Machine Learning for Stony Coral Tissue Loss Disease (SCTLD)

Nonlinear Dimensionality Reduction, Barney-style

Advika Ravishankar Sam Brunacini Christian Bolinas

University of Pittsburgh, NOAA

What is SCTLD?

- Disease causing tissue loss (bad) in coral off the gulf coast
- Researchers suspect this disease is bacterial in origin
- Researchers (and us too!) are attempting to analyze **which bacteria** may have predictive power in identifying this disease

Your mission, should you choose to accept it...

Traditional modeling approaches haven't offered sufficient insight

- "differential abundance analysis"
- Whether some bacteria's presence offers statistically significant increase in SCTLD prevalence
- Like WAR in baseball, sort of

Our task: **investigate modern machine learning approaches** to see if they offer additional insight

- Basically: "Here's some data. Please make sense of it."

Your mission, should you choose to accept it...

Our data (mostly 0s) was an impenetrable blob from a bunch of studies. It looked like this:

	Sample 1	Sample 2	...	Sample 2000
Bacteria 1	1	0	...	0
Bacteria 2	0	0	...	0
...	...	...	...	...
Bacteria 90,000	0	0	...	1

Challenges

Real-world data is messy.

Our modeling approach: problems

Cluster analysis: group related data together, observe patterns

- i.e. find patterns of bacterial presence in SCTLD samples

Normally simple to do (common task, lots of libraries)

Issue: when data is high-dimension (number of columns) relative to number of points (rows), clustering algorithms **don't work**.

Running all different kinds of clustering algorithms, even after parameter tuning, didn't work

- Resulted in as many clusters as there were samples
- Says *literally nothing* about our data

Our modeling approach: solution

"Squish" the data into a lower-dimension space: "dimensionality reduction"

- i.e. get rid of "useless" columns
- Choosing a clustering algorithm is trivial in comparison
- With good data, everything works

Then do cluster analysis!

Our approach

General workflow was empirical in nature:

1. Research dimensionality reduction, modeling technique
2. Try dimensionality reduction technique
3. Try modeling technique

Tried different combinations.

Dimensionality Reduction Approach: Principal Component Analysis

"Industry best practice"

- Uninteresting

Well-studied in the literature

- Uninteresting

Most importantly, **is linear**

- No reason to assume linear relationships a priori
- Misleading results when data (bacterial relationships that cause disease) has nonlinear relationships
- Ecology/ecosystem: **things interact with each other!**

Dimensionality Reduction Approach: Autoencoders

Neural network architecture

Learns the best way to

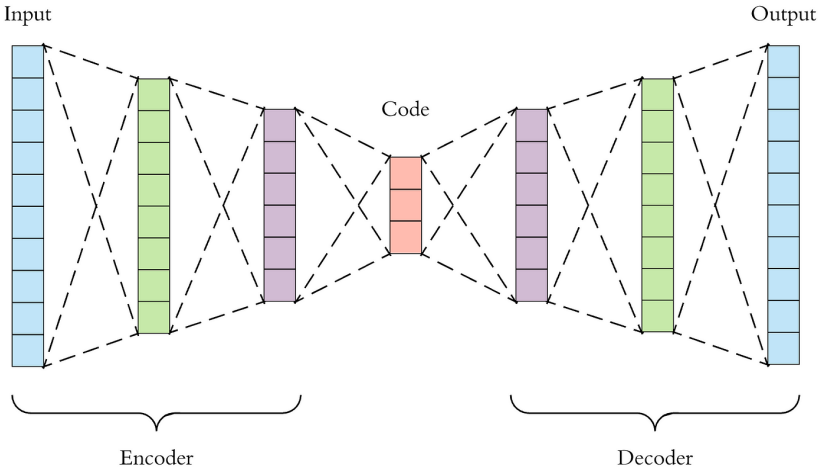
1. Take high-dimension data
2. Map it into low-dimension space

. . . while preserving important information

Neural network = nonlinear = **good**

"Best" mapping from high-dimension to low-dimension space = dimensionality reduction = **what we want**

Dimensionality Reduction Approach: Autoencoders



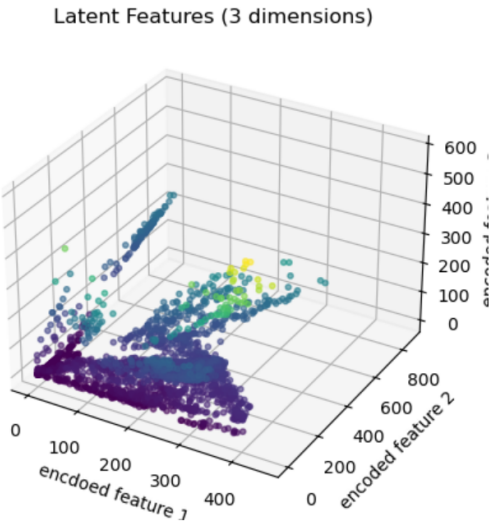
Dimension (height of block tower) “squished”: only important blocks kept

