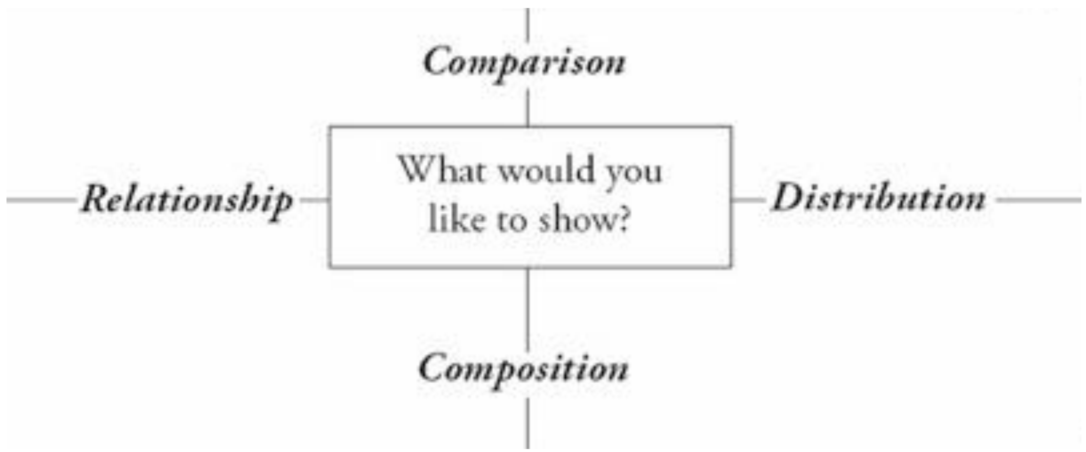
Data Mining & Machine Learning

CS37300
Purdue University

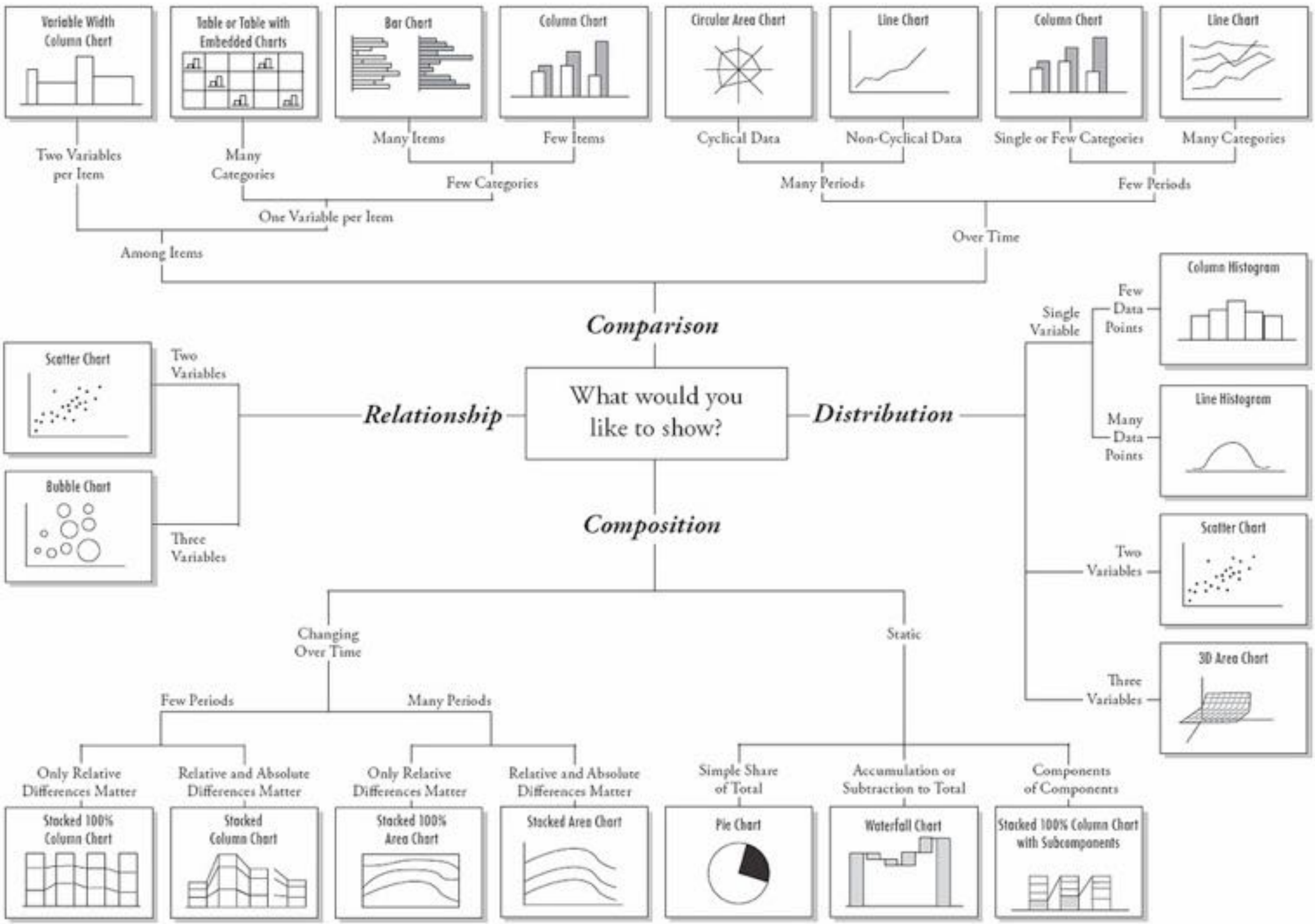
September 11, 2017

Data exploration and visualization



http://extremepresentation.typepad.com/blog/2006/09/choosing_a_good.html

Chart Suggestions—A Thought-Starter



http://extremepresentation.typepad.com/blog/2006/09/choosing_a_good.html

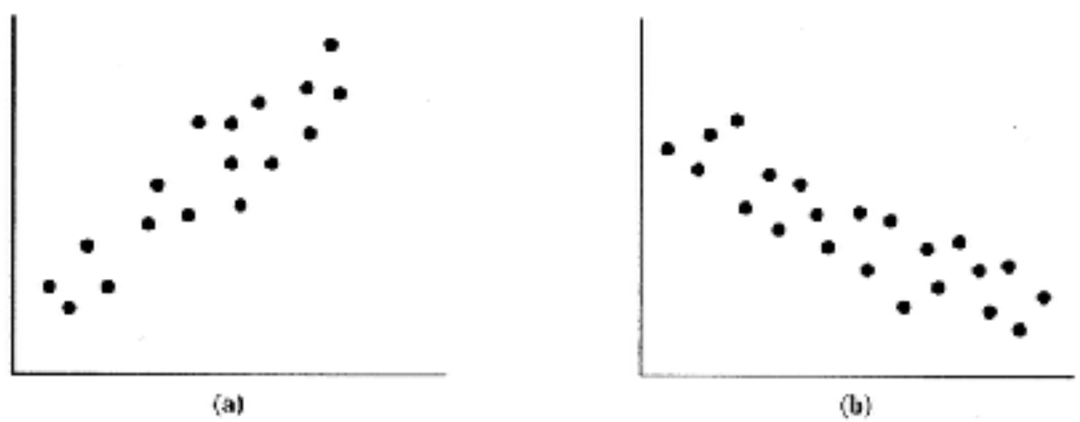
Covariance and correlation

Covariance

- Measures how variables X and Y vary together

$$COV(x,y) = \frac{1}{n}\sum_{i=1}^{n}(x(i)-\bar{x})(y(i)-\bar{y})$$

