University of Glasgow

School of Computing Science

MACHINE LEARNING COURSEWORK REPORT

REGRESSION AND CLASSFICATION

GROUP: M32

Akshata Bhat	2361611	2361611b@student.gla.ac.uk
Rasika Purohit	2393558	2393558p@student.gla.a.c.uk
Soumita Chel	2422493	2422493c@student.gla.ac.uk
Vinita Malu	2419202	2419202m@student.gla.ac.uk

Level M Course Work — 26th November 2018

Table of Contents

1. INTRODUCTION

Overview

Machine learning is improving computer efficiency by implementing algorithms [1]. It has two categories, namely supervised learning and unsupervised learning. Supervised learning is used to develop predictive models based on relevant data provided [1]. The models are built according to the format of data. Supervised learning has training data set, this data acts as input for the algorithm. The algorithm then attempts to predict future values on the basis of training data [2]. Supervised learning has inadequacy for categorization of data [2]. Supervised learning uses methods to achieve efficiency in predicting data using two approaches, regression and classification.

2. LINEAR REGRESSION

Overview

Regression is defined as "learning a continuous function from a set of examples" [6] e.g.: predicting stock prices. Linear modelling is the relationship between attributes and responses [7]. Models are built such that they can make accurate predictions on training set. The model is set to be accurate. The only measure whether an algorithm will work on data is evaluated on test set. There are three forms of models, generalized, where the model can predict dynamic data, overfitting is when the model fails to generalize new data and underfitting is the form of model which is unable to predict the existing data [7].

RandomForestRegressor is used to not limit the scope of predictions. It is basically, a farrago of multiple decision trees which are averaged to come to an almost accurate prediction.

Task 1: To predict performance of 41 unseen machines based on six features (Cache memory size, minimum and maximum number of I/O channels, machine cycle time, and minimum and maximum main memory) by constructing two algorithms and compare their efficiency [3].

a) Algorithm 1

Packages: numpy, pandas, matplotlib.pyplot, seaborn, sklearn.model_selection, sklearn.metrics, sklearn.ensemble, sklearn.preprocessing, sklearn.pipeline, sklearn.linear_model

Methodology:

1. Load the data:
   Create the dataframe using pandas library method read_csv()
2. Remove outliers by filtering data:
   Get the mean value(mu) and standard deviation(sigma) of dataframe using mean() and median() method respectively and set an efficiency parameter, k. The, using mean and median calculate the range of data. Drop the corresponding null values from both dataframe.

```
def remove_outliers(df, k=13):
    mu       = df.mean() # get the mean
    sigma    = df.median()  # get the standard deviation
    filtered = df[(mu - k*sigma < df) & (df < mu + k*sigma)]
    return filtered
```

3. Split, pipeline and scale the data:
   Split the training data to get 20% test data. Using Pipeline() method will combine all features.

```
# split the data
X, y = df.values, y.values
X_train, X_val, y_train, y_val = train_test_split((df.loc[:,['MMIN','MMAX','CACH','CHMIN','CHMAX']]), y, test_size=0.2, random_st

# data preprocessing using sklearn Pipeline
pipeline = Pipeline([
    ('poly', PolynomialFeatures(degree=1, interaction_only=True)), # multiply features together
    ('scale', MinMaxScaler()), # scale data
])
```

MinMax Scaler was used to scale the data for better accuracy

4. Apply Regression Method:

Initially, Elastic Net=1.7 was used to fit the model. However, when we applied Random Forest search to fit the training data we got a much better results.

```
Model MAE: 22.11394642857143
Mean  MAE: 74.95490506329114
```

b) Algorithm 2

Packages: numpy, metrics, sklearn.model_selection, sklearn.ensemble, sklearn.model_selection

Methodology:

1. Load the data:
   First set the printing options and then load the data.
2. Modify the datasets:
   Modify the X_train and x_test set to add two columns to the dataset

```
X_train_mod = np.insert(X_train, -1, 1/X_train[:, 0], axis=1)
X_train_mod = np.insert(X_train_mod, -1, X_train[:, 5]**2,axis=1)
X_train_mod = X_train_mod[:,1:]
```

3. Split and Use Grid Parameters:
   Split the data and use parameter grid to form a dictionary for RandomForestRegrssor to train the data

```
param_grid = {
             'bootstrap': [True],
             'max_depth': [70,80, 90, 100, 110, 120,130],
             'max_features': [2,3,4],
             'min_samples_leaf': [2,3, 4, 5],
             'min_samples_split': [2,3,4,5,6,],
             'n_estimators': [200,300,400,500,800,1000]
             }
```

