Feature Selection and Dimensionality Reduction

"Curse of dimensionality"

- The higher the dimensionality of the data, the more data is needed to learn a good model.

- Learned model may be overly complex, overfitted, especially if some dimensions are irrelevant to the classification task

How to deal with the curse of dimensionality?

- Assume a strong "prior": e.g., data follows a Gaussian distribution (so all we have do to is estimate the parameters)

- Reduce the dimensionality via:
  - Selecting only most relevant / useful features

  - Projecting data to lower dimensional space (e.g., principal components analysis)

Feature Selection

Goal: Select a subset of d features ($d < n$) in order to maximize classification performance.

Types of Feature Selection Methods

- Filter
- Wrapper
- Embedded

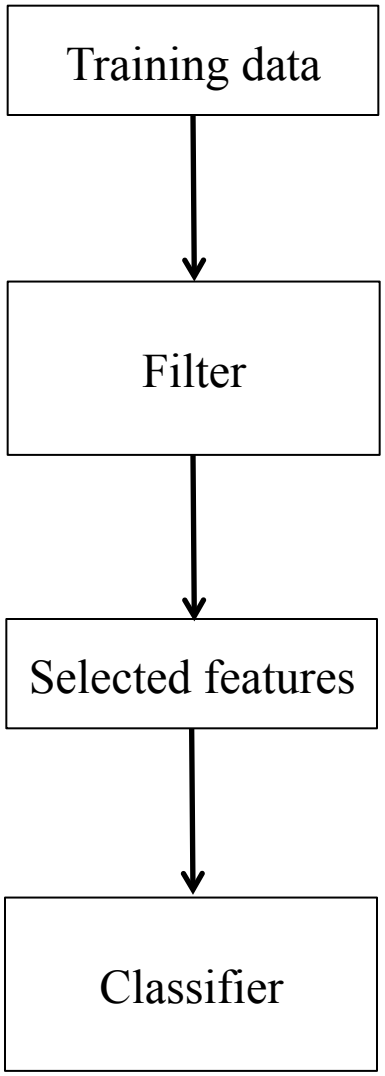
Filter Methods

Independent of the classification algorithm.

Apply a filtering function to the features before applying classifier.

Examples of filtering functions:

- Information gain of individual features
- Statistical variance of individual features
- Statistical correlation among features



Information Gain

Measures, from training data, how much information about the classification task an individual feature *f* provides.

Let S denote the training set.

$$\textit{Information Gain} = \textit{Entropy}\,(\,S\,) - \textit{Entropy}\,(\,S_f\,)$$

Intuitive idea of *entropy*

Entropy (or "information") = "degree of disorder"

Entropy measures the degree of uniformity or non-uniformity in a collection.

Example

$\{1, 1, 1, 1, 1, 1, 1\}$ vs. $\{1, -1, -1, -1, 1, 1, -1\}$

Note, this is the same as

$\{1, 1, 1, 1, 1, 1, 1\}$ vs. $\{-1, -1, -1, -1, 1, 1, 1\}$

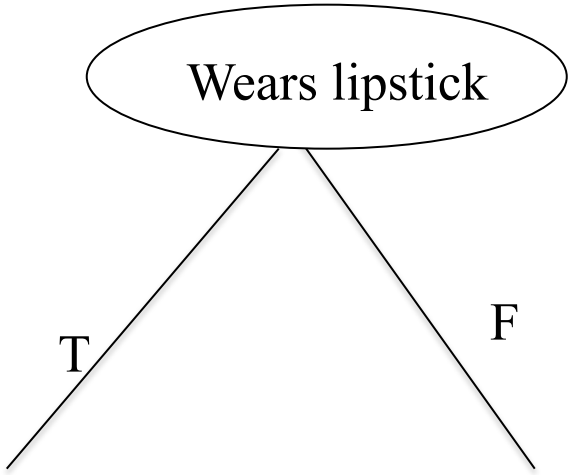
Intuitive idea of *entropy* of a training set

- **Task:** classify as Female or Male
- **Training set 1:** Barbara, Elizabeth, Susan, Lisa, Ellen, Kristi, Mike, John (lower entropy)
- **Training set 2:** Jane, Mary, Alice, Jenny, Bob, Allen, Doug, Steve (higher entropy)

Intuitive idea of *entropy* with respect to a feature

- **Task:** classify as Female or Male
- **Instances:** Jane, Mary, Alice, Bob, Allen, Doug
- **Features:** Each instance has two binary features: "wears lipstick" and "has long hair"

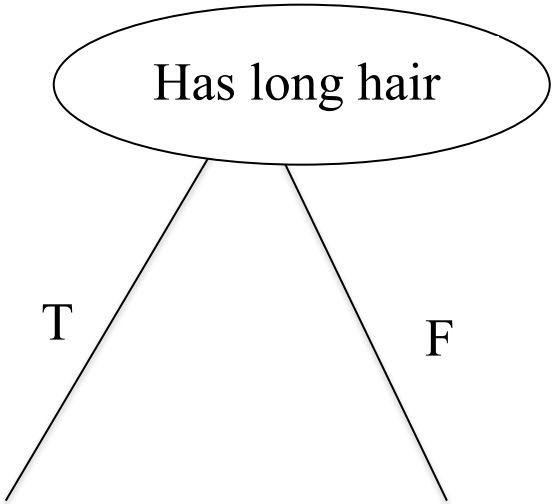
Pure split (feature has low entropy)



Jane, Mary, Alice, Jenny

Bob, Allen, Doug, Steve

Impure split (feature has high entropy)



Jane, Mary, Bob, Steve

Alice, Jenny, Allen, Doug

Entropy of a training set

Let S be a set of training examples.

p_+ = proportion of positive examples.

p_- = proportion of negative examples

$$Entropy(S) = -(p_+ \log_2 p_+ + p_- \log_2 p_-)$$

Roughly measures how predictable collection is, only on basis of distribution of + and − examples.

Examples?

Entropy with respect to a feature

Let f be a numeric feature. Let t be a threshold. Split training set S into two subsets:

S_f^{high} : training examples with $f \geq t$

S_f^{low} : training examples with $f < t$

Let

p_+^{high} = proportion of positive examples in S_f^{high}

p_-^{high} = proportion of negative examples in S_f^{high}

p_+^{low} = proportion of positive examples in S_f^{low}

p_-^{low} = proportion of negative examples in S_f^{low}

Then define:

$$Entropy(S_f^{high}) = -(p_+^{high} \log_2 p_+^{high} + p_-^{high} \log_2 p_-^{high})$$

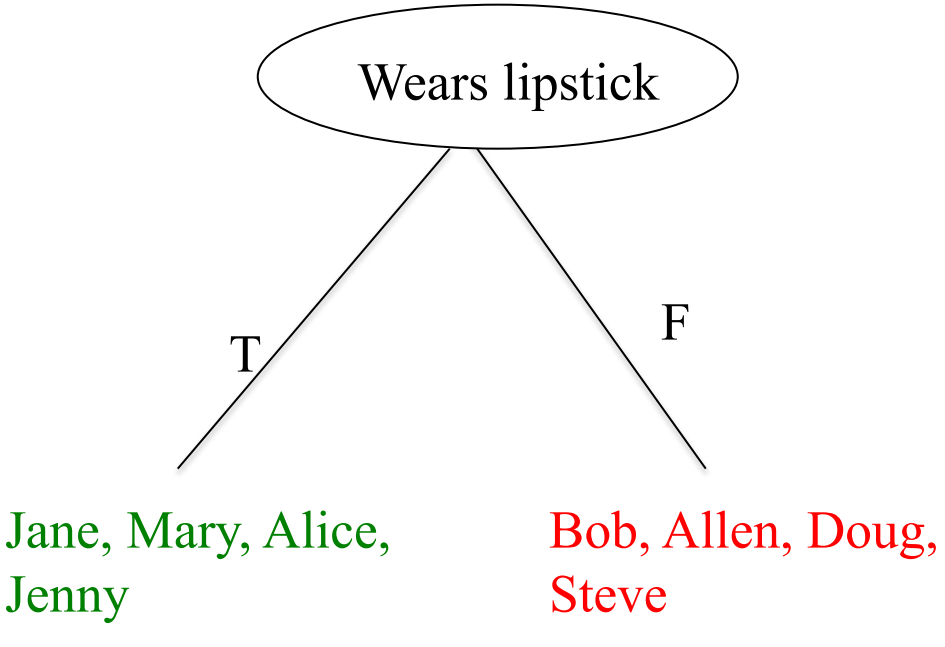
$$Entropy(S_f^{low}) = -(p_+^{low} \log_2 p_+^{low} + p_-^{low} \log_2 p_-^{low})$$

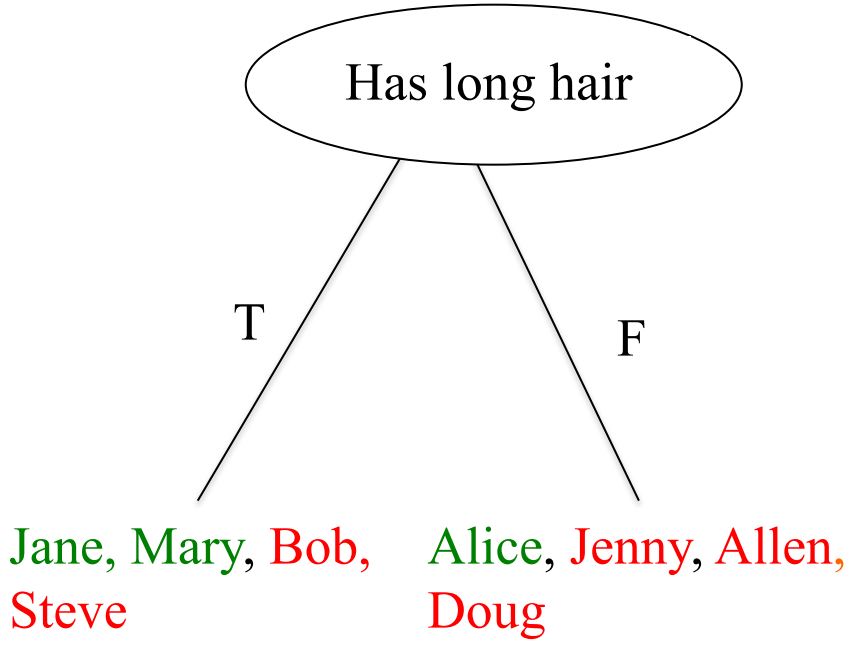
$$Entropy(S_f) = \frac{|S_f^{high}|}{|S|} Entropy(S_f^{high}) + \frac{|S_f^{low}|}{|S|} Entropy(S_f^{low})$$

