Statistical Methods in Machine Learning Exam

Waseem, Zeerak
csp265

May 20, 2015

Contents

1 Question 1

In this question we are asked to

> Build an affine linear model of the data using linear regression and the training data in redshiftTrain.csv only. Report the parameters of the model (do not forget the offset/bias parameter).
> Determine the training error by computing the mean-squared-error of the model over the complete training data set. Compute the mean-squared-error on the test data set redshiftTest.csv.

1.1 Description of software

To answer this question I have used the implementation of Linear Regression from Pedregosa et al. (2011). I use the default parameters. After initialising, I fit the model to the data and predict the test set and the entire training set. The predictions are then reported and scored using the Pedregosa et al. (2011) implementation of calculating the mean squared error.

1.2 Results

	Mean Squared Error
Training Set	0.002811
Test Set	0.003197

Table 1: Mean Squared Error: Linear Regression

Parameters
-0.88026454
-0.00594614
0.10970524
0.27820802
-0.00200288
0.00930554
-0.00864573
0.00402381
-0.10735105
-0.00966409
0.04080815

Table 2: Parameters of the linear regression model

1.3 Discussion

As we see from the results the mean squared error behaves exactly as we would expect it to. The error on the training data is significantly lower than the error on the test data. Given that the error grows moderately on the test set, and that the error exists on the training set we can infer that the model is less likely to suffer from overfitting. Furthermore given how small the errors are we presume that model fits well to the data. Further investigation could be done by plotting and seeing the differences.

2 Question 2

In this question we are asked to investigate whether it's possible to improve the results by using a non-linear model for regression. We are asked to either describe a method that works better or present the results of two non-linear regressors and argue for their correct usage.

2.1 Description of software

In this I have used the implementation of a KNN-regressor and Decision Tree regressor in Pedregosa et al. (2011) for the regression task. Both of these are non-linear, and the implementation of them are used in the same way. That is, we fit the model to the data, and then use the model to predict. Finally we use the Pedregosa et al. (2011) implementation of mean squared error (giving it the prediction and the labels of the data set as argument), to calculate the error. Furthermore we use the GridSearchCV as implemented in Scikit-learn (Pedregosa et al., 2011) which takes the classifier, the parameters to search for and the number of folds as arguments.

2.2 Results

	Training Set	Test Set
Linear Regression	0.002811	0.003197
KNN Regression	0.002282	0.003163
DT Regression	0.000000	0.004514

Table 3: Linear & Non-linear regression results

A note, as the Decision Tree Regressor uses a random seed, the score is the mean of 10 scores. Furthermore a grid search for the best parameter, was inconclusive as to the best number of maximal features switching between the optimal number of features being either $\sqrt{features}$ and $\log_2(features)$.

2.3 Discussion

As table 3 shows, both non-linear regressors perform better on the training set, but only the KNN Regression performs better on the test set.
Given the perfect score for the decision tree regression on the training set for the decision tree we can begin to question whether the relatively poor score (comparitavely) is a result of overfitting the model to the training data.

In order to generalise each time one of the non-linear regressors is called a grid search and cross validation is performed to find the optimal parameters for the model, thus ensuring that the constructed models do not suffer under the parameters set for other data sets.

The results of the regressions do not provide signficant improvements on the test set, but in case of both the non-linear regressors, provide a significantly improved error on the training set. From the result on the training and test set one can infer that non-linear regression provides a slightly better result when using decision trees, but performs a lot worse on the test set when using decision trees, leading to the consideration of overfitting as mentioned.

3 Question 3

In this question we are asked to perform binary classification using a linear and a non-linear method. We are to use grid search for the model selection. Furthermore we are to report specificity, sensitivity, and accuracy of the model. From reading Parikh et al. (2012) we identify that sensitivity and recall different names for the same metric, and that specificity is given by $\frac{TN}{FP+TN}$.

3.1 Description of software

For the parameter selection I have used the grid search and cross validation implementation, GridSearchCV, in Pedregosa et al. (2011). It is employed by supplying the classifier as well as a dictionary containing the parameters to be evaluated and the values for each parameter in a list. Finally the number of folds is given. GridSearchCV returns a model, that yields the best results. The model is then fit on the data and can be used to predict predict the labels on the training and test set the model has been fit.

I have used a support vector machine classifier for both linear and non-linear classifiers, using the 'linear' ($< x, x' >$) and 'rbf' ($-\gamma|x - x'|^2, \gamma > 0$) kernels. Both are run through the grid search and cross validation, before the models are fit the data and the prediction is made.

For the accuracy and recall score we use the implementation in Pedregosa et al. (2011).They, as well as my own specificity method, work by receiving the prediction and the correct labels, and yield the score.