  - Positive if large values of X are associated with large values of Y
  - Negative if large values of X are associated with small values of Y



Measures **linear** relationship

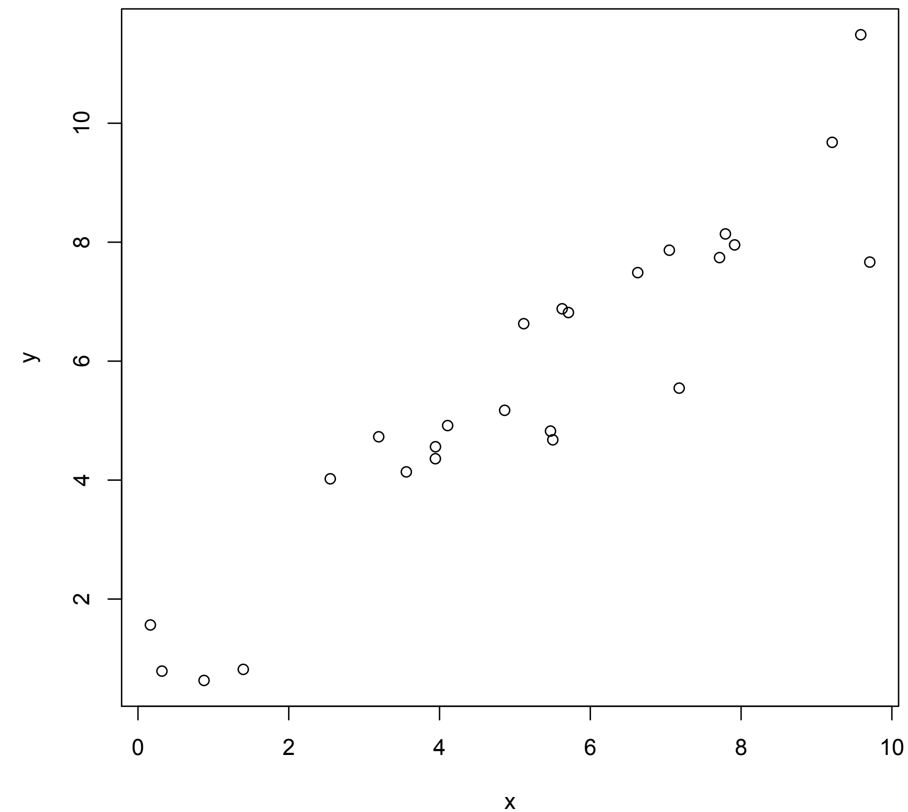
- Covariance matrix (Σ)
  - Symmetric matrix of covariances for p variables

Example

```
import numpy as np

x = np.random.uniform(0,15,10)
y = x + np.random.uniform(0,1,10)

z = np.vstack((x, y))

###compute covariance
print(np.cov(z))
```

```
[[ 21.5552459 ,  21.16218373],
 [ 21.16218373,  20.83754179]]
```

```
> z
```



