Dimensionality Reduction using Dictionary Learning and Visualization using Manifold learning

Anant Sharma (as5194)[1] **Dheeraj Kalmekolan (drk2143)**[1] **Parth Mehta (pm2877)**[1]
Vidya Venkiteswaran (vv2269)[1]

Abstract

Dictionary Learning is a representational learning method of writing the input data as a Linear combination of elements such that the code is sparse which relaxes the condition that components can be non orthogonal. Dictionary learning is very useful specifically for image analysis. It is generally used in applications like Denoising, In-painting, De-mosaicking, Video processing etc. Dictionary learning can also be made to restrict to convert the data into low dimensional space and hence can be used in dimensionality reduction. We wish to analyze different ways to do dimensionality reduction using various techniques and later visualize them using Manifold Learning.

1. Introduction

In Machine Learning, in many algorithms, the speed is greatly hindered with large number of features. Many features don't add a lot of information and hence can be ignored within certain limit. Dimensionality reduction is the technique used to do the aforementioned task. Several algorithms like PCA, Manifold Learning etc can be used for Dimensionality reduction.The main advantage of using Dimensionality reduction is increased speed. Also since we remove the noise, the accuracy also increases in many cases. Dictionary Learning is another algorithm which can be used for dimensionality reduction. Sparse Dictionary Learning expresses the input signal as a sparse Linear combination of Dictionary elements. The Sparsity can be encoded in several different ways and we want to experiment with different ways of encoding sparsity and how they influence the speed, accuracy of classifiers on classification problem. The paper is organized as follow. Section 2 talks about different approaches of dictionary learning, Section 3 talks about Visualizing the Dictionary Learning's output using Manifold Learning. Section 4 discusses results of various classifiers and work on other datasets. Section 5 has a discourse of future extensions and other attempts to learn the process better and limitations of our dictionary learning.

[*]Equal contribution [1]Columbia University in the City of New York, New York, New York, United States of America. Correspondence to: Dheeraj Kalmekolan<drk2143.email@columbia.edu>.

Preliminary work. Under review by the Columbia University in the City of New York. Do not distribute.

2. Dictionary Learning

Basic problem of Dictionary Learning(Mairal et al., 2009) is given a dataset $X = [x_1, x_2, ..., x_k], x_i \in \mathcal{R}^d$ we wish to express it as $D * W$ where $D \in \mathcal{R}^{dxn}$ is the dictionary or the basis and $W \in \mathcal{R}^{nxk}$ where the vectors is R represent the contribution/representation of each component of dictionary. Now encode sparsity we specify the optimization problem as

$$argmin_{D,W}||X - DW||_2^2 + ||W||.$$

Where $||||.$ is a sparse norm. When $n > d$ we call the problem as overcomplete problem and when $n < d$ we call the problem as an undercomplete problem. For dimensionality reduction we will be using the latter one. The main advantage of using dictionary learning is unlike PCA we don't need to specify orthogonality as a constraint. This is mainly useful is classification tasks. To test the power of Dictionary learning we used MNIST dataset(LeCun et al., 2010) as a reference and used KNN classifier as the basis of accuracy.

2.1. L0 norm

Since true sparsity is represented by the L0 norm we tried to optimize the L0 norm problem of Dictionary Learning to reduce the dimensions. The number of dimensions we choose were 100. We used sklearn(Pedregosa et al., 2011) library to solve this problem. Since L0 norm problem is non-convex, we used the Orthogonal Matching Pursuits(OMP) Algorithm. We observed that we get an error rate of about 4.2 % with the 100 dimensional space in about 500 iterations. We also noted that KNN classifier's accuracy with the original MNIST dataset is 5% so we get a

better accuracy with the new dimensional space. Also ,run time complexity of KNN is O(d), we made the algorithm about 8 times faster for a better accuracy rate. However PCA gave an accuracy of 2.7% and solving PCA is much faster problem. So even though our problem is good, we needed improvements in speed and accuracy to make it useful.