$$InformationGain(S_f) = Entropy(S) - Entropy(S_f)$$

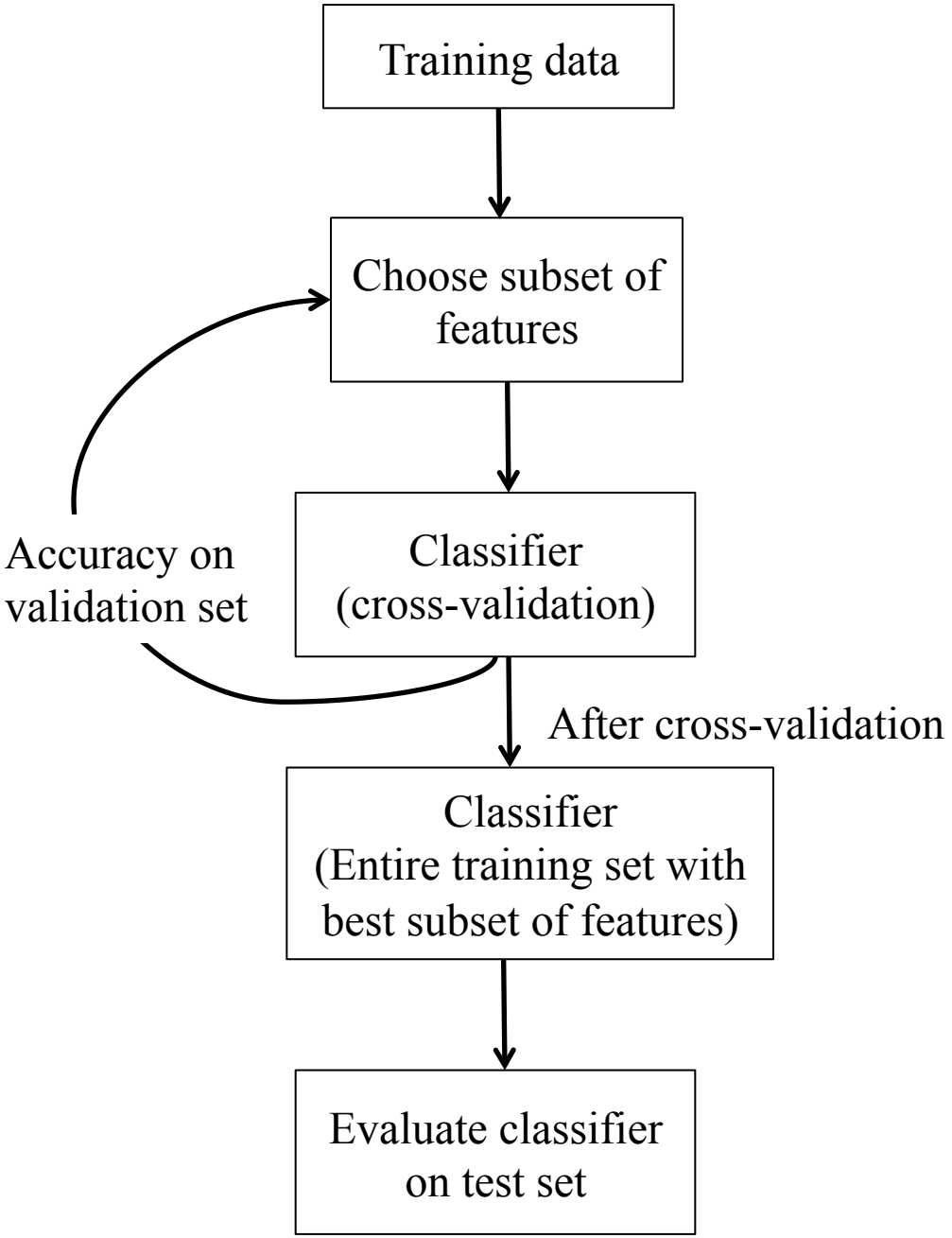
Example

Training set S: Jane, Mary, Alice, Jenny, Bob, Allen, Doug, Steve





Wrapper Methods



Filter Methods

Pros: Fast

Cons: Chosen filter might not be relevant for a specific kind of classifier.

Doesn't take into account interactions among features

Often hard to know how many features to select.

Filter Methods

Pros: Fast

Cons: Chosen filter might not be relevant for a specific kind of classifier.

Doesn't take into account interactions among features

Often hard to know how many features to select.

Wrapper Methods

Pros: Features are evaluated in context of classification

Wrapper method selects number of features to use

Cons: Slow

Filter Methods

Pros: Fast

Cons: Chosen filter might not be relevant for a specific kind of classifier.

Doesn't take into account interactions among features

Often hard to know how many features to select.

Wrapper Methods

Pros: Features are evaluated in context of classification

Wrapper method selects number of features to use

Cons: Slow

Intermediate method, often used with SVMs:

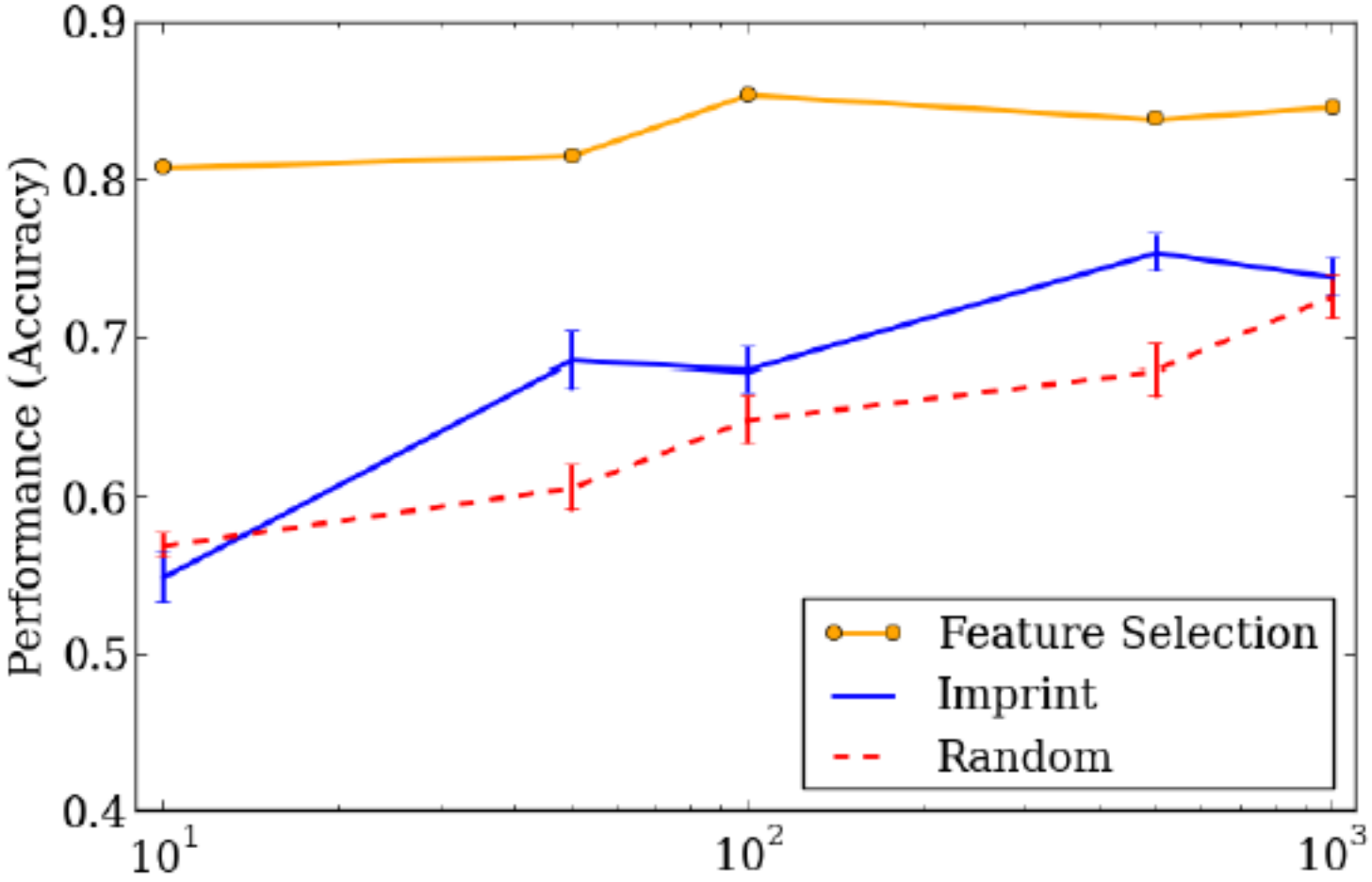
Train SVM using all features

Select features f_i with highest $| w_i |$

Retrain SVM with selected features

Test SVM on test set

From M., Thomure, *The Role of Prototype Learning in Hierarchical Models of Vision*



(a) *Cars v. Planes*

Embedded Methods

Feature selection is intrinsic part of learning algorithm

One example:

L_1 SVM: Instead of minimizing $\|\mathbf{w}\|$ (the "L2 norm") we minimize the L_1 norm of the weight vector:

$$\|\mathbf{w}\|_1 = \sum_{i=1}^{n} |w_i|$$

Result is that most of the weights go to zero, leaving a small subset of the weights.

