

LECTURE 9: Principal Components Analysis

- **The curse of dimensionality**
- **Dimensionality reduction**
- **Feature selection vs. feature extraction**
- **Signal representation vs. signal classification**
- **Principal Components Analysis**



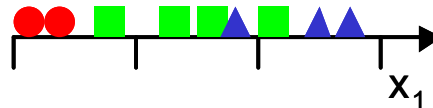
The curse of dimensionality (1)

■ The curse of dimensionality

- A term coined by Bellman in 1961
- Refers to the problems associated with multivariate data analysis as the dimensionality increases
- We will illustrate these problems with a simple example

■ Consider a 3-class pattern recognition problem

- A simple approach would be to
 - Divide the feature space into uniform bins
 - Compute the ratio of examples for each class at each bin and,
 - For a new example, find its bin and choose the predominant class in that bin
- In our toy problem we decide to start with one single feature and divide the real line into 3 segments

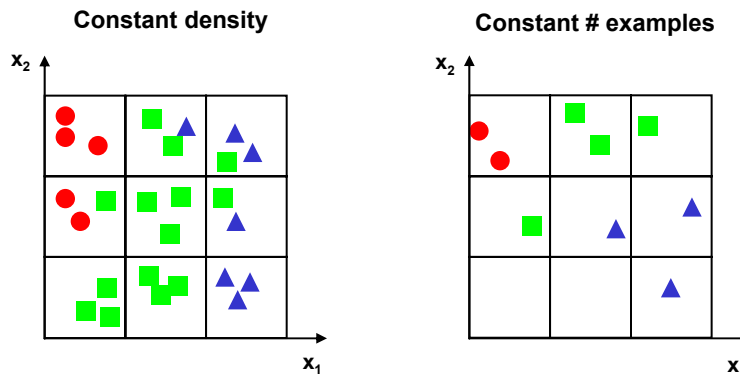


- After doing this, we notice that there exists too much overlap among the classes, so we decide to incorporate a second feature to try and improve separability

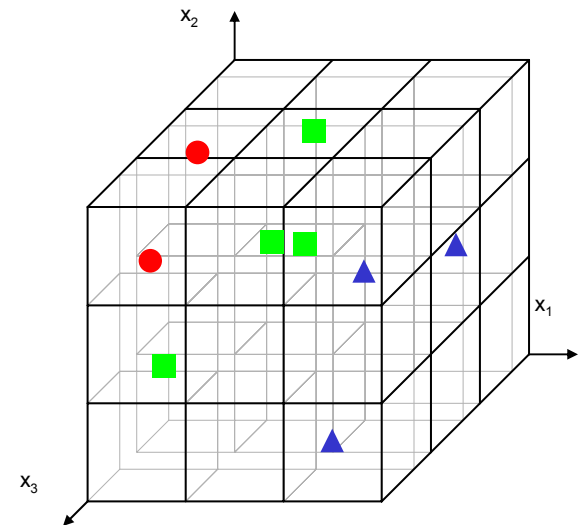


The curse of dimensionality (2)

- We decide to preserve the granularity of each axis, which raises the number of bins from 3 (in 1D) to $3^2=9$ (in 2D)
 - At this point we need to make a decision: do we maintain the density of examples per bin or do we keep the number of examples had for the one-dimensional case?
 - Choosing to maintain the density increases the number of examples from 9 (in 1D) to 27 (in 2D)
 - Choosing to maintain the number of examples results in a 2D scatter plot that is very sparse

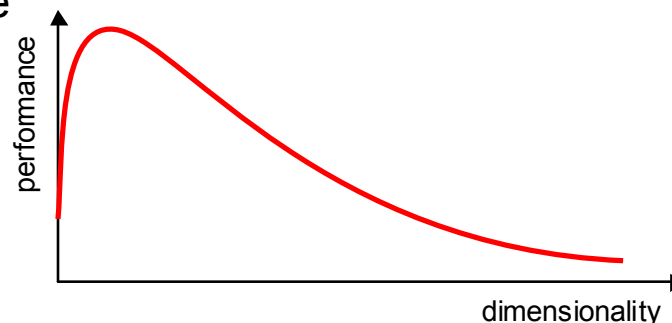


- Moving to three features makes the problem worse:
 - The number of bins grows to $3^3=27$
 - For the same density of examples the number of needed examples becomes 81
 - For the same number of examples, well, the 3D scatter plot is almost empty



