CS1138

Machine Learning

Lecture : Dimensionality Reduction (PCA and t-SNE)

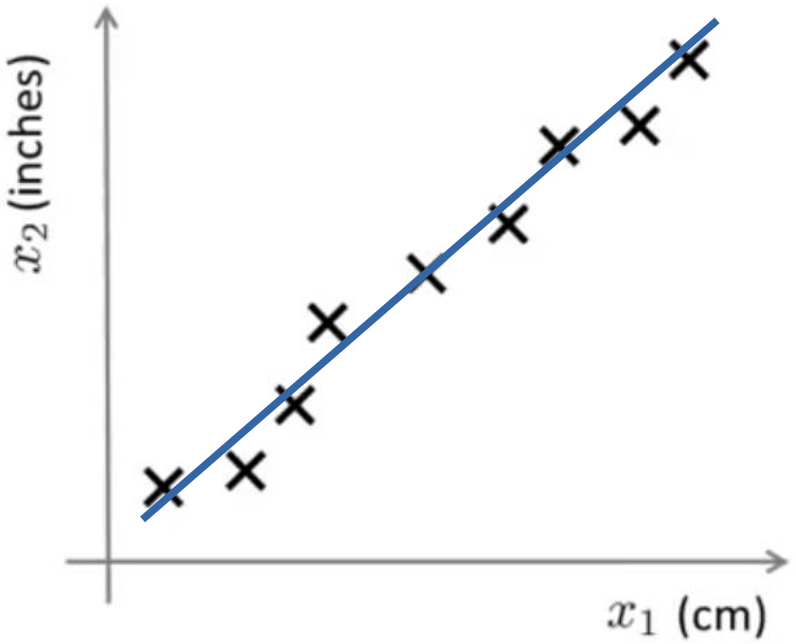
(Credits: Andrew Ng, Jure Leskovec)

Arpan Gupta

Overview

- Dimensionality Reduction: Motivation
- Low Rank: Eigen Faces Example
- PCA
- T-SNE

Data Compression



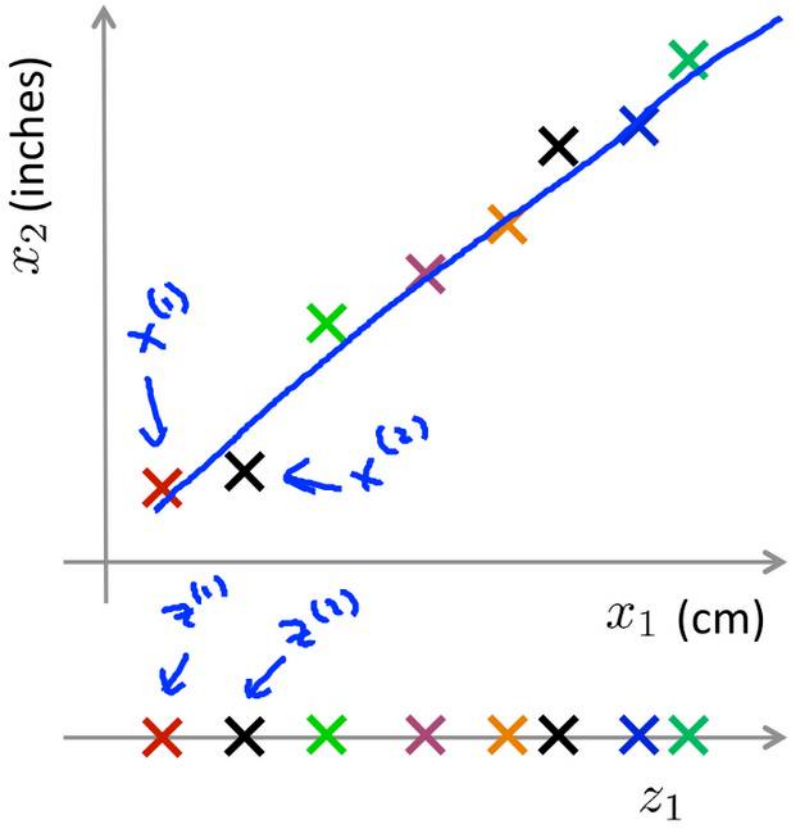
Reduce data from 2D to 1D

Suppose we have a dataset that measures the length of something in cm and in inches.

The features appear to be redundant in nature, and therefore, may be represented by a single feature.

There may be round-off errors or measurement errors.

