THE UNIVERSITY OF SYDNEY

# Machine Learning and Data Mining (COMP 5318)

## Latent Variable Models

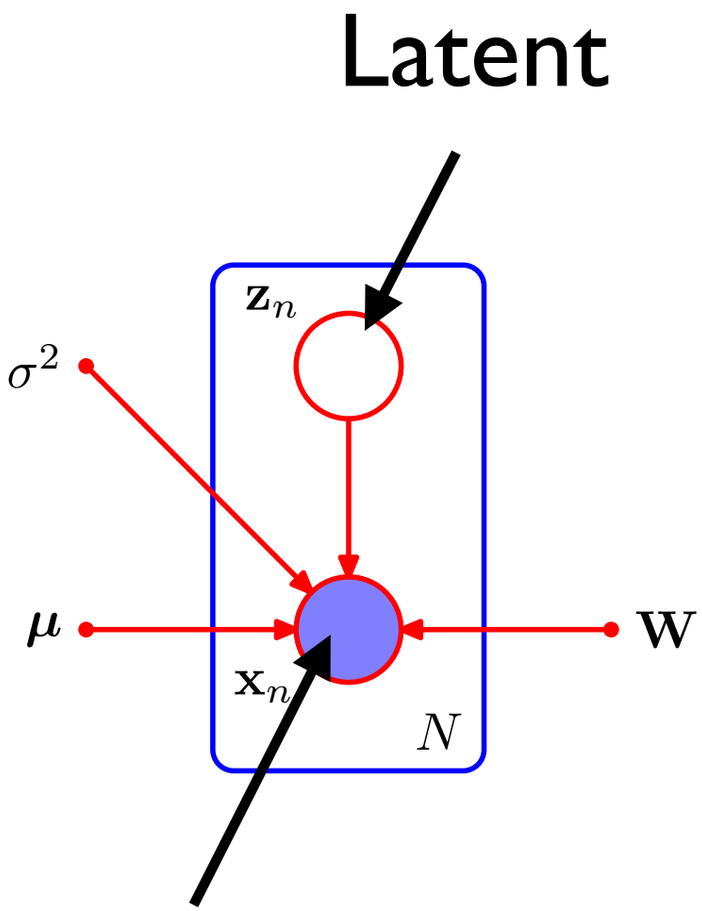
Fabio Ramos
Roman Marchant

# Announcements

THE UNIVERSITY OF SYDNEY

Assignment 2 will be available at the end of today's lecture.

# What are latent variable models?

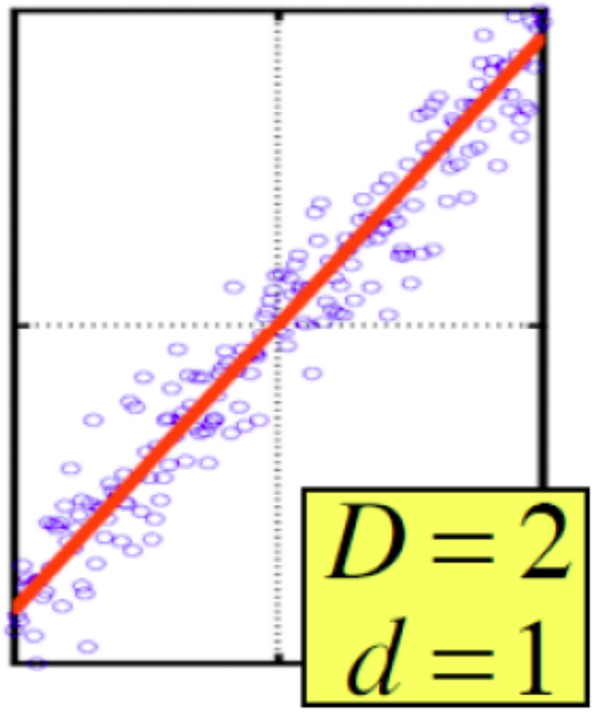


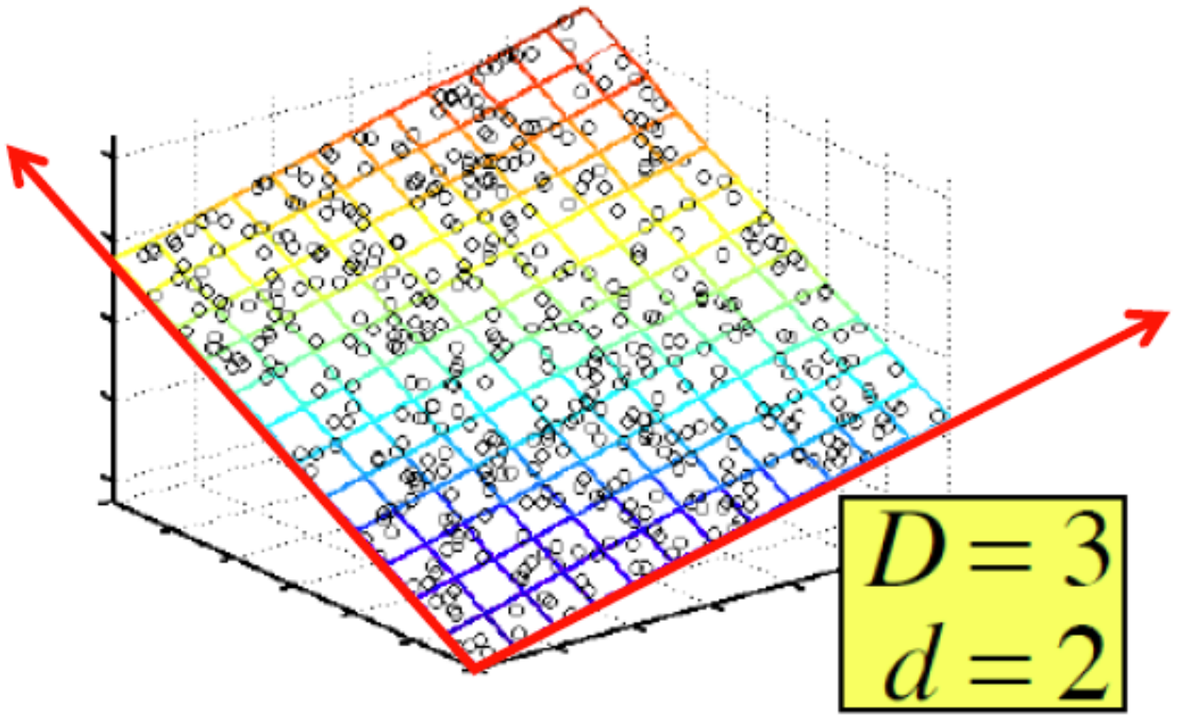

Essentially: models where some variables are not observed

Example: what is the low dimensional representation for a given dataset? In this lecture, we will use latent variable models for dimensionality reduction

They can also be used for clustering, recommender systems and others

# Dimensionality Reduction

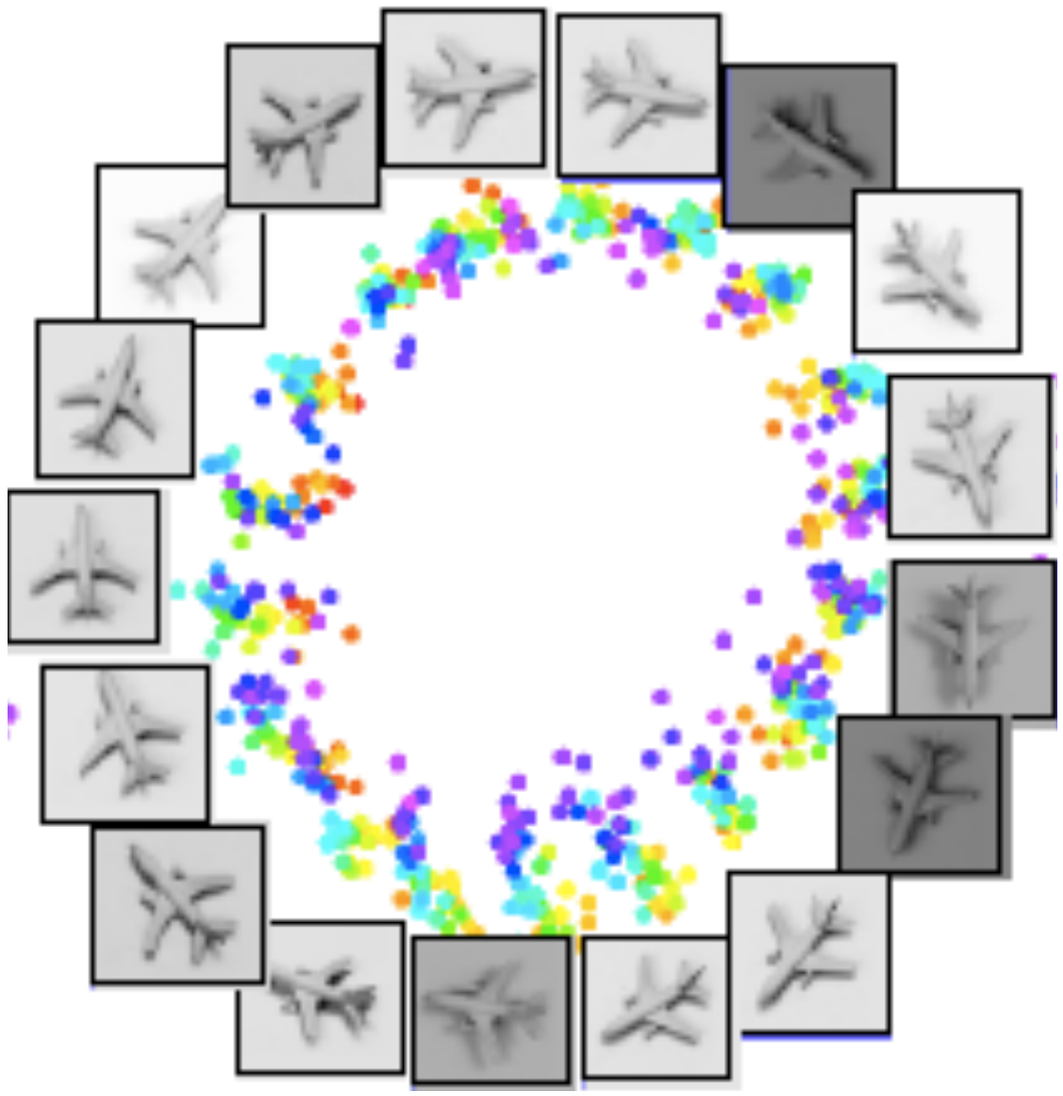



**Assumption:** Data lies on or near a low *d*-dimensional subspace

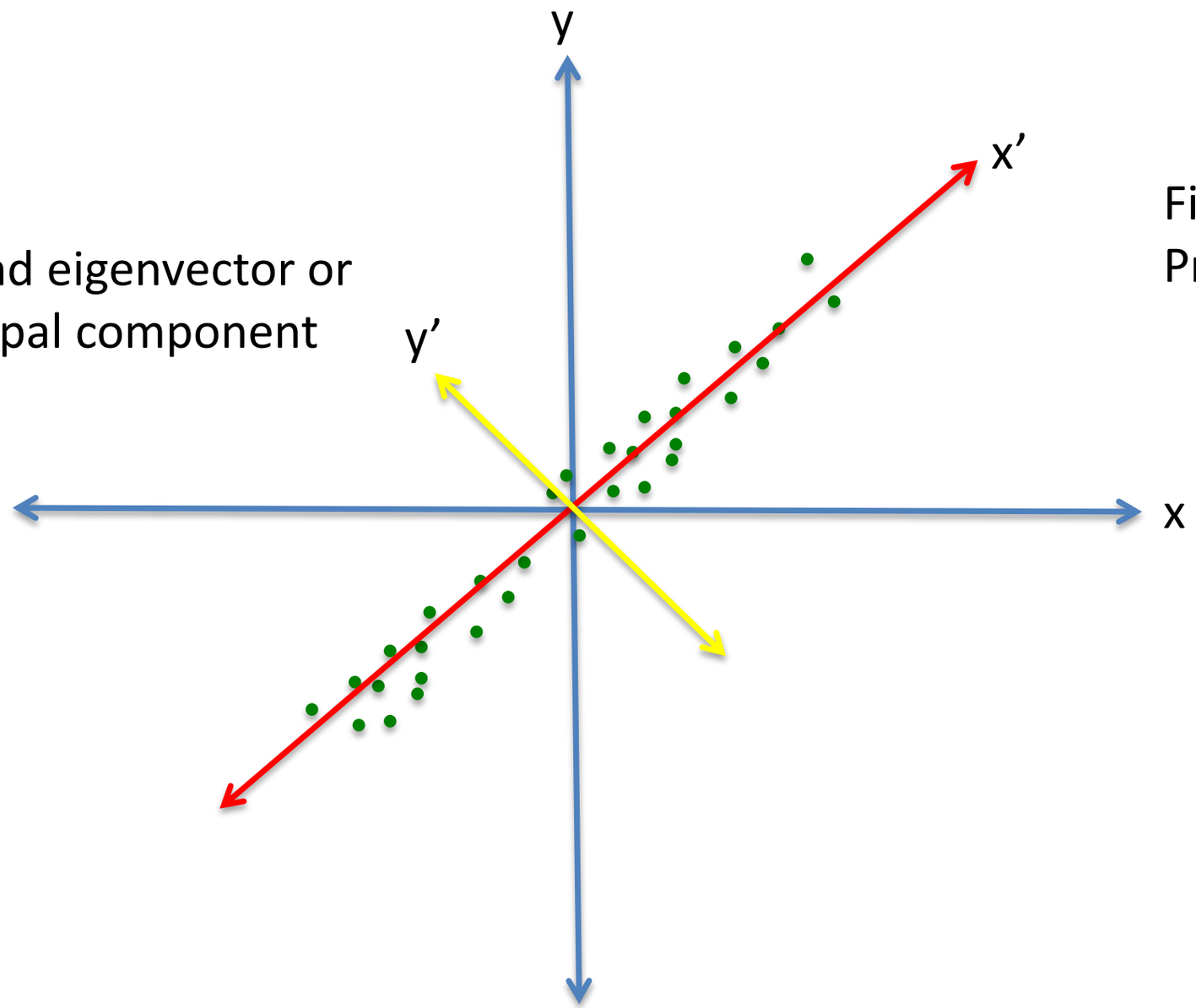
**Axes of this subspace are effective representation of the data**

# Dimensionality Reduction



How to find complicated data structures?

