} \quad g(\vec{x}_{cf}) = y_{\text{cf}} \wedge \hat{p}(\vec{x}_{cf}, y_{cf}) \geq \rho \tag{3.14}$$

Here, $\rho > 0$ denotes a minimum density at which we consider a sample to be plausible.

Density Estimator [23]: A density estimate is an approximation of the probability density function (PDF) that describes how data is distributed. The density estimate indicates:

- Where the data points are concentrated (high density regions) and
- Where data points are sparse (low-density regions)

We consider two prominent estimators:
Kernel Density Estimator (KDE)[3]:
A kernel density estimator (KDE) is a non-parametric method that estimates the density of a data point based on its similarity to neighbouring training samples:

$$\hat{p}_{\mathrm{KDE}}(\vec{x}) = \sum_{k} \alpha_k \cdot k(\vec{x}, \vec{x}_k) \tag{3.15}$$

Here,

- $k(\vec{x}, \vec{x}_k)$ is a kernel function measuring similarity between $\vec{x}$ and $\vec{x}_k$.
- α_k are weights - often $\alpha_k = \frac{1}{N}$; normalized weights.

For 2D Gaussian kernels.

$$k(\vec{x}, \vec{x}_k) = \frac{1}{2\pi h^2} \exp\left(-\frac{\|\vec{x} - \vec{x}_k\|^2}{2h^2} \right)$$

Here, $\vec{x}$ is the point at which the density is being estimated, while $\vec{x}_k$ is the k-th training sample, serving as the center of the kernel. The term $\|\vec{x} - \vec{x}_k\|^2$ denotes the squared Euclidean distance between $\vec{x}$ and $\vec{x}_k$. The bandwidth parameter h controls the smoothness of the kernel; smaller values of h produce sharper estimates, whereas larger values result in a smoother approximation.

Gaussian Mixture Model (GMM) [3]:

A Gaussian Mixture Model (GMM) estimates the probability density of a dataset as a weighted sum of multiple multivariate Gaussian distributions. The density function of a GMM is given by:

$$\hat{p}_{\text{GMM}}(\vec{x}) = \sum_{k=1}^{m} \pi_k \cdot \mathcal{N}(\vec{x} \mid \vec{\xi}_k, \Sigma_k)$$

where:

- π_k is the *mixture weight* (prior probability) of the k-th Gaussian component, satisfying $\sum_{k=1}^{m} \pi_k = 1,$
- $\vec{\xi}_k$ is the *mean vector* (location of peak density) of the k-th component,
- Σ_k is the *covariance matrix* defining the shape, orientation, and scale of the component.

Each Gaussian component represents a subpopulation within the data, allowing GMMs to model complex, multimodal distributions. In comparison to non-parametric estimators such as KDE, GMMs require fewer parameters and are generally more compact and computationally efficient, particularly in high-dimensional settings.

Trade-off Between Proximity and Plausibility in Counterfactual Explanations:
In figure 3.2 illustrates the fundamental trade-off [11] in counterfactual generation between *proximity* and *plausibility*.

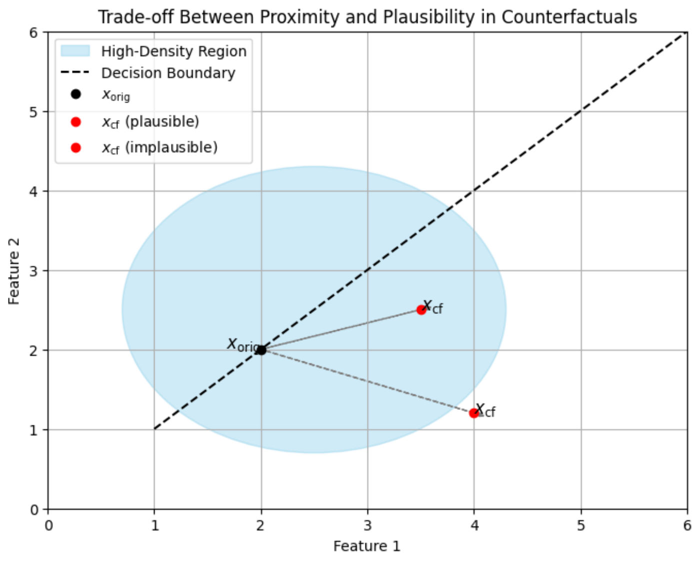


Figure 3.2: Example of plausibility proximity trade-off

The black point ($\vec{x}_{orig}$) represents the original data input. The dashed black line is the decision boundary between two classes, and the light blue area marks the high-density region of the target class — where real data points are likely to exist.

Two red points illustrate counterfactuals:

- The first red point ($\vec{x}_{cf}$) is **plausible**. It lies within the high-density region but is slightly farther from the original input.
- The second red point is **implausible**. It is closer to $\vec{x}_{orig}$ but lies outside the dense region, making it less realistic.

This example highlights that counterfactual generation must balance:
- **Proximity**: The counterfactual should not change too many features.
- **Plausibility**: The counterfactual should resemble a realistic input from the target class.

To balance this trade-off, we enforce the density constraint:

$$\hat{p}_{y_{cf}}(\vec{x}_{cf}) \geq \rho$$

This guarantees that $\vec{x}_{cf}$ lies in a region where examples of the target class typically occur. However, this often increases the distance from the original input, making it harder to find a nearby counterfactual.

4 Experiments and Outcome

We will review the experiments conducted and their results in this chapter. From this, we will identify the counterfactuals using the GLVQ method. The experiments were performed on three datasets: Circles [20], Iris [12], and a subset of the MNIST[17] digits.
For implementation, we have considered the train_test_split method to split the dataset into training and testing, where we reserved 20% for testing in circles and Iris datasets. We will investigate counterfactuals in this dataset and explore how they are modified to implement GLVQ and CEML methods.

4.0.1 Toolbox

The implementation of GLVQ and counterfactuals in this section used the following Python tools:

- Python version `3.9.6`
- `scikit_lvq`
- `matplotlib`
- `CEML` (Counterfactual Explanations for Machine Learning)

4.0.2 Circle Dataset

The make_circles dataset is a synthetic binary classification dataset in which one class of points forms an inner circle surrounded by a second class forming an outer ring. For this experiment, $1,000$ samples were generated with a noise level of 0.1 and a scale factor of 0.5 to prevent overfitting and introduce slight variability.

