SVM Binary Market Classification Model for t-SNE Generated Clusters of NREL Dataset: *Rooftop Solar Photovoltaic Technical Potential in the United States*

ChE 765 – Artificial Intelligence and Machine Learning Fundamentals

McMaster University, Hamilton, ON. Canada
L8S 4L7

August 16, 2020

By: Mohammad Akhtar, M.A.Sc. 2019 – 2021

Table of Contents

Abstract

Dimensionality reduction and visualization techniques *t-stochastic neighbour embedding* (t-SNE) and *uniform manifold approximation and projection* (UMAP) were used to evaluate National Renewable Energy Laboratory's (NREL) market segmentation for rooftop solar technical potential based on *small*, *medium*, and *large* classification labels. The *medium* and *large* class clusters were shown to overlap over a broad range of hyperparameter optimizations leading to the agglomeration of both classes and a revised dataset with binary classifications, *small* and *large*. T-SNE outputs in the low-dimensional embedded feature space were used as inputs for *support vector machine* classification algorithms. While the *polynomial* kernel trick performed poorly at classifying the distinct binary clusters, both the *radial basis function* and *sigmoid* kernel tricks precisely and accurately classified the new rooftop solar technical potential market segments.

1 Introduction

This report proposes a machine learning approach to a binary reclassification of National Renewable Energy Laboratory's (NREL) *Rooftop PV Technical Potential* dataset [1] for the purposes of solar photovoltaic market segmentation and differentiated market analysis. *T-distributed stochastic neighbour embedding* (t-SNE) and *uniform manifold approximation and projection* (UMAP) nonlinear feature extraction algorithms are used to visualize the multivariate dataset in 2 dimensions. T-SNE and UMAP hyperparameters are iteratively adjusted to generate clusters in the lower dimensional feature space. Distinct clusters are deemed to represent differentiated market segments, whereas overlapping clusters are aggregated and reclassified, therefore reducing the total number of classes (or segments) by $n - 1$.

2 Database Description

The original dataset defines a ternary segmentation or classification to categorise rooftop technical potentials based on building size, whereby *small*, *medium*, and *large* building classes have a footprint of $< 5{,}000$ ft^2, $5{,}000 - 25{,}000$ ft^2, and $> 25{,}000$ ft^2, respectively. Light detection and ranging (LiDAR) data, geographic information system (GIS) methods, and PV-generation modeling is used to calculate the rooftop solar photovoltaic potential for approximately 23% of all buildings in the United States [1]. For the purposes of this report, the NREL dataset is restructured as a class-wise matrix of 4000 observations per class with m total observations, n classes, and 10 features:

i. Total # of buildings in LiDAR
ii. Total rooftop area in LiDAR (m^2)
iii. # of buildings with available rooftop area (any size)
iv. Rooftop area available for all buildings (m^2)

v. # of buildings with available rooftop area (> 10 m^2)

vi. Rooftop area available for all buildings > 10 m^2 (m^2)

vii. Building potential capacity with rooftop area > 10 m^2 (kW)

viii. Building potential energy generation with rooftop area > 10 m^2 (kWh/yr.)

ix. # of buildings (in class) per total # of buildings (all classes) (%)

x. Rooftop area available (in class) per total rooftop area (all classes) (%)

3 Objectives

Original ternary class labels are reclassified as *small* and *large* building classes with a footprint of < 5,000 ft^2 and ≥ 5,000 ft^2, respectively. T-SNE and UMAP hyperparameters are iteratively optimized to generate visually distinguishable binary clusters for *support vector machine* (SVM) supervised machine learning classification algorithms. For the sake of brevity, the SVM classification algorithm is applied only to the optimized t-SNE output with the lowest *Kullback-Liebler* (KL) divergence. *Polynomial, radial basis function* (RBF), and *sinusoid* kernel tricks are used to accurately assess the revised simplified binary classifications. Together, t-SNE visualised clusters and SVM classifiers precision/accuracy metrics validate reclassified binary *small* and *large* solar photovoltaic market segmentation and differentiation based on rooftop area technical potential[1].

