Foundations of Data Science
Lecture 2, Module 3

Rumi Chunara, PhD
CS6053

Major Tasks in Data Preprocessing

- **Data cleaning**
  - Fill in missing values, smooth noisy data, identify or remove outliers, and resolve inconsistencies
- **Data integration**
  - Integration of multiple databases, data cubes, or files
- **Finding Structure in Data:**
  - **Data reduction**
  - Dimensionality reduction
  - Numerosity reduction
  - Data compression
  - **Data transformation and data discretization**
  - Normalization
  - Concept hierarchy generation

Discussion

What do we mean by data having structure?

Why do we want to understand it?

What do we do with it?

Information

“Any quantity that can reduce uncertainty about another quantity”

Conditional thinking

So much of what we do in data science involves thinking in terms of…

$$E[Y|X]$$

And deciding whether or not $E[Y|X] \neq E[Y]$

From a decision making standpoint, conditional thinking is being able to make a better judgment about a potential outcome Y if we were to know the value of X.

Information Example

If a carnival operator wanted to reduce the uncertainty about your weight, what information might he use?

Targeted exploration

If we have a Target variable in our dataset, we often wish to:

1. Determine whether a variable contains important information about the target variable (in other words, does a given variable reduce the uncertainty about the target variable).

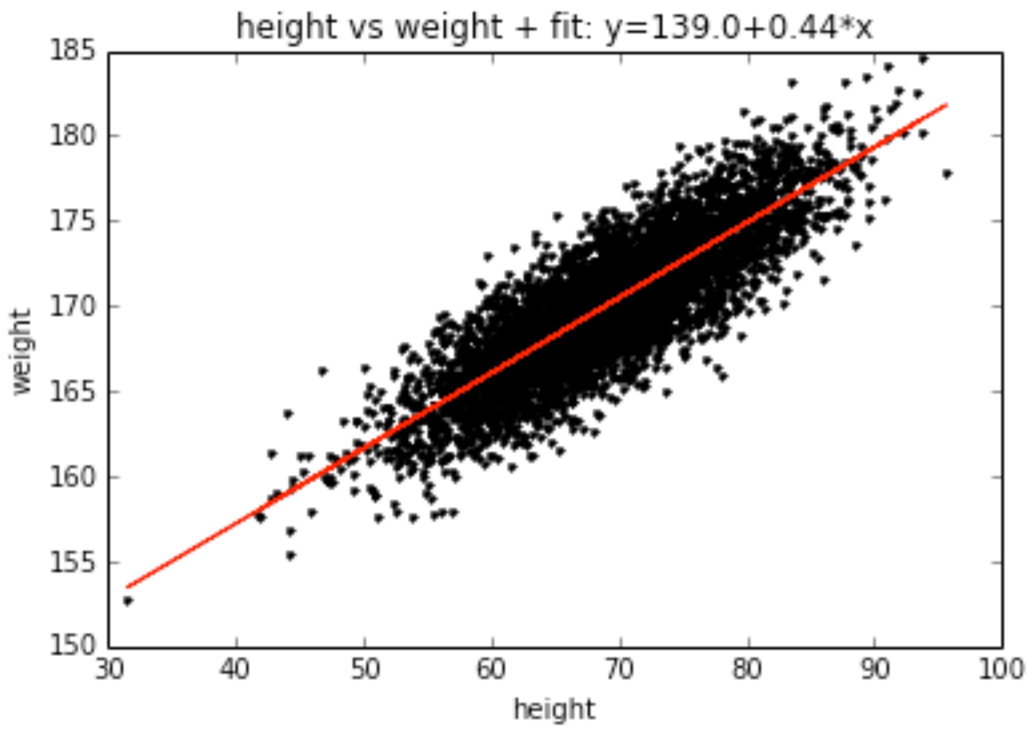
2. Obtain a selection of the variables that are best suited for predicting the target variables.

3. Rank each variable on its ability to predict the target variable.

Reducing Uncertainty

Let’s assume our carnival friend is also a data scientist

Step 1 – Collect data, height vs. weight, regress



By regressing we can learn:

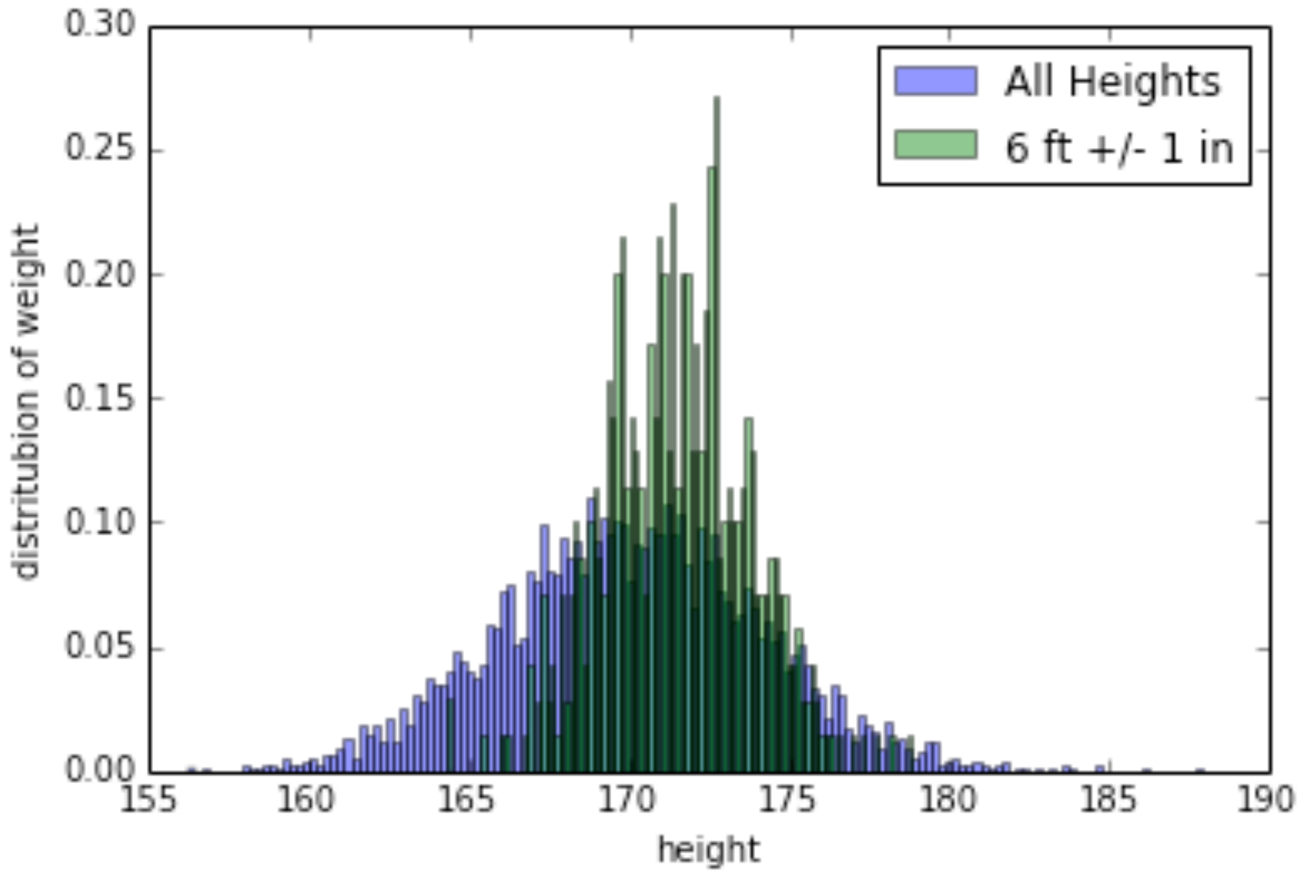
E[Weight|Height]

as opposed to

E[Weight].

