Comparison of Dimensionality Reduction Techniques

Jonathan Tan
Harris School of Public Policy
University of Chicago
Chicago, IL 60637
jonathantan@uchicago.edu

Satej Soman
Harris School of Public Policy
University of Chicago
Chicago, IL 60637
satej@uchicago.edu

Abstract

We compare two dimensionality-reduction techniques: principle components analysis and latent space representation via autoencoder neural networks. Both techniques are applied to a feature set and the reduced feature-set is used as the independent variables in a linear regression to recover a known score. We find that XXXXXX. We also discuss considerations such as explainability and training-time.

1 Background

Dimensionality reduction is an important technique to render high-dimensional datasets more tractable for analysis. Often, in a dataset containing thousands of columns, a small number of those columns are relevant for an analytical task. Robust techniques for identifying the most salient features allow for more compact representations of data, reducing storage costs and analysis time. [insert reference XXXX]

2 Techniques studied

2.1 Principal components analysis

Principal component analysis is a dimensionality reduction technique built upon the singular value decomposition (SVD) of a feature matrix. Specifically, the SVD produces matrices $X = U\Sigma V^T$ such that the right singular vectors of X, i.e. the columns of V, give the principal component directions of X. We can obtain a lower-dimensional representation of the original features by projecting the data onto the first n principal component directions. [2] Below, we vary the number of principal components used to predict wine scores and observe how mean squared error between predicted and true wine scores vary with n.

2.2 Autoencoders

In contrast, autoencoders seek to map inputs to a *non-linear* input manifold.

All of our autoencoders produced output vectors of length 50. We trained neural networks with three main architectures. The shallowest was simply a 50-layer hidden layer in between an input and output layer. The deeper networks we tested progressively stepped the input size down. Most of our architectures had only a single decoding layer.

To train the networks, we used stochastic gradient descent over a variable number of training epochs. To maintain decoded fidelity, our output node delta for the update step was proportional to the difference in input and output vectors [insert reference XXXX].

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

3 Dataset

3.1 Description

We use a dataset of wine reviews from Kaggle [insert reference XXXX] consisting mostly of categorical and text data, each with an associated wine review score on a scale of 80 - 100. The features include a free-text description of the wine, as well as pre-parsed information about the wine, including type, grape variety, region of origin, and winery. The key continuous, numeric column in the feature set is the price of each wine bottle. The label we seek to recover is the wine review score.

3.2 Preprocessing

Much of the Kaggle dataset is text or categorical data. Specifically, the description feature contains paragraph text of descriptors for each wine. To obtain a numeric representation of this text data, we used a simple boolean bag-of-words model. We first preprocessed the text data by converting all words to lowercase, removing non-alphanumeric characters and common stopwords, then tokenizing the space-delimited string into separate words. Lastly, we computed a bit vector for each row that represented a one-hot encoding for each word in the description, over all unique words in all rows of the description feature. [insert reference XXXX]

The bag-of-words model gives a particularly sparse matrix, and we faced initial difficulties in computing the representation with the full 130,000-row dataset (which had roughly 45,000 unique tokens). Specifically, the resulting matrix from the description column alone was over 100GB and, when the process ran in a reasonable amount of time, we had difficulty storing the data on our local machines. As an interim solution, we proceeded with a reduced dataset, taking 10,000 rows from the wine dataset instead.

In the future, other numeric representations of text data could be interesting to explore, e.g. low-dimensional word embeddings produced by neural network-based algorithms such as `word2vec` or `GloVe`. [insert reference XXXX]

4 Methodology

For each technique, we apply the dimensionality reduction to obtain a reduced set of feature vectors (X_j). We then regress wine score (S_i) on these reduced features to estimate the least squares weights for the reduced feature set and then calculate predicted scores from the learned weights. Finally, we compare the mean-squared error arising from the true vs. predicted wine scores for several dimensionality reduction techniques.

