Class Management

- The most recent quiz has been graded (Avg. 5.3/11)
- The reports for the team projects should be submitted by Friday evening
- At minimum your reports should contain the following components
  - Introduction: Summarize the dataset you have chosen and what your analysis achieved
  - Methods: Describe your approach to analysis, algorithms, etc.
  - Discussion: Insights gained by your work as well supporting visualizations, plots, and figures
  - Conclusion

Review of Material for Final Exam

- All assessment points should be completed by now (except latest quiz)
- Coverage will be cumulative since the beginning of the semester
- Provide some review about what to expect

Supervised Learning

- Supervised Learning Algorithms
  - Linear Discriminant Analysis
  - Support Vector Machines
  - Nearest Neighbor Methods
  - Regression and Regularization
  - Decision Trees (Classification and Regression Trees)

Unsupervised Learning

- Unsupervised Learning
  - Techniques for Dimensionality Reduction
  - Principal Components Analysis
  - Locally Linear Embedding
  - other matrix factorizations
  - Hierarchical and k-means clustering

Subdisciplines in Visualization

- *Scalar and Vector Visualization*: scalar functions, streamlines, etc.
- *Scientific Visualization* (scivis): datasets contain samples of continuous functions over subsets of Euclidean Space
- *Information Visualization* (infovis): visual representation of more abstract data such as graphs, text

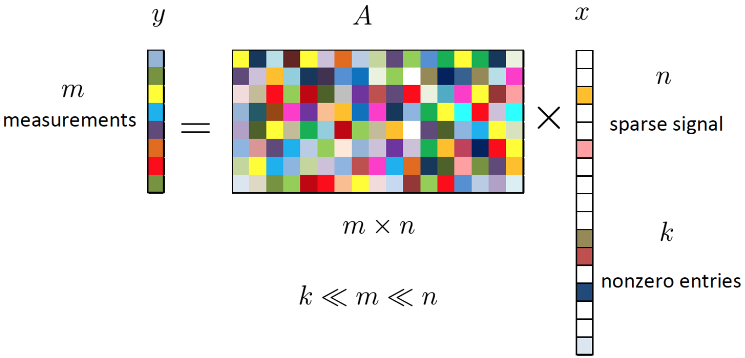
Graph Theory

- Definition: Nodes and Edges
- Terminology: degree, size, connected component, etc.
- Types of topologies: scale-free, random, complete, ...
- Linear algebraic methods
- Algorithms for analyzing the properties of networks and applications to social networks

Other Topics

- Cross-Validation
- Confusion Matrices
- Receiver Operating Characteristic
- Statistical Significance
- Lagrange Multipliers
- Text Processing, Latent Semantic Analysis
- Probability and Linear Algebra
  - Change of basis, inner products, norms, rank
  - Eigenvectors and Eigenvalues
  - Gram-Schmidt Process
  - Sparsity

Picture of the Measurement Process



- *Measurement* process visualized

Recovering Sparse Solution

- In principle we can find the sparsest solution by minimizing of the zero-norm
- $||x||_0$ is the number of non-zero entires in x

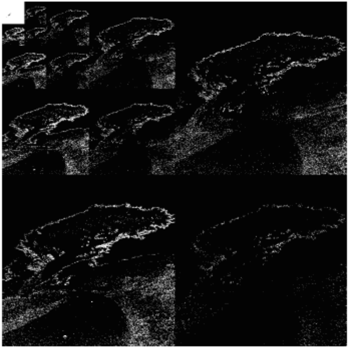
$$\min ||x||_0, \text{ such that, } y = Ax \tag{1}$$

Greedy or Matching Pursuit

- Greedy: make the naive best choice at every iteration and build the estimate of x recursively
- *Matching Pursuit* algorithm
  - Select best dictionary element (d_i, y)
  - Update the coefficient and determine new residual

Sparse Representations of Images


(a)


(b)

- Oftentimes natural signals are *sparse* in a suitable basis

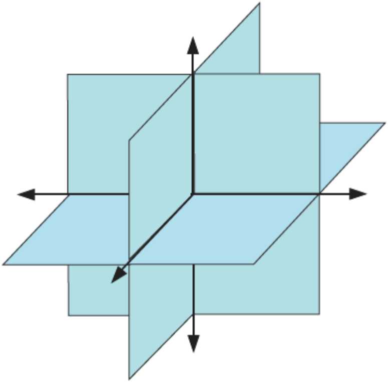
Reconstruction from Largest Wavelet Coefficients

(a)

(b)

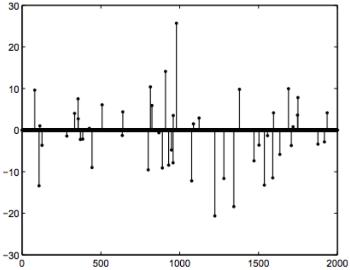
- Reconstruction obtained with the 10 percent largest wavelet coefficients

Geometry of Sparse Coding

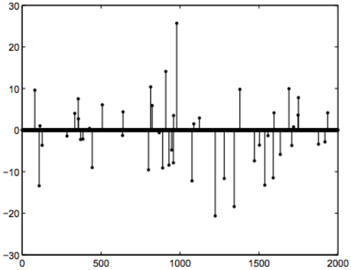


- The 2-sparse signals or vectors in three dimensional space

Example of Sparse Signal Recovery



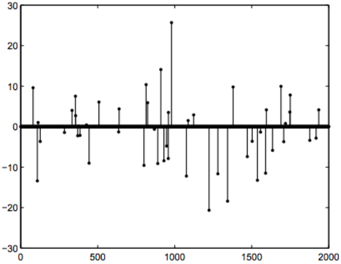
Sparse source signal



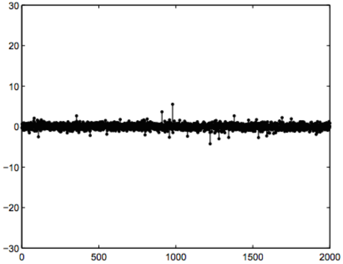
Perfect recovery by ℓ_1-norm minimization