Reducing Uncertainty

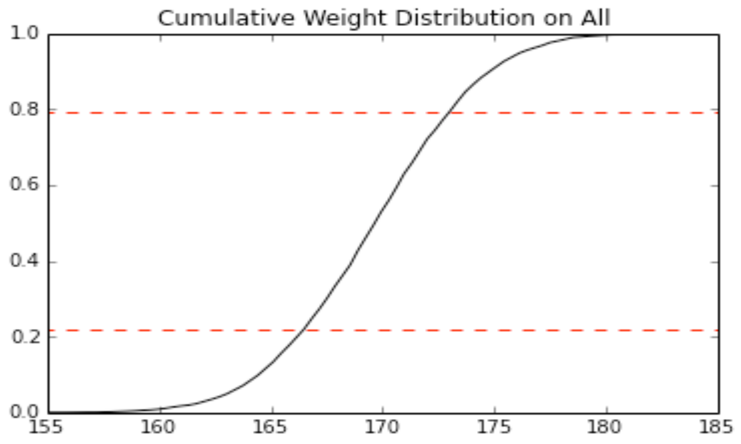
Step 2 – Based on prior knowledge, use the person's height to get a better range of possible weights.



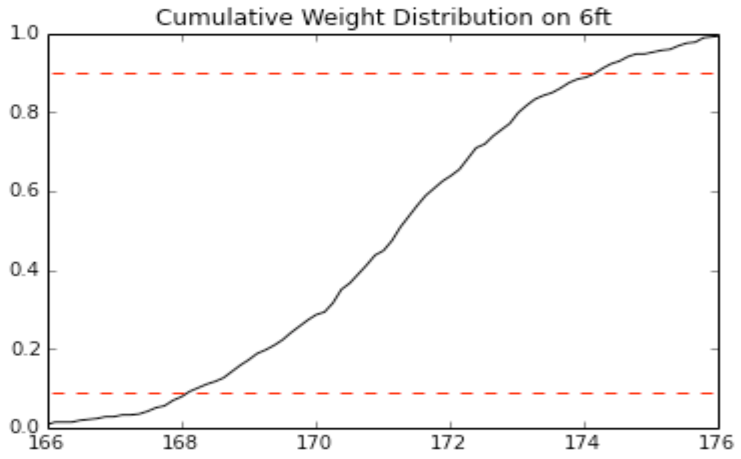
Reducing uncertainty

Step 3 – Guess E[W|H=72]. With a more informed, conditional expectation, he can reduce losses by 50% (40/100 vs. 20/100). That could be a lot of $$$.

Error Rate =
1-P(Correct | Guess E[W])
= 40%



Error Rate =
1-P(Correct | Guess E[W]|H=72)
= 20%



Entropy

This comes from Information Theory, and is a measure of the unpredictability or uncertainty of information content.

Certainty => Closer to 0% or 100% sure that something will happen.

Entropy is fundamental concept used for Decision Trees and Feature Selection/Importance, and is a tool that can satisfy the demands of targeted exploration.

Entropy

More formally…

X is a random variable with {x1, x2 … xn} possible values, the entropy is the expected "information" of an event, where "information" is defined as –log(P(X)).

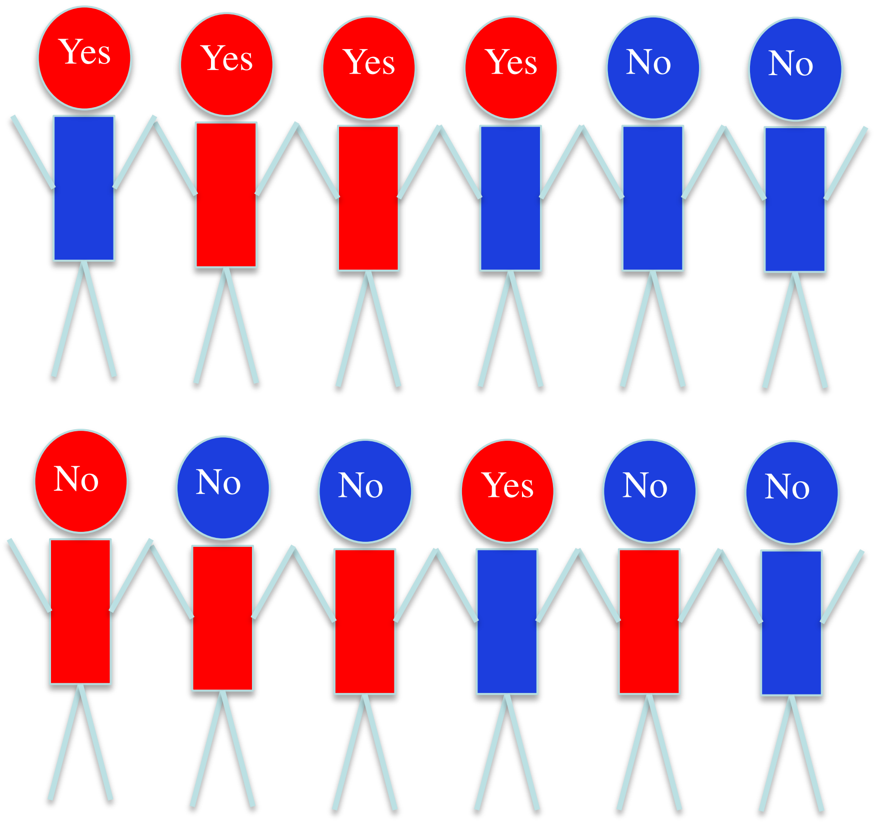
$$H(X) = E[I(X)] = E[-\ln(P(X))].$$

$$H(X) = \sum_i P(x_i)\, I(x_i) = -\sum_i P(x_i) \log_b P(x_i)$$

Equation source:
http://en.wikipedia.org/wiki/Entropy_(information_theory)

Conditional Entropy

We want to explore whether the attributes (head and body colors) give us more information about our target variable (yes/no).