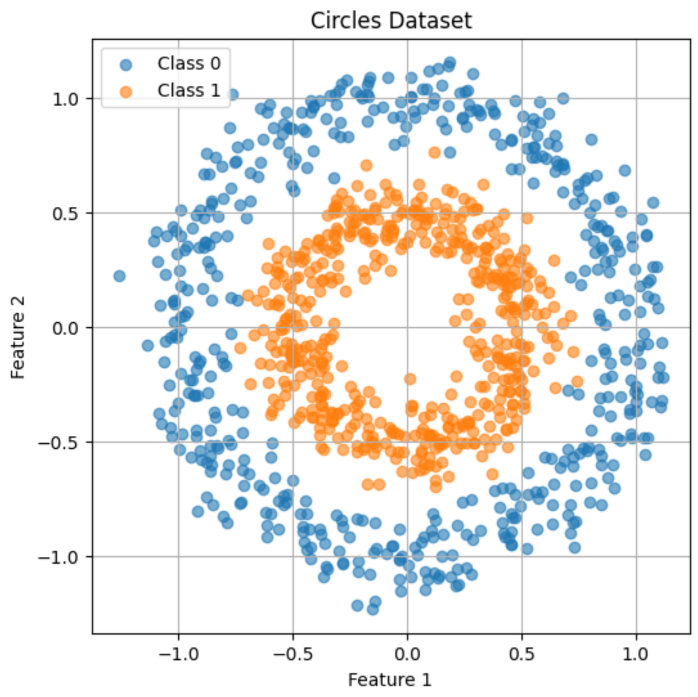


Figure 4.1: Circles Dataset

GLVQ with Circles Data:

In our GLVQ model with circle data, we used eight prototypes per class to cluster the dataset in Figure 4.2 and adjust the prototype positions to minimize a margin-based loss function. We have

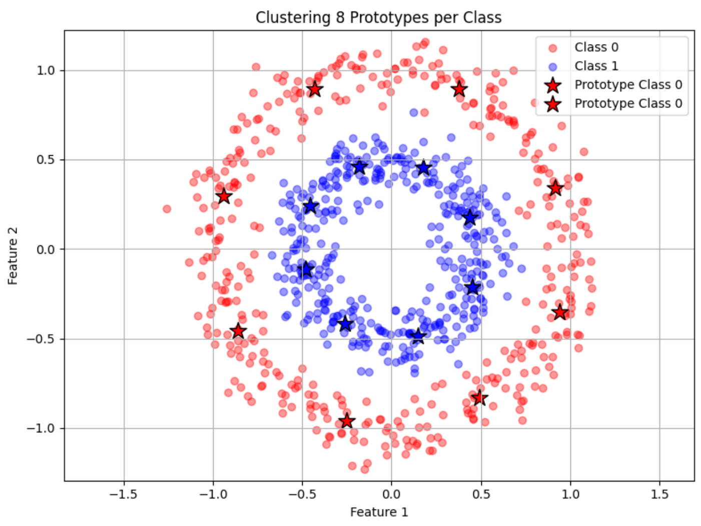


Figure 4.2: Circles Data Clustering

used the Euclidean distance to calculate the dissimilarity between the data points and the prototypes. We have also calculated the value of the cost function based on the margin maximization principle and updated it by minimizing the error during the training. Finally, we achieved an accuracy of 96% for the model using this 2D toy data.

To further evaluate the classification performance of the GLVQ model, we also calculated the ROC(Receiver Operating Characteristic) AUC(Area under the curve) score. We visualized the corresponding curve to illustrate the trade-off between the actual positive rate and the false positive rate at various threshold levels. In Figure 4.3, the area under the curve (AUC) is 98%, indicating that the

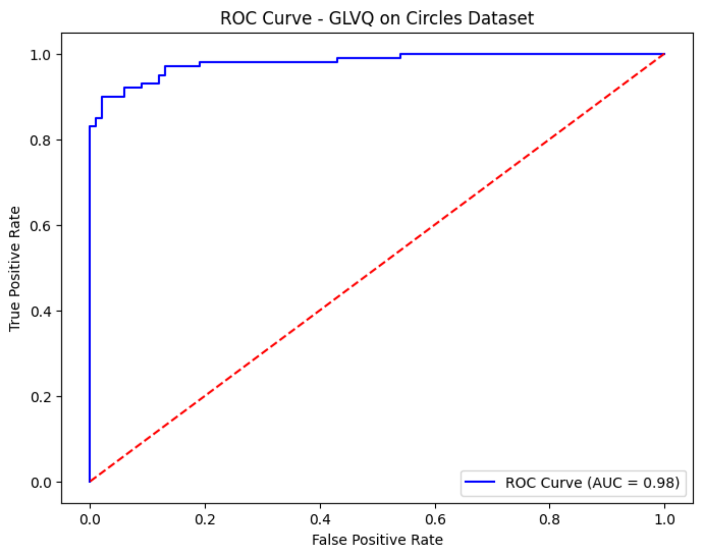


Figure 4.3: Trade off between actual positive and false positive rate for Circles dataset(GLVQ Model)

model exposes an excellent ability to distinguish between the two classes. The ROC curve remains in the upper left corner, which signifies a high actual positive rate.

Counterfactual Generation Via CEML:

To evaluate model interpretability, we generated counterfactual explanations for 50 randomly selected test samples using the Counterfactual Explanations for Machine Learningm framework [1], applied on top of the trained GLVQ model. During the generation of counterfactuals, we compute distances between data and prototypes using the generalized euclidean distance L_2. For regularization, we employed squared euclidean distance (squared L_2 norm). This allowed us to investigate how different combinations of distance and regularization influence the proximity, sparsity, and plausibility of the resulting counterfactuals. We used a range of regularization strengths, denoted as C={0.1, 1.0, 10, 100, 1000} .This values were used to encourage minimal changes in the generated counterfactuals. Lower values of C allowed more flexibility in altering input features, while higher values enforced stronger proximity to the original input. The optimizer was set to auto, allowing CEML to automatically select the most suitable solver based on the problem formulation and parameter configuration.

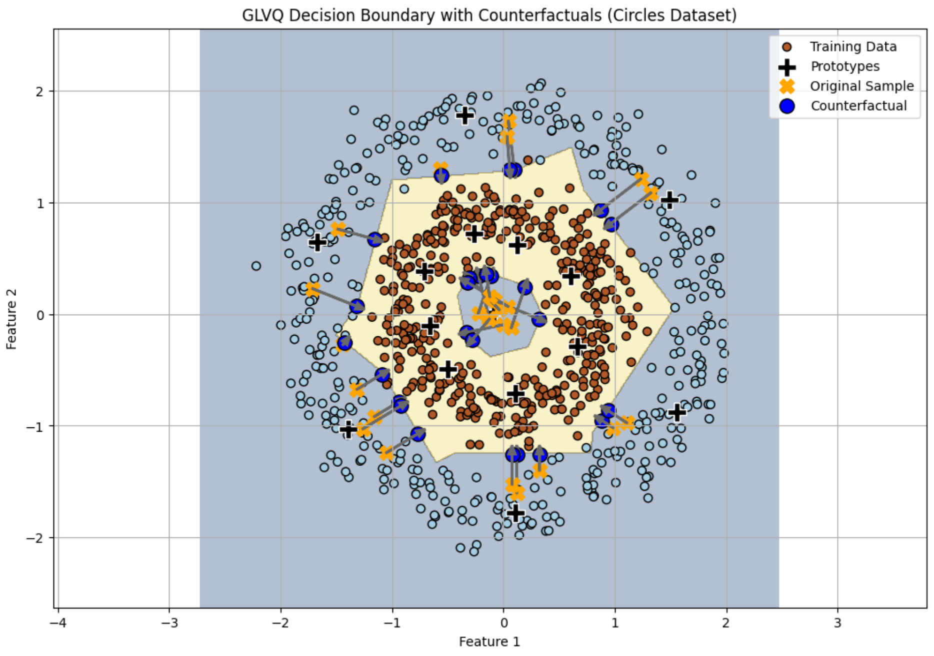


Figure 4.4: GLVQ decision boundary and counterfactual on the circles dataset.

