WAYNE STATE UNIVERSITY

High-Dimensional Data Visualization, Feature Extraction & Selection

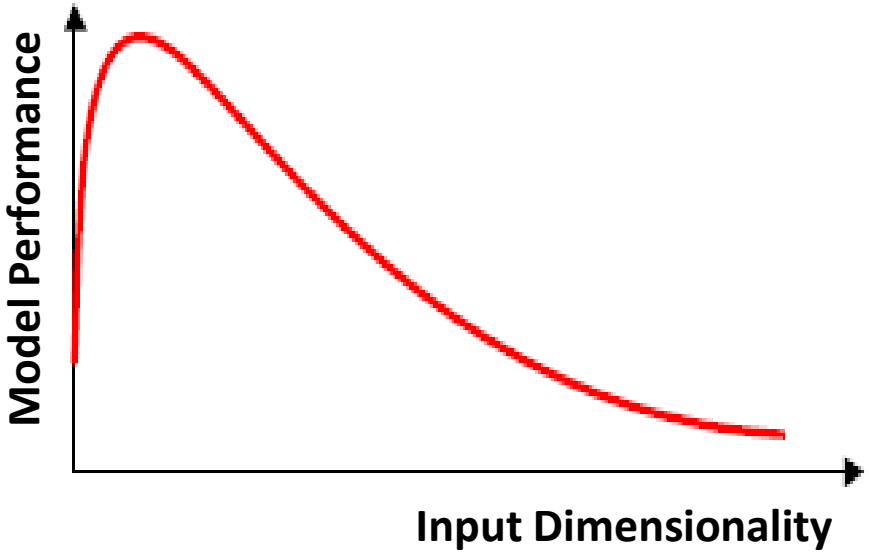
Dr. Ratna Babu Chinnam

Industrial & Systems Engineering

Wayne State University

Curse of Dimensionality

- Phenomena that arises when analyzing data in high-dimensional spaces
- When dimensionality increases (i.e., # of input variables), volume of space increases so fast that available data become sparse
- With a fixed number of training samples, predictive power of model reduces as dimensionality increases (even if variables have useful information)
  - Hughes phenomenon



For a given sample size, there is a maximum number of features above which the model performance degrades!

Hughes, G.F. (January 1968). "On the mean accuracy of statistical pattern recognizers". *IEEE Transactions on Information Theory*. **14** (1): 55–63.doi:10.1109/TIT.1968.1054102

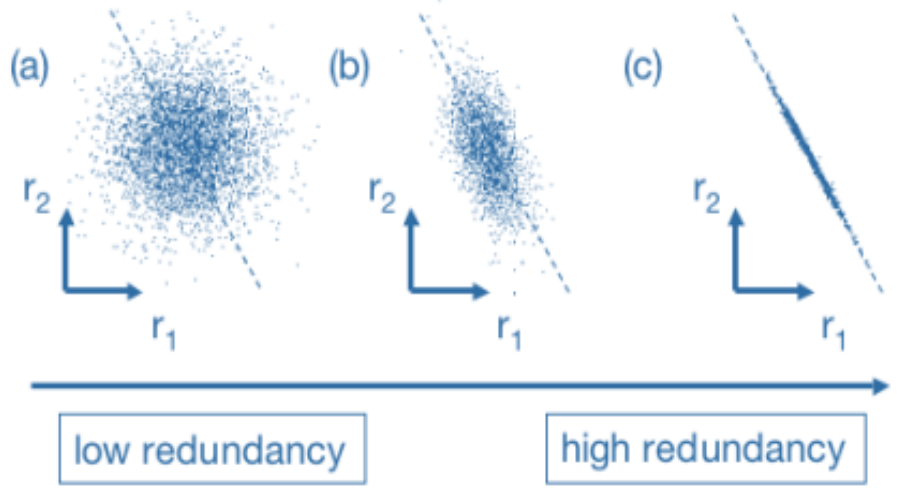
Curse of Dimensionality …

Example: k-Nearest Neighbor (kNN) Approach

- kNN breaks down in high-dimensional space
  - "Neighborhood" becomes very large
- Assume $5{,}000$ points uniformly distributed in a "unit" hypercube and we want to apply 5-NN
- Suppose that the query point is at the origin:
  - In 1-dimension, we must go a distance of 5/5000=0.001 on the average to capture 5 nearest neighbors (on a scale of one unit, these neighbors are close)
  - In 2-dimensions, we must go $\sqrt{0.001}$ distance to get a square that contains 0.001 of the area
  - In d-dimensions, we must go $(0.001)^{1/d}$ distance
  - In 10-dimensions, we must go 0.5012 distance! Are they close neighbors?

