Predicting Salaries From Job Descriptions

Elijah Bernstein-Cooper, Ben Conrad, Ahmed Saif

December 4, 2014

1 Introduction

Under the context of natural language processing, this lab explores the relation between job descriptions and salaries. This topic was the focus of a Kaggle competition whose sponsor, Adzuna, had a database of job listings of which only half provided salary information (the winner received $3000). As applicants will more likely apply to descriptions that give a salary, Adzuna's placement rate (and hence Adzuna's) is improved if they can provide an estimated salary for those descriptions that did not originally include one. The employee recruiting business is structured so that Adzuna generally can't directly ask the companies to provide salary estimates. This is challenging from the legal standpoint, as grossly incorrect salaries may expose Adzuna to claims from applicants and companies, and applicant experience, since Adzuna's estimates must seem plausible to applicants before they will be willing to spend the time applying.

While Adzuna could manually estimate these salaries, scalability encourages throwing computers at the problem. In this lab we will be using Adzuna's job description and salary datasets, divided into training and test sets. These descriptions vary in word count, industry, employment level, and company location, while the salaries are the mean of the provided salary range. The variability in description content leads to notoriously sparse matrices, so we will be interested in the tradeoffs of various feature descriptors. The naive approach to this problem is to count the occurrences of individual words and associate them to salaries; here each word is a feature and as there are many descriptive words the resulting matrices will be sparse. Other feature choices may be individual word length, occurrences of word pairs or triplets (i.e. "technical communication"), n-grams (sequences of n characters), and many others. It is common to ignore stop words like "the","a","it","you","we", etc. because they add little information.

2 Warm-Up

Our goal in the warm-up is to use two job descriptions with known salaries to predict the salary of another job given the description. Here are two examples from the dataset:

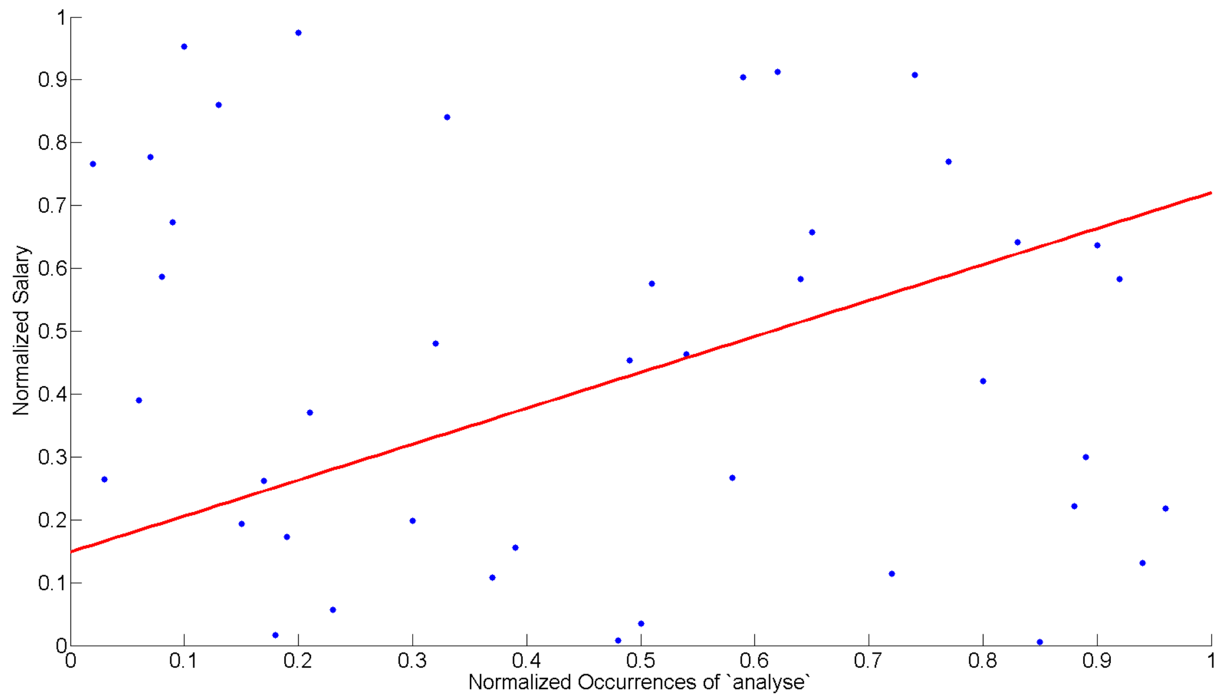
```
Engineering Systems Analyst Dorking Surrey Salary ****K Our client is
located in Dorking, Surrey and are looking for Engineering Systems
Analyst our client provides specialist software development Keywords
Mathematical Modelling, Risk Analysis, System Modelling, Optimisation,
MISER, PIONEEER Engineering Systems Analyst Dorking Surrey Salary ****K
```