Cf., field of "sparse coding"

Dimensionality Reduction:

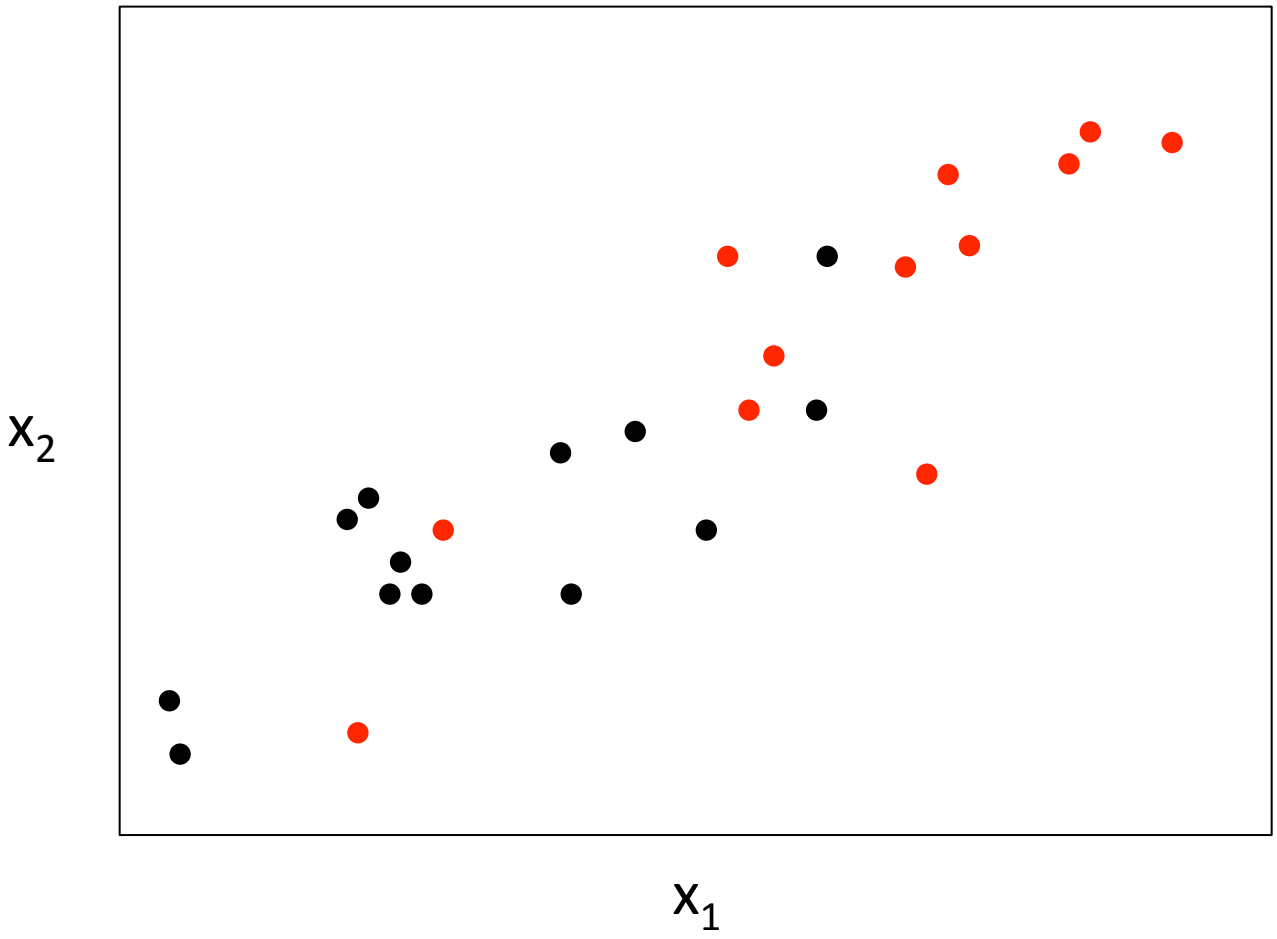
Principal Components Analysis

Reading:

Smith, A Tutorial on Principal Components Analysis (linked to class webpage)