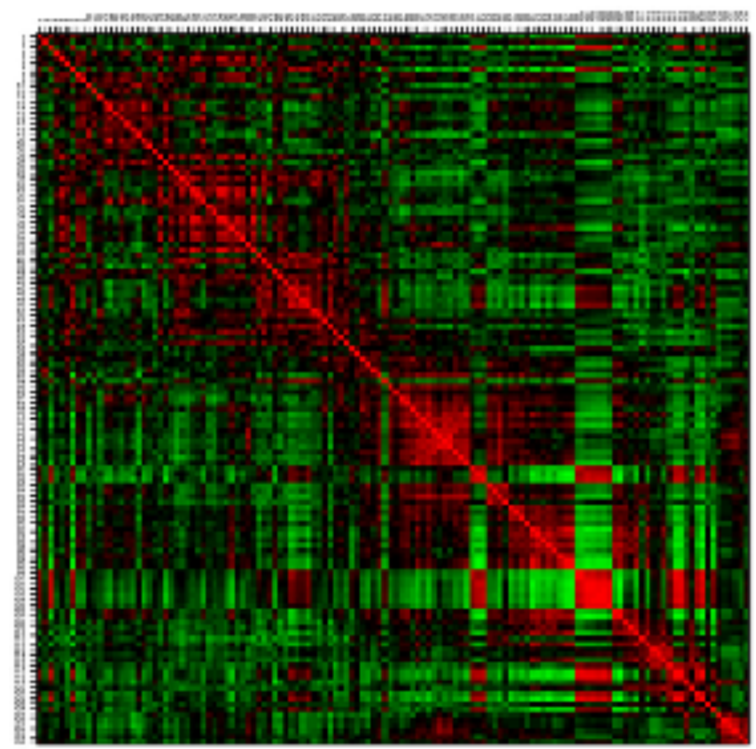

```
[25,] 1.3967384  0.8176381
```

Correlation coefficient

- Covariance depends on ranges of X and Y
- Correlation standardizes covariance by dividing through standard deviations

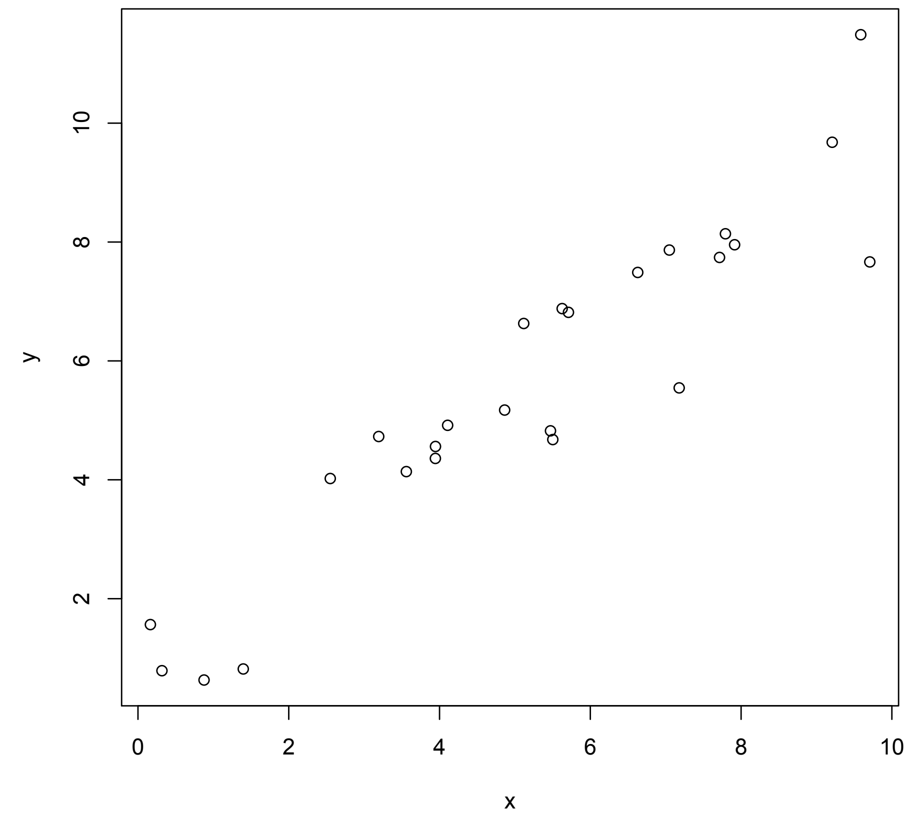
$$\rho(x,y) = \frac{\frac{1}{n}\sum_{i=1}^{n}(x(i)-\bar{x})(y(i)-\bar{y})}{\sigma_x \sigma_y}$$

- Correlation matrix
  - Symmetric matrix of correlations for p variables
  - What values are on the diagonal?



Example (cont)

```
>###compute covariance
> print(np.cov(z))
[[ 21.5552459   21.16218373]
 [ 21.16218373  20.83754179]]

>###compute correlation
> print(np.corrcoef(z))
[[ 1.          0.99852915]
 [ 0.99852915  1.        ]]
```

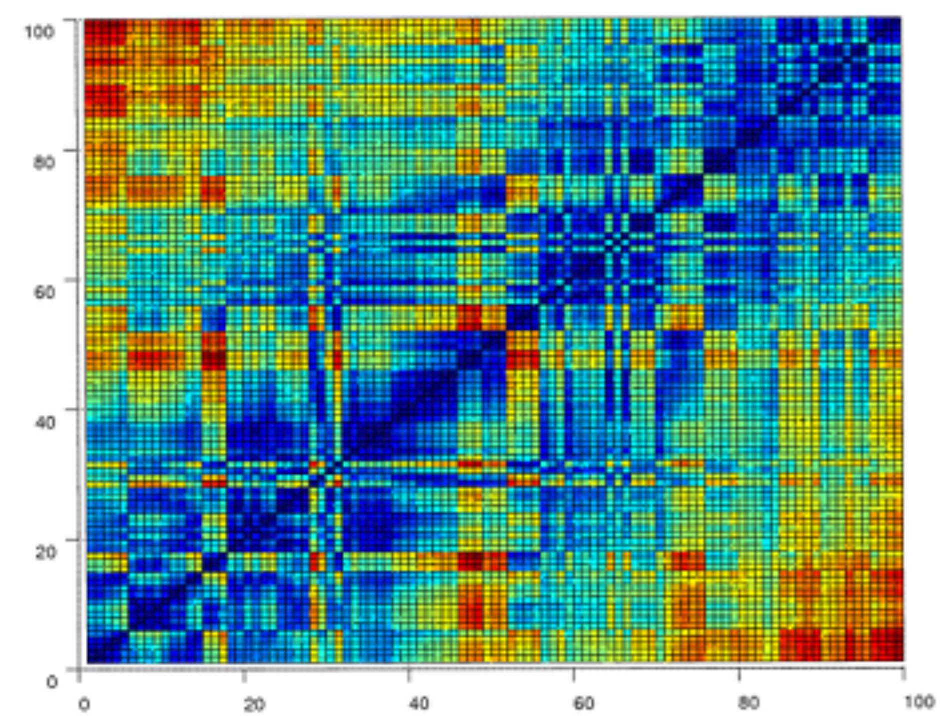


Distance measures

Distance measures

- If data objects have the same fixed set of numeric attributes, then the data objects can be thought of as points in a multi-dimensional space, where each dimension represents a distinct attribute

