README:

Requirement:

1) The code is based on Python 2.7.
2) The required libraries are sklearn, scipy, numpy. Any version updated before 1/15/2019 is valid to run the code.
3) The Gaussian Process regressor is build on sklearn. To enable fast training/sampling, one need to replace the Gaussian Process folder with the Gaussian Process folder provided in this folder. Then in active learning iterations where kernels are not updated, the training/samping/testing speed will be significantly improved.

The core method:

AL_In_Compress_GPR_BayesianOPT(X,Y,trainRate,poolRate,testRate,sampleSize,sampleBatch=1,bestRecov=2,compressRate=0.5,kernelUpdate=1,alpha=1e-5,PCAWrapper=False,PCARate='full',eigenWeigthed=False,sharedSearchSpace=True,sw=0.6,cg=20,fixPhi=None,provideIndex=None).

How to call the core method:

(More detailed line by line comments on the code can be found in the .py file)
#methods for BPCACS
#Input parameters:
#X: Design matrix. X is supposed to have the shape:n*m. where n is the number of data instance and m is the number of features.
#Y: Label matrix. Y is supposed to have the shape:n*k. where k is the number of all potential labels. If the i th data x_i has label j then Y[i,j]=1 otherwise Y[i,j]=0;
#note: The random data split is not built in. You might want to shuffle the data before calling this method to have a more credible result.
#trainRate,poolRate,testRate: The percentages of the data/labels that will be used for initial training, sampling and testing Takes real number that <1 and the sum of the three should also be <=1.
#sampleSize the number of active learning iterations.
#sampleBatch: Tested for batch model compress sensing based AL. It belongs to the future work, please ignore this(fix the value to 1) if you want to replicate the work proposed in ICML paper.
#bestRecov: how many labels per instance you want the classifier to predict. The adaptive recover of labels is the future work. This parameter here is global. The suggested value is the cardinality of the dataset.
#compressRate: the compress rate

#Update the kernel parameters every how many active learning iterations. The update method uses bayesian optimization by default. To change the optimization method to simplex, change the method call bayesianHyperOpt() to directSearchHyperOpt(). To change to traditional max marginal likelihood optimization, just fit() the model again, it will automatically be done by sklearn.

#alpha: The parameter in Gaussian Process that keeps the diagonal of the covariance matrix non-zero.

#PCAWrapper: Boolean. If true, then pca is used to ensure orthogonal.

#PCARate: If 'full', all components will be used. Else if <1, the corresponding percentage of the component will be used.

#To switch to use BPCA and simply replace PCA() with BPCA() by importing bpca.py in this folder.

#eigenWeigthed: use the eigen value to improve sampling process.

#sharedSearchSpace: Whether use a global search space in bayesian opt or use individual search space for each label's kernel optimization.

#sw:search range for each parameter in kernel opt. Adopt log scale

#cg:search step(grids) for direct search.

#fixPhi: For future test purpose. You can pass a fixed compress matrix,Phi or make it adaptive to each AL iteration. If set to None, a random Phi generated from Gaussian distribution is used by default

#ProvidedIndex: a list of indices of train test and pool. If you want to use some prefered index rather than a randomly split one, you can pass the index through this parameter.

#Return the Macro F1 for each AL iteration. and (for test purpose) history of updated kernel parameters.

Testing dataset:

We provide corel5K for testing purpose. The rest of the datasets used in the paper are available via link http://mulan.sourceforge.net/datasets-mlc.html.