The curse of dimensionality (3)

- **Obviously, our approach to divide the sample space into equally spaced bins was quite inefficient**
 - There are other approaches that are much less susceptible to the curse of dimensionality, **but the problem still exists**
- **How do we beat the curse of dimensionality?**
 - By incorporating prior knowledge
 - By providing increasing smoothness of the target function
 - By reducing the dimensionality
- **In practice, the curse of dimensionality means that, for a given sample size, there is a maximum number of features above which the performance of our classifier will degrade rather than improve**
 - In most cases, the additional information that is lost by discarding some features is (more than) compensated by a more accurate mapping in the lower-dimensional space



The curse of dimensionality (4)

■ There are many implications of the curse of dimensionality

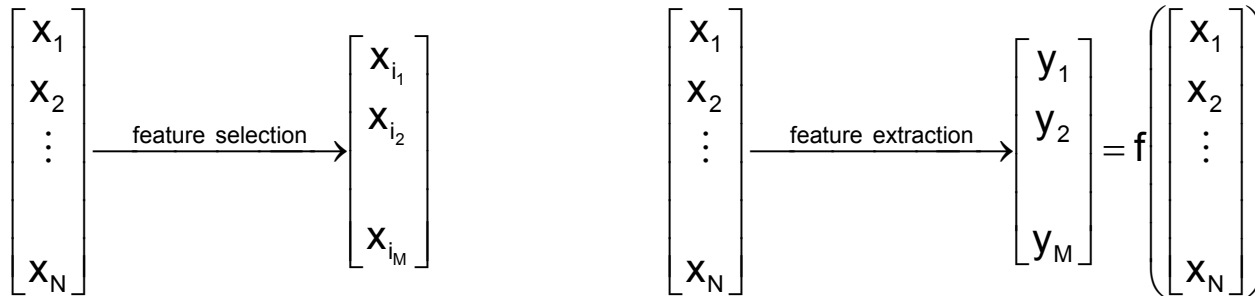
- Exponential growth in the number of examples required to maintain a given sampling density
 - For a density of N examples/bin and D dimensions, the total number of examples is N^D
- Exponential growth in the complexity of the target function (a density estimate) with increasing dimensionality
 - *“A function defined in high-dimensional space is likely to be much more complex than a function defined in a lower-dimensional space, and those complications are harder to discern” –Friedman*
 - This means that, in order to learn it well, a more complex target function requires denser sample points!
- What to do if it ain't Gaussian?
 - For one dimension a large number of density functions can be found in textbooks, but for high-dimensions only the multivariate Gaussian density is available. Moreover, for larger values of D the Gaussian density can only be handled in a simplified form!
- Humans have an extraordinary capacity to discern patterns and clusters in 1, 2 and 3-dimensions, but these capabilities degrade drastically for 4 or higher dimensions



Dimensionality reduction (1)

■ Two approaches are available to perform dimensionality reduction

- **Feature extraction:** creating a subset of new features by combinations of the existing features
- **Feature selection:** choosing a subset of all the features (the ones more informative)



■ The problem of feature extraction can be stated as

- Given a feature space $\mathbf{x}_i \in \mathbf{R}^N$ find a mapping $\mathbf{y} = \mathbf{f}(\mathbf{x}): \mathbf{R}^N \rightarrow \mathbf{R}^M$ with $M < N$ such that the transformed feature vector $\mathbf{y}_i \in \mathbf{R}^M$ preserves (most of) the information or structure in \mathbf{R}^N .
- An **optimal** mapping $\mathbf{y} = \mathbf{f}(\mathbf{x})$ will be one that results in **no increase in the minimum probability of error**
 - This is, a Bayes decision rule applied to the initial space \mathbf{R}^N and to the reduced space \mathbf{R}^M yield the same classification rate



Dimensionality reduction (2)