- Many data mining techniques then use similarity/dissimilarity measures to characterize relationships between the instances, e.g.,
    - Nearest-neighbor classification
    - Cluster analysis



- **Proximity**: general term to indicate similarity and dissimilarity
- **Distance**: dissimilarity only

Metric properties

- A **metric** *d(i,j)* is a dissimilarity measure that satisfies the following properties:
    - $d(i,j) \geq 0$ for all i,j and $d(i,j)=0$ iff $i=j$ — **Positivity**
    - $d(i,j) = d(j,i)$ for all i,j — **Symmetry**
    - $d(i,j) \leq d(i,k)+d(k,j)$ for all i,j,k — **Triangle inequality**

Distance metrics

- Manhattan distance (L1)

$$d_M(x,y) = \sum_{i=1}^{p} |x_i - y_i|$$

- Euclidean distance (L2)

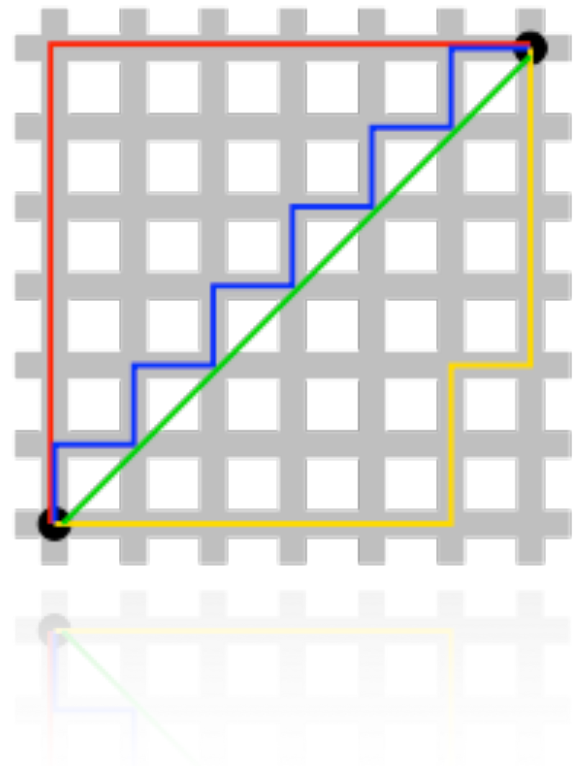
$$d_E(x,y) = \sqrt{\sum_{i=1}^{p} (x_i - y_i)^2}$$

  - Most common metric
  - Assumes variables are commensurate

- **Weighted** Euclidean distance

$$d_{WE}(x,y) = \sqrt{\sum_{i=1}^{p} w_i (x_i - y_i)^2}$$

  - Can weight variables by relative importance



Standardization

- Normalization
  - Removes effect of scale
  - Divide each variable by its standard deviation
  - Weights all variables equally

