Lab 5 - SVMs

Wednesday, July 16, 2008
12:46 PM

PURPOSE

Goal: By the end of this lab, you will understand the how to build and test with a SVM model.

SETUP

Verify that SVMlab folder is installed in your local working directory. (If not, please download it from /usr/ccrma/courses/mir2009/Toolboxes / Utilities / SVMlab folder.)

Add the entire SVMlab folder (with subfolders) to your Matlab path.

Now, we need to reset the permissions of the files in this folder:

Open a Terminal window and cd to the directory where you placed your SVMlab folder.
cd to the folder /Libsvm-2.86/tools/
In the Terminal, type:

```
chmod 755 grid.py
chmod 755 svmtrain
chmod 755 svmpredict
chmod 755 svmscale
```

To verify that everything worked, type in **grid.py.** You should see something that says "Usage: grid.py [-log2c begin, eng] etc."

If you see "permission denied", then something went wrong - repeat the above process.

SECTION 1: BUILDING AN SVM

FEATURE EXTRACT

You've done this a million times by now.
Since we need data that we can assign LABELS to, feature extract a collection of instrument samples (as many features as you want). Note that you can either a) treat the instrument sample .wav file as 1 giant frame (analyzing the entire audio signal), or b) analyze the signal using frames size of say, 80 to 100 ms per audio file and then take the mean of each of the features, creating a mean "signature" for the file.

If you just can't get enough of the drum samples, start with:

Input "Test snares" and call them 1.
Input "Test kicks" and call them 0.

For variety, choose instruments based on samples in:
/scratch/audio/Miscellaneous Loops Samples and SFX/Instrument Samples

Don't forget to scale the feature data, save the scaling coefficients so we can scale the test data to be in the same range.

Create a label vector using class labels {1,-1}
Kernels (linear, polynomial, RBF, sigmoid)

Find the best parameters (C and gamma)

EXPORT DATA

We are using an external program (a python script) to run a grid search and look for the best values of C and gamma. To export your data so that libSVM can read it, I've created a helper function.

mat2libSVMFormat.m

mat2libSVMFormat(data,label,filename)

For example:

```
mat2libSVMFormat(features , labels,'~/Matlab/libsvm-2.86/tools/myData.txt')
```

This file needs to be saved in the save directory as the libSVM tools (grid.py etc) are located.

Open up the file to see the particular format that libSVM prefers - and also to verify that the data was written out correctly.

GRID SEARCH
Now that the data has been exported from Matlab, open a Terminal window.

In the Terminal, type:

```
> grid.py myData.txt
```

You'll see the following sample text output to the screen...

```
[...]
[local] 13 -13 61.9048   (best c=512.0, g=0.001953125,  rate=66.6667)
[local] 13 1 52.381   (best c=512.0, g=0.001953125,  rate=66.6667)
[local] 13 -11 61.9048   (best c=512.0, g=0.001953125,  rate=66.6667)
[local] 13 -5 57.1429   (best c=512.0, g=0.001953125,  rate=66.6667)
[local] 13 -15 57.1429   (best c=512.0, g=0.001953125,  rate=66.6667)
[local] 13 3 57.1429   (best c=512.0, g=0.001953125,  rate=66.6667)
[local] 13 -9 42.8571   (best c=512.0, g=0.001953125,  rate=66.6667)
[local] 13 -3 61.9048   (best c=512.0, g=0.001953125,  rate=66.6667)
512.0 0.001953125  66.6667
```

These numbers show the parameters as they were chosen during a grid search, with the corresponding cross-validation error rate for that particular model. So, in effect, the grid search has built dozens of SVM models with various parameter settings, and chosen the parameters that it believe have the best chance of success given your current training data.

The numbers at the bottom are what we are most interested in.
The first number (512) is C, and the second number (0.001953125) is gamma.
Note that the current cross-validation accuracy is 66.6667% (Since I had a small number of samples in this colleciton, there are many possible best choices which also have 66.7% accuracy)

See the README in libSVM\libsvm-2.85\libsvm-2.85\tools for additional information on the easy.py and grid.py scripts, if you are the curious sort and find yourself interested.

Write down these values of C and gamma --- we'll use these in Matlab to build out SVM model.

