Deep Spectral Clustering

by Chieh, Khan

Deep learning has achieved some of the most amazing results in supervised machine learning. It does suffer from some minor issues, e.g. interpretability, require a lot of data, overfitting, etc. In spite of these problems, it regularly achieve superior results in the supervised setting.

Changing the domain into unsupervised settings, namely in clustering problems the benefits of utilizing neural networks is less obvious. Historically, clustering in neural networks has been performed through the strategy of combining dimensionality reduction and clustering. This is a well established approach in classical clustering in that we often perform PCA/LLE to reduce the dimensionality of the data prior to some clustering algorithm. In neural networks, this dimensionality reduction is done through the usage of autoencoders.

In 2012, in the paper by Baldi :

> Autoencoders, unsupervised Learning and Deep Architectures

The paper discussed the connection between autoencoders and clustering.

In 2013, in the paper by Song et al :

> Autoencoder based Data Clustering

The paper proposed to 1st perform autoencoder and then some sort of clustering algorithm.

In 2014, in the paper by Tian et al :

> Learning Deep Representation for Graph Clustering

The paper provided theretical reasoning why autoencoders + Kmeans is better than spectral clustering.

In 2015, in the paper by Chen :

> Deep Learning with Non-parametric clustering

The paper used DBN instead of autoencoders to perform dimensionality reduction. Once in the feature speace, they designed a graphical model, and used Gibb's sampling to get the posterior.

The idea of performing clustering after dimesionality reduction is such a dominant strategy, that even at 2017, in the paper by Dilokthanakul.

> Deep unsupervised clustering with Gaussian mixture auto encoders

This paper is still using the idea of autoencoders.

There was 1 paper in 2016 ICML that deviated from autoencoders by Xie et al :

Unsupervised Deep embedding for clustering analysis

Instead of using autoencoders, they used a self-training approach. Here are the steps of their algorithm.

1. Perform regular clustering

2. Use the most confident data sets to model a stribution p

3. Use DNN to discover a mapping f_θ from data to feature space with a distribution q such that the $\mathrm{KL}(p||q)$ is minimized.

4. Repeat this process with more data.

The idea of this paper is to use the most confident points as an approximation to the truth data distribution. Given an approximation, DNN is used to bias the data towards the distribution of the most confident data sents. This type of self training strategy allows for the neural network to optimize an objective function of $\mathrm{KL}(p||q)$.

As observed, the idea of performing some sort of dimensionality reduction prior to clustering has been the predominant strategy. Since autoencoders perform dimensionality reduction for neural networks, its usage has been obvious. However, as we know from performing PCA prior to clustering. The compression of data doesn't always yield a good clustering. Instead, it is better to discover a lower dimensional representation of the data that are contrained simultaneously to yield a good clustering result.

To achieve this objective, we introduce the following formulation.

$$\max_{U,\,\varphi(.)} \mathcal{R}(\varphi(X;W),U) \qquad s.t \qquad U^TU = I$$

Let $\mathcal{R}(A,B)$ be a function that measures the relationship between two datasets A and B. This function could be replaced by any similarity measure such as mutual information, correlation, or HSIC. Let the input data be $X \in \mathbb{R}^{n\times d}$, where n is the number of samples and d the number of features. Let U be the corresponding clustering of X. The formulation above wishes to discover a mapping of X and a clustering solution U such that the relationship between $\varphi(X)$ and U is maximized. We let $\varphi(X;W)$ be a neural network with W as all of its weights. The optimization process therefore iterates between optimizing the weights of the neural networks and the clustering U.

For the purpose of this paper, HSIC was used as a relationship measure. (List the reasons why HSIC is an ideal choice). Therefore the formulation could be rewritten as

$$\max_{U,\,\varphi(.)} \mathrm{HSIC}(\varphi(X;W),U) = \max_{U,\,\varphi(.)} \mathrm{Tr}(K_{\varphi(x_i),\varphi(x_j)}HK_UH).$$

Where H is the centering matrix, K_U is a linear Kernel of U where $K_U = UU^T$ and $K_{\varphi(x^{(i)})\varphi(x^{(j)})}$ is another kernel perform on the data after $\varphi(.)$. Since U represents labels, it is reasonable to use the linear kernel to represent this data type. However, the kernel suitable for $K_{\varphi(x^i)\varphi(x^{(j)})}$ depends on the data itself. To prevent the explosion of the objective function, we used a Gaussian kernel to normalize the values to between 0 to 1.