Figure 4.4 highlights the interpretability of the GLVQ model by showing how decision boundaries are formed based on prototype positions. This provides a clear geometric understanding of classification behaviour.

The counterfactual plots show minimal and targeted changes that shift the original sample across the decision boundary. This also confirms that the optimization process produces counterfactuals that are both class-changing but interpretable in 2D space.

The arrows from the original inputs to the counterfactuals visually indicate the direction and magnitude of change. Shorter arrows correspond to less significant feature perturbations, suggesting more plausible and minimal counterfactuals. Although the GLVQ model successfully separates the two classes, the decision boundary appears axis-aligned. To determine the cause, we have tested various models with different parameters and identified the possible source of the issue.

Axis Parallel:

Analysis of the counterfactual trajectories revealed that most perturbation paths were aligned with the coordinate axes. As seen in Figure 4.4, the arrows often point horizontally or vertically. This axis-parallel behaviour arises primarily from the use of L_1 regularization or L_1 distance, both of which promote sparsity and lead to changes in one feature at a time. While this can enhance interpretability, it limits flexibility in complex geometries.

Despite testing all four metric-regularization combinations, axis-aligned paths persisted due to the circular structure of the data and the optimization constraints.

Discussion of Results: We obtained 29 samples for which the counterfactual new input represents a feasible modification of the original sample that crosses the decision boundary and achieves the intended class change. In contrast, for 21 samples, the original prediction was already strongly classified according to the model's decision criteria. As a result, no alternative input could be generated that altered the prediction to the target class within the constraints of the counterfactual search.

The norm of the applied changes across successful counterfactuals ranged from approximately 0.02 to 0.46. The mean norm was approximately 0.27, indicating that, on average, moderate feature adjustments were required to achieve the desired class change. This reflects the varying proximity of the original samples to the decision boundary: some were close and required minimal changes, while others were further away and needed larger modifications to alter the prediction.

Sample Index	Original Prediction	CF Prediction	Status
1	1	–	No CF found
15	1	0	Successful
16	1	0	Successful
22	1	0	Successful
35	1	0	Successful
36	1	–	No CF found
37	1	0	Successful

Table 4.1: Mixed counterfactual generation results

Counterfactual Generation Via GLVQ-OptCF

We also evaluated counterfactual generation using a custom optimization method (**GLVQ-OptCF**), where counterfactuals are computed by solving a constrained optimization problem directly based on the GLVQ model. The objective was to achieve a class change with minimal perturbation, measured using the squared Euclidean norm ($\|\Delta\|_2^2$).

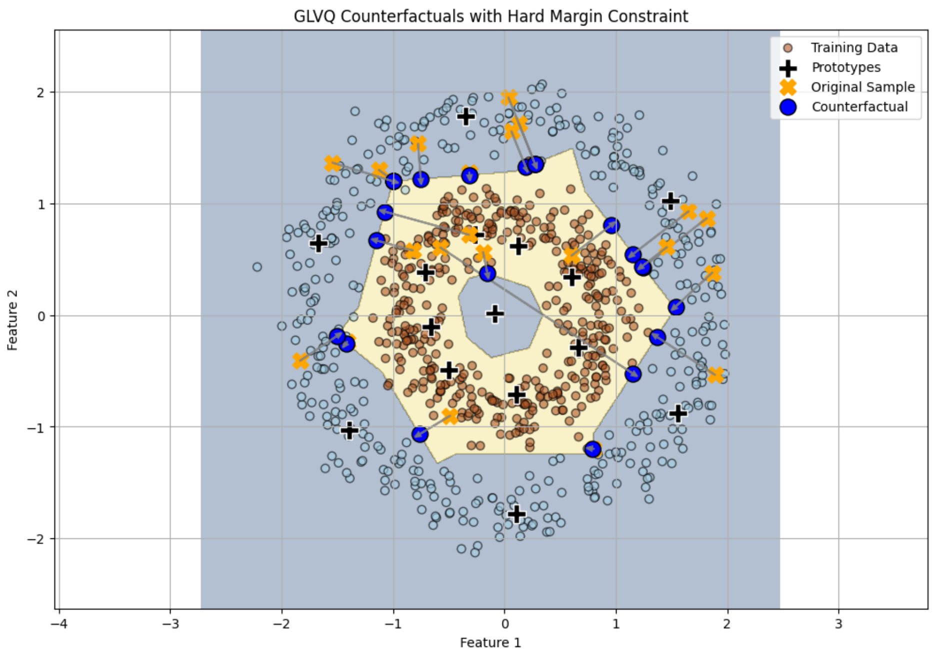


Figure 4.5: Counterfactual for Circles GLVQ-OptCF Model

Out of 50 instances:

- 22 Counterfactuals successfully resulted in successful class flips.
- 28 failed to cross the decision boundary.

Unlike CEML, which tends to generate sparse, axis-aligned changes, the GLVQ-OptCF model 1 model often yielded diagonal paths, reflecting balanced feature adjustments across both dimensions. Of the 22 successful cases:

- 14 instances successfully flipped from class 0 to 1.
- 8 instances flipped from 1 to 0

The perturbation magnitudes varied widely:

- The smallest successful perturbation was $\|\Delta\|_2^2 = 0.0145$, indicating a minimal and highly effective change.
- The largest successful perturbation was $\|\Delta\|_2^2 = 2.0618$, suggesting a more distant counterfactual was needed to achieve the class flip.

Comparison of GLVQ-OptCF and CEML Counterfactual Explanations

We compared counterfactual explanations generated by two different approaches: a custom method GLVQ-OptCF 1 counterfactual and the established CEML framework. Both models were evaluated on a 2D binary classification task using the make_circles dataset.