- http://www.youtube.com/watch?v=BfTMmoDFXyE

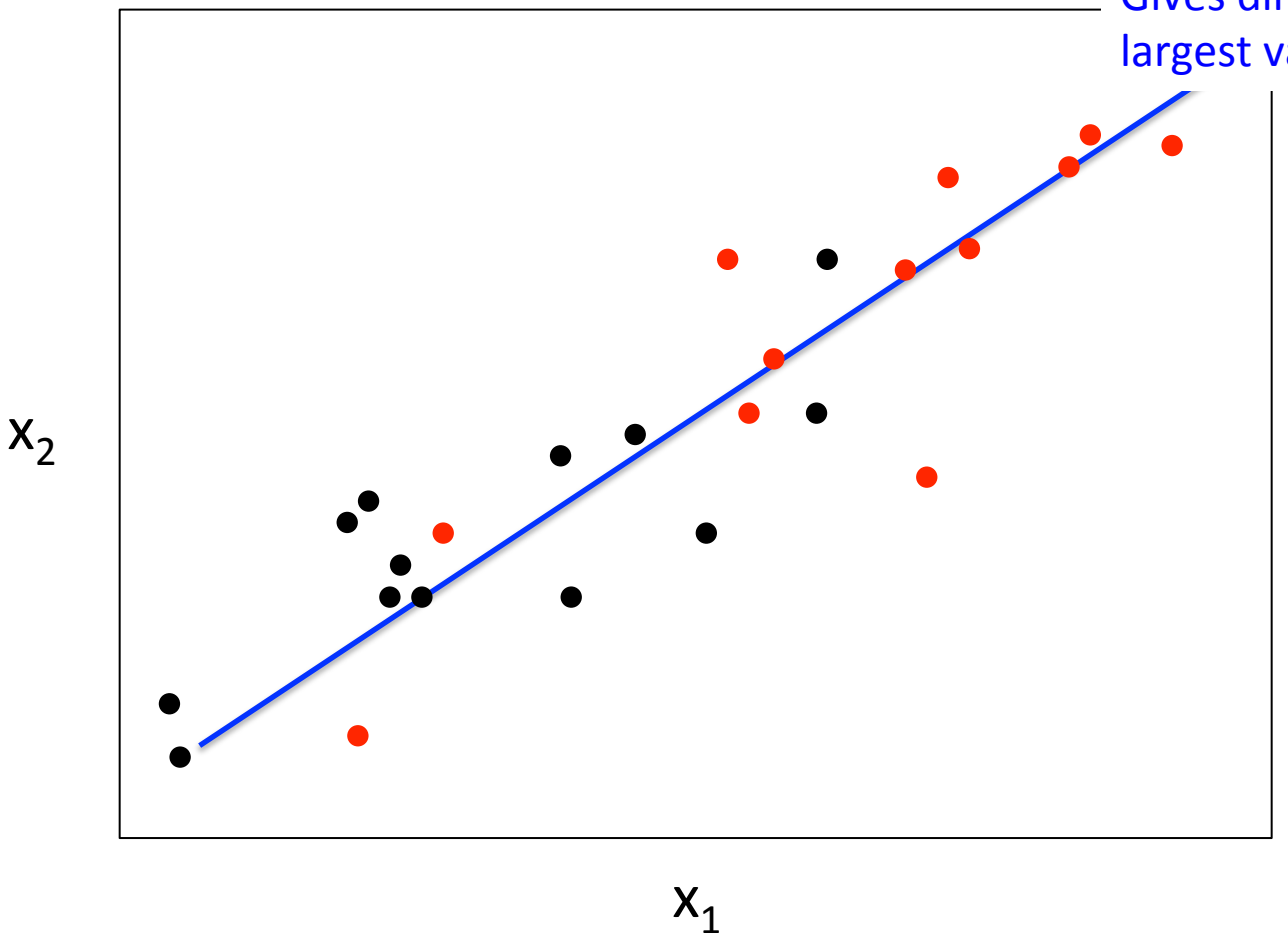
Data



Data

First principal component

Gives direction of
largest variation of the data



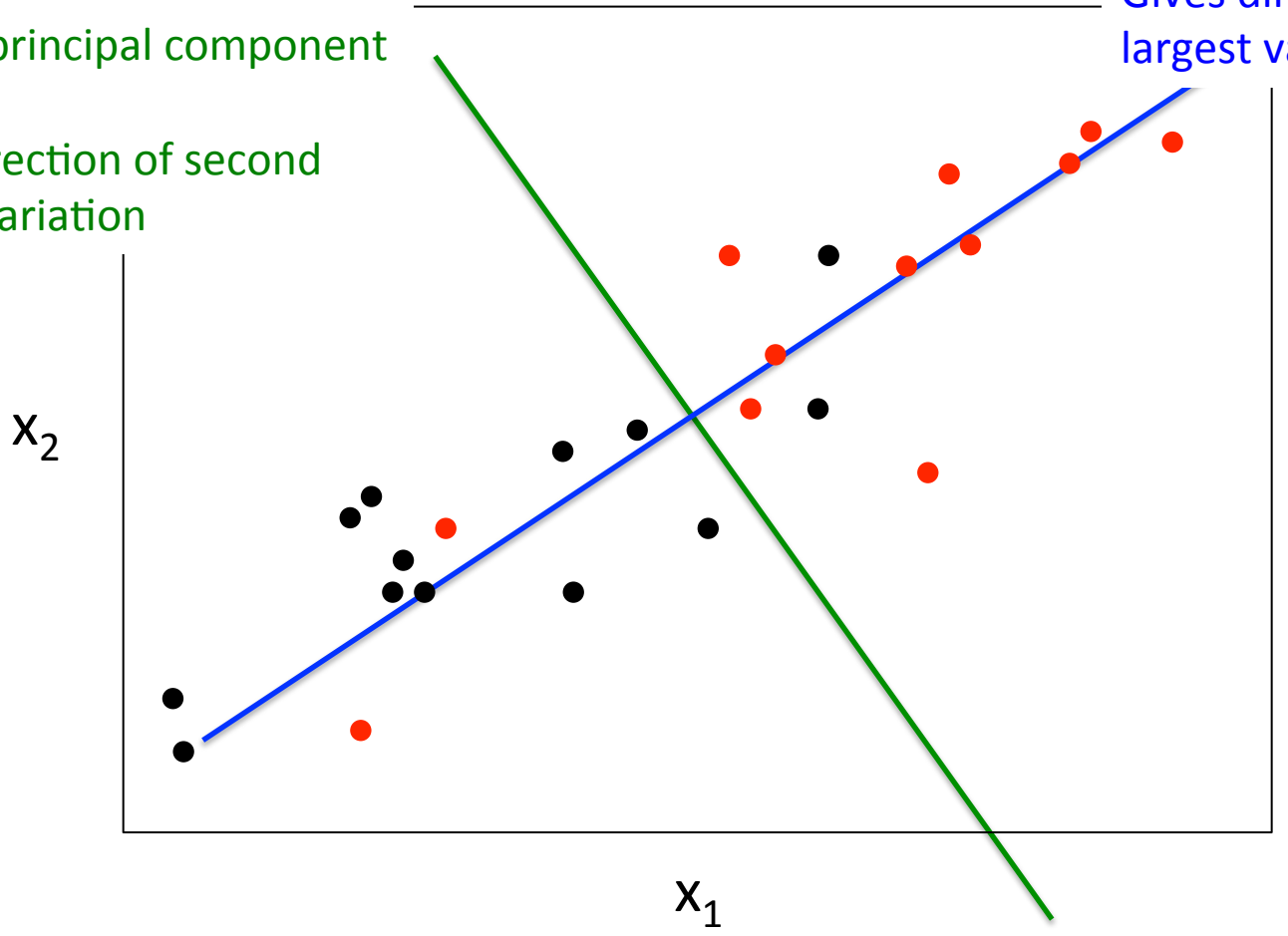
Data

First principal component

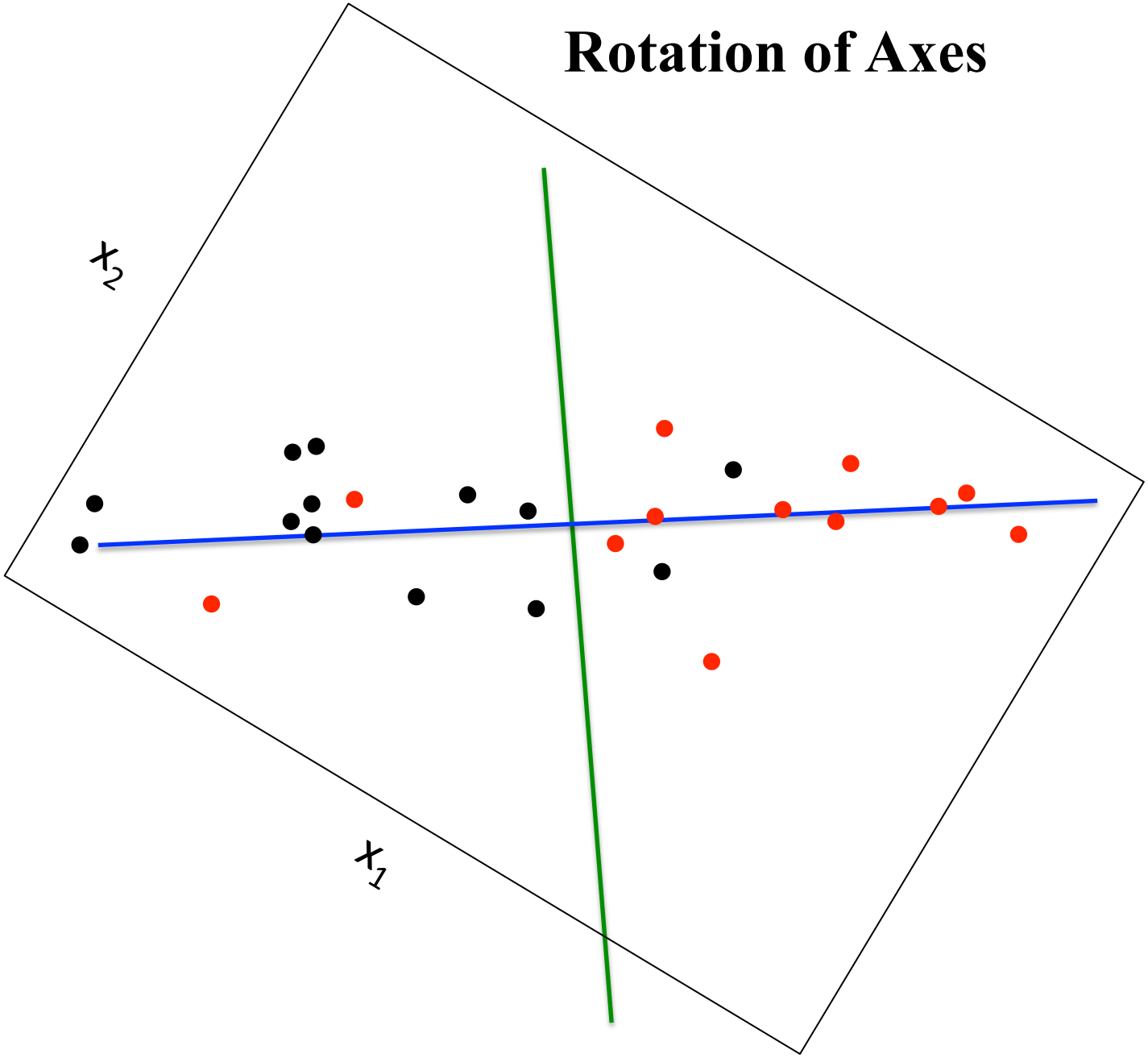
Gives direction of largest variation of the data

Second principal component

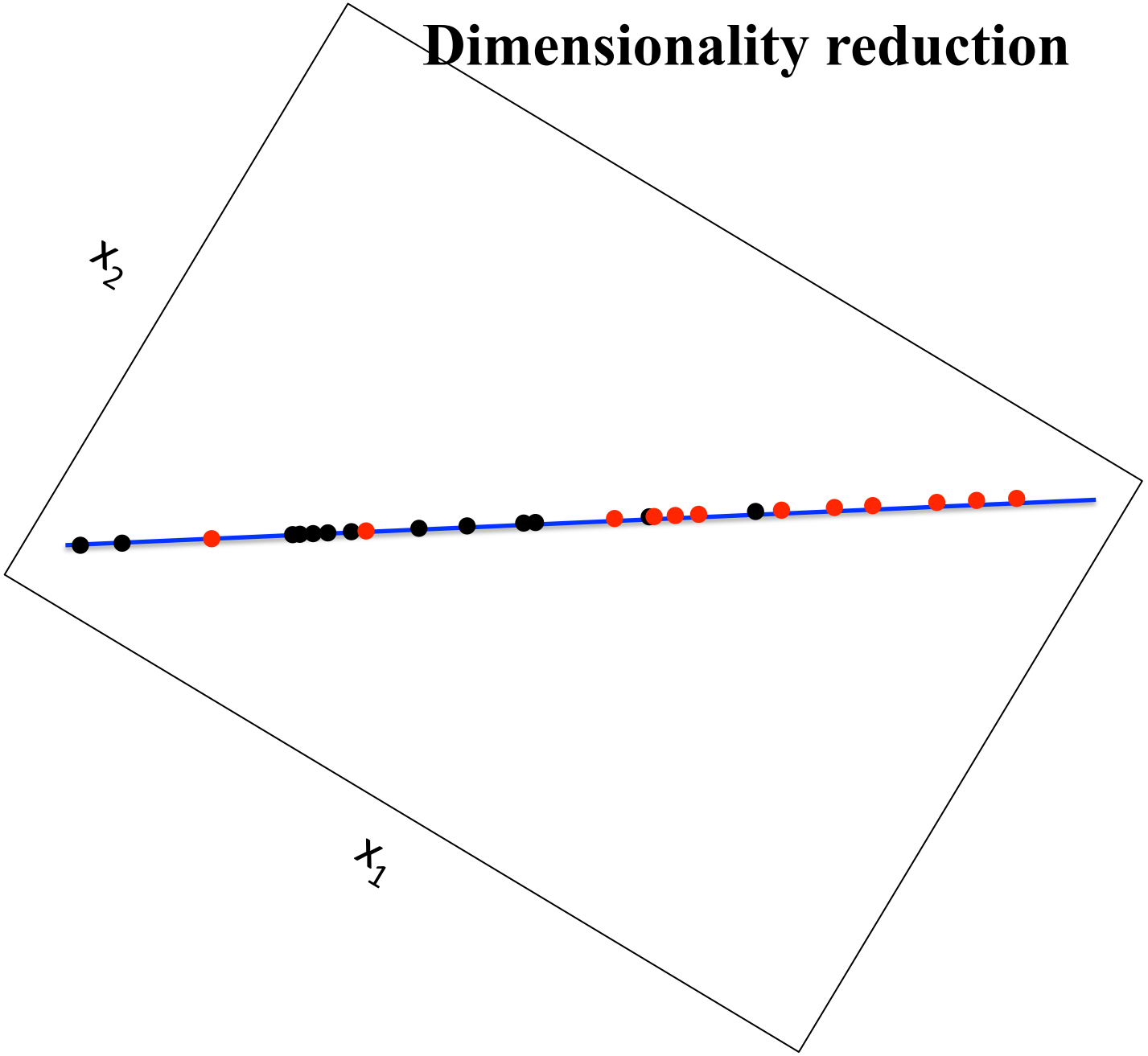
Gives direction of second largest variation



