Incremental Conic Functions Algorithm
User Guide

Version	*v1.0*
Legal Code License	*MIT*
Depends	*Python (2.7)*
Compilation requirements, operating environments & dependencies	*Numpy, Sklearn and Gurobi packages must be installed.*
Authors	*Emre Cimen, Gurkan Ozturk, Omer Nezih Gerek*
Support email for questions	*ecimen@anadolu.edu.tr*
Date	*05 Sep 2017*
Citation	*- Cimen E., Ozturk G. Gerek O. N, "Incremental conic functions algorithm for large scale classification problems", DSP–SoftwareX Special Issue (2017).* *- Cimen E., Ozturk G., and Gerek O.N, ICF: An algorithm for large scale classification with conic functions., Software X*

This document is created to explain how the ICF algorithm is used. Please follow the steps below:

1- Preliminaries: First of all, please be sure that the Python (2.7) and the Numpy, Sklearn and Gurobi packages must be installed. Sklearn package is needed in order to run the k-Means clustering algorithm.

The Gurobi package (http://www.gurobi.com/) is needed to solve *linear programming* problems. Linear programming model is solved in **PCF** function. If one wants to use a different LP solver, the **PCF** function should be modified. $\mathbf{A}_r$, $\mathbf{B}$, $\mathbf{c}_r$ are passed to the **PCF** function and return $\mathbf{w}, \xi, \gamma, \mathbf{c}$ and LP solution status should be returned(Please find scientific explanations in detail from the related papers above).

2- Choosing the file for running: There are 3 python files in the project. If one wants to run the algorithm with n-fold cross validation, **ICF_Fold** file should be used. If one has separate training and test files **ICF_Training_Test** file.

3- Setting the inputs: The inputs of the ICF algorithm is dataset file (path) and the parameters: $\tau_1, \tau_2 \in [0, \infty]; \epsilon, \tau_3, \tau_4 \in [0, 1]$.

ICF_Fold or **ICF_Training_Test** file:

```
#################### INPUTS ############################
data = readData('illustrativeExample.csv') #Dataset path should be given here.
#data = readArff('example.arff')

tolpr=0.95 #epsilon in algorithm 1 This parameter is a threshold to decide whether an LP is necessary, or not, in the ICF algorithm.
# High values for this parameter increases the chance of calling the LP for PCF construction, while its low values favor for algebraic cone construction
# (corresponding to a faster, but possibly lower resolution result).

TF=10 # Means 10 fold cross validation will be done.
#####################
```

Dataset file can be in csv or arff format. If you are using csv file rows should correspond to data points and columns should correspond to the features. All features should be separated with comma "," and class labels should be integers, 1 to η, in the last column. The path should be given in the **readData** function or **readArff** function. **tolpr**

variable keeps the ϵ treshold and **TF** variable keeps the number of cross validation.

ICF_Purity file:

```
####################### PARAMETERS ############################
# Set Algorithm 2 Inputs
#The parameter defines the ratio of how much of redundant data are to be eliminated from set A.
# As tau1 increase, more data elimination occurs.
# Similarly,tau2 defines the data elimination ratio for set B.
# If more data is eliminated from sets, the operations become fast while reducing the training accuracy.
# The other two parameters,tau3 and tau4, define thresholds to increase/decrease number of sub-sets.
# tau1 and tau2 is from [0, inf]; tau3 and tau4 is from [0, 1]
tau1=0.9
tau2=1.1
tau3=0.95
tau4=0.05
```

Please set the $\tau_1, \tau_2 \in [0, \infty]; \epsilon, \tau_3, \tau_4 \in [0, 1]$ parameters.

All settings may effect the computation time and the accuracy. Please find scientific explanations in detail from the related papers.

4- Running the code: Run the **ICF_Fold** or **ICF_Training_Test** file. All function parameters and accuracy rates will be written to the console.

If you test the *illustrativeExample.csv* with 2-fold cross validation, you should get the following results in the console:

```
####################### 1 .Fold ###################################
1 .Fold Training Accuracy: % 97.99
1 .Fold Test Accuracy : % 96.0
------------------------------------------------------------
For class  1 the classifiers are:
1 .Fold 1 .class  1 .cluster classification function that separates A from B: gjr = w.(x-c) + ksi*|w.(x-c)|-gamma
w = [0.07417585844879614, 0.04887718132927679]
ksi = 0.191577399286
gamma = 11.9770915366
center = [ 194.17647059  172.94117647]
------------------------------------------------------------
1 .Fold 1 .class  2 .cluster classification function that separates A from B: gjr = |x-c|_2  - gamma
gamma = 34.3325254651
center = [ 201.6         322.06666667]
------------------------------------------------------------
1 .Fold 1 .class  3 .cluster classification function that separates A from B: gjr = |x-c|_2  - gamma
gamma = 28.8151260914
center = [ 237.375  300.125]
------------------------------------------------------------
1 .Fold 1 .class  4 .cluster classification function that separates A from B: gjr = w.(x-c) + ksi*|w.(x-c)|-gamma
w = [1.440727521172651e-16, 0.002587322121604347]
ksi = 0.134540750323
gamma = 7.78465518957
center = [ 113.5         148.73076923]
------------------------------------------------------------
1 .Fold 1 .class  5 .cluster classification function that separates A from B: gjr = |x-c|_2  - gamma
gamma = 39.0050594789
center = [ 148.27272727  337.13636364]
------------------------------------------------------------
1 .Fold 1 .class  6 .cluster classification function that separates A from B: gjr = w.(x-c) + ksi*|w.(x-c)|-gamma
w = [0.017989078059975846, 4.445470860750914e-18]
ksi = 0.10986186958
gamma = 5.30756326517
center = [ 269.27777778  216.77777778]
```

...

. . .

```
------------------------------------------------------------
2 .Fold 3 .class  4 .cluster classification function that separates A from B: gjr = |x-c|_2  - gamma
gamma = 47.5869198293
center = [  81.88888889  315.77777778]
------------------------------------------------------------
2 .Fold 3 .class  5 .cluster classification function that separates A from B: gjr = w.(x-c) + ksi*|w.(x-c)|-gamma
w = [0.10136386055992785, 0.012427859491685235]
ksi = 0.162401989833
gamma = 10.2609858423
center = [  31.66666667  193.85714286]
------------------------------------------------------------
2 .Fold 3 .class  6 .cluster classification function that separates A from B: gjr = |x-c|_2  - gamma
gamma = 51.7762872975
center = [ 322.42105263  151.15789474]
------------------------------------------------------------
2 .Fold 3 .class  7 .cluster classification function that separates A from B: gjr = |x-c|_2  - gamma
gamma = 47.4160992956
center = [ 270.85   39.4 ]
------------------------------------------------------------
2 .Fold 3 .class  8 .cluster classification function that separates A from B: gjr = w.(x-c) + ksi*|w.(x-c)|-gamma
w = [3.7765869946883147e-10, 6.139459702749147e-12]
ksi = 0.230769209555
gamma = 10.1025632725
center = [  96.22222222  264.33333333]
------------------------------------------------------------
2 .Fold 3 .class  9 .cluster classification function that separates A from B: gjr = |x-c|_2  - gamma
gamma = 38.5098685359
center = [ 338.71428571  242.85714286]
------------------------------------------------------------
2 .Fold 3 .class  10 .cluster classification function that separates A from B: gjr = |x-c|_2  - gamma
gamma = 27.027265936
center = [  68.11111111  221.44444444]
------------------------------------------------------------
####################################################################
Training Accuracy : % 98.195
Test Accuracy : %  95.99
####################################################################
--- 1.63735198975 seconds elapsed ---
```