# Example: compression of X

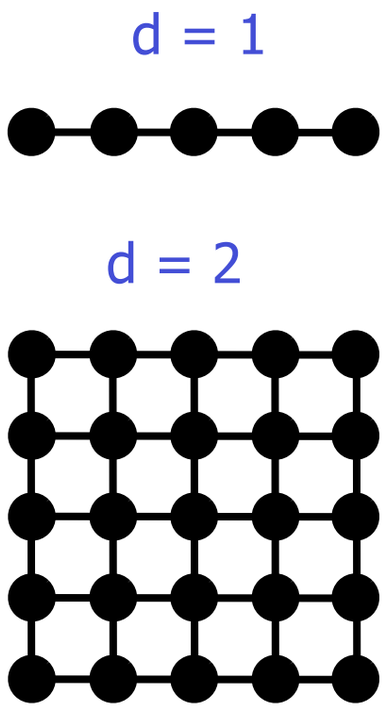


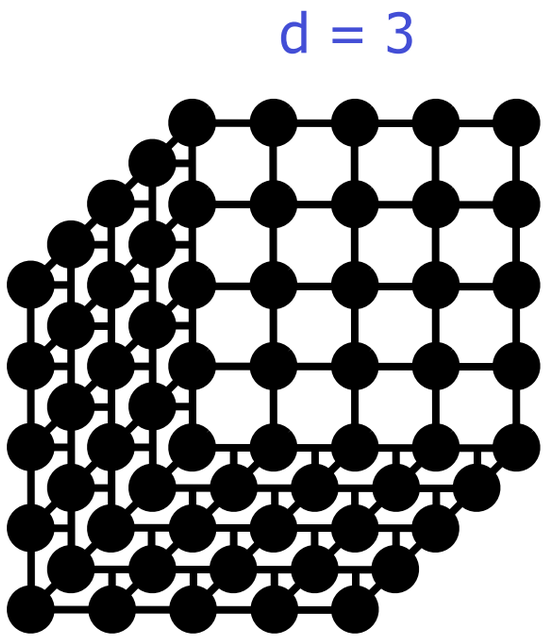

Second eigenvector or
Principal component

First eigenvector or
Principal component

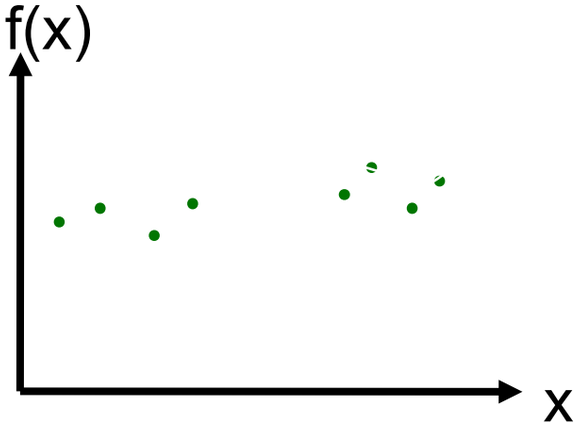
# The empty space phenomenon

How to “fill” a (compact) space regularly, for a given precision




Number of points on a grid increases exponentially with d



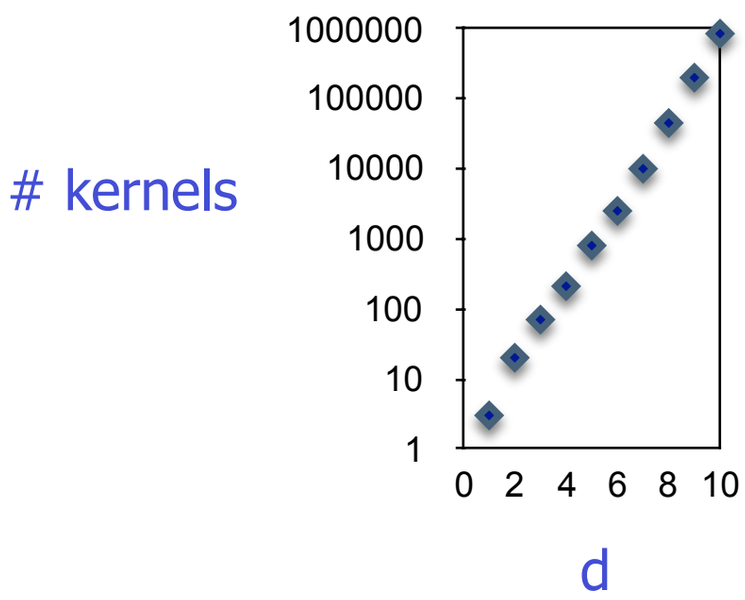

In high dimension d:
- never enough data
- never sure to interpolate

# The curse of dimensionality

THE UNIVERSITY OF SYDNEY

All surprising, or unexpected phenomena, encountered in high-dimensional spaces

Historical example: Silverman (1986)
Number of Gaussian kernels necessary to approximate a (Gaussian) distribution in dimension d, at a given precision




- clear exponential increase
- already high for small d !

# Surprising facts: volume of the sphere

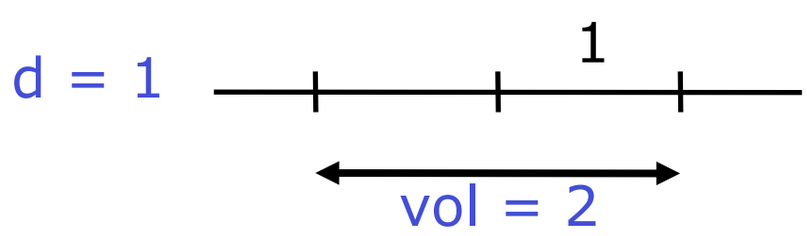

Some facts that are intuitive in low d are less intuitive in high d!

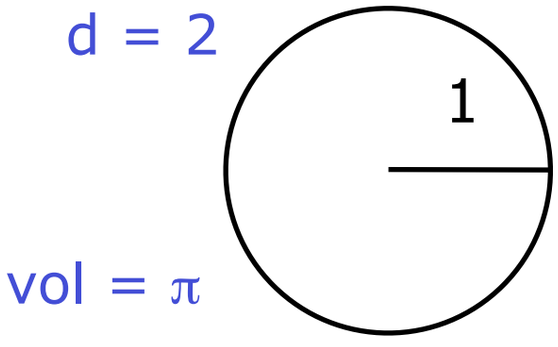
Example: volume of the sphere

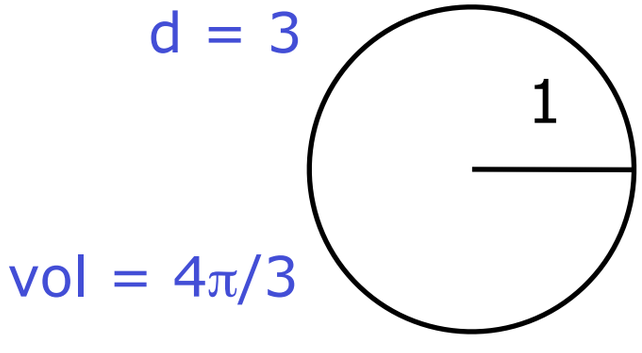
Constant radius (=1)

