COMP 330: Intro to Unsupervised Learning

Chris Jermaine and Kia Teymourian

Rice University

Learning From Unlabeled Data

Sometimes you have a data set without labels

- (height, weight, age, shoe size) quadruples for this class
- Register transactions from Wal-Mart
- User-Movie rating matrix

The goal is explanatory:

- What to learn a model to help "understand" the data, in some sense
- Goal is not to predict some value(s) (though that might be a by-product)

Learning From Unlabeled Data

Sometimes you have a data set without labels

- (height, weight, age, shoe size) quadruples for this class
- Register transactions from Wal-Mart
- User-Movie rating matrix

The goal is explanatory:

- What to learn a model to help "understand" the data, in some sense
- Goal is not to predict some value(s) (though that might be a by-product)

This is "Unsupervised Learning"

Classically Two Types of UL

(1) Clustering

- Grouping similar points together

(2) Latent Variable methods

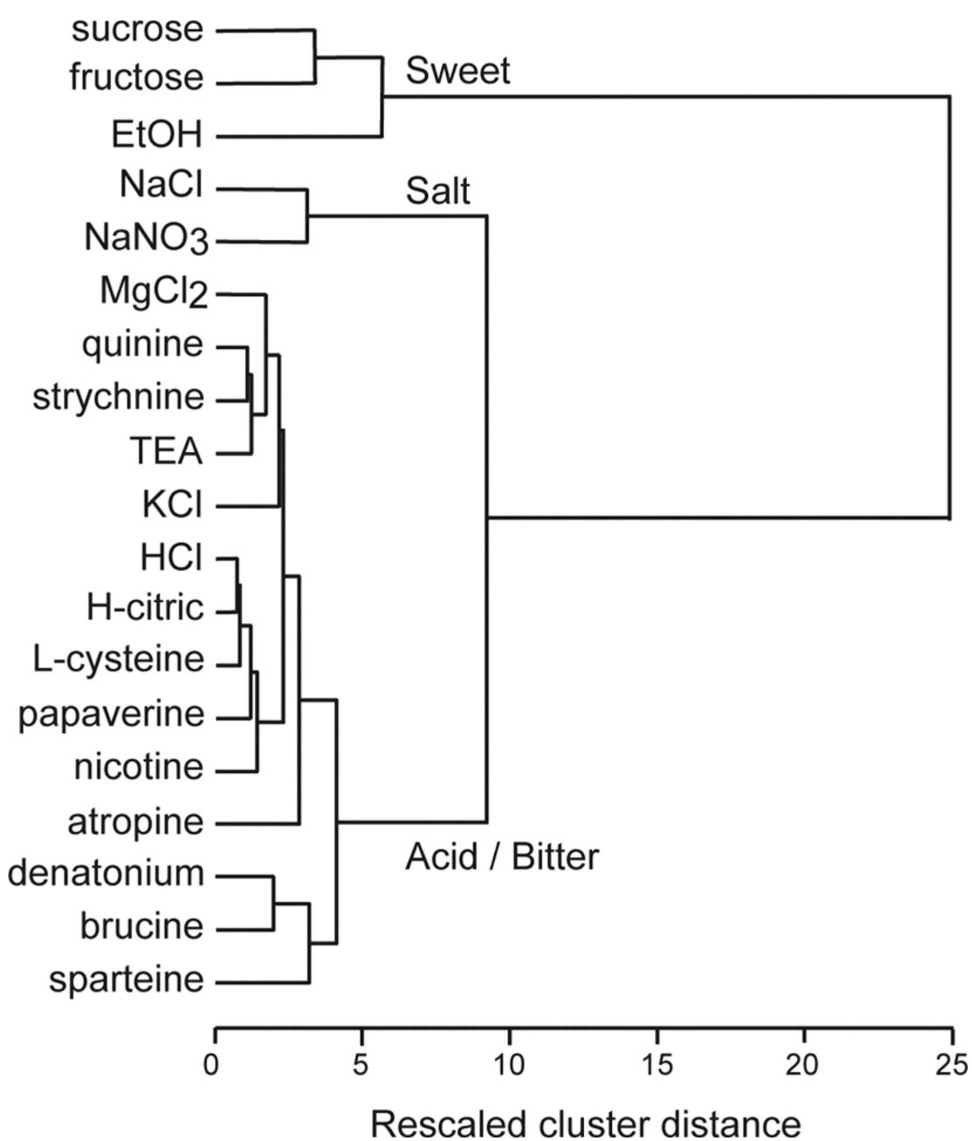
- Learning a model where some unseen variable helps describe the data

Clustering

Goal is to group similar points together

- Classic method is heirarchical clustering
- Also known as agglomerative clustering
- Results in a so-called "Dendogram"
- example...

Hierarchical Clustering



Hierarchical Clustering

Basic Algorithm:

```
while num_clusters > 1 do
  // D is the distance functoin
  find clusters X, Y that minimize D(X,Y)
  join them
end
```

Super-simple

Key question:

- How to define cluster distance?

Single-Linkage Clustering

"Optimistic"

- $D(X, Y)$ is distance between two closest points in X, Y
- That is,

$$D(X, Y) = \min_{x \in X, y \in Y} d(x, y)$$

- Basically Kruskal's algorithm

Single-Linkage Clustering

"Optimistic"

- $D(X, Y)$ is distance between two closest points in X, Y
- That is,

$$D(X, Y) = \min_{x \in X, y \in Y} d(x, y)$$

- Basically Kruskal's algorithm

Drawbacks?