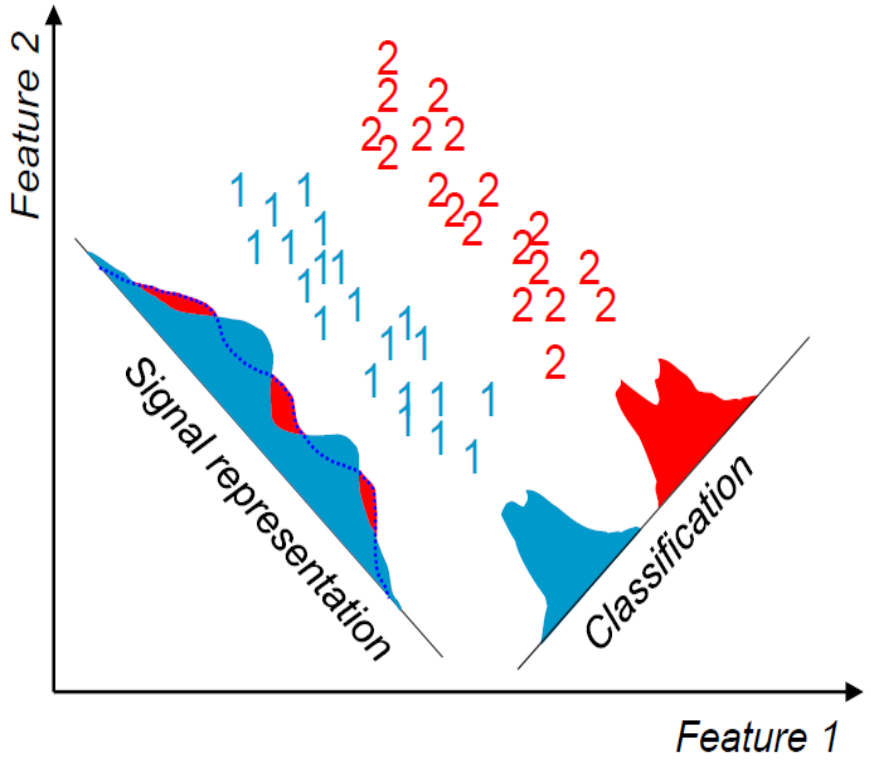
Feature Extraction

- To reduce dimensionality, need feature extraction/selection techniques
- **Goal**: Extract features that eliminate noise and remove redundancy
- Feature extraction builds derived values (features) intended to be informative and non-redundant
- **Feature Engineering**: Expert constructs application-specific features
  - Example: Cepstral features for speech recognition (link)
  - Difficult and time consuming
- If expert knowledge is not available, general dimensionality reduction techniques may help



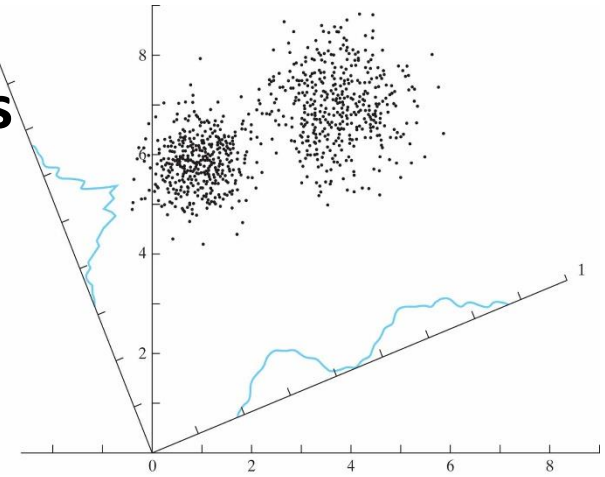
Feature Extraction

- ***Signal Representation***: Goal is to represent samples accurately in a lower-dimensional space
  - Popular methods: Principal Component Analysis (PCA), Kernel PCA, Independent Component Analysis (ICA), Autoencoders (Neural Networks)
- ***Classification***: Goal is to enhance class-discriminatory information in lower-dimensional space
  - Example method: Linear Discriminant Analysis (LDA)



Principal Component Analysis (PCA)

- **Statistical procedure that transforms a dataset of possibly correlated variables into a set of linearly uncorrelated variables called *principal components***
  - **Principal components are NOT "independent"!**
  - # of principal components is $\leq$ number of variables
- First principal component has the largest possible variance
  - Each succeeding component has highest variance possible under the constraint that it is orthogonal to preceding components
- Resulting vectors are an uncorrelated orthogonal basis set
- PCA is done by eigenvalue decomposition of data covariance matrix or singular value decomposition of data matrix
  - Usually after normalizing data matrix for each attribute

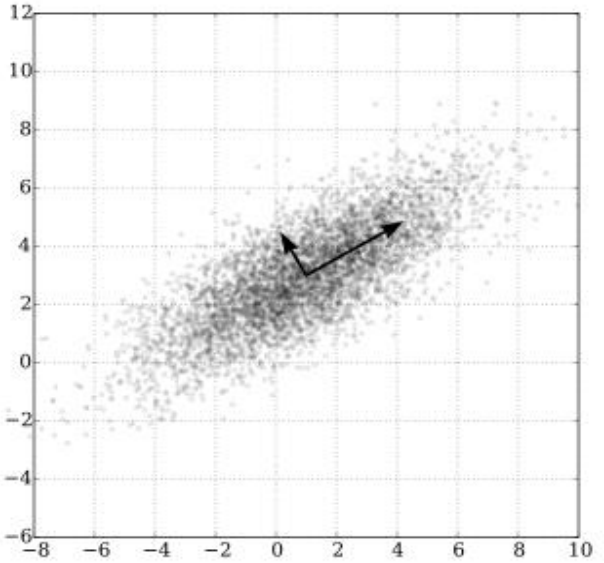


Projection onto PC_1 axis has maximum variance.

- PCs are difficult to explain (eigen vector combinations of original variables)
- Typically exhibit a Pareto structure (few PCs explain most of the dataset variance)
- **Unlike variable selection methods, PCA does not eliminate or reduce the burden of maintaining/collecting all the original variables.**

