Zach Stoebner
CS 4262
Fall 2020

Dimensionality Reduction on Neural Data

PCA & Autoencoding

https://github.com/zstoebs/neural_dim_reduction

Zach Stoebner
CS 4262
Fall 2020

1 Introduction

1.1 Background

Many conventional compression algorithms trade off between two attributes: size of the file and information loss. The most important consideration in many minds, information loss unfortunately does not abide by a law of conservation, so what is lost along the way it cannot be recovered. Although lossless compression algorithms exist, the fact that they cannot efficiently compress data is well-known. However, often times the tasks that we hope to accomplish do not suffer significantly from losing the information required to compress data to a manageable size. In machine learning, dimensionality reduction methods are a subset of conventional data compression that shrink the size of the dataset's examples in the aim of minimizing speed and information loss while maximizing interpretability. [1]

1.2 Motivation

In neuroscience academia, many labs now employ new multi-electrode and optical recording technology that can monitor populations simultaneously and even entire structures, allowing study beyond the single-neuron level. With more timestamps and more neurons, the amount of data acquired by such recording technology scales multiplicatively, which can strain practical machine learning on this raw data to the point where the computation is not worth the time, or even infeasible. In fact, neurons naturally fit the criteria for successful dimensionality reduction because neuronal neighborhoods covary. [2]

Neighboring neurons are both discrete at their axons and continuous through their gap junctions, so they often activate almost simultaneously, conforming to the age-old mantra – "Neurons that fire together, wire together." Essentially, the action potential waveforms can be graphed on the same timescale. However, not all neurons are of the same build; they have different shapes and sizes that cause some to fire faster than others. Within neuronal populations, this differentiation creates classes of neurons which in turn outline a classification problem. The high dimensionality of this data still insinuates expensive computation and is particularly prohibitive for trivial models such as k-nearest neighbors [KNN], which classifies new points by comparing them to every example in the dataset. Trivial models are also the mostly likely to proffer a stark comparison between classification on reduced and unreduced data. Therefore, this project will investigate reducing the dimensionality of neuronal waveforms, the effect that such a transformation has on KNN classification performance, and how two different dimensionality reduction techniques fare in comparison.

2 Methods

2.1 Dataset Summary, Preprocessing, and Feature Extraction

The neurophysiological dataset examined in this project originates from a repository[1] containing the code for a cortical barrel study during whisker-guided locomotion in mice. The subjects were

[1] https://github.com/sofroniewn/tactile-coding

optogenetically stimulated to traverse a winding corridor and neural activity was simultaneously recorded in their whiskers' cortical barrels [3]. Of the 19 original subjects in the dataset, 13 of them have EEG data. Originally, approx. 16,000 neurons were recorded for each of these subjects and from those approx. 30 waveforms were labeled by the authors as either regular, intermediate, or fast spiking and also keyed with other metadata. Each of the subject's waveforms are timeseries of 53 voltage recordings.

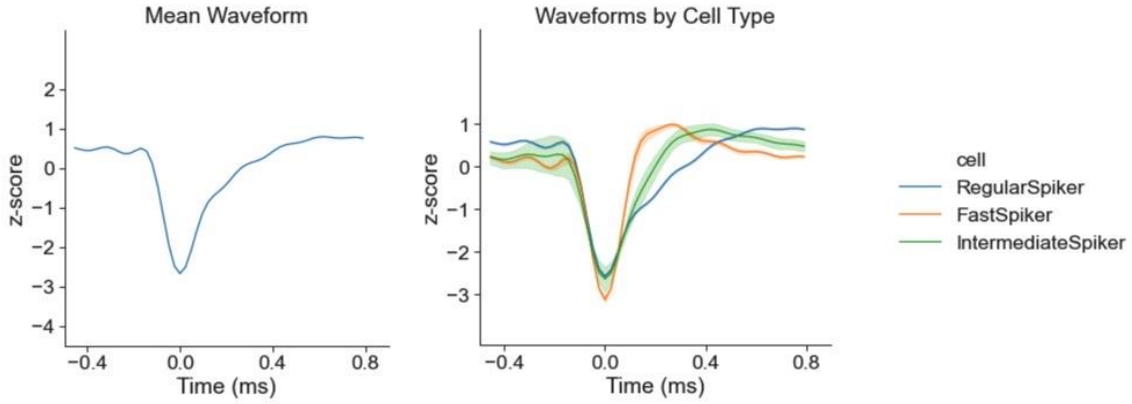

Figure 1. Summary of the waveforms with mean waveform (left) and waveform distributions by cell type (right). Note that the mean waveform is essentially the tightly bounded waveform distribution for regular spikers, which dominate the dataset.