Quantitative Comparison

This comparison table 4.2 shows that CEML has a slightly higher success rate, particularly when restricted to class 1 $\rightarrow$ 0 transitions. GLVQ-OptCF can perform bidirectional flips (both 0 $\rightarrow$ 1 and 1 $\rightarrow$ 0), but with lower overall success. This indicates a trade-off between flexibility and performance stability.

Metric	GLVQ-OptCF	CEML Model
Total Counterfactuals Evaluated	50	50
Successful Class Flips	21	26
Failed Class Flips	29	24
Successful Flips 0 $\rightarrow$ 1	14	N/A (only 1 $\rightarrow$ 0)
Successful Flips 1 $\rightarrow$ 0	7	26

Table 4.2: Quantitative Comparison Between GLVQ-OptCF and CEML

Perturbation Magnitude Comparison

The results show in tabel 4.3 that CEML produces counterfactuals with lower average perturbation (mean = 0.3071) compared to GLVQ-OptCF (mean = 0.4755). Furthermore, the maximum perturbation observed in GLVQ-OptCF (2.0618) is substantially higher than that in CEML (0.4566), suggesting that GLVQ-based optimization occasionally requires larger changes to flip the prediction. Although GLVQ-OptCF achieves a slightly lower minimum perturbation (0.0145), the overall trend indicates that CEML tends to produce more compact and efficient counterfactuals.

Metric	GLVQ-OptCF	CEML Model
Min	0.0145	0.0229
Max	2.0618	0.4566
Mean	0.4755	0.3071

Table 4.3: Comparison of counterfactual perturbation norm between GLVQ-OptCF and CEML

Conclusion:

The comparison reveals a trade-off between interpretability and flexibility in counterfactual generation. The CEML method tends to produce sparse, axis-aligned counterfactuals by modifying a single feature at a time. While this increases interpretability, it limits the model's ability to navigate complex decision boundaries such as those in the circles dataset.

Conversely, GLVQ-OptCF provides more flexible, multi-feature perturbations, often in diagonal directions, which better adapt to the curved boundary geometry. This allows exploration of more complex regions in the input space but may result in less interpretable explanations.

4.0.3 Iris Dataset

The Iris dataset is a well-known benchmark in pattern recognition, introduced by Ronald A. Fisher in 1936. It contains 150 samples divided equally among three species: Setosa (class 0), Versicolor (class 1), and Virginica (class 2). Each sample is described by four continuous features: sepal length, sepal width, petal length, and petal width, all measured in centimeters. The dataset is often used for classification, clustering, and interpretability benchmarking.

GLVQ Model Evaluation

We trained a Generalized Learning Vector Quantization model on this Iris dataset using two prototypes for each class, as shown in Figure 4.6. The generalized Euclidean distance metric was used to compute dissimilarities and the model was trained to minimize a margin-based cost function. The classification accuracy on the test set is 93%. To better assess the classification performance of the GLVQ model on the Iris dataset, we computed the one-vs-rest ROC AUC scores for each

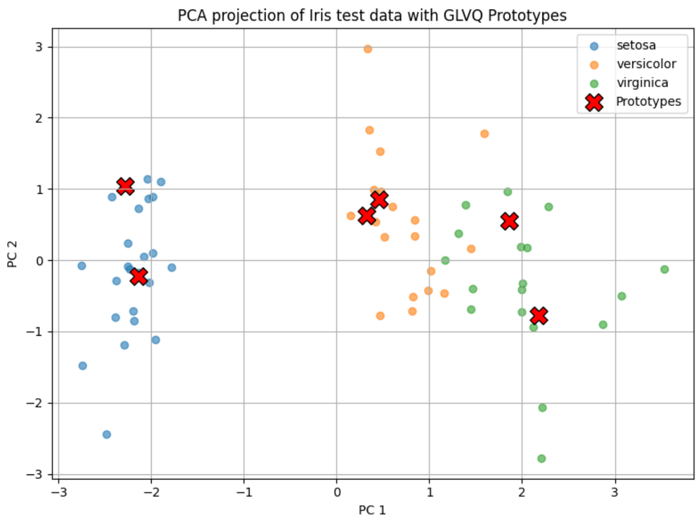


Figure 4.6: Iris test dataset clustering with 2 prototypes per class.

class individually. The corresponding ROC curves are visualized in Figure 4.7, which illustrate the trade-off between the true positive rate and the false positive rate across different classification thresholds. The model achieved an AUC of 1.00 for the Setosa class and 0.95 for both the Versicolor and Virginica classes. These high AUC values indicate that the GLVQ model is highly effective at distinguishing between the three species, with perfect separability for Setosa and strong performance for the other two, which are known to be more overlapping. Overall, this result confirms the model's strong discriminative ability in multi-class settings.

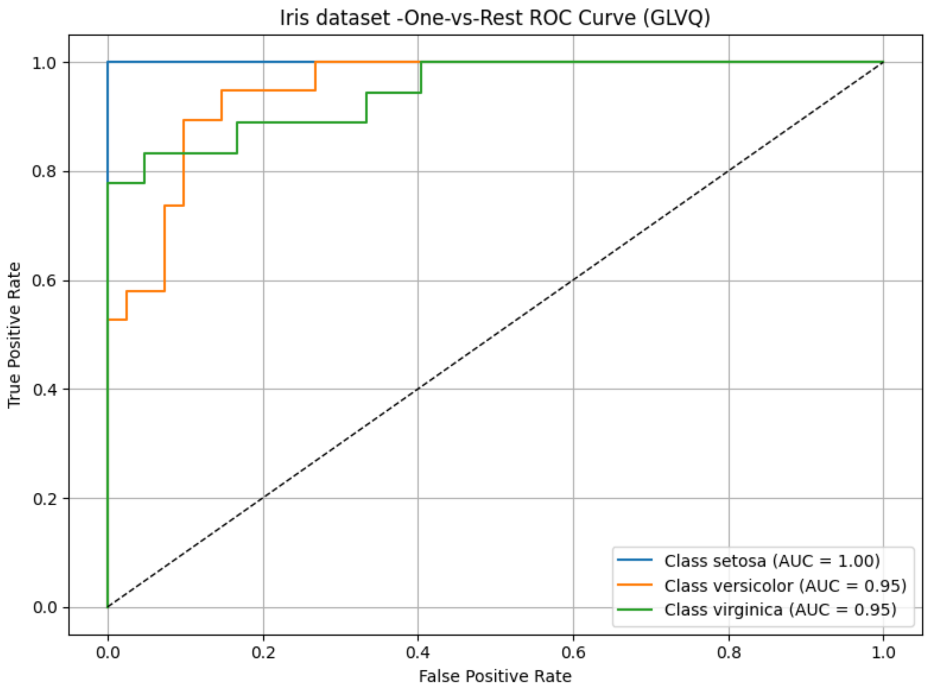


Figure 4.7: ROC AUC in Iris Dataset

Counterfactual Generation Via CEML:

We generated counterfactuals using the CEML framework applied to the trained GLVQ model. A total of 60 samples selected from the test set for evaluation. Class 0(Setosa) was fixed as target class. The regularization strength C was chosen from the set $\{0.1, 1.0, 10.0, 100.0, 1000.0\}$. To penalize deviations between the original input and the counterfactual, we used the generalized Euclidean distance (L_2) as the proximity measure. For optimization, we set the optimizer parameter to `auto`, allowing CEML to automatically select the most suitable solver.

