INFX574 Problem Set 5

Deadline: Wed, May 23rd, 5:30pm

Introduction

Please submit a) your code (notebooks or whatever you are using) *and* b) the results in a final output form (html or pdf). I recommend to use *scikit-learn.linear_model* and *scikit-learn.decomposition.PCA*.

You are welcome to answer some of the questions on paper but please include the result as an image in your final file. Note that notebooks are just markdown documents (besides the code), and hence it's easy to include images.

Also, please stay focused with your solutions. Do not include dozens of pages of computer output (no one is willing to look at that many numbers). Focus on few relevant outputs only.

Working together is fun and useful but you have to submit your own work. Discussing the solutions and problems with your classmates is all right but do not copy-paste their solution! Please list all your collaborators below:

1.
2. ...

Attribute selection and PCA

This problem set focuses on attribute selection and principal component analysis (PCA) using linear regression as the main workhorse. We employ the Boston housing data from the good old days when median house was <50k... As always, the task is to predict the price (medv) using all the relevant features.

Before you proceed, I strongly recommend you to read Chapter 6 of James et al. (2015).

1 "Active" data exploration

As the first step, add more features and explore the data.

1. Load the data. It is included in *sklearn.datasets* and you can find many copies on the internet too. Make sure you know the coding of all variables. In particular, you should be aware if a variable is categorical or numeric. Explain the coding scheme if it's not obvious.
2. Add some (10 or so) engineered features (synthetic features) to the data. As in the previous problem set, you may use various mathematical operations on a single or multiple features to create new ones.
3. Add another set (10 or so) bogus features, variables that have no relationship whatsoever to Boston housing market. You may just pick random numbers, or numbers from irrelevant sources, such as population of Chinese cities or baseball scores of yesteryear. Give these features distinct names (such as B1-B10) so you (and the reader) can easily recognize these later. You should have about 35 features in your data now.

4. Create a summary table where you show means, ranges, and number of missings for each variable. In addition, add correlation between the price and each variable. You may add more statistics you consider useful to this table.

5. Graphical exploration. Make a number of scatterplots where you explore the relationship between features and the value. Include a few features you consider relevant and a few you consider irrelevant here.

2 Warm-up: a few simple models

Your first real task, although just a warm-up, is to run a few linear models and evaluate their performance through cross-validation. You should code the loss function (mean squared errors) manually, and use this hand-made function throughout the rest of the problemset. You can use either a canned CV function from sklearn, or you can code your own.

2.1 Loss function

Write a function that calculates the mean quadratic loss

$$\frac{1}{N}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})'(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})$$

where N is the number of observations.

1. Write this function. The function should take three inputs: a) the estimated model (or just the estimated parameters $\boldsymbol{\beta}$); b) the test data $\mathbf{X}$; and the test labels $\mathbf{y}$. You may add other inputs if you consider it useful, for instance controls for print verbosity.

   Ensure that the function is available through rest of the problem set. Also, please *use this function* for all following cross-validation exercises. If you are using canned cross-validation code, please ensure that the code uses your actual loss function!

2.2 A few simple regressions

Next, let's estimate and cross validate a few simple regressions.

1. Create a small model. It should include 1-2 variables you consider "relevant", i.e. features you think should be closely related to price.

2. 10-fold cross-validate this model to get the average MSE score (the value of your loss function).

3. Now build the next model with 10 features. Add more features you consider relevant but also those you consider irrelevant. Compute 10-fold MSE for this model.

4. Finally, include all your features and compute MSE. We call this the *full model* below.

5. Compare the results. Which has the best performance?

3 Find the best model

3.1 Can we evaluate all models?

How much time would it take to evaluate all possible models? Let's find it out.

1. How many different linear regression models can you build based on the features you have (including the ones you generated)?

2. Run a test: run the following loop a number of times so that the total execution time is reasonably long (at least 5 seconds) but not too long.

   (a) choose a random number of different features

   (b) estimate the model based on these features

   (c) run 10-fold CV to get the MSE score for this model.

3. Based on the test timings, calculate how long time it would take to evaluate all the possible models. Would it be possible to store all the resulting MSE-s in your computer's memory?

3.2 Forward selection

Next, let's implement the forward selection procedure to determine wich is the best model. Use 10-fold CV again to evaluate your models.

1. James et al. (2015, section 6.1), in particular page 207.

2. Create a series of 1-feature models and pick the best one by 10-fold CV.

   Note: always include the constant.

3. Pick the feature with the lowest loss. This is your 1-feature model.

4. Repeat the procedure with more features until all features are included.

5. Pick the best model using C_p, AIC, BIC or adjusted R^2 (consult James et al. (2015, section 6.1.3)). This is your *forward-selection model*.

4 Principal components

Can we compress the feature space with PCA and use substantially fewer features? Let's try.

4.1 Use raw features

Although in case of PCA the featurs should be normalized, let's start with the raw features with no normalization.

I expect you to use canned packages, such as PCA in sklearn, but you are welcome to experiment with eigenvalue decomposition yourself (consult Leskovec et al. (2014, chapter 11), available on canvas).

Note: certain PCA packages may perform normalization by default. Please ensure that you switch off this option.

1. Consult James et al. (2015) sections 6.3 and 10.2.

2. Perform Principal Component Analysis on all the features in your data (except the target (price) *medv*).

   Extract all components (the number should equal to the number of features) and report:

   (a) Variance explained by each component

   (b) Proportional variance explained by each component

   (c) Cumulative variance explained up to each component.

3. Rotate data: rotate the original features according to the principal components. Most packages have this function built-in but you can consult Leskovec et al. (2014, chapter 11.2.1) for details and interpretation.

4. Find the optimal model in rotated data: estimate the regression model explaining the housing value by the rotated features. Start with the first (most important) rotated feature and add rotated features to the model one-by-one. Each time cross-validate your result.

5. Show a plot how cross-validated MSE depends on the number of components. Which number of components will give you the smallest MSE?

4.2 PCA on normalized data

Now let's normalize the data. I expect you to do it manyally, i.e. to write a function that for a given feature vector $\mathbf{x}$ computes

$$\mathbf{x}^{\mathrm{n}} = \frac{\mathbf{x} - \bar{\mathrm{x}}}{\mathrm{sd}\,\mathbf{x}}$$

where $\bar{\mathrm{x}}$ is the mean of the $\mathbf{x}$ and $\mathrm{sd}\,\mathbf{x}$ is it's standard deviation.

1. Code such a function and apply this to all explanatory variables in your data (but not to the target). This gives you a normalized data matrix X^{n}.

2. Repeat the analysis in 4.1 with normalized data.

4.3 What's the best solution?

Compare all your results: full model, forward selection, PCA on raw data, and PCA on normalized data. Which one is most precise? Which one is most compact? Which one is the easiest to do? Which one is the most straightforward to interpret?

References

James, G., Witten, D., Hastie, T., Tibshirani, R., 2015. An Introduction to Statistical Learning with Applications in R. Springer.

Leskovec, J., Rajaraman, A., Ullman, J. D., 2014. Dimensionality Reduction, 2nd Edition. Cambridge University Press, p. 384–414.