Feature Learning

MInDS @ Mines

Dimensionality reduction is a crucial part of machine learning when dealing with high dimensional data. Feature selection is a simple approach to working with the available data. Feature extraction can get more complicated where we are transforming the feature space into a new one. We will cover several feature extraction methods including t-distribution stochastic neighbor embeddings (t-SNE), spectral embeddings, and linear discriminant analysis.

Dimensionality reduction is a common solution to deal with the problems that arise due to the curse of dimensionality. Feature extraction is one approach to dimensionality reduction where we transform the feature space into a new space that achieves the goals we aim to achieve by reducing the data's dimensionality. There are several methods for feature extraction with varying goals in mind.

t-distributed Stochastic Neighbor Embeddings

The t-distributed Stochastic Neighbor Embeddings (t-SNE) method is an embedding approach that transforms high dimensional data into a lower dimensional space. t-SNE is an unsupervised learning method with the goal of maximizing the information described in each dimension. t-SNE is often used for visualizing high dimensional data. t-SNE is a version of Stochastic Neighbor Embeddings (SNE) that uses the Student-t distribution instead of a Gaussian distribution. We won't cover SNE specifically and will just skip to t-SNE.

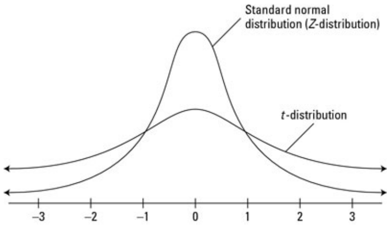

Figure 1: The Student-t distribution compared to the normal distribution. The t-distribution has a "fatter tail" which allows us to model more extreme events as more likely than a normal distribution. With regards to SNE, this allows the model to more accurately detect variations in data with small Euclidean distances between them.

t-SNE computes the similarity between two points as the probability that a point x_j would be selected as a neighbor of point x_i, had x_i been the center of the distribution. $p_{j\mid i}$ is this similarity, which is the probability of x_j being a neighbor, given that x_i is the center of the distribution. This similarity metric is therefore calculated as,

$$p_{j\mid i} = \frac{\mathsf{exp}(-||\mathsf{x}_i - \mathsf{x}_j||_2^2/2\sigma_i^2)}{\sum_{k\neq i}\mathsf{exp}(-||\mathsf{x}_i - \mathsf{x}_k||_2^2/2\sigma_i^2)}, \tag{1}$$

where σ_i^2 is the variance of the Gaussian distribution centered at x_i. The general idea here is that we are comparing the probability of x_j with all the probabilities of the other points, assuming a Gaussian distribution. The similarity to x_j is therefore a relative calculation compared to all the other points.

We then set the pairwise similarity between two points, x_i and x_j, to be,

$$p_{ij} = \frac{p_{j\mid i} + p_{i\mid j}}{2n}, \tag{2}$$

where n is the total number of samples in the data.

We represent the learned embedding, the data in the transformed space, for point x_i as y_i where x_i is in d dimensions and y_i is in r dimensions, with $r < d$. The goal of t-SNE is to learn Y, the transformed data from X, such that we minimize the relative entropy between the distributions of the data in each space. The objective is,

$$\min_{\mathsf{Y}} \sum_{i \neq j} p_{ij} \log \left(\frac{p_{ij}}{q_{ij}} \right), \tag{3}$$

where q_{ij} is the joint distribution of the two points in the embedding's space assuming it follows a Student-t distribution,

$$q_{ij} = \frac{(1 + ||\mathsf{y}_i - \mathsf{y}_j||_2^2)^{-1}}{\sum_{k \neq i}(1 + ||\mathsf{y}_i - \mathsf{y}_k||_2^2)^{-1}}. \tag{4}$$

We set $p_{ii} = 0$ and $q_{ii} = 0$.

Here we have two sets of probabilities for each 2 points, p_{ij} in the original space and q_{ij} in the embedding's space, with the idea being that the information described by each is as close as possible to each other. The relative entropy is minimized.

This completes the definition of t-SNE with regards to its theoretical approach, however one portion that we skipped over was the σ_i^2 selected when calculating $p_{j \mid i}$. Its value is effectively a hyperparameter however, not directly. Instead of specifying the variance of the distribution, we specify the perplexity value which is a measure of the distribution's sample prediction effectiveness. We can calculate perplexity as,

$$\text{Perplexity}(P_i) = 2^{\mathcal{H}(P_i)} = 2^{-\sum_j p_{j \mid i} \log p_{j \mid i}}, \tag{5}$$

where $\mathcal{H}(P_i)$ is the entropy of the distribution. With respect to t-SNE, perplexity is used to represent the number of neighbors to consider per point. We can think of a higher perplexity as a higher weight to the global relationships and a lower perplexity as a higher weight to the local relationships.

To determine the embeddings that minimize the objective for t-SNE we use gradient descent. This leads to an additional parameter of a learning rate as well as a number of iterations. ”How to Use t-SNE Effectively” [1] shows the various issues that can come up with t-SNE and how to get around them. For using t-SNE to visualize data, the following issues exist:

- Hyperparameters in t-SNE are crucial
- Cluster sizes are meaningless
- Distance between clusters can be meaningless
- Random noise may result in some misleading patterns
- Shapes may appear in the output that are misleading
- Testing multiple perplexities may help with understanding the data

t-SNE can be a very effective dimensionality reduction method but it may also be difficult to use and requires more effort to get right.

