Assignment2

May 5, 2016

1 Assignment 2

1.1 Introduction

In this assignment we will replicate a gene expression data analysis experiment. We will use both unsupervised clustering, and a supervised approach using the Support Vector Machine classifier.

The data is highly dimensional, in other words there are many more features than samples/observations ($p \gg N$). This is typical of gene expression data and of some other medical data problems that you might encounter, such as proteomic data or other biomedical data. When the number of features/dimensions is **much bigger** than the number of samples/observations, this is a high-dimensional problem.

The dataset was described and analysed in the following publication:

S. Ramaswamy, P. Tamayo, R. Rifkin, S. Mukherjee, C.H. Yeang, M. Angelo, C. Ladd, M. Reich, E. Latulippe, J.P. Mesirov, T. Poggio, W. Gerald, M. Loda, E.S. Lander and T.R. Golub. **Multiclass cancer diagnosis using tumor gene expression signatures**. PNAS, Proceedings of the National Academy of Sciences. 2001 Dec 18; 98(26): 15149–15154.

The full text is available via PubMed: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC64998/pdf/pq2601015149.pdf

1.2 Deliverable

The deliverable of this assignment is to replicate the gene expression analysis performed by Ramaswamy et al. in the paper cited above.

1.3 Get the Data

Let's first get the data, which has been made available by the authors of the Elements of Statistical Learning (Hastie, Tibshirani and Friedman, 2nd ed., 2009, Springer Verlag).

In section 18.3, pp. 654–661 of this book, the authors re-analysed the dataset used by Ramaswamy et al. above and have made the formatted gene expression data available via the book's companion website.

The dataset comprises $p = 16,063$ gene expressions for $N = 144$ tumour samples in the training set and $N = 54$ tumour samples in the test set. The data describe 14 different types of cancer. Regarding this dataset, we can safely say that $p \gg N$.

We will now retrieve the data from the Elements of Statistical Learning's website using `pandas` and `urllib2`:

```
In [16]: import urllib2
         import csv
         import pandas as pd
         import numpy as np
         from scipy import stats
         import matplotlib.pyplot as plt
         %matplotlib inline

         url_X_train = 'http://statweb.stanford.edu/~tibs/ElemStatLearn/' \
```

```
                'datasets/14cancer.xtrain'
url_y_train = 'http://statweb.stanford.edu/~tibs/ElemStatLearn/' \
                'datasets/14cancer.ytrain'
url_X_test = 'http://statweb.stanford.edu/~tibs/ElemStatLearn/' \
               'datasets/14cancer.xtest'
url_y_test = 'http://statweb.stanford.edu/~tibs/ElemStatLearn/' \
               'datasets/14cancer.ytest'

# We know there are 144 tumours in the training set and 54 is the test set,
# so let's make some column names:
column_names_train = ["Tumour_Sample_" + str(_) for _ in np.arange(144)+1]
column_names_test = ["Tumour_Sample_" + str(_) for _ in np.arange(54)+1]

# We will use Pandas to read and properly format the text-based data.
# The delimiter is a regular expression to look for zero or more
# repetitions of whitespace (\s).
X_train = pd.read_csv(url_X_train, delimiter='\s*',
                      engine='python', names=column_names_train)
X_test = pd.read_csv(url_X_test, delimiter='\s*',
                     engine='python', names=column_names_test)

# Get the labels and store as a list. There are 14 different cancers in the dataset.
y_train = urllib2.urlopen(url_y_train).read().strip().split()
y_test = urllib2.urlopen(url_y_test).read().strip().split()

# There are 14 different types of cancer, numbered 1 to 14, in the vectors
# y_test and y_train above.
# For visualising, you may find the names of the cancer types useful:
cancer_names_longform = ["Breast adenocarcinoma", "Prostate adenocarcinoma",
                         "Lung adenocarcinoma", "Collerectal adenocarcinoma",
                         "Lymphoma", "Bladder transitional cell carcinoma",
                         "Melanoma", "Uterine adenocarcinoma", "Leukemia",
                         "Renal cell carcinoma", "Pancreatic adenocarcinoma",
                         "Ovarian adenocarcinoma", "Pleural mesothelioma",
                         "Central nervous system"]

cancer_names_shortform = ["breast", "prostate", "lung", "collerectal",
                          "lymphoma", "bladder", "melanoma",
                          "uterus", "leukemia", "renal", "pancreas",
                          "ovary", "meso", "cns"]

# For testing you may want a merged training and test set.
# To save memory, these are commented out for now.
# X = pd.concat([X_train, X_test])
# y = y_train + y_test
```