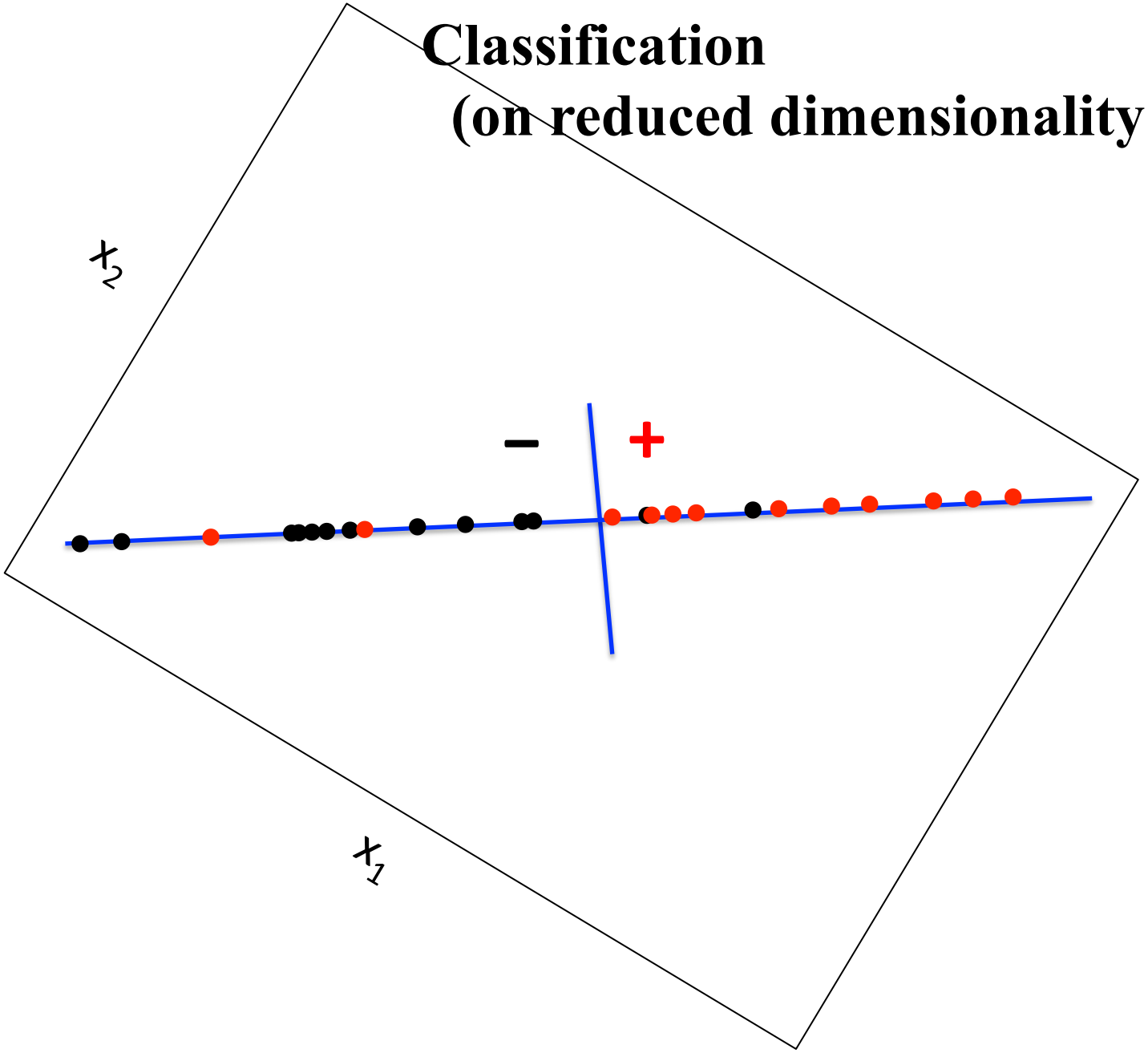
Rotation of Axes



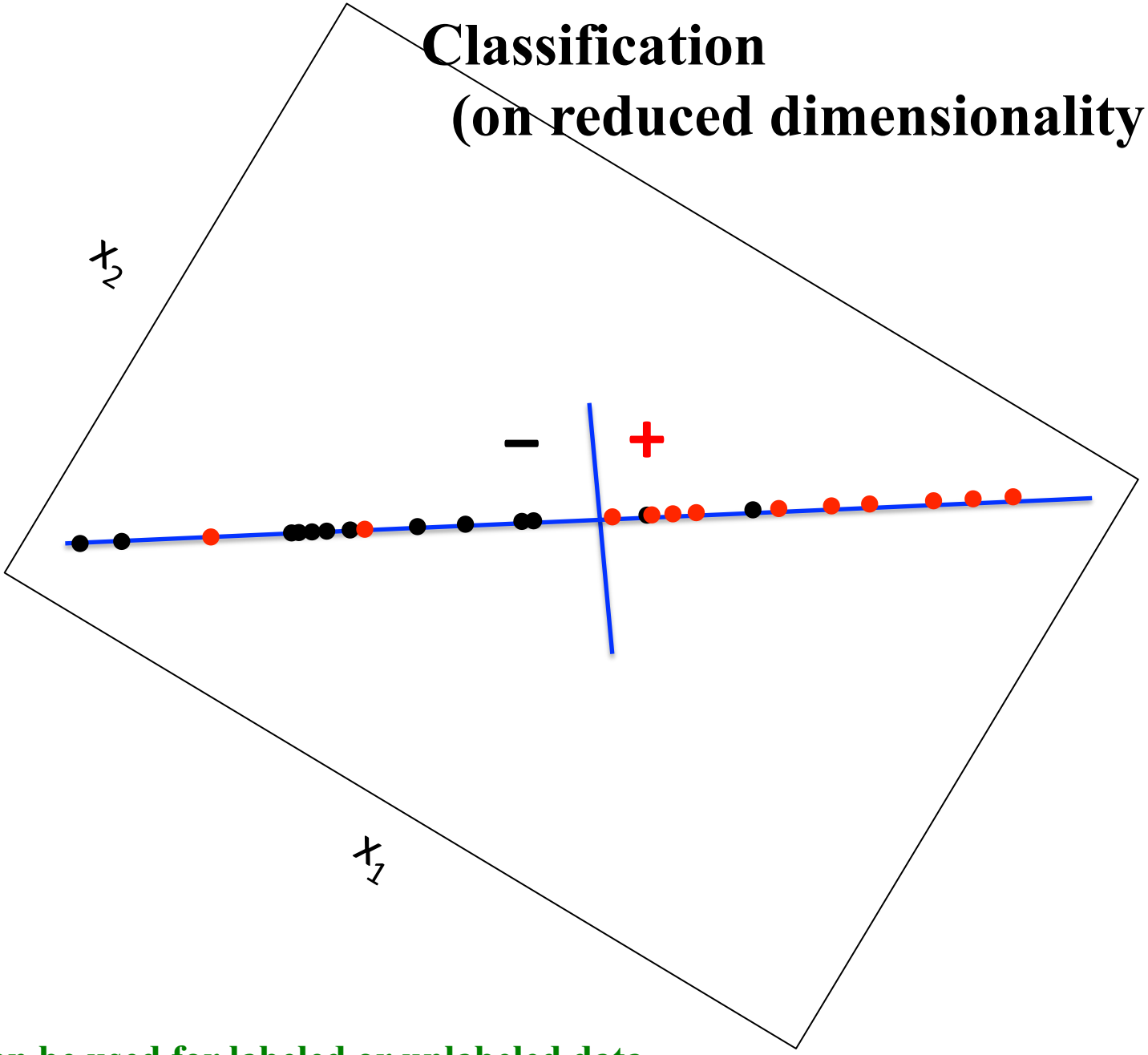
Dimensionality reduction



Classification
(on reduced dimensionality space)



Classification
(on reduced dimensionality space)



Note: Can be used for labeled or unlabeled data.

Principal Components Analysis (PCA)

- Summary: PCA finds new orthogonal axes in directions of largest variation in data.
- PCA used to create high-level features in order to improve classification and reduce dimensions of data without much loss of information.
- Used in machine learning and in signal processing and image compression (among other things).

Background for PCA

- Suppose features are A_1 and A_2, and we have n training examples. x's denote values of A_1 and y's denote values of A_2 over the training examples.

- Variance of an feature:

$$\text{var}(A_1) = \frac{\sum_{i=1}^{n} (x_i - \bar{x})^2}{(n-1)}$$

- Covariance of two features:

$$\mathrm{cov}(A_1, A_2) = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{(n-1)}$$

- If covariance is positive, both dimensions increase together. If negative, as one increases, the other decreases. Zero: independent of each other.

- Covariance matrix
  - Suppose we have *n* features, A_1, ..., A_n.
  - Covariance matrix:

$$C^{n\times n} = (c_{i,j}), \text{where } c_{i,j} = \text{cov}(A_i, A_j)$$

	Hours(H)	*Mark(M)*
Data	9	39
	15	56
	25	93
	14	61
	10	50
	18	75
	0	32
	16	85
	5	42
	19	70
	16	66
	20	80
Totals	167	749
Averages	13.92	62.42

Covariance:

H	M	$(H_i - \bar{H})$	$(M_i - \bar{M})$	$(H_i - \bar{H})(M_i - \bar{M})$
9	39	-4.92	-23.42	115.23
15	56	1.08	-6.42	-6.93
25	93	11.08	30.58	338.83
14	61	0.08	-1.42	-0.11
10	50	-3.92	-12.42	48.69
18	75	4.08	12.58	51.33
0	32	-13.92	-30.42	423.45
16	85	2.08	22.58	46.97
5	42	-8.92	-20.42	182.15
19	70	5.08	7.58	38.51
16	66	2.08	3.58	7.45
20	80	6.08	17.58	106.89
Total				1149.89
Average				104.54

Table 2.2: 2-dimensional data set and covariance calculation

$$\begin{pmatrix} \mathrm{cov}(H,H) & \mathrm{cov}(H,M) \\ \mathrm{cov}(M,H) & \mathrm{cov}(M,M) \end{pmatrix}$$

$$= \begin{pmatrix} \mathrm{var}(H) & 104.5 \\ 104.5 & \mathrm{var}(M) \end{pmatrix}$$

$$= \begin{pmatrix} 47.7 & 104.5 \\ 104.5 & 370 \end{pmatrix}$$

Covariance matrix

Review of Matrix Algebra

- Eigenvectors:
  - Let **M** be an $n \times n$ matrix.
    - **v** is an *eigenvector* of **M** if $\mathbf{M} \times \mathbf{v} = \lambda\mathbf{v}$
    - λ is called the *eigenvalue* associated with **v**
  - For any eigenvector **v** of **M** and scalar a,

$$\mathbf{M} \times a\mathbf{v} = \lambda a\mathbf{v}$$

  - Thus you can always choose eigenvectors of length 1:

$$\sqrt{v_1^2 + \ldots + v_n^2} = 1$$

  - If **M** is symmetric with real entries, it has n eigenvectors, and they are orthogonal to one another.
  - Thus eigenvectors can be used as a new basis for a n-dimensional vector space.

Principal Components Analysis (PCA)

1. Given original data set S = $\{\mathbf{x}^1, ..., \mathbf{x}^k\}$, produce new set by subtracting the mean of feature A_i from each x_i.