Data Compression



To come up with a new feature z1, that has projections of the points on this axis. This approximates the dataset in one dimension.

Reduce data from 2D to 1D

$$x^{(1)} \in \mathbb{R}^2 \rightarrow z^{(1)} \in \mathbb{R}$$

$$x^{(2)} \in \mathbb{R}^2 \rightarrow z^{(2)} \in \mathbb{R}$$

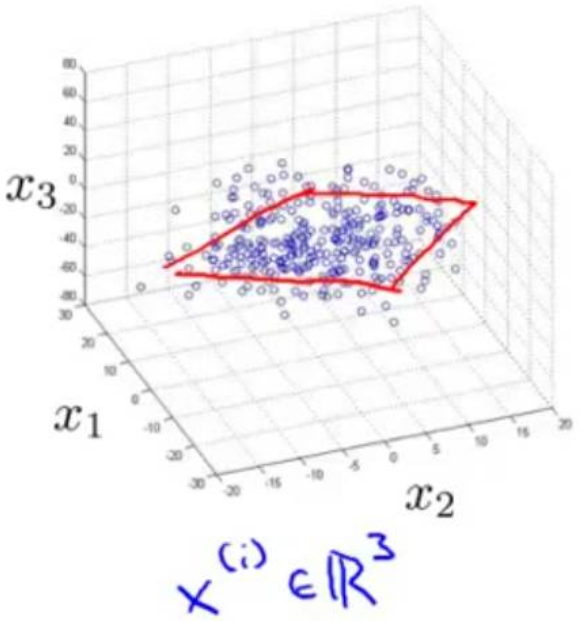
$$\vdots$$

$$x^{(m)} \in \mathbb{R}^2 \rightarrow z^{(m)} \in \mathbb{R}$$

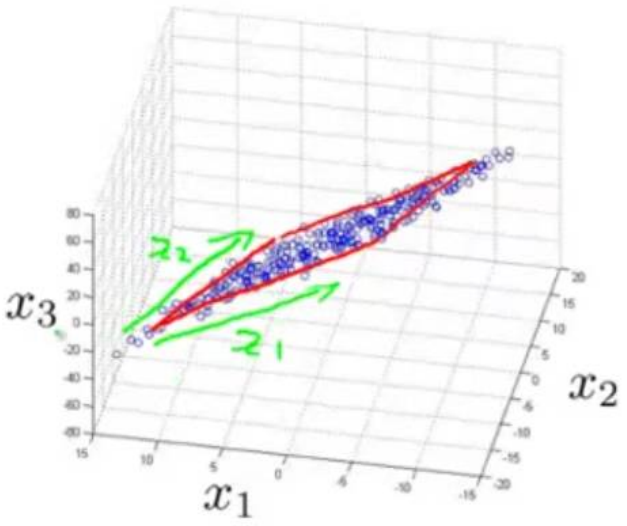
Data Compression

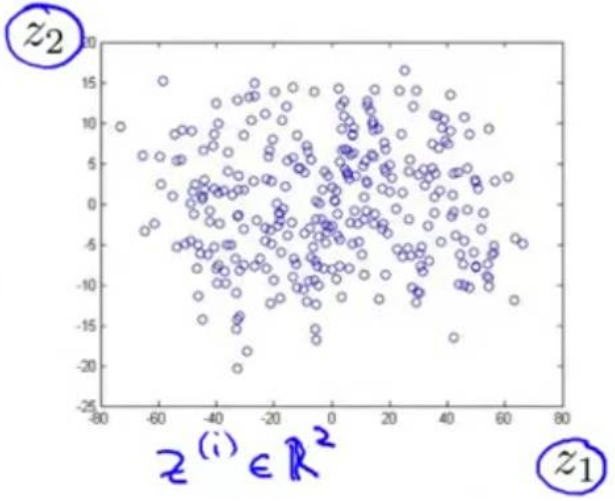
1000D → 100D

Reduce data from 3D to 2D



$x^{(i)} \in \mathbb{R}^3$





$z^{(i)} \in \mathbb{R}^2$

$$z = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} \qquad z^{(i)} = \begin{bmatrix} z_1^{(i)} \\ z_2^{(i)} \end{bmatrix}$$

Motivation: Dimensionality Reduction

- Curse of dimensionality: Reducing the size of feature set sometimes improves performance, especially for K-NN or Decision Trees.
- Run ML algorithms more efficiently with less no. of features, while ensuring that it’s predictive performance is maintained.
- Easier Data Collection
- Storage Space.
- Interpretability and Data Visualization.