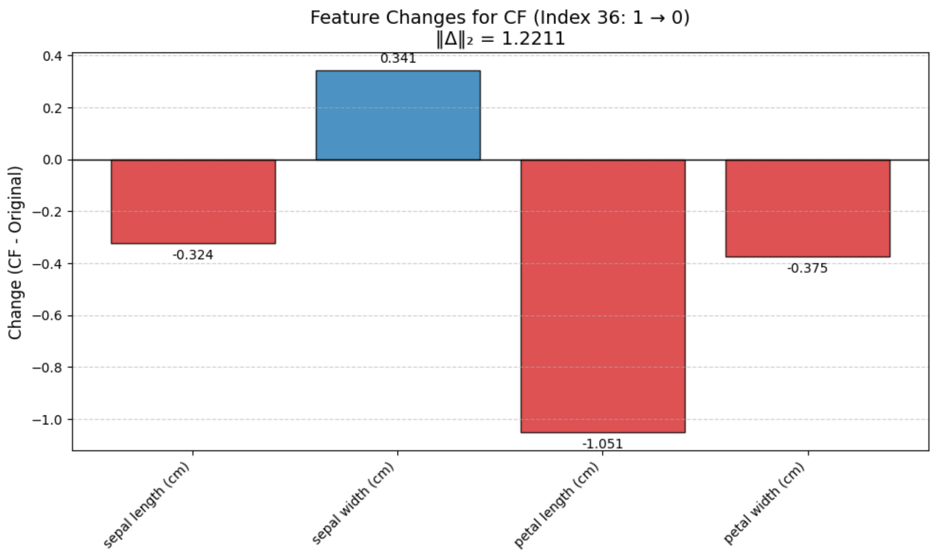


Figure 4.8: Counterfactuals in Iris Dataset

In one illustrative example Figure 4.8, a successful class flip from class 1 to class 0 was achieved primarily by reducing the petal length with moderate changes in sepal length and petal length. The total perturbation magnitude was 1.22, indicating a moderate perturbation. This confirms that petal-related features are most influential for classification into Setosa, aligning with biological expectations.

Result Summary:

We successfully generated 23 counterfactuals that changed the model's prediction to the target class 0(Setosa). However, for 28 samples, no counterfactual was found. This means either the model was very confident in its original prediction, or the optimization could not find a solution under the current settings.

The minimum perturbation observed was 1.22, while the maximum reached 3.90. On average, the perturbation magnitude was approximately 2.09, indicating that moderate to substantial feature changes were required to flip the model's prediction to the target class (Setosa).

Sample Index	Original Prediction	CF Prediction	Status
0	1	0	Successful
1	1	–	No CF found
2	2	0	Successful
3	2	–	No CF found
4	1	0	Successful
5	1	–	No CF found
6	1	–	No CF found

Table 4.4: Summary of counterfactual results for the first 10 samples in Iris test data.

Counterfactual Via GLVQ-OptCF

We then applied our custom method, GLVQ-OptCF, to generate counterfactuals for the same 60 test samples, targeting the same class (Setosa). The squared Euclidean distance was used as the proximity metric. The regularization strength was fixed at $C = 0.1$, and counterfactuals were generated by solving a constrained minimization problem based on prototype-based distance margins.

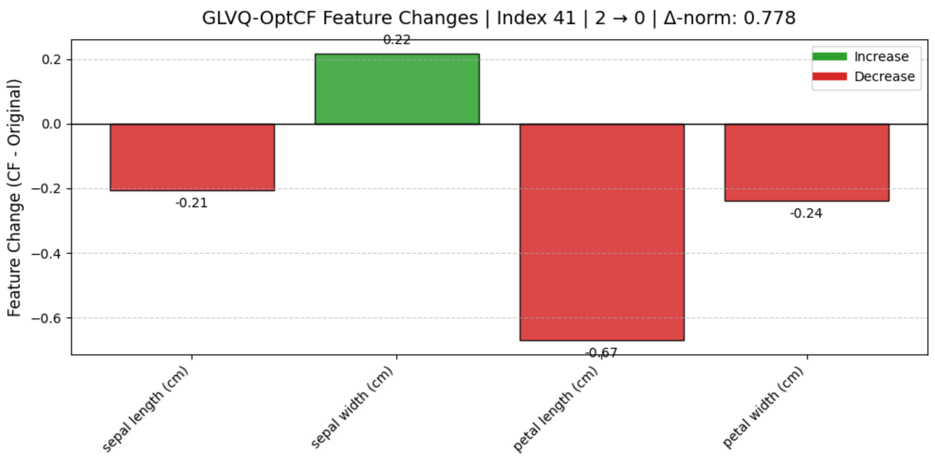


Figure 4.9: GLVQ-OptCF Model Counterfactual for Iris data

The most significant change occurred in *petal length*, suggesting that this feature (see Figure 4.9) was the primary contributor to the class flip from class 2 (*Virginica*) to class 0 (*Setosa*). Minor adjustments in *sepal width* and *petal width* also contributed. The counterfactual remained relatively close to the original input ($\|\Delta\|_2^2 = 0.778$), indicating high plausibility.

Overview:

The GLVQ-OptCF approach showed strong capability in producing counterfactuals targeting a fixed class—Setosa (class 0). Among the 60 evaluated samples, 15 were already classified as Setosa and required no modification, while 23 instances successfully flipped their class to the target. The average perturbation norm among these non-trivial flips was approximately 1.66, indicating moderate changes. Most adjustments involved petal-related features, which aligns with known class-discriminative attributes in the Iris dataset. The optimization procedure proved robust, with only two cases resulting in failure. Overall, the results demonstrate that GLVQ-OptCF can generate plausible and interpretable counterfactuals, effectively leveraging the geometric structure of prototype-based classification boundaries.

Comparison Between GLVQ-OptCF and CEML

We performed a comparative evaluation of both counterfactual generation methods:

Quantitative Comparison

As shown in Table 4.5, both GLVQ-OptCF and CEML performed similarly in terms of successful

Metric	GLVQ-OptCF	CEML
Total Samples Evaluated	60	60
Successful Class Flips	38	36
Failures / Optimization Failures	22	24

Table 4.5: Quantitative Comparison of Counterfactual Generation Iris dataset

class flips, with GLVQ-OptCF achieving 38 and CEML 36. However, GLVQ-OptCF encountered 22 optimization failures, while CEML showed 24 such issues, indicating more stable performance.

