

---

# **GLVQ with Renyi Divergence**

***Release 1.0***

**Foqrul Islam**

**Jul 27, 2021**



**CONTENTS:**

<b>1</b>	<b>Renyi</b>	<b>1</b>
1.1	Renyi_final module . . . . .	1
<b>2</b>	<b>Installation guide</b>	<b>5</b>
2.1	Execution . . . . .	5
<b>3</b>	<b>Indices and tables</b>	<b>7</b>
	<b>Python Module Index</b>	<b>9</b>
	<b>Index</b>	<b>11</b>



## 1.1 Renyi\_final module

**class** `Renyi_final.GLVQ(class_prototype)`

Bases: `object`

Generalized Learning Vector Quantization (GLVQ)

**Attributes:**

**class\_prototype:** The number of prototype of each class to be learned

**Renyi\_Divergence**(*data\_in, prototypes*)

Calculate the Renyi divergence between datapoints and prototypes.

**Parameters**

- **data\_in** – A  $n \times m$  matrix of datapoints.
- **prototypes** – A  $n \times m$  matrix of prototypes of each class

**Returns** A  $n \times m$  matrix with Renyi divergence between datapoints and prototypes

**calculate\_d\_minus**(*data\_labels, prot\_labels, pro\_types, Renyi\_div*)

Calculate the distance between data points and prototypes

**Parameters**

- **data\_labels** – A  $n$ -dimensional vector containing the labels for each datapoint.
- **prot\_labels** – A  $n$ -dimensional vector containing the labels for each prototype.
- **pro\_types** – A  $n \times m$  matrix of prototypes of each class.
- **Renyi\_div** – A  $n \times m$  matrix with Renyi divergence between datapoints and prototypes.

**Returns** **d\_minus**: A  $n$ -dimensional vector having distance between datapoints and prototypes with different label. **w\_minus**: A  $m \times n$  matrix of nearest incorrect matching prototypes. **w\_minus\_index**: A  $n$ -dimensional vector having the indices for nearest prototype to datapoints with different label.

**calculate\_d\_plus**(*data\_labels, prot\_labels, pro\_types, Renyi\_div*)

Calculate the distance between data points and prototypes

**Parameters**

- **data\_labels** – A  $n$ -dimensional vector containing the labels for each datapoint.
- **prot\_labels** – A  $n$ -dimensional vector containing the labels for each prototype.
- **pro\_types** – A  $n \times m$  matrix of prototypes of each class.

- **Renyi\_div** – A  $n \times m$  matrix with Renyi divergence between datapoints and prototypes.

**Returns** **d\_plus**: A  $n$ -dimensional vector having distance between datapoints and prototypes with the same label. **w\_plus**: A  $m \times n$  matrix of nearest correct matching prototypes. **w\_plus\_index**: A  $n$ -dimensional vector having the indices for nearest prototype to datapoints with the same label.

**change\_in\_w\_minus**(*data\_in, learning\_rate, classifier, w\_minus, w\_minus\_index, d\_plus, d\_minus*)  
Calculate the update of prototypes

**Parameters**

- **data\_in** – A  $n \times m$  matrix of datapoints.
- **learning\_rate** – Learning rate (step size)
- **classifier** – Classify the vector as a winner or runner up prototype.
- **w\_minus** – A  $m \times n$  matrix of nearest incorrect matching prototypes.
- **w\_minus\_index** – A  $n$ -dimensional vector having the indices for nearest prototype to datapoints with different label.
- **d\_plus** – A  $n$ -dimensional vector having the distance between datapoints and prototypes with the same label.
- **d\_minus** – A  $n$ -dimensional vector having the distance between datapoints and prototypes with different label.

**Returns** The result of the updated prototype after calculating the update of prototypes with the same or different label.

**change\_in\_w\_plus**(*data\_in, learning\_rate, classifier, w\_plus, w\_plus\_index, d\_plus, d\_minus*)  
Calculate the update of prototypes

**Parameters**

- **data\_in** – A  $n \times m$  matrix of datapoints.
- **learning\_rate** – Learning rate (step size)
- **classifier** – Classify the vector as a winner or runner up prototype.
- **w\_plus** – A  $m \times n$  matrix of nearest correct matching prototypes.
- **w\_plus\_index** – A  $n$ -dimensional vector having the indices for nearest prototype to datapoints with the same label.
- **d\_plus** – A  $n$ -dimensional vector having the distance between datapoints and prototypes with the same label.
- **d\_minus** – A  $n$ -dimensional vector having the distance between datapoints and prototypes with different label.