$$x'_k = \frac{x_k - \bar{x}_k}{\hat{\sigma}_k}$$

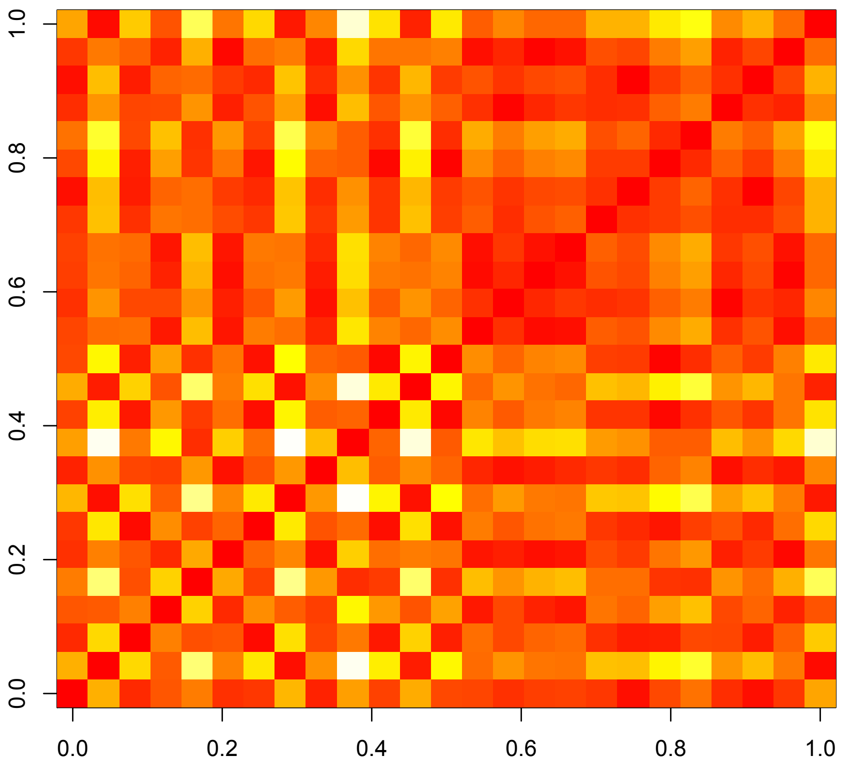
subtract mean

divide by stdev

$$d'_E(x, y) = \sqrt{\sum_{i=1}^{p} (x'_i - y'_i)^2}$$

Example (cont)

```
print(t)
           1          2          3          4          5
1  0.0000000  7.3455738  1.7390998  3.6589298  5.1026250  1.98
2  7.3455738  0.0000000  8.9503324  3.7808055 12.2991538  5.36
3  1.7390998  8.9503324  0.0000000  5.3528744  3.3837272  3.60
4  3.6589298  3.7808055  5.3528744  0.0000000  8.7366015  1.79
5  5.1026250 12.2991538  3.3837272  8.7366015  0.0000000  6.97
6  1.9881553  5.3667753  3.6016281  1.7959366  6.9781055  0.00
7  2.2926969  9.5085597  0.5634703  5.9161521  2.8205210  4.16
8  7.5596343  0.5799917  9.2042741  3.9290276 12.5727002  5.60
9  1.4782647  6.0434830  2.9115168  2.5838461  6.2590354  0.79
10 6.6018916 13.9187745  4.9744346 10.2595056  1.8483139  8.55
```



Correlation among variables

- Variables contribute independently to additive measure of distance
- May not be appropriate if variables are highly correlated
- Can standardize variables in a way that accounts for covariance

Diameter,
Height$_1$,
Height$_2$,
...,
Height$_{100}$

Mahalanobis distance

$$d_{MH}(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x} - \mathbf{y})^T \Sigma^{-1} (\mathbf{x} - \mathbf{y})}$$

pxp covariance matrix

- Automatically accounts for scaling
- Corrects for correlation between attributes
- Tradeoff:
  - Covariance matrix can be hard to estimate accurately
  - Memory and time complexity is quadratic rather than linear

Distance measures for binary data

- d(x,y) when items x and y are p-dimensional binary vectors
- Let n_{11} be the number of attributes where both items have value 1, etc.

$$n_{11} = \sum_i^p \mathbb{I}(x_i + y_i = 2)$$

	y=1	y=0
x=1	n_{11}	n_{01}
x=0	n_{10}	n_{00}

- Matching coefficient
  - Hamming distance normalized by number of bits

$$d_{MC}(x,y) = \frac{n_{11} + n_{00}}{n_{11} + n_{00} + n_{10} + n_{01}}$$

- Jaccard coefficient
  - If we don't care about matches on zeros

$$d_{JC}(x,y) = \frac{n_{11}}{n_{11} + n_{00} + n_{10} + n_{01}}$$

Dimensionality reduction methods

Dimensionality reduction

- Identify and describe the “dimensions” that underlie the data
    - May be more fundamental than those directly measured but hidden to the user
- Reduce dimensionality of modeling problem
    - Benefit is simplification, it reduces the number of variables you have to deal with in modeling
- Can identify set of variables with similar behavior

Dimensionality reduction methods

- Principal component analysis (PCA)
  - Linear transformation, minimize unexplained variance
- Factor analysis
  - Linear combination of small number of **latent** variables
- Multidimensional scaling (MDS)
  - Project into low-dimensional subspace while preserving distance between points (can be non-linear)

Principal component analysis (PCA)

- High-level approach, given data matrix **D** with **p** dimensions:
  - Preprocess **D** so that the mean of each attribute is 0, call this matrix **X**
  - Compute pxp covariance matrix: $\Sigma = X^T X$
  - Compute eigenvectors/eigenvalues of covariance matrix:

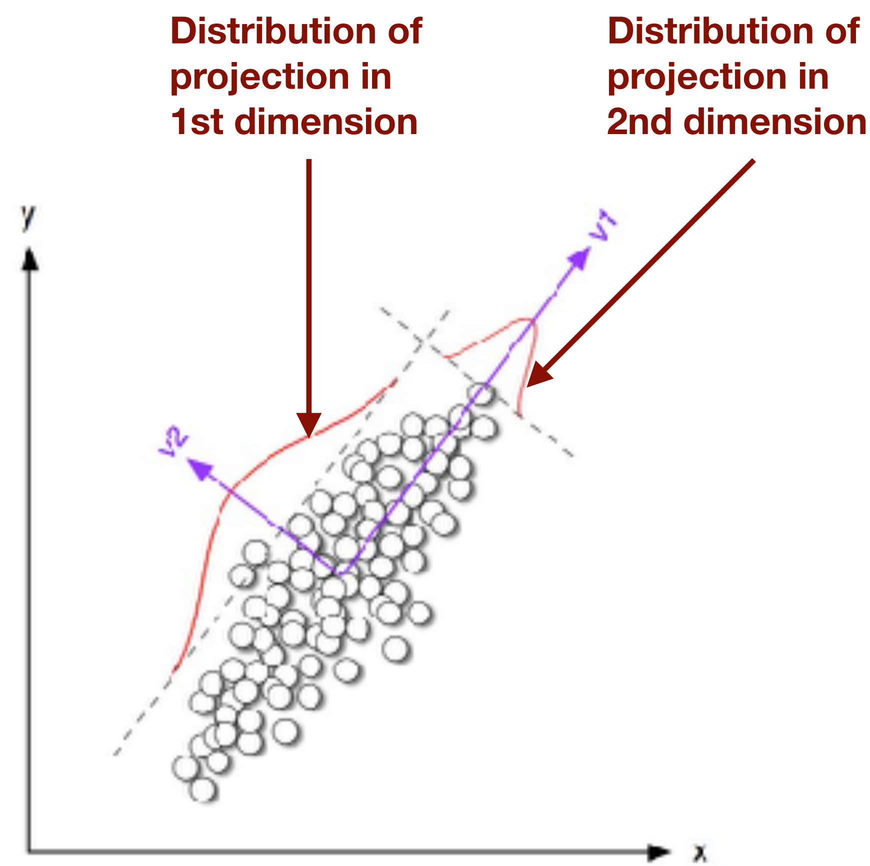
$$\mathbf{A}\Sigma\mathbf{A}^{-1} = \Lambda$$
$$(\Sigma - \lambda\mathbf{I})\mathbf{a} = 0$$