4 Methodology

4.1 Dimensionality Reduction for Data Visualisation

Feature extraction techniques are applied to the NREL rooftop solar photovoltaic technical potential multivariate dataset for data visualization in 2 or 3 dimensions. Preprocessing

[1] Herein, *original* dataset refers to the *ternary* NREL classification, and *revised* refers to the proposed *binary* classification used for SVM classification algorithms.

via standardisation and data densification improves the versatility and robustness of the original dataset. Linear dimensionality reduction technique, *principal component analysis* (PCA) acts as a baseline for class[2] cluster visualisation in *t-distributed stochastic neighbour embedding* (t-SNE) and *uniform manifold approximation and projection* (UMAP) non-linear algorithms.

4.1.1 Principal Component Analysis (PCA)

The original 10-dimensional dataset is projected onto a 2-dimensional hyperplane, maximizing global variance across a minimum number of principal component axes. Eigenvalue decomposition is performed on the covariance matrix of the standardised dataset. Resultant eigenvectors are sorted in order of decreasing eigenvalue. Data is projected onto a hyperplane of the first and second principal component axes, which hierarchically explain the maximum amount of variance in the dataset.

4.1.2 T-Distributed Stochastic Neighbour Embedding (t-SNE) [2][3]

t-SNE, unlike linear dimensionality reduction and visualisation algorithms such as PCA, is a feature extraction and visualisation technique well-suited for the interpretation of complex nonlinear manifold structures and polynomial relationships between features [3]. Whereas PCA seeks to maximize global variance by preserving large pairwise distance, t-SNE preserves small pairwise distances or local similarities, which allows it to capture non-linear structures [3]. T-SNE may be initialized using PCA for high-dimensional datasets (> 50), however and expectedly, it yielded no discernible benefit in the scenarios explored in this report.

The t-SNE algorithm improves on the *stochastic neighbour embedding* (SNE) algorithm by using an alternative optimizable cost function, which uses a symmetrized variation of the SNE cost function with simpler gradients and a *t-distribution* in place of a *Gaussian distribution* to

[2] Herein, *class* and *label* are used interchangeably.
[3] Unless otherwise cited, the information in this section is cited from Laurens van de Maaten's original paper.

measure similarities in the low-dimensional space. The algorithm can simply be described as, "the [minimization] of divergence between two distributions: a [Gaussian] distribution that measures pairwise similarities of the [high-dimensional] input objects and a [Student's-t] distribution that measures pairwise similarities of the corresponding low-dimensional points in the embedding" [4].

4.1.2.1 Hyperparameters

The *number of components* define the dimension of the embedded space. Since t-SNE is primarily a visualization algorithm the low dimensional feature space is limited to 2 or 3 dimensions. Squared Euclidean p*airwise distances* are employed in the high-dimensional and the low-dimensional space when calculating the distances between instances in a feature array [5]; alternative metrics such as Chebyshev's inequality were tested unsuccessfully to improve cluster separation for the original dataset. The remaining relevant hyperparameters can be separated into *cost function* and *optimization parameters*.

- Cost Function Parameter: *Perplexity* may be interpreted as the manifold learning algorithms' equivalent of effective number of nearest neighbours. Typical values are between 5 and 50, however this paper evaluates t-SNE outputs for perplexity values ranging from 5 to 400. It is defined as,

$$Perp\,(P_i) = 2^{H(P_i)} \quad [1]$$

  where $H(P_i)$ is the Shannon entropy of probability distribution, P_i measured in bits,

$$H(P_i) = -\sum_j p_{j|i} \log_2 p_{j|i} \quad [2]$$

- Optimization Parameters
  - T, *number of gradient iterations*: The cost function given by Equation (8) minimizes the gradient of the Kullback-Liebler divergence between high-dimensional Gaussian and low-dimensional embedded Student-t based (Cauchy distribution) affinities given

by Equation (7) as a measure of "the faithfulness with which q_{ij} models $p_{j|i}$. The number of iterations to minimize the cost function using a gradient descent method ranges from 500 to 5000. Higher number of iterations were observed to be computationally intensive, whereas lower number of iterations were computationally less expensive.