- Sparse signal with dimension 2000 and 0-norm 50
- Recovered from 400 noise-free observations

Example of Sparse Signal Recovery



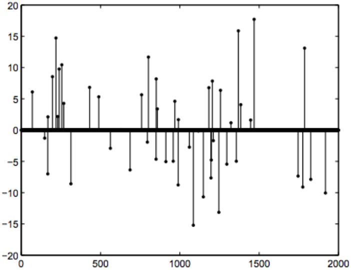
Sparse source signal



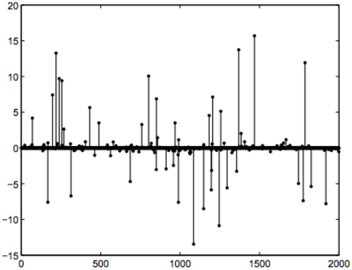
Estimated by ℓ_2-norm minimization

- Sparse signal with dimension 2000 and 0-norm 50
- Recovered from 400 noise-free observations,

Example of Sparse Signal Recovery



Sparse source signal



Estimated by ℓ_1-norm minimization

- Sparse signal recovery but with noisy observations

Locally Linear Embedding

- Algorithm for non-linear dimensionality reduction
- Developed due to the inherent limitations in linear methods

Covariance and Sample Covariance

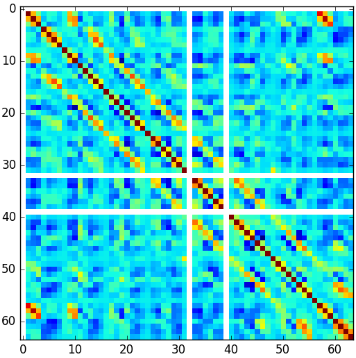
- Many simple data analysis problems involve finding the eigenvectors of the sample covariance matrix given the data
- Test for relationships between variables/features
- PCA: Gives a natural basis in which to express the data

$$\mathbf{R} = E[\mathbf{x}^T \mathbf{x}] \tag{2}$$

$$R = \frac{1}{n-1} \sum_{i=1}^{N} (x_i - \mu)^T (x_i - \mu) \tag{3}$$

$$\frac{1}{n-1} \mathbf{X}^T \mathbf{X} \tag{4}$$

Visualizing the Correlation or Covariance

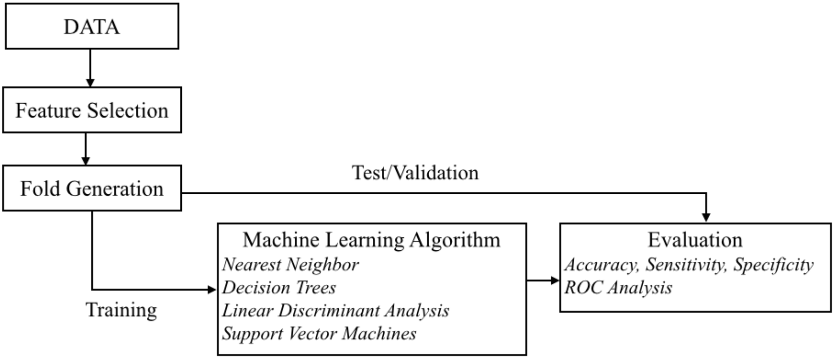


- Rendering the Correlation as an image immediately reveals which variables are related

Overview of Typical Machine Learning Workflow

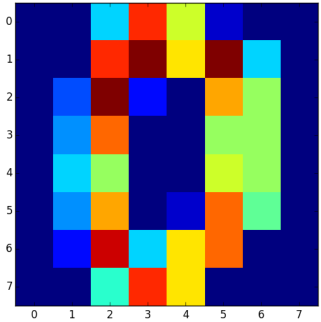
Box 1

A (supervised) ML workflow will minimally contain: *data*,*feature selection process*, *cross-validation*,*an ML algorithm*, and a means to *evaluate* performance



Working with the zipcode "digits" dataset

- Dataset of 8×8 images of hand-written characters



Linear Discriminant Analysis

- Linear Discriminant Analysis (LDA) is a supervised learning technique that attempts to provide a one-dimensional projection of the data such that discrimination between classes is maximized
- As in previous examples $\{x_1, x_2, \ldots, x_n\}$ we have a set of observations where each observation belongs to one of a finite number of classes

LDA

- Attempt to find some *linear* function of the values of feature values such that we can use the output to classify new data instances
- Which coefficients w maximize separability?

$$\phi = w^T x \tag{5}$$

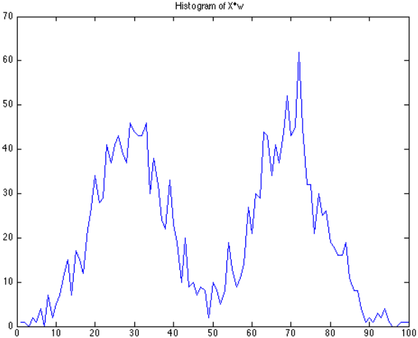
Performance Metrics

- In any binary classification task two types of error can occur, false positives and false negatives
- In general we need to keep track of both of these errors to understand how well our classifier is performing
- The *Receiver Operating Characteristic* (ROC) keeps track of both of these error rates as we vary the threshold

$$\text{threshold} > w^T x \tag{6}$$

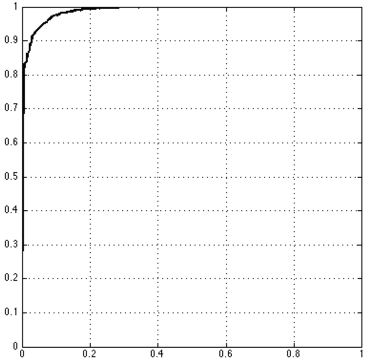
Receiver Operating Characteristic

- Imagine we look at a histogram of our test statistic
- Our threshold will fall somewhere along the x-axis and will determine the two error rates