$\mathbf{A}$: matrix of eigenvectors
Λ : diagonal matrix of eigenvalues
$\mathbf{a}$: 1st principal component, eigenvector assoc. with largest eigenvalue (λ)

  - Eigenvectors $\mathbf{A}$ are the **principal component** vectors, where each $\mathbf{a}$ is a px1 column vector of projection weights

Learning PCA models

- **Model space**: set of p **orthonormal** basis vectors (if data is p-dimensional)
  - All basis vectors have norm of 1
  - Any pair of basis vectors have dot-product of 0
  - E.g., any orthogonal set of v_1 and v_2
- **Scoring function**:
  - 1st basis (eg. v_1) ***maximizes*** variance of projected data
  - 2nd basis (eg. v_2) again ***maximizes*** variance of projected data, but has to be orthogonal to previous bases, ...



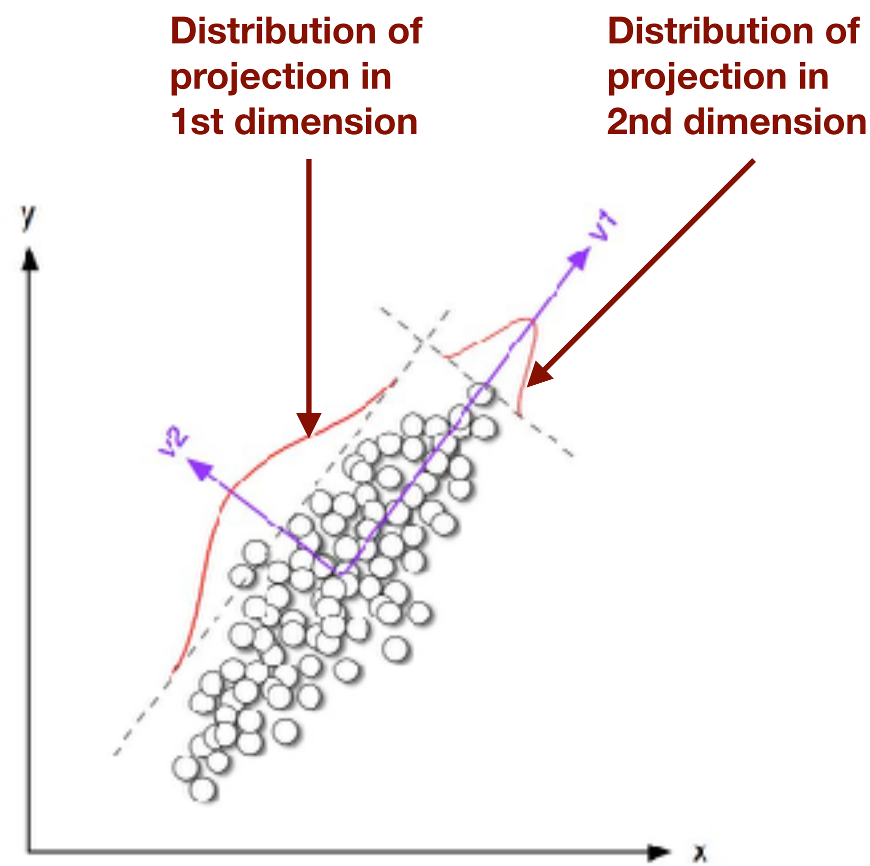
Learning PCA models

- Variance of dimension **i** is λ_i
- Sum of eigenvalues is equal to the sum of the variances of the original attributes

$$\sum_{j=1}^{p} \sigma_j^2 = \sum_{j=1}^{p} \lambda_j$$

- New dimensions are orthogonal, thus transformed features have 0 covariance
- **Search**: Solving eigensystem corresponds to finding the orthonormal basis that maximize variance

$$\mathbf{A}\Sigma\mathbf{A}^{-1} = \Lambda$$



Applying PCA

- New data vectors are formed by projecting the data onto the first few principal components (i.e., top k eigenvectors)

$$\mathbf{x} = [x_1, x_2, \ldots, x_p] \quad \text{(original instance)}$$

$$\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_p] \quad \text{(principal components)}$$

$$x'_1 = \mathbf{a}_1 \mathbf{x} = \sum_{j=1}^{p} a_{1j} x_j$$

$$\ldots$$

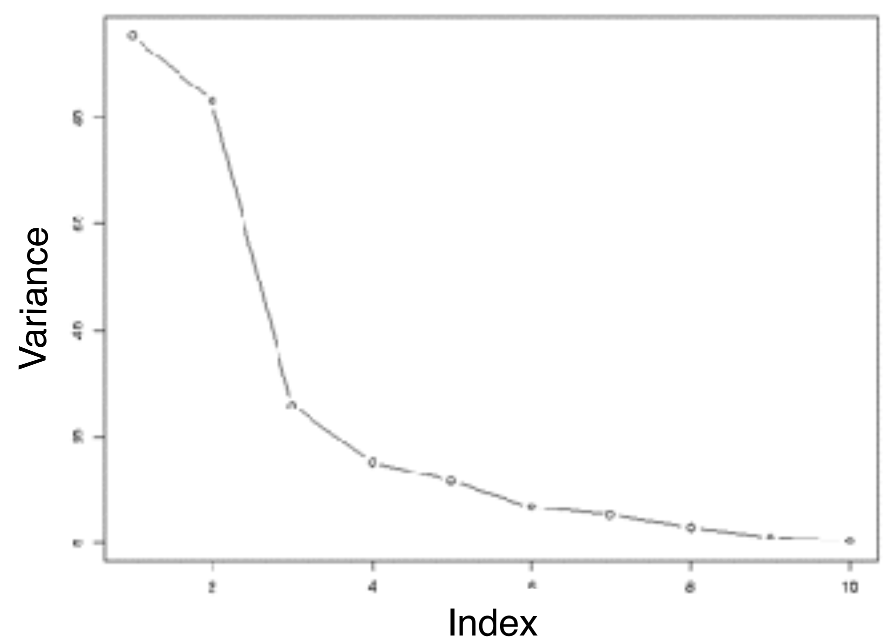
$$x'_m = \mathbf{a}_m \mathbf{x} = \sum_{j=1}^{p} a_{mj} x_j \quad for\ m < p$$

