HOMEWORK - 2

SSI PATTERN RECOGNITION

April 23, 2019

Submitted to
Desire SIDIBE

ASRAF ALI Abdul Salam Rasmi
Masters in Computer Vision
Centre Universitaire Condorcet
Universite de Bourgogne

Linear Classification

1 SPAM OR NOT SPAM

We want to build a classifier to filter spam emails. We will use a dataset which contains a training set and a test set of, respectively, 3065 and 1536 emails. The data is given is the folder $SpamData$. Each email has been processed and a set of 57 features were extracted as follows:

- 48 features, in [0, 100], giving the percentage of words in a given message which match a given word on a predefined list (called vocabulary). The list contains words such as "business", "free", "George", etc.
- 6 features, in [0, 100], giving the percentage of characters in the email that match a given character on the list. The characters are ; ([! $ #.
- Feature 55: the average length of an uninterrupted sequence of capital letters.
- Feature 56: the length of the longest uninterrupted sequence of capital letters.
- Feature 57: the sum of the lengths of uninterrupted sequence of capital letters.

1.1 The max and mean of the average length of uninterrupted sequences of capital letters in the training set.

A function $getMaxMean$ has been implemented to compute Max and Mean of a given Row Vector. The output is shown below.

```
>>
The Max of the Average length of Uninterrupted Sequences of Capital letters in the Training set is 1102.5
The Mean of the Average length of Uninterrupted Sequences of Capital letters in the Training set is 4.9006
>>
```

Figure 1: $getMaxMean$

1.2 The max and mean of the lengths of the longest uninterrupted sequences of capital letters in the training set.

```
>>
The Max of the lengths of Longest Uninterrupted Sequences of Capital letters in the Training set is 9989
The Mean of the lengths of Longest Uninterrupted Sequences of Capital letters in the Training set is 52.675
>>
```

Figure 2: $getMaxMean$

1.3 Preprocessing

Before training a classifier, we can apply several preprocessing methods to this data. We will try the following ones:

1. Standardize the columns so they all have zero mean and unit variance.
2. Transform the features using $log(x_{ij}+0.1)$, i.e. add 0.1 to each feature for every example and take the log. We add a small number to avoid taking log of zero!
3. Binarize the features using $I(x_{ij} > 0)$, i.e. make every feature vector a binary vector.

All the above mentioned preprocessing techniques are implemented in the function $preProcess$.

2 LOGISTIC REGRESSION WITH REGULARIZATION

For each version of data, we will now try to fit a Logistic Regression model. Here we tried to implement Regularized Logistic Regression, so that we need to compute the Regularization Parameter λ. We tried to compute λ using **Cross Validation**.

2.1 Regularization Parameter

A function $getLambda$ has been implemented to compute the Regularization Parameter λ using Cross Validation. The algorithm of the function is as follows.

1. Divide the Training features into Training(80%) and Validation(20%) set.
2. Initialize λ with random values between [0,0.1]. Lambda value should be as small as possible because higher values of λ will cause Underfitting.

3. For each value of λ,
    - Compute the Regression Parameter.
    - Predict the Output Label for both Training and Validation set.
    - Compute the Prediction error for both Training and Validation set.
4. The value of λ for which the validation error is minimum is taken for building the classifier.

2.1.1 Results

The computed λ value for different preprocessed input features is shown in the following graphs along with Training and Validation Errors.