The dataset was constructed with the subject table data by reading in each subject's CSV containing the waveforms. The cell type labels were extracted from the CSV containing the units and metadata for the table entries. The labels and the waveforms were then aligned according to their corresponding source IDs. Prior research indicates that neuronal data should be standardized to z-score values where each example in the dataset is in the same distribution, not the features [2], [3]. For both the PCA and autoencoding versions, the waveforms were accordingly z-scored; for autoencoders, the data were further normalized to a [0,1] interval to mitigate any generative learning issues involving negative inputs. No traditional feature extraction was performed since the purpose of the project was to reduce the dataset's dimension from 53-length feature vectors to only a handful of features. Overall, the dataset is composed of 302 waveforms. Unavoidably, the dataset is unbalanced; regular spikers comprise 247 of the examples while intermediate spikers only make up 4 examples. Figure 1 summarizes the waveforms in the dataset and the similarity between the mean waveform and the waveform distribution for regular spikes demonstrates the imbalance among the classes.

2.2 Approach

To compare and contrast the baseline, PCA, and autoencoding on the same model, a hand-written KNN classifier was implemented using Euclidean distance and majority vote for classification. To test the reduced and unreduced datasets on KNN with the greatest statistical power, a testing function was wrapped around a hyperparameter search for the best k value. The

search function employed cross-validation on a stratified split of the dataset, which mitigated the effects of the unbalanced classes, to determine the performance at a certain k. Even if the cross-validation returned the best accuracy seen so far in the search, it would continue searching higher k values for some specified leeway number of iterations, either until a better k was found or until the leeway was exhausted. Once a good k-value was found, a KNN model with the chosen k was evaluated on the test set and the accuracy was returned, as well as the accuracy on a reclassification of the training set for debugging purposes. A keen acumen would notice that this forward search is biased towards lower values of k. However, lower k values are sufficient for the model to achieve greater than 90% accuracy on these waveforms in a just few seconds. Additionally, the seed affects the randomization of the split so the results for seeds 42, 66, and 123 were compared.

2.3 Experiments

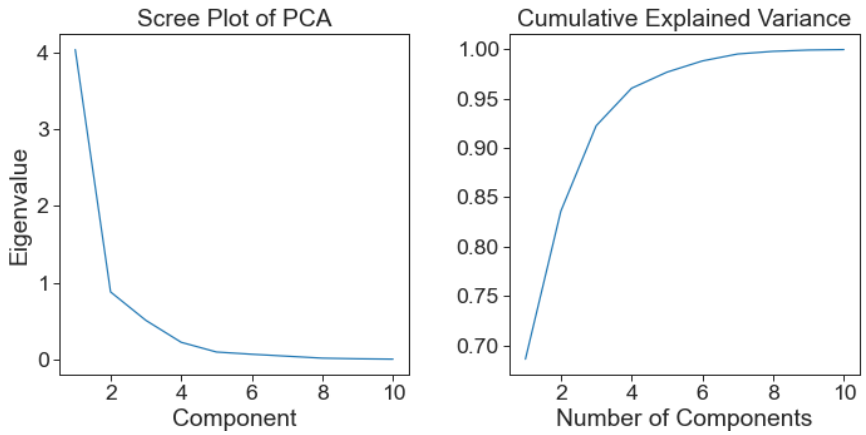

Figure 2. Scree plot (left) and cumulative explained variance of the first N components (right) from PCA applied to the waveforms.

The experiments for PCA and autoencoding had the same structure: 1. find the best reduced dimensionality, 2. reduce the dataset, and 3. test with KNN. For PCA, the scree plot and cumulative explained variance, both shown in Figure 2, were visualized to determine the best number of principal components to include from the transformation. For the purposes of the project, 3 principal components were selected because they explain greater than 90% of the variance in the data and this dimensionality allows visualization of the reduced dataset in 3D space. Scikit-Learn's PCA tool was used, which applies SVD as its decomposition [4].

