CS230

salient-SNE interprets non-linear clustering

Ethan S. Dyer
Department of Physics
Stanford University
edyer@stanford.edu

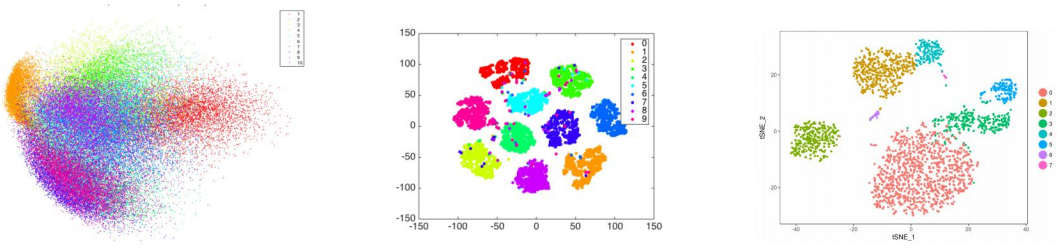
Karthik A. Jagadeesh
Department of Computer Science
Stanford University
kjag@stanford.edu

Abstract

Various techniques such as principal component analysis (PCA) exist to reduce the dimensionality of large sets of points. PCA falls short trying to tease structure out of non-trivial high-dimensional data sets. t-SNE, a non-linear reduction method, is effective at separating many high dimensional datasets, but it is not well suited for on-line processing of additional data points and the results of t-SNE are not easily interpretable in terms of meaningful aspects of the original data. We introduce salient-SNE, a parametric t-SNE approach, to address these challenges.

1 Introduction

Dimensionality reduction is a common technique used to visualize and interpret datasets. Various linear and nonlinear techniques exist to reduce the dimensionality of large sets of points. PCA is frequently used to linearly transform data by identifying the principal components (PCs) in the direction of the largest variance in the data. PCA has the benefit that it is easily interpretable, as we can understand how each PC weights the input features. Though linear reduction techniques such as PCA are transparent, they fall short when trying to tease structure out of non-trivial high-dimensional data sets. t-SNE is a popular non-linear reduction method that has been shown to perform well at separating many types of data [1,7,8].



Background on t-SNE: t-SNE converts high dimensional data into a matrix of pairwise similarities and aims to capture and transform both the local and global structure inherent in the data into a low dimensional space. The dimensionality reduction is accomplished by first randomly assigning a low dimensional representation of each point and creating a second matrix of pairwise similarities between all points based on this low dimensional representation. t-SNE then iteratively updates the low dimensional representation of the points to minimize the KL divergence between the two matrices of pairwise similarities from the input and output space [1].

There are a few challenges with t-SNE making it difficult to use for proper in depth analysis. First off, t-SNE, in its original incarnation, computes the embedding map as a global function of all data

CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

points. As such, it is not well suited to on-line processing of data points, or addition of additional data. Second, the results of t-SNE are not easily interpretable in terms of meaningful aspects of the original data. Indeed, there are many data sets for which t-SNE provides beautiful clusters, but for which the salient features driving the clustering remain obscure. We have built a parametric t-SNE [2] implemented using deep neural networks and will combine this design with saliency maps to understand the set of features driving clustering.

2 Related work

2.1 Neural Network approaches to t-SNE

van der Maaten has developed a parametric approach to t-SNE using neural networks [2]. This approach uses a pre-training step similar to salient-SNE (our method) but is completely unsupervised without looking at the class labels for the input images used during pre-training. This approach does not make an attempt to analyze the saliency maps to see that the model learns to see if they are meaningful representations. When we recreated a similar unsupervised network, we found that the saliency maps were very noisy.

Min et. al. have developed a supervised neural network approach resulting in improved clustering of MNIST images [3]. This approach first requires the network to be pre-trained using supervised learning followed by a unsupervised non-linear clustering training phase similar to the parametric t-SNE described previously. We improve on this method by using modern activation functions at each hidden layer and more recently devised gradient descent strategy (adam) along with the additional saliency map interpretations.

2.2 Saliency Mapping

Simonyan et al. introduced a neat method to visualize image classification models using saliency maps [4]. In this paper, they describe the process for computing the $\frac{\partial S_c}{\partial I}$, the derivative of the class score with respect to input image to identify how the input image features impact the class score. We also employ this strategy to visualize the saliency maps for our pretrained model. This approach cannot be easily transferred over in an unsupervised setting because there is score that is being calculated per class. We modify this formulation slightly to compute the "directional" saliency map betwen a point and the center of a target cluster.

2.3 Adversarial Attacks

Szegedy et al. introduce the notion of adding subtle and carefully chosen noise to an image to generate an adversarial example such that a classifier misclassifies the perturbed image as a different class [5]. The approach taken to build an adversarial example highly resembles the approach we take in salient-SNE to identify the "directional" saliency. To prevent the "directional" saliency maps from resembling noise, we constrain the model to be overly simplistic (a single hidden layer) ensuring that the saliency maps we observe are interesting.

