Dec 9, 2018

STAT-5630

Statistical Machine Learning

Fall 2018

Project - Second Deliverable

Prediction of Epileptic Seizures using Machine Learning Method

Group: Data-Awesome

Alphonse N. Akakpo (ana2cy)

Ugur Kilic (uk7ud)

Zhangfan Jiang (zj6uc)

1. Introduction

The disease in which patients suffer seizures caused by a brain functionality disorder is called epilepsy. Epileptic seizures due to disorder in brain functionality can be dangerous for patient's health. Having medications at high doses for most patients can help to prevent seizure, however, patients frequently suffer side effects since using the medications often. Also, medications are not effective for 20-40% of patients according to American Epilepsy Society. Surgical removal could be one possibility to remove the disease but even after surgical removal of epilepsy-causing brain tissue, many patients continue to experience spontaneous seizures. Patients with epilepsy experience persistent anxiety due to the possibility of a seizure occurring even though they try surgery.

Prediction of epileptic seizures before the beginning of the onset have the potential to help patients to prevent the seizure by an immediate medication right before the seizure. Machine learning techniques and computational methods can be used for predicting epileptic seizures from Electroencephalograms (EEG) signals. In order for EEG-based seizure forecasting systems to work efficiently, computational algorithms must safely predict the periods of seizure and provide a sufficient time before the onset of seizure starts. If the onset of the seizure can be identified, the patient can be warned of a possible seizure and medications could be administered only when needed to prevent impending seizures, reducing overall side effects.

Early prediction of epileptic seizures can provide enough time before it actually occurs and the attack can be avoided by the drug. Epileptic seizures have four different stages: Interictal (between seizures, or baseline), Preictal (prior to seizure), Ictal (seizure), and Postictal (after seizures). The objective of this study is to predict the onset of seizure using different machine learning models and find the best model by comparing the models. Seizures can be predicted by detecting the beginning of the preictal state. For this study, Logistic regression was used as a classifier to predict the data is from a preictal state or not.

2. Data Description

The data set for this project includes seven individuals - five dogs and two human subjects with naturally occurring epilepsy. The data was supplied by an online competition through Kaggle.com. Data was recorded in the form of 10-minute-long iEEG signals which is obtained by electrodes placed on the surface of cerebral cortex and the sampling rate was 400 Hz. Each individual has interictal phase (negative class), preictal phase (positive class) and a test set which is not labelled. We used data from two dogs, dog 1 and dog 2, for which the number of samples are summarized in Table 1.

Table 1: Summary of the dataset used in this project

Class	Dog 1	Dog 2
Interictal	480	500
Preictal	24	42
Test	502	1000
Total	1006	1542

Number of samples for dog 1 and dog 2 are 1006 and 1542, respectively. There are 16 channels for each sample and so each 10-minute recording (one sample) can be represented as a matrix where rows are time and columns are electrode channels (1 to 16). The interictal data was labelled as no seizure (0) and preictal data was considered as the starting point of seizure (1). Number of samples in each recording is therefore 400 Hz x 600 sec x 16 channels ≈ 4 million. For dog 1, we have 1002 recording which corresponds to 4 million x 1002 recordings ≈ 4 billion total number of data points. We should note that there is a significant imbalance between the number of preictal and interictal samples. 480 interictal and 24 preictal for dog 1 and 500 interictal and 42 preictal for dog 2. We wanted to emphasize this point since it might be effective in the model that we are going to study.

3. Feature Extraction and Selection

In this section, we extract the features from the original EEG recording signals that are later used to train classifiers. Specifically, we consider the following features: (1) time domain features, (2) wavelet features, and (3) power spectral features.

For time domain features, the EEG signal in each channel is first resampled to 400 data points by interpolating the Fourier transform of the original signal. The correlation coefficients matrix is then calculated. We select eigenvalue of correlation coefficients matrix and coefficients in the upper portion of the correlation coefficients matrix as the first part of time domain features. We also calculate the entropy of the EEG signal in each channel and select the maximum, the minimum, and the mean of the entropy over all 16 channels as the second part of time domain features. The concatenation of two parts of features yields the final feature in the time domain that has in total 139 dimensions.