- *Early exaggeration*: The gradient descent method used to minimize the t-SNE cost function is optimized by multiplying all the high-dimension probabilities by a scalar value (e.g. 12) in the initial stages of optimization. This encourages the cost function to focus on "modelling the large p_{ij}'s by fairly large q_{ij}'s, which controls how tight natural clusters in the original high-dimensional space are in the embedded lower-dimensional space and how distant the clusters are from each other. Early exaggeration values ranging from 1 to 96 were tested with the expectation that larger values will visually optimize clear, distinct separations between natural clusters.
- η, *learning rate*: In the early stages of cost function optimization, Gaussian noise is introduced to the map points with each iteration. Therefore, gradually reducing the rate at which Gaussian noise decays drastically decreases the possibility of the cost function becoming stuck in a bad local minimum. The sparsity or density of natural clusters in the embedded space is affected by the learning rate, whereby too high or too low rates may result in any point becoming "approximately equidistant from it's nearest neighbours … [or] compressed in a dense cloud with few outliers", respectively []. Learning rates ranging from 10 to 1000 were tested, with the upper limit determined by the correlation [6],

$$\eta = \frac{m}{12} \qquad [3]$$

- α, *momentum*: Momentum is a scalar gradient multiplier intended to accelerate optimization to avoid poor local minima. This hyperparameter remained unexplored in this report. Optimization was restricted to standard values for iterations < 250 and ≥ 250,

$$\alpha^t = 0.5, t < 250$$

$$\alpha^t = 0.8, t \geq 250$$

- θ, *Barnes-Hut approximation threshold*: To accelerate compute times, the tree-based Barnes-Hut approximated was used to approximate the t-SNE gradient. θ in Equation (4) is "the angular size of a distant node as measured from a point" [5], which quantifies the speed/accuracy trade-off for Barnes-Hut t-SNE. Values of 0 and 0.2 were explored, with the lower threshold exponentially increasing computation times while proving inconsequential in improving accuracy for the revised dataset.

$$\frac{r_{cell}}{\|y_i - y_{cell}\|^2} < \theta \quad [4]$$

4.1.2.2 Simplified Algorithm

The t-SNE algorithm starts with an input data set in high-dimensional space,

$$X = \{x_1, x_2, \dots, x_n\}$$

where X represents the 10-dimensional multivariate NREL dataset.

Perplexity and optimization hyperparameters mentioned above are selected as inputs for an iterative cost function minimizing algorithm for the number of gradient descent iterations, *T*. The first step is to compute pairwise probabilities in the high-dimensional space employing the Gaussian kernel using Equation (5),

$$p_{j|i} = \frac{e^{-\frac{\|x_i - x_j\|^2}{2\sigma_i^2}}}{\sum_{k \neq i} e^{-\frac{\|x_i - x_k\|^2}{2\sigma_i^2}}} \quad [5]$$

The resultant matrix is symmetrized using Equation (6),

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n} \quad [6]$$

Y^0 represents the guessed initial condition for t-SNE outputs,

$$Y^0 = \{y_1, y_2, \dots, y_n\}$$

The initial solution is used to compute t-distributed low-dimensional affinities in the embedded space using Equation (7),

$$q_{ij} = \frac{\left(1 + \|y_i - y_j\|^2\right)^{-1}}{\sum_{k \neq l}(1 + \|y_k - y_l\|^2)^{-1}} \quad [7]$$

The cost function is approximated and iteratively optimized (i.e. Kullback-Liebler divergence minimized) using Equation (8),

$$\frac{\delta C}{\delta y_i} = 4 \sum_j (p_{ij} - q_{ij})(y_i - y_j)\left(1 + \|y_i - y_j\|^2\right)^{-1} \quad [8]$$