Data =

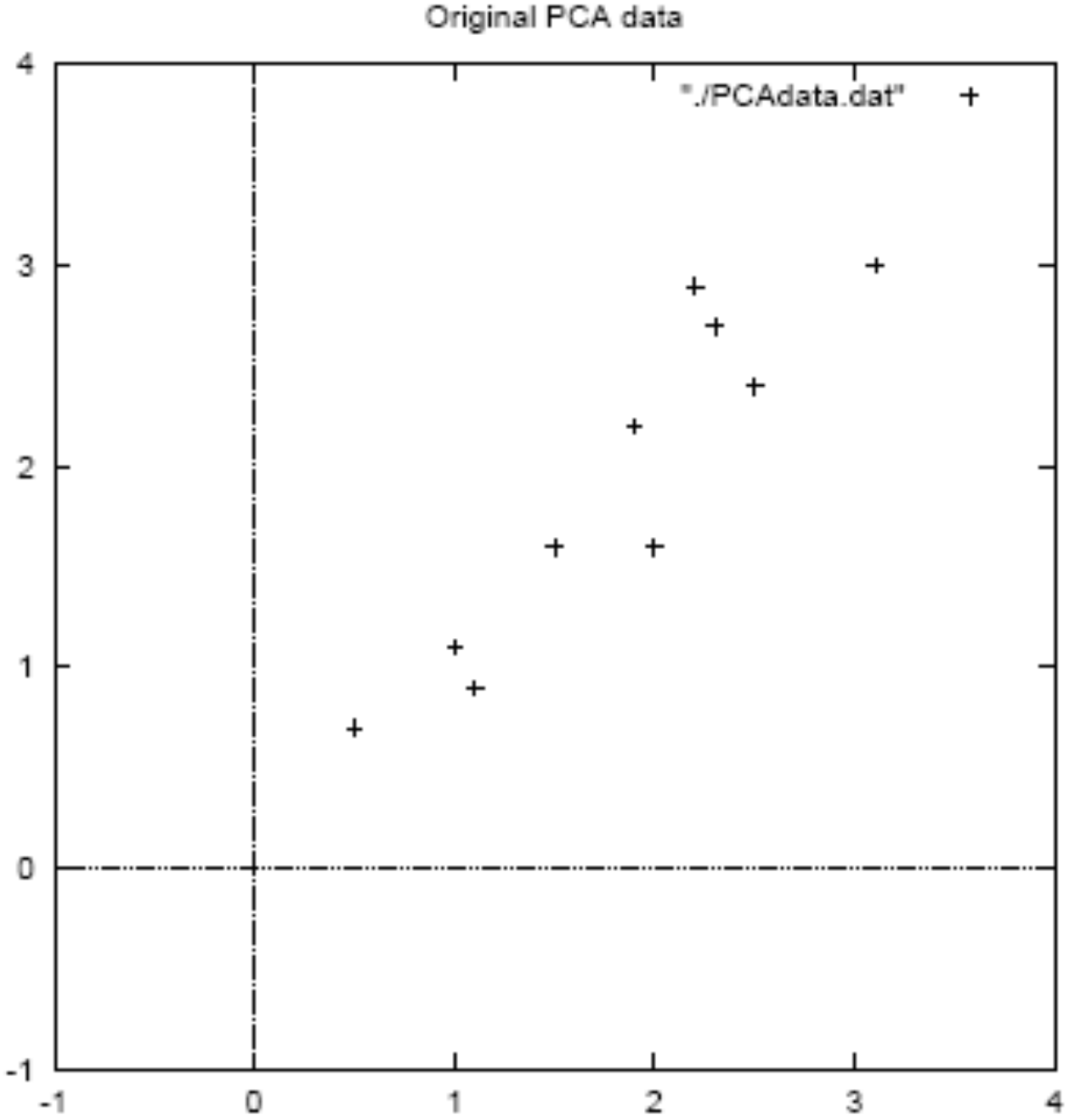
x	y
2.5	2.4
0.5	0.7
2.2	2.9
1.9	2.2
3.1	3.0
2.3	2.7
2	1.6
1	1.1
1.5	1.6
1.1	0.9

Mean: 1.81 1.91

DataAdjust =

x	y
.69	.49
-1.31	-1.21
.39	.99
.09	.29
1.29	1.09
.49	.79
.19	-.31
-.81	-.81
-.31	-.31
-.71	-1.01

Mean: 0 0


Original PCA data
"./PCAdata.dat"
4
3
2
1
0
-1
-1
0
1
2
3
4

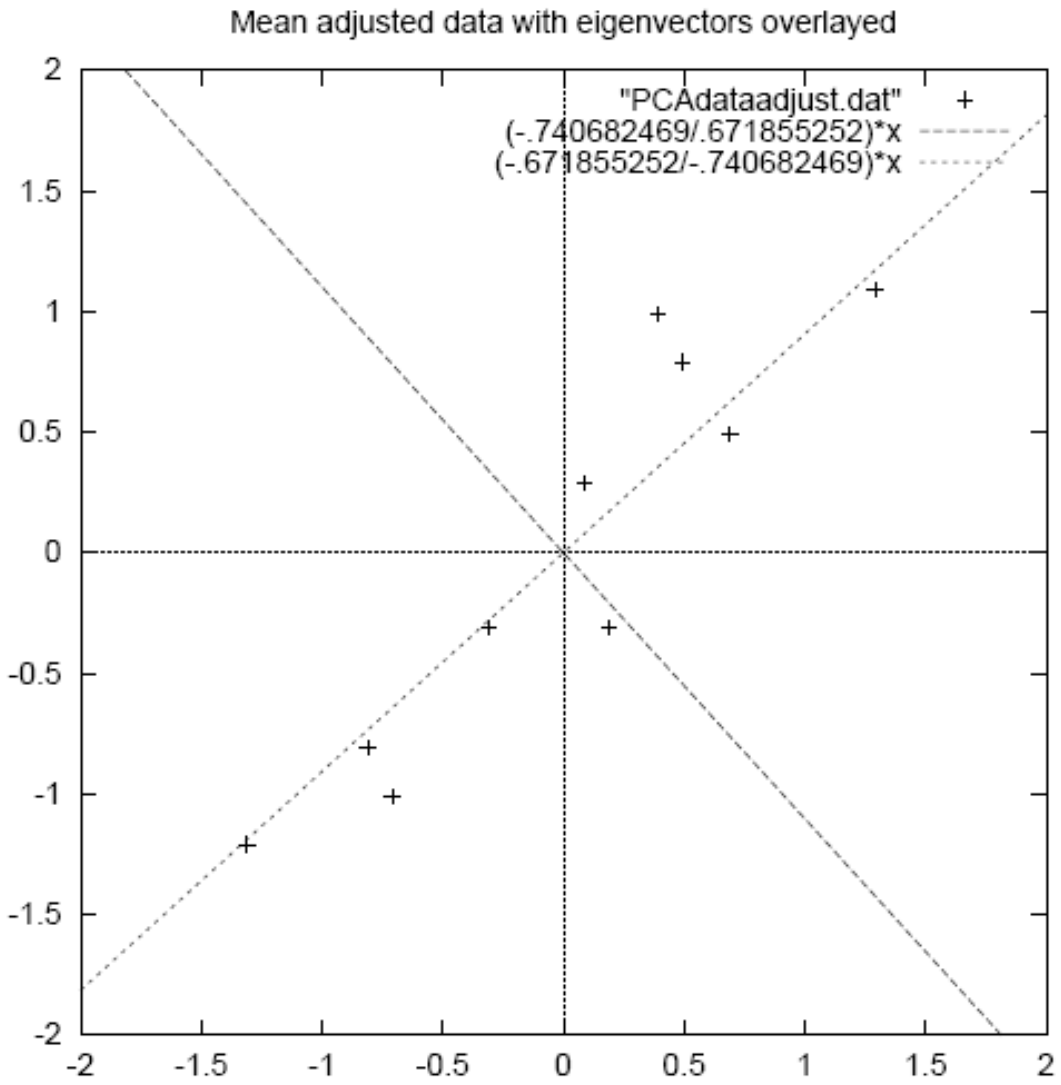
2. Calculate the covariance matrix:

$$cov = \begin{array}{c} \\ x \\ y \end{array} \begin{array}{c} \begin{array}{cc} x & y \end{array} \\ \begin{pmatrix} .616555556 & .615444444 \\ .615444444 & .716555556 \end{pmatrix} \end{array}$$

3. Calculate the (unit) eigenvectors and eigenvalues of the covariance matrix:

$$eigenvalues = \begin{pmatrix} .0490833989 \\ 1.28402771 \end{pmatrix}$$

$$eigenvectors = \begin{pmatrix} -.735178656 & -.677873399 \\ .677873399 & -.735178656 \end{pmatrix}$$



Eigenvector with largest eigenvalue traces linear pattern in data

Figure 3.2: A plot of the normalised data (mean subtracted) with the eigenvectors of the covariance matrix overlayed on top.

4. Order eigenvectors by eigenvalue, highest to lowest.

$$\mathbf{v}_1 = \begin{pmatrix} -.677873399 \\ -.735178956 \end{pmatrix} \quad \lambda = 1.28402771$$

$$\mathbf{v}_2 = \begin{pmatrix} -.735178956 \\ .677873399 \end{pmatrix} \quad \lambda = .0490833989$$

In general, you get *n* components. To reduce dimensionality to *p*, ignore *n-p* components at the bottom of the list.

Construct new "feature vector" (assuming $\mathbf{v}_i$ is a column vector).

Feature vector = $(\mathbf{v}_1, \mathbf{v}_2, \ldots \mathbf{v}_p)$

$$FeatureVector1 = \begin{pmatrix} \mathbf{v}_1 & \mathbf{v}_2 \\ -.677873399 & -.735178956 \\ -.735178956 & .677873399 \end{pmatrix}$$

or reduced dimension feature vector :

$$FeatureVector2 = \begin{pmatrix} -.677873399 \\ -.735178956 \end{pmatrix}$$

5. Derive the new data set.

$$TransformedData = RowFeatureVector \times RowDataAdjust$$

where *RowDataAdjust = transpose of mean-adjusted data*

$$RowFeatureVector1 = \begin{pmatrix} -.677873399 & -.735178956 \\ -.735178956 & .677873399 \end{pmatrix}$$

$$RowFeatureVector2 = \begin{pmatrix} -.677873399 & -.735178956 \end{pmatrix}$$

$$RowDataAdjust = \begin{pmatrix} .69 & -1.31 & .39 & .09 & 1.29 & .49 & .19 & -.81 & -.31 & -.71 \\ .49 & -1.21 & .99 & .29 & 1.09 & .79 & -.31 & -.81 & -.31 & -1.01 \end{pmatrix}$$

This gives original data in terms of chosen components (eigenvectors)—that is, along these axes.

	x	y
	-.827970186	-.175115307
	1.77758033	.142857227
	-.992197494	.384374989
	-.274210416	.130417207
Transformed Data=	-1.67580142	-.209498461
	-.912949103	.175282444
	.0991094375	-.349824698
	1.14457216	.0464172582
	.438046137	.0177646297
	1.22382056	-.162675287

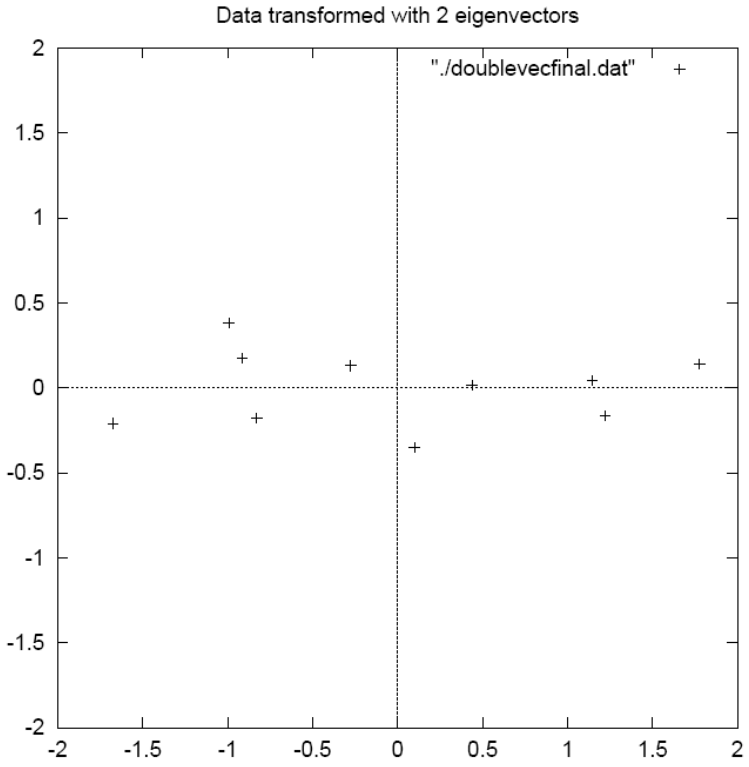


Figure 3.3: The table of data by applying the PCA analysis using both eigenvectors, and a plot of the new data points.