with a salary of \$25,000 and

```
A subsea engineering company is looking for an experienced Subsea Cable
Engineer who will be responsible for providing all issues related to
cables. They will need someone who has at least 1015 years of subsea
cable engineering experience with significant experience within
offshore oil and gas industries. The qualified candidate will be
responsible for developing new modelling methods for FEA and CFD. You
will also be providing technical leadership to all staff therefore you
must be an expert in problem solving and risk assessments. You must
also be proactive and must have strong interpersonal skills. You must
be a Chartered Engineer or working towards it the qualification. The
company offers an extremely competitive salary, health care plan,
training, professional membership sponsorship, and relocation package
```

having a salary of \$85,000.

One method to predict the salary from another description is least squares estimation. Least squares estimation can be thought of as an optimization problem which aims to minimize the error estimated and measured data. In our case, we want to predict salaries from the words contained in the job descriptions. For now, let's consider only occurrences of "analyse" in the description. If you plot the number of occurrences against the job salary, you might produce something like



In solving the least squared problem, we're looking for the line which passes nearest to all of the points, as measured by the Euclidean distance, or

$$error = \sqrt{(x_i - x_L)^2 + (y_i - y_L)^2} \tag{1}$$

If we considered two words, say "analyse" and "qualified", we would now have a 3D space to find our least squared solution with one axis being occurrences of "analyse," another "qualified," and the third the salary. Here, our solution will be a plane that slices the 3D space; adding more words - or features as they're more generally described - increases the dimensionality of this space, and our solution becomes a hyperplane. In all cases, we want to minimize the distance from all of the data points to the lower-dimensional estimate.

We would like to determine the words that best predict salary, or even better the frequency of the words which best predict salary. Here we show the 11 most common words for each description:

Description 1	
****k	2
analysis	1
analyst	3
and	1
are	1
client	2
development	1
dorking	3
engineering	3
for	1
in	1

Description 2	
1015	1
a	2
all	2
also	2
an	3
and	5
assessments	1
at	1
be	6
cable	2
cables	1

We can collect these word counts into matrix $\boldsymbol{A}$, and the salaries into vector $\boldsymbol{b}$. $\boldsymbol{A}$ will have 2 rows, one for each description and as many columns as there are unique words between the two descriptions. $\boldsymbol{b}$ will have 2 rows, one salary for each description, and one column. $\boldsymbol{A}$ and $\boldsymbol{b}$ will look like the following

$$\boldsymbol{A} = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 & 1 & 3 & 1 & 1\cdots \\ 0 & 1 & 2 & 2 & 2 & 3 & 0 & 0 & 5 & 0\cdots \end{bmatrix}$$

$$\boldsymbol{b} = \begin{bmatrix} 25000 \\ 85000 \end{bmatrix}$$

with the first 2 corresponding to the two occurrences of '****k' in the first descripton and the later 1, 5 column to occurrences of 'and' in both descriptions.

We can then set up our problem as

$$\boldsymbol{b} = \boldsymbol{A}\boldsymbol{x} \tag{2}$$

where $\boldsymbol{x}$ contains the weights (importance) of each word in predicting the salary of the job. We can find a solution for $\boldsymbol{x}$, $\hat{\boldsymbol{x}}$, by minimizing the residual errors between $\boldsymbol{b}$ and $\boldsymbol{A}\boldsymbol{x}$. This is the same as minimizing the sum of squared residuals,

$$\|\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}\|_2^2 \tag{3}$$

This optimization has a well-known solution for $\hat{\boldsymbol{x}}$

$$\hat{\boldsymbol{x}} = (\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{b} \tag{4}$$

when $\boldsymbol{A}$ is positive semidefinite and, therefore, invertible.