4. Fit and Predict with Random Forest Regressor:
   Fit the given train data into model and predict the value for test data

```
cpu_model.fit(X_train_CV, y_train_CV)

y_train_pred = cpu_model.predict(X_test_CV)
y_train_pred = np.maximum(y_train_pred, 0)
```

5. Check for Accuracy:
   Calculate the accuracy and then fit and predict values for modified data

```
print("Accuracy:",metrics.r2_score(y_train_pred, y_test_CV))

y_pred = cpu_model.predict(X_test_mod)
y_pred = np.maximum(y_pred, 0)
```

6. Arrange results:
   Arrange answer in two columns. First column, Id, is an enumeration from 0 to n-1, where n is the number of test points. Second column, PRP, is the predictions.

```
test_header = "Id,PRP"
n_points = X_test_mod.shape[0]
y_pred_pp = np.ones((n_points, 2))
y_pred_pp[:, 0] = range(n_points)
y_pred_pp[:, 1] = y_pred
np.savetxt('my_submission.csv', y_pred_pp, fmt='%d,%f', delimiter=",",
           header=test_header, comments="")
```

Result:

Both the algorithms run efficiently and produce a similar prediction. Both the algorithms use RandomForestRegressor to gain efficient prediction.

3. CLASSIFICATION

Overview

Classification produces accuracy values based on input data sets. Classification models include logistic regression, decision tree, random forest, gradient-boosted tree, multilayer perceptron, one-vs-rest, and Naive Bayes[4]

K-nearest neighbour(KNN) – this algorithm analyses the given dataset and returns the most common value as the prediction[4].

Support Vector machine(SVM) – it basically separates classes for predicting accuracy[5].

Task 2: To classify epi vs stroma superpixels by constructing two algorithms and compare their efficiency [3].

a) Algorithm 1

Packages: numpy, from skleran: neighbours, svm, preprocessimg, metrics, sklearn.model_selection

Methodology:

1. Load the data:
   Load the files and split the training set to get test set with test_size = 0.2
2. Pre-processing data:
   Pre-process training data using quantile transformer to fit test set

```
scaler = preprocessing.QuantileTransformer(random_state=0).fit(X_train_mod)
X_train_preprocess = scaler.transform(X_train_CV)
X_test_preprocess = scaler.transform(X_test_CV)
X_test_final_preprocess = scaler.transform(X_test_mod)
```

3. Apply SVM for classification:
   Implement SVM and predict data for both derived test set and given test data to check accuracy of the predicted values.

```
clf = svm.SVC(C=1.0,kernel='linear',decision_function_shape='ovr')
clf.fit(X_train_preprocess,y_train_CV)

y_pred_test = clf.predict(X_test_preprocess)

print("Accuracy:",metrics.accuracy_score(y_pred_test, y_test_CV))

y_pred = clf.predict(X_test_final_preprocess)
```

4. Arrange the results:
   Arrange answer in two columns. First column, Id, is an enumeration from 0 to n-1, where n is the number of test points. Second column, EpiOrStroma, is the predictions.

```
test_header = "Id,EpiOrStroma"
n_points = X_test_mod.shape[0]
y_pred_pp = np.ones((n_points, 2))
y_pred_pp[:, 0] = range(n_points)
y_pred_pp[:, 1] = y_pred
np.savetxt('my_submission.csv', y_pred_pp, fmt='%d', delimiter=",",
           header=test_header, comments="")
```

b) Algorithm 2

Packages: numpy, pandas, seaborn, sklearn.

Methodology:

1. Load the data:
   Load the data and transform the dataset. Delete few columns and null value columns to improve efficiency.

```
for col in ['Mean.Layer.1','Mean.Layer.2','Mean.Layer.3','Standard.deviation.Layer.1',
            'Standard.deviation.Layer.2','Standard.deviation.Layer.3','Skewness.Layer.1','Skewness.Layer.3',
            'Ratio.Layer.2','Mean.of.inner.border.Layer.1','Mean.of.inner.border.Layer.2',
            'Mean.of.inner.border.Layer.3','Mean.of.outer.border.Layer.1','Mean.of.outer.border.Layer.2',
            'Mean.of.outer.border.Layer.3','Border.Contrast.Layer.1','Border.Contrast.Layer.2',
            'Border.Contrast.Layer.3','Contrast.to.neighbor.pixels.Layer.1..3.','Contrast.to.neighbor.pixels.Layer.2..3.',
            'Contrast.to.neighbor.pixels.Layer.3..3.','Perimeter..polygon...Pxl.','Stddev.of.length.of.edges..polygon...Pxl.',
            'Density.of.sub.objects..mean..1.','Direction.of.sub.objects..mean..1.',
            'GLCM.Contrast..quick.8.11..Layer.1..all.dir..','GLCM.Dissimilarity..quick.8.11..Layer.1..all.dir..',
            'GLCM.Entropy..quick.8.11..Layer.1..all.dir..','GLCM.Mean..quick.8.11..Layer.1..all.dir..',
            'GLCM.StdDev..quick.8.11..Layer.1..all.dir..','Edge.Contrast.of.neighbor.pixels..Prototype..Layer.1..3.',
            'Edge.Contrast.of.neighbor.pixels..Prototype..Layer.2..3.','Edge.Contrast.of.neighbor.pixels..Prototype..Layer.3..3.
            'StdDev..to.neighbor.pixels.Layer.1..3.','StdDev..to.neighbor.pixels.Layer.2..3.',
            'StdDev..to.neighbor.pixels.Layer.3..3.','Circular.StdDev.Layer.3..R1..User..3..R2..Same...R1...border.',
            'Mean.Diff..to.neighbors.Layer.1..0.','Mean.Diff..to.neighbors.Layer.2..0.','Mean.Diff..to.neighbors.Layer.3..0.',
            'Mean.Diff..to.neighbors..abs..Layer.1..0.','Mean.Diff..to.neighbors..abs..Layer.2..0.',
            'Mean.Diff..to.neighbors..abs..Layer.3..0.','Shape.index','Average.length.of.edges..polygon...Pxl.']:
    df.drop(col,axis=1, inplace=True)
    X_test.drop(col,axis=1, inplace=True)
```

2. Split and Scale Data:
   Split the data and apply standard scalar because the predictors are of different range of values, scaler gets the values in a standard format

```
scaler=StandardScaler()

X_train=scaler.fit_transform(X_train)
```

3. Use KNeighbourClassifier:
   Apply KNeighbourClassifier with number of neighbours as parameter, fit the training data in the classifier.

```
clf = KNeighborsClassifier(n_neighbors=17)

clf.fit(X_train,y_train.loc[:,'EpiOrStroma'])
```

4. Predict and Check Accuracy:
   Fit the given train data into model and predict the value for test data. Then measure the accuracy of the algorithm.

```
y_test_pred2 = clf.predict(scaler.transform(X_test))

print(y_test_pred2.shape)

y_test_pred2 = clf.predict(scaler.transform(X_test))

y_test_val = clf.predict(scaler.transform(X_val))

print("Accuracy: ",metrics.accuracy_score(y_test.loc[:,'EpiOrStroma'], y_test_val))

print(y_test_pred2.shape)

df_sub = pd.DataFrame({'Id': np.arange(1596), 'EpiOrStroma': y_test_pred2})
```

Result:

Both the algorithms produced same accuracy of 93.33%

4. LITERATURE REVIEWS

Topic 1: Auto-Encoding Variational Bayes, DiederikP Kingma, Max Welling

Review 1:

First, the authors show that a reparameterization of the variational lower bound with an independent noise variable yields a lower bound estimator that can be jointly optimized w.r.t. variational and generative parameters using standard gradient based stochastic optimization methods. Look for methods to efficiently learn the parameters of directed probabilistic models whose continuous latent variables have intractable posterior distributions The variational approach to Bayesian inference involves the introduction of an approximation to the intractable posterior, used to maximize the variational lower bound on the marginal likelihood.

Solid lines denote the generative model pθpθ, dashed lines denote the variational approximation qϕ to the intractable posterior pθ. The variational parameters are learned jointly with the generative model parameters. the authors introduced a recognition model qϕ: an approximation to the intractable true posterior pθ. In contrast with the approximate posterior in mean-field variational inference, it is not necessarily factorial and its parameters are not computed from some closed-form expectation. These methods are targeted at either unnormalized models or limited to sparse coding models, in contrast to our proposed algorithm for learning a general class of directed probabilistic models. Since the SGVB algorithm can be applied to almost any directed model with continuous latent variables, there are plenty of future directions: learning hierarchical generative architectures with deep neural networks used for the encoders and decoders, trained jointly with SGVB; time-series models; application of SGVB to the global parameters; supervised models with latent variables, useful for learning complicated noise distributions. Then: Z Z qθ log p(z) dz = N log N dz J J 1X 2 log(2π) 2 2 j=1 j And: Z Z qθ logqθ dz = N log N dz J J 1X log(2π) 2 2 j=1 10 Therefore: Z −DKL qϕ||pθ (z = qθ log pθ log qθ2) −2 −2 2 j=1 When using a recognition model qϕ then and s. d. are simply functions of x and the variational parameters ϕ, as exemplified in the text.

