# t-SNE Example

*D2K Course Staff*

*February 17 2019*

## Introduction

One of them newfangled clustering methods that all the cool stat kids are using these days is called t-SNE, which stands for t-Distributed Stochastic Neighbor Embedding. The main idea of t-SNE is to visualize our data in a low-dimensional space (by default 2 dimensions) while maintaining as much of the original high-dimensional structure as possible. This is measured by comparing a similarity score for the relative pairwise distances between points in the original data and in the low-dimensional representation.

### Disadvantages

This setup does have a couple of disdvantages:

1. Results can depend on the random starting values of $Y_i$ chosen.
2. The method does not actually assign points to clusters, in the same manner that your basic k-means or your convex clustering methods would. The idea here is just to find the low-dimenisonal representation that best maintains relative pairwise distances between the points. If we want to get automatically assigned clusters, we could then run a clustering algorithm on the low-dimensional representation returned by t-SNE.

### Math Stuff

For those that care about this sort of stuff, the math behind measuring similarity can be found at:

https://lvdmaaten.github.io/publications/papers/JMLR_2008.pdf

As a quick and undetailed summary:

- We calculate the conditional probability that a point would call another point its "nearest neighbor" in the original dimensional space as proportional to the pairwise Euclidean distance between the points over the sum of all the distances between the points:

$$p_{j|i} \propto \frac{\exp(-\|x_j - x_i\|^2)}{\sum_k \exp(-\|x_k - x_i\|^2)}.$$

  This is done for all pairs of points.
- We calculate the conditional probability that a point would call another point its "nearest neighbor" in the low dimensional space in the same manner:

$$q_{j|i} \propto \frac{\exp(-\|y_j - y_i\|^2)}{\sum_k \exp(-\|y_k - y_i\|^2)}.$$

- We attempt to find the set of $y_i$ that minimizes the KL divergence between the $p_{j|i}$ and $q_{j|i}$, where

$$KL = \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}.$$

  Algorithmically, this can be done via gradient descent.

## Example

In this example, we will analyze the `wine` data set from the `rattle.data` package. We will be clustering assuming that we do not know the wine type, and we will attempt to see if we can find different groups of wines in the data, perhaps to see if we can find classifications to ensure similar taste or composition amongst all wines of the same labeled type.

```
library(rattle.data)
data(wine)
head(wine)
```

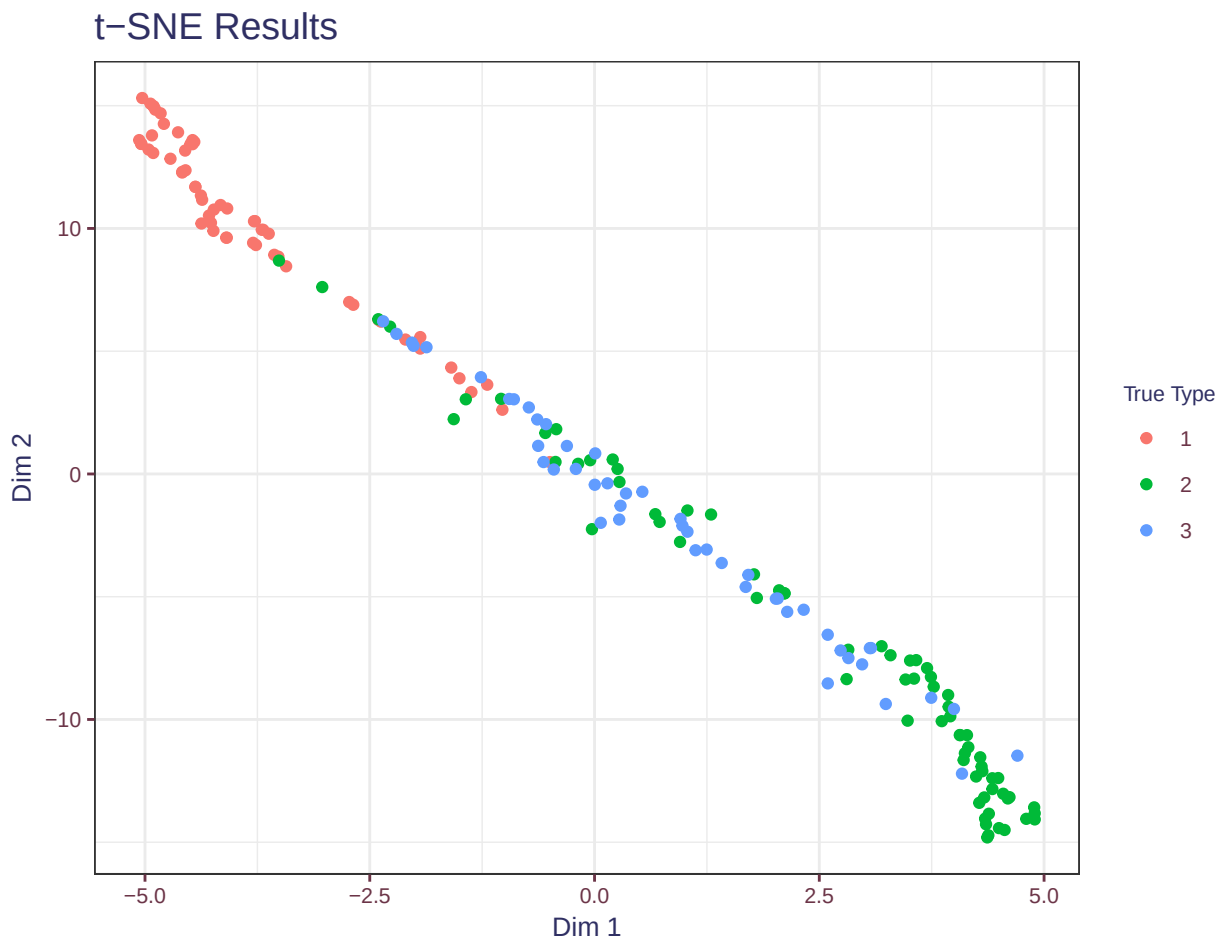
```
  Type Alcohol Malic  Ash Alcalinity Magnesium Phenols Flavanoids
1    1   14.23  1.71 2.43       15.6       127    2.80       3.06
2    1   13.20  1.78 2.14       11.2       100    2.65       2.76
3    1   13.16  2.36 2.67       18.6       101    2.80       3.24
4    1   14.37  1.95 2.50       16.8       113    3.85       3.49
5    1   13.24  2.59 2.87       21.0       118    2.80       2.69
6    1   14.20  1.76 2.45       15.2       112    3.27       3.39
  Nonflavanoids Proanthocyanins Color  Hue Dilution Proline
1          0.28            2.29  5.64 1.04     3.92    1065
2          0.26            1.28  4.38 1.05     3.40    1050
3          0.30            2.81  5.68 1.03     3.17    1185
4          0.24            2.18  7.80 0.86     3.45    1480
5          0.39            1.82  4.32 1.04     2.93     735
6          0.34            1.97  6.75 1.05     2.85    1450
```

We will use the R package `Rtsne` to run t-SNE on the data.

```
library(Rtsne)
```

```
set.seed(316)
mod1 <- Rtsne::Rtsne(wine[, -1])
```

```
ggplot() +
  geom_point(aes(x = mod1$Y[, 1], y = mod1$Y[, 2],
                 color = wine$Type)) +
  labs(title = "t-SNE Results",
       x = "Dim 1", y = "Dim 2",
       color = "True Type") +
  theme1
```




As we can see from these results, there is a pretty clear cluster for the Type 1 wine. However, Types 2 and 3 seem to be too similar to discern in to separate types.