Mithun Mohan Raj

Assignment 2: Support vector machines

Algorithm:

"Support Vector Machine(SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges. In this algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiate the two classes very well.**(1)**

Kernel functions:

"The kernel function can be interpreted as a measure of similarity between the samples xi and xj which it allows SVM classifiers to perform separations even with very complex boundaries. the kernel tricks extends the class of decision functions to the non-linear case by mapping the samples from the input space X into a high-dimensional feature R without ever having to compute the mapping explicitly, in the hope that the samples will gain meaningful linear structure in R.

Linear kernel:

The Linear kernel is the simplest kernel function. It is given by the inner product <x,y> plus an optional constant c. Kernel algorithms using a linear kernel are often equivalent to their non-kernel counterparts.

$$k(x, y) = x^T y + c$$

Polynomial kernel

The Polynomial kernel is a non-stationary kernel. It is well suited for problems where all the training samples are normalized. The parameters which must be settled are the slope gamma. The constant term r and the polynomial degree d.

$$K(x_i, x_j) = (\sigma\, x_i^T x_j + r)^d, \sigma > 0$$

RBF kernel:

RBF (Gaussian) kernels are a family of kernels where a distance measure is smoothed by a radial function (exponential function) [10]. This kernel nonlinearly maps samples into a higher dimensional space so it, unlike the linear kernel, can handle the case when the relation between class labels and attributes is nonlinear.

$$K(x_i, x_j) = \exp(-\sigma \left\| x_i - x_j \right\|^2), \sigma > 0$$

Sigmoid kernel

The kernel must satisfy Mercer's theorem, and that requires that the kernel be positive definite. However, the Sigmoid kernel, which, despite its wide use, it is not positive semi-definite for certain values of its parameters. Thus, the parameters σ , r must be properly chosen otherwise, the results may be drastically wrong, so much so that the SVM performs worse than random.

$$K(x_i, x_j) = \tanh(\sigma x_i^T x_j + r)$$ "**(2)**

Hyper-Parameters:

"Soft-margin Constant C:

"C affects the trade-off between complexity and proportion of nonseparable samples and must be selected by the user. More specifically the trade-off between errors on training data set and margin maximization. A smaller C will allow more errors and margin errors and usually produce a larger margin. When C goes to Infinity, svm becomes a hard-margin. Note that the value of C is chosen by users based on training experiments.

The degree of the polynomial kernel:

The dimensionality of polynomial kernel function $k(x, x') = (x \cdot x')^d$ is dependent on its degree parameter . The lowest degree polynomial is the linear kernel (or no kernel at all), which is not sufficient when a non-linear relationship between feature exists. Higher degree kernel can map patterns into higher dimensional space and therefore yield a higher dimensional hyperplane.

The width parameter of the RBF kernel:

Gaussian kernel is expressed as $k(x, x') = exp(-\gamma(x \cdot x')^2)$. γ defines the region near a fixed point. When γ is small, a given data point has a non-zero kernel value relative to any example in the set of support vectors. γ increase, the locality of the support vector expansion increases, leading to greater curvature of the decision boundary."**(3)**

Data-set:

Glass dataset from the UCI Machine learning repository consists of 214 training instances of 7 different types of glasses where each instance has 11 set of attributes including the class label.

```
Attribute Information:
   1. Id number: 1 to 214
   2. RI: refractive index
   3. Na: Sodium (unit measurement: weight percent in corresponding oxide, as
                  are attributes 4-10)
   4. Mg: Magnesium
   5. Al: Aluminum
   6. Si: Silicon
   7. K: Potassium
   8. Ca: Calcium
   9. Ba: Barium
```

```
10. Fe: Iron
11. Type of glass: (class attribute)
    -- 1 building_windows_float_processed
    -- 2 building_windows_non_float_processed
    -- 3 vehicle_windows_float_processed
    -- 4 vehicle_windows_non_float_processed (none in this database)
    -- 5 containers
    -- 6 tableware
    -- 7 headlamps
```

Training set:

- **80%** of the original dataset : 171 instances of glass data with 9 features excluding class labels and "Id" in the original dataset.

Training labels:

- **80%** of the original dataset: 171 instances of glass data with only including the "class labels"

Test data:

- **20%** of the original data : 43 instances of glass data with 9 features excluding class labels and "Id" in the original dataset.

Test labels:

- **20%** of the original data : 43 instances of glass data with only including the "class labels"

Cross-validation Procedure:

1. After random shuffling the original data,Split the shuffled data into training set,training labels,test set,test labels.

   x_train, x_test, y_trainlabel, y_testlabel = train_test_split(x, y, test_size = 0.20,shuffle=False)

2. Define the support vector classifier for the respective kernel.

```
svc =SVC(kernel='rbf', gamma=g,C=C)
```

3. For each fold ,
    - Choose 80% of the original training data as new training set and remaining 20% as the validation set.
    - Fit the svm model with the new training data.