Transformed Data=

x	y
-.827970186	-.175115307
1.77758033	.142857227
-.992197494	.384374989
-.274210416	.130417207
-1.67580142	-.209498461
-.912949103	.175282444
.0991094375	-.349824698
1.14457216	.0464172582
.438046137	.0177646297
1.22382056	-.162675287

Intuition: We projected the data onto new axes that captures the strongest linear trends in the data set. Each transformed data point tells us how far it is above or below those trend lines.

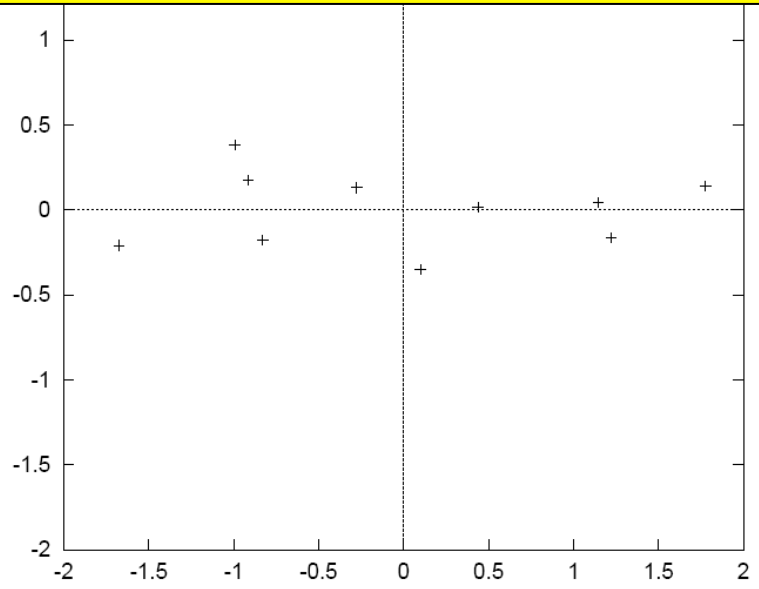
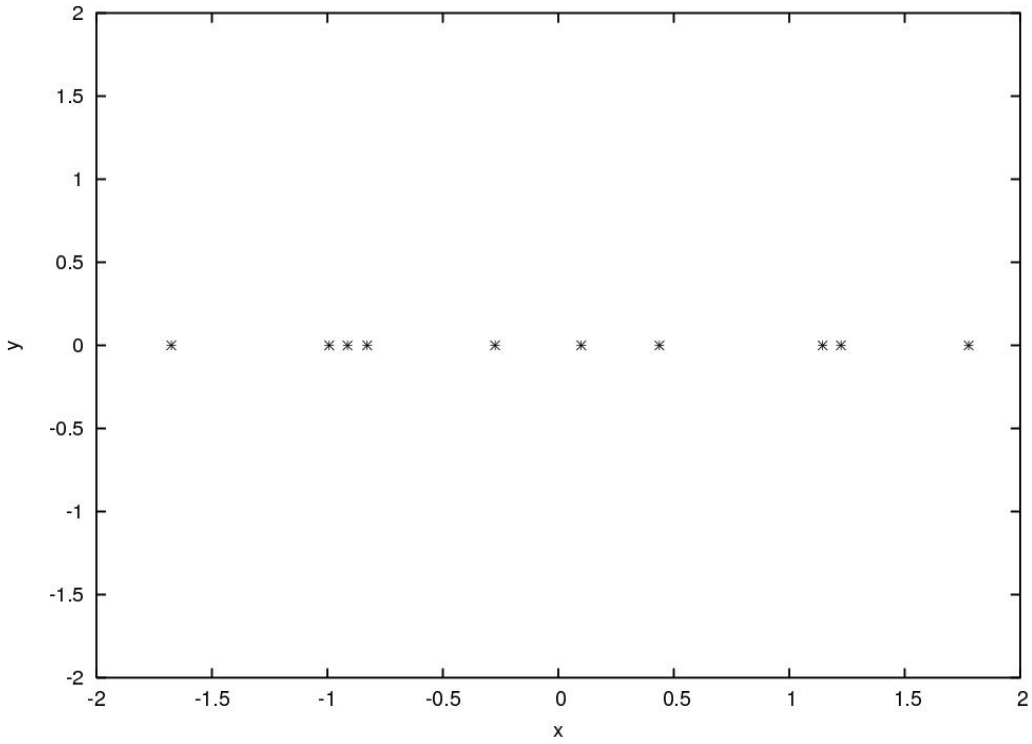


Figure 3.3: The table of data by applying the PCA analysis using both eigenvectors, and a plot of the new data points.

Transformed Data (Single eigenvecto

x
-.827970186
1.77758033
-.992197494
-.274210416
-1.67580142
-.912949103
.0991094375
1.14457216
.438046137
1.22382056



Reconstructing the original data

We did:

$$TransformedData = RowFeatureVector \times RowDataAdjust$$

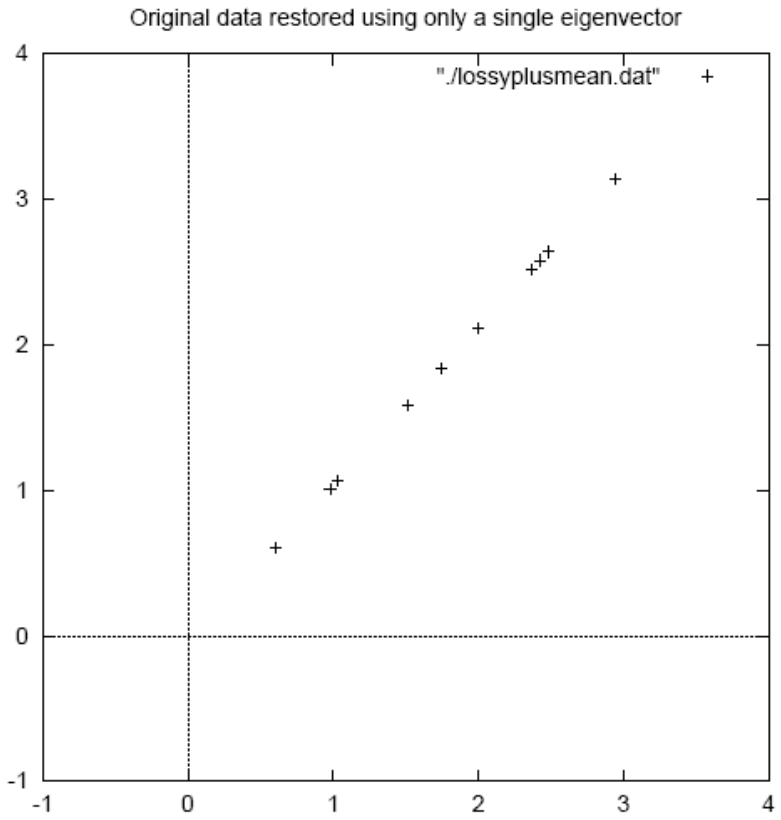
so we can do

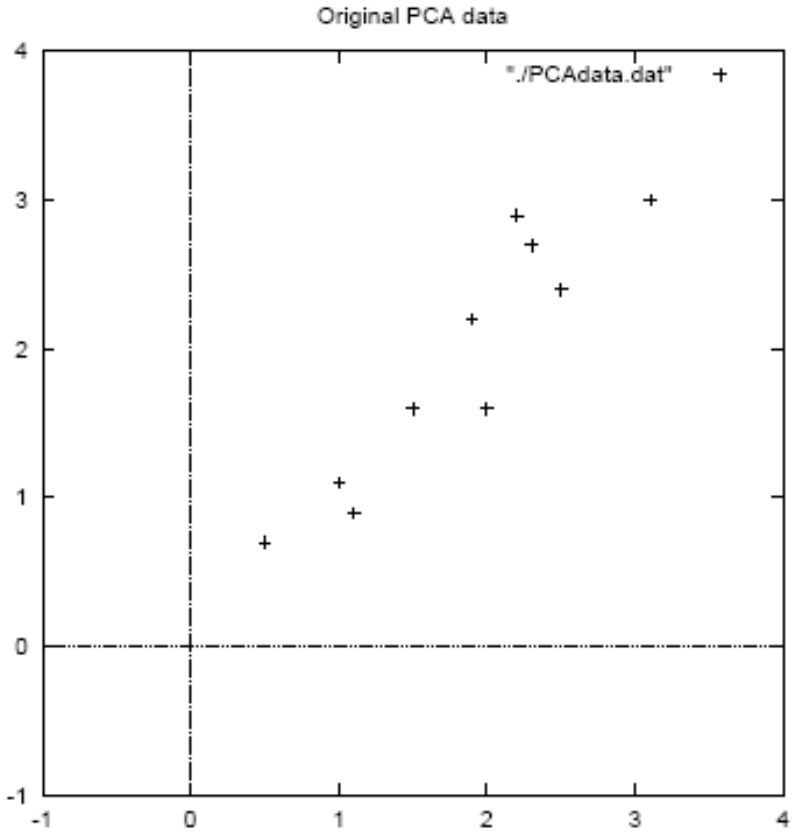
$$RowDataAdjust = RowFeatureVector^{-1} \times TransformedData$$

$$= RowFeatureVector^{T} \times TransformedData$$

and

$$RowDataOriginal = RowDataAdjust + OriginalMean$$


Original data restored using only a single eigenvector
"./lossyplusmean.dat"
4
3
2
1
0
-1
-1
0
1
2
3
4