To compute wavelet features, we use EEG signal in every 6-second window with 3-second overlapping and apply wavelet decomposition of each window at level 5 using the Daubechies 1 wavelet. We compute the mean, the mean of absolute values, the root mean squared values, and the standard deviation of the wavelet decomposition vector, leading to 24 features (6 sets of decomposition vector and 4 features per vector) per channel. This process is applied independently to all 16 channels. The correlation feature is extracted from 16-channel wavelet decomposition vectors, as described in time domain feature extraction. In addition to the correlation features and wavelet decomposition features, we also calculate the entropy of wavelet decomposition vector and choose the mean and the standard deviation of entropy as parts of the wavelet features. The final wavelet feature has a dimension of 522.

The last feature we considered is the power spectral features. Analogous to wavelet feature extraction, we select EEG signal in every 6-second with 50% overlapping and apply the power spectral density estimate of the signal in each window using Welch's overlapped segment averaging estimator. The idea is to compute the total energy in different frequency bands. The following 9 bands are selected in this experiment: [0.1,2], [2,4], [4,8], [8,12], [12,18], [18,22], [22,30], [30,50], and [50,200] (Hz), which produces a 16-by-9 matrix where the i-th row contains the estimated energy for the i-th channel. Two additional features, i.e., the correlation features for the energy matrix, and the mean and the standard deviation of the entropy, are also used for the power spectral features, which has in total 282 dimensions.

4. Method and Results

After extracting the features from each recording, we had a feature matrix for dog 1 as 1006 x 943 and for dog 2 as 1542 x 943. For the competition, the data was released so that the interictal and preictal data sets would be training data and the test set would be test data. Since we couldn't find the labels for the test data, we emerged the interictal and preictal for both dogs and created new data frame. The sizes of new data frames were 943 x 504 for dog 1 and 943 x 542 for dog 2. We splitted both data as 70% to 30% where 70% was considered as training data and 30% as testing data. This may be misleading in the regression because the number of samples of preictal phase is considerably low compared to interictal phase.

We used logistic regression for classification. Before that, we added new columns to the matrices as 0 for the interictal and 1 for the preictal. The resulting data was 944 x 504 for dog 1 and 944 x 542 for dog 2 where the last column indicates the occurrence of seizure. In the logistic regression, the hypothesis that we applied has the form of a logistic function of;

$$h_\theta(x) = \frac{1}{1+ e^{-\theta^T x}}$$

where θ is the parameter vector to be optimized and x is the vector of features for each sample.

Applying this approach to our data we got the accuracy results as shown in Table 2. We have created models for dog 1 and dog 2 by splitting the data set (interictal + preictal) into training and test sets and using our models on training set. We then predicted the test results for them again. Furthermore, we predicted each dog seizures by using other dog's logistic model. Table 2 indicates that the we got almost the same accuracy when we applied the models for the same dog as about 67%. We predicted dog 1 using dog 2 model and we got 50% accuracy which indicates a bad model. However, using dog 1 model on dog 2, we got the best accuracy as 76 %. Those variations could be resulted from the imbalance between interictal and preictal phases.