      svc.fit(newx_train,newy_trainlabel)
    - Let the trained svm model make predictions on validation set

      y_pred = svc.predict(newx_test)
    - Get the accuracy score of the model by comparing the predicted label and the original label

      accuracy_score(y_testlabel,y_pred)
4. Repeat the step 3 five times to perform 5-cross validation and for each fold the validation set is chosen on training data(20%) in a successive manner.
5. Get the accuracy scores from five folds and store it in a list or array.

   scores = cross_val_score(svc, x_train, y_trainlabel, cv=5, scoring='accuracy')

Hyper-parameter evaluation:

The range for each of the hyper-parameter that is used is given below:

Cs = [2-5,2**-4,2**-3,2**-2,2**-1,2**0,2**1,2**2,2**3,2**4,2**5,2**6,2**7,2**8,2**9,2**10]**

gamma_range = [2-15,2**-13,2**-11,2**-9,2**-7,2**-5,2**-3,2**-1,2**1,2**3,2**5,2**4,2**6,2**7,2*8]**

degree=[1,2,3,4,5,6,7,8]

coeff=[1,0.9,0.8,0.7,-1]

Pseudocode to evaluate the hyper-parameters:

1. For linear kernel, the hyper parameter c is used:

   for C in Cs:
2. For Rbf kernel, the hyper-parameters c and gamma are used:

   for C in Cs:

   for g in gamma_range:

3. For polynomial kernel , the hyper-parameters c ,gamma and degree is used:

```
for g in gamma_range:
 for p in degree:
  for c in Cs:
```

4. For sigmoid kernel, the hyper-parameters coeff0, c and gamma are used:

```
for co in coeff:
 for g in gamma_range:
  for c in Cs:
```

5. Define the support vector classifier for the corresponding kernel and give the appropriate hyper-parameters for the respective kernel.

```
svc =SVC(kernel='poly',degree=p,C=c,gamma=g)
```

6. Perform the 5 cross-validation procedure to get five accuracy scores for hyper-parameters passed in.

```
scores = cross_val_score(svc, x_train, y_trainlabel, cv=5, scoring='accuracy')
```

7. Determine the maximum of the five accuracy scores and append the score along with the corresponding hyper-parameters of the kernel in a list.

```
opt.append((p,c,np.amax(scores),g))
```

8. Sort the list according to the accuracy score in the descending order and the first row of the sorted list gives you the optimal hyper-parameter for the corresponding kernel.

```
sort(key=operator.itemgetter(2),reverse=True)
print("optimal hyperparameter
 are:","degree=",opt[0][0],"c=",opt[0][1],"gamma=",opt[0][3],"accuracy=",opt[0][2])
```

9. Repeat the steps 5 to 8 for all the kernels to get the hyper-parameters for the corresponding kernels.

Results(One vs one):

From the range for hyper-parameters mentioned in the above section , the optimal hyper-parameters obtained for each kernel are :

For rbf kernel:
optimal hyperparameter are: c= 32 gamma= 0.5 accuracy= 0.8108108108108109

For linear kernel:
optimal hyperparameter are: c= 64 accuracy= 0.7727272727272727

For poly kernel:
optimal hyperparameter are: degree= 2 c= 16 gamma= 0.25 accuracy= 0.8648648648648649

For sigmoid kernel:
optimal hyperparameter are: gamma= 0.0001220703125 c0eff= -1 c= 64 accuracy= 0.4666666666666667

Overall accuracy(One vs One):

After finding the optimal parameters for each kernel, the overall accuracy for each kernel is calculated by the following steps:

A. Define the support vector classifier with the optimal hyper-parameters corresponding to each kernel.
```
svc = SVC(kernel='poly',C=optc,degree=optp,gamma=optg)
```
B. Fit the model using the training data and training labels.
```
svc.fit(x_train,y_trainlabel)
```
C. Use the trained model to make predictions on test data.
```
y_pred = svc.predict(x_test)
```
D. Calculate the accuracy of the model by comparing the predicted and original test labels.
```
print("overall accuracy for poly",accuracy_score(y_testlabel,y_pred))
```
E. Perform the above steps to get the overall accuracy for all the kernels.

Results:

results for rbf :
overall accuracy for rbf 0.7906976744186046

results for linear:
overall accuracy for linear 0.7209302325581395

results for poly:
overall accuracy for poly 0.7209302325581395

Results for sigmoid:
overall accuracy for sigmoid 0.4883720930232558

One vs Rest classifier

Algorithm:

"One vs rest decomposition divides an m class problem into m binary problems. Each problem is face up by a binary classifier which is responsible of distinguishing one of the classes from all other classes. The learning step of the classifiers is done using the whole training data, considering the patterns from the single class as positives and all other examples as negatives. In validation phase, a pattern is presented to each one of the binary classifiers and then, the classifier which gives a positive output indicates the output class."**(4)**

Steps to implement one vs rest classifier:

1, create the one vs rest classifier by using the Onevsrestclassifier function available in the library .

```
svc =OneVsRestClassifier(SVC(kernel='rbf', gamma=g,C=C,class_weight='balanced'))
```

2, Give the appropriate hyper-parameters for the kernels.

3, Perform the cross-validation procedure as done in the one vs one classifier.

4,calculate the optimal hyper-parameters .

