Assignment 2

1. Introduction

In this assignment, the `spam` data set is used, containing the frequencies for different words in emails. This information will be used to train a support vector machine (SVM) models to predict these emails as spam or non-spam (binary classification). An SVM is a sparse (i.e. uses only a subset of the points to build the model) method that uses a margin-maximizing approach to solve both classification and regression tasks. The assignment focuses more on the theory of machine learning than SVM's, leading us to explore why we use the holdout method to assess model performance.

In the first step, the data is split into some different sets:

- training
- validation
- training and validation
- test

2. Model Training

We then proceed to train a number of models using a range of values for `C`, a cost parameter. At this stage, training is done using the training dataset and errors are calculated using the validation set. This is a simple grid search that produces an error statistic for each of the values tried for `C`, from 0.3 to 4.8. We then select the `C` which gives the lowest error.

Optimal `C` selected:

```
## [1] 1.2
```

We then use this value of C to refit the model and calculate errors in different configurations:

- Case 0: train using **training** data and calculate errors for **validation** set
- Case 1: train using **training** data and calculate errors for **test** set
- Case 2: train using **training+validation** data and calculate errors for **test** set
- Case 3: train using all data and calculate errors using **test** set

Remember, all configurations uses the same value for the cost parameter, which has been estimated from the **validation** set.

The different error rates for the filters:

```
## err0
```

```
## [1] 0.07
```

```
## err1
## [1] 0.08489388

## err2

## [1] 0.08364544

## err3

## [1] 0.03370787
```

3. Questions and Answers

1. Which filter do we return to the user? filter0, filter1, filter2 or filter3? Why?

We return `filter3` as we are now done with model estimation and are ready to use the filter. At this stage we use all available data to finalize our model. We only use the holdout method with different data sets to tune the model and to verify that the method of fitting a model to data is suitable - SVM in this case - or perform model selection if there are many different models. Please see motivation below for further explanation.

2. What is the estimate of the generalization error of the filter returned? err0, err1, err2 or err3? Why?

The best estimate of the generalization error would be that of `err1` (0.085).

Motivation:

Let us examine the different scenarios:

In all four cases, the models are fitted with a hyperparameter `C` of 1.2, which has been determined using training and validation data, i.e. the value for `C` that minimizes the error of the validation set.

The hyperparameter `C` is a user-defined parameter that controls regularization. Larger values of `C` adjusts the slack, that is it penalises the model for points that are misclassified or in the margin more. The effect of this, for large values of `C`, is that the model will attempt to fit every point despite that fit not being generalizable, i.e. overfitting occurs. On the other end of the spectrum a low `C` generalizes the model well as misclassified (or near-misclassified) points do not incur that much extra loss. This smoothens the model a lot. As we can see, `C` is directly related to the regularization and therefore the generalization error of the model. This parameter has been selected using the validation set, excellent!

As we arrive at the optimal cost hyperparameter using the validation dataset, using the same set to test the model performance (as done with `err0`) would underestimate the generalization error. Why? We would then be using the same set to both tune hyperparameters (e.g. decide cost parameter) and test the model, creating a leak. This should exclude `err0` from consideration for this purpose.

Why not use the error calculated from `filter3`? Well, now we are again using the test data itself to fit the model, which would introduce a huge leakage if we were to assess the model performance using the error it gives. Indeed, the test error of this filter is much lower than the other filters' and close to the error of the training set.

Why then would we not pick `err2`, or do we expect the best approximation of the generalization error to be that of `err1`? For `err2` we are again risking some bleeding when using the same set multiple times (testing on a model trained on training and validation sets and optimized for the validation set). Utilizing three isolated sets and only using them for their specific purposes and then discarding them would avoid the possibility of data leakage. This is exactly what we are doing to calculate `err1`. The generalization error obviously does not change depending on what data we use to train the final model, leading us to pick the *theoretically* best approximation of the error, i.e. `err1`, but the filter that uses all data available to us, `filter3`.

In short, the training set lets us train a model, the validation set lets us configure this model in an optimal way and the test set lets us see how well the method of training the model generalizes. The most important lesson here, and what machine learning is all about, is that for actual finalization and deployment of a model, none of our sets are important. We are doing all of this work just to figure out what method using which settings for the hyperparameters are appropriate to model the type of data we will feed it - is the machine able to learn from future data of the same kind? If our model fitting approach seems to work well, we apply the method to our data.