3.2 Results

	Linear Model	Non-Linear Model
Recall	0.996875	1.000000
Specificity	0.971787	1.000000
Accuracy	0.984351	1.000000

Table 4: Scores on training set

	Linear Model	Non-Linear Model
Recall	1.000000	1.000000
Specificity	0.949367	0.987342
Accuracy	0.974843	0.993711

Table 5: Scores on test set

As we see in table 4 both the linear and the non-linear models do extremely well on all metrics reported. Notably the non-linear model reports perfect scores on every metrics. This could have been an indication of overfitting the data, but as we see that both models predict exceptionally well on the test set, that conclusion can be temporarily rejected. Increasing the size of the test data set, and including outliers, would reveal more as far as overfitting goes.

3.3 Discussion

Sensitivity is the fraction of all correct positive predictions over the sum of correctly classified poisitive samples and the samples incorrectly classified as negative. Specificity is similar with regards to negative, that is specificity is the fraction of all correctly classified negative predictions over the sum of the correctly classified negative predictions and the incorrectly classified positive predictions.

In the context of the application, sensitivity reports the fraction of correctly classified genuine users while the specificity reports the fraction of correctly classified fraudulent logins. This can be used to investigate how well the system is able to detect fraudulent and genuine users.

4 Question 4

In this question we are asked to perform a principle component analysis, plot the eigenspectrum and plot the data projected onto the first two components.

4.1 Description of software

I have implemented my own PCA following the method outlined in (Bishop, 2007, chapter 12). Given a data set, without labels, I normalise the data, then calculate the mean for each dimension, then sample the covarience, get the eigenvectors of the covarience, sort the vectors, which allows me to plot the spectrum as well as the principle components.

4.2 Results

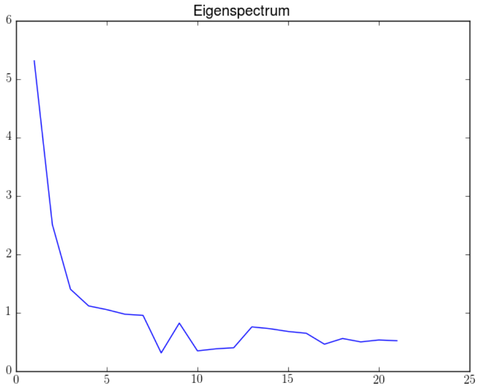

Figure 1: Eigenspectrum

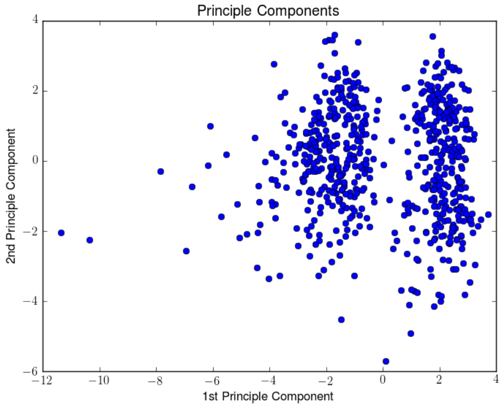

Figure 2: The principle components

5 Question 5

In this question we are asked to perform 2-means clustering and report the 21 dimensional centers. Then project the centers onto the plot of the principle components and discuss the results.

5.1 Description of software

In addition to the software described in section 4.1 I also use the Pedregosa et al. (2011) implementation of K-Means. The K-means implementation takes the the number of clusters, in our case 2, after which it is fit on the data set. The implementation has an attribute `cluster_centers_` allowing us to access the $2, 21$-dimensional clusters sought.