In cases when $\boldsymbol{A}$ is not positive semidefinite, as in the $\boldsymbol{A}$ derived from the word counts above, we can use the right psuedo-inverse of $\boldsymbol{A}$. In Matlab this is

$$\hat{\boldsymbol{x}} = \texttt{pinv}(\boldsymbol{A}) * \boldsymbol{b} \tag{5}$$

The least squares solution to this problem is

$$\hat{\boldsymbol{x}} = \begin{array}{cl} \left[\begin{matrix} 1819.6517 \\ 1730.9558 \\ 1427.6805 \\ 1250.2887 \\ 1213.1011 \\ 1213.1011 \\ 1213.1011 \\ 909.8258 \\ 909.8258 \\ 909.8258 \\ 732.4341 \\ \vdots \end{matrix}\right] & \begin{matrix} \text{“}\textit{be}\text{” from } [0,6] \\ \text{“}\textit{and}\text{” from } [1,5] \\ \text{“}for\text{” from } [1,4] \\ \text{“}engineering\text{” from } [3,2] \\ \text{“}must\text{” from } [0,4] \\ \text{“}will\text{” from } [0,4] \\ \text{“}you\text{” from } [0,4] \\ \text{“}an\text{” from } [0,3] \\ \text{“}subsea\text{” from } [0,3] \\ \text{“}the\text{” from } [0,3] \\ \text{“}modelling\text{” from } [2,1] \\ \vdots \end{matrix} \end{array}$$

The bracketed numbers give the number of occurrences of that word in each of the descriptions; with only two samples it should not be surprising that the most heavily-weighted words are unique to each description. But notice how some words are identically-weighted and that from these 11 ‘most important’ words, few of them are highly descriptive.

Activity 1

Construct the above $\boldsymbol{A}$ and $\boldsymbol{b}$ matrices using the 1st and 2nd descriptions in `activity1.mat`. $\boldsymbol{A}$ should be a $2\times m$ matrix where m is the number of unique words between the two descriptions. $\boldsymbol{b}$ should be a 2×1 matrix. Can you reproduce the above solution for $\hat{\boldsymbol{x}}$? Report on the uniqueness of the solution for $\hat{\boldsymbol{x}}$ in this problem. Support your conclusion.

The following commands may be helpful:

```
word = strrep('word','charactersToReplace','replaceThemWith')
parts = strsplit('sentence')
lowercase = lower('UpperCase')
uwords = unique(wordArray)
sortedArray = sort(unsortedArray)
```

Next we wish to predict the salary of another description using our $\hat{\boldsymbol{x}}$.

```
Our client is part of an international hotel chain that require an
experienced Cluster Revenue Manager to be based in Hertfordshire. The
Cluster Revenue Manager will drive and influence revenue for three to
four hotels. As Cluster Revenue Manager you will maximise revenue,
market share and profits for multiple hotels through the strategic
coordination of revenue management processes and procedures. The
Cluster Revenue Manager will drive the continued development and growth
of customer service standards, revenue and profits from multiple hotels
and to deliver the companys mission relating to profit, people,
customer and quality. You will currently be a Cluster Revenue Manager
or a Regional/ Area Revenue Manager looking after a minimum of two
propertys or a Revenue Manager in a large unit managing both rooms and
conference space. This job was originally posted as
www.caterer.com/JobSeeking/ClusterRevenueManager_job****
```

having a salary of $45,000.

Activity 2

Use your solution for $\hat{\boldsymbol{x}}$ in Activity 1 to estimate the salary of the third description in the warm-up test data, warmup_test.mat. Since $\hat{\boldsymbol{x}}$ is a weighting of words from the first two descriptions, ignore any words that are unique to the third description.

We now construct the matrix $\boldsymbol{A}$ for this description. This matrix will only contain frequencies of words that were present in the previous two descriptions, so many words in this new description will be left out. We can then use our weights for best predicting words, $\hat{\boldsymbol{x}}$ to estimate the salary of the job for this new description, now contained in the matrix $\boldsymbol{b}$. Our estimated salary will be stored in $\hat{\boldsymbol{b}}$, which we estimate from Equation 2.

The estimated salary is \$3,032.80. About \$41,967 different from the true salary associated with the job. Would Adzuna likely use this technique of linear least squares? Consider how populated our $\boldsymbol{A}$ matrix is for the third description. Is the matrix well-populated or sparsely-populated? Based on your intuition, are all of the words in $\hat{\boldsymbol{x}}$ likely to be good predictors of salary? How might paring down the keywords in our data matrix benefit us? In the following sections we will introduce ways to include many more descriptions in our analysis, and derive more accurate predictors of a salary based on the description.

