System for Large Scale Spam Classification

1 Overall Design and Usage

At time t=0

Stage - 1

Take (k+2) splits in a folder named "trainData".

Use 2 out of the (k+2) splits for training and tuning to get the hyperparameters ($h_1 h_2 ... h_n$). This will be done via the UI (Refer section 2).

(On Hadoop) Use the above hypeparameters to train k models from the remaining k splits. This can be done by running the command-
hadoop jar mr_QH.jar filepaths1 outmr
where filepaths1 contains the paths to all the k training files. outmr is an output folder currently not of any use. This folder needs to be removed before running the command.
These models will be written to a file "weightsPerModel.txt". These models will be used to convert a set of 2 arff files of n features to two arff files of k features. This is included in this command.

Stage - 2

Use the above 2 arff files for again training and tuning to get parameters of size k via the UI.

k parameter vectors from stage 1 and 1 parameter vector from stage 2 will be combined to get a final parameter vector (W of size n). This can be done by running the following commmand.

java -jar finalModelGenerator.jar <featureWeightsPerModelFilePathStage1> <modelWeightsFilePathStage2> <outputModelFilePath> <outputFinalParamsFilepath> <Numfeatures> <k>

This final W will be used at any time t for classification by using the following command:
java -jar ClassifydataFromModel.jar modelFile hadoop/trainData/trainset-2-exe-split6.arff

---

At time t=t1

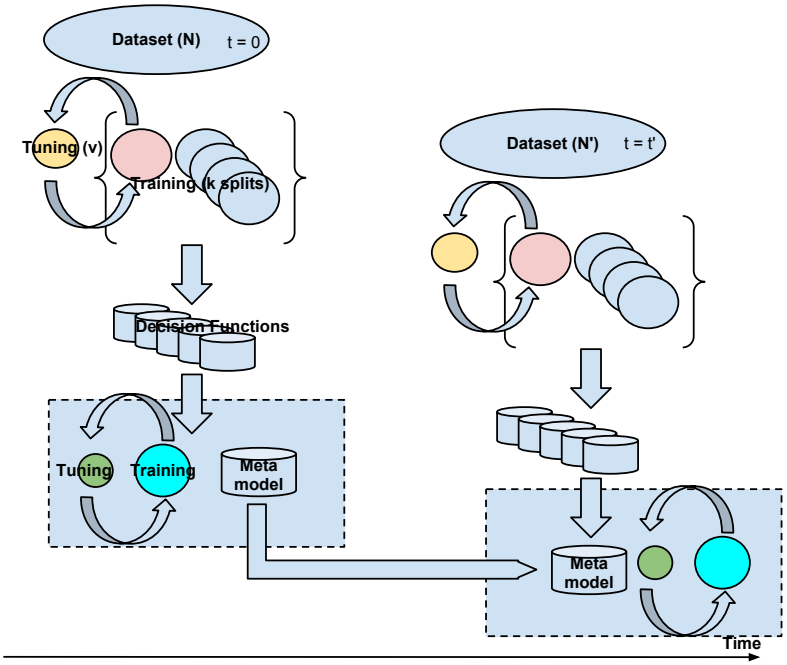

Figure 1: System Design

Similar to t=0.
Only difference will be at 2nd stage where 1 model which has been learnt till now(time t1 -1) will also be used. Add the params to the file "weightsPerModel.txt". obtained from Stage1 of time t1.

2 Interactive Model Tuning

2.1 Steps to configure and deploy the application

1. Extract meta.war
2. Open WEB-INF/web.xml and look for ⟨param-name⟩root_path⟨/param-name⟩. Change the corresponding param value to a valid path in the file system.
3. Repackage the meta.war and deploy it in a web container. We have tested it with tomcat7

OR

1. Deploy meta.war in a web container. We tested with tomcat7. In tomcat7 the applcation is deployed in webapps/.

2. Browse to webapps/meta/WEB-INF/. Open web.xml and look for ⟨param-name⟩root_path⟨/param-name⟩. Change the corresponding param value to a valid path in the file system.

3. Restart the web container.

2.2 Steps to use the application

1. Once deployed, the application is available at http://⟨host_IP⟩:⟨port⟩/meta/confusion.jsf

2. Use the upload files option to upload the two sample arff files available in the 'files' folder (in the distribution). Alternatively, you could manually copy the arff files to the ⟨root_path⟩/meta folder and provide the list of these files in a file called 'files' in the same folder (A sample 'files' file is provided in the 'files' folder of the distribution). You might have to restart the application server for the application to pick up these files.

3. These files should now be available in the "Train file" and "Holdout file" dropdown. Select "10K-train-random.arff" as the train file and "10K-test-random.arff" as the holdout file. Click on Update.

4. This will invoke model training and the confusion matrix will be updated.

5. Change the FPs from 70 to 65 and click on Submit. This will invoke the underlying iterative training. Observe as the confusion matrix and the chart get updated if the user intention is met.

6. You could play around with the FP count. In the case when the user intention cannot be met, a message is shown to intimate this to the user. When the user intention is met, the optimized model and its parameters would be serialized to ⟨root_path⟩/model/optim_⟨timestamp⟩.model and ⟨root_path⟩/model/optim_⟨timestamp⟩.params. The hyperparameters of the tuned model are written to ⟨root_path⟩/optimized.dat. They are available between the @begin_model and @end_model containers in this file.