- Naive solution $O(n^3)$
- Possible to use variant of Prim's algorithm to get $O(n^2)$
- "Chaining"

Complete-Linkage Clustering

"Pessimistic"

- $D(X, Y)$ is distance between two most distant points in X, Y
- That is,
$$D(X, Y) = \max_{x \in X, y \in Y} d(x, y)$$
- Tends to produce more compact clusterings
- Best solution is also $O(n^2)$

What About The Distance?

How to compute $d(x, y)$?

- Classical method: if x, y vectors, use l_p norm of $x - y$
- Drawbacks?

What About The Distance?

How to compute $d(x, y)$?

- Classical method: if x, y vectors, use l_p norm of $x - y$
- Drawbacks?

Mahalanobis distance is more robust

- Let μ be the mean vector of the data set
- Let S be the observed covariance matrix of data set
- That is, let X be the matrix where the ith row is $x_i - \mu$
- Then $S = \frac{1}{n-1} X^T X$
- Mahalanobis computed as:

$$d(x, y) = \left((x - y)^T S^{-1} (x - y) \right)^{\frac{1}{2}}$$

- Intuition?

Other Clustering Methods?

Naturally they exist...

Will talk about a few over the next few weeks!

Latent Variable Methods

What is a Latent Variable Method?

- By postulating the existance of "latent" variables
- Latent: missing or unobserved (back to the coin flip!)
- Difference: latent variables typically imagined
- Used to help simplify/explain the data
- Often probabilistic (MLE, Bayesian)
- Can be optimization-based

Classic Example: NNMF

Motivation:

- Have a 2-D table
- Entries in table describe outcome of interaction
- Example: Netflix challenge

Classic Example: NNMF

Motivation:

- Have a 2-D table
- Entries in table describe outcome of interaction
- Example: Netflix challenge

We have a bunch of (movie, user, score) triples

Stored in an n by m matrix V (n movies, m users)

- Idea: map ith movie to a latent k-dimensional point m_i
- And map jth user u to a latent k-dimensional point u_j
- Such that the score user i gives to movie $j \approx m_i \cdot u_j$

Classic Example: NNMF

Motivation:

- Have a 2-D table
- Entries in table describe outcome of interaction
- Example: Netflix challenge

We have a bunch of (movie, user, score) triples

Stored in an n by m matrix V (n movies, m users)

- Idea: map ith movie to a latent k-dimensional point m_i
- And map jth user u to a latent k-dimensional point u_j
- Such that the score user i gives to movie $j \approx m_i \cdot u_j$

Many formulations; one is:

$$\min_{\{m_1, m_2, \ldots, u_1, u_2, \ldots\}} \sum_{i,j} (V_{i,j} - m_i \cdot u_j)^2$$

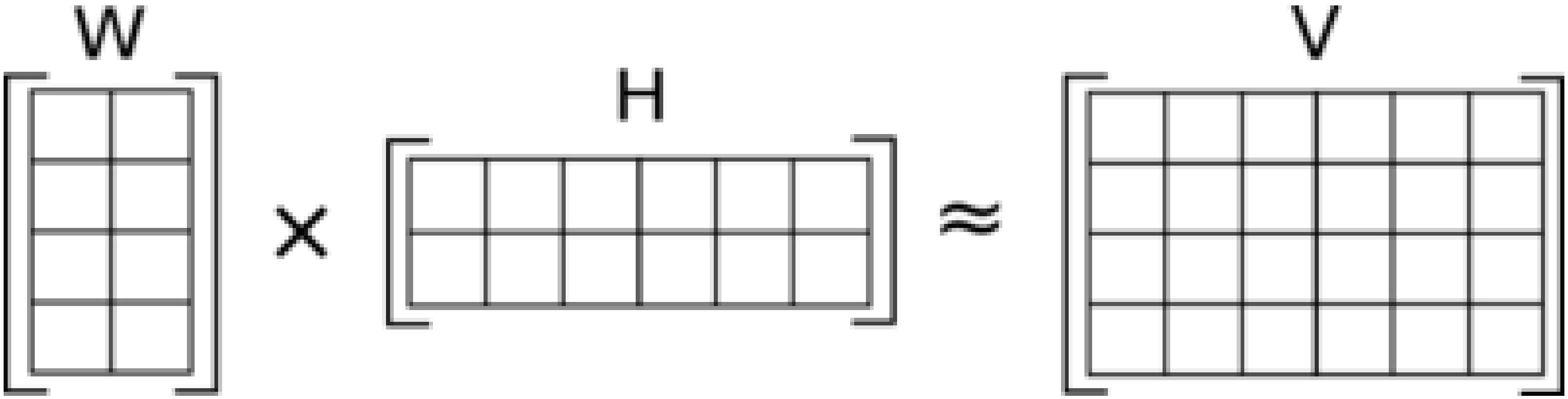
Classic Example: NNMF

Many formulations; one is:

$$\min_{\{m_1, m_2, \ldots, u_1, u_2, \ldots\}} \sum_{i,j} (V_{i,j} - m_i \cdot u_j)^2$$

- Can be solved many ways: example: gradient descent
- "Non-negative" since often we want all latent vectors to be positive
- Turns out that the latent space is often meaningful!

Why Called "Matrix Factorization"?



View W matrix as latent positions of movies

View H matrix as latent positions of users

We are trying to learn W, H from V

Closing Comments

“Supervised” vs. “Unsupervised” distinction not always hard

Closing Comments

"Supervised" vs. "Unsupervised" distinction not always hard

"Clustering" vs. "Latent Variable" distinction not always hard

- All but the most ad-hoc clustering algorithms can be written as latent variable problems

Questions?