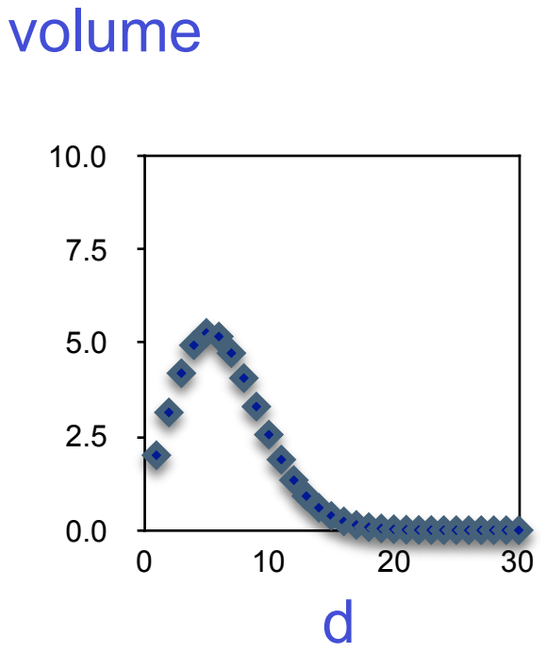
Increasing space dimension

$$V(d) = \frac{\pi^{d/2}}{\Gamma(d/2+1)} r^d$$

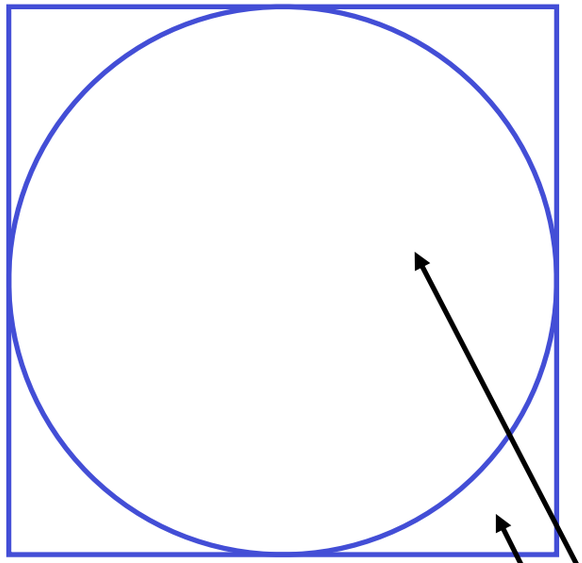


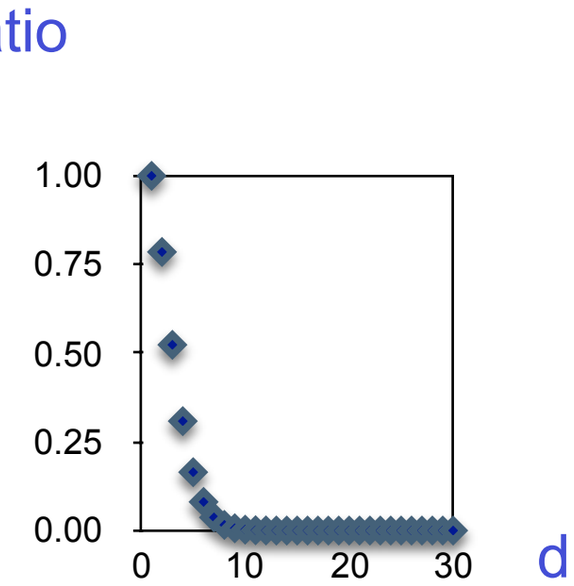



# Ratio volume sphere / cube

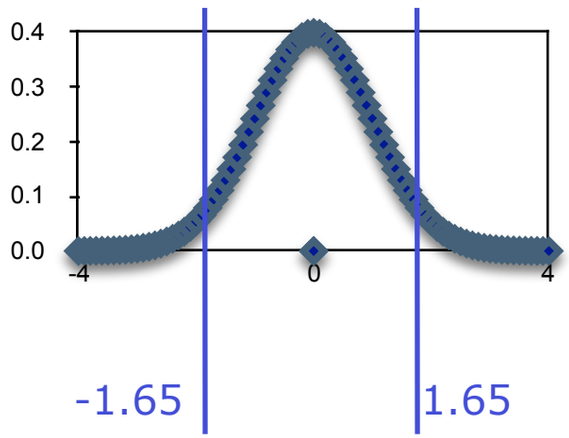


in HD, all points are here
and not here



# High-dimensional Gaussians

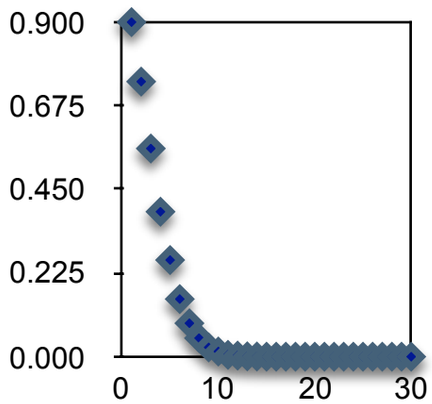

% points whose distance to centre is < 1.65




d=1

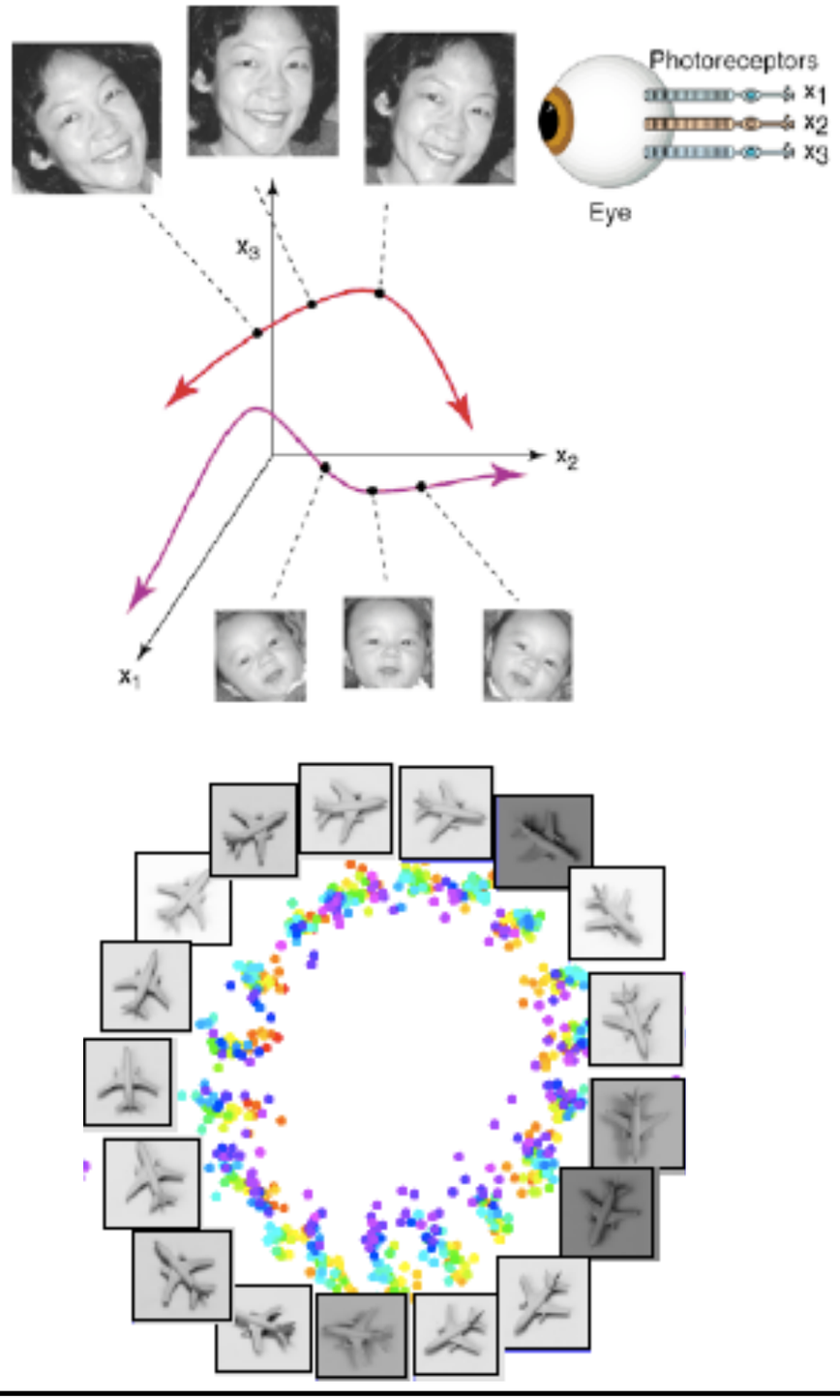
% = 0.9

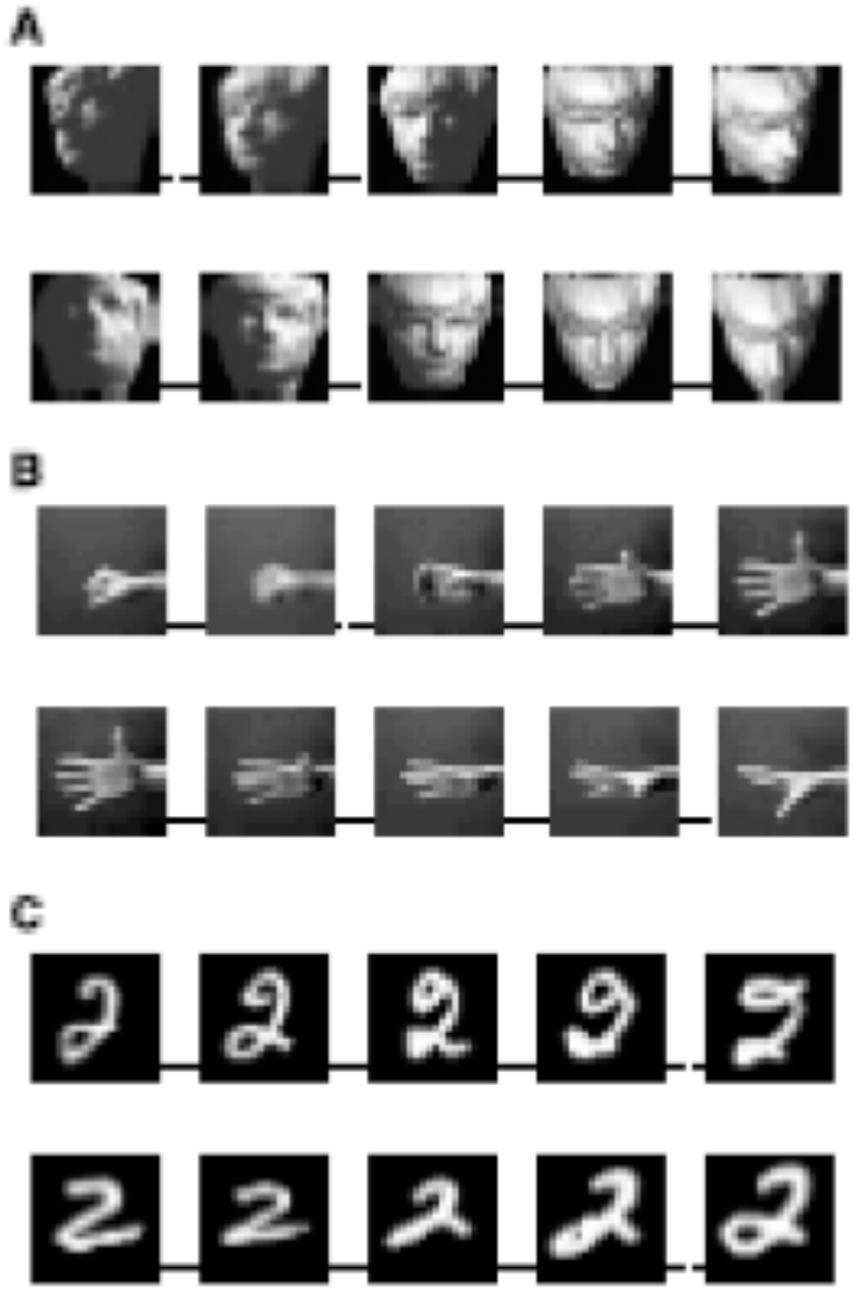
% points




The Gaussian is not local anymore!

# But there is hope!



THE UNIVERSITY OF SYDNEY

# Principal Component Analysis

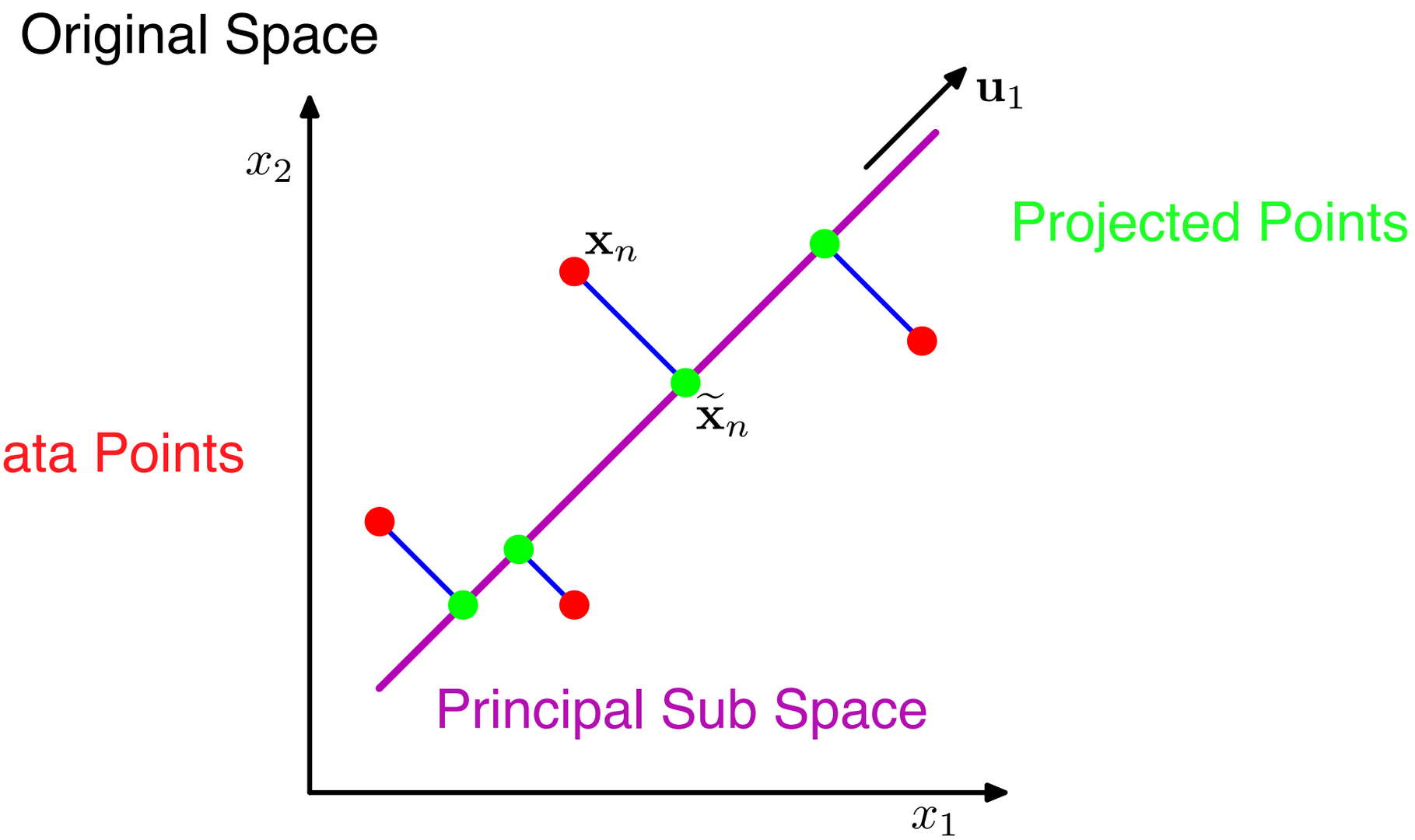
C. Bishop, *Pattern Recognition and Machine Learning,* Chapter 12: Continuous Latent Variables
Springer New York, 2006

# PCA

THE UNIVERSITY OF SYDNEY

*The central idea of principal component analysis (PCA) is to reduce the dimensionality of a data set consisting of a large number of interrelated variables, while retaining as much as possible of the variation present in the data set. This is achieved by transforming to a new set of variables, the principal components (PCs), which are uncorrelated, and which are ordered so that the first* few *retain most of the variation present in* all *of the original variables.*
*[Jolliffe, Pricipal Component Analysis, $2^{nd}$ edition]*

# Geometric intuition



# PCA Formulation

Let $\{\mathbf{x}_n\}_{n=1,\ldots,N}$ be a dataset of observations with dimensionality $D$

Goal, to represent the data in a lower dimensional space $M$ (known).

$$\bar{\mathbf{x}} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n$$

$$X = [(\mathbf{x}_1 - \bar{\mathbf{x}}) \quad (\mathbf{x}_2 - \bar{\mathbf{x}}) \quad (\mathbf{x}_3 - \bar{\mathbf{x}}) \quad ...]$$

Translating the coordinate system to the location of the mean.

# PCA Formulation

$$\mathbf{S} = \frac{1}{N} X X^{\mathrm{T}}$$

$$= \frac{1}{N} \begin{bmatrix} (\mathbf{x}_1 - \bar{\mathbf{x}}) & (\mathbf{x}_2 - \bar{\mathbf{x}}) & (\mathbf{x}_3 - \bar{\mathbf{x}}) & \dots \end{bmatrix} \begin{bmatrix} (\mathbf{x}_1 - \bar{\mathbf{x}})^{\mathrm{T}} \\ (\mathbf{x}_2 - \bar{\mathbf{x}})^{\mathrm{T}} \\ (\mathbf{x}_3 - \bar{\mathbf{x}})^{\mathrm{T}} \\ \dots \end{bmatrix}$$

$$\mathbf{S} = \frac{1}{N} \sum_{n=1}^{N} (\mathbf{x}_n - \bar{\mathbf{x}})(\mathbf{x}_n - \bar{\mathbf{x}})^{\mathrm{T}}$$

- $\mathbf{S}$ is the covariance matrix. (scatter matrix)
- $\mathbf{S}$ is the square and symmetric
- $\mathbf{S}$ can be very large (in vision, N is the number of pixels)

# PCA Derivation

Let us assume, for now, that $M = 1$

$\mathbf{u}_1$ is the direction of this sub-space

$\mathbf{u}_1$ is a $D$ dimensional vector.

$\mathbf{u}_1$ is a unit vector, i.e. $\mathbf{u}_1^{\mathrm{T}}\mathbf{u}_1 = 1$

If we project the data point $\mathbf{x}_n$ into a scalar value $\mathbf{u}_1\mathbf{x}_n$ .

The mean of the projected data becomes

$$\mathbf{u}_1^{\mathrm{T}}\bar{\mathbf{x}} = \frac{\mathbf{u}_1^{\mathrm{T}}}{N}\sum_{n=1}^{N}\mathbf{x}_n$$

The variance becomes

$$\frac{1}{N}\sum_{n=1}^{N}\{\mathbf{u}_1^{\mathrm{T}}\mathbf{x}_n - \mathbf{u}_1^{\mathrm{T}}\bar{\mathbf{x}}\}^2 = \mathbf{u}_1^{\mathrm{T}}\mathbf{S}\mathbf{u}_1$$

# PCA Derivation

Maximise the projected variance $\mathbf{u}_1^{\mathrm{T}}\mathbf{S}\mathbf{u}_1$ with respect to $\mathbf{u}_1$

* problem: $\|\mathbf{u}_1\| \to \infty$ remember:

$\mathbf{u}_1$ is a unit vector, i.e. $\mathbf{u}_1^{\mathrm{T}}\mathbf{u}_1 = 1$

Introduce Lagrange multiplier $\lambda_1$

$$\mathbf{u}_1^{\mathrm{T}}\mathbf{S}\mathbf{u}_1 + \lambda_1\left(1 - \mathbf{u}_1^{\mathrm{T}}\mathbf{u}_1\right)$$

Derivative equal to zero: $\mathbf{u}_1$ Must be an eigen vector of S

Maximum variance when $\mathbf{u}_1$ is the eigenvector with the larges eigenvalue $\lambda_1$

# PCA Theorem

THE UNIVERSITY OF SYDNEY

Theorem

Each $\mathbf{x}_n$ can be written as: $\mathbf{x}_n = \bar{\mathbf{x}} + \sum_{i=1}^{M} z_{ni}\mathbf{u}_i$

Where $\mathbf{u}_i$ are the $M$ eigenvectors of $\mathbf{S}$ with non-zero eigenvalues.

The scalars $z_{ni}$ are the coordinates of $\mathbf{x}_n$ in the principal component space.

# PCA to compress data

Have we achieved compression of the data?

If we consider $M = D$ then there is no compression,

This case results in a realignment of the data w.r.t. principal components.

However! Correlated data will result in $z_{ni}$ being zero or close to zero.

i.e. $\{\mathbf{x}_n\}_{n=1,\ldots,N}$ lies in a lower-dimensional subspace.