1.4 Data Exploration

Now that the data have been loaded in `X_train`, `X_test`, `y_train`, and `y_test`, we can take a look a closer look at our data. Note: It is convention to use large `X` for data matrices, and small `y` for target vectors.

As can be seen, in our training set we have $p = 16,063$ genes/features and $N = 144$ tumours/samples:

```
In [17]: X_train.shape
```

```
Out[17]: (16063, 144)
```

To see a preview of the data, we can use the `head` and `tail` functions:

```
In [18]: X_train.head()
```

Out[18]:

	Tumour_Sample_1	Tumour_Sample_2	Tumour_Sample_3	Tumour_Sample_4	\
0	-73	-16	4	-31	
1	-69	-63	-45	-110	
2	-48	-97	-112	-20	
3	13	-42	-25	-50	
4	-86	-91	-85	-115	

	Tumour_Sample_5	Tumour_Sample_6	Tumour_Sample_7	Tumour_Sample_8	\
0	-33	-37	-18	-26	
1	-39	-90	28	-23	
2	-45	-75	10	2	
3	14	-46	30	34	
4	-56	-45	-56	-54	

	Tumour_Sample_9	Tumour_Sample_10	...	Tumour_Sample_135	\
0	-40	22	...	274	
1	-264	-14	...	-915	
2	-335	-21	...	-303	
3	18	26	...	29	
4	-163	-42	...	-171	

	Tumour_Sample_136	Tumour_Sample_137	Tumour_Sample_138	Tumour_Sample_139	\
0	-96	-96	-124	-201	
1	-221	-458	-664	-259	
2	-119	-134	-361	22	
3	243	109	21	140	
4	-224	-630	-519	-277	

	Tumour_Sample_140	Tumour_Sample_141	Tumour_Sample_142	Tumour_Sample_143	\
0	-196	34	-56	-245	
1	-369	-81	-818	-235	
2	-263	-146	-1338	-127	
3	162	-151	-57	197	
4	-277	-174	-989	-562	

	Tumour_Sample_144
0	-26
1	-1595
2	-2085
3	-334
4	-455

[5 rows x 144 columns]

```
In [19]: X_test.tail()
```

Out[19]:

	Tumour_Sample_1	Tumour_Sample_2	Tumour_Sample_3	Tumour_Sample_4	\
16058	-265	-45	76.5	-230	
16059	-290	-61	100.1	-136	
16060	-235	-22	-69.6	-228	
16061	-826	-263	-1444.3	-404	

```
16062            -262             -52          -121.6             -259

       Tumour_Sample_5  Tumour_Sample_6  Tumour_Sample_7  Tumour_Sample_8  \
16058               68              211              183               25
16059              -22              -21              -62                0
16060               27               34              149               28
16061            -1088            -1233            -1797             -425
16062              -58              -47              -34              -15

       Tumour_Sample_9  Tumour_Sample_10        ...         Tumour_Sample_45  \
16058                0              -35        ...                      -18
16059              -11              -51        ...                     -121
16060               17              -57        ...                       56
16061             -189             -392        ...                     -778
16062               -1              -32        ...                     -102

       Tumour_Sample_46  Tumour_Sample_47  Tumour_Sample_48  Tumour_Sample_49  \
16058               975               197                49                73
16059               -24                -9                 8              -107
16060               757               117               -11                35
16061             -2054             -1208              -606             -1391
16062               -44               -75               -30              -201

       Tumour_Sample_50  Tumour_Sample_51  Tumour_Sample_52  Tumour_Sample_53  \
16058               716              1148               570              1306
16059                17               -54              -196               -23
16060                23                89                53               136
16061             -1295             -2558             -2058             -2222
16062               -39                 9               -48               -23

       Tumour_Sample_54
16058               -16
16059               -98
16060               -52
16061              -297
16062               -87

[5 rows x 54 columns]
```