Table 2: Model accuracies from different models

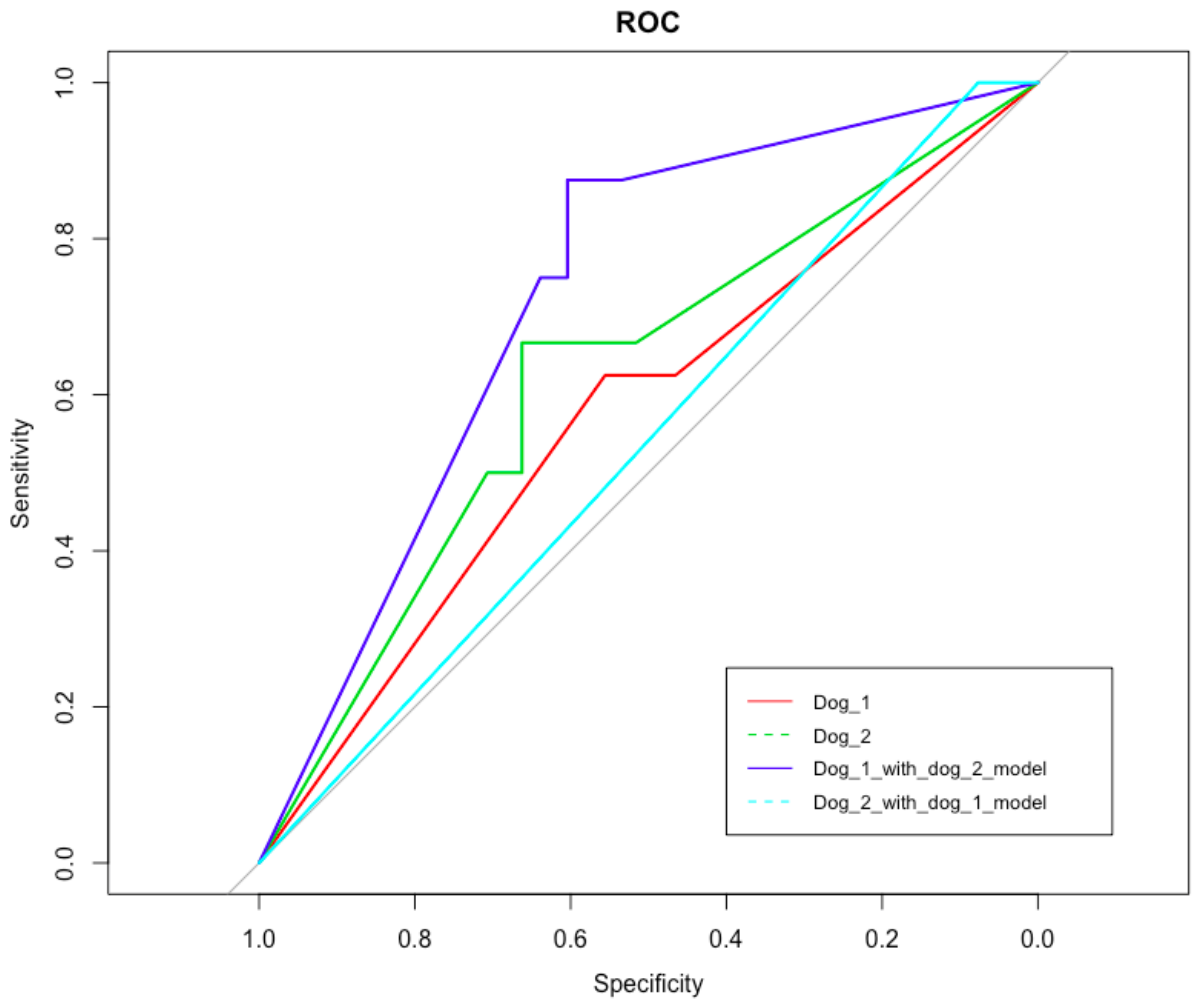
Dog1 accuracy from dog1 model	`> table(test$Label, test$predict.seizure>0.5)`
	
	`  FALSE TRUE`
	`0  98  48`
	`1  1  5`
	`> acc`
	`[1] 0.6776316`

Dog2 accuracy from dog2 model	`> table(test$V944, test$predict.seizure>0.5)`
	
	`  FALSE TRUE`
	`0  101  49`
	`1  5  8`
	`> acc`
	`[1] 0.6687117`

Dog1 accuracy from dog2 model	`> table(dog1.test$Label, dog1.test$predict.dog1.with.dog2.model>0.5)`
	
	`  FALSE TRUE`
	`0  73  74`
	`1  2  3`
	`> dog1.with.dog2.model.acc`
	`[1] 0.5`

Dog2 accuracy from dog1 model	`> table(dog2.test$V944, dog2.test$predict.dog2.with.dog1.model>0.5)`
	