# PCA to compress data


Achieve compression: $M < D$

1. Sort the eigenvectors $\mathbf{u}_i$ according to their associated eigenvalue.

$$\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_M$$

2. Discard smaller eigen values, i.e. assuming $\lambda_i \approx 0 \;\; \forall i > k$

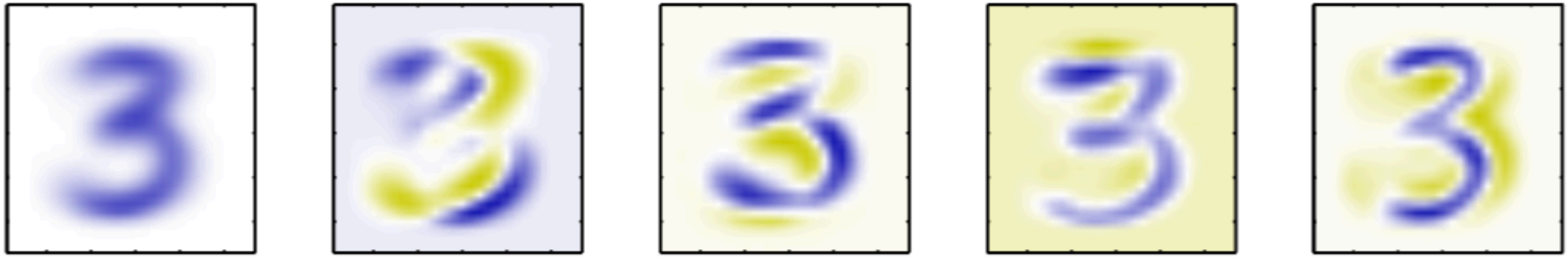
Then

$$\mathbf{x}_n \approx \bar{\mathbf{x}} + \sum_{i=1}^{K} z_{ni}\mathbf{u}_i$$

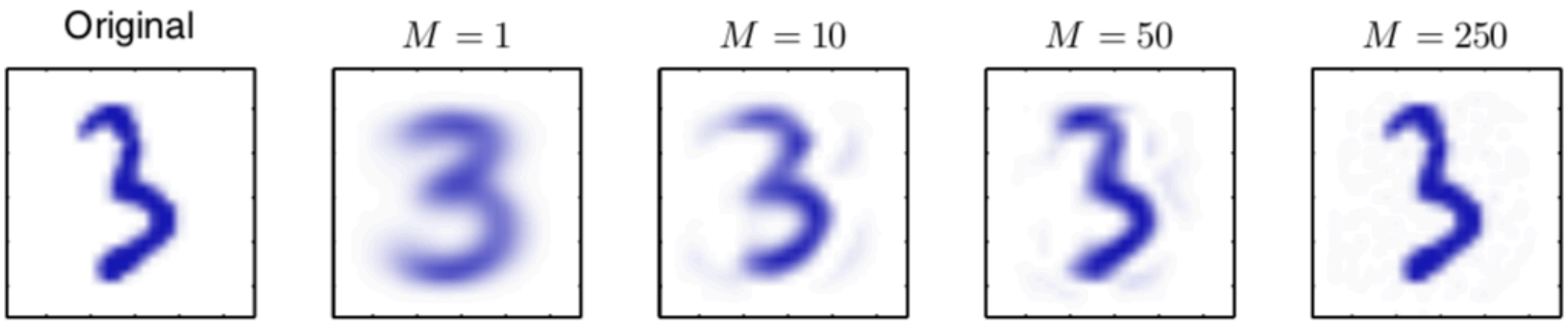
# Reconstruction of digits

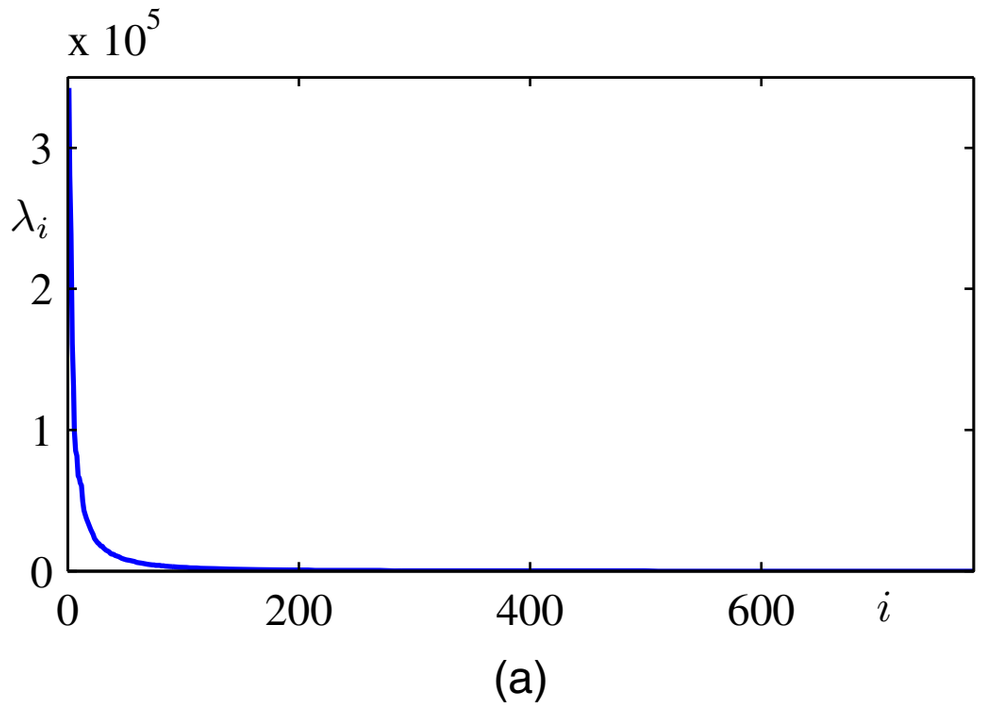

An eigen vector of the covariance matrix is a vector in the original $D$ dimensional space.

| Mean | $\lambda_1 = 3.4 \cdot 10^5$ | $\lambda_2 = 2.8 \cdot 10^5$ | $\lambda_3 = 2.4 \cdot 10^5$ | $\lambda_4 = 1.6 \cdot 10^5$ |
|---|---|---|---|---|



# Reconstruction of digits



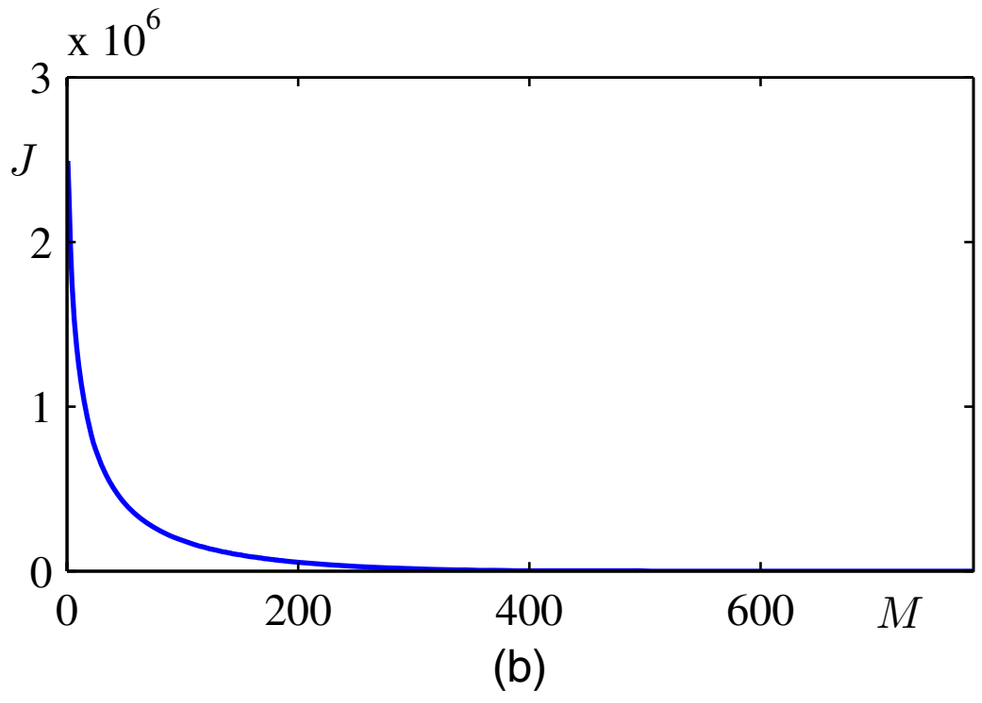




**Figure 12.4** (a) Plot of the eigenvalue spectrum for the off-line digits data set. (b) Plot of the sum of the discarded eigenvalues, which represents the sum-of-squares distortion $J$ introduced by projecting the data onto a principal component subspace of dimensionality $M$.

# PCA Algorithm

THE UNIVERSITY OF SYDNEY

Goal:

To project linearly while keeping the variance of the data

Covariance matrix $\mathbf{S}$ of the data

$$\mathbf{S} = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n - \bar{\mathbf{x}})(\mathbf{x}_n - \bar{\mathbf{x}})^{\mathrm{T}}$$

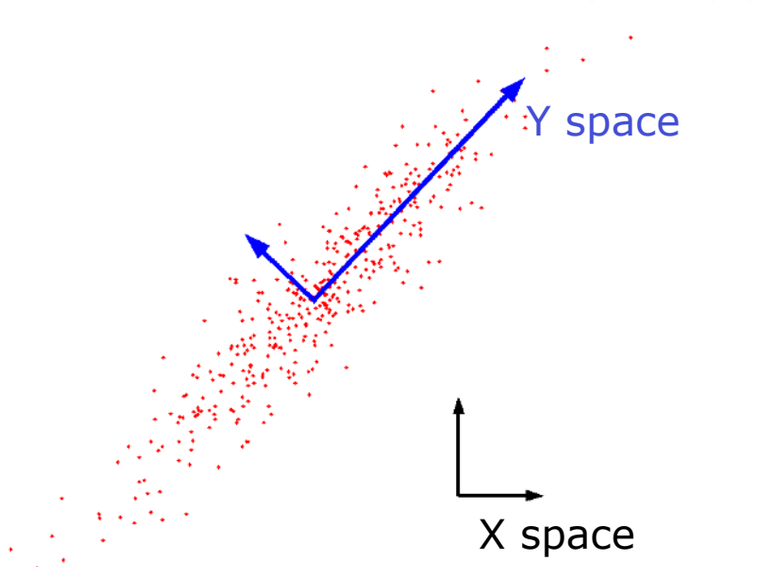
Eigenvectors and eigenvalues of $\mathbf{S}$

$\mathbf{u}_i$ = main directions

$\lambda_1$ = variance along each direction

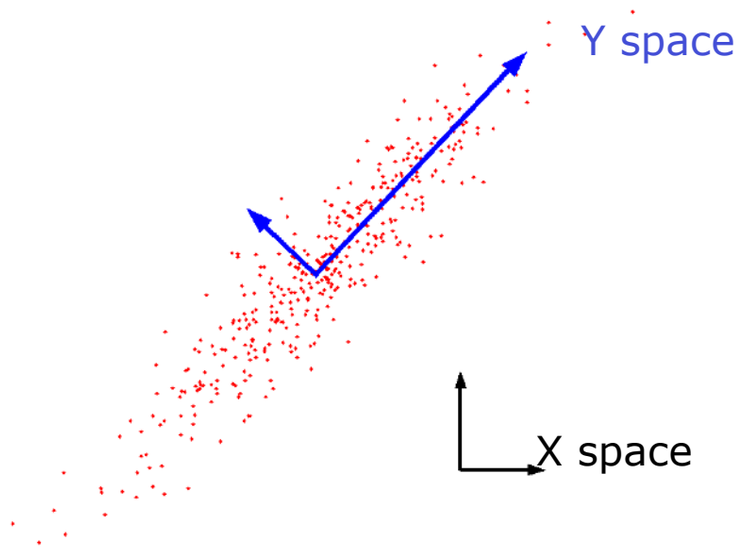
Projection & Reconstruction

$$\mathbf{x}_n \approx \bar{\mathbf{x}} + \sum_{i=1}^{K} z_{ni}\mathbf{u}_i$$



# Metric multidimensional scaling

Is it necessary to know the X-axes in order to find the Y-axes ones?




Of course not!
It should be possible to find the principal directions (the Y axes) only from the relative positions between data.
That is metric MDS…

# Metric multidimensional scaling


Suppose all we were given were distances or symmetric "dissimilarities" $\Delta_{ij}$.

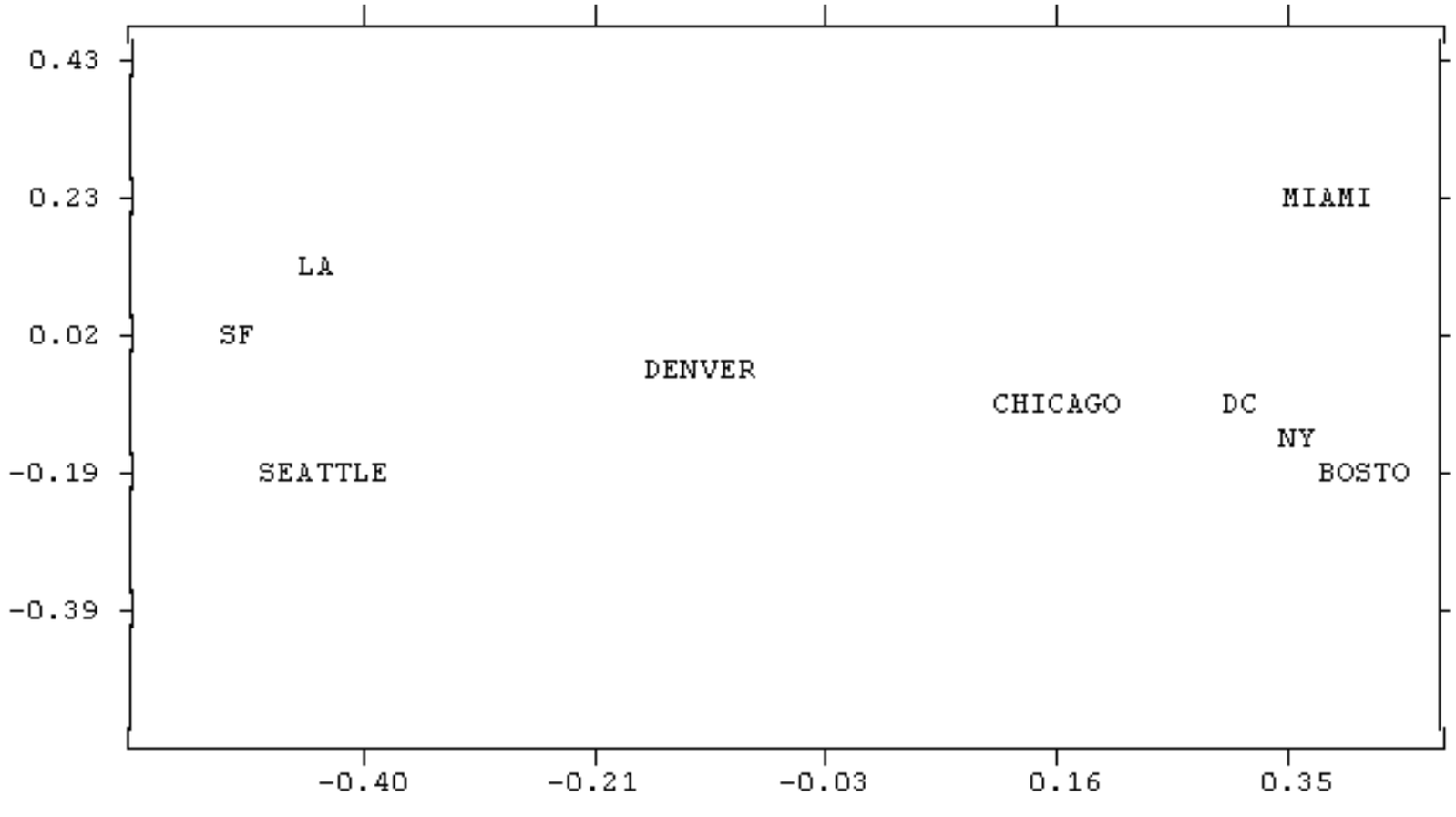
$$\Delta = \begin{bmatrix} 0 & \Delta_{12} & \Delta_{13} & \Delta_{14} \\ \Delta_{12} & 0 & \Delta_{23} & \Delta_{24} \\ \Delta_{13} & \Delta_{23} & 0 & \Delta_{34} \\ \Delta_{14} & \Delta_{24} & \Delta_{34} & 0 \end{bmatrix}$$

**Goal**: Find vectors $\mathbf{y}_i$ such that $\|\mathbf{y}_i - \mathbf{y}_j\| \approx \Delta_{ij}$.

This is called **Multidimensional Scaling (MDS)**.