- In general, the optimal mapping $y=f(x)$ will be a non-linear function
 - However, there is no systematic way to generate non-linear transforms
 - The selection of a particular subset of transforms is problem dependent
 - For this reason, feature extraction is commonly limited to linear transforms: $y=Wx$
 - This is, y is a linear projection of x
 - NOTE: When the mapping is a non-linear function, the reduced space is called a **manifold**

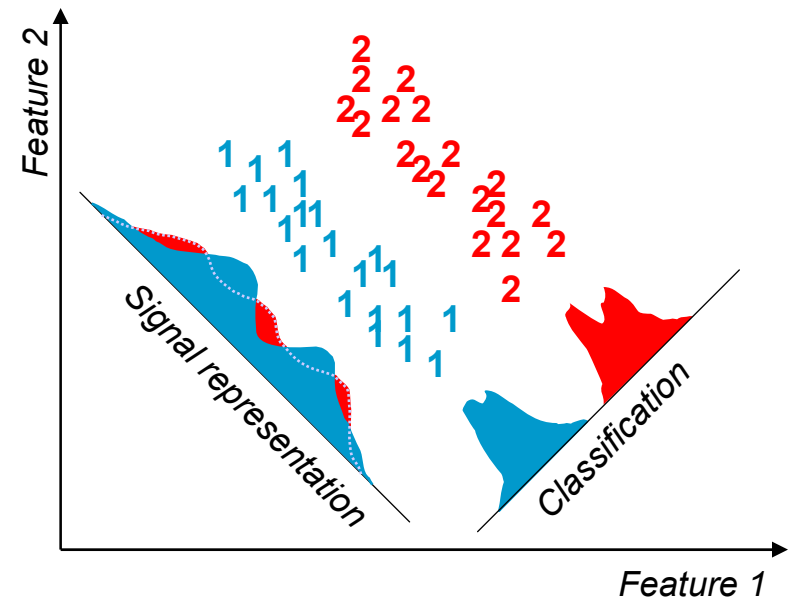
$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} \xrightarrow{\text{linear feature extraction}} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_M \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1N} \\ w_{21} & w_{22} & \cdots & w_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ w_{M1} & w_{M2} & & w_{MN} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}$$

- We will focus on linear feature extraction for now, and revisit non-linear techniques when we cover multi-layer perceptrons



Signal representation versus classification

- The selection of the feature extraction mapping $y=f(x)$ is guided by an objective function that we seek to maximize (or minimize)
- Depending on the criteria used by the objective function, feature extraction techniques are grouped into two categories:
 - **Signal representation:** The goal of the feature extraction mapping is to represent the samples accurately in a lower-dimensional space
 - **Classification:** The goal of the feature extraction mapping is to enhance the class-discriminatory information in the lower-dimensional space
- Within the realm of linear feature extraction, two techniques are commonly used
 - Principal Components Analysis (PCA)
 - uses a signal representation criterion
 - Linear Discriminant Analysis (LDA)
 - uses a signal classification criterion



Principal Components Analysis, PCA (1)

- **The objective of PCA is to perform dimensionality reduction while preserving as much of the randomness (variance) in the high-dimensional space as possible**

- Let x be an N -dimensional random vector, represented as a linear combination of orthonormal basis vectors $[\varphi_1 | \varphi_2 | \dots | \varphi_N]$ as

$$x = \sum_{i=1}^N y_i \varphi_i \quad \text{where} \quad \varphi_i | \varphi_j = \begin{cases} 0 & i \neq j \\ 1 & i = j \end{cases}$$

- Suppose we choose to represent x with only M ($M < N$) of the basis vectors. We can do this by replacing the components $[y_{M+1}, \dots, y_N]^T$ with some pre-selected constants b_i

$$\hat{x}(M) = \sum_{i=1}^M y_i \varphi_i + \sum_{i=M+1}^N b_i \varphi_i$$

- The representation error is then

$$\Delta x(M) = x - \hat{x}(M) = \sum_{i=1}^N y_i \varphi_i - \left(\sum_{i=1}^M y_i \varphi_i - \sum_{i=M+1}^N b_i \varphi_i \right) = \sum_{i=M+1}^N (y_i - b_i) \varphi_i$$