3 Larger Datasets and Regularized Least Squares

We have seen how a least squares fit to only two descriptions chooses the most common word in the job description associated with the highest salary to best predict a higher salary. Predicting the word "be" goes against intuition. In the warm-up activity, our least squares fit was underdetermined, meaning that there were more variables than equations. With more descriptions we may be able to have more equations than variables, the overdetermined case, which has a unique solution which will likely be more intuitive.

Activity 3

Following the same method as in the warm-up, describe how the frequency matrix will change with added descriptions. If we include every word from each of the descriptions in our frequency matrix, is it likely that the number of salaries will be greater than the number of words in the frequency matrix? Describe how the computation time scales with the size of the frequency matrix. Is it practical to include every word from each description in the frequency matrix?

Lets consider a more realistic example where we are provided 10 descriptions from an individual industry sector each with a salary provided. We will be working with the industry "Customer Services Jobs".

Activity 4

Load the activity data, activity_train.mat and activity_train.mat. With each data set construct a frequency matrix of words, excluding common words you are confident will not be important to a description's salary. Derive a linear least squares solution for $\hat{\boldsymbol{x}}$ in Equation 2 with the first 10 descriptions in the training set. Predict the salaries of the first 10 descriptions in the test set. Report the L_2 norm of the difference between the true salaries and predicted salaries for the test set. Report the word which is best able to predict the salary of a description. State whether this problem is over determined or under determined.

We can see again that the least squares problem is again not unique. One way to derive a unique solution in an under determined linear equation is to use regularized least squares. This is similar to least squares, but with an additional smoothing term in the minimization

$$\min_x \|\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}\|_2^2 + \lambda \|\boldsymbol{x}\|_2^2 \tag{6}$$

where $\lambda > 0$. λ is the smoothing parameter. Larger λ values favor smaller L_2 norms, while small λ values return the minimization to linear least squares. Essentially regularization allows for a unique solution that can be smoothly varying, and not sensitive to small changes in the frequency matrix. The solution to regularized least squares is given by

$$\hat{\boldsymbol{x}} = (\boldsymbol{A}^T\boldsymbol{A} + \lambda \boldsymbol{I})^{-1}\boldsymbol{A}^T\boldsymbol{b} \tag{7}$$

where $\boldsymbol{I}$ is the identity matrix. We will explore a similar regularization in the next section which will help us minimize the number of words that we include our solution.

4 The Lasso Method

One of the issues with predicting a job salary is choosing what key words we should use to predict them. As we have seen, the number of words even for small datasets used to construct frequency matrices and solution vector quickly grows out of hand. Fortunately, there are ways of automatically choosing what words we should choose in predicting the salary for the ads. One such method is the Lasso method, another form of regularization similar to regularized least squares. The Lasso method attempts to solve the following equation

$$\min_x \|\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}\|_2^2 + \lambda \|\boldsymbol{x}\|_1 \tag{8}$$

which is minimizing the least square solution along with the absolute values word weights. By making λ large you are making $\boldsymbol{x}$, the number of words you predict salaries with, small. Smaller λ values correspond to using all words, or back to the linear least squares minimization. **Larger values remove words that are in x and sets them zero.** This form of regularization removes words with low frequencies thus removing the sparsity in the matrix, i.e., removing mostly zero word occurrences in lots of examples.