Dimensionality Reduction Approach: Variational Autoencoders

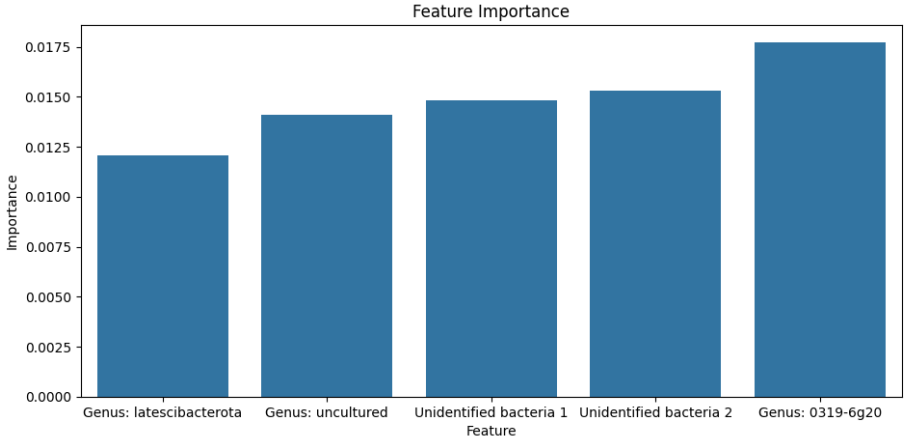
Autoencoder variation that learns probability distribution rather than specific function

- More robust against noise
- Real data is noisy
- **Good**

Dimensionality Reduction Approach: Variational Autoencoders: Results



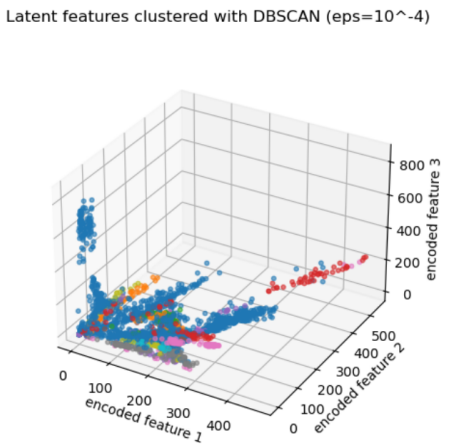
Dimensionality Reduction Approach: Variational Autoencoders: Results



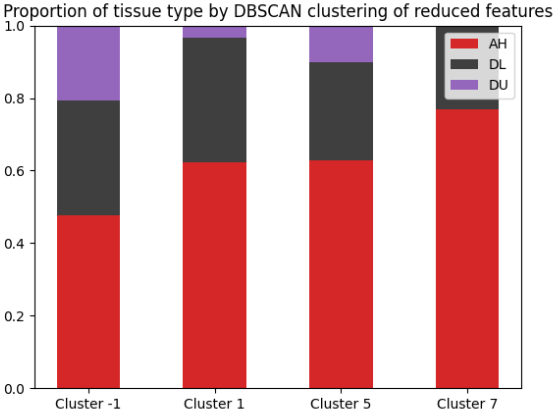
Clustering Approach: Overview

- Hierarchical, agglomerative algorithms
  - Fixed amount of clusters (clearly not known a priori...)
- K-means
  - Not robust against outliers (bad IRL)
  - Not robust against noise (bad IRL)
  - Fixed amount of clusters (clearly not known a priori...)
- DBSCAN
  - Robust against outliers and noise
  - Amount of clusters **not relevant a priori** to algorithm
  - **Generally works really well!**

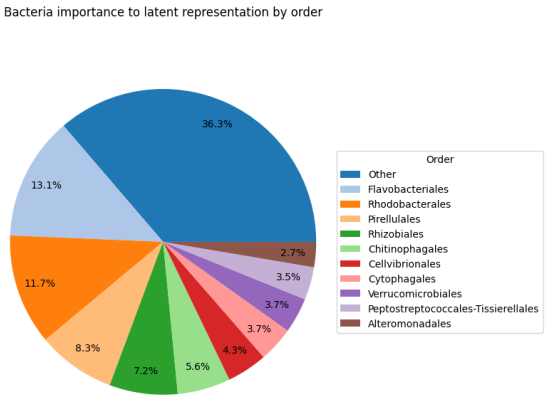
Clustering Approach: DBSCAN: Results



Clustering Approach: DBSCAN: Results



Overall. . .