Points in the embedded space are updated in an iterative process using Equation (9) looping from the point of computing low-dimensional affinities executing until the maximum number of iterations is reached,

$$Y^t = Y^{t-1} + \eta \frac{\delta C}{\delta y} + \alpha(t)(Y^{t-1} - Y^{t-2}) \quad [9]$$

The resultant output in a 2 or 3-dimensional embedded feature space is given by,

$$Y^T = \{y_1, y_2, \dots, y_n\}$$

4.1.3 Uniform Manifold Approximation and Projection (UMAP)

T-SNE performed very well in clustering the revised *small* and *large* binary classifications. However, over 150+ combinations of hyperparameter inputs and optimizations failed to generate meaningful distances between the *medium* and *large* class clusters in the original dataset. As a result, a relatively novel non-linear dimension reduction algorithm, UMAP was implemented to determine whether t-SNE failed due to its inherent weaknesses or if the original classification is indeed itself, flawed. UMAP "seeks to learn the manifold structure of your data and find a low dimensional embedding that preserves the essential topological structure of that manifold" [7]. Essentially, UMAP offers 2 key advantages over t-SNE for the objective and scope of this report:

1. T-SNE preserves small pairwise distances or local similarities at the expense of preserving global data structure. Whereas UMAP learns and preserves the essential structure of the manifold from the high-dimensional space to the embedded low dimensional space. In theory, UMAP should perform better at visualizing distinct clusters for classification algorithms.
2. T-SNE is simply too computationally intensive compared to UMAP resulting in considerably longer compute times for the former.

4.1.3.1 Hyperparameters [8,9]

- *n*: The size of local neighbourhood used for manifold approximation determines the local versus global structure offset. A large local neighbourhood would favour the global structure, whereas a smaller local neighbourhood will force UMAP to focus on the local structure of the dataset. Neighbourhood sizes ranging from 2 to 100 were explored to observe any extreme and intermediate variations between offsetting local and global structure visualizations.
- *min-dist*: The effective minimum distance between embedded points determines the density or sparsity of clusters in the low-dimensional feature space. Values ranging from 0.1 to 0.99 were tested to distinctly identify the original ternary classification labels.
- *transform_queue_size*: Instead of optimizing the number of training epochs to improve the accuracy of low-dimensional embedding, large values of the transform operation ranging from 200 to 800 were tested for embedding new points by performing more accurate nearest neighbour evaluations.

4.1.3.2 Simplified Comparison to t-SNE Algorithm [9]

Algorithmically, UMAP shares some equivalent expressions with t-SNE. Whereas t-SNE uses perplexity, UMAP uses the number of nearest neighbours as shown in Equation (10),

$$k = 2^{\sum_i p_{ij}} \quad [10]$$

High-dimensional affinities are "local fuzzy simplicial set memberships, based on the smooth nearest neighbours distances as shown in Equation (11),

$$p_{j|i} = e^{\left[\frac{-d(x_i,x_j)-\rho_i}{\sigma_i}\right]} \quad [11]$$

The resultant matrix is symmetrized "by fuzzy set union using the probabilistic t-conorm" as shown in Equation (12),

$$p_{ij} = \left(p_{j|i} + p_{i|j}\right) - p_{j|i}p_{i|j} \quad [12]$$

Affinities in the embedded dimension for UMAP and t-SNE vary by user-defined *a* and *b* values for Equation (13) where *a*/*b* values for UMAP and t-SNE are 1.929/0.7915 and 1/1, respectively.

$$q_{ij} = \left(1 + a\left\|y_i - y_j\right\|_2^{2b}\right)^{-1} \quad [13]$$

4.2 Support Vector Machine (SVM) Classifiers

Support vector machine supervised machine learning algorithm was used for binary classification of the revised dataset. SVM classifiers were implemented on optimised t-SNE outputs on a 2-dimensional embedded feature space for the new *small* and agglomerated *large* class clusters. Classification performance precision and accuracy were evaluated for *polynomial*, *radial basis function* (RBF), and *sinusoid* kernel tricks.