# Example: From distances to a map


| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| | | BOST | NY | DC | MIAM | CHIC | SEAT | SF | LA | DENV |
| 1 | BOSTON | 0 | 206 | 429 | 1504 | 963 | 2976 | 3095 | 2979 | 1949 |
| 2 | NY | 206 | 0 | 233 | 1308 | 802 | 2815 | 2934 | 2786 | 1771 |
| 3 | DC | 429 | 233 | 0 | 1075 | 671 | 2684 | 2799 | 2631 | 1616 |
| 4 | MIAMI | 1504 | 1308 | 1075 | 0 | 1329 | 3273 | 3053 | 2687 | 2037 |
| 5 | CHICAGO | 963 | 802 | 671 | 1329 | 0 | 2013 | 2142 | 2054 | 996 |
| 6 | SEATTLE | 2976 | 2815 | 2684 | 3273 | 2013 | 0 | 808 | 1131 | 1307 |
| 7 | SF | 3095 | 2934 | 2799 | 3053 | 2142 | 808 | 0 | 379 | 1235 |
| 8 | LA | 2979 | 2786 | 2631 | 2687 | 2054 | 1131 | 379 | 0 | 1059 |
| 9 | DENVER | 1949 | 1771 | 1616 | 2037 | 996 | 1307 | 1235 | 1059 | 0 |



# Metric MDS - Algorithm

THE UNIVERSITY OF SYDNEY

Goal:

To project linearly while keeping the (N-1)*N/2 pairwise distances

Computation:

1. Matrix D of the squared distances

   $D = [ d^2_{i,j} ]$

2. Eigenvectors and eigenvalues of D

   $V_i$ = *coordinates* along the main directions

   $\lambda_i$ = variance along each direction

3. Projection

   $Y = \text{sqrt}( \text{diag}( \lambda_{1 \le i \le p} ) ) \, V_{1 \le i \le p}^T$

Result of PCA = result of metric MDS !

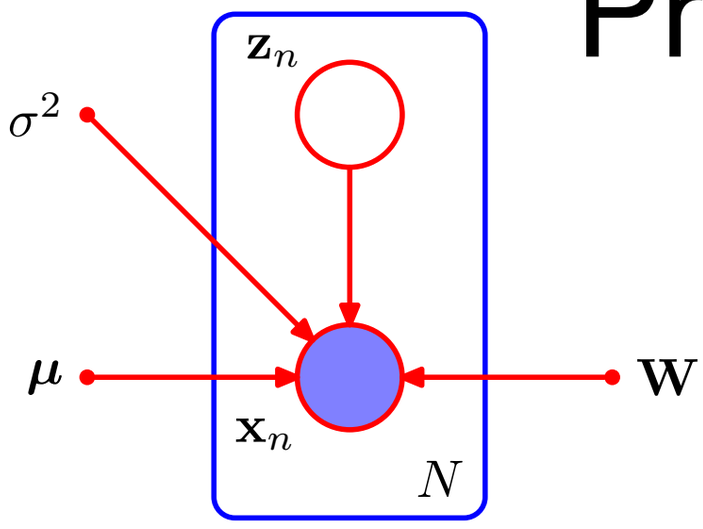
Only distances are needed i.e. more independent from representation.

# Probabilistic PCA

THE UNIVERSITY OF SYDNEY

- Probabilistic version assuming Gaussian data
- Solution can be found as a maximum likelihood optimisation
- Computationally efficient
- Generative model: it can be sampled from to generate examples in the low dimensional space.

# Probabilistic PCA




**Parameters:**

$\mu \rightarrow$ D-dimensional vector

$\mathbf{W} \rightarrow$ D x M matrix

$\sigma^2 \rightarrow$ scalar

Observed x $\rightarrow$ D-dimensional

Latent z $\rightarrow$ M-dimensional

Prior for z (latent variable)

$$p(\mathbf{z}) = \mathcal{N}(\mathbf{z}|\mathbf{0}, \mathbf{I}).$$

Conditional distribution for x

$$p(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}|\mathbf{W}\mathbf{z} + \boldsymbol{\mu}, \sigma^2\mathbf{I})$$

x and z are linearly related

$$\mathbf{x} = \mathbf{W}\mathbf{z} + \boldsymbol{\mu} + \boldsymbol{\epsilon}$$

D-dimensional noise variable

$$\epsilon \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$$

# Generative view of PPCA

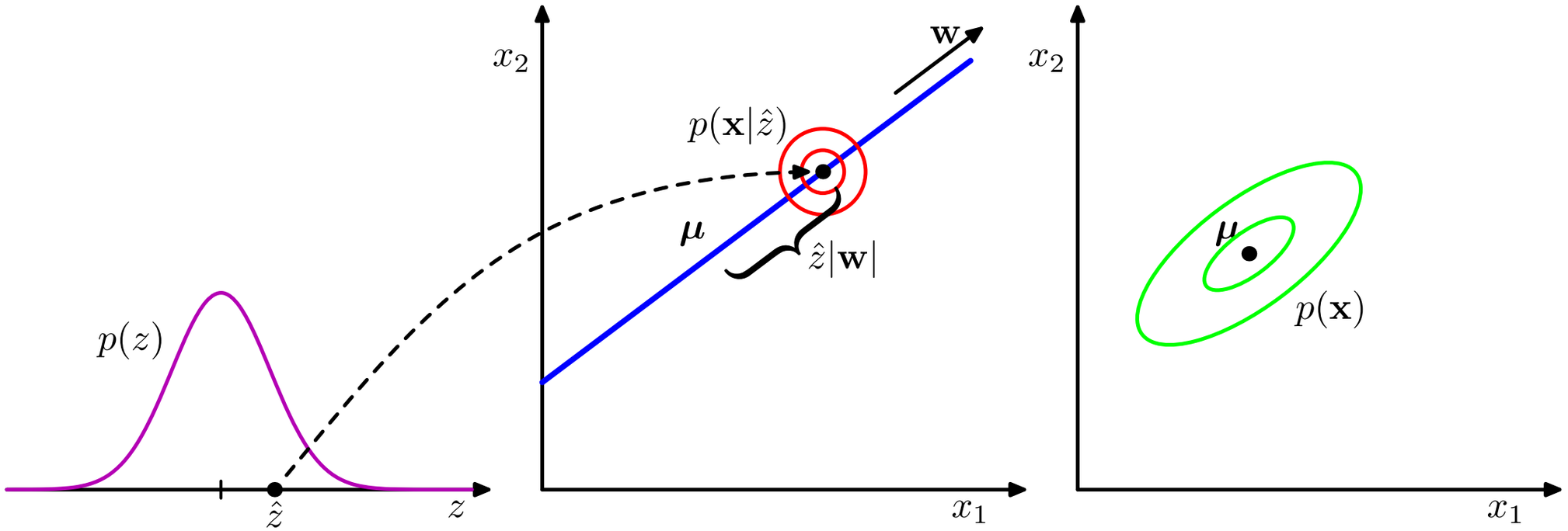
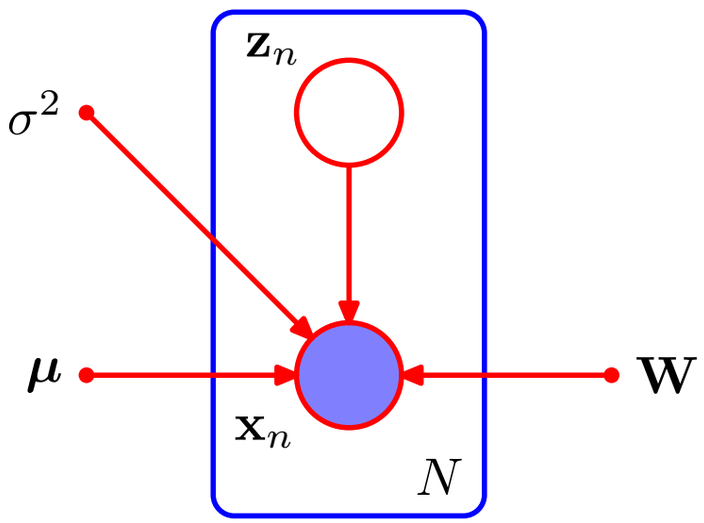



**Figure 12.9** An illustration of the generative view of the probabilistic PCA model for a two-dimensional data space and a one-dimensional latent space. An observed data point $\mathbf{x}$ is generated by first drawing a value $\widehat{z}$ for the latent variable from its prior distribution $p(z)$ and then drawing a value for $\mathbf{x}$ from an isotropic Gaussian distribution (illustrated by the red circles) having mean $\mathbf{w}\widehat{z} + \boldsymbol{\mu}$ and covariance $\sigma^2\mathbf{I}$. The green ellipses show the density contours for the marginal distribution $p(\mathbf{x})$.

Bishop's book, page 572

# Predictions with PPCA




We want to compute

$$p(\mathbf{z}|\mathbf{x}) = \frac{p(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{p(\mathbf{x})}$$

From Bayes' rule:

$$p(\mathbf{x}) = \int p(\mathbf{x}|\mathbf{z})p(\mathbf{z})\,\mathrm{d}\mathbf{z}.$$

This has closed form: $p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \mathbf{C})$

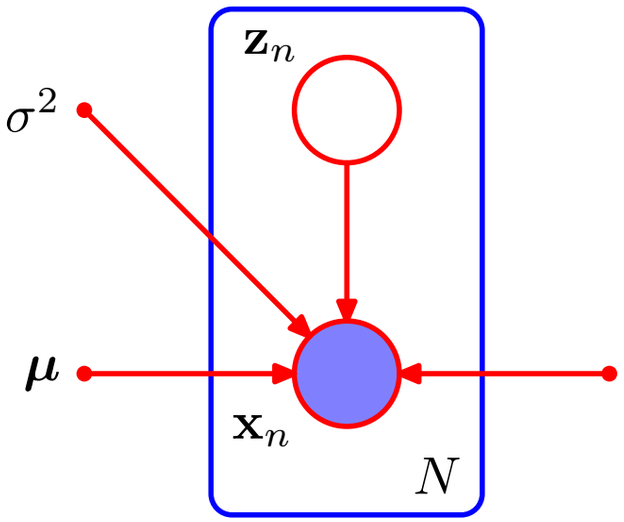
where $\mathbf{C} = \mathbf{W}\mathbf{W}^{\mathrm{T}} + \sigma^2\mathbf{I}.$

Finally,

$$p(\mathbf{z}|\mathbf{x}) = \mathcal{N}\left(\mathbf{z}|\mathbf{M}^{-1}\mathbf{W}^{\mathrm{T}}(\mathbf{x}-\boldsymbol{\mu}), \sigma^{-2}\mathbf{M}\right)$$

where $\mathbf{M} = \mathbf{W}^{\mathrm{T}}\mathbf{W} + \sigma^2\mathbf{I}.$

# Estimating parameters: Maximum likelihood solution




Given a dataset

$$\mathbf{X} = \{\mathbf{x}_n\}$$

The log-likelihood is

$$\ln p(\mathbf{X}|\boldsymbol{\mu}, \mathbf{W}, \sigma^2) = \sum_{n=1}^{N} \ln p(\mathbf{x}_n|\mathbf{W}, \boldsymbol{\mu}, \sigma^2)$$

$$= -\frac{ND}{2}\ln(2\pi) - \frac{N}{2}\ln|\mathbf{C}| - \frac{1}{2}\sum_{n=1}^{N}(\mathbf{x}_n - \boldsymbol{\mu})^{\mathrm{T}}\mathbf{C}^{-1}(\mathbf{x}_n - \boldsymbol{\mu})$$

Setting the derivatives to zero, the parameters can be computed in closed form

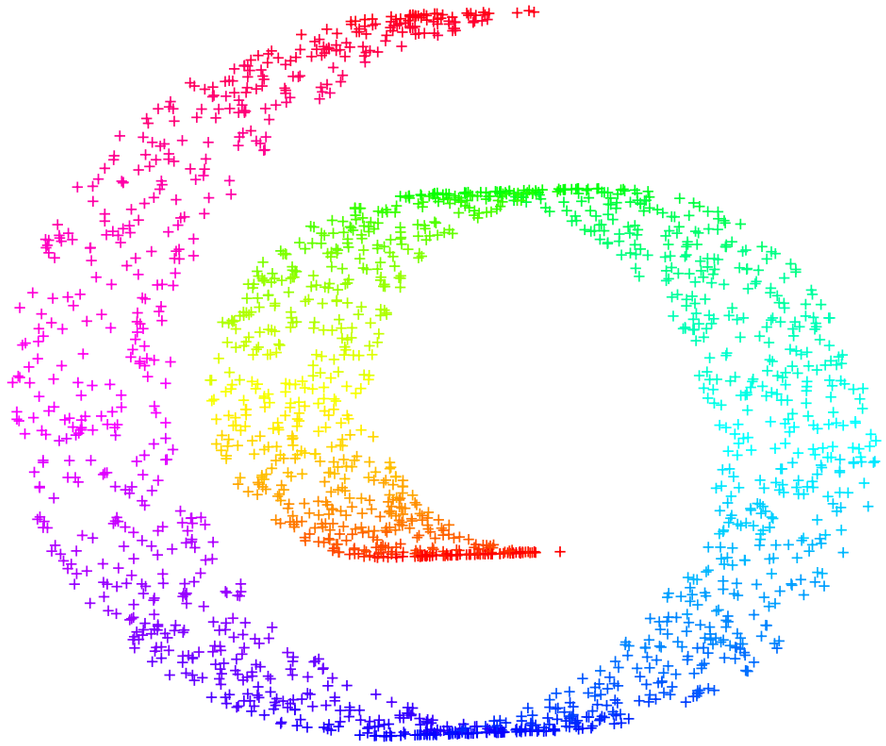
$$\boldsymbol{\mu} = \overline{\mathbf{x}} \qquad \mathbf{W}_{\mathrm{ML}} = \mathbf{U}_M(\mathbf{L}_M - \sigma^2\mathbf{I})^{1/2}\mathbf{R} \qquad \sigma^2_{\mathrm{ML}} = \frac{1}{D-M}\sum_{i=M+1}^{D}\lambda_i$$