Perturbation Magnitude Comparison

Magnitude Metric	GLVQ-OptCF	CEML
Mean perturbation	1.66	1.77
Minimum perturbation (non-zero)	0.4103	0.7815
Maximum perturbation	3.2056	3.9027

Table 4.6: Comparison of Counterfactual Magnitude ($\|\Delta\|_2^2$ Norm)

Table 4.6 compares the size of perturbations. Both methods produced comparable average changes, though GLVQ-OptCF reached a lower minimum perturbation, suggesting potential for more efficient flips in some cases. On the other hand, CEML produced slightly larger maximum perturbations, reflecting a wider range of solution magnitudes.

Interpretation and Rashomon Effect

While both methods successfully generated plausible counterfactuals, the paths to the class change varied. The same prediction flip (e.g., to Setosa) could be achieved via different combinations of feature changes, particularly around petal length and width. This phenomenon illustrates the Rashomon Effect: the presence of multiple valid explanations or models that can achieve the same output. In counterfactual explainability, this means that diverse counterfactuals may all be valid, depending on the chosen distance metric, regularization strength, or optimization method.

Conclusion:

The experiments on the Iris dataset highlight that both CEML and GLVQ-OptCF can generate meaningful and interpretable counterfactual explanations. While CEML tended to produce counterfactuals with smaller perturbations, its performance was more dependent on careful tuning of the regularization parameter. In contrast, GLVQ-OptCF demonstrated greater geometric adaptability by directly incorporating the structure of prototypes and decision boundaries into the optimization process. This resulted in more consistent performance, even for samples located in more complex regions of the input space. Overall, the results reinforce the potential of prototype-based approaches for generating reliable and interpretable counterfactuals in multi-class classification tasks.

4.0.4 MNIST dataset

The MNIST(Modified National Institute of Standards and Technology) dataset consists of a training set containing $60{,}000$ images and a test set of $10{,}000$ images of handwritten digits, figure 4.10 shows a sample of this dataset. Each image has been size-normalized and centered in a fixed resolution of 28×28 pixels. In our work, we have selected a subset of the MNIST dataset containing the digits 3, 5, 7, and 9. The training set contains $22,164$ images, while the test set comprises $5,541$ images.

We trained a GLVQ model, utilizing four prototypes for each class, and with a maximum number of iterations to 1000. The trained model achieved a test accuracy of 89.6%.

Figure 4.10: MNIST[17] Dataset 28×28 pixels

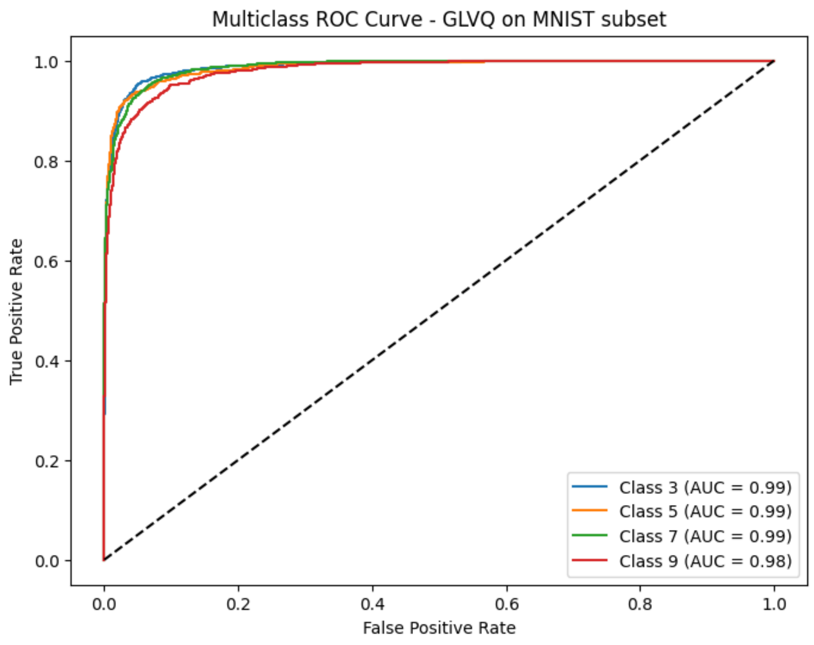


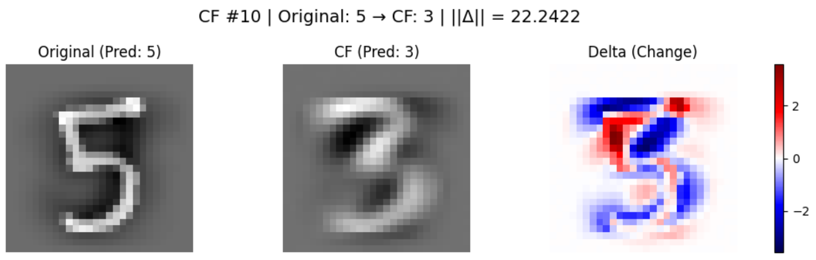
Figure 4.11: ROC AUC to visualize Multiclassification performance

To further evaluate the multi classification performance of the GLVQ model, we computed the ROC AUC scores in one-vs-rest for each class. The ROC curves figure in 4.11 demonstrate strong separability, with each curve approaching the top-left corner, the area under the curve is close to 1.0 for all classes, indicating excellent class separability.

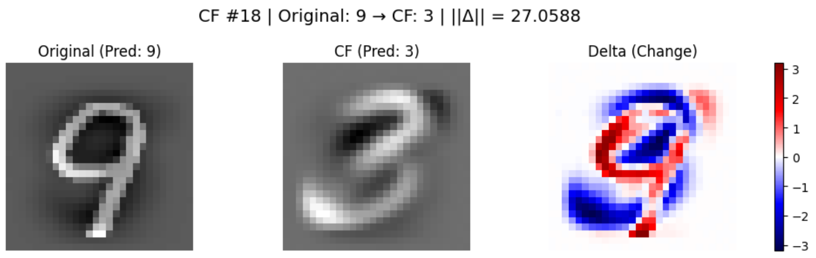
MNIST Counterfactual Generation using CEML

To evaluate the interpretability of the GLVQ model, we used the CEML framework to generate counterfactuals targeting digit 3 as the fixed class. In figure 4.12 shows two representative examples where the original predictions were 5 and 9, respectively, both successfully transformed into 3 via counterfactual generation.