	`  FALSE TRUE`
	`0  122  31`
	`1  8  2`
	`> dog2.with.dog1.model.acc`
	`[1] 0.7607362`

Due to the strong imbalance between inter-ictal and preictal samples, resulting accuracies were not resulted in a good performance. Below figure shows the receiver operating characteristics (ROC) curve and the area under ROC curve (AUROC) can be considered as the main metric to evaluate the performance. ROC curve shows the tradeoff between FPR and TPR for different threshold values. Having a higher AUROC indicates better performance which was established by estimating dog 2 with dog 1 model.



Recursive feature selection overfits because of the size of the data set, as mentioned. Therefore, in this experiment we identified the most informative features by sorting the features by absolute value of their corresponding weight learned with logistic regression. The results are reported in Table 3.

Table 3: Top ten features ranked by weight absolute value in logistic regression

Rank	Feature description
1	Maximum entropy from all channels
2	Norm. energy in ch8 band h2; 4) Hz
3	Norm. energy in ch13 band h12; 18) Hz
4	Norm. energy in ch12 band h18; 22) Hz
5	Time domain correlation feature
6	Norm. energy in ch0 band h12; 18) Hz
7	RMS of Wavelet ch3 D4_coefficients
8	Norm. energy in ch14 band h0:1; 2) Hz
9	Norm. energy in ch11 band h2; 4) Hz
10	Norm. energy in ch10 band h18; 22) Hz

5. Discussion and Conclusion

In this work we studied epileptic seizure prediction from iEEG recordings as competitors in American Epilepsy Society Seizure Prediction Challenge. Or findings show that it is a difficult task, made harder by the challenge design choices. Mostly it is the unbalanced labelled data set with not enough positive examples for proper training. Additionally, from the nature of the data and there is always the natural variance – epilepsy caused by brain tissue in various locations.

Further, a problem of this challenge from scientific standpoint is lack of leave-one-out validation. All the test data originated from the same individual as train data, hence bold claim about generalization can't be made. Moreover, these test data (without labels) were available. Author of the highest scoring solution 3 (0.8399 AUC) admitted that he managed to cluster some of the test data, identifying segments likely coming from the same 1h window before a seizure. Thus, effectively reducing the problem from classification of 10-minute segments to classification of~1h segments. Therefore, there is still a long way towards a reliable online seizure forecasting.

Due to the low processing power, we only made use of Dog1 and Dog2 dataset for this project. Since the test data has no label, we made use of the provided training data. This was also our first experience in EEG and time-series analysis, making cleaning of the data and extraction of features inorganic compared to our existing knowledge. Upon successful utilization of calculating and understanding spectral power, our deadline was near. or both training and testing. The model was built with Dog1 and tested on Dog2. Initially we tried using same data from same Dog for both testing and training but this yielded very low accuracy possibly Mostly due to the unbalanced labelled data set with not enough positive examples for proper training. Using Dog1 to training and testing on Dog2 yield a much higher accuracy above 70% compared to the initial which is consistently between 50%-60%.

This was also our first experience in EEG and time-series analysis, making cleaning of the data and extraction of features inorganic compared to our existing knowledge. Upon successful utilization of calculating and understanding spectral power, our deadline was near, in future, we would ideally love to use all the dataset, let's say being able to train with Dog1, Dog2, Dog3 and testing on 4 & 5. Or a much better suggestion is running a deep learning model with all the dataset (without feature selection), thus with an GPU enabled machine for accelerated training.