We use the **Conditional Entropy**

$$H(Y|X) \equiv \sum_{x \in \mathcal{X}} p(x) H(Y|X=x)$$

Where,

- Y is binary target variable
- X is the attribute
- x is a value of an attribute
- p(x) is the likelihood X=x
- H(Y|X=x) is the entropy of Y where X=x

Example: Conditional Entropy

To compute conditional entropy of an attribute: $H(Y|X) \equiv \sum_{x \in \mathcal{X}} p(x) H(Y|X=x)$

Compute $P(X_i)$, $P(Y|X_i)$, $H(Y|X_i)$ for each attribute value X_i:

Attribute	Value	P(Xi)	P(Y=yes|Xi)	P(Y=no|Xi)	H(Y|Xi)
Body	Red				
Body	Blue				
Head	Red				
Head	Blue				

Now apply the Conditional Entropy formula to the given attributes:

$H(Y|\text{Body}) =$

$H(Y|\text{Head}) =$

Types of Structure

Pairwise (i.e., between 2 variables)

Global (i.e., across a matrix, or within a set of variables)

Supervised (technically includes the above, but with special emphasis on a single target variable)

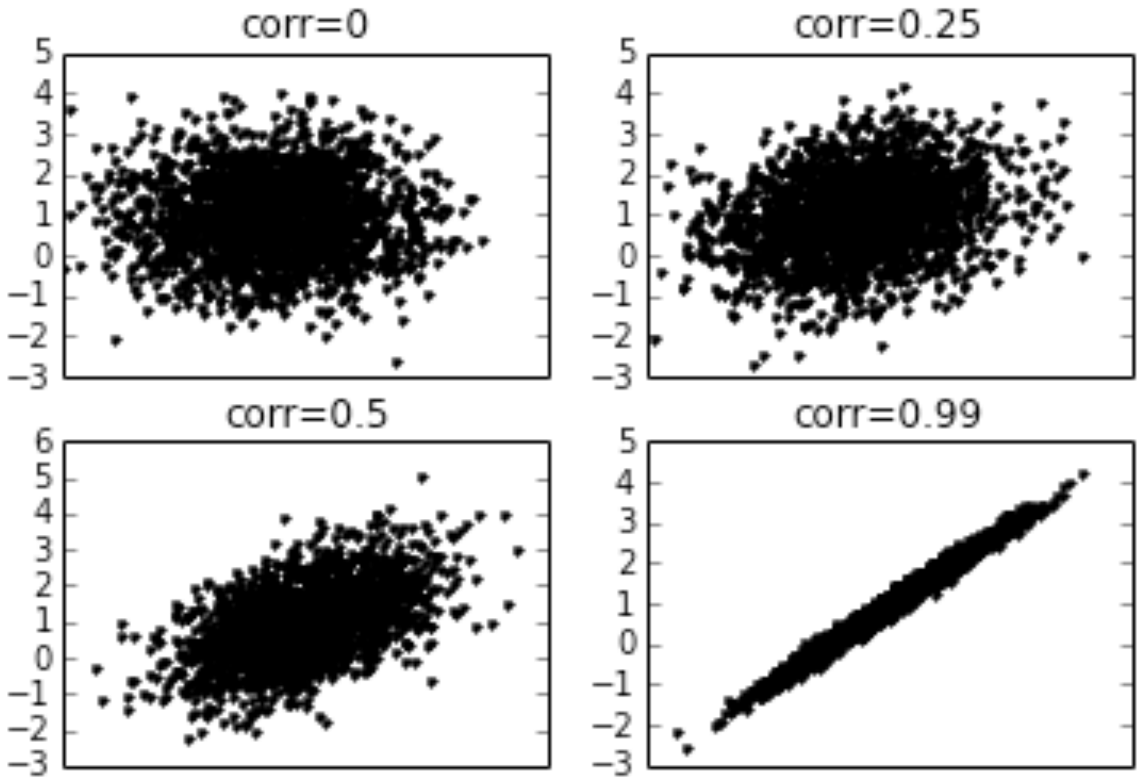
Types of Structure

Pairwise (i.e., between 2 variables)

Covariance and Correlation

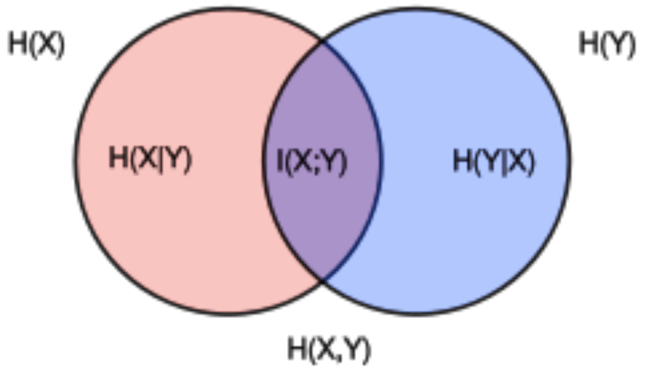
These are probably the most used and thought of metrics when considering pairwise structure. These are statistical quantities and have a fairly intuitive geometric interpretation.

Scatter of Bi-Variate Normally Distributed Variables with various Correlations



Mutual Information

This comes from Information Theory, is used often for feature importance ranking and is related to important aspects of Decision Tree algorithms and Unsupervised learning.



Mutual Information

$$I(X;Y) = \sum_{y \in Y} \sum_{x \in X} p(x,y) \log\left(\frac{p(x,y)}{p(x)\,p(y)}\right),$$

Image and formula source: http://en.wikipedia.org/wiki/Mutual_information

Mutual Information

Let’s break this down

Mutual Information

$$I(X;Y) = \sum_{y \in Y} \sum_{x \in X} p(x,y) \log\left(\frac{p(x,y)}{p(x)\,p(y)}\right),$$

This is in the form of E[F], i.e., E[X]=Σp(F)*F.
In this case, what is X?

Mutual Information

Let’s break this down

Mutual Information

$$I(X;Y) = \sum_{y \in Y} \sum_{x \in X} p(x,y) \log \left(\frac{p(x,y)}{p(x)\,p(y)} \right),$$

F=log(p(x,y)/(p(x)*p(y))….but what is this?

Mutual Information

Let's break this down

Mutual Information

$$I(X;Y) = \sum_{y \in Y} \sum_{x \in X} p(x,y) \log\left(\frac{p(x,y)}{p(x)\,p(y)}\right),$$

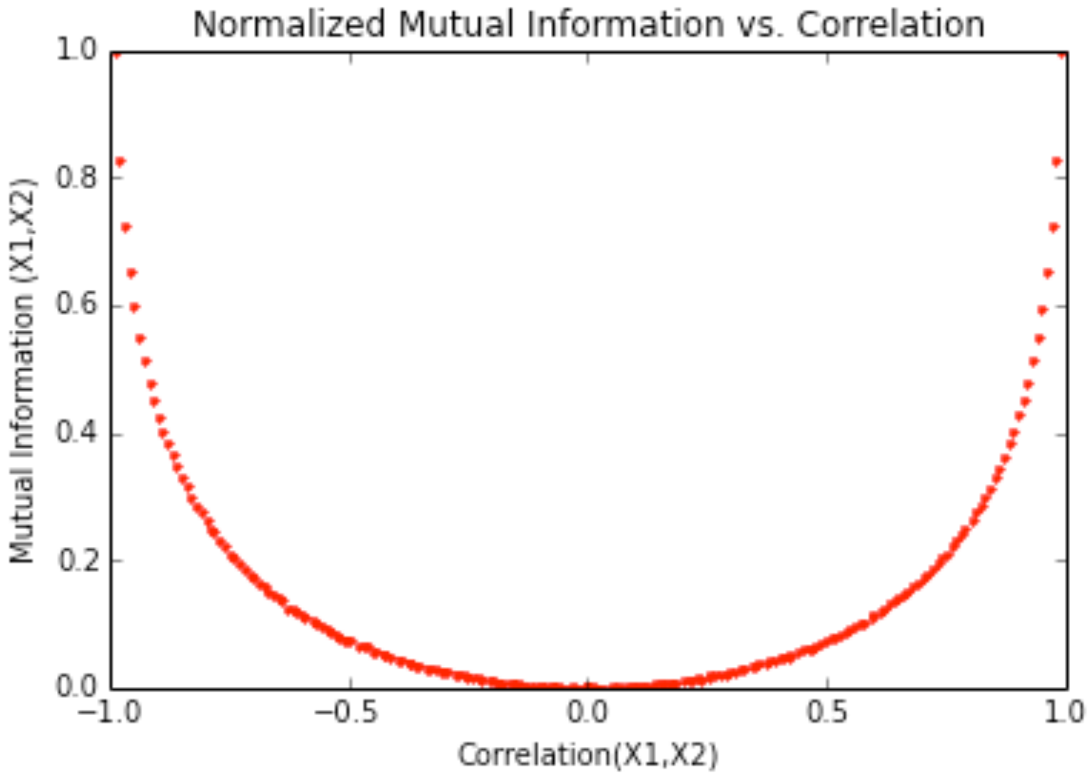
F=log(p(x,y)/(p(x)*p(y))….but what is this?