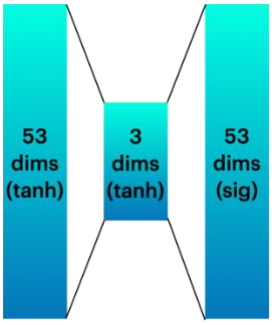

Figure 3. Architecture of the autoencoder for 3D reduction.

To match this paradigm in the autoencoder, a bottleneck of size 3 was elected to yield 3-dimensional encodings of the waveforms. After encountering numerous open issues with the Keras wrapper for Scikit-Learn's cross-validation grid search, autoencoder architectures were explored empirically. The learning curve and the decoded waveforms were used to evaluate different models, particularly to discern whether a certain model was suffering from mode collapse [5]. After trial and error, shallow models performed best, were faster, and had the most interpretable projections because they were less likely to suffer mode collapse. The ultimate architecture for the model is shown in Figure 3 and the chosen hyperparameters for the model's training were 40 epochs with

uniform initialization using Adam as the optimizer and a batch size of 1. The first hidden layer and the bottleneck layer use hyperbolic tangent activation and the output layer applies sigmoid activation.

3 Results

As a baseline, a KNN model was fit to the unreduced waveforms. As one would expect, the results shown in Table 1 were always greater than or equal to the performance of the model on the reduced dataset and hence provide statistical grounds for postulating the significance of the effect of dimensionality reduction on classification.

Seed	**42**	**66**	**123**
Chosen k	1	3	2
Test accuracy	0.968	1.0	0.903
Debug accuracy	1.0	0.985	0.993
Total runtime	3.29	4.08	3.77

Table 1. Baseline results for a KNN fit on 53-dimensional waveform feature vectors.

3.1 PCA

The 3-dimensional PCA transformation of the waveforms is visualized in Figure 4. Now that the dataset is reduced to an interpretable dimension, the classes can be viewed in an understandable space. Moreover, since >90% of the variance is captured in the first 3 principal components, the classes are sequestered almost discretely into their cluster spaces and it is more apparent than ever that regular spiking neurons are the dominant class in this dataset. The results of the KNN fit on the reduced dataset using PCA are shown in Table 2. The accuracies were slightly below that of the baseline but still above 90% accuracy for both the test and debug.

Seed	**42**	**66**	**123**
Chosen k	3	4	3
Test accuracy	0.968	1.0	0.903
Debug accuracy	0.982	0.974	0.985
Total runtime	3.60	4.33	3.35

Table 2. PCA results for a KNN fit on 3-dimensional waveform components.

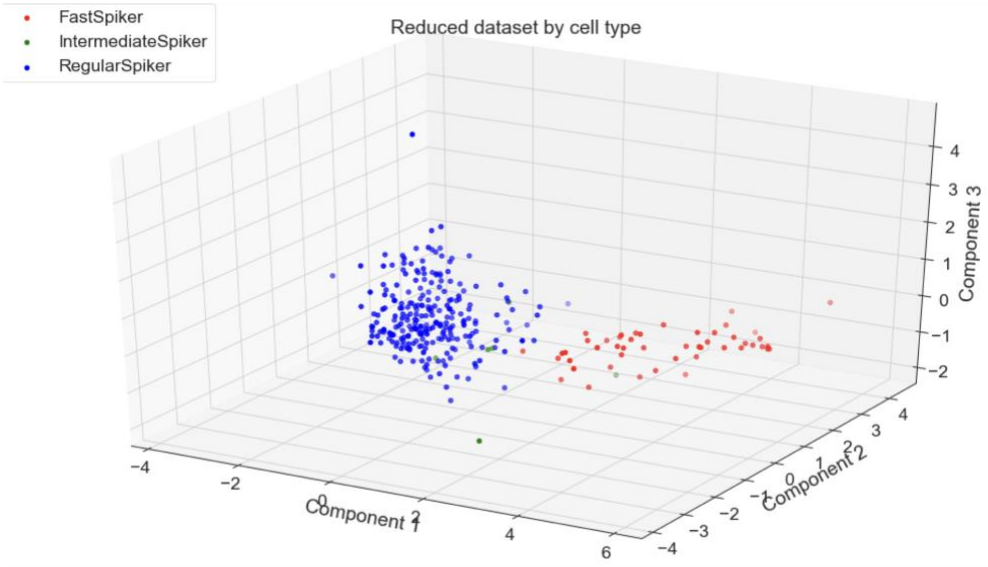

Figure 4. 3D spatial distribution of the waveform principal components from PCA.