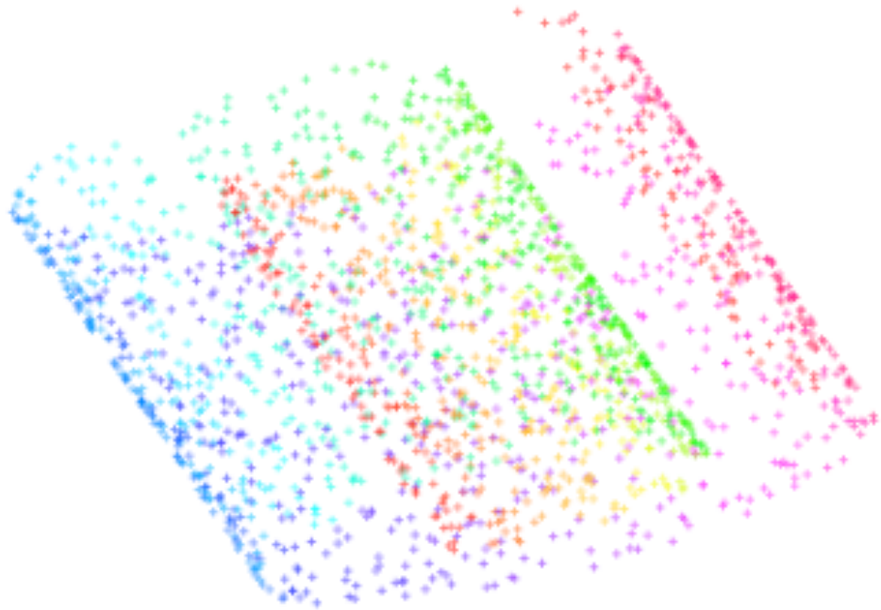
Check Bishop's book for the definitions

THE UNIVERSITY OF SYDNEY

# Research Topics

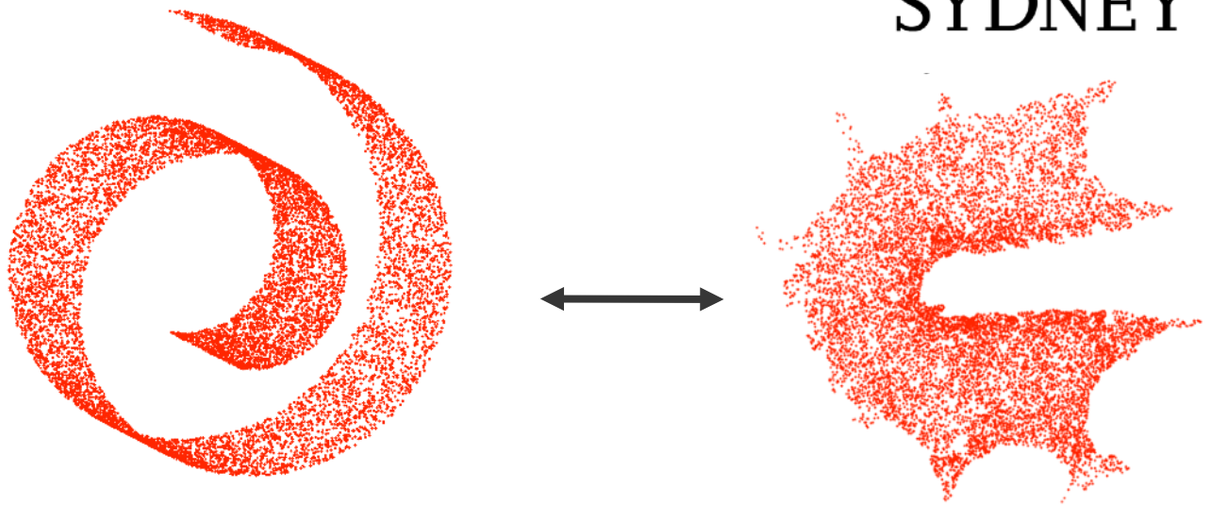
# Problems with Linear Mappings





PCA

# A perfect method



1. A bijective mapping ?

2. A “nice” mapping ?

3. A mapping that preserves distances ?

4. A mapping that preserves topology (neighbors) ?

Importance (and difficulty) to evaluate projections

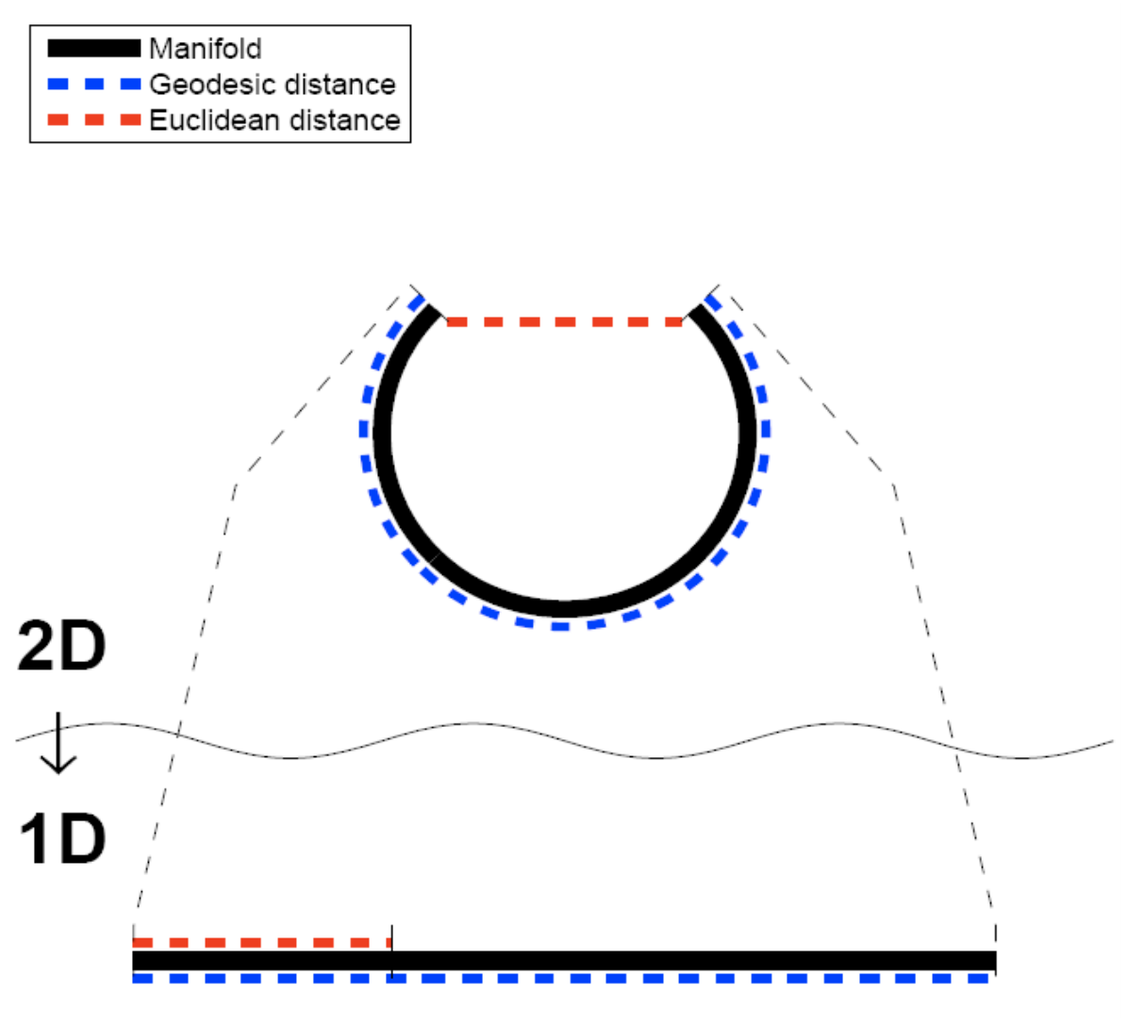
# Graph distances

Euclidean distances must always be increased to be projected

Solution: to measure distances along the manifold

Less discrepancy between $d_x$ and $d_y$, so easier optimization, etc.

So-called geodesic distance

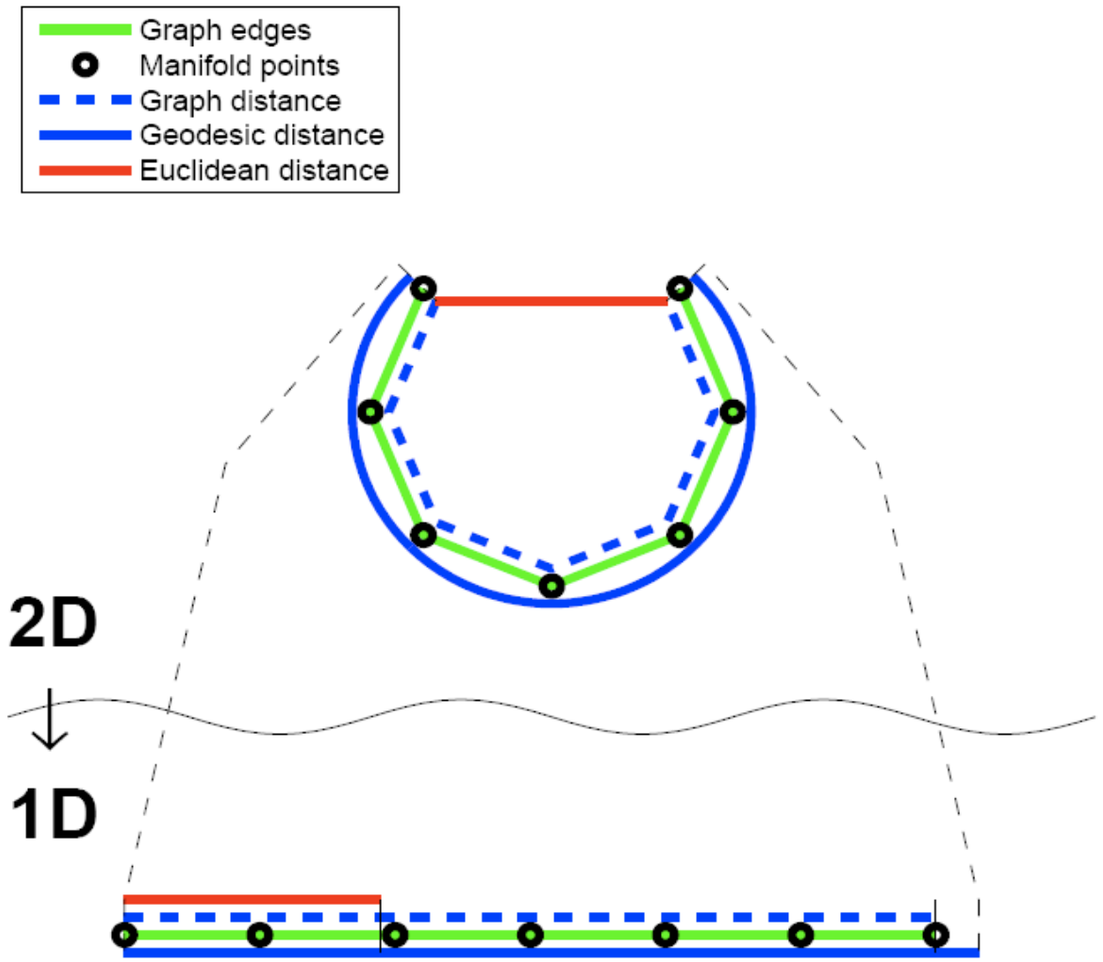


# Approximating graph distances

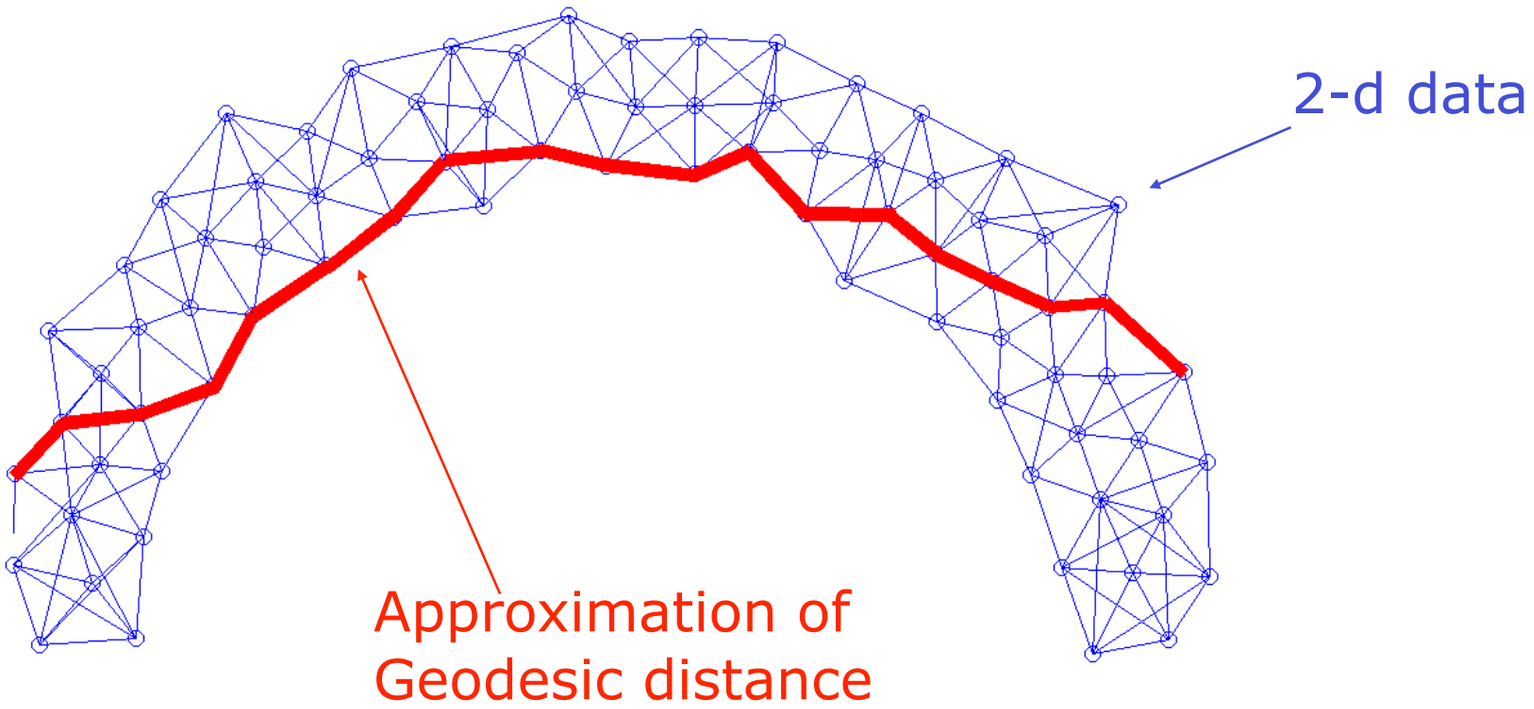
Problem: in d>1, geodesic distances may be measured along an infinity of paths
→ functional optimisation, intractable in practice

Idea: approximate manifolds with graphs

Graph distance
= sub of edge lengths
≈ geodesic distance



# Geodesic distances




How to build the graph from the data?