Bacteria importance to latent representation by order
36.3%
13.1%
11.7%
8.3%
7.2%
5.6%
4.3%
3.7%
3.7%
3.5%
2.7%
Order
Other
Flavobacteriales
Rhodobacterales
Pirellulales
Rhizobiales
Chitinophagales
Cellvibrionales
Cytophagales
Verrucomicrobiales
Peptostreptococcales-Tissierellales
Alteromonadales

Analysis

Challenge: quantify model performance

Internal metric: silhouette score

- Compares how similar an object is to its own cluster compared to other clusters
- Works well with arbitrary cluster shapes/it's unbiased as to cluster *shape*
- Other metrics (e.g. Davies-Bouldin) biased (e.g. towards convex clusters)

HOWEVER...

Analysis

"Useful to stakeholders" $>$ numbers

External metric:

- Qualitative cluster quality
- Interactive visualization ("fly around" 3D space to explore)

Tech Talk: Our tech stack

Tech Talk: Our tech stack

Standard Python ML ecosystem

- ML/DL libraries
- Notebooks: thought process, code, visualization, analysis in one place
- Rapid iteration

Tensorflow: making deep models

- SCI CRC JupyterHub for GPU compute

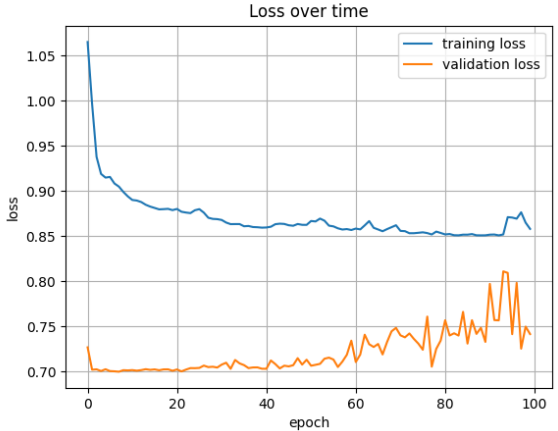
scikit-learn: clustering algorithms

Pandas: working with tabular data

Matplotlib: figures

Plotly: interactive visualization (clusters)

Tech Talk: Training VAEs: First Attempt



Tech Talk: Training VAEs: Sparsity

He Normal initialization

- Prevents vanishing gradients, dead neurons

ReLU -> Leaky ReLU

- Prevents neurons from inadvertently “dying”

Sigmoid final layer

Binary cross-entropy instead of MSE for loss

Tech Talk: Training VAEs: Overfitting

Model architecture

- More gradual dimension reduction
- Significantly more neurons
  - Learning was limited by insufficient number of parameters (model size)

Adaptive learning rate scheduler

Added dropout

Tech Talk: Training VAEs: Final Result

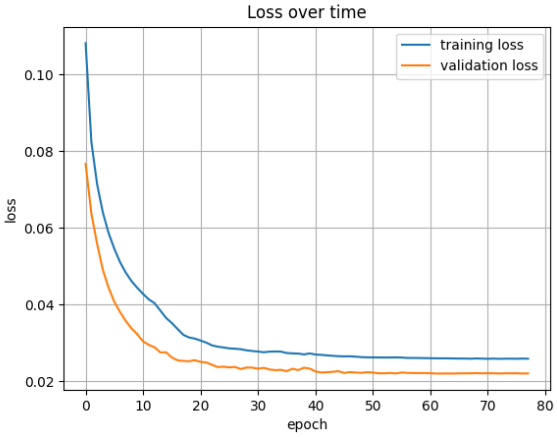


Figure 1: Note: Final loss ~0.02

Tech Talk: Training VAEs: First Attempt, Again For Comparison

Figure 2: Initial loss was ~0.75! Improvement by orders of magnitude!

Going Forward: Dimensionality Reduction

Still nonlinear dimensionality reduction

Still benchmarking vs. VAEs

- (Other) (such as sparse) autoencoders
  - **denoising autoencoder**
  - **concrete autoencoder**
- Kernel PCA (like SVM)

Going Forward: Clustering

UMAP + Mapper combo

- Empirically pretty good for high-dimensional, sparse data in this domain

HDBSCAN

- Better than DBSCAN for varying densities

Going Forward: Other

Use clusters as base for binary classifier with predictive power

Basically, is this similar to any well-defined groupings of SCTLD tissue samples?

```
classifier :: [bacteria] -> bool
```

Thanks for listening!