Zach Stoebner
CS 4262
Fall 2020

3.2 Autoencoding

The 3-dimensional encoding of the waveforms from the aforementioned autoencoder model are displayed in Figure 5. The convergent loss was greater than 40%, and less than 45% for the best models. Similar to PCA, the autoencoder separates the encodings in 3D space into discrete clusters. The autoencoding results on a KNN model are shown in Table 3. The performance was slightly worse for the testing but consistently better on the debug.

Seed	**42**	**66**	**123**
Chosen k	1	1	1
Test accuracy	0.968	0.968	0.903
Debug accuracy	1.0	1.0	1.0
Total runtime	2.21	2.03	2.10

Table 3. Autoencoding results for a KNN fit on 3-dimensional waveform components.

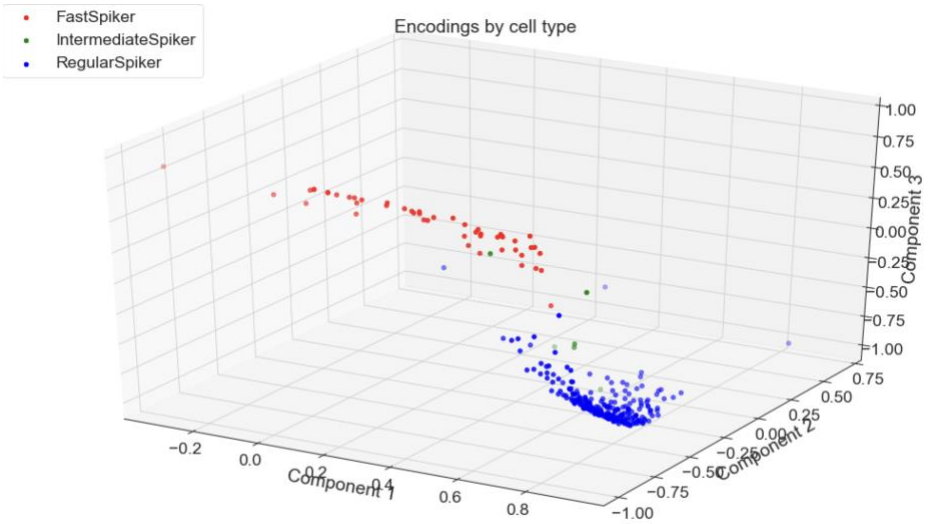

Figure 5. 3D spatial distribution of the encodings from the bottleneck layer of the autoencoder.

4 Discussion

For both PCA and autoencoding, the accuracy is only slightly worse than that of the baseline. For PCA, the test accuracy is exactly the same for the 3 seeds while for the autoencoder it is only slightly worse. On the other hand, for the debug accuracy, PCA performs worse than the baseline while the autoencoder performs better. Given that trend, it might suggest that the autoencoder is somewhat overfitting the dataset, diminishing its generalizability. However, the test accuracy suggests that it is not significantly detrimental. All in all, dimensionality reduction still yields data suitable for high performance, even with information loss.

Delving deeper into the chosen k values, the baseline settles on fewer neighbors than PCA. This phenomenon is likely attributed to the curse of dimensionality where higher dimensions exponentially increase the total feature space, which reduces the efficiency of distance computations, but the parts of that space where the classes reside are the fringes and more dispersed due to non-zero values on incomprehensible axes [6]. Hence, the nearest neighbor is very likely to be in the same class. Interestingly, the encodings result in a chosen k of 1, even in

3 dimensions. Referring to Figure 5, this choice can be verified as the best option given that the class clusters are further apart and therefore more linearly separable than the clusters for the PCA reduction, in which, referring to Figure 4, more examples of opposing classes transgress the territory of other classes. Theoretically, the shallowest autoencoder, such as the one in this implementation, should essentially perform PCA [7], but, in these results even at its simplest level, the autoencoder may have performed better than PCA ever could.