Topic 2: Generative Adversarial Nets

Review 2:

This paper proposes a new framework, which combines the concept of generative and discriminative models of deep learning, using adversarial features, which gives better results in terms of qualitative and quantitative aspects than the existing ones. It is an efficient way of experimenting with unsupervised learning, to generate subsets of actual data to get the actual data distribution. But its main disadvantage is that due to large probabilistic computations, it is not always possible to have distribution generated which corresponds to the actual data. It is also a way of visualizing unsupervised learning, which predicts unknown data by generating data based on assumptions. It requires multiple subtasks to solve the actual problem of prediction, which can be a major disadvantage of it. It has already been worked upon on scenarios, where discriminative features were used on generative framework. It did not yield good results due to probabilistic computations [1]. Also work has been done with two models in competitive fashion[2] Generative adversarial net is broken down into subsets of stochastic gradient descent, where each steps to apply is a hyperparameter. There are two propositions put forward for this algorithm, one for given generator being fixed, optimal discriminator is computer. In the next one, based on the capacity of generator and optimal discriminator, the discriminator can reached its optimum when proper update happens for generator and another parameter specified in the paper.The experiment is done on a range of datasets, by estimating probability on the test data set on changing Gaussian Parzen window. One of the major advantages found after running the experiment was that Markov chain is never needed. The main disadvantage is that discriminator and generator must be synchronized during training.

Topic 3: Distributed Representations of Words and Phrases and their Compositionality

Review 3:

This paper describes the computationally efficient and simple model architecture that learns accurate representations of the words and phrases. The previous methods used the neural network architectures for learning word vectors from the large database without dense matrix multiplication. The Skip-Gram model maximizes the average log probability of the training words from the large dataset by evaluating the less nodes. An alternative approach has used to train the model using ranking the data above noise .

This paper shows an approach which is an improvement on the Skip-gram model, involves the simple version of Noise Contrastive Estimation (NCE) and basic mathematic algebraic operations which enhance both the quality of the vectors and the training speed. They used Negative sampling which works only on the probability distribution of samples. And they used sub-sampling of frequent words that is the co-occurrence of the frequent vectors and the words do not change significantly. The co-occurrence measures by the probability of the words in the training set. They created the phrases by finding the words that occur frequently together and infrequently in other contexts. And from that they trained the model with many reasonable phrases without increasing the size of the vocabulary.

The results of sub-sampling improve the training speed and make the word presentation accurate. They constructed the phrase-based training corpus and trained the Skip-gram models. They found the Skip-gram model exhibit a linear structure and the phrase-based training model performs better on the large datasets. They compared the new approach with the previous neural network-based representations of words and found the training time of the Skip-gram model is very less than the required time by the previous model architectures.

Topic 4: Computational analysis of cell-to-cell heterogeneity in single-cell RNA-sequencing data reveals hidden subpopulations of cells

Review 4:

The paper introduces the problems faced in the study of gene expression in RNA sequences of a cell due to confounders like cell cycle. These factors do not allow the genes to express completely and prevent us from identifying hidden gene patterns in cells. The authors come up with an approach called scLVM which overcomes this problem. This involves two stages:

The first part involves re-creating cell-cycle state. The second step uses the information obtained in the previous step to obtain corrected gene expression. This helps to recognize the effect of confounding factors on gene expression and study variation in gene expression levels that is independent of the cell cycle.

They authors use two major computational techniques to study this.

First they calculate the variance of gene expression due to unobserved factors and then check for gene pairs that were highly correlated to one another across following 3 types of cells:

Type 1: T-Cells without cell-cycle correction

Type 2: T-Cells with scLVM correction

Type 3: T-Cells having the gold standard cell-cycle state

There are graphical representations that show the application of scLVM to identify subpopulations in differentiating T-cells.

Secondly, a nonlinear Principal Component Analysis was used on the first dataset and a clear segregation of the cells based on cell-cycle stage. When this method was used on cells with scLVM-corrected genes, the separation of cells based on cell-cycle stage was not observed. This implied that the genes dependent on cell-cycle was removed significantly. Finally, this experiment was extended to show that scLVM could be used to remove variation in gene expression caused by additional TH2 differentiation factor (along with the cell-cycle factor).They found the results to be consistent with the results with just cell-cycle confounding factor.

5. BIBLIOGRAPHY

1) https://www.mathworks.com/help/stats/machine-learning-in-matlab.html
2) https://searchenterpriseai.techtarget.com/definition/supervised-learning
3) https://moodle.gla.ac.uk/pluginfile.php/1440129/mod_resource/content/3/ml_coursework_1819%20.pdf
4) https://www.geeksforgeeks.org/regression-classification-supervised-machine-learning/
5) https://medium.com/machine-learning-101/chapter-2-svm-support-vector-machine-theory-f0812effc72
6) A first course in Machine Learning, Simon Rogers, mark Girolami
7) Introduction to Machine Learning with Python, Andreas C. Müller & Sarah Guido