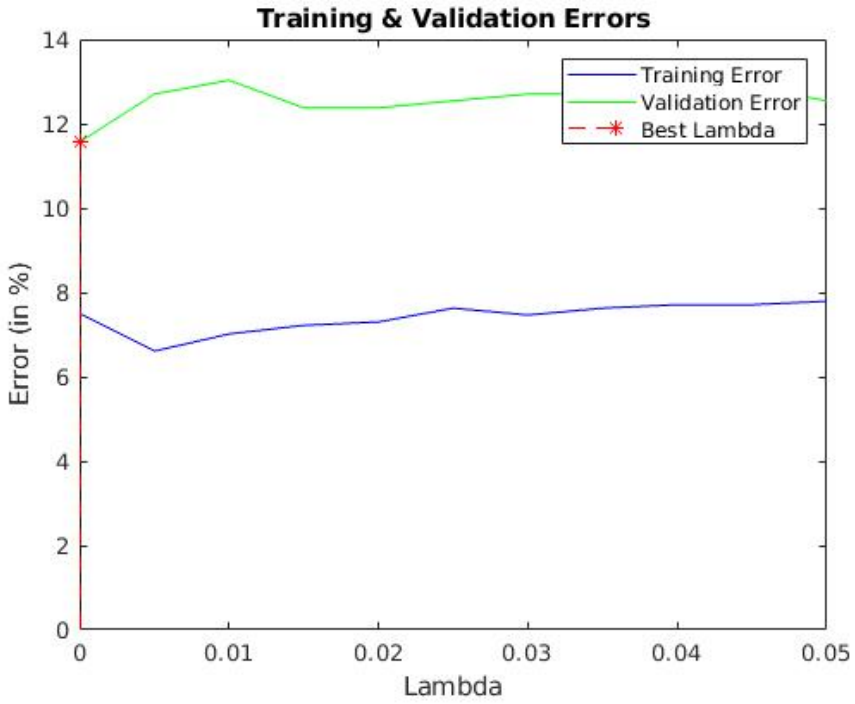

Figure 3: λ for Standardized Input features

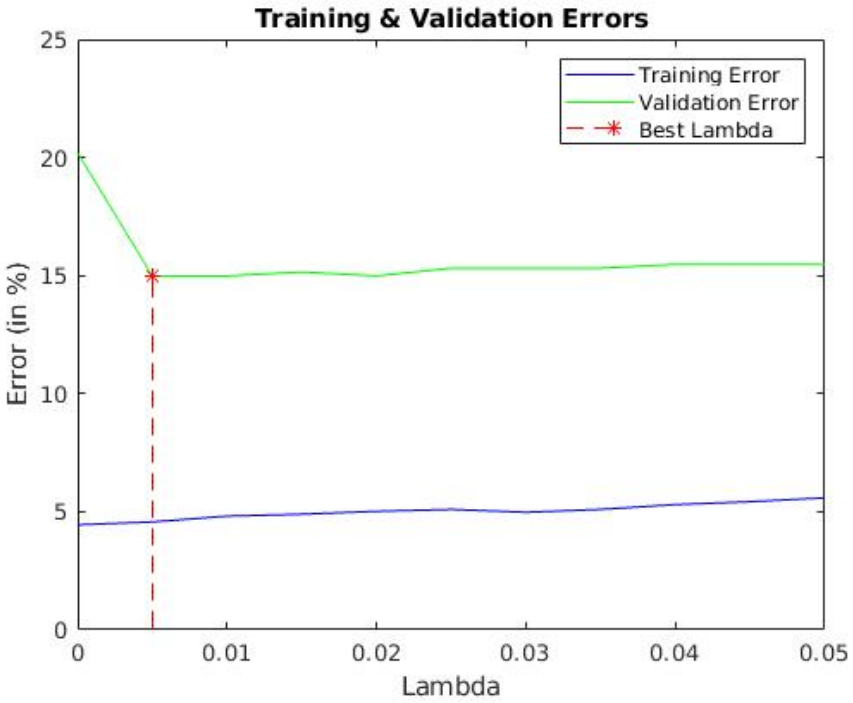

Figure 4: λ for Log transformed Input features

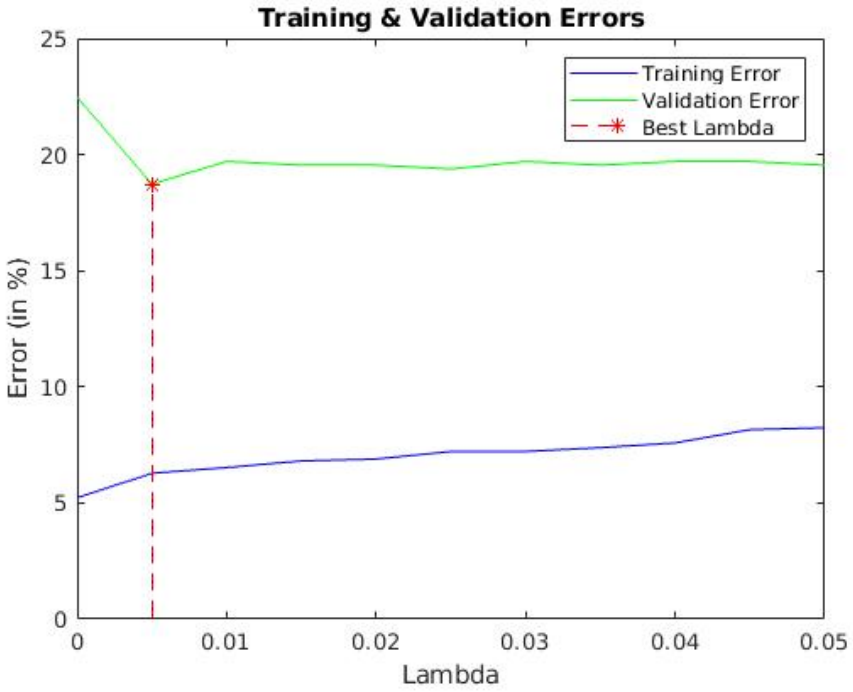

Figure 5: λ for Binarized Input features

2.2 Regression Parameter

Logistic Regression uses a **prediction function**[1] which is given by,

$$y(x) = \sigma(w^T \phi(x))$$

where ϕ is the basis function and σ is the **Sigmoid function**[1] which is given by,