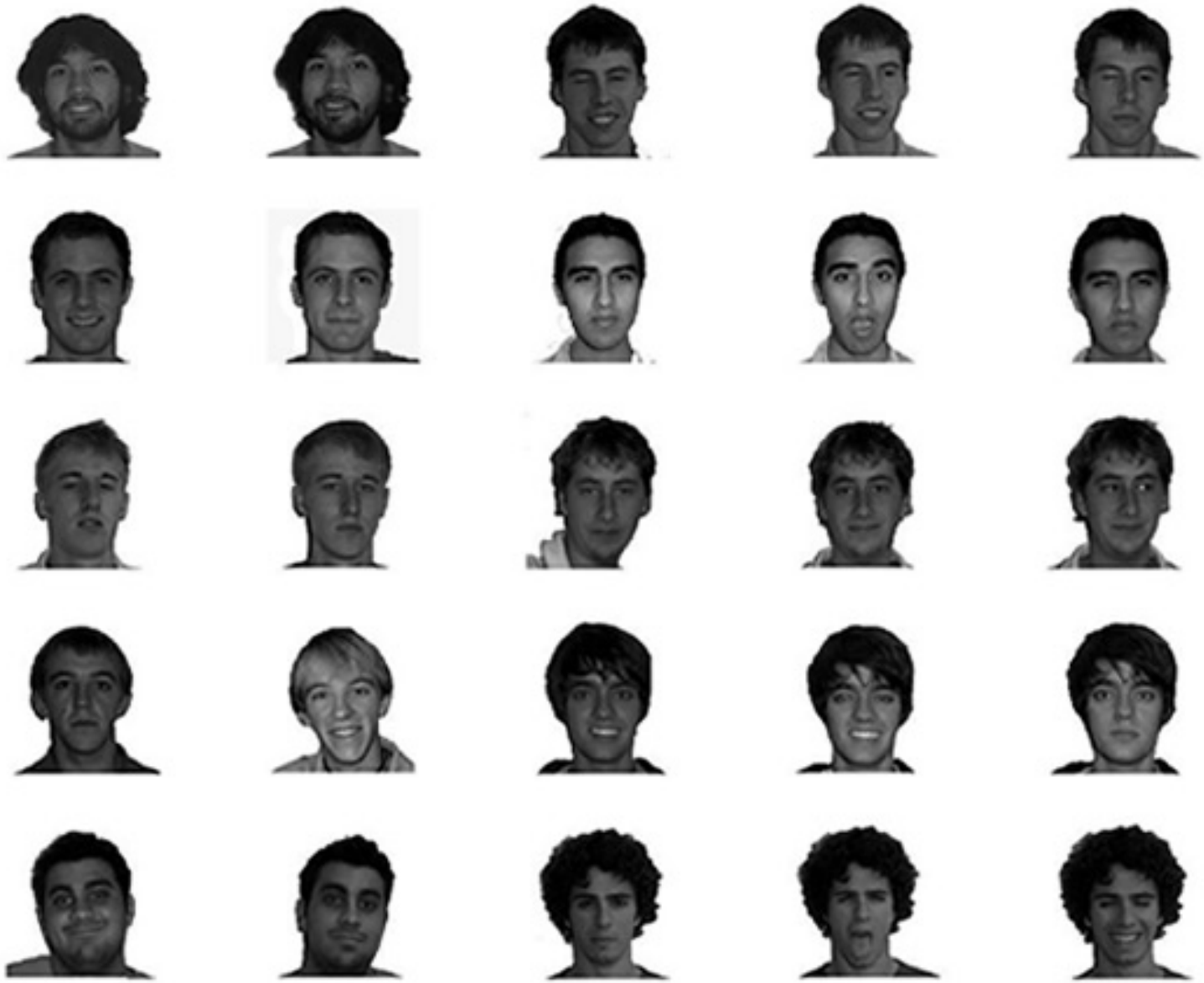

Original PCA data
"./PCAdata.dat"
4
3
2
1
0
-1
-1
0
1
2
3
4

What you need to remember

- General idea of what PCA does
  - Finds new, rotated set of orthogonal axes that capture directions of largest variation
  - Allows some axes to be dropped, so data can be represented in lower-dimensional space.
  - This can improve classification performance and avoid overfitting due to large number of dimensions.

Example: Linear discrimination using PCA for face recognition (“Eigenfaces”)

1. Preprocessing: “Normalize” faces
    - Make images the same size
    - Line up with respect to eyes
    - Normalize intensities



2. Raw features are pixel intensity values (2061 features)

3. Each image is encoded as a vector Γ_i of these features

4. Compute "mean" face in training set:

$$\Psi = \frac{1}{M}\sum_{i=1}^{M}\Gamma_i$$

From W. Zhao et al., Discriminant analysis of principal components for face recognition.

The average face and first four eigenfaces

- Subtract the mean face from each face vector

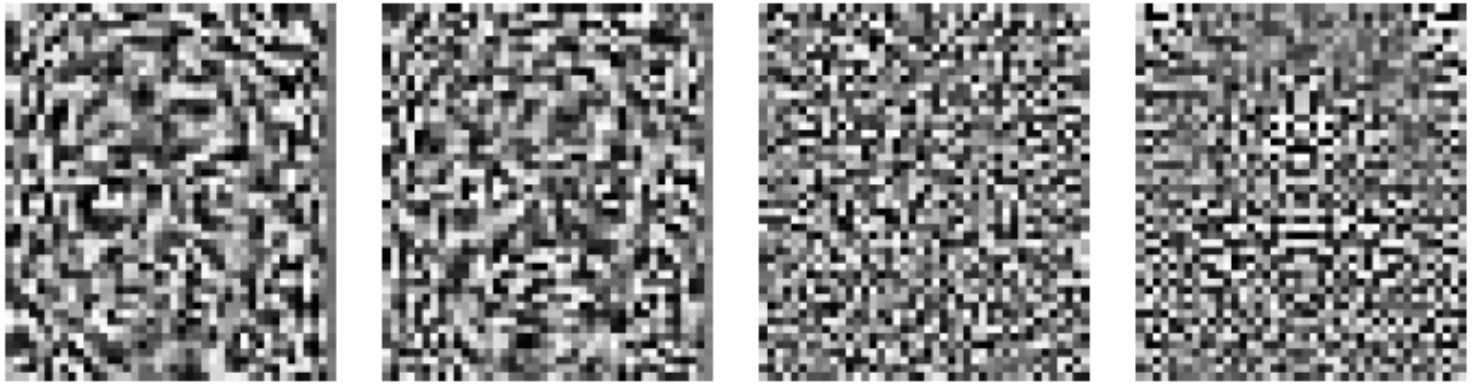
$$\Phi_i = \Gamma_i - \Psi$$

- Compute the covariance matrix **C**

- Compute the (unit) eigenvectors $\mathbf{v}_i$ of **C**

- Keep only the first *K* principal components (eigenvectors)

Eigenfaces 15, 100, 200, 250, 300



Eigenfaces 400, 450, 1000, 2000

Interpreting and Using Eigenfaces

The *eigenfaces* encode the principal sources of variation in the dataset (e.g., absence/presence of facial hair, skin tone, glasses, etc.).

We can represent any face as a linear combination of these "basis" faces.

Use this representation for:

- Face recognition
  (e.g., Euclidean distance from known faces)

- Linear discrimination
  (e.g., "glasses" versus "no glasses", or "male" versus "female")

Eigenfaces Demo

- http://demonstrations.wolfram.com/FaceRecognitionUsingTheEigenfaceAlgorithm/

Kernel PCA

- PCA: Assumes direction of variation are all straight lines
- Kernel PCA: Maps data to higher dimensional space

Kernel PCA

- Note that covariance terms can be expressed in terms of dot products:

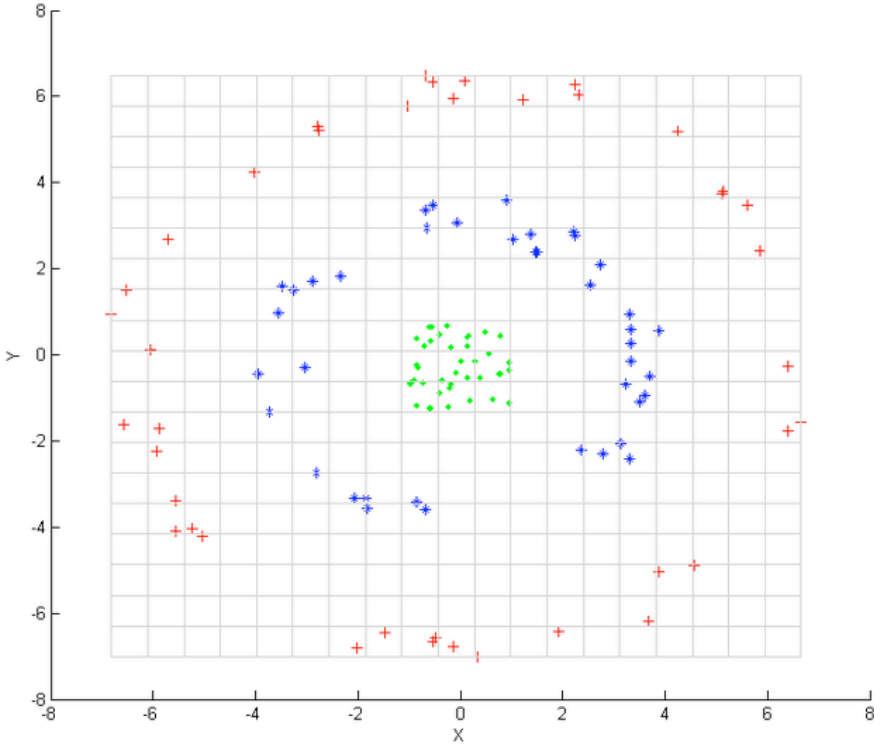
$$\mathrm{cov}(A_1, A_2) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{(n-1)} = \frac{\sum_{i=1}^{n} x_i' y_i'}{(n-1)} = \frac{\mathbf{x}' \cdot \mathbf{y}'}{(n-1)}$$

- Use kernel function K(**x**,**y**) to project to higher-dimensional space:

$$\mathrm{cov}(A_1, A_2) = \frac{K(\mathbf{x}', \mathbf{y}')}{(n-1)}$$

From Wikipedia

Original data



Data after kernel PCA