Example

- Each point along this curve corresponds to a single value of the threshold parameter



LDA recipe

- The LDA solution can be found explicitly in terms of the data by differentiating the objective with respect to w and setting the result equal to 0

$$\frac{d}{dw} J(w) = 0 \tag{7}$$

- When we work this out (homework) the solution we arrive at is equal to the following

$$\hat{w} = S_W^{-1}(\mu_1 - \mu_2) \tag{8}$$

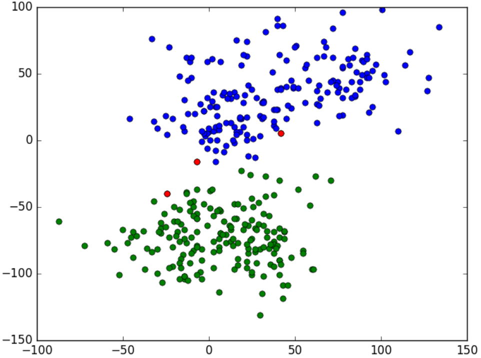
Defining a Symmetry Feature

```
def symlr(t):
    s = np.fliplr(t)
    y = t - s
    val = np.sum(y[:, 0:3])
    return val


def symud(t):
    s = np.flipud(t)
    y = t - s
    val = np.sum(y[0:3, :])
    return val
```

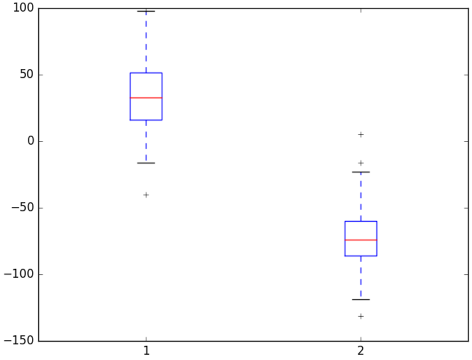
- The above functions calculate a symmetry measure on our 8 by 8 images

Symmetry features for digits 5 and 6



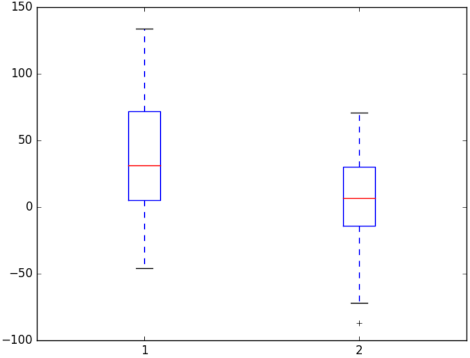
- Are both features equally discriminative?

Discriminative Feature



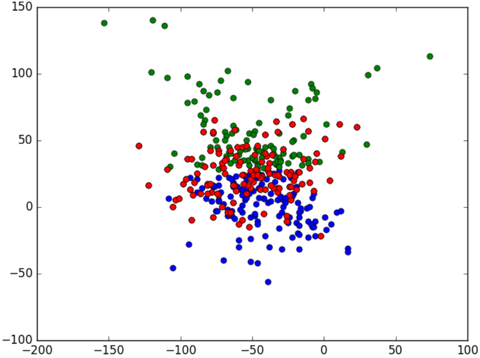
- Box-Plot Shows clear difference between the two classes

Non-discriminative Feature



- Still significant difference, but classes are overlapping

Same result for digits 3 and 9



- Less discriminant power in these features for a different digits

Solution

- Solve constrained optimization with lagrange multipliers

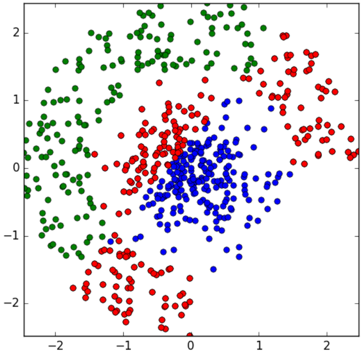
$$L_P = (1/2)||\beta||^2 - \sum \alpha_i [y_i (x_i^T \beta + \beta_0) - 1] \tag{9}$$

$$\beta = \sum \alpha_i y_i x_i \tag{10}$$

$$0 = \sum \alpha_i y_i \tag{11}$$

$$f(x) = \sum_{i=1}^{N} \alpha_i y_i (x_i, x) + \beta_0 \tag{12}$$

Linear Discriminant Analysis Solution



- High error rate since decision boundary is only a hyperplane

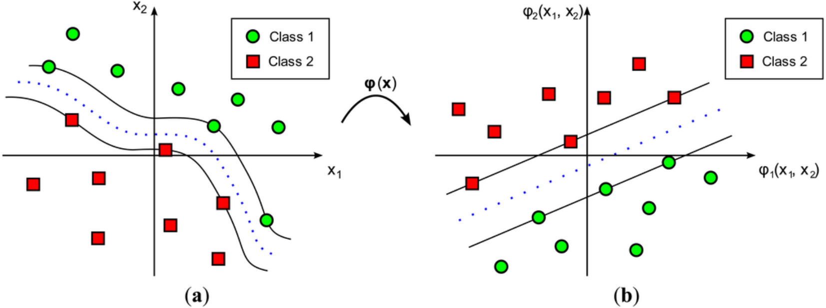
The Classification Rule

- Usage of the kernel trick results in the following classification rule
- The kernel function is selected in advance the algorithm finds the weights as well as the support vectors that maximize the margin

$$\sum_{s \in \text{support vectors}} \alpha_s K(x_{test}, x_s) + \text{bias term} > 0 \tag{13}$$