5, With the optimal hyper-parameters ,find the overall accuracy as done in the one vs one classifier.

Results:

The range for each of the hyper-parameter that is used is given below:

- **Cs = [2**-5,2**-4,2**-3,2**-2,2**-1,2**0,2**1,2**2,2**3,2**4,2**5,2**6,2**7,2**8,2**9,2**10]**
- **gamma_range = [2**-15,2**-13,2**-11,2**-9,2**-7,2**-5,2**-3,2**-1,2**1,2**3,2**5,2**4,2**6,2**7,2*8]**
- **degree=[1,2,3,4,5,6,7,8]**
- **coeff=[1,0.9,0.8,0.7,-1]**

Results for rbf kernel :

optimal hyperparameter are: c= 2 gamma= 8 accuracy= 0.8064516129032258

training time is 0.011903762817382812

overall accuracy for rbf 0.627906976744186

Results for linear kernel

optimal hyperparameter are: c= 64 accuracy= 0.75

training time 0.06051206588745117

overall accuracy for linear 0.6046511627906976

Results for poly kernel:

optimal hyperparameter are: degree= 2 c= 16 gamma= 0.015625 accuracy= 0.8064516129032258

training time 0.1334233283996582

overall accuracy for poly 0.6744186046511628

Results for sigmoid kernel:

optimal hyperparameter are: gamma= 0.0001220703125 c0eff= -1 c= 1 accuracy= 0.7

training time 0.006943702697753906

overall accuracy for sigmoid 0.4418604651162791

One vs One V/S One vs Rest

	One vs One		**One vs Rest**	
Kernels	**Overall accuracies**	**Training time (seconds)**	**Overall accuracies**	**Training time(seconds)**
Rbf	0.7906976744186046	0.003471851348876953	0.627906976744186	0.011903762817382812
Linear	0.7209302325581395	0.019840002059936523	0.6046511627906976	0.06051206588745117
Polynomial	0.7209302325581395	0.1755836009979248	0.6744186046511628	0.1334233283996582
Sigmoid	0.4883720930232558	0.0019834041595458984	0.4418604651162791	0.006943702697753906

Conclusion :

From the above table ,we can infer that one vs one classifier performs better on the given datasets than the one vs rest classifier in terms of training time and prediction accuracy.

Re-weighting procedure

Algorithm:

“weights are the "class weights" for unbalanced class problems (assuming that the class weight is 1.0 by default for all classes). In simple words: Using weights for the classes will drag the decision boundary away from the center of the under-represented class more towards the over-represented class.”**(4)**

Procedure:

1, Define the support vector classifier for the corresponding kernel and give the appropriate hyper-parameters for the respective kernel.

2, Create a class_weight() function for calculating weights for each of the classes available in the input data given to that function :

A,Count the number of classes in data.
B,Count the number of samples in each class.
C,count the number of samples of data.

Weight for ith class = total number of data samples /(number of samples in ith class * Number of classes)

This class_weight() should return the weights for all the classes in the data given.

3, perform the cross-validation procedure and call the class_weight() function for each of the five folds and fit the model with re-weighted training and validated data.

4,calculate the optimal hyper-parameters .

5, With the optimal hyper-parameters ,find the overall accuracy as done in the one vs one classifier.

6, Repeat the above steps for all the kernel and compute the results.

Results :

The range for each of the hyper-parameter that is used is given below:

- **Cs = [2**-5,2**-4,2**-3,2**-2,2**-1,2**0,2**1,2**2,2**3,2**4,2**5,2**6,2**7,2**8,2**9,2**10]**
- **gamma_range = [2**-15,2**-13,2**-11,2**-9,2**-7,2**-5,2**-3,2**-1,2**1,2**3,2**5,2**4,2**6,2**7,2*8]**
- **degree=[1,2,3,4,5,6,7,8]**

- **coeff=[1,0.9,0.8,0.7,-1]**

Results for rbf:
optimal hyperparameter are: c= 64 gamma= 0.0078125 accuracy= 0.7837837837837838
overall accuracy for rbf = 0.5813953488372093

Results for linear:
optimal hyperparameter are: c= 64 accuracy= 0.75
overall accuracy for linear = 0.6744186046511628

Results for poly:
optimal hyperparameter are: degree= 4 c= 0.03125 gamma= 0.015625 accuracy= 0.8108108108108109
overall accuracy for poly = 0.7209302325581395

Results for sigmoid:
optimal hyperparameter are: gamma= 0.0001220703125 c0eff= 0.7 c= 256 accuracy= 0.5454545454545454
overall accuracy for sigmoid 0.4186046511627907

Refernces:

(1) https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/

(2) **https://arxiv.org/ftp/arxiv/papers/1507/1507.06020.pdf(kernel)**

(3) https://yunhaocsblog.wordpress.com/2014/07/27/the-effects-of-hyperparameters-in-svm/(hyp

(4) https://sci2s.ugr.es/ovo-ova(ova)

(5) **(5)https://www.researchgate.net/post/What_is_the_Weight_vector_parameter_in_Support_Vector_Machine_in_machine_learning**