The implementation of the Lasso method is not so trivial however. There is no closed-form solution to solving the Lasso method. however there are good approximations that derive solutions quickly. The following describes the pseudo-code for implementing the Lasso method:

```
1. Initialize two weight vectors for the frequency matrices, $\hat{\boldsymbol{x}}$ and $\hat{\boldsymbol{x}}'$, consisting of zeros, with lengths equal to the number of unique words in all descriptions.
2. Take singular value decomposition of frequency matrix $\boldsymbol{A}$ by [T, S, V] = svd(A)
3. Initialize $\alpha$ to largest singular value of decomposition of $\boldsymbol{A}$ $1/\max(\boldsymbol{S})$
4. Set $\Delta$, the 2-norm of the difference between $\hat{\boldsymbol{x}}$ and $\hat{\boldsymbol{x}}'$ to a large value.
5. While $\Delta$ is larger than the tolerance and the iteration number is less than the maximum iteration do
    (a) Set $\hat{\boldsymbol{x}} = \hat{\boldsymbol{x}}'$
    (b) Set $\boldsymbol{Y} = \hat{\boldsymbol{x}} + \alpha\ \boldsymbol{A}^T(\boldsymbol{b} - \boldsymbol{A}\hat{\boldsymbol{x}})$
    (c) Set $\hat{\boldsymbol{x}}' = \text{sign}(\boldsymbol{Y})\ \max(|\boldsymbol{Y}| - \alpha\lambda,\ 0)$
    (d) Increase the iteration number
    (e) Set $\Delta = \|\hat{\boldsymbol{x}}$ - $\hat{\boldsymbol{x}}'\|$
6. Return $\hat{\boldsymbol{x}}$ as the predictor.
```

Add some visualization like `[~,xinds] = sort(abs(xhat),'descend'); plot(1:length(xhat),xinds,'.')`

Activity 5a

Construct a function for the Lasso method. Perform the Lasso method on all 100 descriptions using the datasets `activity_test.mat` and `activity_train.mat`. Set $\lambda = 0.1$. The resulting $\hat{\boldsymbol{x}}$ value should be all zeros except for 2 values, corresponding to the predicting words "experience" and "client". Predict the salaries of the test data using the training data. Report the 2-norm of the difference between the predicted and the true salaries. Make a histogram of the residuals between the predicted and the true salaries.

Activity 5b

Perform the Lasso method with different values of $\lambda = 0.0001, 0.001, 0.01, 0.1$. Record the number of important words, (words with non-zero weights), the important words, and the 2-norm of the difference between the true and predicted salaries. Summarize the results in a table. Make a histogram of the residuals between the predicted and the true salaries for each of the λ values as well.

Activity 5c

Experiment with the Lasso method using smaller values of λ. What λ value do you think keeps the number of words reasonable while still including enough words in the weight matrix to accurately predict the salary? This can be an art, there is not a correct answer.

5 Using the Results

We've seen that accurate salary prediction is a function of the algorithm and the size of the training set, and while there are many additional improvements, let's conclude by exploring some results of this analysis.

Recall that the basic form of the least squares problem is $\boldsymbol{b} = \boldsymbol{A}\boldsymbol{x}$. We formed $\boldsymbol{A}$ by counting the number of occurrences of each word in the job description and placed the corresponding salaries in $\boldsymbol{b}$. $\boldsymbol{A}$ can be viewed as a mapping from word-weight space (home of

$\boldsymbol{x}$) to salary space, with the reverse mapping coming from $\boldsymbol{A}$'s inverse. $\boldsymbol{x}$ was then the weight or descriptiveness of each word - how useful its number of occurrences were in predicting the correct salary.

$\boldsymbol{A}$ is a linear operator, so if a particular applicant wants to make \$50,000, we can determine which words they should look for when reading descriptions by computing $\boldsymbol{sum}(\texttt{pinv}(\boldsymbol{A})\mathbf{50000}, \mathbf{2})$. If the applicant includes these descriptive words in their objective statement or elsewhere in their resume, it seems reasonable to expect that their application will be successful (though this is a different - if similar - problem). We can also use the $\boldsymbol{A}$ and $\boldsymbol{x}$ computed from the training data to analyze a resume and suggest what salary their resume implicitly seeks.

Activity 6a

The loaded data also contains the Modern Major General's song and a sample undergraduate electrical engineering resume as `majorGen` and `elecGrad`. Predict how much the Major General and EE grad would make if applying to jobs similar to those in the training set.

Activity 6b

The salaries fall substantially below the poverty line in any country; why is this the case? Compare the most important keywords against those contained in `majorGen` and `elecGrad`; it may help to consider how the predicted salary is computed.

It is beyond the scope of this lab to consider other machine learning strategies that may be more successful and/or faster in solving this problem, but having reached this point it's worth commenting that this method is easy to understand. That is, it's easy to inspect and debug and can give good results without heavily relying on statistical concepts. Achieving good results requires some tuning; even though the test salary prediction error was below [final error], it had difficult predicting the Major General and EE grad salaries. This difference is due to the differing writing styles between the job descriptions, Major General's song, and EE grad resume.