Programming Assignment # 3
K-fold Cross-Validation

This assignment will help you better understand how to use K-fold cross-validation in order to find the best parameters for regularization (although the same approach might be used for any other parameter that a learning algorithm requires), as well as how to compute an estimate of the expected accuracy of your learner. In this assignment, you will estimate the accuracy of the learned classifier (which includes this learned parameter) in 3 ways : (i) directly from the training set, (ii) using an incorrect implementation of cross-validation, and (iii) using a correct implementation of cross-validation. Finally, you will get to compare these estimated accuracies with a better estimate : the accuracy on the hold out set that was not used in any part of the training process.

You will need the file `ex3_crossVal.m` in this part of the assignment. This file loads the train set and hold out set, as well as the cross-validation folds, therefore, you won't need to generate those.

1 Creating a classifier

1.1 Complete function : `returnBestLambda.m`

We will first train a classifier to predict labels for new samples. In this part of the assignment you will use 5-fold cross-validation to select the best value for the regularization parameter λ^* (The algorithm is shown in Figure 1). For this exercise you will try the following values :

$$\lambda = [0, 0.01, 0.03, 0.1, 0.3, 1, 3, 10, 30]$$

For simplicity, we have included a function named `logisticRegression` that will return an optimized vector θ_{opt} given a training set, and a regularization parameter λ. This function employs the functions that you completed in the previous part of the assignment.

Now you should complete the file `returnBestLambda.m`, which defined a function of that name. This function receives three parameters as input : a training

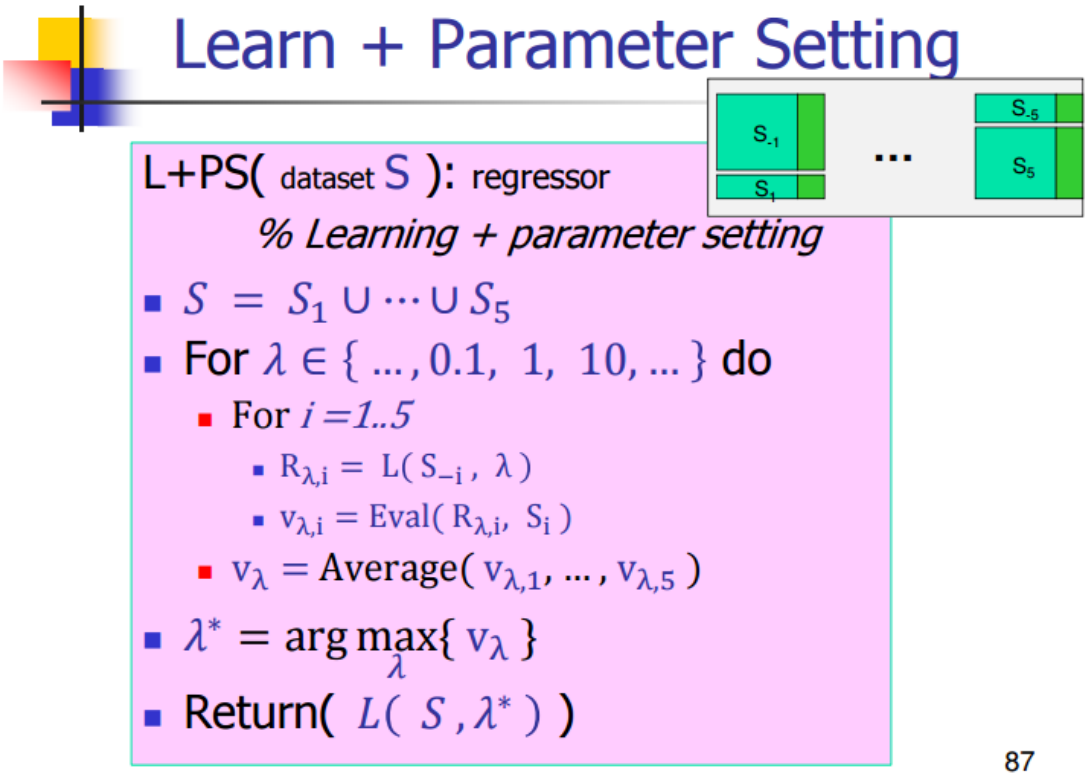

FIGURE 1 – Algorithm to find the best value of the regularization parameter λ among the different values tested.
[Slide 87 of the file 2b-linearRegression.pdf on eClass]