- We can measure this representation error by the mean-squared magnitude of Δx
- Our goal is to find the basis vectors φ_i and constants b_i that minimize this mean-square error

$$\bar{\varepsilon}^2(M) = E[|\Delta x(M)|^2] = E \left[\sum_{i=M+1}^N \sum_{j=M+1}^N (y_i - b_i)(y_j - b_j) \varphi_i^T \varphi_j \right] = \sum_{i=M+1}^N E[(y_i - b_i)^2]$$



Principal Components Analysis, PCA (2)

- As we have done earlier in the course, the optimal values of b_i can be found by computing the partial derivative of the objective function and equating it to zero

$$\frac{\partial}{\partial b_i} E[(y_i - b_i)^2] = -2(E[y_i] - b_i) = 0 \Rightarrow b_i = E[y_i]$$

- Therefore, we will replace the discarded dimensions y_i 's by their expected value (an intuitive solution)
- The mean-square error can then be written as

$$\begin{aligned} \bar{\epsilon}^2(M) &= \sum_{i=M+1}^N E[(y_i - E[y_i])^2] = \sum_{i=M+1}^N E[(x\phi_i - E[x\phi_i])^T (x\phi_i - E[x\phi_i])] \\ &= \sum_{i=M+1}^N \phi_i^T E[(x - E[x])(x - E[x])^T] \phi_i = \sum_{i=M+1}^N \phi_i^T \Sigma_x \phi_i \end{aligned}$$

- where Σ_x is the covariance matrix of x
- We seek to find the solution that minimizes this expression subject to the orthonormality constraint, which we incorporate into the expression using a set of Lagrange multipliers λ_i

$$\bar{\epsilon}^2(M) = \sum_{i=M+1}^N \phi_i^T \Sigma_x \phi_i + \sum_{i=M+1}^N \lambda_i (1 - \phi_i^T \phi_i)$$

- Computing the partial derivative with respect to the basis vectors

$$\frac{\partial}{\partial \phi_i} \bar{\epsilon}^2(M) = \frac{\partial}{\partial \phi_i} \left[\sum_{i=M+1}^N \phi_i^T \Sigma_x \phi_i + \sum_{i=M+1}^N \lambda_i (1 - \phi_i^T \phi_i) \right] = 2(\Sigma_x \phi_i - \lambda_i \phi_i) = 0 \Rightarrow \Sigma_x \phi_i = \lambda_i \phi_i$$

$$\text{NOTE: } \frac{d}{dx} (x^T A x) = (A + A^T) x \quad \begin{matrix} \text{if } A \text{ is} \\ \text{symmetric} \end{matrix} = 2Ax$$

- So ϕ_i and λ_i are the eigenvectors and eigenvalues of the covariance matrix Σ_x



Principal Components Analysis, PCA (3)

- We can then express the sum-square error as

$$\bar{\varepsilon}^2(M) = \sum_{i=M+1}^N \varphi_i^T \Sigma_x \varphi_i = \sum_{i=M+1}^N \varphi_i^T \lambda_i \varphi_i = \sum_{i=M+1}^N \lambda_i$$

- In order to minimize this measure, λ_i will have to be smallest eigenvalues
 - Therefore, to represent x with minimum sum-square error, we will choose the eigenvectors φ_i corresponding to the largest eigenvalues λ_i .

PCA dimensionality reduction

The optimal* approximation of a random vector $x \in \mathbb{R}^N$ by a linear combination of M ($M < N$) independent vectors is obtained by projecting the random vector x onto the eigenvectors φ_i corresponding to the largest eigenvalues λ_i of the covariance matrix Σ_x

*optimality is defined as the minimum of the sum-square magnitude of the approximation error



Principal Components Analysis, PCA (4)

■ NOTES

- Since PCA uses the eigenvectors of the covariance matrix Σ_x , it is able to find the independent axes of the data under the unimodal Gaussian assumption
 - For non-Gaussian or multi-modal Gaussian data, PCA simply de-correlates the axes
- The main limitation of PCA is that it does not consider class separability since it does not take into account the class label of the feature vector
 - PCA simply performs a coordinate rotation that aligns the transformed axes with the directions of maximum variance
 - **There is no guarantee that the directions of maximum variance will contain good features for discrimination**