Non-linear Mapping to a new Feature Space

- Non-linear boundary in the original feature space becomes non-linear in the new feature space



Common Kernels

- The polynomial, radial basis function, linear and neural net are the most common kernels

$$K(x, y) = (xy^T + 1)^p \tag{14}$$

$$K(x, y) = e^{-||x-y||^2/2\sigma^2} \tag{15}$$

$$K(x, y) = \tanh(kxy^T - \delta) \tag{16}$$

Higher Dimensional Feature Space, Polynomial

- Higher dimensional feature space based on polynomials would look something like the following

$$\mathbf{z} = [1, x_1, x_2, x_1 x_2, x_1^2, x_2^2] \tag{17}$$

- Vectors in the transformed feature space are then 6 dimensional $\mathbf{z} = [z_1, \ldots, z_6]$
- The advantage of SVM is that it allows to work in the higher dimensional feature space without having to actually compute the *vectors* in the higher dimensional space
- What is the kernel function for the z space in terms of x?

Non-linear Support Vector Machines

- Describe the problem and solution in the z space

$$L = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j z_i^T z_j \tag{18}$$

$$f(x) = \mathrm{sgn}(w^T z + b) \tag{19}$$

$$w = \sum_{z_i \in \mathrm{SV}} \alpha_i y_i z_i \tag{20}$$

$$y_i(w^T z_i + b) = 1 \tag{21}$$

- All the elements of the problem and solution can be stated in terms of the inner product alone

Polynomial Kernel

- Recall the Polynomial Kernel Given above, lets expand it for $d = 2$ and a two dimensional input space

$$K(x, y) = (1 + x^T y)^d$$

$$K(x, y) = (1 + x^T y)^2$$

$$K(x, y) = (1 + x_1 y_1 + x_2 y_2)^2$$

$$K(x, y) = 1 + 2x_1 y_1 + 2x_2 y_2 + (x_1 y_1)^2 + (x_2 y_2)^2 + 2x_1 y_1 x_2 y_2$$

Establishing the Existence of the Transformed Space

- Recall that the support vector machine relied on an implicit transformation to a new space
- Since we do not actually compute the feature vectors in the new space we can seemingly choose *any* function of two arguments for the kernel

$$K(x, y) = (\phi(x), \phi(y))$$

- In certain cases the existence of the transformed space is guaranteed by Mercer's Theorem

The Gram Matrix

- The Gram Matrix is a matrix of inner products of all the different pairs of data points

$$K = \begin{bmatrix} K(x_1, x_1) & K(x_1, x_2) & \cdots \\ K(x_2, x_1) & K(x_2, x_2) & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix} \tag{22}$$

- Can also be expressed in terms of the data matrix

$$K = XX^T$$

Mercer's Theorem

- Mercer's Theorem says that the gram matrix should be *positive semidefinite*
- Assumption: $K(x, y)$ is a symmetric function
- Theorem: $K(x, y)$ can be expressed as an inner product $K(x, y) = (\phi(x), \phi(y))$ if the gram matrix is positive semidefinite for any possible collection of vectors $\{x_1, x_2, \cdots, x_N\}$

Splitting Criterion

- For any feature the best split point s can be found very quickly for a given feature
- Therefore it is also easy to scan over all of the features and determine the best split point
- A given iteration outputs a pair: (feature to split over, the split point)

Tree Parameters

- Large tree might overfit the data (number of nodes)
- One option to control the complexity by only splitting if the error improves greater than some threshold value
- Other options available in learning packages include setting the maximum depth of the tree or the minimum number of nodes in a given leaf

Splitting Criterion for Classification

- For classification we split based upon a different error criterion, the mis-classification rate in each node

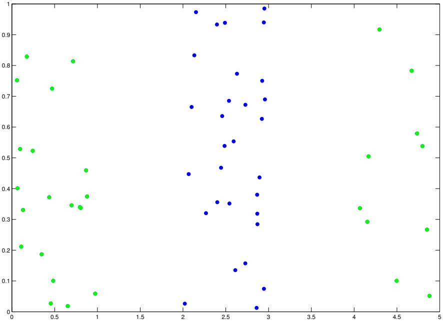
$$\frac{1}{N_m} \sum_{i \in R_m} I(y_i \neq k(m)) \tag{23}$$

Algorithm for Growing a Tree

- Split the nodes in a *greedy* fashion
- Step 1: Begin with root node containing all the training examples
- Step 2: For each feature find the split minimizes node impurity or error in the two child nodes and pick minimum over all features and all splits
- Step 3: Stop splitting when criterion is met, or apply Step 2 recursively

Simple Example

- Two dimensional case where data fits very neatly into rectangular regions
- Obviously only the first feature (horizontal axis) is important here
- Assume the three regions contain respectively 20, 30, and 10 examples



Regression

- In *regression* dependent variables is modeled as a function of some set of *explanatory* variables
- *linear regression*: the underlying model is assumed to be linear and can be expressed as matrix plus an error term

$$\mathbf{y} = \mathbf{X}h + \epsilon \tag{24}$$

$$\hat{h} = \left(X^T X\right)^{-1} X^T y \tag{25}$$

Ordinary Least Squares

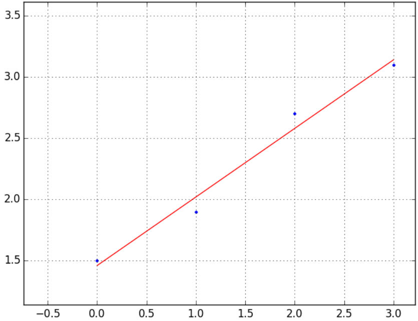
- One explanatory variable and one dependent variable the problem becomes easy to visualize
- We are given pairs of values (x, y): $(0, 1.5), (1, 1.9), (2, 2.7), (3, 3.1)$

$$y = Ah \tag{26}$$

$$\begin{bmatrix} 1.5 \\ 1.9 \\ 2.7 \\ 3.1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} h_0 \\ h_1 \end{bmatrix} \tag{27}$$