set (X), the labels of the training set (Y), and a matrix (let's call it "folds") of dimensions $m \times 2$, where m is the number of training samples. The output is the value λ_{best} (among the values tested) that yielded the highest accuracy in the 5-fold cross-validation process.

The matrix "folds" holds information regarding the arrangement of samples for performing cross-validation (to create the sets S_{-i}, S_i). The first column of "folds" contains a random permutation of the X indices (i.e., index of a row of X that represents one instance in the dataset). The second column specifies the fold to which this sample is assigned. For example, if the first entry of the matrix is $[20, 3]$ it means that the row 20 of X belongs to the 3^{rd} fold.

You can use the function `generateSets.m` to extract from X the training and test folds for cross-validation as well as the respective labels. This function gets as input 4 variables : X, Y, *folds*, and *iterNum*, where X is the part of data on which we want to perform cross-validation, Y is a vector of the respective labels, folds

is that same as described in the previous paragraph, and *iterNum* specifies the fold which is going to be assigned as the test fold (and of course, the rest of the folds would be assigned as training).

1.2 Complete function : `learningAlgorithm.m`

Now that you used 5-fold cross-validation to define the best value of λ_{all} (among the values that you tried), you can use this value along with the entire training set to compute the vector θ_{all} that you will use to perform classification.

Now you can complete the file : `learningAlgorithm.m`. This function is equivalent to the algorithm shown in Figure 1 and to the block "L + PS" (Learning + Parameter Setting) from the Figure 2.

This is your final classifier, the one that you will use to classify new instances (here, the hold out set). The rest of this assignment explores ways to **find a good estimate of the accuracy of this classifier on new data, i.e., the quality of the learning algorithm.**

2 Estimating the quality of the learning algorithm

Now that we have our classifier, we can estimate the accuracy of our learning algorithm. We will try three different methods (two of them are incorrect) and then compare those three results with a more reliable estimate of the accuracy on the classifier, based on its accuracy on a hold out set, that was not used in any stage of the learning phase.

2.1 Accuracy on the train set (Incorrect method)

In this part of the exercise, you will complete the function `computeAccuracy_TrainSet.m`. You will first find a classifier on the entire training set (as you just did in the previous section), and then evaluate the accuracy on that same training set. Note that this method for estimating the error is incorrect, and you should know that it is not a measure of the quality of your classifier. (However, it can sometimes be useful, as a bad performance on the training set usually means that your model is not expressive enough, or that your features are not discriminative enough. On the other side, if this accuracy is high, but the performance on new instances is low, it is a sign of overfitting.)