5 Results & Discussion

5.1 Cluster Visualization and Feature Extraction

Tables 1 and 2 accompanied by Figures 3.1-3.20 and 4.1-4.18, respectively demonstrate that both t-SNE and UMAP were unsuccessful at distinctly visualizing the *medium* and *large* clusters in the original dataset. Tables 3 and 4 accompanied by Figures 5.1-5.5 and 6.1-6.5, respectively show the successful optimized outputs for the revised classification clusters. Table 5 lists the compute times for the optimized hyperparameter combinations of PCA, t-SNE, and UMAP algorithms. The 2-component low-dimension embedded output matrix for t-SNE with hyperparameters 400, 60, 50, and 5000 for perplexity, early exaggeration, learning rate, and maximum iterations was used as input for SVM classification algorithms.

5.2 Binary Market Segmentation for Re-Classified Dataset

The t-SNE generated output was separated into training and testing data sets for classification modelling and performance evaluations. Tables 6, 8, and 10 show the classification reports for SVM polynomial (degree = 10), RBF, and sigmoid kernel tricks, respectively. Whereas, Tables 7, 9, and 11 show the confusion matrices for SVM polynomial (degree = 10), RBF, and sigmoid kernel tricks, respectively. The polynomial kernel performed the poorest, whereas both RBF and sigmoid kernel tricks accurately classified and predicted both clusters, with RBF being marginally more accurate with .99 precision, recall, and F1 scores for more *small* and *large* classes.

6 Conclusions & Future Work

This report has re-evaluated the ternary market segmentation proposed by NREL and demonstrated that the original *medium* and *large* classes are virtually indistinguishable from each other. Neither t-SNE or UMAP were able to successfully visually three distinct clusters in the embedded feature space for the original dataset. The revised dataset with the original *small* class and agglomerated *large* class, which combined the original *medium* and *large* classes were shown to be more accurate and precise classification as per SVM RBF kernel performance.

Future works should apply predictive modelling machine learning algorithms to the *small* and *large* classes/segments. Additionally, closed-loop clustering algorithms should be explored further for either t-SNE or UMAP visualization techniques to optimize each's respective hyperparameters using feedback from classification algorithms such as SVM.

A References

[1] P. Gagnon, R. Margolis, J. Melius, C. Phillips and R. Elmore, "Rooftop Solar Photovoltaic Technical Potential in the United States: A Detailed Assessment", National Renewable Energy Laboratory, 2016.

[2] L. van der Maaten and G. Hinton, "Visualizing Data using t-SNE", Journal of Machine Learning Research, vol. 1, no. 48, 2008.

[3] A. Violante, "An Introduction to t-SNE with Python Example", Towards Data Science, 2018. [Online]. Available: https://towardsdatascience.com/an-introduction-to-t-sne-with-python-example-5a3a293108d1.

[4] L. van der Maaten, "Accelerating t-SNE using Tree-Based Algorithms", Journal of Machine Learning Research, vol. 15, no. 1-21, 2014.

[5] "sklearn.manifold.TSNE — scikit-learn 0.23.2 documentation", Scikit-learn.org, 2020. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html.

[6] A. Belkina, C. Ciccolella, R. Anno, R. Halpert, J. Spidlen and J. Snyder-Cappione, "Automated optimized parameters for T-distributed stochastic neighbor embedding improve visualization and analysis of large datasets", Nature Communications, vol. 10, no. 1, 2019. Available: 10.1038/s41467-019-13055-y.

[7] N. Oskolkov, "How Exactly UMAP Works", Towards Data Science, 2019. [Online]. Available: https://towardsdatascience.com/how-exactly-umap-works-13e3040e1668.

[8] “Basic UMAP Parameters — umap 0.4 documentation", Umap-learn.readthedocs.io, 2018. [Online]. Available: https://umap-learn.readthedocs.io/en/latest/parameters.html.

[9] L. McInnes, J. Healy, N. Saul and L. Großberger, "UMAP: Uniform Manifold Approximation and Projection", Journal of Open Source Software, vol. 3, no. 29, p. 861, 2018. Available: 10.21105/joss.00861.