Figure 1. MNSIT data representation through Dictionary Learning

2.2. L1 norm

Since we needed to increase the speed of computation it would be easier if we can solve a convex optimization problem. So we tried using L1 norm as sparsity measure to optimize the problem. Now we used Lasso and Lars optimizers to solve the problems. Even though we observed slight increase in speed, the objective was still converging after 500 iterations. So instead we used FISTA(Beck & Teboulle, 2009) to make it converge in 300 iterations. We observed that when we go to 500 iterations FISTA gives a slightly better error rate of 3.87%.

2.2.1. FISTA

Since FISTA has a higher convergence rate, we used the algorithm(Algorithm 1) on MNIST data.

2.3. Other sub-optimal Norms

We also tried Nuclear norm as we wanted to penalize W if it had higher dimensions. We observed a slightly better error rate of 3.57%. W also tried to put an L2 regularization on dictionary to avoid overfitting and got an accuracy of 3.67%.

2.4. Best Solution

The best solution we found out was using an Elastic Net(Zhou et al., 2011) instead of solving the Lasso problem on W. This means that W instead of D was being overfit. We got an error rate of 2.97% which is fairly close to that of PCA. Combined with FISTA it converges even faster. Hence we have solved both the problems of Speed and accuracy using this approach

Algorithm 1 FISTA

Input: W, b, λ, maxIter
x=Array of shape W.columns X b.columns
obj = []
$t = 1$
z = copy(x)
$L = ||W||_2^2$
$t_{old} = time()$
for $i = 1$ **to** $maxIter$ **do**
  $x_{old} = copy(x)$
  $z = z + W^T(b - Wz)$
  $x = SOFT - THRESHOLDING(z, \lambda/L)$
  $t_{new} = t$
  $t = (1. + \sqrt{1. + 4.0 * t^2})/2$
  $z = x + ((t_{old} - 1.)/t * (x - x_{old}))$
  $obj_{new} = 0.5 * ||Wx - b||_2^2 + \lambda||W||$
  $obj.add((time - t_0, obj_{new}))$
end for

3. Manifold Learning

Manifold learning(Gu & Xu, 2006) is a technique for non-linear dimensionality reduction, as opposed to PCA, which is a linear dimensionality reduction. It helps in visualising data which might have many dimensions, by trying to find the lower dimensional manifold which is present in the higher dimensional data.

After comparing the Isometric Mapping, Locally Linear Embedding and t-SNE, we found that the best results are obtained from t-SNE.

3.1. t-SNE

t-distributed Stochastic Neighbor Embedding(t-SNE)(Maaten & Hinton, 2008) converts the similarity of the data to probabilities. It's computationally intensive, and takes the most time compared to the other embedding models. It tries to learn the joint probability distribution of the data distribution in the higher dimensional space and then tries to replicate that in the low dimensional space by optimizing the KL divergence using Stochastic Gradient.

We learned how the clusters get formed in the final output of t-SNE by making a GIF of the training process. To do this, we had to patch the gradient descent method of tsne in scikit library, by adding code which could save the intermediate positions of the points, which was later converted into a GIF.

