On Intrinsic Dimensionality

Gregory Timmons
North Carolina State University
gbtimmon@ncsu.edu

Pooja Asher
North Carolina State University
pmasher@ncsu.edu

1. INTRODUCTION

Machine learning continues to improve and our ability to generate predictive models has increased significantly. Improved methods to generate models allow us to improve our accuracy and recall in prediction tasks. By 1998, the best published classification of the MNIST dataset was able to produce a classification with an error rate of 1.1% [2]. In 2013 methods using neural networks were able to classify with an error rate of 0.21% [5] which is a huge improvement in terms of error rate.

In 1981 Barry Boehm first published his COCOMO model for effort estimation in software engineering which was an early model for predicting how many months a project should take given a set of attributes that describe the project [1]. The original model was derived using a sample set of 61 projects. In 2000 COCOMO II was published with the hope of creating a model more suited to modern development and used a training set of 161 projects. Both models were generated by using regression to produce a function for expected effort.

In the over 30 years since the first publication of the COCOMO model improvements to our ability to predict effort could be hoped for, however this may not have been true. Menzies, Yang, Mathew, Boehm, Hihn recently put forth a paper where they show a major negative result in the realm of software effort estimation: In a comprehensive examination of modern software effort estimation techniques against the older COCOMO model, no detectable improvement was found when compared using a Scott-Knot procedure [4].

This results is disappointing for many reasons. It shows that effort estimation is still a exceedingly difficult task and it suggests the model is not the limiting factor in our ability to predict effort. This may not be a surprising result given the highly subjective manner in which the data must be collected however it is surprising the amount of research that has gone into improving estimation models when a result like Dr Menzies' suggests that improving the model may not be a fertile area of research in software effort estimation.

This result lead our group to ask what we believe is an important question in Data Science. When do we know that the model is a limiting factor in our predictive success and when has a model reached its fullest potential as a predictive device with a given quality of input. In practical applications this question is very important in directing the effort of researchers and developers, although we were aware of no direct methods to estimate or measure this quality. Our project focuses on the concept of Intrinsic Dimensionality and weather or not that can be used to characterize the predictability of data. We will examine 2 methods of intrinsic dimensionality estimation and then look at the implication of intrinsic dimensionality on predictability. We also propose a novel metric for measuring the significance of the data The first is a simple method based on PCA feature reduction and the second is an MLE based estimation method that was proposed by Levina and Bickel[3]. Then we will explore weather or not intrinsic dimensionality can be used as an effective predictor for success in modeling of data and then discuss why or why might not this approach works.

2. DATASETS

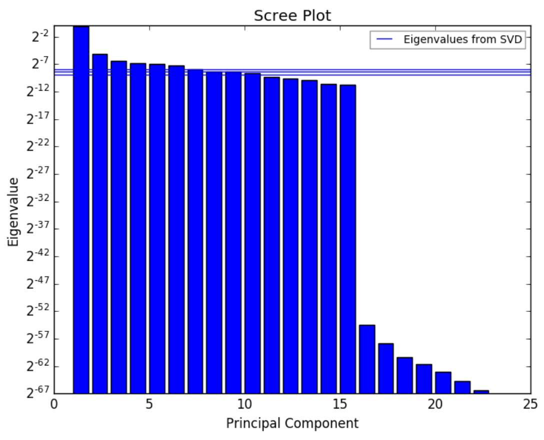
For this project we will make examination of 6 data sets. 3 of the sets are from software estimation literature and include the original COCOMO data set, a set of COCOMO style attributes from NASA project in 1993 and a set of Maxwell effort estimation from Finish banking projects. 3 other sets are taken from other areas where there has been a history of success with prediction.

Dataset	Attrs	Targs	Obs
coc81	25	1	63
nasa93	25	1	93
maxwell	26	1	51
mnist	784	1	200
iris	4	1	150
forest fire	10	1	517

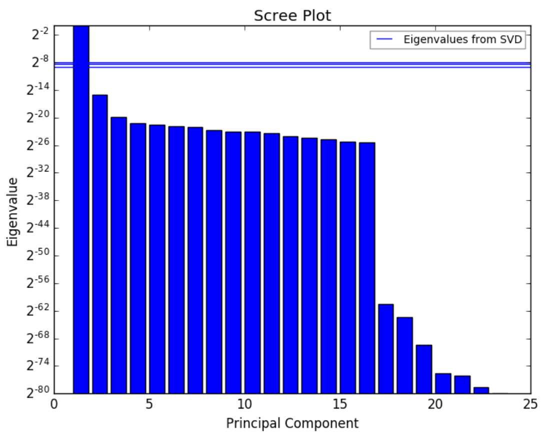
3. PCA THRESHOLD ESTIMATION

The first method of Intrinsic Dimensionality estimation we used was the simplest. This method is implemented by finding the eigenvectors and eigenvalues of the covariance matrix of the data set. Then the eigenvalues are sorted in descending order. All of the eigenvalues greater than a given threshold are kept and the number kept are used as an indicator of intrinsic dimensionality.

We examined scree plots to visually see the eigenvalue break down. The example plot below displays the eigenvalues for the coco81 data set. Datasets with and with out dependent variables are included. The y axis is displayed in a log base 2 scale to better show the distribution of eigenvalues and lines are drawn for the 0.03, 0.02 and 0.01 thresholds. We examine the the estimated dimensionality with a threshold of 0.03, 0.02 and 0.01 for the data set both with and without the dependent variables included in the data.