6. Appendix

R code will be supplied as an .R file. MATLAB codes for feature extraction attached below.

```
function E = entropy_bin(x,width)

bins = min(x):width:max(x);
bins = bins(1:end-1);
bin_width = diff(bins);
[p,~] = histcounts(x,bins,'normalization','probability');
bin_width = bin_width(p>0);
p = p(p>0);
E = -sum(p.*( log2(p) - log2(bin_width)));
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function eng_bands = getBandsEng(spectrums,feq,bands)

n_bands = size(bands,1);
n_feq = size(feq,1);
[n_seq,h,~] = size(spectrums);
idx_bands = cell(1,n_bands);

for i = 1:n_feq
    for j = 1:n_bands
        if bands(j,1) <= feq(i) && feq(i) < bands(j,2)
            idx_bands{j} = [idx_bands{j},i];
        end
    end
end
eng_bands = zeros(n_seq,n_bands);
for i = 1:n_seq
    for j = 1:n_bands
        eng = 0;
        for k = 1:h
            win = squeeze(spectrums(i,k,:));
            start_idx = idx_bands{j}(1);
            end_idx = idx_bands{j}(end);
            eng = eng + sum(win(start_idx:end_idx));
        end
        eng_bands(i,j,:) = eng;
    end
end
eng_bands = normalization(eng_bands,1);
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function feat = getCorrFeat(data)

R = corrcoef(data);
e = eig(R);
R_fliplr = fliplr(R);
ur = triu(R_fliplr);
ur = ur - diag(diag(ur));
corr_coefs = ur(ur~=0);
feat = [e;corr_coefs]';
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [spectrums, feq, spec_ent] =
getSpectrums(data,window_size,overlap_size)

[n_ch,n_seg] = size(data);
n_windows = ceil(n_seg/(window_size - overlap_size));
window_scale = kaiser(window_size,2.4)';
spec_ent = zeros(1,n_ch);
feq_cap = 198;
spectrums = zeros(n_ch,n_windows-2,feq_cap);
feq = zeros();

for i = 1:n_ch
    idx = 1;
    ent = 0;

    for start_idx = 1:window_size-overlap_size:n_seg
        end_idx = min(n_seg,start_idx+window_size)-1;
        if end_idx - start_idx < window_size-1
            break
        end
        win = data(i,start_idx:end_idx).* window_scale;
        w = triang(801); w= w(1:800);
        [pxx,f] = pwelch(win,w,400,800,400);
        s = log(pxx(1:feq_cap));
        ent = ent + entropy_bin(s,0.1);
        th = prctile(s,75);
        s(s<=th) = 0 ;
        spectrums(i,idx,:) = s;
        feq = f(1:feq_cap);
        idx = idx+1;
    end
    spec_ent(i) = ent/idx;
end
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function feat = getSpecFeat(data,window_size,overlap_size)

[spectrums, feq, spec_ent] = getSpectrums(data,window_size,overlap_size);
% spectrums: 16x198x198
% feq: 1x198
% spec_ent: 1x16

n_ch = size(spectrums,1);
for i = 1:n_ch
    tmp_coeff = squeeze(spectrums(i,:,:));
    spectrums(i,:,:) = normalization(tmp_coeff,2);
end

bands = [0.1,2; 2,4; 4,8; 8,12; 12,18; 18,22; 22,30; 30,50; 50,200];
eng_bands = getBandsEng(spectrums,feq,bands);
corr = getCorrFeat(eng_bands');
eng_flatten = reshape(eng_bands.',1,[]);
feat = [eng_flatten,corr,mean(spec_ent),std(spec_ent)];
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function feat = getTimeFeat(data)
% extract time domain feature from original data
% Input:
%   data: matrix in the 'data' field of the original struct
% Output:
%   feat: 1-by-N feature vector

n = size(data,1);
E = zeros(n,1);

for i = 1:n
    E(i) = entropy_bin(data(i,:),1);
end

if size(data,2) > 400
    resampled = interpft(data',400);
    resampled = resampled';
else
    resampled = data;
end

resampled = normalization(resampled,1);
corr_feat = getCorrFeat(resampled');
feat = [corr_feat, max(E),min(E),mean(E)];
```

```
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function feat = getWaveletFeat(data,window_size,overlap_size)