It was interesting to note that t-SNE has overlapping clus-

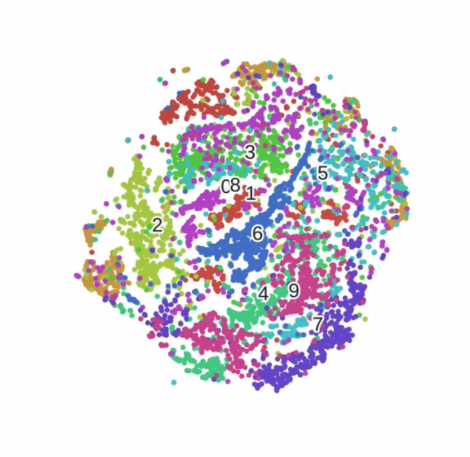

Figure 2. Manifold Learning on 10% MNIST data

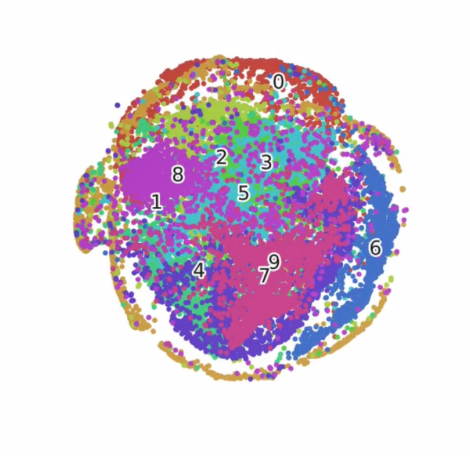

Figure 3. Manifold Learning on 30% MNIST data

ters of digits that actually look a little similar, like 4 and 9, and 0 and 8. Digits which don't look similar are as expected quite far apart from each other in the t-SNE result.

4. Results and Interpretation

The results obtained by using various classifiers are captured in the following table:

Classifier	Error
KNN	4.1 %
Linear SVM	14.81 %
Gaussian Naive Bayes	16.88 %
Multi-Layer Perceptron	7.95 %

We can compare the results we have obtained with the state of the art error rates obtained on these classifiers without using sparse coding. On the basis of the results, we can infer that the neighborhood property of the data points in being maintained as the KNN error was very low. The Gaussian Naive Bayes Classifier also gave about the same error, which indicates that the distribution of the data is also being preserved. However, we get a higher error rate for the Linear SVM classifier, which suggests that the linearity of the data is not being maintained.

5. Future Work

We want to work on making dictionary learning invariant to affine transformations. On the AffNist dataset, which is MNIST data on which various affine transformations have been applied, our error rate is 30%, but this can be reduced using hierarchical dictionary learning. We plan to apply the manifold learning algorithms on a more challenging dataset like the CIFAR-10 image dataset. It will be interesting to see which of the embedding models works best with that dataset. We have applied TSNE-visualization on the sparse coded vectors returned after dictionary learning. It will be interesting to compare these manifolds with the ones created on the normal dataset.

6. Conclusion

One of the observations is that Orthogonality of components is important for dimensionality reduction as PCA does better in most cases. However we see that with enough tuning parameters and change in Objective functions we can make dictionary learning perform better.

We observed that the clusters formed by t-SNE become less and less clear as the we include more points from the dataset. We think this happens because as we add more data, we're adding more noise to the dataset as well, thereby reducing the distance between various digits in the lower dimensions.

Acknowledgements

The authors would like to thank Dr. Lior Horesh and Dr. Julie Novak for providing insightful suggestions about the project, especially the suggestion to learn the manifold that the data lives upon and to make our dictionary learning process invariant to affine transformations.

The work is presented as a part of the final project for the course COMS 4772: Advanced Machine Learning in the Spring semester of 2017.

References

Beck, Amir and Teboulle, Marc. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1):183–202, 2009.

Gu, Rui-jun and Xu, Wen-bo. An improved manifold learning algorithm for data visualization. In *Machine Learning and Cybernetics, 2006 International Conference on*, pp. 1170–1173. IEEE, 2006.

LeCun, Yann, Cortes, Corinna, and Burges, Christopher JC. Mnist handwritten digit database. *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, 2, 2010.

Maaten, Laurens van der and Hinton, Geoffrey. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

Mairal, Julien, Bach, Francis, Ponce, Jean, and Sapiro, Guillermo. Online dictionary learning for sparse coding. In *Proceedings of the 26th annual international conference on machine learning*, pp. 689–696. ACM, 2009.

Pedregosa, Fabian, Varoquaux, Gaël, Gramfort, Alexandre, Michel, Vincent, Thirion, Bertrand, Grisel, Olivier, Blondel, Mathieu, Prettenhofer, Peter, Weiss, Ron, Dubourg, Vincent, et al. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(Oct):2825–2830, 2011.

Zhou, Tianyi, Tao, Dacheng, and Wu, Xindong. Manifold elastic net: a unified framework for sparse dimension reduction. *Data Mining and Knowledge Discovery*, 22(3): 340–371, 2011.