This is a quantity with the following properties:

- If X,Y are independent, F=0
- If X,Y are completely dependent, F is at a maximum
- F is symmetric (F(x,y)=F(y,x))

MI versus Correlation

Scikit-learn has functions for MI and normalized Mutual Information. We can see that MI and correlation are monotonically related concepts, though MI is strictly positive so does not indicate negative dependencies.



Putting Metrics of Pair-wise Data Structure to Use

- **General understanding of dependencies in data**
- **Validating assumptions for statistical modeling**
- **Feature ranking and selection**
- **Decision Tree Algorithms**

Types of Structure

<u>Global</u> (i.e., across a matrix, or within a set of variables)

Data is Multivariate

- We usually want to go beyond pairwise similarity and understand how much “information” is actually embedded in a matrix.
- We also might want to know what groups of features are related.
- In an Nx2 matrix, this problem reduces to the pairwise methods just discussed.

Singular Value Decomposition

Although this is not a linear algebra course, this equation happens so often in data analysis that it is worth learning (again).

$$\underset{NxM}{\mathbf{X}} = \underset{NxM}{\mathbf{U}} \; \underset{MxM}{\mathbf{\Sigma}} \; \underset{(MxM)^T}{\mathbf{V}^{\mathbf{T}}}$$

We will not dive into all of the theoretical aspects of SVD and related topics. Our goal here is to understand it intuitively and be able to use it as a tool for solving other common Data Science problems.

Singular Value Decomposition

Lets go through this piece by piece:

$\mathbf{U}$

U holds the left singular vectors. Each row contains k elements that correspond to the latent factors of each row of X. U is orthonormal.

$\mathbf{\Sigma}$

Σ is a diagonal matrix that holds the singular values (in descending order). The singular values are essentially weights that determine how much that latent factor contributes to the matrix.

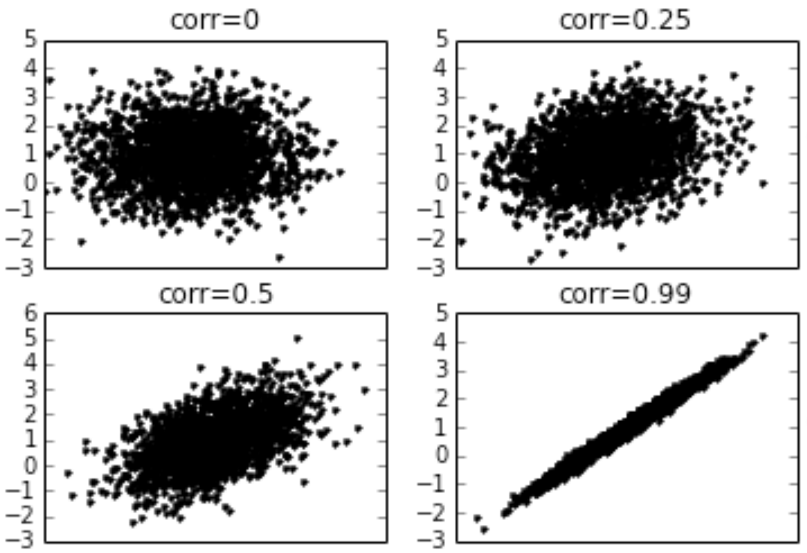
$\mathbf{V^T}$

V holds the right singular vectors. Each row contains k elements that correspond to the latent factors of each column of X. V is orthonormal.

Singular Value Decomposition

How does this relate to structure?

Lets recall these scatter plots. Each plot can be expressed as an Nx2 Matrix. The SVD is a tool that can help us understand how much information or structure is actually in the matrix.

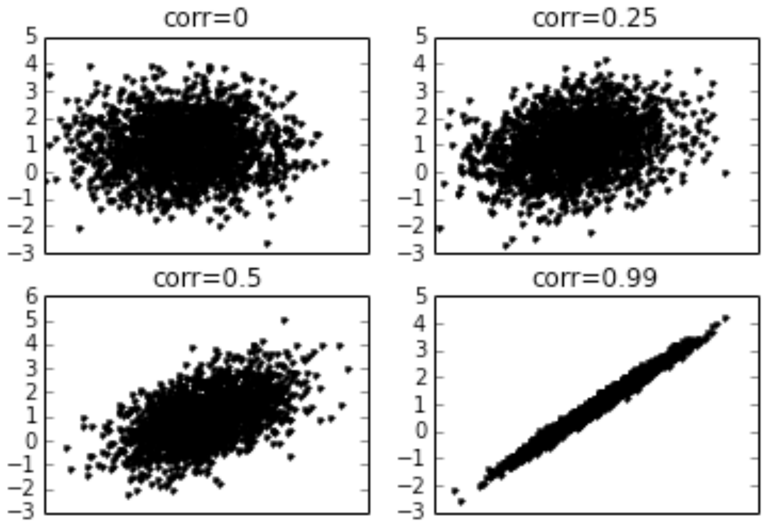


More independence of columns = more information or degrees of freedom.

Less independence of columns = less information or degrees of freedom.

Singular value decomposition

Using Scipy we can easily compute the SVD.:
U,Sig,Vt = scipy.linalg.svd(matrix)