[1] Martin Wattenberg, Fernanda Viégas, and Ian Johnson. How to use t-sne effectively. *Distill*, 2016

Spectral Embeddings (or Laplacian Embeddings)

Spectral Embeddings, or Laplacian embeddings, are unsupervised embeddings learned from the manifold of the provided data. The goal of the Spectral Embeddings is to provide a representation in a lower dimension space where the distances between any two points is weighted by the similarity between the points in the original space.

We can represent data using a manifold where we provide the similarity or distances between every two points to construct a graph. Using this graph we can calculate three matrixes; the similarity matrix S, the diagonal degree matrix D and the Laplacian matrix L. The similarity matrix S is an $n \times n$ square matrix representing the similarity between every pair of points where, s_{ij} is the similarity between points x_i and x_j. This similarity can be calculated by using a distance metric or by determining some intrinsic relationship between the datapoints. The diagonal degree matrix has,

$$d_{ii} = \sum_{j=1}^{n} s_{ij}. \qquad (6)$$

Note that similarity between two points is $\frac{1}{\text{Distance}}$. We often normalize the similarities such that a distance of 0 is a similarity of 1, and therefore, all $s_{ii} = 1$.

The value in the degree matrix represents the connectedness of a point to the rest of the graph. Finally, the Laplacian matrix $\mathsf{L} = \mathsf{D} - \mathsf{S}$.

The goal with Spectral Embeddings is to learn an embedding y_i for each data point x_i such that,

$$\min_{\mathsf{Y}} \sum_{i=1}^{n}\sum_{j=1}^{n} s_{ij}||\mathsf{y}_i - \mathsf{y}_j||_2^2 \qquad (7)$$

That is to say that the goal is to minimize the distance in the embedded space weighted by the similarity between the points in the original space. In order to calculate the value of Y we determine the r eigenvectors of the Laplacian matrix corresponding to the smallest r non-zero eigenvalues.

When we covered Spectral Clustering, we mentioned how it is simply applying K Means clustering to the Spectral Embeddings. We also saw that spectral clustering was able to achieve superior results, especially when our data presented a manifold structure. With Spectral Embeddings, we can utilize the power of the learned representation with any method or application we select.

Linear Discriminant Analysis

Linear Discriminant Analysis (LDA) is a supervised method that can be used for classification as well as dimensionality reduction. Similar to other methods, we aim to determine a new representation for our data in a smaller dimension space. Different from other methods we've covered so far however is that LDA is a supervised method. First, we define the learned embedding for a particular point x_i in d dimensions as

$$\mathsf{y}_i = \mathsf{W}^T \mathsf{x}_i, \qquad (8)$$

where W is the $r \times d$ projection matrix by which we transform the original data from d dimensions into the new r-dimensional space.

LDA's goal is to determine an embedding that maximizes the discriminant ability or the class separation. LDA is therefore a supervised method since it requires us to know the target classes of each point. We determine the mean point for a particular class,

$$\mathsf{m}_k = \frac{1}{n_k} \sum_{i \in \mathcal{C}_k} \mathsf{x}_k, \tag{9}$$

where $\mathcal{C}_k$ is the set of all points belonging to class k. For simplicity, we will focus on the 2-class problem. In order to maximize the model's ability to separate the classes, we should consider two metrics; the between-class variance and within-class variance. Ideally we would like the variance between the classes to be large and the variance within each class to be small. We calculate the between-class covariance matrix as,

We can expand the model to a multi-class problem by applying it several times where we learn a new model for each class, treating all other classes as one.

$$\mathsf{S}_B = (\mathsf{m}_1 - \mathsf{m}_2)(\mathsf{m}_1 - \mathsf{m}_2)^T, \tag{10}$$

and the within-class covariance matrix as,

$$\mathsf{S}_W = \sum_{i \in \mathcal{C}_1} \left((\mathsf{x}_i - \mathsf{m}_1)(\mathsf{x}_i - \mathsf{m}_1)^T \right) + \sum_{j \in \mathcal{C}_2} \left((\mathsf{x}_j - \mathsf{m}_2)(\mathsf{x}_j - \mathsf{m}_2)^T \right). \tag{11}$$

The objective for LDA is given as,

$$\max_{\mathsf{W}} \mathbf{tr} \left(\frac{\mathsf{W}\mathsf{S}_B\mathsf{W}^T}{\mathsf{W}\mathsf{S}_W\mathsf{W}^T} \right). \tag{12}$$

This is consistent with our goal to maximize class separation in the embedding since we are maximizing the ratio[2] between the variance between classes in the new space and the variance within each class in the new space. A higher variance between classes in the new space increases our objective, and a lower variance within each class in the new space increases our objective.

[2] This ratio is called the Fischer criterion.

References

[1] Martin Wattenberg, Fernanda Viégas, and Ian Johnson. How to use t-sne effectively. *Distill*, 2016.