If ***m=p*** then data is transformed
If ***m<p*** then transformation is lossy and dimensionality is reduced

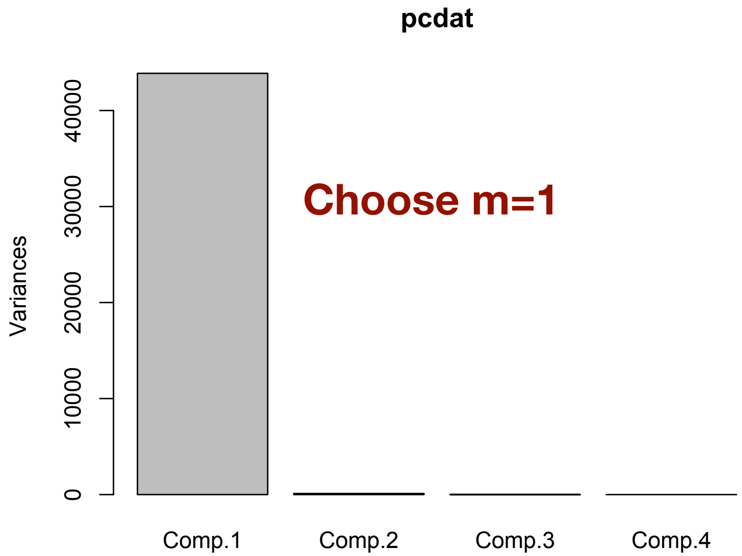
$$\mathbf{x}' = [x'_1, x'_2, \ldots, x'_m] \quad \text{(transformed instance)}$$

Applying PCA (cont’)

- Goal: Find a new (smaller) set of dimensions that captures most of the variability of the data
- Use **scree plot** to choose number of dimensions
    - Choose $m < p$ so projected data captures much of the variance of original data



PCA example on Iris data



Choose m=1

```
> x <- scale(as.matrix(d[,1:4]),scale=FALSE)
> sigma <- t(x)%*% x
> sigma
          V1       V2       V3        V4
V1 102.16833  -5.8510 189.7787  77.01867
V2  -5.85100  28.0126 -47.9352 -17.57920
V3 189.77867 -47.9352 463.8637 193.16173
V4  77.01867 -17.5792 193.1617  86.77973

> pcdat <- princomp(d[,1:4])
> summary(pcdat)
Importance of components:
                          Comp.1     Comp.2     Comp.3      Comp.4
Standard deviation     2.0485788 0.49053911 0.27928554 0.153379074
Proportion of Variance 0.9246162 0.05301557 0.01718514 0.005183085
Cumulative Proportion  0.9246162 0.97763178 0.99481691 1.000000000
> plot(pcdat)
> loadings(pcdat)

Loadings:
   Comp.1 Comp.2 Comp.3 Comp.4
V1  0.362 -0.657 -0.581  0.317
V2        -0.730  0.596 -0.324
V3  0.857  0.176        -0.480
V4  0.359         0.549  0.751

               Comp.1 Comp.2 Comp.3 Comp.4
SS loadings      1.00   1.00   1.00   1.00
Proportion Var   0.25   0.25   0.25   0.25
Cumulative Var   0.25   0.50   0.75   1.00
```

First component explains 92% of data variance

PCA example on Iris data

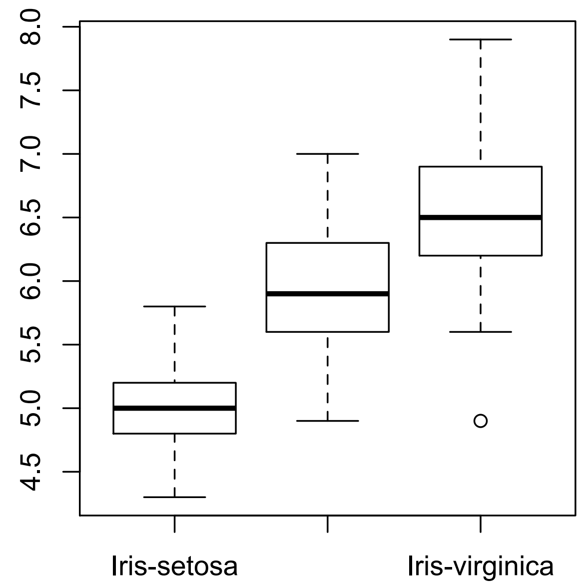
m=1, transform data to one dimension

$$\mathbf{x} = [x_1, x_2, \ldots, x_p] \quad \text{(original instance)}$$

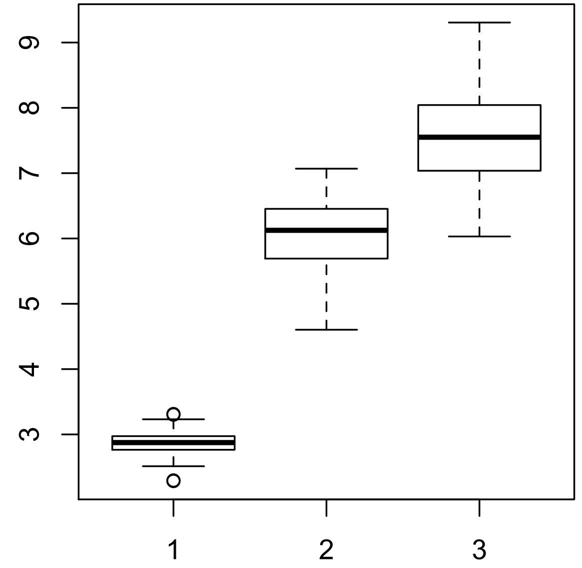
$$\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_p] \quad \text{(principal components)}$$

$$x'_1 = \mathbf{a}_1 \mathbf{x} = \sum_{j=1}^{p} a_{1j} x_j$$

Original data (1st variable)