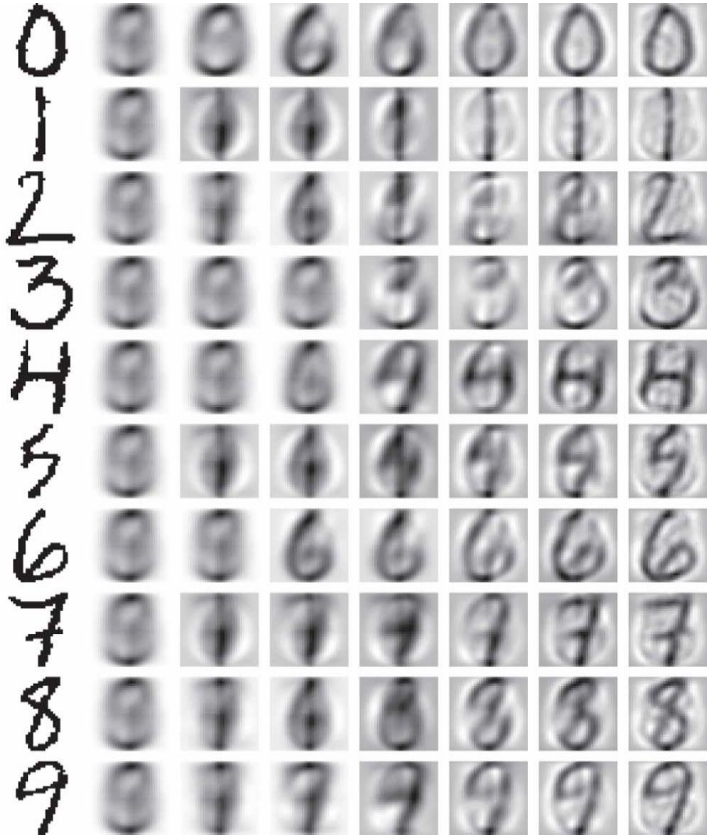
Principal Component Analysis (PCA): Intuition

- PCA can be thought of as fitting an n-dimensional ellipsoid to the data, where each axis of ellipsoid represents a principal component
  - If some axis of ellipse is small, then variance along that axis is also small
  - By omitting that axis and its corresponding component, we lose only a small amount of information
- To find the axes of the ellipse, we compute covariance matrix of data, and calculate eigenvalues and corresponding eigenvectors.
- The proportion of variance that each eigenvector represents can be calculated by dividing the eigenvalue corresponding to that eigenvector by sum of all eigenvalues.

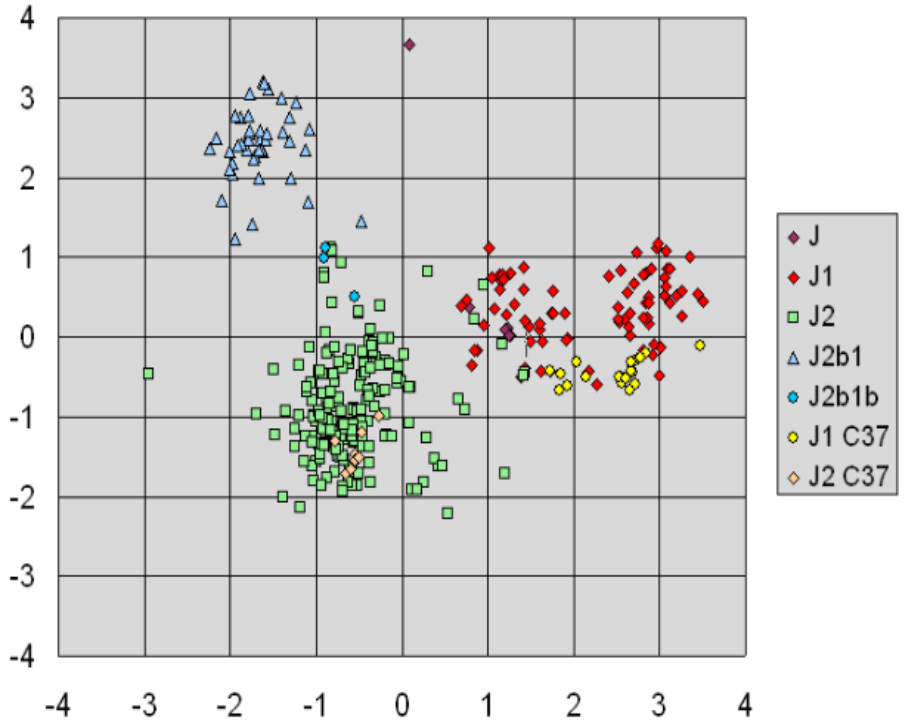


PCA of a multivariate Gaussian distribution. Vectors shown are eigenvectors of covariance matrix scaled by the square root of the corresponding eigenvalue, and shifted so their tails are at the mean.

PCA Examples



Digital compression of handwritten digits (32x32 = 1,024 pixels) using principal-components analysis (1,700 samples). First column is a sample original image and 2nd column shows computed means across dataset. Remaining columns show reconstructed images with increasing number of PCs (1, 2, 5, 16, 32, and 64). (Source: Dr. Juha Karhunen through Haykin 2009)

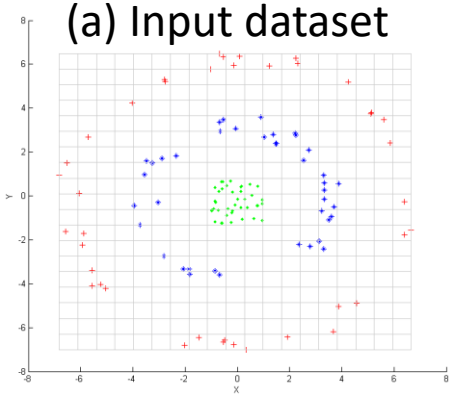


Scatterplot (1st two PCs) of Y-STR haplotypes from 37 Y-chromosomal STR markers from 354 individuals. PCA nicely separates clusters corresponding to genetic descent. (Wikipedia)

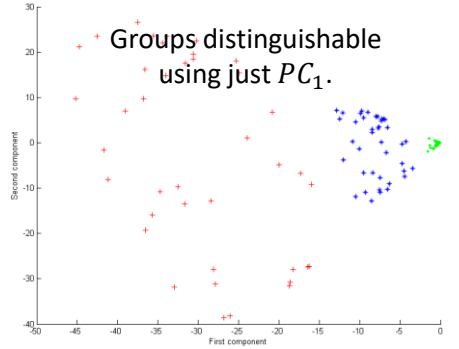
Kernel PCA: Non-linear Features

- Given that PCA only explores linear transformations, depending on application, it might be necessary to explore non-linear feature extraction methods.
- **Kernel-PCA involves mapping original data into the high dimensional feature space with the hope that nonlinearities that are present in original input space might become linear in feature space.**
  - If true, PCA can be carried out in feature space
- Using the kernel trick, we are able to avoid making calculations in the feature space.
- For Matlab, the Kernel Methods Toolbox from Matlab Central supports Kernel PCA and other kernel-based algorithms.