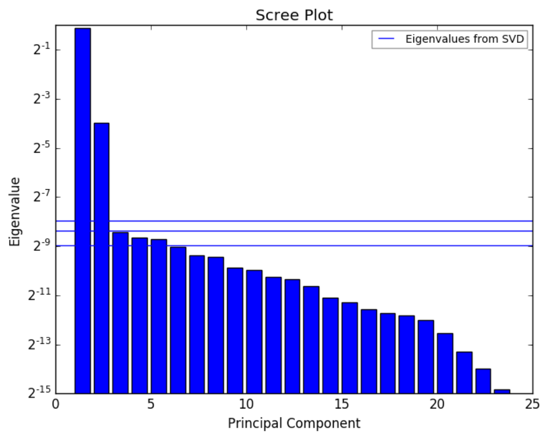


Scree plot in log_2 scale of maxwell without DV

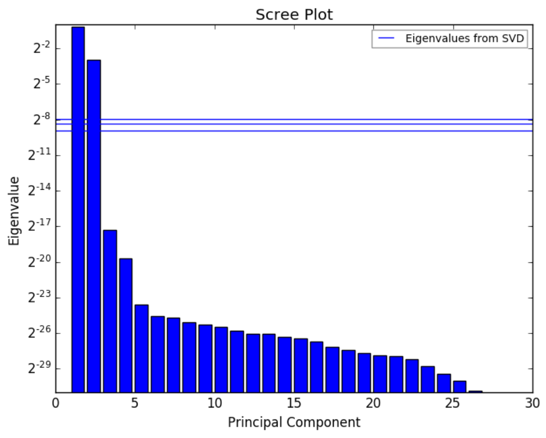


Scree plot in log_2 scale of maxwell with DV

As another example, the scree plots of the maxwell data set are presented below – here we see a greater difference between the scree plot when DV are included and not included.



Scree plot in log_2 scale of coco81 without DV



Scree plot in log_2 scale of coco81 with DV

The table below shows the computed eigenvalues for all of the data sets as estimated by this technique.

Dataset	I03	I02	I01	DI03	DI02	DI01
coc81	2	2	6	2	2	6
nasa93	1	1	1	1	1	5
maxwell	2	2	2	2	2	2
mnist	4	5	8	4	5	8
iris	1	2	2	2	2	2
forest fire	1	1	1	1	1	2

4. MLE ESTIMATION

The second method we used to measure intrinsic dimensionality was a method suggested in a paper by Levina and Bickle [3]. This method works by assuming that the number of points that lie within a certain radius of each point in the data set can be described with a Poisson distribution. As suggested by Levina and Bickle we use the k nearest neighbors rather than some set radius for our calculations. The algorithm is describe in the python code below :

```python
def intrinsic_dimension(
    X,
    kmin=6,
    kmax=12
):

    n = X.shape[0]
    X = X.copy().astype(float)
    X = X[np.lexsort(np.fliplr(X).T)]

    # Standardization
    X -= X.mean(axis=0)
    X /= X.std(axis=0) + 1e-7

    # Compute matrix of log
    # nearest neighbor distances
    X2 = (X**2).sum(1)

    dist = (
        X2.reshape(-1, 1)
        + X2
        - 2*np.dot(X, X.T)
    )

    dist.sort(1)
    dist[dist <= 0] = 10e-7
    knn = .5 * np.log(dist[ : , 1 :kmax+1])

    # Compute the MLE
    S = np.cumsum(knn, 1)
    K = np.arange(kmin, kmax+1)
    dhat = ( -(K-2) / (
              S[:, kmin-1:kmax]
              - knn[:, kmin-1:kmax ] * K
          )  )

    return dhat.mean()
```

This table shows our measurements of ID for the data sets. This first table is data sets without the DV, the second table add the DV back in.

dataset	k∈ $[3, 6]$	k∈ $[4, 8]$	k∈ $[5, 10]$	k∈ $[6, 12]$
coco81	6.94	6.70	6.19	5.94
nasa93	-inf	-inf	-inf	-inf
maxwell	9.24	9.42	9.33	9.31
mnist	6.08	4.99	6.06	5.33
iris	2.57	2.50	2.44	2.40
fire	5.36	5.37	5.40	5.41

MLE estimates without DV

dataset	k∈ $[3, 6]$	k∈ $[4, 8]$	k∈ $[5, 10]$	k∈ $[6, 12]$
coco81	7.04	6.73	6.28	5.87
nasa93	4.83	4.52	4.34	4.20
maxwell	10.12	9.49	9.12	9.02
mnist	6.58	5.58	6.00	5.22
iris	2.93	2.84	2.80	2.73
fire	5.63	5.66	5.69	5.70

MLE estimates without DV

5. SANITY CHECKS

To help to validate the numbers produced above we created a system

6. REFERENCES

[1] B. W. Boehm. *Software Engineering Economics.* Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition, 1981.

[2] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[3] E. Levina and P. J. Bickel. Maximum Likelihood Estimation of Intrinsic Dimension. In *NIPS*, 2004.

[4] T. Menzies, Y. Yang, G. Mathew, B. W. Boehm, and J. Hihn. Negative results for software effort estimation. *CoRR*, abs/1609.05563, 2016.

[5] L. Wan, M. Zeiler, S. Zhang, Y. L. Cun, and R. Fergus. Regularization of neural networks using dropconnect. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 1058–1066, 2013.