Sample Problem

- linear fit $y = h_0 + h_1 x$



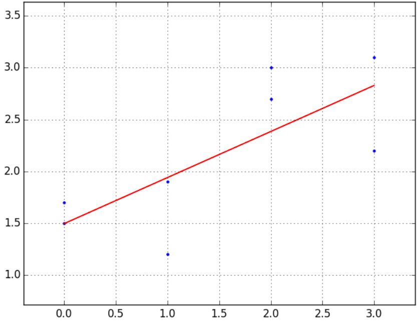
Ordinary Least Squares

- In general we may have multiple observations for a given value of the dependent variable

$$
\begin{bmatrix} 1.5 \\ 1.9 \\ 2.7 \\ 3.1 \\ 1.7 \\ 1.2 \\ 3.0 \\ 2.2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 0 \\ 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} h_0 \\ h_1 \end{bmatrix} \tag{28}
$$

Sample Problem

- linear fit $y = h_0 + h_1 x$



Ordinary Least Squares

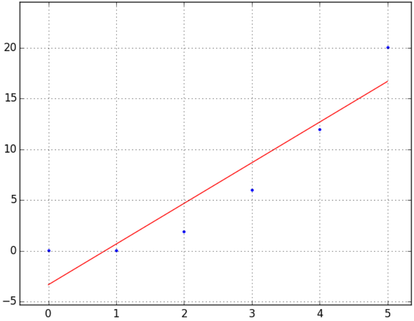
- Our data might have a more complicated relationship on the explanatory variables

$$y = Ah \tag{29}$$

$$\begin{bmatrix} 0.0319 \\ 0.0313 \\ 1.9135 \\ 5.9970 \\ 11.9835 \\ 20.0628 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \\ 1 & 5 \end{bmatrix} \begin{bmatrix} h_0 \\ h_1 \end{bmatrix} \tag{30}$$

Sample Problem

- linear fit $y = h_0 + h_1 x$



Ordinary Least Squares

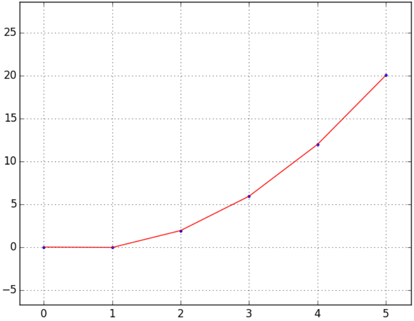
- Introduce a quadratic term in the design matrix

$$y = Ah \tag{31}$$

$$\begin{bmatrix} 0.0319 \\ 0.0313 \\ 1.9135 \\ 5.9970 \\ 11.9835 \\ 20.0628 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \\ 1 & 4 & 16 \\ 1 & 5 & 25 \end{bmatrix} \begin{bmatrix} h_0 \\ h_1 \\ h_2 \end{bmatrix} \tag{32}$$

Sample Problem

- quadratic fit $y = h_0 + h_1 x + h_2 x^2$



Ridge Regression and LASSO

- Two common means of regularizing least squares are called ridge regression and lasso
- Ridge regression: $J(h) = ||y - \mathbf{X}h||_2 + \lambda \sum_k h_k^2$
- LASSO: $J(h) = ||y - \mathbf{X}h||_2 + \lambda \sum_k |h_k|$
- Consider what happens as the parameter λ is varied from 0 to infinity

Characteristic Polynomial

- The Characteristic Polynomial of a matrix is computed by means of a special determinant
- The roots of characteristic polynomial are the *eigenvalues*
- Eigenvectors are vectors satisfying the expression

$$\Delta(\lambda) = |(\lambda I - A)| \tag{33}$$

$$Rv = \lambda v \tag{34}$$

$$0 = (\lambda I - R)v \tag{35}$$

Diagonal Factorization, Spectral Decomposition

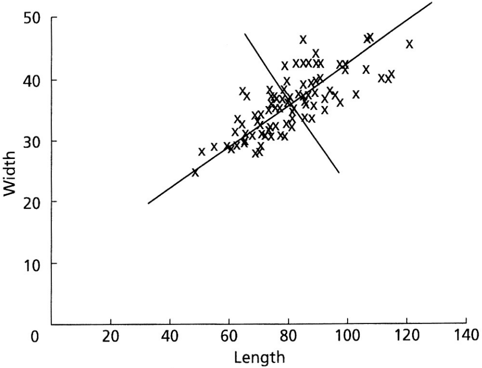
- Reversing the process we can write a matrix as a similarity transformation of a diagonal matrix
- If A is real-symmetric then the roots of its characteristic polynomial are guaranteed to be real.
- Eigenvectors belonging to distinct eigenvalues are orthogonal to one another $u \cdot v = 0$

$$A = PDP^{-1} \tag{36}$$

Principal Component Analysis in Two Dimensions

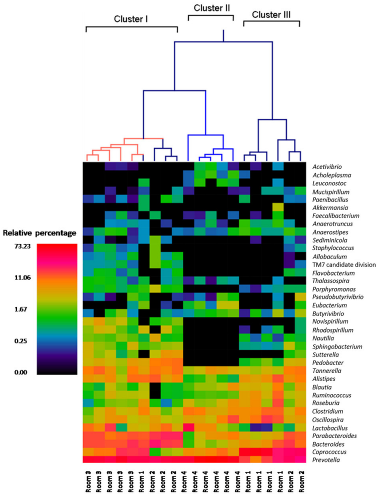
Box 1

A (supervised) ML workflow will minimally contain: *data*,*feature selection process*, *cross-validation*,*an ML algorithm*, and a means to *evaluate* performance