Transformed data



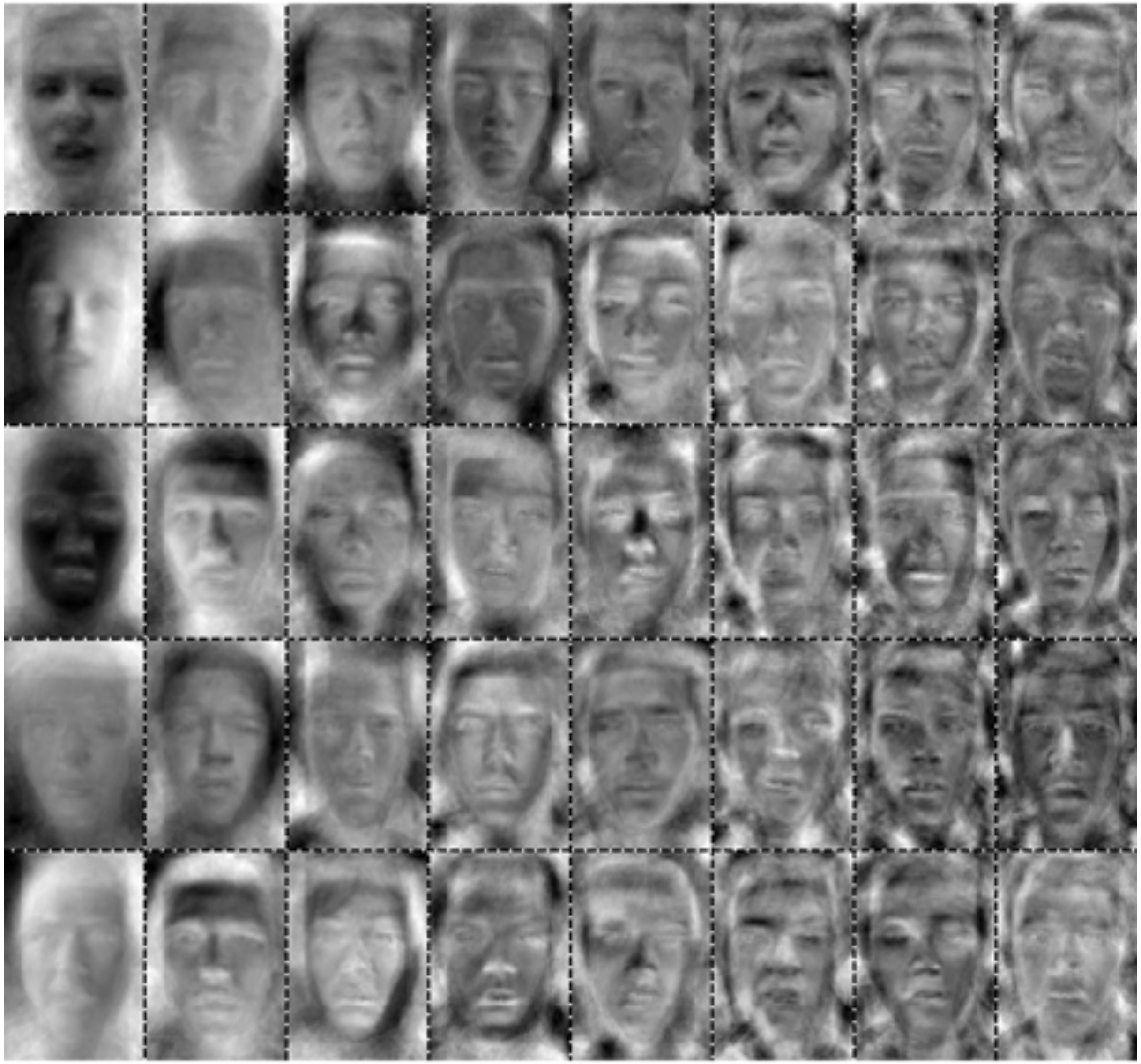
Example: Eigenfaces

PCA applied to images of human faces.

Reduce dimensionality to set of basis images.

All other images are linear combo of these "eigenpictures".

Used for facial recognition.



First 40 PCA dimensions

Principal component analysis

- **Task**:
  - Reduce dimensionality of data while capturing intrinsic variability
- **Data representation**:
  - **X** data matrix (n x p)
- **Knowledge representation**:
  - p x m matrix of weights that represent:
    Set of *m* alternative dimensions, where each dimension is represented by a p-dimensional vector of weights (e.g., [0.36, -0.08, 0.86, 0.36])

Principal component analysis

- **Learning**:
  - **Scoring function**:
    1) Minimize squared deviation from original points to projected points
    2) Maximize variance along each orthogonal direction
    (can show these two are equivalent)
  - **Search**: *Implicit* search by analytically determining basis vectors with best score (achieved by solving eigensystem with the covariance matrix Σ)
- **Inference**:
  - Project points into new m-dimensional space
    - E.g., $$x'_1 = \mathbf{a}_1 \mathbf{x} = \sum_{j=1}^{p} a_{1j} x_j$$

PCA complexity
$O(np^2+p^3)$