Let's see how the classes are distributed. First let's look at the number of unique values, which should equal 14, as we know we have 14 different cancer types:

```
In [20]: len(np.unique(y_train))

Out[20]: 14
```

We can see how the cancer types are distrubuted using the `itemfreq` function of the SciPy `stats` package:

```
In [21]: stats.itemfreq(y_train)

Out[21]: array([['1', '8'],
                ['10', '8'],
                ['11', '8'],
                ['12', '8'],
                ['13', '8'],
```

```
       ['14', '16'],
       ['2', '8'],
       ['3', '8'],
       ['4', '8'],
       ['5', '16'],
       ['6', '8'],
       ['7', '8'],
       ['8', '8'],
       ['9', '24']],
      dtype='|S21')
```

Using the `cancer_names_longform` list we declared above, we can print tumour frequencies nicely:

```
In [22]: for freq in stats.itemfreq(y_train):
             print "%s samples appear %s times (shortform: %s)." \
             % (cancer_names_longform[int(freq[0])-1],
                freq[1],
                cancer_names_shortform[int(freq[0])-1])
```

```
Breast adenocarcinoma samples appear 8 times (shortform: breast).
Renal cell carcinoma samples appear 8 times (shortform: renal).
Pancreatic adenocarcinoma samples appear 8 times (shortform: pancreas).
Ovarian adenocarcinoma samples appear 8 times (shortform: ovary).
Pleural mesothelioma samples appear 8 times (shortform: meso).
Central nervous system samples appear 16 times (shortform: cns).
Prostate adenocarcinoma samples appear 8 times (shortform: prostate).
Lung adenocarcinoma samples appear 8 times (shortform: lung).
Collerectal adenocarcinoma samples appear 8 times (shortform: collerectal).
Lymphoma samples appear 16 times (shortform: lymphoma).
Bladder transitional cell carcinoma samples appear 8 times (shortform: bladder).
Melanoma samples appear 8 times (shortform: melanoma).
Uterine adenocarcinoma samples appear 8 times (shortform: uterus).
Leukemia samples appear 24 times (shortform: leukemia).
```

You can take a quick look at some statistics values for each gene using the useful `describe` function (we use `transpose` to perform the analysis on a gene-by-gene basis). For example you may want to look at mean expression levels for each gene to see if they are over-expressed or under-expressed:

```
In [23]: # Note: The transpose() function here does not permanently
         # transpose the data stored in X_train.
         X_train.transpose().describe()
```

Out[23]:

	0	1	2	3	4	\
count	144.000000	144.000000	144.000000	144.000000	144.000000	
mean	-63.645833	-257.881250	-215.511806	26.672917	-224.707639	
std	65.367864	223.797476	296.444985	115.764797	164.191345	
min	-245.000000	-1595.000000	-2085.000000	-334.000000	-989.000000	
25%	-100.250000	-312.750000	-281.750000	-26.500000	-282.750000	
50%	-55.500000	-207.500000	-119.500000	14.500000	-186.500000	
75%	-24.750000	-108.750000	-47.750000	73.000000	-107.000000	
max	274.000000	28.000000	63.000000	456.000000	-10.000000	