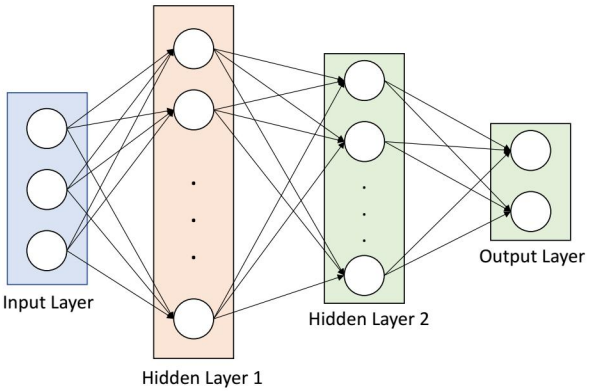
Elsayed et al. take adversarial examples one level further by trying to build noise that not only confuses a classifier but also confuses humans [6]. They are able to learn "meaningful" noise that results in 10 different trained models to incorrectly classify. The intuition is that a single classifier can be fooled due to specific weights and parameters the model has learned, 10 different classifiers will all have slightly different parameters and as a result the noise confusing a single classifier wont work on all 10. salient-SNE can incorporate some of these methods to reduce the noise that the "directional" saliency map learns.

3 Dataset and Features

The publicly available MNIST dataset [7] was used to evaluate the performance and interpretability of salient-SNE. MNIST contains 65,000 numbers ranging from 0 to 9. 55,000 images are part of the training set and 10,000 images are part of the test set used to evaluate the model. Each 28X28 image from MNIST was converted into a vector of 784 entries and used as the input into the fully connected neural network. No other transformations or modifications were performed on the data.

4 Methods

The salient-SNE architecture is composed of an input layer, 2 hidden layers with 100 and 10 hidden units each and an output layer with 2 hidden units. We employed a hybrid supervised and unsupervised approach to cluster and learn interpretable saliency maps of the data. The model is simple by design to ensure the network doesn't overfit to the training data. This ensures that the saliency we observe later are not noise.



4.1 Pre-train a 1 hidden layer network

As a first step, we built a fully connected network with a single hidden layer with 100 hidden units and a softmax output layer with 10 units. The labels from all training data points were used to train the network using the standard softmax loss function (defined by σ_j and L). The cost (J) is calculated after each iteration by summing the losses over all samples included in the mini batch and is used to perform the back propagation and update all the weights.

$$\sigma(z_j) = \frac{e^{z_j}}{\sum_{k=1}^{m} e^{z_k}}, \quad L(z_j) = y \cdot -\log(\sigma(z_j)), \quad J_{\text{softmax}} = \sum_{j=0}^{n} L(z_j)$$

4.2 Fine tune using t-SNE loss function

Then, we modify the previous network and add an additional fully connected output layer consisting of 2 units which takes as input the activations from the output layer of the pre-trained network. The output layer activations for all data points in the mini batch are converted into a similarity matrix Q and the input data points are converted into a similarity matrix P. The similarity matrix P is computed by fitting a gaussian distribution to each input data point and measuring the probability of observing every other data point given this distribution. The similarity matrix Q is computed by fitting a student t-distribution to each point and measuring the probability of observing every other point in the output space. The cost is computed as the KL divergence $KL(P||Q)$ and the result is used to compute the back propagation and fine tune the weights learned during the pre-training step.

$$P(i|j) = \frac{exp(-||x_i - x_j||^2/2\sigma_i^2)}{\sum_{i \neq k} exp(-||x_i - x_k||^2/2\sigma_i^2)}$$

$$P_{i,j} = P(i|j) + P(j|i)$$

$$Q_{i,j} = \frac{(1 + ||y_i - y_j||^2)^{-1}}{\sum_{i \neq k}(1 + ||y_i - y_k||^2)^{-1}}$$

$$J_{\text{t-SNE}} = KL(P||Q) = \sum_{i \neq j} P_{i,j} \cdot \log \frac{P_{i,j}}{Q_{i,j}}$$

4.3 Interpret the salient-SNE model

As a final step, to interpret the model, we compute $\frac{\partial Y}{\partial X}$ to measure how changes in the input will affect the output representation. We then compute a vector $\vec{v}_{a,b}$ as defined below to find the direction between the center of two clusters and compute $\frac{\partial Y}{\partial X}\Big|_a \cdot \vec{v}_{a,b}$ to first evaluate the derivative at center of the source cluster and multiply by the vector $\vec{v}_{a,b}$ to identify a "directional saliency" in the direction of the target cluster.

