## Option 1: Combining Multiple Features in Classification

The option helps you to study (i) the effect of combining features in classification; and (ii) how to use SVM (utilizing common packages).

**Data Set**: The given dataset contains 50 categories/classes. The **training set** has 4786 samples in the file 'trainData.mat', and the **testing set** has 1833 samples in the file 'testData.mat'. Each sample is described by the rows of 3 different **feature** matrices, i.e. $\boldsymbol{X}_1$, $\boldsymbol{X}_2$, and $\boldsymbol{X}_3$ in the corresponding file, and the category vector is always $\boldsymbol{Y}$. All the 3 features are normalized histograms, which means the elements are non-negative and the sum of each feature equals to 1 (i.e., $\sum_j \boldsymbol{X}_k(i,j) \equiv 1$).

NOTE: A '*.mat' file is a Matlab file which can be read into your program by the command *load*.

**Step 0:** Classification by individual features.

**Output**: The classification accuracy for the testing set in the follow cases (1) and (2).

**Instructions**:

(1) For each of the 3 features in the training set, $\boldsymbol{X}_k$ $(1 \le k \le 3)$, train a multi-class linear SVM classifier, i.e. $h_k(\mathbf{x})$. Get the prediction result of $h_k(\mathbf{x})$ based on the same feature $\boldsymbol{X}_k$ in the testing set, and compare to $\boldsymbol{Y}$ for the classification accuracy.

*NOTE*:

(i) There are many SVM toolboxes. We recommend the libSVM, which can be downloaded in: http://www.csie.ntu.edu.tw/~cjlin/libsvm/ For Matlab users, you can directly use the code in folder 'libSVM\'. Run 'make.m' to install on your PC. The instructions and examples on how to use the package can be found in the file 'README0'. We can use the function *svmtrain* for training, and *svmpredict* for prediction.

(ii) You may fix the penalty parameter '-c 10' and linear kernel '-t 0' for *svmtrain.*

(2) Based on the SVM classifiers $h_k(\mathbf{x})$, we can also estimate $p_k(w_i|\mathbf{x})$ as the posterior probability of sample $\mathbf{x}$ belongs to the $i$-th category ($w_i$) by feature $\boldsymbol{X}_k$ $(1 \le k \le 3)$. Train the SVM classifiers and report the classification accuracy in the testing set based on the 3 features.

*NOTE:*

(i) This is also implemented by libsvm if we use the parameter '-b 1' for *svmtrain* (See README0).

(ii) You may fix the penalty parameter '-c 10' and linear kernel '-t 0' for *svmtrain.*

**Step 1**: Feature combination by fusion of classifiers.

**Output**: The classification accuracy in the testing set and compare it to that of (2) in Step 0.

**Instructions**: Directly combine the 3 SVM classifiers with probability output i.e. $p_k(w_i|\mathbf{x})$ $(1 \le k \le 3)$, in (2) of Step 0. Combine the 3 classifiers by probability fusion as $p(w_i|\mathbf{x}) = \sum_k p_k(w_i|\mathbf{x})/3$. The final recognition result is $w_{i*} = \text{argmax}_i\, p(w_i|\mathbf{x})$.

**Step 2**: Feature combination by linear cascade.

**Output**: The classification accuracy in the testing set and compare it to that of (1) in Step 0.

**Instructions**: Directly cascade the 3 histogram features $\mathbf{X}_k, 1 \le k \le 3$ into a single feature, i.e. $\mathbf{X} = [\mathbf{X}_1\ ,\ \cdots, \mathbf{X}_K]$; train a linear SVM classifier based on $\mathbf{X}$ and get the classification accuracy for the testing set.

*NOTE*: You may fix the penalty parameter '-c 10' and linear kernel '-t 0' for *svmtrain*.

**Step 3**: Feature combination by non-linear kernel fusion.

**Output**:

(1) The classification accuracy of the 3 chi-square kernels and compare it to that of (1) in Step 0.

(2) The classification accuracy of $\mathbf{K}_a$ and $\mathbf{K}_b$, then compare them to that of (1) in Step 3.

**Instructions**:

(1) (Chi-Square kernel classification) Derive Chi-Square **kernel matrix** $\mathbf{K}_k$ from the histogram feature of $\mathbf{X}_k, 1 \le k \le K$. Then train 3 kernel SVMs from the $\mathbf{K}_k$ and get their accuracy.
*NOTE*:

(i) The Chi-Square kernel is a special non-linear kernel defined for histogram feature as below

$$k(\mathbf{x},\mathbf{y}) = 1 - \sum_{i=1}^{n} \frac{(x_i - y_i)^2}{\frac{1}{2}(x_i + y_i)}$$

Then the kernel matrix is defined by $\mathbf{K}_k(i,j) = k(\mathbf{X}_k(i,:), \mathbf{X}_k(j,:))$.

(ii) The usage of *svmtrain* with a kernel matrix is specified in the Example section of 'README'.
(iii) You may fix the penalty parameter '-c 10' for *svmtrain*.

(2) (Kernel Fusion) Use the following two schemes to fuse the 3 kernels into 1 kernel for recognition, and report the testing accuracies:

(i) $\mathbf{K}_a = 3^{-1} \cdot \sum_k \mathbf{K}_k$ ; (ii) $\mathbf{K}_b = (\prod_k \mathbf{K}_k)^{1/K}$ .

**Some general requirements for all options:**

1. In your project report, please include a section that specifies the respective contributions of each individual group member.
2. While your report should briefly summarize what you did, you do not need to repeat the detailed procedures/equations etc. that are already described in the project descriptions (or papers, for the options based on given papers).
3. Your report should clearly document all major outcomes/results of your project, e.g., plots, error rates, sample images, etc. While the graders may run your code to verify your work, they are not supposed to run your code to obtain the results for grading. (Therefore, you should never say things like "To get the required plots, please run MyCode1.m"; Instead, just include the plots in your report.)
4. Your report may discuss anything interesting you found in the process of doing your project. This is the opportunity for showing your understanding of the problem beyond mere implementation of some given algorithms.
5. Your report should also serve as a manual for running your code to verify your results.
6. Do NOT embed any code in your report. Submit your report, your programing files or data files if any as a single zip file.