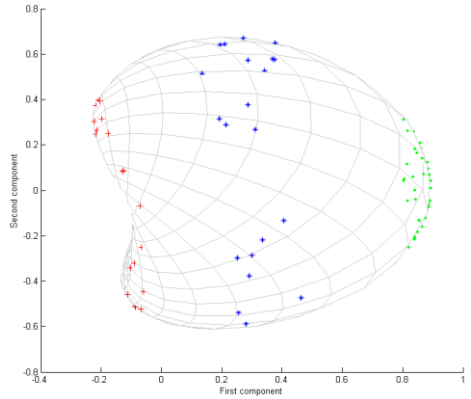
(a) Input dataset



(b) Polynomial kernel of 2nd order with unit additive constant.

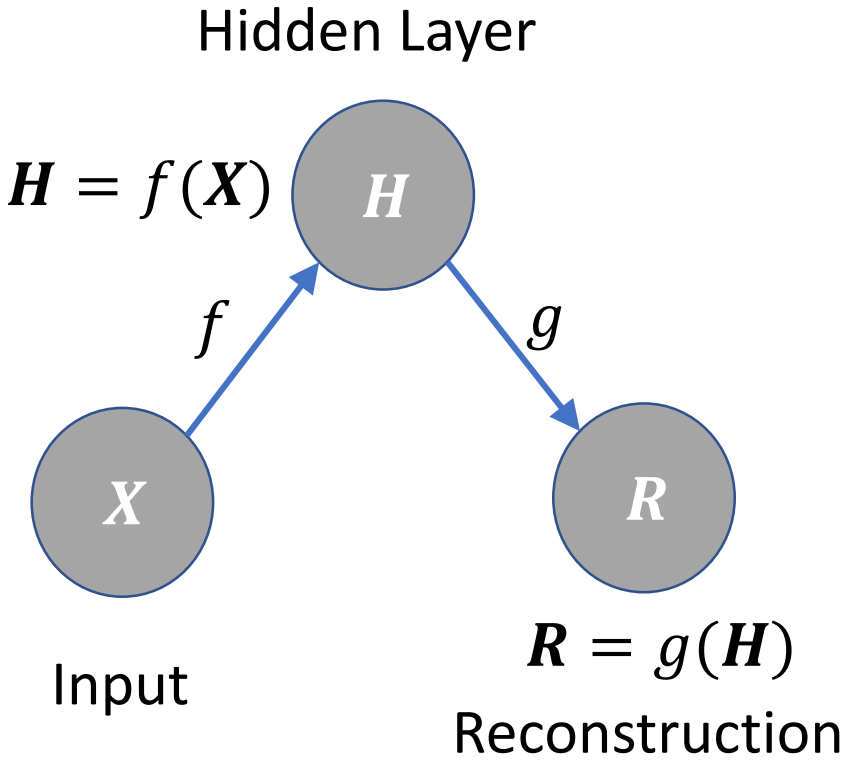


(c) Output with a Gaussian kernel.



Feature Extraction: Autoencoders for Dimensionality Reduction

- An autoencoder is a neural network trained to copy its input to its output
- Network has encoder and decoder functions
- Autoencoders should not copy perfectly
  - Restricted by design to copy only approximately
  - By doing so, it learns useful properties of data
- Modern autoencoders use stochastic mappings (variational autoencoders)



Mapping Data to Low Dimensions: Visualization

WAYNE STATE UNIVERSITY

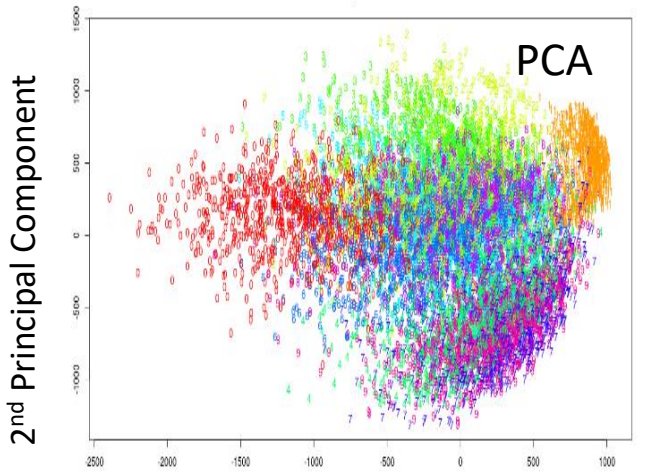
t-SNE[1]: Popular unsupervised non-linear method for high-dimensional data

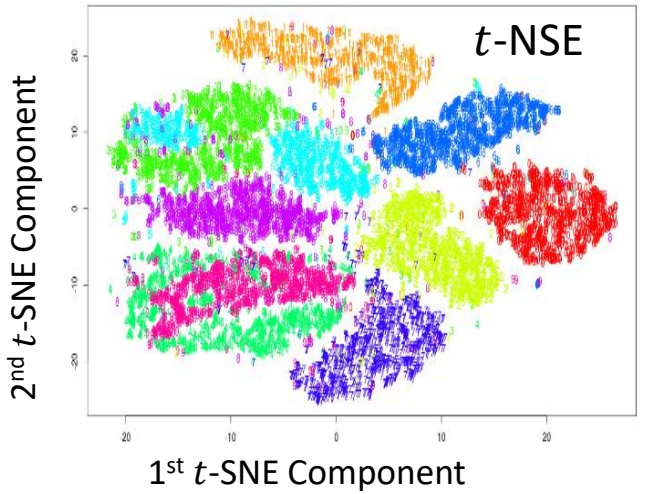
- PCA not that useful for high-dimensional data
- t-SNE creates compelling 2-D "maps" from data with hundreds or even thousands of dimensions
- **Goal: Find a faithful representation for high-dimensional data in 2D or 3D space**
- Algorithm is non-linear and adapts to underlying data, performing different transformations on different regions
- A tunable parameter ("perplexity") balances attention between local and global aspects of data
- Original t-SNE only works with data it is given
  - It does ***not*** produce a model that you can then apply to "new" data; Cannot extract features!!!