	5	6	7	8	9	\
count	144.000000	144.000000	144.000000	144.000000	144.000000	
mean	-279.172917	-113.795139	-119.026389	-11.290278	68.313194	

```
std    161.841157   335.428306   86.485875   67.216980  173.884559
min   -809.000000 -1621.000000 -347.000000 -413.000000 -322.000000
25%   -381.000000  -263.750000 -176.000000  -33.250000  -16.250000
50%   -266.000000   -76.000000  -96.000000   -7.000000   41.000000
75%   -133.000000    85.500000  -64.000000   21.000000  115.250000
max    -30.000000   655.000000  196.000000  121.000000  889.000000

           ...           16053       16054       16055       16056  \
count      ...      144.000000  144.000000  144.000000  144.000000
mean       ...     -123.929861 1271.216667   89.045833  538.594444
std        ...      104.513201  832.840702  112.680539  534.374008
min        ...     -419.000000  -67.000000 -136.000000  -49.000000
25%        ...     -183.000000  515.750000   20.250000  162.500000
50%        ...     -112.000000 1216.000000   70.300000  446.500000
75%        ...      -46.750000 2049.250000  137.500000  682.400000
max        ...      270.000000 3283.000000  751.000000 3712.000000

            16057       16058       16059       16060       16061  \
count  144.000000  144.000000  144.000000  144.000000  144.000000
mean    -3.494444  147.833333  -69.670139   21.419444 -1257.648611
std    104.447686  276.576271  101.915246  220.260558  698.918118
min   -330.000000 -311.000000 -521.000000 -361.000000 -3567.000000
25%    -44.000000    6.750000 -107.250000  -77.250000 -1675.500000
50%     -6.000000   85.500000  -41.500000  -11.000000 -1156.000000
75%     27.250000  210.500000   -9.000000   42.000000  -774.000000
max    422.000000 1936.000000  127.000000 1754.000000  -152.000000

            16062
count  144.000000
mean  -103.130556
std    107.657973
min   -620.000000
25%   -142.250000
50%    -66.500000
75%    -35.500000
max     60.000000

[8 rows x 16063 columns]
```

1.5 Summary

Now that we have read the data in a form which we can easily use, we move on to the deliverables that must be completed for Assignment 2.

2 Deliverables for Assignment 2

2.1 Clustering

Task: Perform hierarchical clustering mimicking the approaches used by Ramaswamy et al. in their paper cited above. Plot a dendogram of your results (SciPy provides dendogram plotting functions, see http://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.cluster.hierarchy.dendrogram.html for example) - or visualise your clustering in any other way you deem reasonable.

Both SciKit Learn and SciPy offer hierarchical clustering algorithms, see http://docs.scipy.org/doc/scipy/reference/cluster.hierarchy.html and http://scikit-learn.org/stable/modules/clustering.html.

Notice that not all clustering techniques are useful for all purposes. In the case of this assignment, we know the number of clusters we are searching for - this is a requirement for certain clustering algorithms. Other algorithms may require parameters you might not immediately have available to you.

```
In [ ]: # Your clustering code. Use as many cells as required,
        # use Markdown cells to document where necessary.
```

2.2 Classification

Task: Use Support Vector Machines and a One Vs. All (OVA) approach to replicate the results from the Ramaswamy et al. paper.

SciKit Learn provides an `SVM` package for Support Vector Machines (see http://scikit-learn.org/stable/modules/svm.html).

Visualise your results appropriately using plots and tables to describe classification results on the test set.

```
In [ ]: # Your classification code. Use as many cells as required,
        # use Markdown cells to document where necessary.
```

3 Important Notes

3.1 Hints

- You may find that scaling or normalising your data will yield better results. See the SciKit-Learn `scale` function: http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.scale.html.
- The `preprocessing` module contains much other useful functionality, see: http://scikit-learn.org/stable/modules/classes.html#module-sklearn.preprocessing.
- Cross validation train/test split indexes can be easily creating using SciKit Learn, see http://scikit-learn.org/stable/modules/classes.html#module-sklearn.cross_validation
- Look up the dataset's analysis in Elements of Statistical Learning, specifically sections 18.3 (SVM One Vs. All, One Vs. One, etc.) and 13.3 (k-nearest neighbours).

3.2 Grading

Your grade will depend on a) quality/inventiveness of approach b) quality of plots or visualisations.

3.3 Submission

In Jupyter, click File -> Download As -> IPython Notebook (.ipynb) and send your completed notebook by email.