5.2 Results

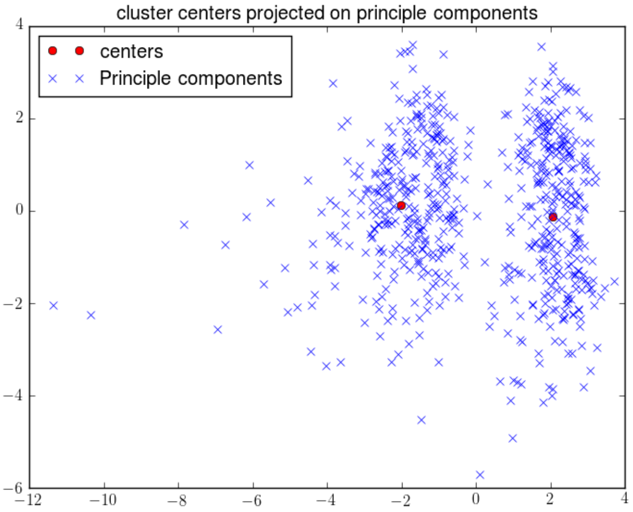

Figure 3: Clusters

Cluster 1	Cluster 2
-0.58059034	0.59717864
0.31162575	-0.32052935
0.33137633	-0.34084423
0.60352028	-0.62076372
-0.55895616	0.57492634
0.39645813	-0.40778551
0.70939172	-0.72966006
-0.12401008	0.12755322
0.46541403	-0.47871158
0.23935885	-0.24619767
-0.06244501	0.06422915
0.51736633	-0.53214823
-0.26469363	0.2722563
0.4662318	-0.4795527
0.03759908	-0.03867334
0.49394562	-0.50805835
0.05008879	-0.0515199
0.53750762	-0.55286498
-0.27000487	0.2777193
0.72935266	-0.75019131
0.53275234	-0.54797384

Table 6: K-means centroids

5.3 Discussion

The clusters are a projected onto the 2-dimensional plot of the principle components. From the plot we see both centroid nodes are roughly placed in the middle of the principle component space. This suggests that our method of implementing PCA is correct and that the data set does indeed form around the clusters, thus validating the correctness of the components.

6 Question 6

In this question we are asked to build a linear and a non-linear model that can classify a multiclass data set and report the results.

6.1 Description of software

The procedure and software used in this is the same as used in section 3.1. That is grid searching for parameters and performing cross validation and then fitting the returned model to the data. I do however use another metric for evaluating the classification error, that is I use a `zero one loss` metric, subtracting the the 1 from the score. I use the `zero one loss` implemented in Pedregosa et al. (2011) which takes the predicted labels and the actual labels as arguments.

6.2 Results

	Training Set	Test Set
Linear Model	0.938086	0.932331
Non-linear model	0.970607	0.947368

Table 7: 1 - Zero One Loss

As we can see from table 7 we achieve a 1.5% decrease in classification error, on the test set, when using a non-linear model. Furthermore the classification performs 3.2% better on the training set compared to the non-linear model.

6.3 Discussion

I have chosen the procedure described above as it allows for building a model that is adapted to the data set that it is working on, rather than simply having a procedure to create a model that is specifically built for one data set. Furthermore allowed me to only make slight modifications to compared to the code required for section 3.1. There are some significant computational costs to this procedure, that could be avoided by not attempting to obtain the best parameters or cross validating. The benefit of having a well fit model will however often outweight the computational costs, particularly when they are as low as they are for the given data set.

7 Question 7

In this question we are asked read Langford (2005) and discuss how three of the methods of overfitting can affect the photometric redshift estimation task.

7.1 Discussion

Overfitting is an issue that it is necessary to be weary of when creating a model. It is quite well illustrated with the means squared error on the training and test data sets for the decision tree regression model. The three methods of overfitting I have chosen to discuss will relate directly to the decision tree regression and its performance.

7.1.1 Parameter Tweak Overfitting

Parameter Tweak overfitting is unlikely to be the reason that the decision tree regression performs as poorly as it does. The decision tree implementation in Scikit-learn (Pedregosa et al., 2011) has a number of parameters, the most significant to its performance beign the depth of the tree and the maximal number of features. The reason that it seems unlikely is that Langford (2005) refers to the parameter tweak to be a case of overfitting with regards to the test data, not the training data.

On the other hand the reason it could be the source of error, is that I employ cross validation, which splits the training data into a number of training and test sets. This allows for the regressor to fit its model closely to the test sets within the training set, thus attaining parameter tweak overfitting.

7.1.2 Brittle Measure

As mentioned we use cross validation as well as parameter search. It is exactly this the blog post warns about. Using measures that can be very brittle. This may have been the case for the decision tree regression, having used both a parameter search and cross validation. In order to mitigate this risk one could simply leave out the cross validation and simply do a parameter search for the best parameters.

7.1.3 Traditional overfitting

Finally we have traditional overfitting, that is the model is too complex for the data set - or the training set simply is too small, for instance meaning that there are few or no outliers. In fact this is what I believe is the issue with the decision tree regression. While the idea of the algorithm is fairly simple, the data sets it can be successfully used upon should be more complex. I believe that this is the cause given the perfect prediction on the training set and the fairly horrible prediction on the test set.

References

Bishop, C. M. (2007). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 1st ed. 2006. corr. 2nd printing 2011 edition.

Langford, J. (2005). Clever methods of overfitting. `http://hunch.net/?p=22`.

Parikh, R., Mathai, A., Sekhar, G. C., and Thomas, R. (2012). Understanding and using sensitivity, specificity and predictive values. *Indian Journal of Ophthalmology*, 56:45–50. `http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2636062/`.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.