Sample

28x28 pixels

Visualizing MNIST Dataset
(Handwritten Letters)





REFERENCE L.v.d. Maaten, G. Hinton. **Visualizing data using t-SNE.** Journal of Machine Learning Research, Vol 9(Nov), pp. 2579—2605, 2008. [PDF]

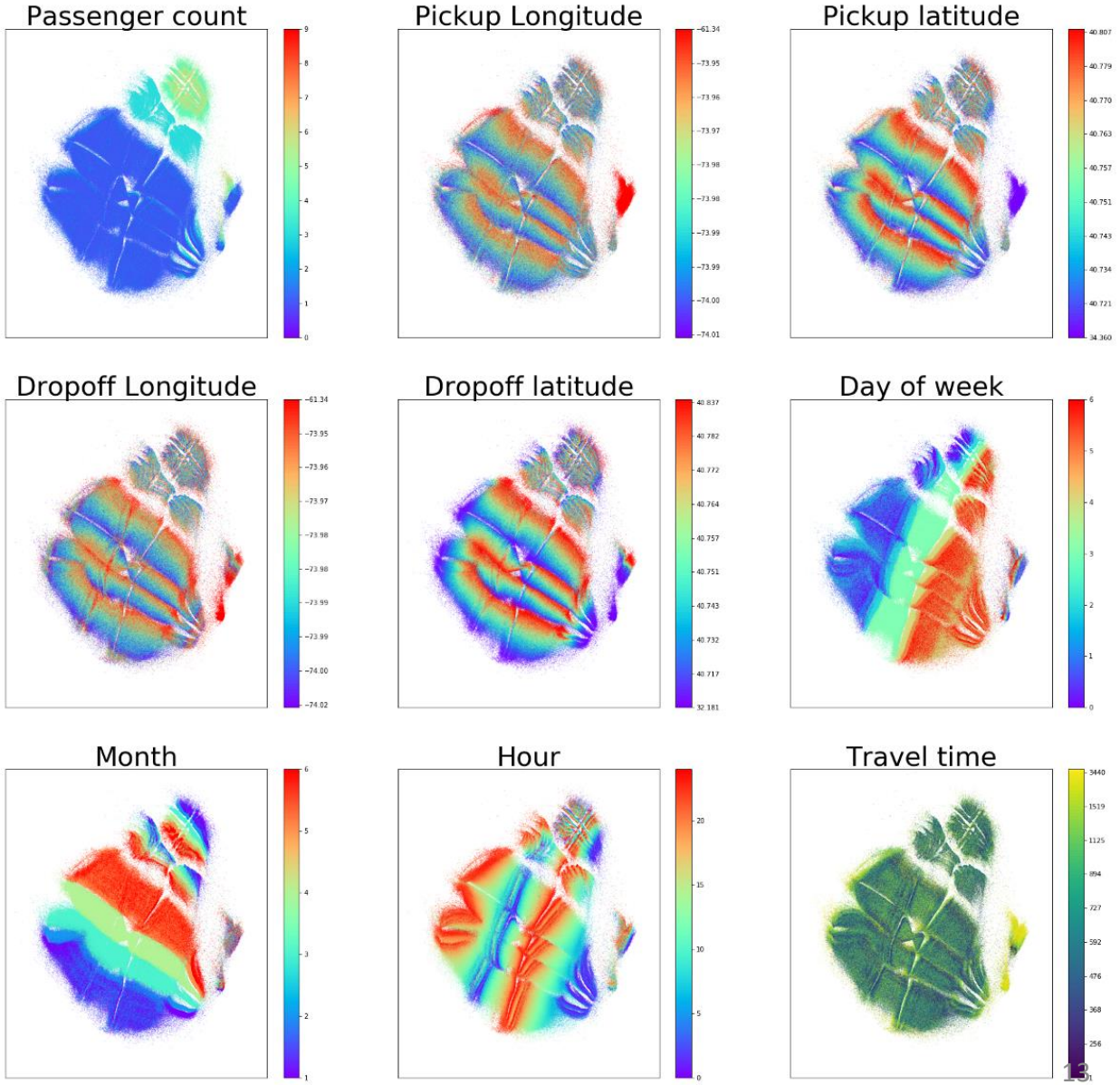
t-SNE Algorithm: More details

- Models probability distribution of *neighbors* around each point
  - In the original high-dimensional space, modeled as a Gaussian distribution
  - In 2-D output space, this is modeled as a t-distribution
  - Fatter tails of t-distribution help to spread points more evenly in 2-D space
- **Perplexity**: Equivalent to number of nearest neighbors considered when matching the original and fitted distributions for each point
  - Low perplexity: We care about local scale and focus on closest other points
  - High perplexity takes more of a "big picture" approach
- Because distributions are distance based, all data must be numeric
  - Convert categorical to numeric variables by encoding or a similar method
- Normalize data to ensure each variable is on same scale
- Great Playground: LINK
- Post with Examples & Code: LINK
- Implementations in R & Python: LINK