Assuming that the kernels are centered, the objective function can be written in the formulation of spectral clustering.

$$\mathrm{Tr}\Big(K_{\varphi(x^i)\varphi(x^{(j)})}UU^T\Big) = \mathrm{Tr}\Big(U^TK_{\varphi(x^i)\varphi(x^{(j)})}U\Big)$$

Therefore, there is a strong relationship between finding a high HSIC relationship and a good clustering quality in the spectral clustering sense. In spectral clustering, the concept of normalized Kernel allows for a better clustering results. The kernel is normalized by dividing by the Degree Matrix D. Adding the Degree matrix into the formulation, the new objective function becomes.

$$\begin{array}{ll} \max\limits_{\varphi(.),U} & \mathrm{Tr}\Big(D^{-1/2} K_{\varphi(x^i)\varphi(x^{(j)})} D^{-1/2} H K_U H \Big) \\ s.t & U^T U = I \end{array}$$

A basic spectral clustering is perform at the 1st iternation by using the Gaussian kernel. This allows us to discover K_U and H. If we let each row of X be $x^{(i)}$, $x^{(i)}$ is first pass through a neural network of l layers with each layer having h hidden nodes. For the sake of simplicity, each hidden layer have the same number of nodes. The output of inputing $x^{(i)}$ is $\varphi\big(x^{(i)}\big)$. Passing all samples through the neural network, we get $X \to \varphi(X)$. With this, we have mapped the original dataset to another space. In this new space, we calculate the Gaussian Kernel,

$$k\big(\varphi\big(x^{(i)}\big), \varphi\big(x^{(j)}\big)\big) = e^{-\frac{\left(\varphi\left(x^{(i)}\right) - \varphi\left(x^{(j)}\right)\right)^2}{2\sigma^2}}.$$

Once $K_{\varphi(x^i)\varphi\left(x^{(j)}\right)}$ is computed, the Degree matrix is computed from the Kernel with

$$D_{j,j} = \sum_{i=1}^{d} K_{i,j} \quad , \quad D_{i \neq j} = 0$$

The objective function, therefore, can be view in terms of the weights of the neural networks, W.

$$\begin{array}{ll} \max\limits_{W,U} & \mathrm{Tr}\big(D^{-1/2} K(W) D^{-1/2} H K_U H \big) \\ s.t & U^T U = I \end{array}$$

This objective can be optimized by iterating the optimization between U and W. Using the weights W previous computed, U can be calculated using eigen-decomposition.

$$\begin{array}{ll} \max\limits_{U} & \mathrm{Tr}\big(U^T H D^{-1/2} K(W) D^{-1/2} H U \big) \\ s.t & U^T U = I \end{array}$$

$$\begin{array}{ll} \max\limits_{U} & \mathrm{Tr}(U^T \Phi U) \\ s.t & U^T U = I \end{array}$$

Given U, W can be computed using backward propagation. Because this is a highly non-convex problem, and back propagation uses gradient descent, the proper initalization point heavily determines the both the quality of the clustering and the speed of convergence. For this reason, we recommend to use the following archetecture and weights for the intialization state. Assuming each sample $x^{(i)}$ has 2 features, x_1 and x_2. The initialization weights are set up such that the input is very close to the output. From this perspective, the initialization is the result of spectral clustering itself. From this point, the gradient descent further improves the objective function.

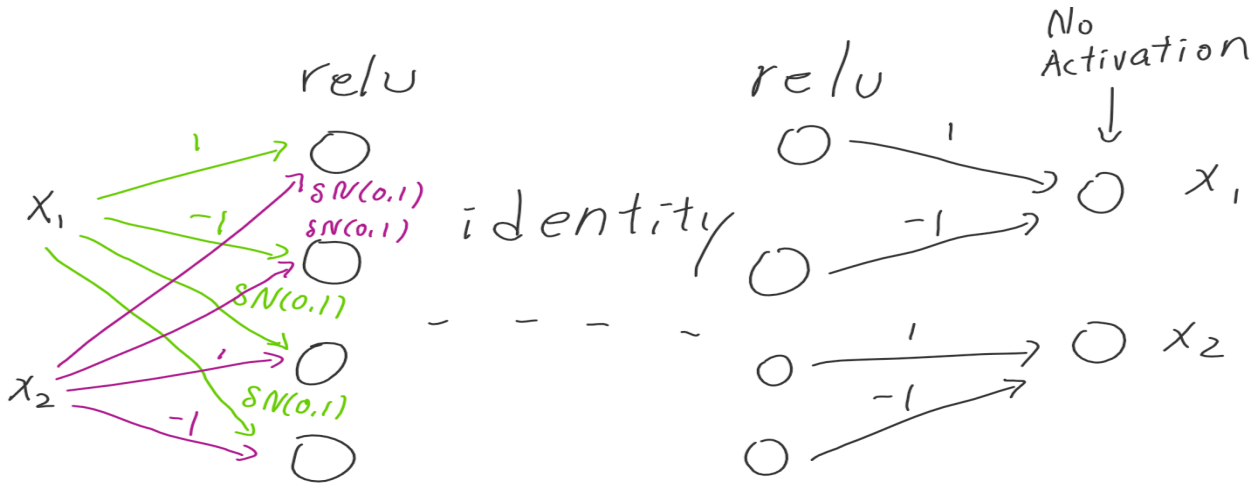
relu
relu
No Activation
identity
x_1
x_2
1
-1
SN(0,1)
SN(0,1)
SN(0,1)
SN(0,1)
1
-1
1
-1
x_1
x_2

Figure 1.