■ Historical remarks

- Principal Components Analysis is the oldest technique in multivariate analysis
- PCA is also known as the Karhunen-Loève transform (communication theory)
- PCA was first introduced by Pearson in 1901, and it experienced several modifications until it was generalized by Loève in 1963

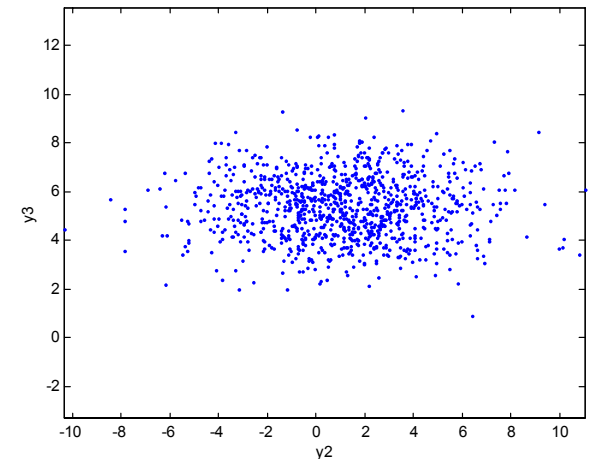
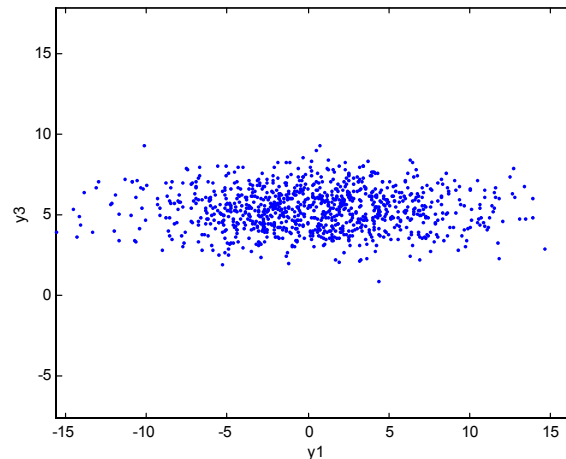
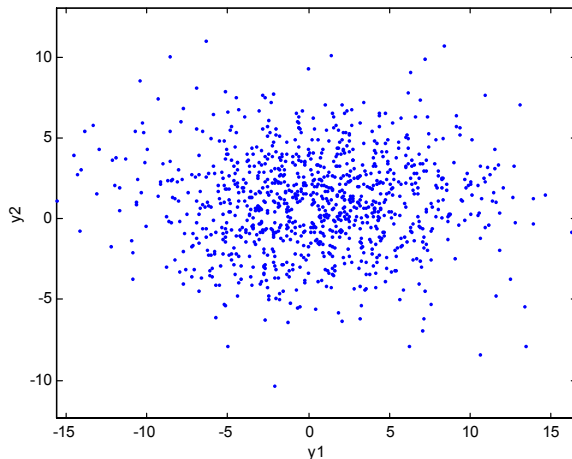
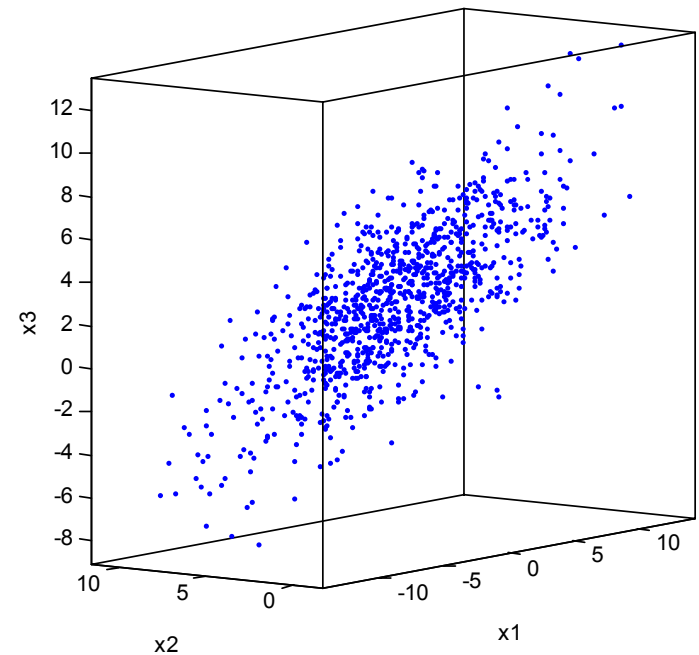