For the final point of comparison, note the runtimes for the model testing with each of these reduced datasets. First of all, the runtime refers to the total time to search for the chosen k, which includes cross-validation at each iteration up to the best plus the leeway iteration, and test on the testing set. In one regard, the autoencoding results ran the fastest because the best model was 1-nearest neighbor. However, the 1-nearest neighbor for the baseline ran a whole second slower than the encodings. Further along that line, the PCA-reduced dataset runtimes were comparable to that of baseline or less at the same seeds, yet it was searching higher values of k. As expected, the reduced dimensionality of the dataset significantly increased the speed fitting the model and digestibility of the data, without significant decrease in performance.

4.1 Autoencoder Challenges and Details

The challenges in the implementation of this project revolved around the autoencoder; applying PCA was straightforward since it is widely understood and optimized. As mentioned above, the Keras wrapper paired with Scikit-Learn's cross-validation grid search functionality has a number of known issues that prohibit scoring metrics other than accuracy. For autoencoders, accuracy is not representative and often never converges, rather the loss converges, although the convergent loss may not necessarily converge to a relatively low value that one would expect for other deep learning methods. The encodings are also highly sensitive to the hyperparameters; change one and the projection hyperplane is starkly different, resulting in crescents, spoons, lines, etc. Occasionally, the autoencoder would zero out one or two of the encoding components, which suggests that it learns that fewer components result in better decodings but this phenomenon also warns of mode collapse. Choosing the right hyperparameters and even iterating on those same hyperparameters a few times to get the best projection is advisable.

5 Conclusion

In this project, dimensionality reduction was shown to be a viable alternative to compromising on the constraints of high dimensional neuronal waveforms. Both the conventional method, PCA, and the deep learning method, autoencoding, successfully reduced the dataset to 3 dimensions with insignificant loss in classification accuracy while significantly reducing the runtime. Of equal importance, these methods reduced the data to dimensions that are interpretable to human understanding. Although the extent of PCA has essentially been explored, autoencoders are fairly novel and complex methods that have the potential for so much more than PCA. That said, a future avenue for this work may reside in the nonlinear

dimensionality reduction of overlapping data to classify these waveforms to their corresponding subjects.

6 Resources

- Scree and cumulative explained variance plots: https://jmausolf.github.io/code/pca_in_python/

- Matplotlib 3D scatter plot: https://stackabuse.com/seaborn-scatter-plot-tutorial-and-examples/

- Keras autoencoder guide: https://blog.keras.io/building-autoencoders-in-keras.html

- Hyperparameter grid search for Keras:
    - https://machinelearningmastery.com/grid-search-hyperparameters-deep-learning-models-python-keras/
    - https://stackoverflow.com/questions/49823192/autoencoder-gridsearch-hyperparameter-tuning-keras
    - https://towardsdatascience.com/autoencoders-for-the-compression-of-stock-market-data-28e8c1a2da3e

7 References

[1] D. DeMers and G. W. Cottrell, “Non-Linear Dimensionality Reduction,” vol. %6, pp. 580--587 %&, 1993.

[2] J. P. Cunningham and B. M. Yu, “Dimensionality reduction for large-scale neural recordings,” *Nat. Neurosci.*, vol. 17, no. 11, pp. 1500–1509, 2014, doi: 10.1038/nn.3776.

[3] N. J. Sofroniew, Y. A. Vlasov, S. A. Hires, J. Freeman, and K. Svoboda, “Neural coding in barrel cortex during whisker-guided locomotion,” *eLife*, vol. 4, no. DECEMBER2015. pp. 1–19, 2015, doi: 10.7554/eLife.12559.

[4] C. Mathematics, “FINDING STRUCTURE WITH RANDOMNESS : STOCHASTIC ALGORITHMS FOR CONSTRUCTING APPROXIMATE MATRIX DECOMPOSITIONS N . HALKO , P . G . MARTINSSON , AND J . A . TROPP Technical Report No . 2009-05 September 2009,” *Techniques*, 2009.

[5] A. Srivastava, L. Valkov, C. Russell, M. U. Gutmann, and C. Sutton, “VEEGAN: Reducing mode collapse in gans using implicit variational learning,” *arXiv*, no. Nips 2017, 2017.

[6] T. et. all. Hastie, “Springer Series in Statistics The Elements of Statistical Learning,” *Math. Intell.*, vol. 27, no. 2, pp. 83–85, 2009, [Online]. Available: http://www.springerlink.com/index/D7X7KX6772HQ2135.pdf.

[7] S. Ladjal, A. Newson, and C. H. Pham, “A PCA-like autoencoder,” *arXiv*, 2019.