$$\sigma(z) = \frac{1}{(1+e^{-z})}$$

To compute the optimal value for the Regression parameter w, we can use Gradient Descent[1]. Gradient descent to find w can be implemented as follows.

$$w \leftarrow w - \alpha \nabla E(w)$$

where α is the learning rate and $\nabla E(w)$ is the Gradient function.

The function $regressionParameter$ is implemented to compute the optimal value for w. **Gradient descent** is performed by MATLAB built-in function $fminunc$. In order to perform optimization we need to compute a **Cost Function** [1] (Cross Entropy function) which is given by,

$$E(w) = -log\ p(y|w) = -\sum_{n=1}^{N} \{y_n log(\phi_n) + (1-y_n) log(1-\phi_n)\}$$

with $\phi_n = p(C_1|\phi(x_n)) = y(\phi_n) = \sigma(w^T \phi(x_n))$. The **Gradient Function** is given by,

$$\nabla E(w) = \sum_{n=1}^{N} (y(\phi_n) - y_n)\phi_n$$

A function $costFunction$ is implemented which computes both cost and gradient functions for the given data.

2.3 Prediction Error

After getting the regression parameters from training data, we apply it on both training and test data to compute Prediction error. This is implemented with two functions, $predictRegress$ to predict the output label given Input features and Regression parameter(w), and $errorPredict$ to calculate the error.

Preprocessing	Train Error	Test Error
Standardization	8.2219	8.138
Log Transformation	5.1876	6.0547
Binarization	6.9821	7.5521

Table 1: Prediction Error for Logistic Regression

2.4 Best Preprocessing Strategy...?

The best preprocessing technique for the given data is **Log Transformation**, as we can see both the training and test errors are minimum. This is because in the given data the last 3 columns are having higher order values. As we know the log of a larger value will be a smaller value.

However, **Standardization** is the widely used technique which gives greater accuracy as explained in [2].

3 NAIVE BAYES CLASSIFIER

Naive Bayes classifier is implemented based on **Bayesian Decision rule**. Let us consider two classes C_1 & C_2. For a given input feature X, we will find the conditional probability of both classes given the input feature and the input feature will be assigned to the class with highest probability, i.e.

$$C_X = \begin{cases} C_1, & \text{if } P(C_1/X) > P(C_2/X) \\ C_2, & \text{if } P(C_1/X) \leq P(C_2/X) \end{cases}$$

And the Conditional probability is computed as,

$$P(C_i/X) \propto P(X/C_i)P(C_i)$$

where,

$$P(C_i) = \frac{No.\ of\ samples\ in\ C_i}{No.\ of\ samples} = \frac{N_i}{N}$$

and $P(X/C_i)$ can be given as a Normal distribution, i.e.

$$P(X/C_i) = \frac{1}{\sqrt{2\pi} \times \sigma} e^{\left(\frac{X-\mu}{\sigma}\right)^2}$$

Assuming the input features X as i.i.d.,

$$P(X/C_i) = \prod_{n=1}^{N} P(X_n/C_i)$$

A MATLAB function $naiveBayes$ has been implemented where for the given test features the function will return test labels taking into account the **Probability of Classes**, $P(C_1)$ & $P(C_2)$ computed from train features and train labels. A table showing the prediction error for Naive Bayes Classifier is shown below. We can see that Logistic Regression gives better accuracy than Naive Bayes for the given dataset.

	Implemented function		Built - in function	
Preprocessing	Train Error	Test Error	Train Error	Test Error
Standardization	17.6183	18.6198	17.5856	18.5547
Log Transformation	16.3458	18.099	16.3458	18.099
Binarization	39.739	38.737	-	-

Table 2: Prediction Error for Naive Bayes Classifier

Bibliography

[1] D. Sidibe, “Pattern recognition - lecture notes,” 2019.

[2] T. Hastie, R. Tibshirani, and J. Friedman, “The elements of statistical learning: data mining, inference, and prediction, springer series in statistics,” 2009.