SOURCE: Link

Feature Extraction with t-SNE

- Brilliant post that shows how to adapt t-SNE so that we can get a model for use with "new" data
  - Uses a feed-forward neural net to achieve dimensionality reduction
  - Employs t-SNE cost function for training network
- Example: Kaggle NYC Taxi Dataset
  - Features: Passenger count, Pickup longitude, pickup latitude, dropoff longitude, dropoff latitude, day of week, month, hour
  - Dependent Variable: Travel time
- Method ideal for larger feature datasets



SOURCE: Link

Mapping Data to Low Dimensions: Visualization

Uniform Manifold Approximation and Projection (UMAP): Dimension reduction technique for visualization, but also for general non-linear dimension reduction

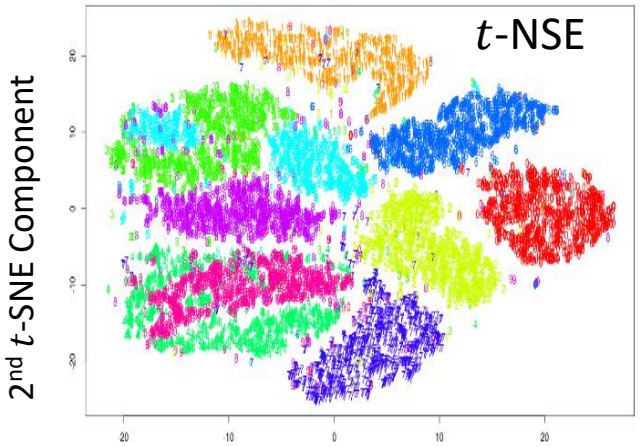
- **UMAP is fast**. Handles large datasets and high dimensional data without much difficulty (even with > million dimensions)
- **UMAP scales well in embedding dimension.** You can use UMAP as a general purpose dimension reduction technique as a preliminary step to other machine learning tasks.
- **UMAP often performs better than *t*-SNE** at preserving some aspects of global structure of the data
- **UMAP supports a wide variety of distance functions**, including non-metric distance functions (cosine and correlation distance).
- **UMAP supports adding new points to existing embedding.** Means that UMAP can be used as a preprocessing transformer in $sklearn$ pipelines.
- **UMAP supports supervised and semi-supervised dimension reduction**. If you have label information (even if just partial labelling)—simply providing it as y parameter in fit method.
- It has theoretical foundations in manifold learning (paper).

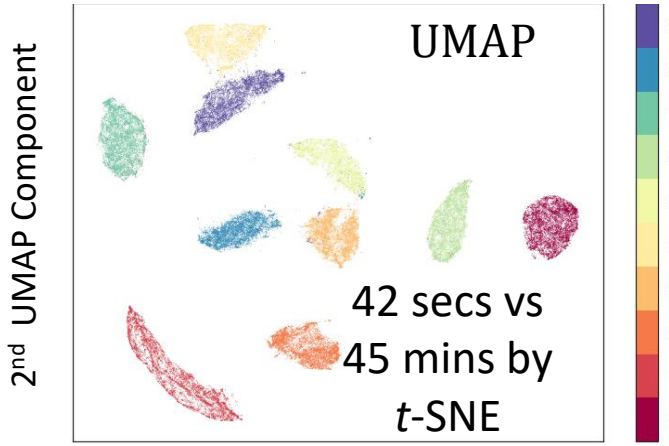
Source: UMAP | GitHub

Visualizing MNIST Dataset
(Handwritten Letters)

Sample

28x28 pixels





UMAP Algorithm: More details

How to use UMAP (Python)

- UMAP package inherits from sklearn classes:

```
import umap
from sklearn.datasets import load_digits
digits = load_digits()
embedding = umap.UMAP().fit_transform(digits.data)
```

- Several parameters can be set; major ones are:
  - **n_neighbors**: Determines # of neighboring points used in local approximations of manifold structure.
  - **min_dist**: Controls how tightly embedding is allowed compress points together.
  - **metric**: Metric used to measure distance. User defined function can also be passed as long.
- An example of making use of these options:

```
import umap
import umap.plot
from sklearn.datasets import load_digits
digits = load_digits()
embedding = umap.UMAP(n_neighbors=5,
                      min_dist=0.3,

metric='correlation').fit_transform(digits.data)
umap.plot.points(mapper, label=digits.target)
```

Installation (Python)

- UMAP depends upon scikit-learn, and thus scikit-learn's dependencies such as numpy and scipy. UMAP adds a requirement for numba for performance reasons.
- Requirements:

  numpy
  scipy
  scikit-learn
  numba
- Recommended packages:

  pynndescent

  For plotting:

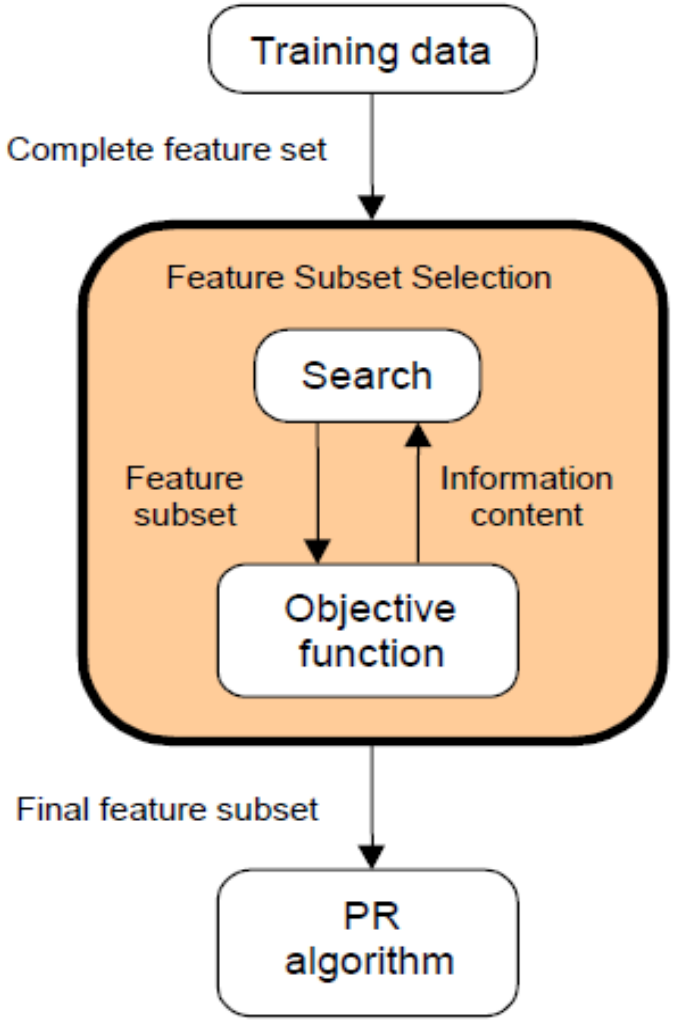
  matplotlib
  datashader
  holoviews
- Installation:

```
pip install umap-learn
```

Source: UMAP | GitHub

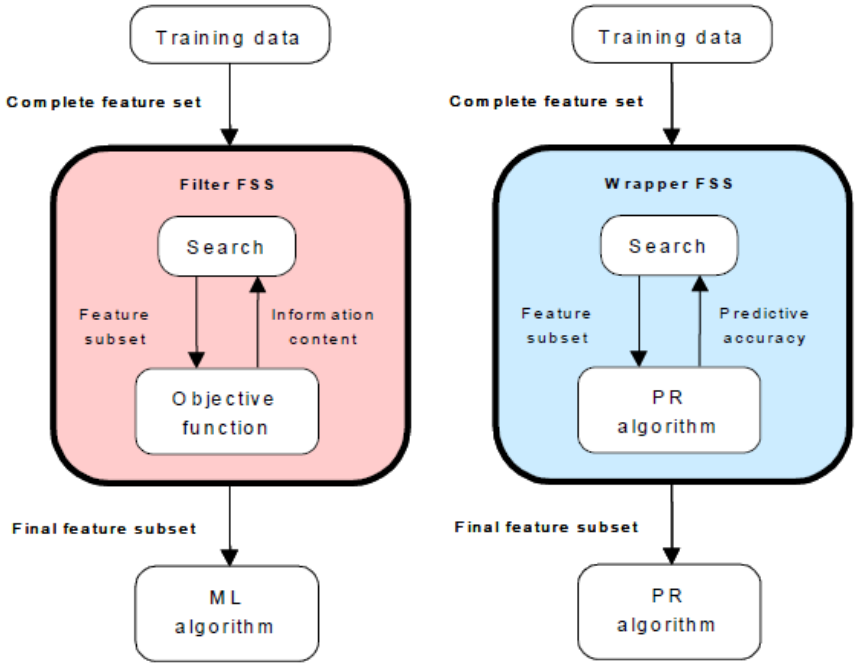
Feature Selection

- Process of selecting a subset of relevant features (variables) for use in model construction
- Premise: Data contains features that are redundant or irrelevant
- Involves a combination of *search technique* and *evaluation measure*
- Total enumeration not practical
  - 10 out of 100 variables involves $>10^{13}$ subsets
- Choice of evaluation metric influences algorithm
  - ***Wrappers***, ***Filters*** and ***Embedded Methods***



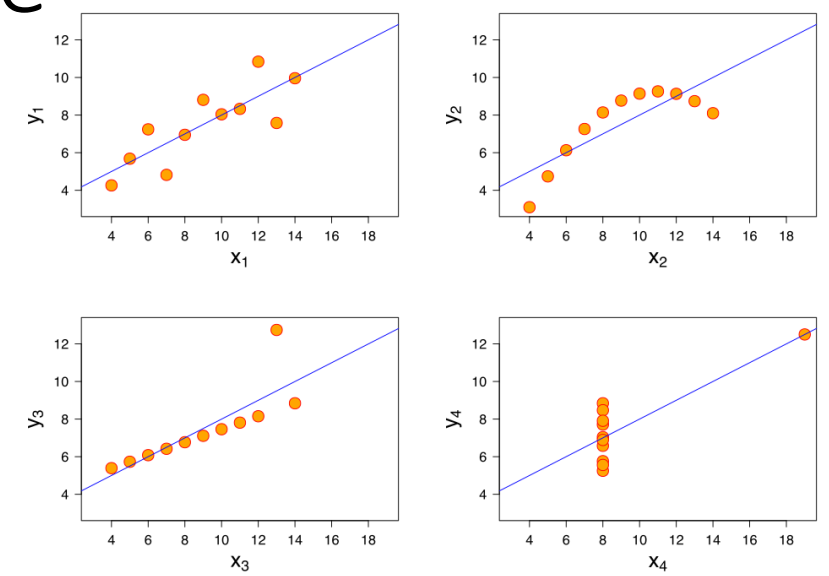
Feature Selection: Filters, Wrappers, Embedded Methods

- ***Wrappers***: Use a predictive model to score feature subsets
  - Computationally intensive, but provide best feature set for that particular type of model
- ***Filters***: Use a proxy measure instead of error rate to score a feature subset
  - Measure is chosen to be fast to compute, while still capturing the usefulness of feature set
  - Examples: Mutual Information, Correlations
- ***Embedded Methods***: Perform feature selection as part of model construction process
  - Lasso, Stepwise Regression, Decision Trees