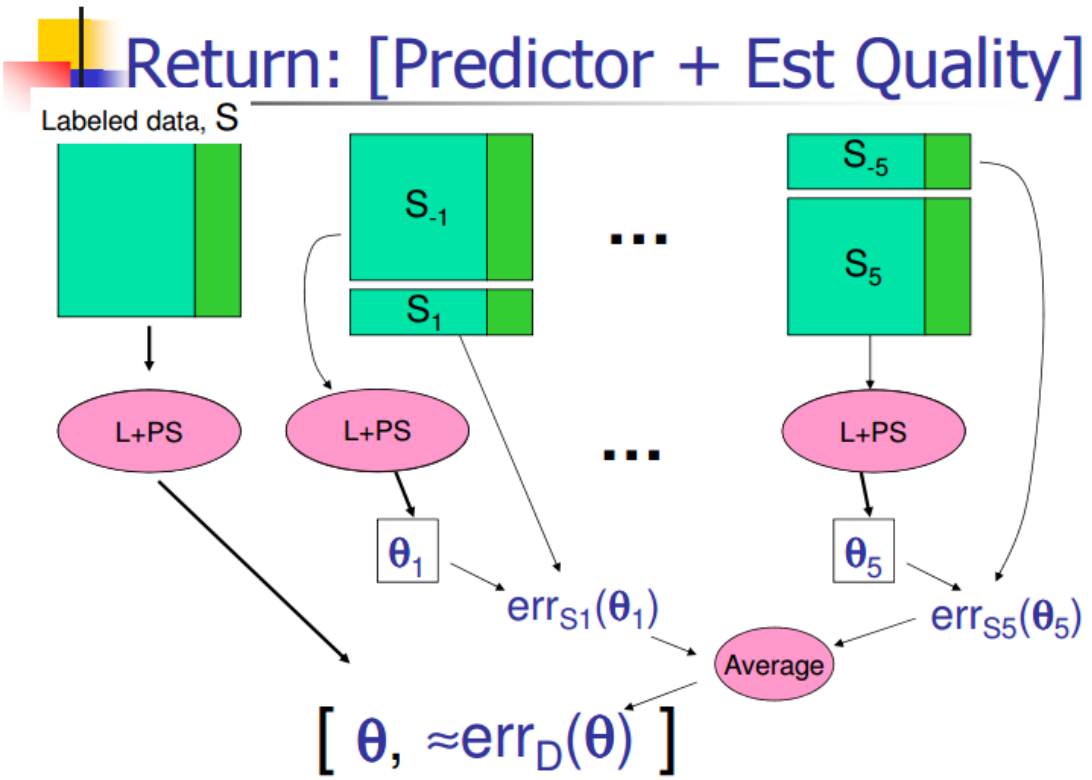

FIGURE 2 – General diagram for creating a classifier and estimating the accuracy of the L + PS learning algorithm (on dataset S) using 5-fold cross-validation. [Slide 83 of the file 2b-linearRegression.pdf on eClass]

2.2 Accuracy using cross-validation, using a single lambda (Incorrect method)

In this part of the exercise, you will complete the function `computeAccuracy_Lambda.m`. You will first find the best regularization parameter λ_{best} using the entire training set (i.e., the one you found on the previous section). Then, in each fold, find θ_i for the training partition (corresponding to S_{-i} in Figure 2) using λ_{best} (Note that in this part of the exercise, you are NOT using the algorithm L + PS to generate θ_i, since that algorithm would find its own λ_i, and you want to use λ_{best}), and then find the accuracy on the respective test partition (corresponding to S_i in Figure 2). Repeat this procedure for all 5 folds. Finally, output the average accuracy over the 5 folds.

2.3 Accuracy using cross-validation. (Correct method)

In this part of the exercise, you will implement the algorithm suggested by Figure 2, where the same L + PS is used (to find $lambda$ and the classifier), both based on the entire dataset, and also for each fold. We will perform an internal cross-validation whithin each fold to determine the parametrs of the model, namely λ_i which in turn gives us θ_i. Start with the 5-fold cross-validation that you used throughout this assignment. For each fold S_{-i} (i.e., 80% of the Xtrain), perform an internal 5-fold (or let us call it subfold (i.e, $80\% \times 80\% = 64\%$ of Xtrain)) cross-validation to find the λ_i that is best whithin this fold. With that λ_i you can find θ_i in the respective fold. Note that this internal cross-validation process is already implemented inside our function `learningAlgorithm`, which will return θ_i. Then, using θ_i, compute the accuracy on S_i to obtain $err_{S_i(\theta_i)}$. Repeat for all external 5 folds and report the average accuracy.

The difference with respect to the last section, is that now you will select λ_{best} based only on the training set inside every fold, and you will compute the accuracy based on the temporal test set inside every fold. This temporal test set was NOT used for making any decision. After averaging the accuracies over the 5 folds, we expect to have a good estimate of the quality of our learning algorithm.

You should now complete the file :`computeAccuracy_CV.m`