**Returns** The result of the updated prototype after calculating the update of prototypes with the same or different label.

**create\_data\_prototype**(*data\_in, data\_labels, class\_prototype*)  
Calculate prototypes with labels. The calculation is based on the mean or at random based on the prototype in each class.

**Parameters**

- **data\_in** – A  $n \times m$  matrix of datapoints
- **data\_labels** – A  $n$ -dimensional vector containing the labels for each datapoint

- **class\_prototype** – The number of prototype in each class to be learned. If the number of prototypes

in each class is 1, the prototypes are assigned in the mean position, otherwise it is assigned at random.

#### Returns

**pro\_labels:** A n-dimensional vector having the labels for every prototype

**Prototypes:** A n x m matrix of prototypes for the training purpose.

**data\_predictor**(*input\_value*)

The prediction of the labels for the data. The data are represented by the test-to-training distance matrix. Every datapoint will be assigned to the closest prototype.

**Parameters** **input\_value** – A n x m matrix of distances from the test to the training datapoints.

**Returns** **y\_label** - A n-dimensional vector having the predicted labels for each and every datapoint.

**do\_processing**(*data\_in, data\_labels, learning\_rate, epochs*)

Main function to train the algorithm

#### Parameters

- **data\_in** – A m x n matrix of distance.
- **data\_labels** – A m-dimensional vector containing the labels for each datapoint.
- **learning\_rate** – Learning rate (step size).
- **epochs** – The maximum number of optimization iterations.

**Returns** A n-dimensional updated prototype vector.

**new\_proto\_types** = **array([], dtype=float64)**

**normalize\_data**(*data\_in*)

Normalizing the data so that the data is between the range 0 and 1 and also we can further the data as a probability distribution.

**Parameters** **data\_in** – A n x m matrix of our input data.

**Returns** **Data\_normalized:** A n x m matrix with values between 0 and 1

**plot**(*data\_in, data\_labels, prot\_types, prot\_labels*)

Visualizing the data and prototypes in a scatter plot.

#### Parameters

- **data\_in** – A n x m matrix of datapoints.
- **data\_labels** – A n-dimensional vector containing the labels for each datapoint.
- **prot\_types** – A n x m matrix of prototypes of each class.
- **prot\_labels** – A n-dimensional vector containing the labels for each prototype.

**prototype\_data\_labels** = **array([], dtype=float64)**

**sigmoid\_calc**(*j, beta=10*)

Calculate the sigmoid activation function.

#### Parameters

- **j** – A n x m matrix of datapoints or any value
- **beta** – The value of the parameter is taken as 10

**Returns** It returns values in the range 0 to 1



## INSTALLATION GUIDE

The following requirements need to be fulfilled to run the program:

- python with minimum version 2.8
- numpy with minimum version 1.19.0
- matplotlib
- Scikit Learn.

### 2.1 Execution

- Copy the file in a folder
- Install the required packages mentioned in the 'Installation guide'
- Run the program



## INDICES AND TABLES

- `genindex`
- `modindex`
- `search`



## PYTHON MODULE INDEX

r

Renyi\_final, [1](#)



## INDEX

### C

`calculate_d_minus()` (*Renyi\_final.GLVQ method*), 1  
`calculate_d_plus()` (*Renyi\_final.GLVQ method*), 1  
`change_in_w_minus()` (*Renyi\_final.GLVQ method*), 2  
`change_in_w_plus()` (*Renyi\_final.GLVQ method*), 2  
`create_data_prototype()` (*Renyi\_final.GLVQ method*), 2

### D

`data_predictor()` (*Renyi\_final.GLVQ method*), 3  
`do_processing()` (*Renyi\_final.GLVQ method*), 3

### G

`GLVQ` (*class in Renyi\_final*), 1

### M

`module`  
    *Renyi\_final*, 1

### N

`new_proto_types` (*Renyi\_final.GLVQ attribute*), 3  
`normalize_data()` (*Renyi\_final.GLVQ method*), 3

### P

`plot()` (*Renyi\_final.GLVQ method*), 3  
`prototype_data_labels` (*Renyi\_final.GLVQ attribute*), 3

### R

`Renyi_Divergence()` (*Renyi\_final.GLVQ method*), 1  
`Renyi_final`  
    *module*, 1

### S

`sigmoid_calc()` (*Renyi\_final.GLVQ method*), 3