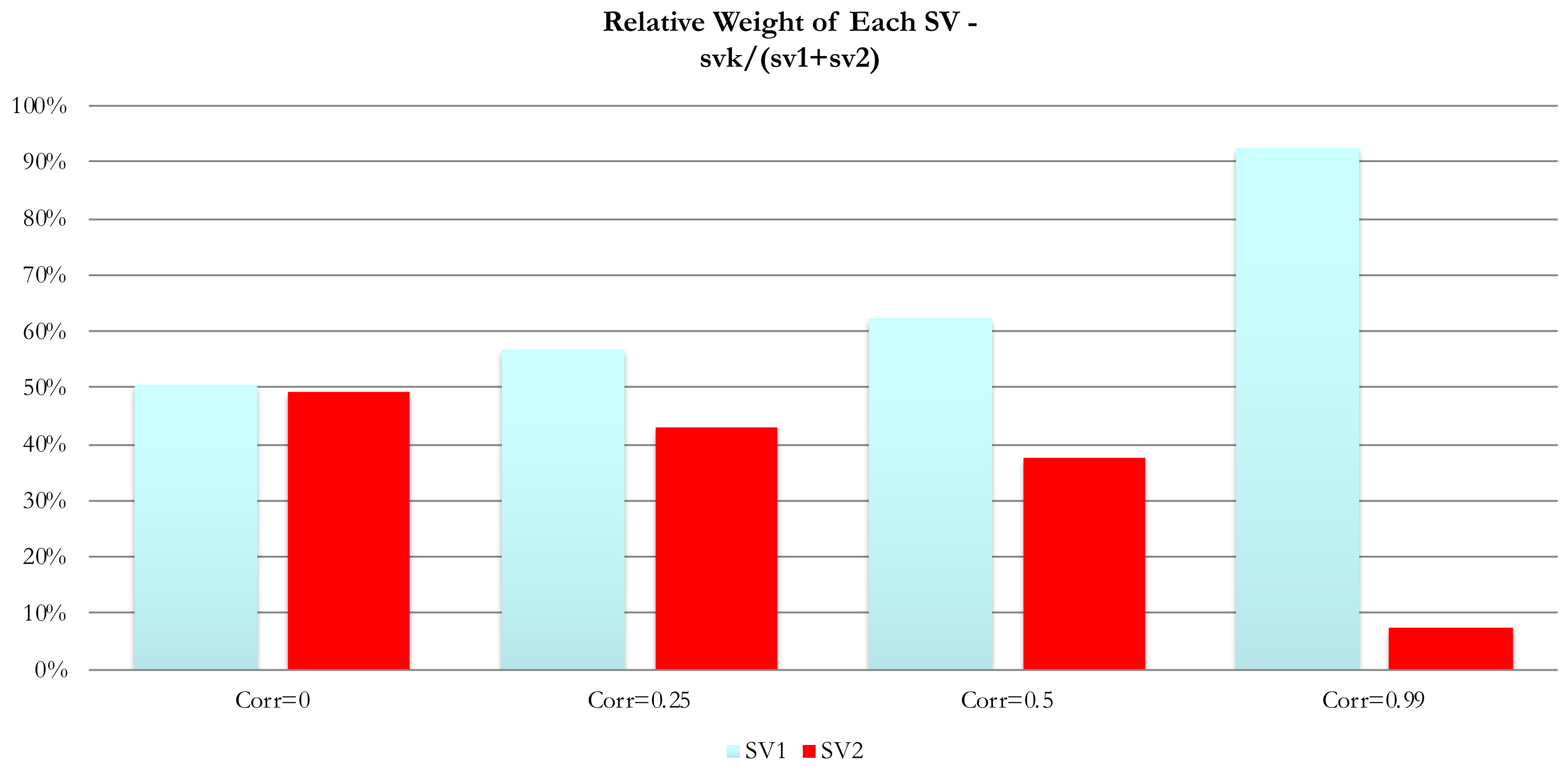
$$\Sigma = \begin{bmatrix} 45 & \\ & 44 \end{bmatrix} \quad \Sigma = \begin{bmatrix} 50 & \\ & 38 \end{bmatrix}$$

$$\Sigma = \begin{bmatrix} 53 & \\ & 32 \end{bmatrix} \quad \Sigma = \begin{bmatrix} 62 & \\ & 5 \end{bmatrix}$$

The magnitude of the singular-values are generally determined by the magnitude of the values in X. The relative difference between singular values is a function of the level of independence of the columns.

Singular Value Decomposition

We can see that less independence of the columns leads to singular values with more skew from each other.

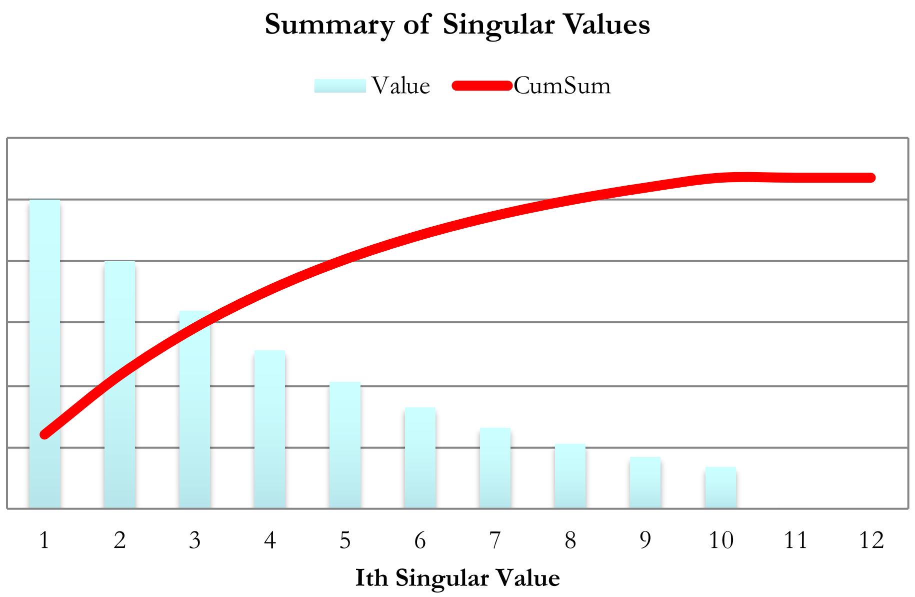


Singular Value Decomposition

This idea generalizes to more dimensions, which is where the SVD is extremely useful.

$$\Sigma = \left(\begin{array}{cccccccccccc}
1 & & & & & & & & & & & \\
 & 0.8 & & & & & & & & & & \\
 & & 0.64 & & & & & & & & & \\
 & & & 0.512 & & & & & & & & \\
 & & & & 0.4096 & & & & & & & \\
 & & & & & 0.32768 & & & & & & \\
 & & & & & & 0.262144 & & & & & \\
 & & & & & & & 0.2097152 & & & & \\
 & & & & & & & & 0.16777216 & & & \\
 & & & & & & & & & 0.134217728 & & \\
 & & & & & & & & & & 0 & \\
 & & & & & & & & & & & 0
\end{array}\right)$$

The skew of the singular values, and the shape of the cumulative sum curve can give us a sense of the degree of independence in a many dimensional matrix.