5 Results

6 Conclusions

- Autoencoders with logistic activation performed the best at dimensionality reduction (according to our regression mean-squared error metric), followed by regular principal-components analysis.
- Neural networks with only ReLU activations do not perform well for dimensionality reduction, compared to other techniques we looked at.
- For the architectures we studied, varying the number of training epochs did not change the mean-squared error.
- Neither combining layers with different activation functions (input and hidden ReLU and sigmoid output), nor adding symmetric decoding layers improved autoencoder performance.

7 Future work

There are a number of ways this work can be extended. As dicussed above, we could use alternate encoding schemes for the free-text data. Additional lines of inquiry involve either different neural-network architectures, different loss functions, or alternative dimensionality reduction techniques.

method	description	MSE
autoencoder	sigmoid, shallow, 1000 epochs	5.649318
autoencoder	sigmoid, shallow, 10000 epochs	5.649318
autoencoder	sigmoid, shallow, 10 epochs	5.649318
autoencoder	sigmoid, shallow, 100 epochs	5.649318
autoencoder	sigmoid, symmetric, 100 epochs	5.653316
autoencoder	sigmoid, medium, 1000 epochs	5.653316
autoencoder	sigmoid, medium, 10 epochs	5.653316
autoencoder	sigmoid, medium, 100 epochs	5.653316
autoencoder	sigmoid, medium, 10000 epochs	5.653316
autoencoder	sigmoid, deep, 10000 epochs	5.665386
autoencoder	sigmoid, deep, 1000 epochs	5.665386
autoencoder	sigmoid, deep, 100 epochs	5.665386
autoencoder	sigmoid, deep, 10 epochs	5.665386
pca	100 components	231.751525
pca	50 components	307.355949
pca	25 components	366.782639
pca	20 components	399.109070
pca	15 components	412.171071
pca	10 components	426.216375
pca	5 components	481.270942
pca	3 components	484.869192
pca	2 components	516.873620
autoencoder	ReLU, deep, 10000 epochs	1074.289408
autoencoder	ReLU, deep, 1000 epochs	1074.289408
autoencoder	ReLU, deep, 100 epochs	1074.289408
autoencoder	ReLU, deep, 10 epochs	1074.289408
autoencoder	ReLU, shallow, 10 epochs	1732.283164
autoencoder	ReLU, shallow, 100 epochs	1732.283164
autoencoder	ReLU, shallow, 1000 epochs	1732.283164
autoencoder	ReLU, shallow, 10000 epochs	1732.283164
autoencoder	ReLU, medium, 10000 epochs	1742.411749
autoencoder	ReLU, medium, 1000 epochs	1742.411749
autoencoder	ReLU, medium, 100 epochs	1742.411749
autoencoder	ReLU, medium, 10 epochs	1742.411749
autoencoder	ReLU, sigmoidOut, 100 epochs	1742.411749
pca	1 components	4524.018520

We could have used regression mean-squared error as the loss function when training neural networks, but there is not a clear analog of this when doing principal-components analysis, so we opted to simply force the decoded output to be as close as possible to the unencoded input.

Further, we had hoped to compare the methods discussed to another technique called t-Distributed Stochastic Neighbor Embedding (t-SNE) [insert reference XXXX]. Other uses of this technique, which specifically targets two- or three-dimensional representations of high-dimensional data. These techniques exist on a spectrum of stochasticity: PCA is deterministic, while autoencoders have stochastic elements, and t-SNE is completely stochastic and very sensitive to hyperparameter tuning.

References

[1] Wine Reviews (2017). *Accessed at https://kaggle.com/zynicide/wine-reviews.* Last accessed Dec 2, 2019.

[1] Mikolov, T., Chen, K., Corrado, G. & Dean, J. (2013) Efficient Estimation of Word Representations in Vector Space. *Accessed at https://arXiv:1301.3781.* Last accessed Dec 10, 2019.

[2] Elden, L. (2007). *Matrix Methods in Data Mining and Pattern Recognition*, pp. 66. Philadelphia, PA: Society for Industrial and Applied Mathematics.

[3] Hinton, G.E. & Salakhutdinov, R.R. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504507, 2006.