Clustering

- Methods: Hierarchical and k-means
- Look at the example again and what conclusions should we draw from this experiment?
- Important to observe that the dendrogram effectively defines a permutation of the features

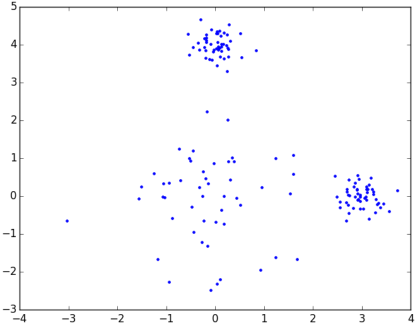


Clustering

- Divide an (unlabeled) dataset into a fixed number of clusters (or groups) such that the instances within each cluster are similar to one another and different from the instances in other clusters
- Purpose: Discover the underlying structure in some dataset, or to find an optimal reduced representation
- k-means
- Gaussian Mixture Models with E-M algorithm
- Hierarchical Clustering

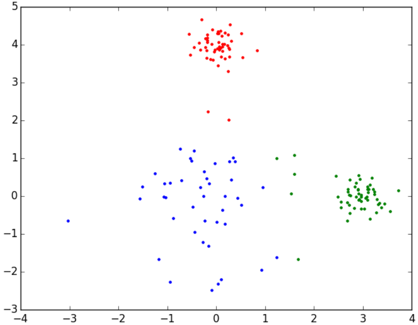
Example

- Easily visualized for two-dimensional data



Distinction between clustering and classification

- In classification problems we know already some set of classes to which each of our data instances belongs
- In clustering, conversely we have a completely flat dataset it is up to us to discover the structure



Clustering Function

- If we write our data instances $a_1, \ldots, a_n$ then we need some metric to quantify how well our data is clustered
- We can use the euclidean distance for this task: $d(a_i, a_j) = ||a_i - a_j||_2^2$
- Assume that we know in advance the number of clusters we want to fit to our data K
- A clustering is a function that assigns a label to each data instance: $f : \text{data} \to \{1, \ldots, K\}$

Objective for Clustering

- The clustering problem can also be formulated as an optimization
- Here f is the classification and n_k is the number of instances assigned to class k

$$J = \frac{1}{2} \sum_{k=1}^{K} \frac{1}{n_k} \sum_{f(i)=k, f(j)=k} d(a_i, a_j) \tag{37}$$

Clustering for Euclidean Distance

- It can be shown that the objective reduces as follows for the case of the Euclidean Distance, where A_k is the average data instance in group k

$$J = \sum_{k=1}^{K} \sum_{f(i)=k} ||a_i - A_k||_2^2 \tag{38}$$

- This suggests a more general approach where we iteratively minimize first over the groupings then the cluster centers

K-Means algorithm

$$J = \sum_{k=1}^{K} \sum_{f(i)=k} ||a_i - F_k||_2^2 \tag{39}$$

- Fix the number of clusters and pick an initial (random) guess of the cluster centers F_k
- Follow two steps until convergence:
- Assign each point to the nearest cluster center
- Recalculate the cluster centers as the average of all of the points in the cluster

Aspects of the K-Means Algorithm

- The cluster centers become a representative point for *all* of the points in a given cluster
- The minimum distance criterion partitions the space into a set of convex polyhedra
- Although the algorithm is guaranteed to converge the solution can be highly dependent upon the initial conditions, often desirable to run the algorithm multiples times and choose the result with the best (smallest) within cluster variance
- Another name for k-means is *vector quantization*
- Combinatorial dependence on the size of the dataset: infeasible to check all possible solutions

K-Means in scipy

- Scipy provides two options for implementing the k-means algorithm: `scipy.cluster.vq.kmeans` and `sklearn.cluster.KMeans`

Clustering Recap

- Last lecture: introduced clustering and a specific clustering algorithm called k-means
- *Clustering*: Process of dividing a dataset into clusters or groups such that all the instances in each cluster are similar to one another
- Properties of k-means: 1) number of clusters specified in advance (k) 2) Assignment dependent upon random initialization algorithm
- Algorithm: a) Cluster centers compute assignment b) Assignment compute new cluster centers

k-means versus Hierarchical Clustering

- Today we will introduce a new type of clustering algorithm called hierarchical clustering
- Unlike k-means the assignment will not depend upon initialization of the algorithm
- Also the number of clusters is not specified in advance
- Input data: measurement of distances between all pairs of data points, $(x_1, x_2, \ldots, x_n)$, use the matrix of distances, $d(x_i, x_j)$

Top-down and Bottom-up Clustering

- There are two types of hierarchical clustering algorithms: *agglomerative* and *divisive*
- Agglomerative: Begin with all data instances in separate clusters and repeatedly merge clusters until we have a single group (merge two 'nearest' clusters)
- Divisive: Opposite of agglomerative, begin with single large group and split until all clusters are singleton
- We will look agglomerative algorithms today
- From the definition we need a way to measure distances between two clusters (not only between data instances): this is known as the *linkage*

Linkages

- Given two groups of data instances $A = \{x_1, x_2, x_3\}$ and $B = \{x_4, x_5, x_6\}$
- Compare A and B using the distances their elements
- Distance here can be any metric we choose including the euclidean distance
- Goal: some function d that takes clusters (groups) as arguments and returns a number representing the dissimilarity of those clusters, $d(A, B)$