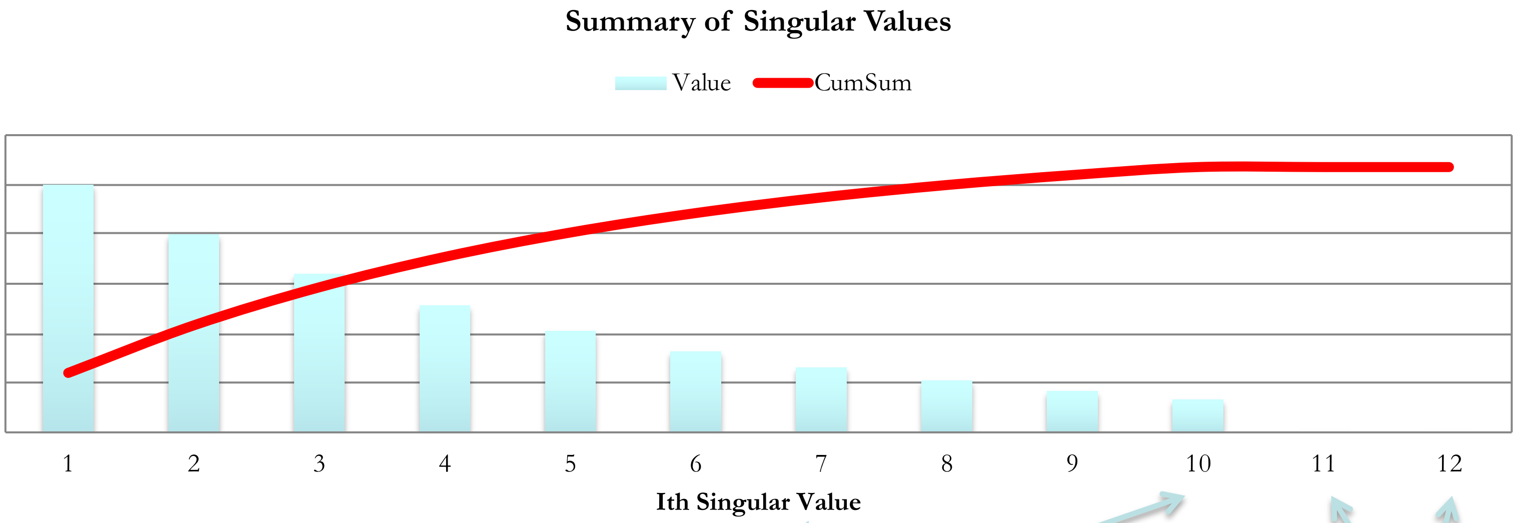
An Incredibly Useful Application of SVD

One of the most powerful applications of the SVD is creating a low rank approximation of a data matrix. Many of the following methods are based on using the SVD to get a low rank approximation.

- **Data Compression**
- **Dimensionality Reduction**
- **Recommender Systems**
- **Clustering in High Dimensions**

The Low Rank Approximation

The rank of a matrix is the size of the largest number of independent columns of a matrix. The rank can be found by counting the number of singular values >0.



These SVs are greater than 0, but they are small. The add very little information to this matrix, and could even represent noise. What would happen if we set them to zero?

These SVs are zero so the rank of this matrix is 10

The Low Rank Approximation

We can build a matrix X_k that approximates our original matrix by doing the following:

1. Compute the SVD of X
2. Truncate U, Σ and V to take only the k highest columns & singular values
3. Multiply back together to get X_k

Low Rank Approximation – K<<M

$$\mathbf{X_k = U_k \Sigma_k V_k{}^T}$$

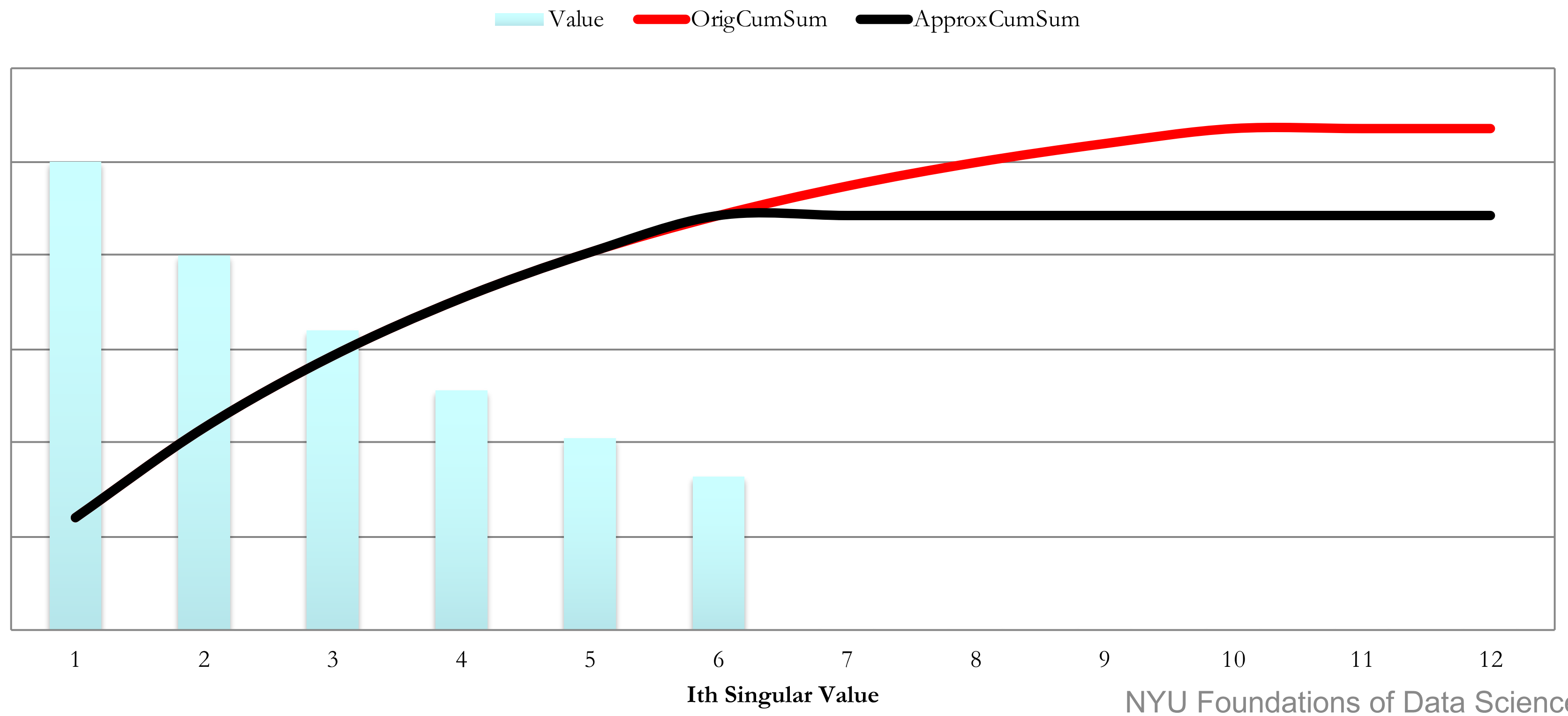
NxM *NxK* *KxK* *(MxK)^T*

The Low Rank Approximation

Our original matrix had 12 columns with a rank of 10. In our approximation, we decided to use k=6. The cumulative sum of singular values is related to the amount of information contained in the matrix. By using k=6, we lost some information, but not half.

What do we gain with the low rank approximation?

What do we lose?



One Benefit of Rank-k SVD

Some facts

- A floating point number uses 8 bytes of memory
- An MxN matrix of floating point numbers uses ***8*M*N*** bytes of memory.

Instead of storing X, lets use X_k, a low rank approximation

$\mathbf{U_k}$	…this is 8*N*K bytes	When separated…
$\mathbf{\Sigma_k}$	…this is 8*1*K bytes $\Rightarrow$	***8*K*(N+M+1)***
$\mathbf{V_k^T}$	…this is 8*K*K bytes	bytes

If you choose K<<M, you could save a ton of space…

i.e. k=100,N=1MM,M=10k, the full X is 80 GB, whereas storing the decomposed components of X_k would only be 0.8 GB!