In this experiment,we evaluated 100 test samples. The counterfactual generation was configured as follows: regularization strength of $\{0.1, 1.0, 10.0, 100.0, 1000.0\}$, set the optimizer to auto, and specified SLSQP as the solver in the optimizer arguments. We used the weighted Manhattan distance L_1 to compute distances between data points and prototypes, and selected the squared Euclidean



(a) CF index 10



(b) CF index 18

Figure 4.12: Counterfactual explanations for the MNIST subset dataset

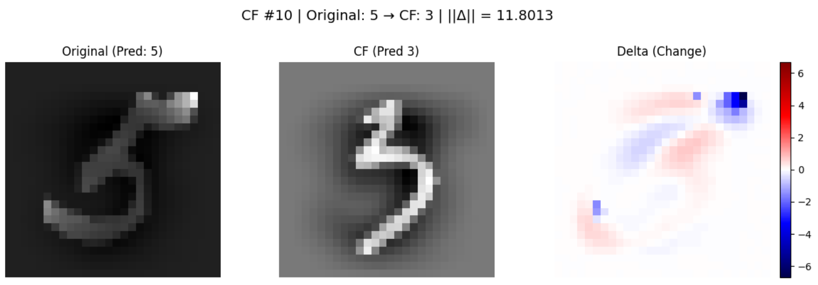
distance as the proximity measure. A total of 24 counterfactuals were successfully generated that flipped the prediction to class 3, while the remaining samples retained their original predictions. The output of several representative samples is shown in table 4.7. The perturbation magnitudes for the successfully generated counterfactuals using CEML ranged from 22.61 to 46.63, with most values falling between 25 and 35. These perturbations represent the Euclidean norm of the feature changes applied to the original inputs to achieve the desired class flip to digit 3. The variation in norm size reflects the differing proximities of the original samples to the decision boundary and indicates that substantial changes were often required to produce valid counterfactuals in the high-dimensional MNIST space.

Sample Index	Original Prediction	Status
0	7	No CF found
2	7	No CF found
3	7	Successful
11	9	Successful
26	7	Successful
49	5	Successful
74	5	Successful

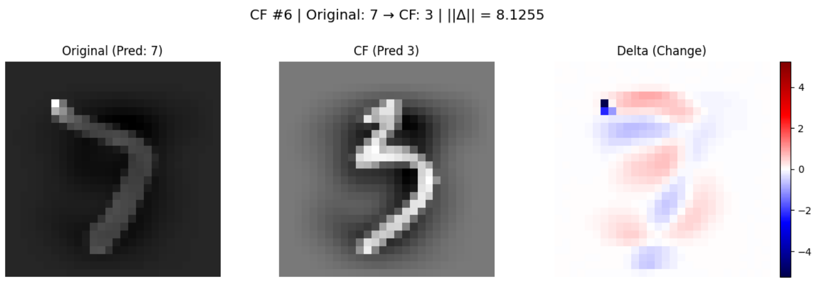
Table 4.7: Selected counterfactual results with CF Status

MNIST Counterfactual Generation using GLVQ-OptCF

We further applied our custom GLVQ-OptCF method to generate counterfactuals for the same 100 test samples, again targeting digit 3. The squared Euclidean distance was used both as the distance metric and the proximity measure.



(a) CF index 10



(b) CF index 5

Figure 4.13: Counterfactual explanations for the MNIST (GLVQ-OptCF Model)

Figure 4.13 illustrates two counterfactual examples were generated using the GLVQ-OptCF. Out of 100 test samples, 15 counterfactuals were successfully generated and transformed into the target class (digit 3), 33 samples already belonged to the target class 3, and the remaining 52 retained their original class predictions under the GLVQ model.
The perturbation magnitudes among the 15 successful counterfactuals ranged from 0.15 to 19.12, with a mean norm of approximately 9.28. This variability suggests that some samples were near the decision boundary and needed only minor changes, while others required substantial feature modifications to achieve a class change.

Comparison Between GLVQ-OptCF and CEML CF generation

Metric	CEML	GLVQ-OptCF
Total Samples Evaluated	100	100
Already Predicted as Class 3	–	33
Required CF Generation	100	67
Successful Counterfactuals Found	24	15
Failed to Generate CFs	76	52

Table 4.8: Quantitative Comparison of Counterfactual Generation on MNIST

Tables 4.8 and 4.9 summarize the performance of the CEML and GLVQ-OptCF frameworks in generating counterfactuals on a subset of the MNIST dataset.

Quantitatively, CEML produced a higher number of successful counterfactuals (24 out of 100 samples), whereas GLVQ-OptCF generated 15 successful counterfactuals, with an additional 33 samples already predicted as the target class and thus requiring no modification.

Metric	CEML	GLVQ-OptCF
Perturbation Norm Range	22.61 – 46.63	0.15 – 19.12
Mean Perturbation Norm	30.5	9.28
Distance Metric Used	Weighted L_1	Squared L_2
Feature Space	784-D pixels	784-D pixels

Table 4.9: Comparison of Perturbation Norms ($\|\Delta\|_2^2$) on MNIST

In terms of perturbation magnitude, a significant difference was observed. The counterfactuals produced by CEML had sqaured Euclidean norms ranging from 22.61 to 46.63, with a mean of approximately 30.5. In contrast, GLVQ-OptCF achieved counterfactual flips with much smaller changes, ranging from 0.15 to 19.12 and averaging around 9.28. This suggests that while CEML was more successful in generating counterfactuals under default conditions, the GLVQ-OptCF framework produced more localized and potentially more plausible explanations by leveraging the geometric structure of the prototype-based model.

Conclusion

The comparison between CEML and GLVQ-OptCF highlights fundamental trade-offs in counterfactual generation methods. CEML achieved a higher success rate in generating counterfactuals (24% vs. 15%) but required significantly larger perturbations, with a mean perturbation norm of approximately 30.5 and a range between 22.61 and 46.63. In contrast, GLVQ-OptCF generated fewer successful counterfactuals but with substantially smaller perturbations (mean 9.28; range 0.15–19.12), indicating more minimal and interpretable changes.

Additionally, GLVQ-OptCF benefits from leveraging the internal structure of the GLVQ model and produces solutions that are often better aligned with the learned decision boundaries. However, it may fail more often due to stricter optimization constraints. These findings suggest a trade-off between counterfactual success rate and perturbation minimality, and the choice of method should depend on whether interpretability or coverage is prioritized.

5 Conclusion and Future work

In this master's thesis, we explored how to generate counterfactual explanation using a prototype-based classification model called Generalized Learning Vector Quantization. Counterfactuals are valuable in machine learning because they offer insights into what minimal changes to an input could result in a different model prediction, thereby enhancing interpretability and trust.

We focused on a formulation based on the squared Euclidean distance, which allowed us to express the counterfactual generation problem as a convex optimization task, which is easier to solve and gives more predictable results. To avoid complexity when multiple prototypes exist per class, we selected one prototype from the target class and one from another class. This modification helped keep the problem linear and efficient.