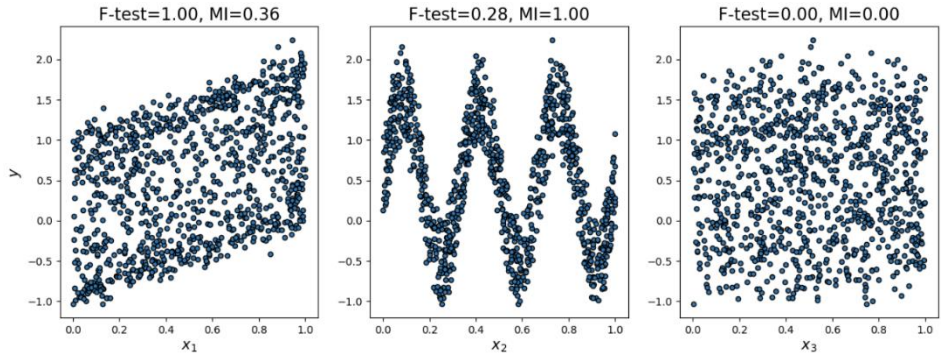


Filters: Few Examples & Python Code

- ***F-Test***: Hypothesis test to compare **regression** models: $Y = \beta_0 + \epsilon$ and $Y = \beta_0 + \beta_1 X + \epsilon$, where X is feature of interest.
  - Errors from both models are compared for significant difference
  - Only captures linear relationship
  - Equivalent to working with statistical correlation
  - Python: Scikit learn package
    sklearn.feature_selection.f_regression
    sklearn.feature_selection.f_classif
- ***Chi-Square Test:*** Used for **categorical features** and **classification**
  - Sensitive to small frequencies
  - Python: Scikit learn package
    sklearn.feature_selection.chi2(X, y)
- ***Mutual Information****:* Determines how similar the joint distribution $p(X, Y)$ is to the products of factored marginal distribution $p(X)p(Y)$; If two variables are independent, their MI is zero. $0 \leq MI \leq 1$
  - Unlike correlation, mutual information contains information about all dependence—linear and nonlinear.
  - Handles both continuous and discrete values easily
  - Python: Scikit learn package | Example
    sklearn.feature_selection.mututal_info_regression
    sklearn.feature_selection.mututal_info_classif



Anscombe's quartet. All four sets are identical when examined using simple summary statistics.| Source



F-Test captures the linear relationship well. MI captures any kind of relationship between two variables.| Source
Python Code

Feature Selection: Matlab

- *Matlab*: Statistics and Machine Learning Toolbox function 'sequentialfs' carries out sequential feature selection. Input arguments include predictor and response data and a function handle to a file implementing the criterion function. Optional inputs allow you to specify SFS (sequential forward selection) or SBS (sequential backward selection), required or excluded features, and the size of the feature subset.
- *Matlab Central*: Feature Selection Library: Library supports a number of feature selection methods, including:
  - Filters: Correlation based ranking of input features, Laplacian, and others.
  - Wrappers: Mutual Information, Minimum Redundancy Maximum Relevance (mRMR), Relief-F (for both classification and regression), and the very promising Infinite feature selection method (for classification).
- *Matlab Central*: Evolutional Feature Selection Toolbox: Allows application of Metaheuristics and Evolutionary Algorithms to Feature Selection in Various Modes:
  - Selection of variable number of features using Genetic Algorithm (GA): Fixed number of best features selection (e.g., the '5' most important features) or as a discrete combinatorial optimization problem using Simulated Annealing and/or Ant Colony Optimization (ACO)
  - As a real-valued optimization problem using Particle Swarm Optimization (PSO)
  - Multi-objective feature selection using Non-dominated Sorting Genetic Algorithm II (NSGA-II).

High-Dimensional Data Visualization & Feature Selection: Python Case Study

WAYNE STATE UNIVERSITY

Feature Selection & Visualization: Breast Cancer Dataset

Sources:

- Primary: Kaggle Blog
- Secondary: Scikit Learn Feature Selection

The goal here is to illustrate feature visualization and selection using several methods.

- Feature selection with correlation, univariate feature selection, recursive feature elimination, recursive feature elimination with cross validation and tree based feature selection methods are used with random forest classification.
- Apart from these, principle component analysis are used to observe number of components.

Breast Cancer Wisconsin (Diagnostic) Dataset: **Predict whether the cancer is benign or malignant!**

Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image. n the 3-dimensional space is that described in this paper: Bennett, K. P., & Mangasarian, O. L. (1992). Robust linear programming discrimination of two linearly inseparable sets. Optimization methods and software, 1(1), 23-34. | Link | Dataset: UCI Machine Learning Repository

Attribute Information:

- ID number,
- Diagnosis (M = malignant, B = benign),
- Ten real-valued features are computed for each cell nucleus: a) radius (mean of distances from center to points on the perimeter) b) texture (standard deviation of gray-scale values) c) perimeter d) area e) smoothness (local variation in radius lengths) f) compactness (perimeter^2 / area - 1.0) g) concavity (severity of concave portions of the contour) h) concave points (number of concave portions of the contour) i) symmetry j) fractal dimension ("coastline approximation" - 1.0).
- The mean, standard error and "worst" or largest (mean of the three largest values) of these features were computed for each image, resulting in 30 features. For instance, field 3 is Mean Radius, field 13 is Radius SE, field 23 is Worst Radius.

All feature values are recoded with four significant digits. Missing attribute values: none. Class distribution: 357 benign, 212 malignant.

If there is abundant data, *feature selection* might not be necessary for modeling.

Feature selection becomes more important with data sparsity. We of course always prefer compact models with fewer features and decent accuracy over complex models with slightly better performance (Occam's Razor)!

The Breast Cancer Dataset is somewhat sparse with over 30 features and 569 observations.

Canvas Course Website:

- Python Jupyter Notebook Code
- HTML Output
- Breast Cancer Dataset