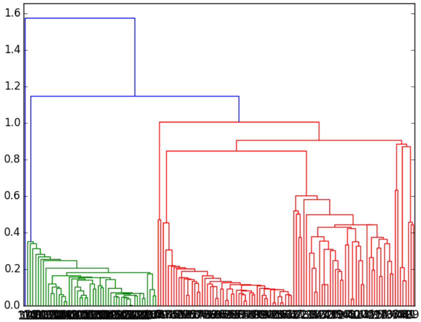
Single Linkage

- In single linkage we return the *smallest* distance between two groups
- Two groups are ‘close’ together if any of their points are near to each other

$$d(A, B) = \min_{i \in A, j \in B} d(x_i, x_j) \tag{40}$$

Visualizing Hierarchical Clustering with Dendrograms

- A *dendrogram* is a natural visual representation of the hierarchical clustering process (single linkage example)
- Cutting a dendrogram a fixed height gives a clustering of the data



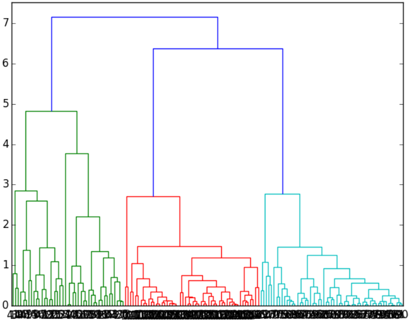
Dendrograms

- Tree where each node represents a cluster
- Leaf nodes represent singletons
- Root node contains all the data instances
- The height at which we draw a node is proportional to the dissimilarity of the two nodes
- Also the order of data instances is usually permuted so that the tree is more visually appealing (closer nodes placed near each other)

Complete Linkage

- In complete linkage we return the *maximum* distance between two groups
- Distance of the farthest pair

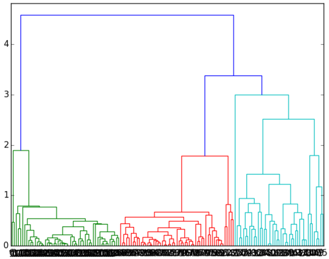
$$d(A, B) = \max_{i \in A, j \in B} d(x_i, x_j) \tag{41}$$



Average Linkage

- In complete linkage we return the *average* distance between all pairs of points in the two groups
- Distance of the farthest pair

$$d(A,B) = \frac{1}{|A||B|} \sum_{i \in A, j \in B} d(x_i, x_j) \tag{42}$$



Properties of hierarchical clustering

- As we run the algorithm (i.e. as we merge more nodes together) the distances between the clusters we are merging is always increasing
- For examples it is convenient to use the euclidean distance $||x - y||_2$ but in general we have many different distance metrics to choose from

Properties of hierarchical clustering

- As we run the algorithm (i.e. as we merge more nodes together) the distances between the clusters we are merging is always increasing
- For examples it is convenient to use the euclidean distance $||x - y||_2$ but in general we have many different distance metrics to choose from

Defining different metrics

- Different ways to compute distance between data instances

$$d(x, y) = \sqrt{\sum_{i=1}^{N} (x_i - y_i)^2}, \text{ Euclidean} \tag{43}$$

$$d(x, y) = \sum_{i=1}^{N} |x_i - y_i|, \text{ Taxicab} \tag{44}$$

$$d(x, y) = \max_i |x_i - y_i|, \text{ Chebyshev} \tag{45}$$

$$d(x, y) = \left(\sum_{i=1}^{N} (x_i - y_i)^p \right)^{1/p}, \text{ Minkowski} \tag{46}$$

Problems with Single and Complete Linkages

- Single linkage can result in clusters that are too spread out while complete linkage suffers from chaining
- Average linkage is a compromise between single and complete although it also has its own difficulties for example hard to interpret dendrogram

Problem Statement for Images

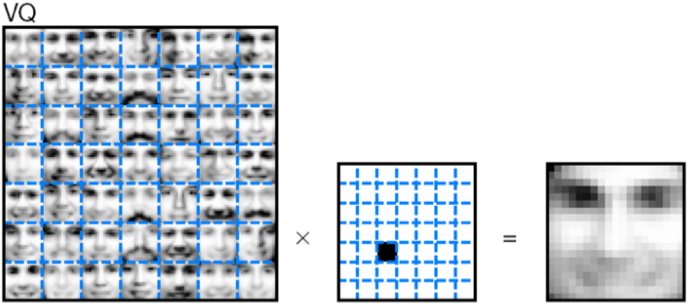
- Given a set of images we want to be able to do the following
- Create a set of basis images from which we can create new images by linear combination
- Find weights that produce any input image from the basis image
  - One set of weights for each input image

slide content adapted from materials from Marshall Tappen

Some possible solutions

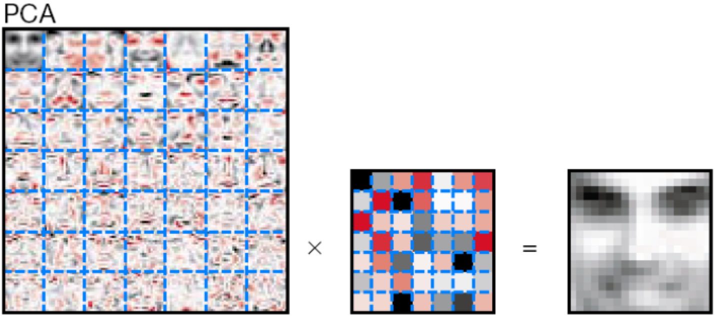
- Vector Quantization
- Principal Components Analysis
- Non-negative Matrix Factorization

Vector Quantization



- Find closest match
- Similar to nearest-neighbor classification
- *Limitations*: scales with number of basis images, provides no analysis