- Connect each data to its *k* nearest neighbours, or
- Connect each data to all other ones in a $\varepsilon$-ball
- Ensure connectivity of the graph

# Isomap

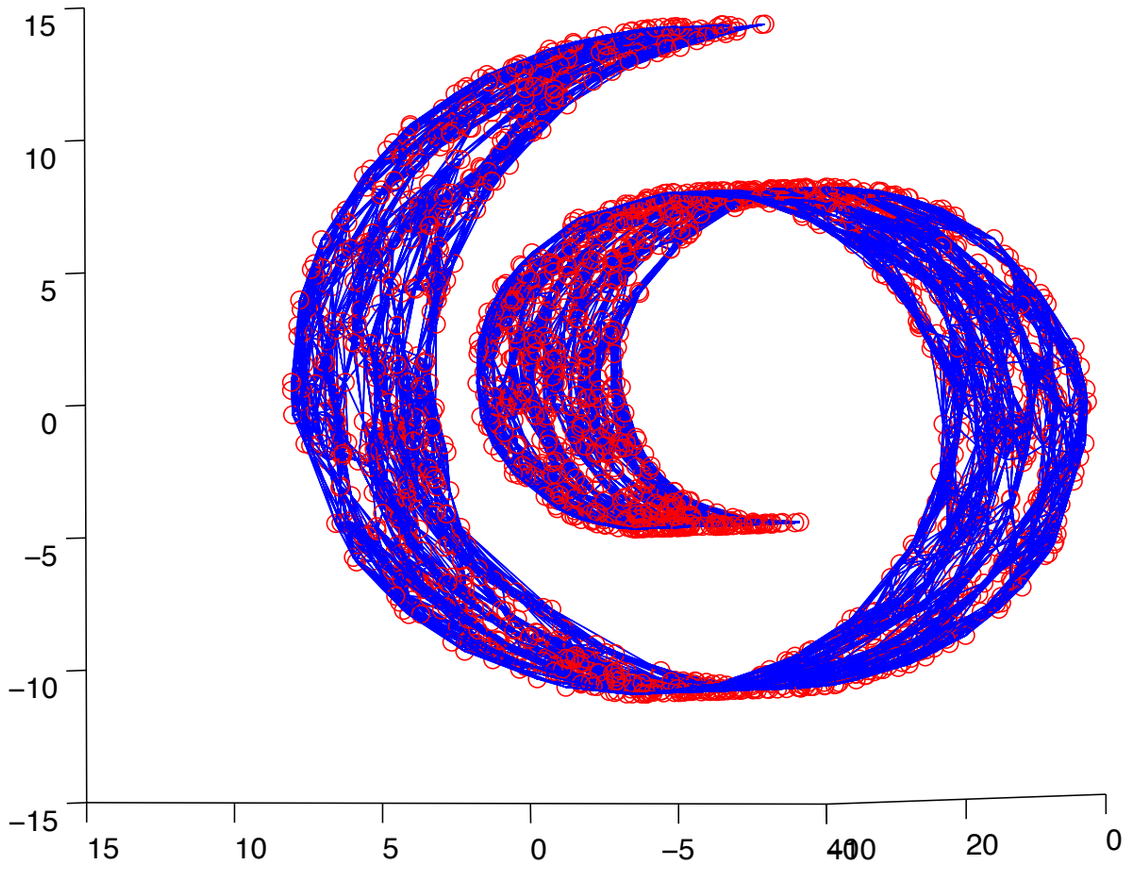
Build distance matrix

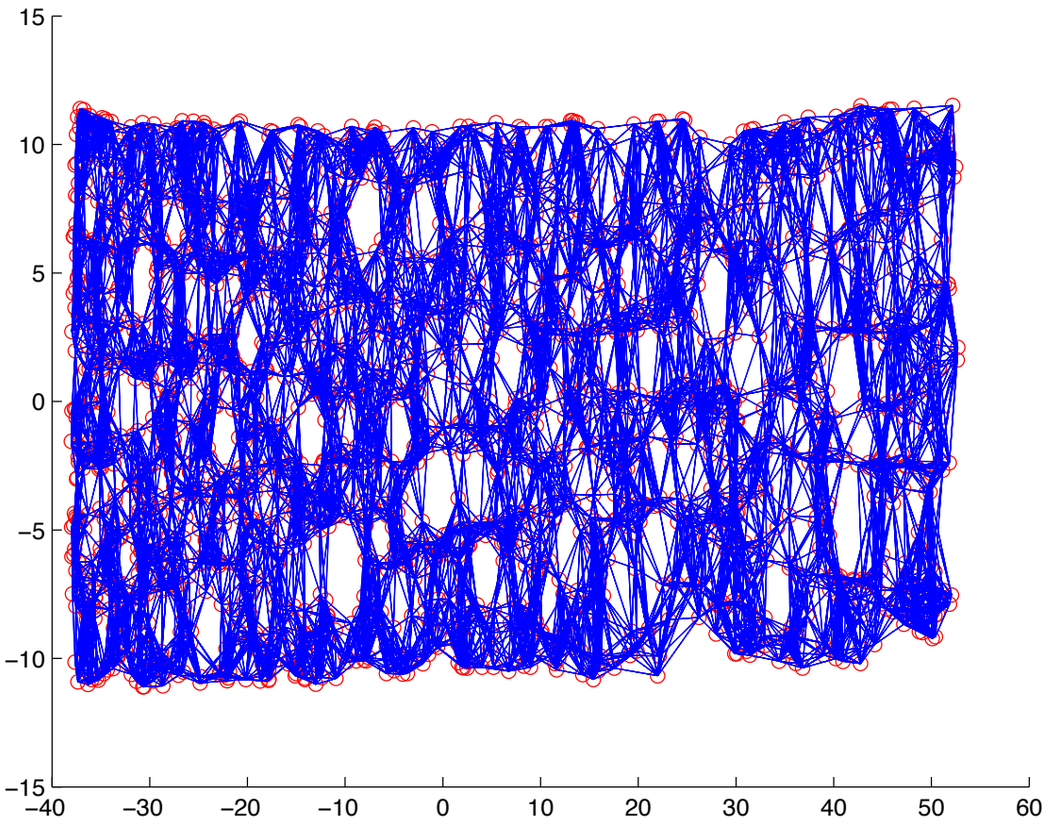
Compute nearest neighbour graph

Approximate geodesic distance by shortest path

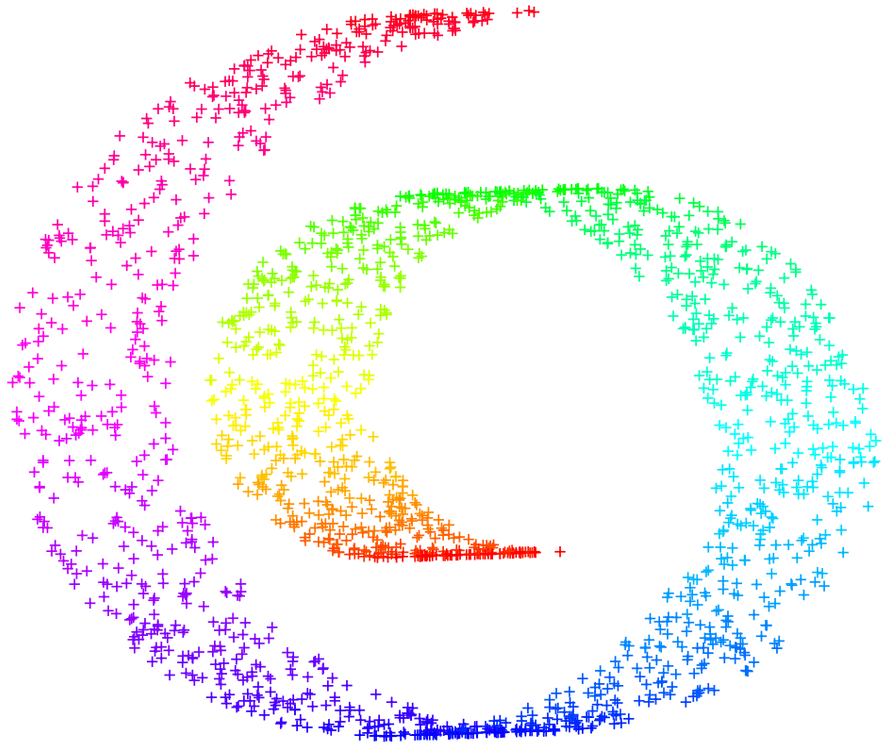
Metric MDS with geodesic (graph) distances