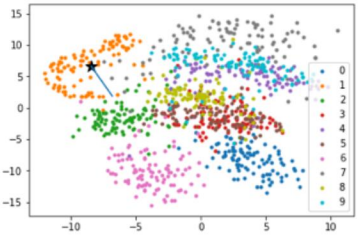
$$\vec{v}_{a,b} = a \text{ - } b$$

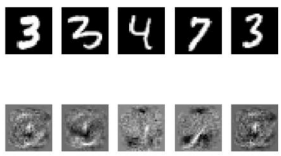
$$salient\text{-}SNE(a \rightarrow b) = \frac{\partial Y}{\partial X}\Big|_a \cdot \vec{v}_{a,b}$$

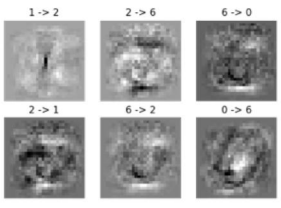
5 Experiments/Results/Discussion

5.1 Unobserved points effectively map to low dimensional space

We first assess the effectiveness of the trained salient-SNE model by mapping a set of previously unobserved high dimensional test data points onto the low dimensional space. The different digits align into mostly distinct clusters of points excepting 3 and 5 which form an overlapping cluster and 4 and 9 which form another overlapping cluster. This is not surprising and 3 and 5 have many overlapping pixels and the 4 and 9 are very similarly shaped as well. The clusters are not as cleanly separated as observed in the original t-SNE model, but this may be improved by training the model for more epochs and expanding the hyper parameter tuning.







5.2 Learning saliency mapping during pre-training phase

In order effectively interpret why the model is forming clusters, we need to first ensure that the model has learned a meaningful saliency map for input images. We computed the saliency map for a set of random images (see figure above). The saliency map for each input image is different as expected, even for images within the same class label. Well formed digits, such as the first three, have a easily discernible saliency map while more morphed digits such as the four are harder to distinguish. The

saliency maps are pointing to the pixels in the input image that are the most indicative of the correct class label.

5.3 Learning a "directional" saliency map

To understand the input features that are influencing salient-SNE to place a input point in one cluster compared to another, we calculated a "directional" saliency map measuring the input features that most influence a change from the center of one class to the center of another class. For the purpose of this test, we only measured the differences between adjacent clusters, as clusters further apart would result in noise due to interference.

For each directional map (see figure above), we computed the a vector $\vec{v}_{a,b}$ from the source image a to the target image b, which is typically the center of the cluster of points for the target class. As an example, changing the input image 1 to a canonical 2 requires down-weighting a vertical line of pixels at the center of the image and up-weighting pixels around the center (in the shape of a 2). Indeed, this is somewhat similar to what we observe in the "directional" saliency map (see figure above). Similarly, changing a 0 to a 6 requires upweighting a horizontal line of pixels near the center and down-weighting pixels at the top right of the image. This is clearly seen in the "directional" saliency map shown above.

6 Conclusion/Future Work

These results show promise, but there is still much room for improvement. One recently mentioned approach to improve the clarity of saliency maps is to use integrated gradients [9]. Integrated gradient average the gradients of an image over a change set of α brightness values. This will allow the saliency map to capture the differing features that are distinguishing at different brightness levels. This same approach can be applied to salient-SNE to build a better directional" saliency.

Another recent paper [6] describes a method to develop "noise" that can perturb an image to not only fool a classifier but also fool humans. The strategy to generate noise that can fool humans is to generate noise that will fool 10 different models. This indicates that the noise isn't specifically designed to the weights of any one model. This same strategy can perhaps be used to generate cleaner "directional" saliency maps that have less noise.

7 Contributions

E.S.D. and K.A.J. worked together to perform the experiments (pretrain, finetune, and saliency analysis) and write the manuscript. The code is available at https://github.com/karthikj89/salient-SNE.

References

[1] Maaten, Laurens van der, and Geoffrey Hinton. "Visualizing data using t-SNE." Journal of machine learning research 9.Nov (2008): 2579-2605.

[2] Van Der Maaten, Laurens. "Learning a parametric embedding by preserving local structure." RBM 500.500 (2009): 26.

[3] Min, Martin R., et al. "Deep supervised t-distributed embedding." Proceedings of the 27th International Conference on Machine Learning (ICML-10). 2010.

[4] Simonyan, Karen, Andrea Vedaldi, and Andrew Zisserman. "Deep inside convolutional networks: Visualising image classification models and saliency maps." arXiv preprint arXiv:1312.6034 (2013).

[5] Szegedy, Christian, et al. "Intriguing properties of neural networks." arXiv preprint arXiv:1312.6199 (2013).

[6] Elsayed, Gamaleldin F., et al. "Adversarial Examples that Fool both Human and Computer Vision." arXiv preprint arXiv:1802.08195 (2018).

[7] LeCun, Yann. "The MNIST database of handwritten digits." http://yann. lecun. com/exdb/mnist/ (1998).

[8] Satija, R. "Seurat: R toolkit for single cell genomics." (2016).

[9] Sundararajan, Mukund, Ankur Taly, and Qiqi Yan. "Axiomatic attribution for deep networks." arXiv preprint arXiv:1703.01365 (2017).