Principal Components Analysis

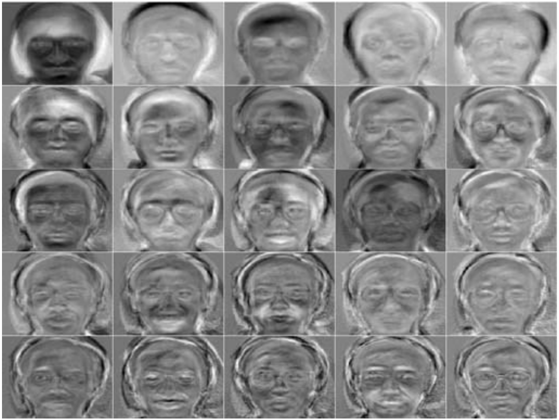


- Basis images are, by construction, orthogonal
- Reconstruction by linear combination

Aspects of PCA

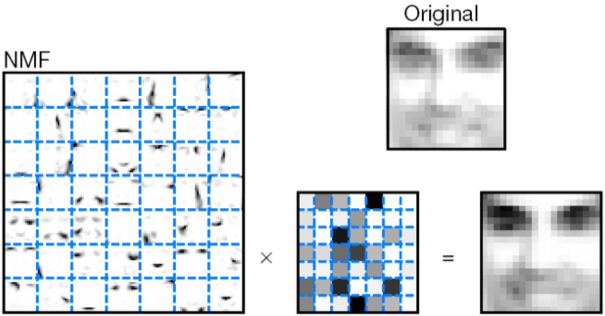
- PCA involves arbitrary linear combinations, we add some and se subtract some
- The basis images obtained do not necessarily correspond with our intuition
- In some contexts subtraction is not a sensible operation
- How is a face subtracted, what is subtraction in the context of document classification.

Eigenfaces Example



- Probably not how faces are represented in the brain

Non-negative matrix factorization

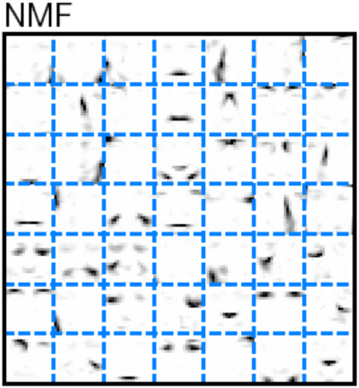


- Similar to PCA, but the coefficients cannot be non-negative

NMF Basis Image Properties

- Only allowing addition makes more intuitive sense in certain contexts and has some correspondence with how neurons operate
- Constraining the reconstruction coefficients to be positive often leads to nice basis images
  - Basis images represent different *parts* of the objects being studied

Non-negative matrix factorization

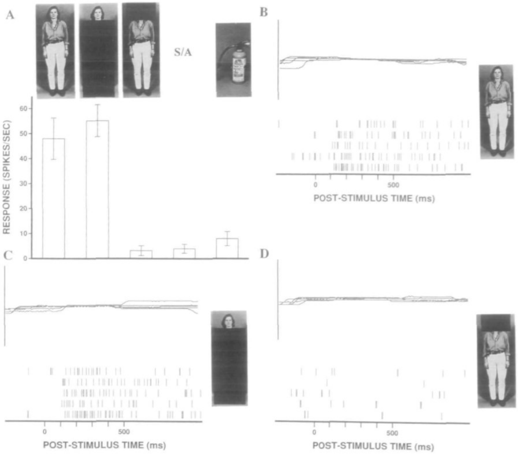


- The factorization has naturally found what we might think of as parts of the faces

Comparison of PCA and NMF

- PCA
  - Produces an optimal set of basis images
  - But this optimality might not be useful for your application
- NMF
  - Produces coefficients with a constraint
  - Can naturally produce a nicer basis, not constrained to be orthogonal

Evidence from Neuroscience



- Different visual stimuli presented to macaque monkey

Wachsmuth et al 1994

- Of 53 neurons,
- 32 percent responded to head only
- 9 percent responded to body only
- 41 percent responded to both the head and the body in isolation
- 17 percent responded to the whole body only
- Suggestive a parts based encoding of figures in the brain

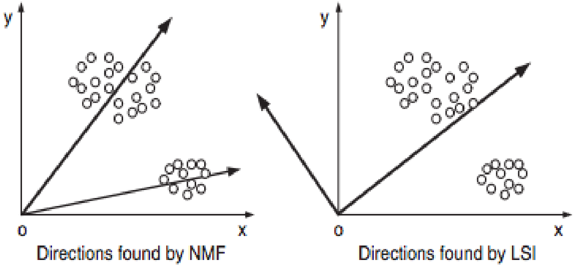
Non-negative Matrix Factorization

- Non-negative Matrix Factorization provides a similar decomposition of the dt matrix
- Naturally additive model since the values are all constrained to be non-negative

$$\arg\min_{W,H} ||X - WH||^2 = \sum_{i,j} X_{ij} - WH_{ij}$$

Document Clustering

- *Document Clustering*: Partitioning a corpus into a predefined number of clusters related to a coherent topic
- NMF applied to text analysis by Xu, Liu, Gong "Document Clustering Base on Non-negative Matrix Factorization"



NMF Document Clustering Algorithm

- *Document Clustering Algorithm*:
  - Construct the term-document matrix $\mathbf{X}$ from the given corpus
  - Find an NMF decomposition of $\mathbf{X}$
  - Normalize the factors U and V
  - Examine each column of V and look for the component with the largest value and assign the corresponding document to cluster k
- Standard Datasets for Document Clustering: NIST Topic Detection and Tracking (TDT2), Reuters dataset

Performing Non-Negative Matrix Factorization in Python

```
import numpy as np
from sklearn.decomposition import NMF
<...>
X = <...>
<...>
model = NMF(n_components=2, init='random', random_state=0)
W = model(fit(X))
H = model.components_
```

- Demonstration of the singular value decomposition routine in the numpy linear algebra package
- This can be used to experiment with LSA on a small document corpus

Task

- Write a script to perform NMF decomposition on a few simple matrices
- Characterize the error between the reconstruction and the original data as you change the rank
- Create a plot of error vs. rank