PCA example (1)

- In this example we have a three-dimensional Gaussian distribution with the following parameters

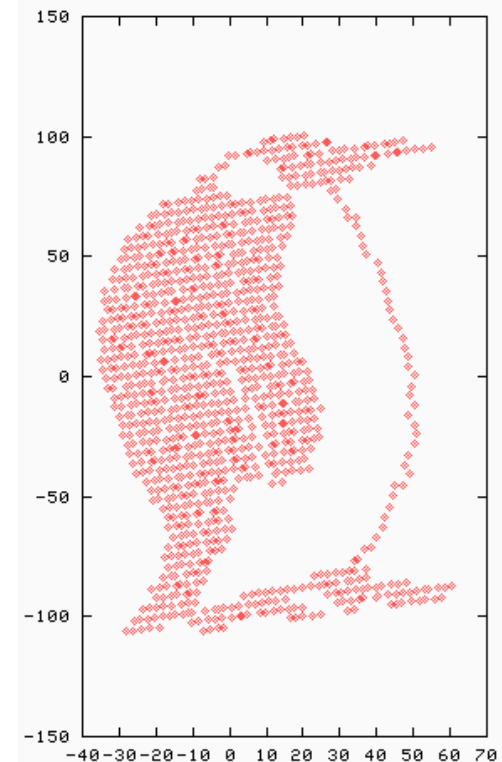
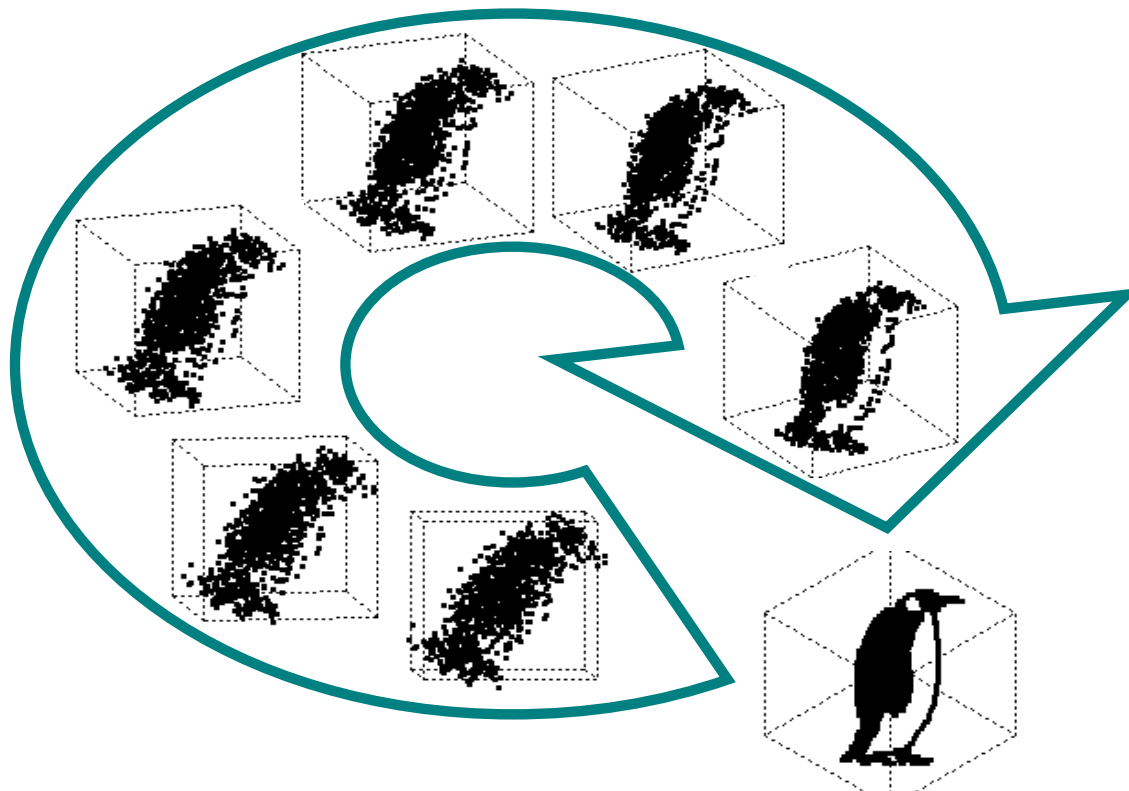
$$\mu = [0 \ 5 \ 2]^T \text{ and } \Sigma = \begin{bmatrix} 25 & -1 & 7 \\ -1 & 4 & -4 \\ 7 & -4 & 10 \end{bmatrix}$$