Major Tasks in Data Preprocessing

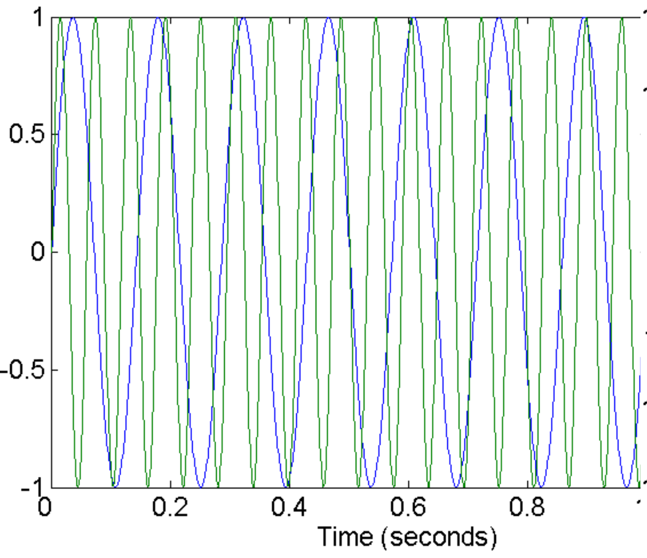
- **Data cleaning**
  - Fill in missing values, smooth noisy data, identify or remove outliers, and resolve inconsistencies
- **Data integration**
  - Integration of multiple databases, data cubes, or files
- **Data reduction**
  - Dimensionality reduction
  - Numerosity reduction
  - Data compression
- **Data transformation and data discretization**
  - Normalization
  - Concept hierarchy generation

Data Reduction Strategies

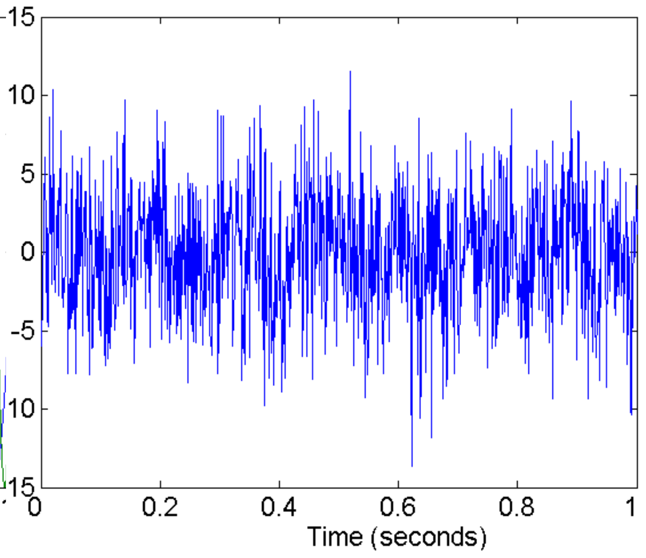
- **Data reduction**: Obtain a reduced representation of the data set that is much smaller in volume but yet produces the same (or almost the same) analytical results
- Why data reduction? — A database/data warehouse may store terabytes of data. Complex data analysis may take a very long time to run on the complete data set. Also we might want to understand underlying similarities.
- Data reduction strategies
  - Dimensionality reduction, e.g., remove unimportant attributes
    - Wavelet transforms
    - Principal Components Analysis (PCA)
    - Feature subset selection, feature creation
  - Numerosity reduction (some simply call it: Data Reduction)
    - Regression and Log-Linear Models
    - Histograms, clustering, sampling
    - Data cube aggregation
  - Data compression

Mapping Data to a New Space

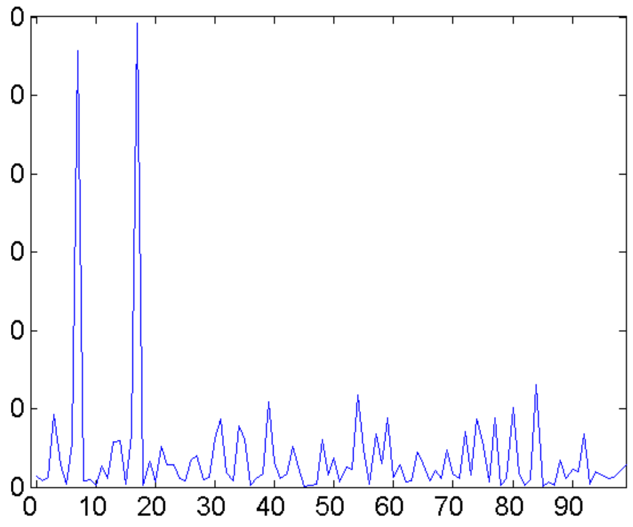
- Fourier transform
- Wavelet transform



Two Sine Waves



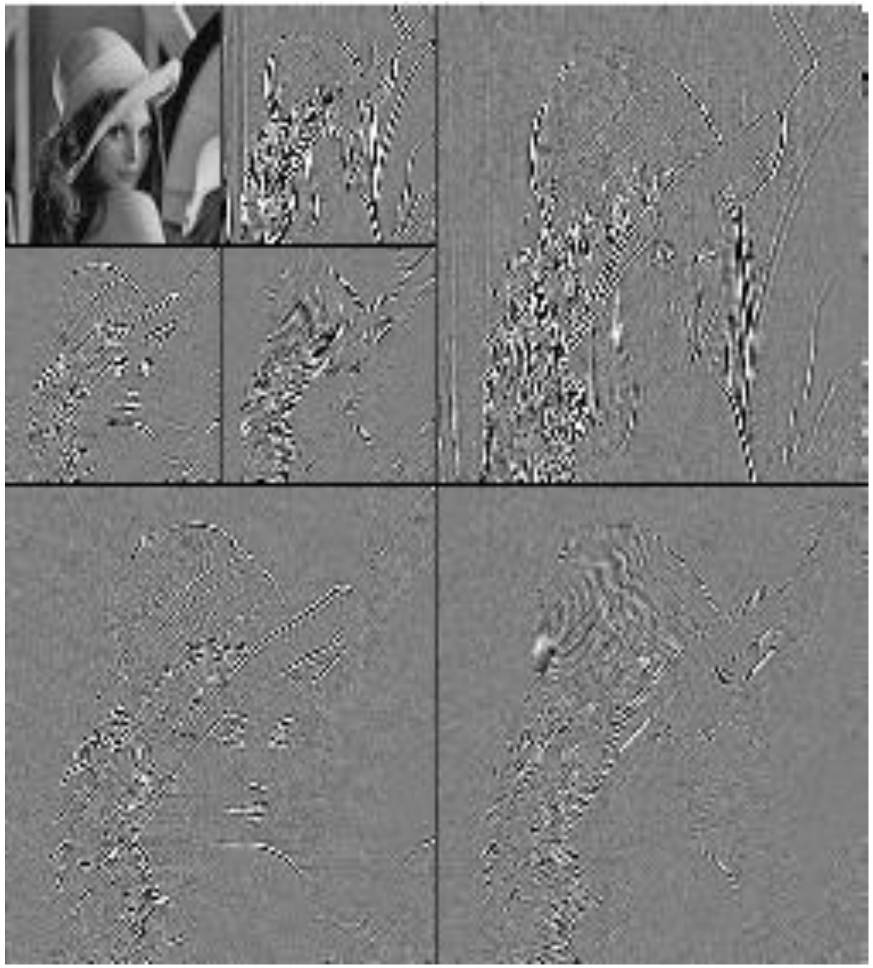
Two Sine Waves + Noise



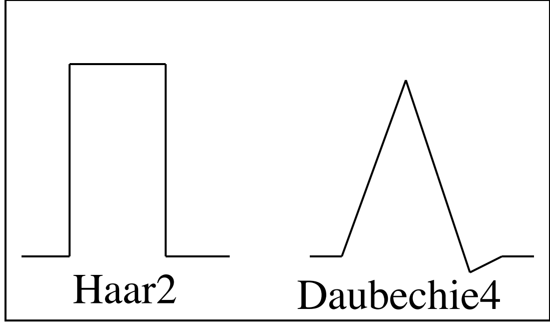
Frequency

What Is Wavelet Transform?