To make sure the generated counterfactuals were not only correct but also realistic, we added checks using Kernel Density Estimation and Gaussian Mixture Models. These methods helped evaluate whether the counterfactuals lie in a region where real data exists.

Our proposed method, GLVQ-OptCF, was evaluated on multiple datasets, including synthetic and real-world examples. The results demonstrate that our approach successfully generates actionable and realistic counterfactuals with minimal perturbations. When compared to an existing baseline, Counterfactual Explanations for Machine Learning, our method showed strong performance in terms of accuracy, simplicity, and plausibility.

While our approach works well, it does have some limitations—for example, fixing prototype pairs might reduce flexibility in some cases, and density estimation may be less reliable in very high-dimensional data. In the future, we hope to explore ways to make prototype selection more adaptive and use more advanced models to measure plausibility.

In conclusion, this thesis presents a structured and interpretable approach for counterfactual generation within the GLVQ framework, contributing to the broader goal of making machine learning models more transparent and trustworthy.

Bibliography

[1] André Artelt. *CEML: Counterfactual Explanations for Machine Learning*. https://github.com/andreArtelt/ceml. Accessed: 2025-06-01. 2019.

[2] André Artelt. "Contrasting Explanations in Machine Learning. Efficiency, Robustness & Applications". PhD thesis. Dissertation, Bielefeld, Universität Bielefeld, 2023, 2024.

[3] André Artelt and Barbara Hammer. "Convex Density Constraints for Computing Plausible Counterfactual Explanations". In: *CoRR* abs/2002.04862 (2020). arXiv: 2002.04862. URL: https://arxiv.org/abs/2002.04862.

[4] André Artelt and Barbara Hammer. "Convex optimization for actionable\& plausible counterfactual explanations". In: *arXiv preprint arXiv:2105.07630* (2021).

[5] André Artelt and Barbara Hammer. "Efficient computation of counterfactual explanations of LVQ models". In: *arXiv preprint arXiv:1908.00735* (2019).

[6] André Artelt and Barbara Hammer. "On the computation of counterfactual explanations - A survey". In: *CoRR* abs/1911.07749 (2019). arXiv: 1911.07749. URL: http://arxiv.org/abs/1911.07749.

[7] André Artelt et al. "Contrastive Explanations for Explaining Model Adaptations". In: *CoRR* abs/2104.02459 (2021). arXiv: 2104.02459. URL: https://arxiv.org/abs/2104.02459.

[8] Michael Biehl, Barbara Hammer, and Thomas Villmann. "Prototype-based models in machine learning". In: *Wiley interdisciplinary reviews. Cognitive science* 7 (Jan. 2016). DOI: 10.1002/wcs.1378.

[9] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. USA: Cambridge University Press, 2004. ISBN: 0521833787.

[10] Leo Breiman. "Statistical modeling: The two cultures (with comments and a rejoinder by the author)". In: *Statistical science* 16.3 (2001), pp. 199–231.

[11] Susanne Dandl et al. "Countarfactuals–generating plausible model-agnostic counterfactual explanations with adversarial random forests". In: *World Conference on Explainable Artificial Intelligence*. Springer. 2024, pp. 85–107.

[12] Ronald A. Fisher. "The Use of Multiple Measurements in Taxonomic Problems". In: *Annals of Eugenics* 7.2 (1936), pp. 179–188.

[13] Barbara Hammer and Thomas Villmann. "Generalized relevance learning vector quantization". In: *Neural Networks* 15.8–9 (2002), pp. 1059–1068.

[14] Amir-Hossein Karimi et al. "Model-agnostic counterfactual explanations for consequential decisions". In: *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS)*. PMLR. 2020, pp. 895–905.

[15] Teuvo Kohonen. "Learning Vector Quantization (LVQ)". In: *Self-Organizing Maps*. Vol. 30. Springer Series in Information Sciences. Springer, 1995, pp. 175–189. DOI: 10.1007/978-3-642-97610-0_12.

[16] Teuvo Kohonen and Teuvo Kohonen. "Learning vector quantization". In: *Self-organizing maps* (2001), pp. 245–261.

[17] Yann LeCun, Corinna Cortes, and Christopher J.C. Burges. *The MNIST Database of Handwritten Digits*. http://yann.lecun.com/exdb/mnist/. Accessed: 2025-06-01. 1998.

[18] Christoph Molnar. *Interpretable Machine Learning*. 2nd ed. https://christophm.github.io/interpretable-ml-book/. Leanpub, 2022.

[19] David Nova and Pablo A Estévez. “A review of learning vector quantization classifiers”. In: *Neural Computing and Applications* 25 (2014), pp. 511–524.

[20] Fabian Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[21] Atsushi Sato and Keiji Yamada. “Generalized Learning Vector Quantization”. In: *Advances in Neural Information Processing Systems*. Ed. by D. Touretzky, M.C. Mozer, and M. Hasselmo. Vol. 8. MIT Press, 1995. URL: https://proceedings.neurips.cc/paper_files/paper/1995/file/9c3b1830513cc3b8fc4b76635d32e692-Paper.pdf.

[22] Sagar Sharma, Simone Sharma, and Anidhya Athaiya. “Activation functions in neural networks”. In: *Towards Data Sci* 6.12 (2017), pp. 310–316.

[23] Bernard W Silverman. *Density estimation for statistics and data analysis*. Routledge, 2018.

[24] Kelei Sun et al. “A convolutional neural network model based on improved softplus activation function”. In: *International Conference on Applications and Techniques in Cyber Intelligence ATCI 2019: Applications and Techniques in Cyber Intelligence 7*. Springer. 2020, pp. 1326–1335.

[25] Arnaud Van Looveren and Janis Klaise. “Interpretable Counterfactual Explanations Guided by Prototypes”. In: *Machine Learning and Knowledge Discovery in Databases. Research Track*. Ed. by Nuria Oliver et al. Cham: Springer International Publishing, 2021, pp. 650–665. ISBN: 978-3-030-86520-7.

[26] Thomas Villmann, Stefan Haase, and Frank-Michael Schleif. “Learning vector quantization methods for interpretable classification learning”. In: *Computational Statistics* 32.2 (2017), pp. 575–593.

[27] Sandra Wachter, Brent Mittelstadt, and Chris Russell. “Counterfactual explanations without opening the black box: Automated decisions and the GDPR”. In: *Harv. JL & Tech.* 31 (2017), p. 841.

Statutory Declaration in Lieu

I – MD RAJIBUL ISLAM – do hereby declare in lieu of an oath that I have composed the presented work independently on my own and without any other resources than the ones given.

All thoughts taken directly or indirectly from external sources are correctly acknowledged.

This work has neither been previously submitted to another authority nor has it been published yet.

Mittweida, 10. July 2025 — Rajibul

Location, Date — MD RAJIBUL ISLAM, M.Sc.