- The three pairs of principal component projections are shown below
 - Notice that the first projection has the largest variance, followed by the second projection
 - Also notice that the PCA projections de-correlates the axis (we knew this since Lecture 3, though)



PCA example (2)

- **This example shows a projection of a three-dimensional data set into two dimensions**
 - Initially, except for the elongation of the cloud, there is no apparent structure in the set of points
 - Choosing an appropriate rotation allows us to unveil the underlying structure. (You can think of this rotation as "walking around" the three-dimensional set, looking for the best viewpoint)
- **PCA can help find such underlying structure. It selects a rotation such that most of the variability within the data set is represented in the first few dimensions of the rotated data**
 - In our three-dimensional case, this may seem of little use
 - However, when the data is highly multidimensional (10's of dimensions), this analysis is quite powerful



PCA example (3)

■ Compute the principal components for the following two-dimensional dataset

- $X=(x_1, x_2)=\{(1,2), (3,3), (3,5), (5,4), (5,6), (6,5), (8,7), (9,8)\}$
 - Let's first plot the data to get an idea of which solution we should expect

■ SOLUTION (by hand)

- The (biased) covariance estimate of the data is:

$$\Sigma_x = \begin{bmatrix} 6.25 & 4.25 \\ 4.25 & 3.5 \end{bmatrix}$$

- The eigenvalues are the zeros of the characteristic equation

$$\Sigma_x v = \lambda v \Rightarrow |\Sigma_x - \lambda I| = 0 \Rightarrow \begin{vmatrix} 6.25 - \lambda & 4.25 \\ 4.25 & 3.5 - \lambda \end{vmatrix} = 0 \Rightarrow \lambda_1 = 9.34; \lambda_2 = 0.41;$$

- The eigenvectors are the solutions of the system

$$\begin{bmatrix} 6.25 & 4.25 \\ 4.25 & 3.5 \end{bmatrix} \begin{bmatrix} v_{11} \\ v_{12} \end{bmatrix} = \begin{bmatrix} \lambda_1 v_{11} \\ \lambda_1 v_{12} \end{bmatrix} \Rightarrow \begin{bmatrix} v_{11} \\ v_{12} \end{bmatrix} = \begin{bmatrix} 0.81 \\ 0.59 \end{bmatrix}$$
$$\begin{bmatrix} 6.25 & 4.25 \\ 4.25 & 3.5 \end{bmatrix} \begin{bmatrix} v_{21} \\ v_{22} \end{bmatrix} = \begin{bmatrix} \lambda_2 v_{21} \\ \lambda_2 v_{22} \end{bmatrix} \Rightarrow \begin{bmatrix} v_{21} \\ v_{22} \end{bmatrix} = \begin{bmatrix} -0.59 \\ 0.81 \end{bmatrix}$$

- HINT: To solve each system manually, first assume that one of the variables is equal to one (i.e. $v_{i1}=1$), then find the other one and finally normalize the vector to make it unit-length