- Decomposes a signal into different frequency subbands
- Data are transformed to preserve relative distance between objects at different levels of resolution
- Allow natural clusters to become more distinguishable
- Used for image compression



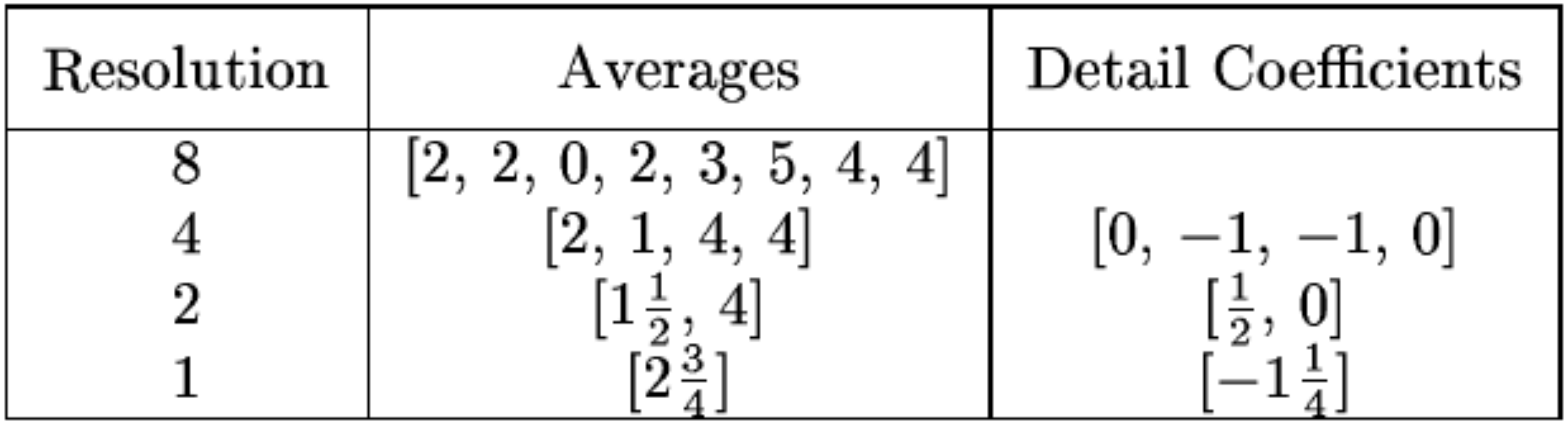
Wavelet Transformation



- Discrete wavelet transform (DWT) for linear signal processing, multi-resolution analysis
- Compressed approximation: store only a small fraction of the strongest of the wavelet coefficients
- Similar to discrete Fourier transform (DFT), but better lossy compression, localized in space
- Method:
  - Length, L, must be an integer power of 2 (padding with 0's, when necessary)
  - Each transform has 2 functions: smoothing, difference
  - Applies to pairs of data, resulting in two sets of data of length L/2
  - Applies two functions recursively, until reaches the desired length

Wavelet Decomposition

- Wavelets: A math tool for space-efficient hierarchical decomposition of functions
- S = [2, 2, 0, 2, 3, 5, 4, 4] can be transformed to $S_\wedge$ = [$2^3/_4$, $-1^1/_4$, $^1/_2$, 0, 0, -1, -1, 0]
- Compression:



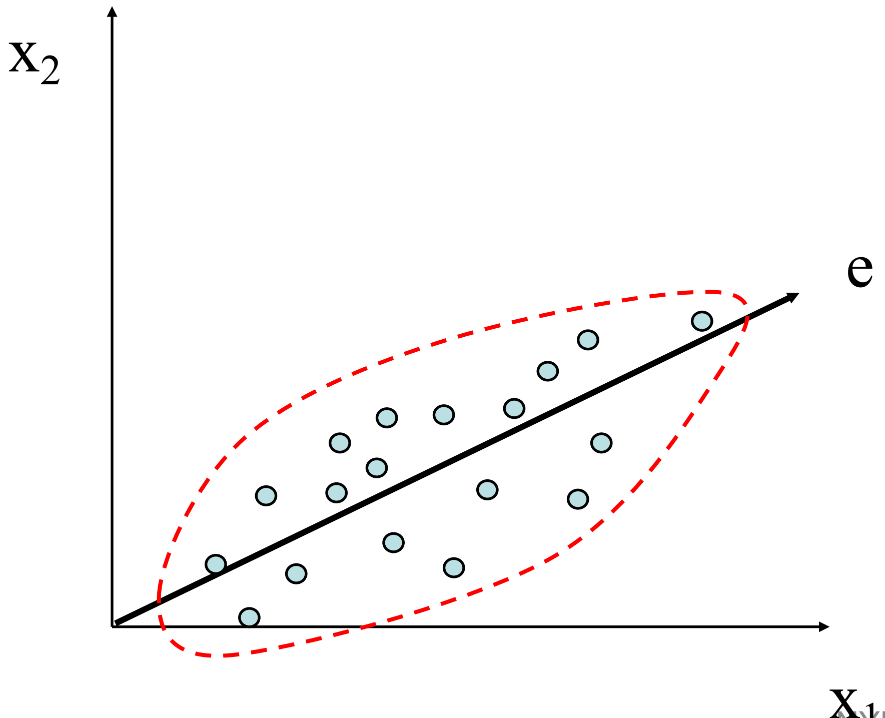
Resolution	Averages	Detail Coefficients
8	$[2, 2, 0, 2, 3, 5, 4, 4]$	
4	$[2, 1, 4, 4]$	$[0, -1, -1, 0]$
2	$[1\frac{1}{2}, 4]$	$[\frac{1}{2}, 0]$
1	$[2\frac{3}{4}]$	$[-1\frac{1}{4}]$

Why Wavelet Transform?

- Use hat-shape filters
  - Emphasize region where points cluster
  - Suppress weaker information in their boundaries
- Effective removal of outliers
  - Insensitive to noise, insensitive to input order
- Multi-resolution
  - Detect arbitrary shaped clusters at different scales
- Efficient
  - Complexity O(N)
- Only applicable to low dimensional data

Principal Component Analysis (PCA)

- Find a projection that captures the largest amount of variation in data
- The original data are projected onto a much smaller space, resulting in dimensionality reduction. We find the eigenvectors of the covariance matrix, and these eigenvectors define the new space



Principal Component Analysis (Steps)

- Given N data vectors from n-dimensions, find $k \leq n$ orthogonal vectors (*principal components*) that can be best used to represent data
  - Normalize input data: Each attribute falls within the same range
  - Compute k orthonormal (unit) vectors, i.e., *principal components*
  - Each input data (vector) is a linear combination of the k principal component vectors
  - The principal components are sorted in order of decreasing "significance" or strength
  - Since the components are sorted, the size of the data can be reduced by eliminating the *weak components*, i.e., those with low variance (i.e., using the strongest principal components, it is possible to reconstruct a good approximation of the original data)
- Works for numeric data only

Summary

- **Data quality**: accuracy, completeness, consistency, timeliness, believability, interpretability
- **Data cleaning**: e.g. missing/noisy values, outliers
- **Data integration** from multiple sources:
  - Entity identification problem
  - Remove redundancies
  - Detect inconsistencies
- **Data reduction**
  - Dimensionality reduction
  - Numerosity reduction
  - Data compression
- **Data transformation and data discretization -> more next time**
  - Normalization
  - Concept hierarchy generation