SECTION 2: USING AN SVM

To build an SVM:

Type **svmtrain** in Matlab to review all of the myriad of options for it. (If you cannot find svmtrain, then make sure to add the folder libsvm-mat-2.86 to your Matlab path)

An example of how to train it on your feature data using the parameters returned by the grid search:

```
model = svmtrain(labels,features,'-t 2 -g 0.001953125   -c 512')
```

The "-t 2" specifies RBF kernel. "-g" species the value of gamma and "-c" specifies C.

To test with your SVM:

Feature extract some examples, and don't forget to **rescale** the data to the same mf and sf (scale factors) as before.

Now, to evaluate, all you do is:

```
svmpredict(labels,   features,     model)
```

"Labels?", you ask.
Yes, if you know the labels for your testing data, insert them into this vector. So, for example, you can insert the training labels and training feature data into this function, and svmpredict will automatically calculate the accuracy for you.

If you do not know the labels for your test data (likely the case), then insert a vector of zeros equal to the number of test samples that you have.

SECTION 3: HAVING FUN

Try redoing some of the previous labs' instrument classifiers or artist/genre classifiers using an SVM.

BONUS (only in your have time): CROSS VALIDATION

In today's lecture, we covered k-fold cross-validation.
You'll need some of this code and information to calculate your accuracy rate on your classifiers -- that is, if you chose to do this as your project.

Let's say we have 10-fold cross validation...

1. Divide test set into 10 random subsets.
2. 1 test set is tested using the classifier trained on the remaining 9.
3. We then do test/train on all of the other sets and average the percentages.

To achieve the first step (divide our training set into k disjoint subsets), use the function crossvalind.m (posted in the Lab 7 folder)

> INDICES = CROSSVALIND('Kfold',N,K) returns randomly generated indices
> for a K-fold cross-validation of N observations. INDICES contains equal
> (or approximately equal) proportions of the integers 1 through K that
> define a partition of the N observations into K disjoint subsets.
>
> You can type **help crossvalind** to look at all the other options.

Here is an outline of how to perform cross-validation on a classifier:

```
% cross_validation
k = 10;                              % how many folds do you want?
N =  size(features,1) ;              % this is the total number of observations or rows that we have
indices = crossvalind('Kfold',N,k)   % divide test set into 10 random subsets
for i = 1:10

     % SEGMENT DATA INTO FOLDS
     disp(['fold: ' num2str(i)])
     test = (indices == i) ;         % which points are in the test set
    train = ~test;                   % all points that are NOT in the test set

     % SCALE
     [trainingFeatures,mf,sf]=scale(features(train,:));

     % BUILD NEW MODEL
      model = knn(numFeatures,1,1,trainingFeatures,labels(train,:));

     % EVALUATE WITH TEST DATA
     model_output = knnfwd(model ,features(test,:))

     %  COUNT ERRORS
     errors(i) = mean ( model_output ~= labels(test,:) )

end
disp(['cross validation error: '  num2str(mean(errors))])
```

Recommended reading

A Practical Guide to Support Vector Classification
http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf

It provides an introduction to the libsvm tools, and motivations for why they were developed. It also highlights common mistakes.

Additional Resources

SVM Practical (How to get good results without cheating)
http://www.kyb.tuebingen.mpg.de/bs/people/weston/svmpractical/

Libsvm and Libsvm Tools
http://www.csie.ntu.edu.tw/~cjlin/libsvm/
http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/

The interactive Matlab SVM Demo that I demonstrated on Day 5:
http://homepages.cae.wisc.edu/~ece539/matlab/
http://homepages.cae.wisc.edu/~ece539/matlab/svmdemo.m

HELP!

Troubleshooting

If you are experiencing really bizarre results, it's sometimes worthwhile to double-check that the labels are set correctly. ("1" for positive example and "0" for negative example.) Not indicating the correct label will gravely affect the model.

Also, try deleting the temporary output feature file. Sometimes, the file isn't updated by Matlab -- but it's a silent error...

OPTIONAL : How to Build libSVM from source code.
Hopefully, you do NOT need to do this step - I've done the work for you. But for the curious...

The following steps will build the libsvm executables from their source - this is necessary to run them on our Linux machines.

1. Download the folder libsvm to your local Matlab folder.
2. Within the libsvm folder, open the file Makefile with a text editor.
3. On the 2nd line, change /usr/local/matlab to /opt/matlabR2006b
4. Save the file.
5. Open a Terminal window and cd to the folder containing the Makefile
6. Type make