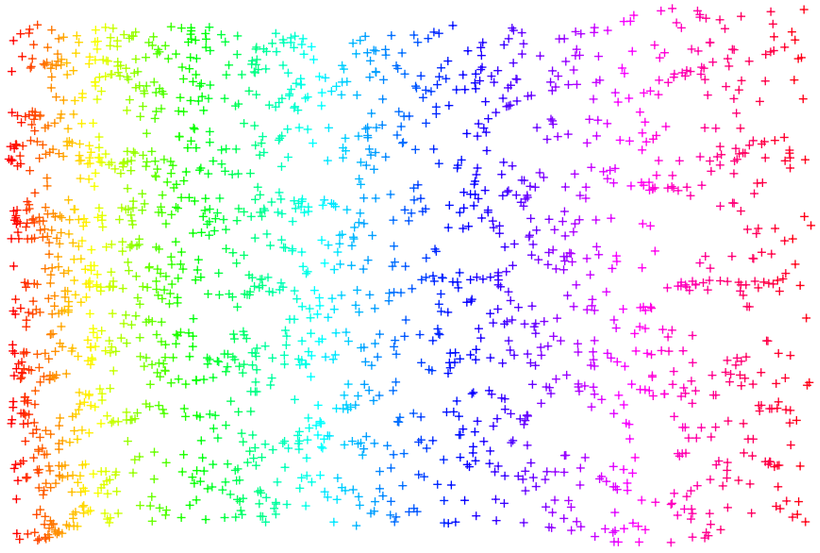
# Isomap


15
10
5
0
−5
−10
−15
15
10
5
0
−5
−10
20
0

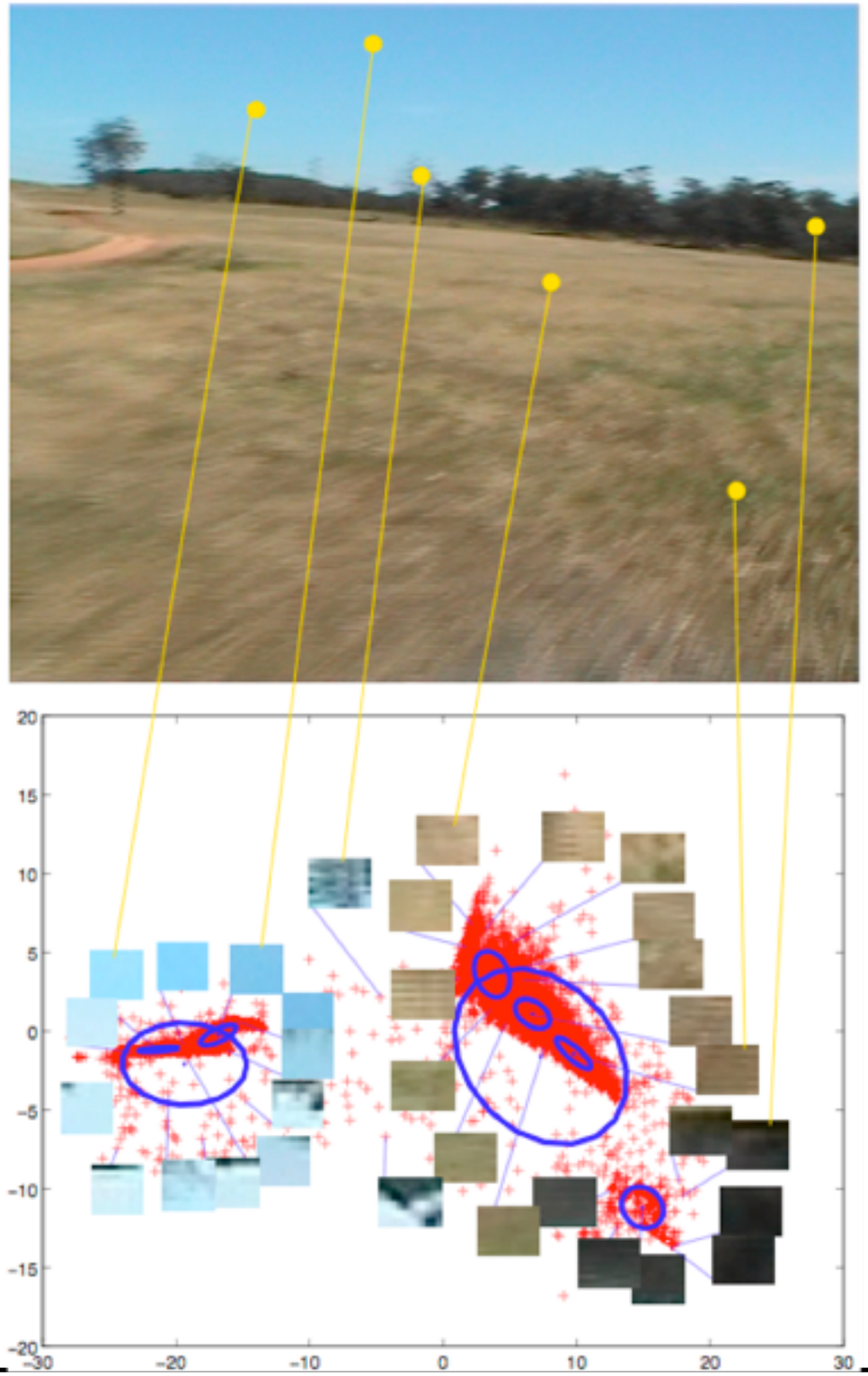

15
10
5
0
−5
−10
−15
−40
−30
−20
−10
0
10
20
30
40
50
60

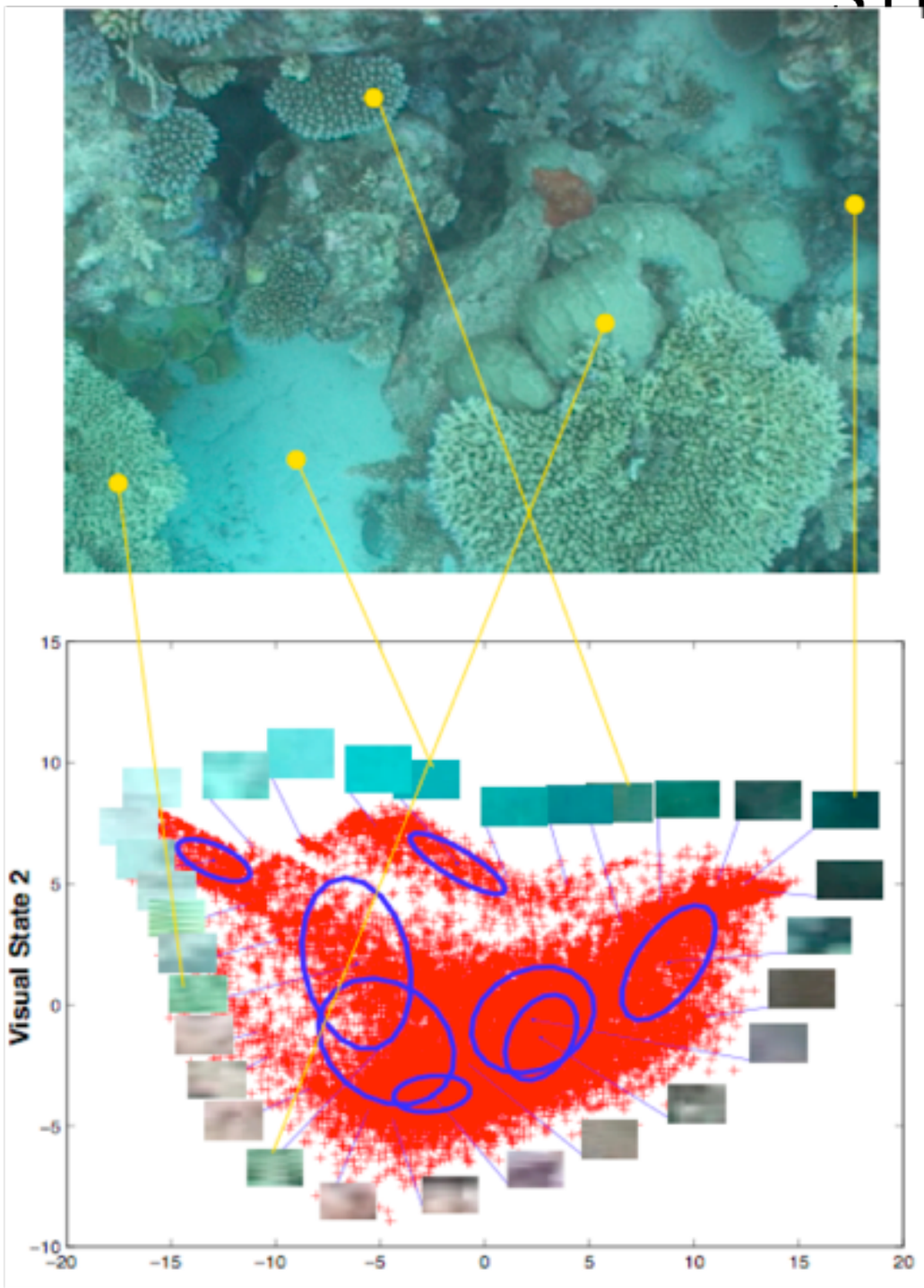
# Isomap



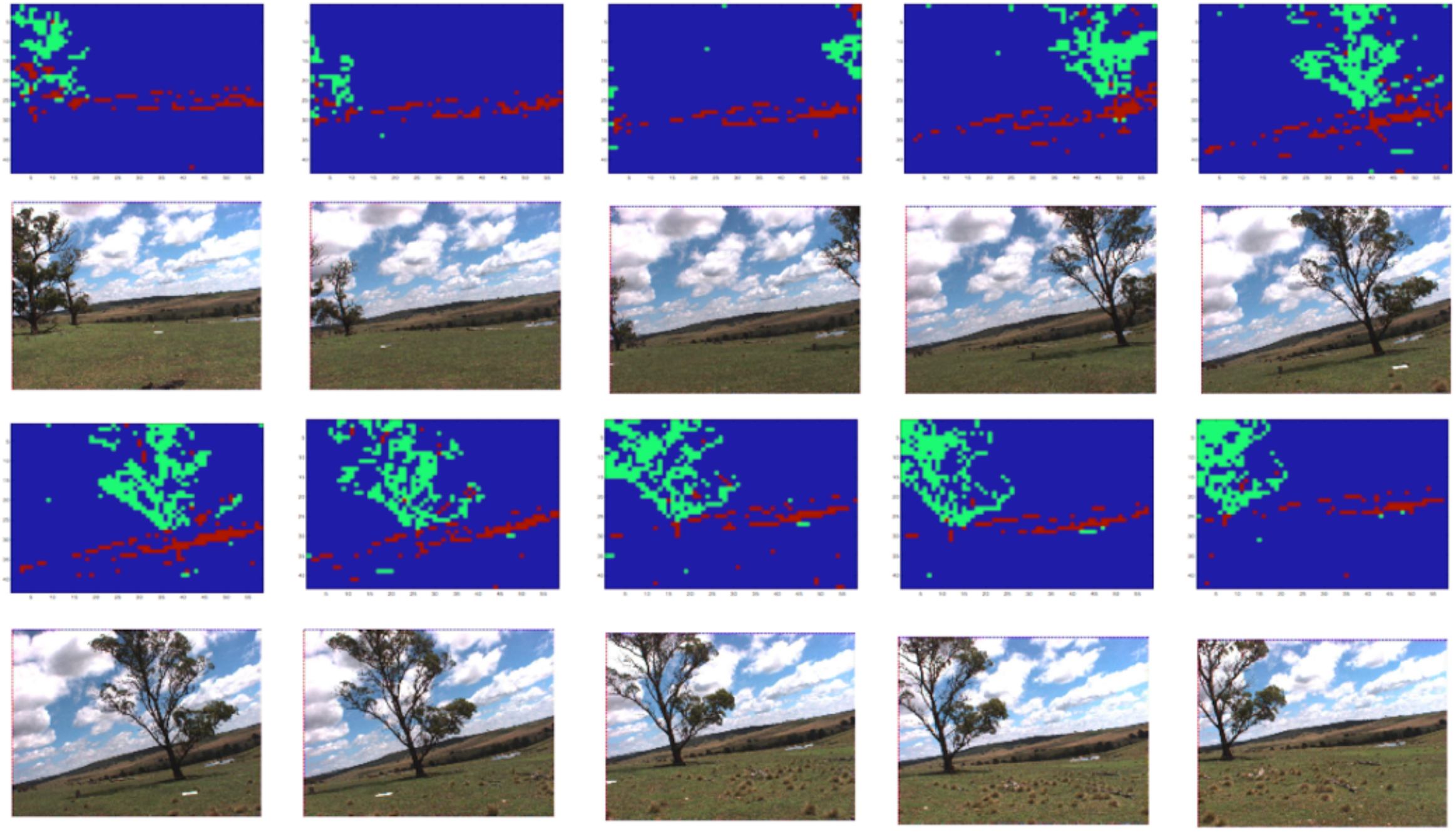


# Unstructured Environments

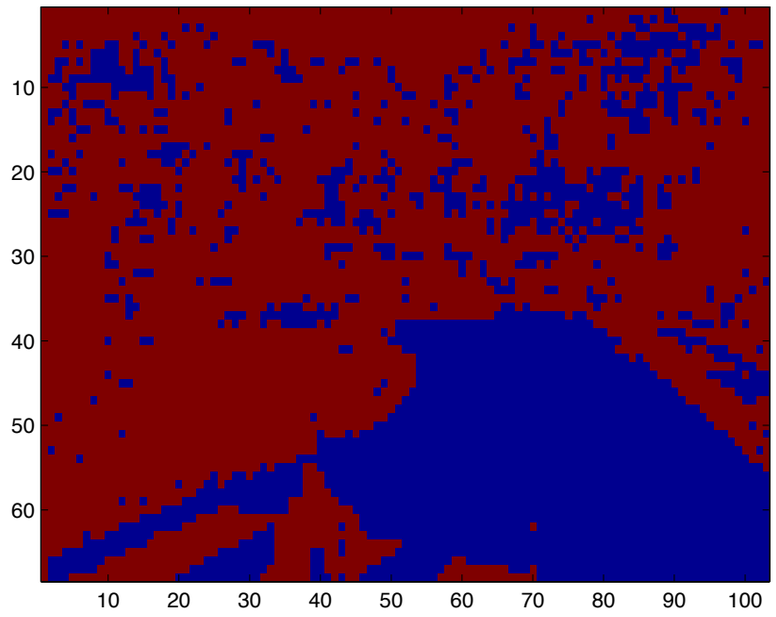


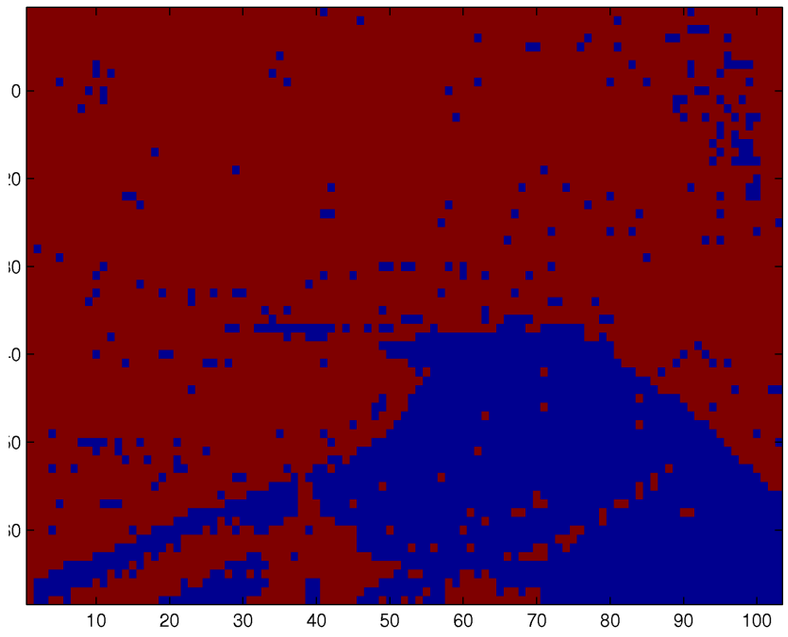


# Segmentation



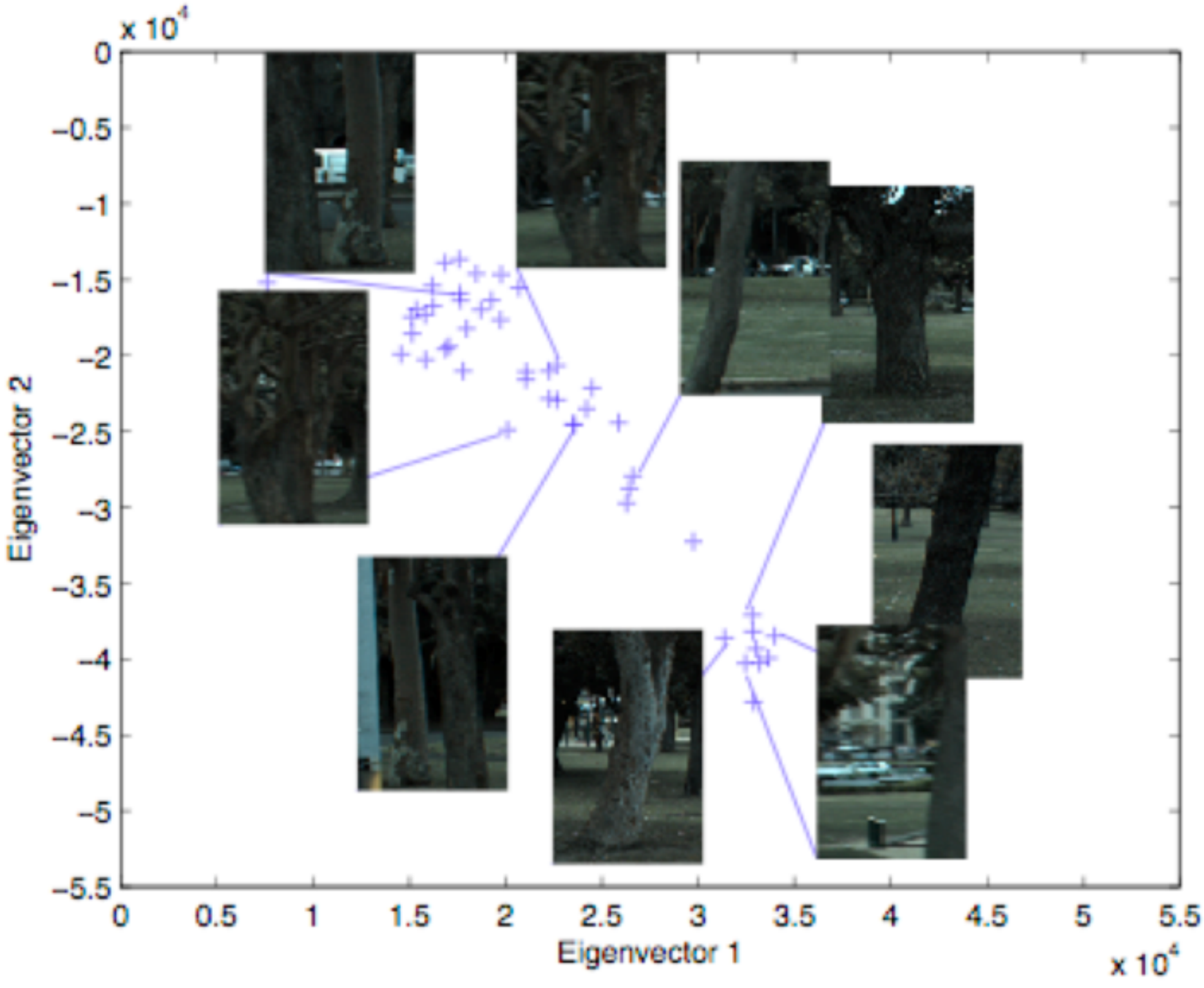
# Road Segmentation



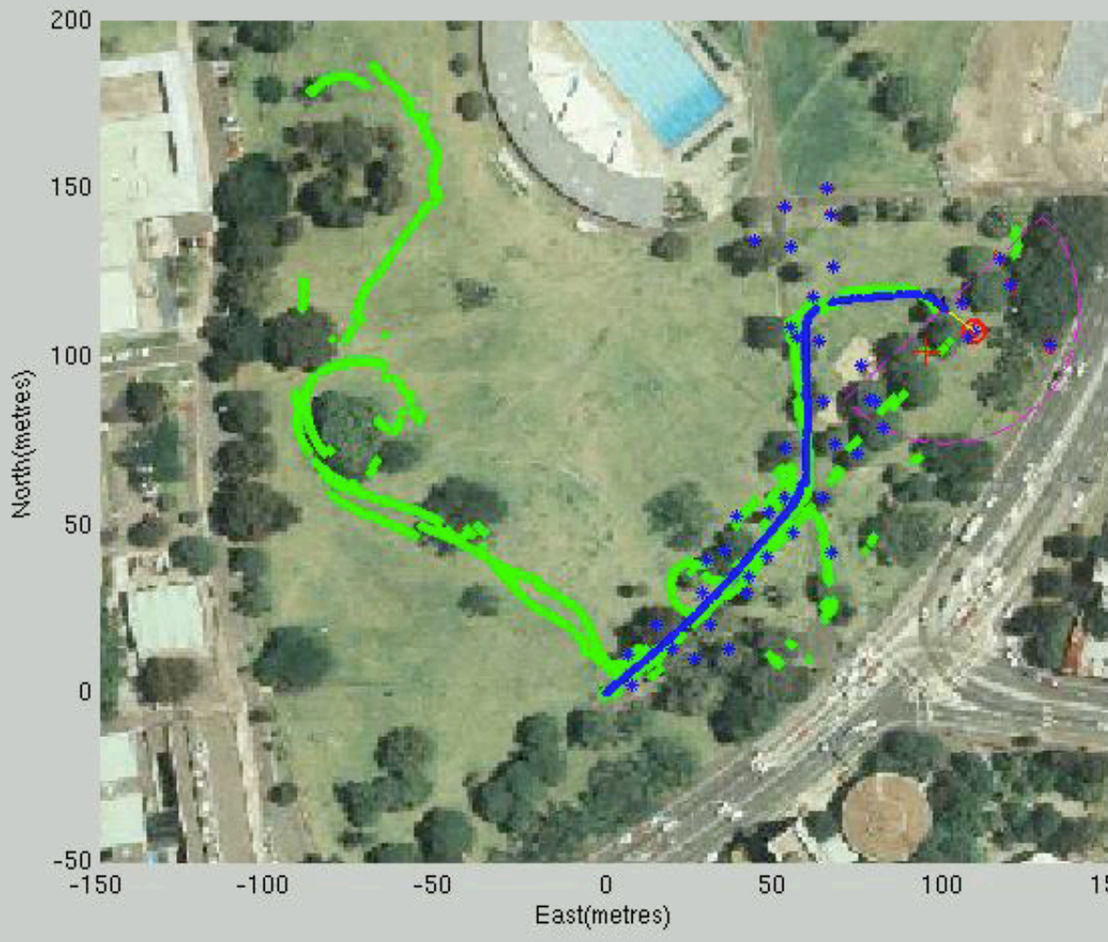
# Multi-View Representations

# Trees Manifold



# Closing the Loop in SLAM

THE UNIVERSITY OF SYDNEY


time=214.14 secs cputime=2768.69 secs
pos:[100.94][114.66][-46.95]