

# CV\_M14\_L01

# Embedding-Based Methods

**CS/DSA 5970: Machine Learning Practice**

# Non-Linear Manifolds

- As we have seen, manifolds are not generally linear
  - E.g., two features can vary together, but not linearly
- Manifolds can also loop back onto themselves
  - E.g., two features that do not have a one-to-one relationship

# IPAD\_M14\_L01b

# Non-Linear Manifolds

- PCA: linear manifolds only
  - Construct a global model of the manifold
- Kernel PCA: can express non-linearities
  - Simple case: representation of the manifold is a global model
  - Kernel trick: captures the model in terms of a weighted sum over the training set samples
- Can also take a sample-based approach in the original space!

# Locally Linear Embedding

- Training set in N-dimensional feature space
- Measure distance between each pair of training set samples
  - For each sample, identify the closest neighbors
  - The closest neighbors give us a sense of the shape of the local neighborhood
- Place corresponding points in a new M-dimensional space:
  - Select these points, so that the distances to the neighbors are preserved



*The* UNIVERSITY *of* OKLAHOMA

# CV\_M14\_L02

# Locally Linear Embedding

**CS/DSA 5970: Machine Learning Practice**

# Locally Linear Embedding (LLE)

- Phase 1: Build local models
- Phase 2: Embedded corresponding points into a lower-dimensional space

# Phase 1: Build Local Models

- Use Euclidean distance metric to identify the  $k$  nearest neighbors for each point
  - Generally, these nearest neighbors define a local manifold
  - The dimensionality of this local neighborhood is at most  $k-1$
- For each point, identify a weighted sum of the neighbors that predicts the location of the point

# IPAD\_M14\_02b

- Algorithm, part 1



*The* UNIVERSITY *of* OKLAHOMA

# CV\_M14\_L03

# Locally Linear Embedding: Embedding

## CS/DSA 5970: Machine Learning Practice

# LLE: Embedding Phase

- Each  $X_i$  has a corresponding  $Z_i$  in an  $M$ -dimensional space
- Pick the location of the  $Z_i$ 's that respect the neighborhood models that we learned in the previous step
  - These are the weights that we have already determined

# IPAD\_M14\_L03b



*The* UNIVERSITY *of* OKLAHOMA

# CV\_M14\_L04

# Locally Linear Embedding: Query

CS/DSA 5970: Machine Learning Practice

# IPAD\_M14\_04b



*The* UNIVERSITY *of* OKLAHOMA

# CV\_M14\_L05

# Example: Locally Linear Embedding

CS/DSA 5970: Machine Learning Practice

# Live demo



*The* UNIVERSITY *of* OKLAHOMA

# CV\_M14\_L06

# Example 2: Locally Linear Embedding

CS/DSA 5970: Machine Learning Practice

# Example 2: Locally Linear Embedding

New data set:

- Original space: 3D
- Varying density
  - 1D manifold mixed with 2D manifolds

# Live demo



*The* UNIVERSITY *of* OKLAHOMA

# CV\_M14\_L07

# Multidimensional Scaling

**CS/DSA 5970: Machine Learning Practice**

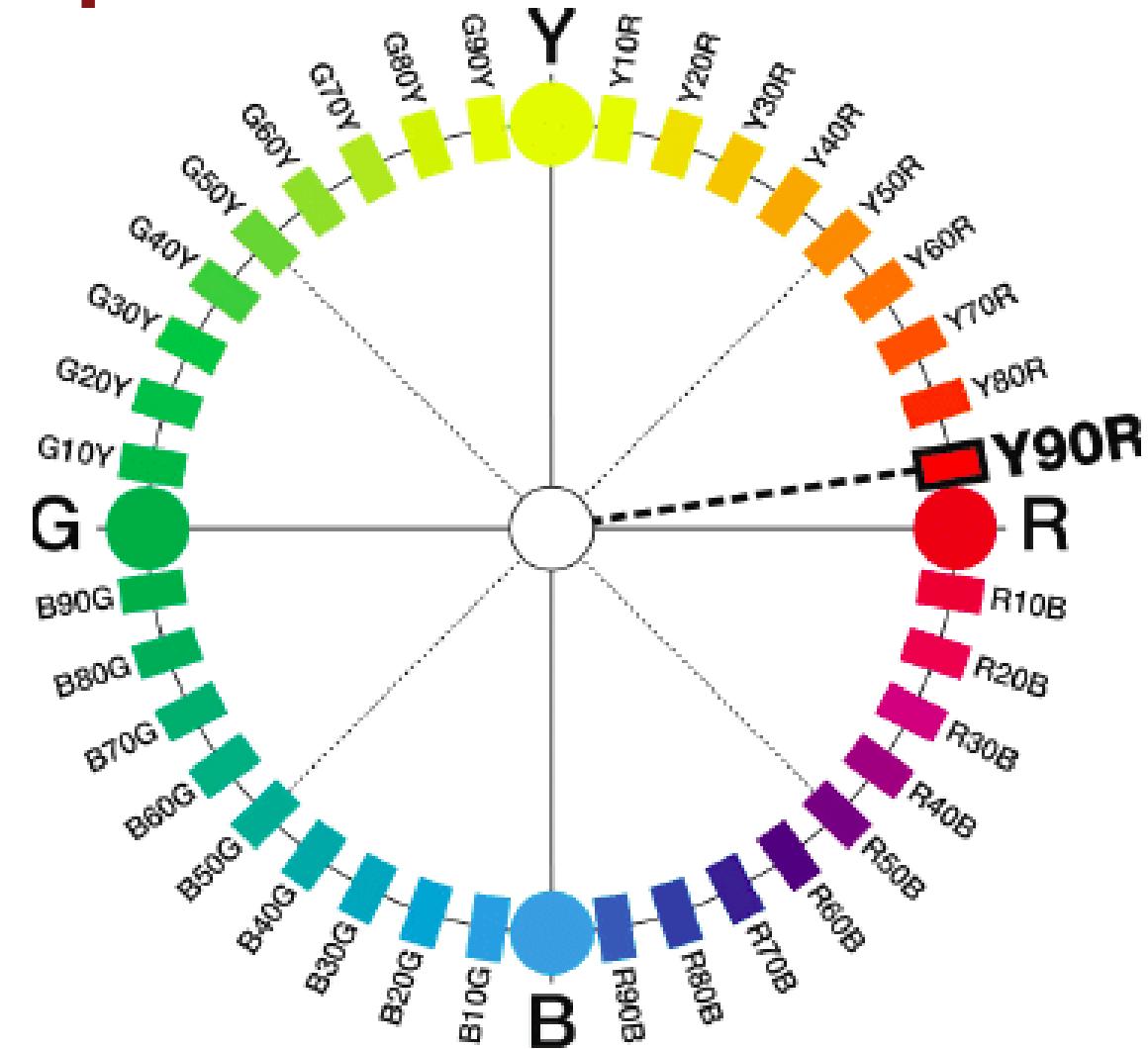
# Euclidean Distance Metric

- Easy to compute
- In many data sets, it is not trivial or appropriate to compare samples in this way:
  - Different features have different units and different scales
  - For some representations, we can't simply take a difference between two values (e.g., angles)

# Color Perception

How well does a human distinguish colors?

- Are two colors different?
- If so, by how much?



Tseghai et al.; researchgate.net

# Multidimensional Scaling

Useful for:

- Situations where we want to use different distance metrics
- Situations where we can't measure the features, but can measure the distances

# Multidimensional Scaling

## Algorithm outline

- Compute or measure all pair-wise distances between samples
- Embed a set of points into  $M$  dimensions that respect these differences

# IPAD\_M14\_L07b

# Multidimensional Scaling

## Notes

- The MDS cost function is a global metric
- All pair-wise distances must be respected (not just the nearest neighbors)



*The* UNIVERSITY *of* OKLAHOMA

# CV\_M14\_L08

# Example: Multidimensional Scaling

CS/DSA 5970: Machine Learning Practice

# Multidimensional Scaling

Both data sets:

- Swiss roll
- Arrow



*The* UNIVERSITY *of* OKLAHOMA

# CV\_M14\_L09

# Geodesic Distance

**CS/DSA 5970: Machine Learning Practice**

# Geodesic Distance

- Euclidean distance: not always meaningful in high-dimensional spaces
- Here, assume that Euclidean distance is only meaningful for short distances
- Use this neighborhood relation to define a weighted graph structure among the  $k$ -nearest neighbors
- Geodesic distance between all pairs of points: shortest distance in this graph

# IPAD\_M14\_L09b



*The* UNIVERSITY *of* OKLAHOMA

# CV\_M14\_L10

# ISOMap

## CS/DSA 5970: Machine Learning Practice

# ISOMap

- Compute geodesic distance for each pair of points in the training set
- Use multi-dimensional scaling to embed corresponding points into a new space
- Advantage over Euclidean distance: points that are somewhat near in Euclidean space, but are far away in geodesic distance are considered far away from one-another

# Live demo



*The* UNIVERSITY *of* OKLAHOMA

# CV\_M14\_L11

# t-Stochastic Neighbor Embedding

**CS/DSA 5970: Machine Learning Practice**

# t-Stochastic Neighbor Embedding

Similarity metric in the original space:

- For a given sample: the probability of selecting one of the other samples from the training set to be its neighbor
- Gaussian distribution: highest similarity when the two samples are the same & drops off as they move apart

Embedded space:

- Select  $Z_i$ 's so that the probability distributions are the same across the two spaces

# IPAD\_M14\_L11

# t-SNE

- Use of the probability distribution emphasizes nearest points and treats all far points the same
- PDs really emphasize clusters of points
- Perplexity hyper-parameter:
  - Higher values: include more neighbors in the computation
  - Gives us smoother functions
- No good way to query after the fact:
  - Hence, this is often used for visualization of the given data set



*The* UNIVERSITY *of* OKLAHOMA

# CV\_M14\_L12

# Example: T-SNE

**CS/DSA 5970: Machine Learning Practice**

# Live demo



*The* UNIVERSITY *of* OKLAHOMA

# CV\_M14\_L13

# Dimensionality Reduction: Final Thoughts

**CS/DSA 5970: Machine Learning Practice**

# Dimensionality Reduction Methods

- Global methods: PCA and (sort of) Kernel PCA
- Local models: LLE, MDS, ISOmap, tSNE
  - And, with the kernel trick, Kernel PCA

# Embedding Methods

- PCA: first thing to try
- LLE:
  - Capture local manifold, but ignore larger structure
- MDS:
  - Allows us to use any distance metric that we want
  - We don't even need to have a feature-based representation of the samples

# Embedding Methods

- ISOMap:
  - Very curved manifolds, especially those that loop back onto themselves
- t-SNE:
  - Looking for pockets (clusters) of samples
  - Most often used for visualization purposes

# Dimensionality Reduction Uses

- Visualization: give domain experts and data analysis practitioners a better understanding of the geometry of the feature space
- Preprocessing for other methods
  - By unwarping curved manifolds, linear models potentially become viable again
  - By reducing dimensionality, we have less of an opportunity to overfit the data