[w_coeff,w_entropy] = getWavelets(data,window_size,overlap_size);
% w_coeff: 16x198x24
% wentropy: 1x16


[n,h,w] = size(w_coeff);
for i = 1:n
    tmp_coeff = squeeze(w_coeff(i,:,:));
    w_coeff(i,:,:) = normalization(tmp_coeff,2);
end

bands = zeros(n,w);

for i = 1:n
    for j = 1:h
        tmp = squeeze(w_coeff(i,j,:));
        bands(i,:) = bands(i,:) + tmp';
    end
    bands(i,:) = bands(i,:)/n;
end
bands = normalization(bands,1);

corr = getCorrFeat(bands');

bands_flatten = reshape(bands.',1,[]);
feat = [bands_flatten,corr,mean(w_entropy),std(w_entropy)];
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [w_coeff,w_entropy] = getWavelets(data,window_size,overlap_size)

[n_ch,n_seg] = size(data);
n_windows = ceil(n_seg/(window_size - overlap_size));

window_scale = kaiser(window_size,2.4)';
n_f = 24;
w_coeff = zeros(n_ch,n_windows-2,n_f);
w_entropy = zeros(1,n_ch);
for i = 1:n_ch
    idx = 1;
    ent = 0;
```

```
    for start_idx = 1:window_size-overlap_size:n_seg
        end_idx = min(n_seg,start_idx+window_size)-1;
        if end_idx - start_idx < window_size-1
            break
        end
        win = data(i,start_idx:end_idx).* window_scale;
        [c,l] = wavedec(win,5,'db1'); % w_decomp
        l = cumsum([1,l(1:end-1)]);
        w_feat = zeros(1,n_f);
        e = zeros(1,6);
        for k = 1:6
            tmp = c(l(k):l(k+1)-1);
            w_feat(4*k-3:4*k) = [mean(tmp), mean(abs(tmp)),...
                                 mean(tmp.^2), std(tmp)];
            e(k) = entropy_bin(tmp,0.1);
        end
        ent = ent+mean(e);
        w_coeff(i,idx,:) = w_feat;
        idx = idx+1;
    end
    w_entropy(i) = ent/idx;
end

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function out = normalization(in,axis)
% zero mean and unit var

means = mean(in,axis);
stds = std(in,0,axis);

out = (in-means) ./ stds;
end



function parsave(save_name,feat,label)
    save(save_name,'feat','label')
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
function [feat,label] = extractFeatureDog(file_name)
% extract all features from dog data
% Input:
%   file_name: mat file name
% Output:
%   feat: feature vector with xxx elements
%   label: binary label 0 - interictal, 1 - preictal

mat_file = load(file_name);
var_name = fieldnames(mat_file); var_name = var_name{1};
label = double(contains(var_name,'preictal'));

seg = getfield(mat_file,var_name); %#ok<GFLD>
data = seg.data;

% entropy and correlation feature
time_domain_feat = getTimeFeat(data);

% wavelet and entropy feature
window_size = 2400;
overlap_size = 1200;
wavelet_feat = getWaveletFeat(data,window_size,overlap_size);

% spectral engery feature
spectral_feat = getSpecFeat(data,window_size,overlap_size);

% concatenate
feat = [time_domain_feat,wavelet_feat,spectral_feat];
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
clear;
fullFileName = which(mfilename); ptr = fileparts(fullFileName); cd(ptr);
addpath(genpath('../'))

dog_id = 'Dog_1';

% config directory
save_dir = fullfile('..','results',dog_id);
if exist(save_dir) ~= 7 %#ok<EXIST>
    mkdir(save_dir);
end

file_list = dir(fullfile('..','data',dog_id,[dog_id,'_*.mat']));
n_file = length(file_list);

% config parallel pool
p = gcp('nocreate');
if isempty(p)
    cluster = parcluster('local')
    numWoker = cluster.NumWorkers
    parpool('local',numWoker)
end

% run
tic
parfor i = 1:n_file
   file_name = file_list(i).name;
   [feat,label] = extractFeatureDog(file_name);

   [~,f,~] = fileparts(file_name)
   save_name = fullfile(save_dir,[f,'_feat.mat']);
   parsave(save_name,feat,label);
end
toc

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```