Data Visualization

$x \in \mathbb{R}^{50}$

Country	x_1 GDP (trillions of US$)	x_2 Per capita GDP (thousands of intl. $)	x_3 Human Develop-ment Index	x_4 Life expectancy	x_5 Poverty Index (Gini as percentage)	x_6 Mean household income (thousands of US$)	...
Canada	1.577	39.17	0.908	80.7	32.6	67.293	...
China	5.878	7.54	0.687	73	46.9	10.22	...
India	1.632	3.41	0.547	64.7	36.8	0.735	...
Russia	1.48	19.84	0.755	65.5	39.9	0.72	...
Singapore	0.223	56.69	0.866	80	42.5	67.1	...
USA	14.527	46.86	0.91	78.3	40.8	84.3	...
...	...	...	...	...	...	...	

[resources from en.wikipedia.org]

Andrew Ng

Data Visualization

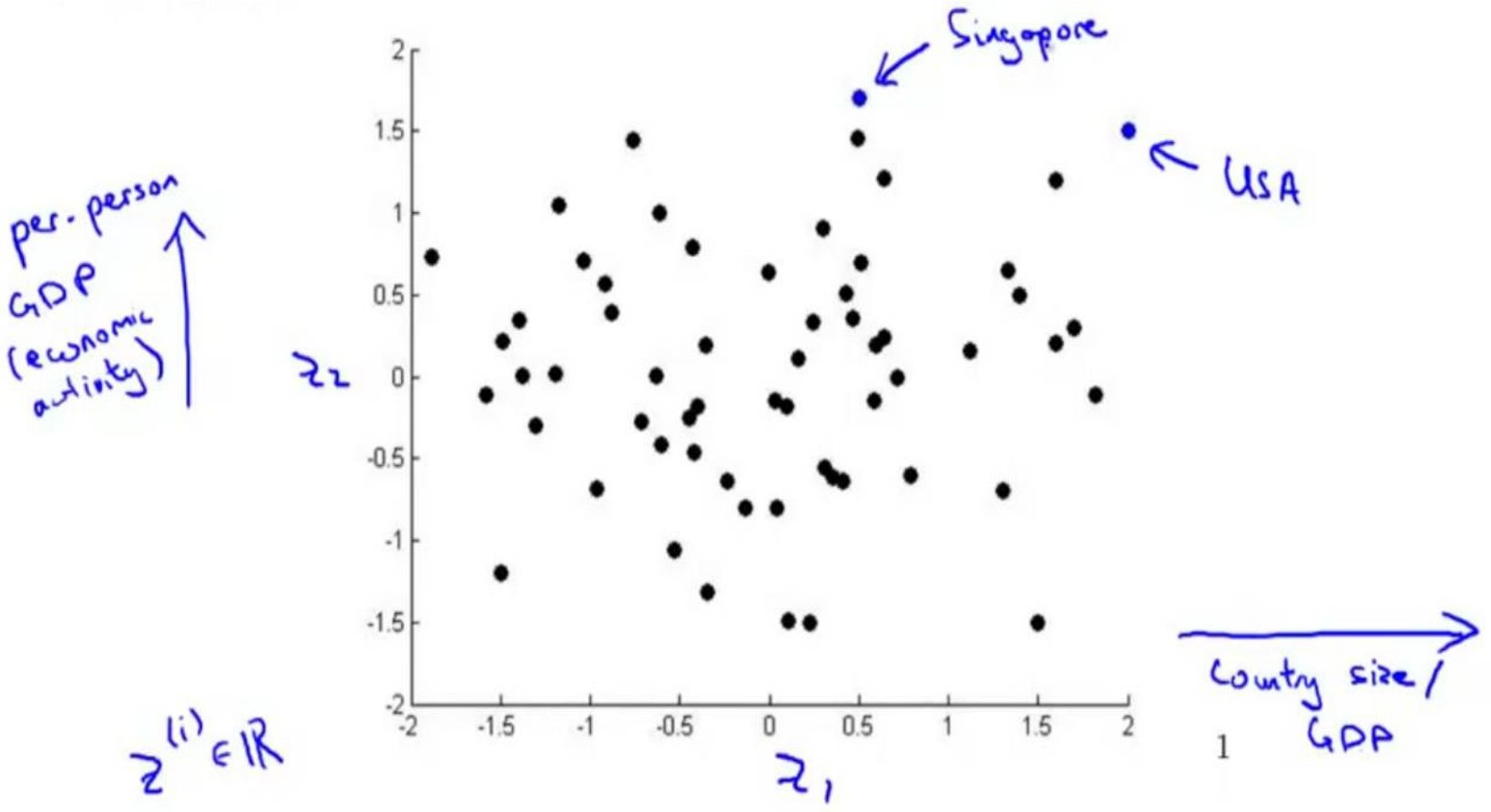
$z^{(i)} \in \mathbb{R}^2$

Country	z_1	z_2
Canada	1.6	1.2
China	1.7	0.3
India	1.6	0.2
Russia	1.4	0.5
Singapore	0.5	1.7
USA	2	1.5
...	...	...

Reduce the data from 50D (50 dimensions) to 2D.

It is upto us to infer the z_1 and z_2 values.

Data Visualization



High Dimensional Data in low dimensional subspaces

- In many datasets, (especially image datasets like MNIST dataset of handwritten digits, it is assumed that the data is not scattered throughout their high dimensional space. Instead, it lies in a low dimensional subspace.
- **Assumption:** Data lies on or near a low d-dimensional subspace.
- Axes of this subspace are effective representation of the data.
- We say that the data is of “low rank”.

Q: What is **rank** of a matrix **A**?

A: Number of **linearly independent** columns of **A**

For example:

- Matrix **A** = $\begin{bmatrix} 1 & 2 & 1 \\ -2 & -3 & 1 \\ 3 & 5 & 0 \end{bmatrix}$ has rank **r=2**

Why do we care about low rank?

- We can write **A** as two "basis" vectors: [1 2 1] [-2 -3 1]
- And new coordinates of :[1 0] [0 1] [1 -1]

High Dimensional Data of “Low Rank”

day / customer	We 7/10/96	Th 7/11/96	Fr 7/12/96	Sa 7/13/96	Su 7/14/96
ABC Inc.	1	1	1	0	0
DEF Ltd.	2	2	2	0	0
GHI Inc.	1	1	1	0	0
KLM Co.	5	5	5	0	0
Smith	0	0	0	2	2
Johnson	0	0	0	3	3
Thompson	0	0	0	1	1

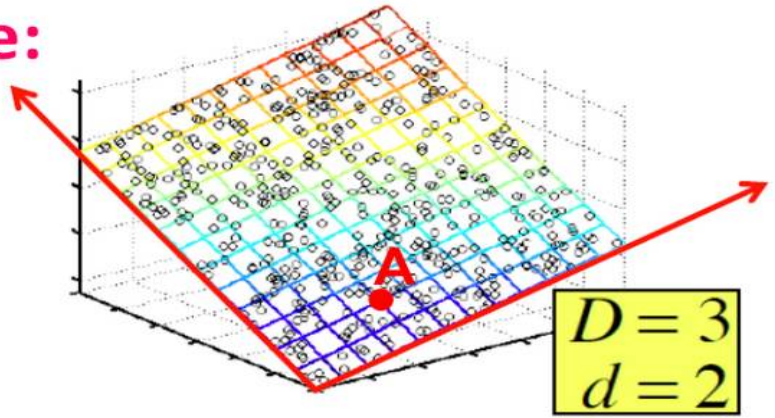
The above matrix is really “2-dimensional.” All rows can be reconstructed by scaling [1 1 1 0 0] or [0 0 0 1 1]

Rank is Dimensionality

- **Cloud of points 3D space:**
  - Think of point positions as a matrix:

1 row per point:

$$\begin{bmatrix} 1 & 2 & 1 \\ -2 & -3 & 1 \\ 3 & 5 & 0 \end{bmatrix} \begin{matrix} \text{A} \\ \text{B} \\ \text{C} \end{matrix}$$



- **We can rewrite coordinates more efficiently!**
  - Old coordinate system: [1 0 0] [0 1 0] [0 0 1]
  - **New coordinate system: [1 2 1] [-2 -3 1]**
  - Then **A** has new coordinates: [1 0]. **B**: [0 1], **C**: [1 -1]
    - **Notice: We reduced the number of coordinates!**

Example: Eigenfaces (Turk and Pentland, 1991)

- Assume that most face images lie on a low-dimensional subspace determined by the first k (k<<<d) directions of maximum variance.
- Use PCA to determine the vectors or “eigenfaces” u_1, u_2, ..., u_k that span the subspace.
- Represent all face images in the dataset as linear combinations of eigenfaces. Find the coefficients by dot product.

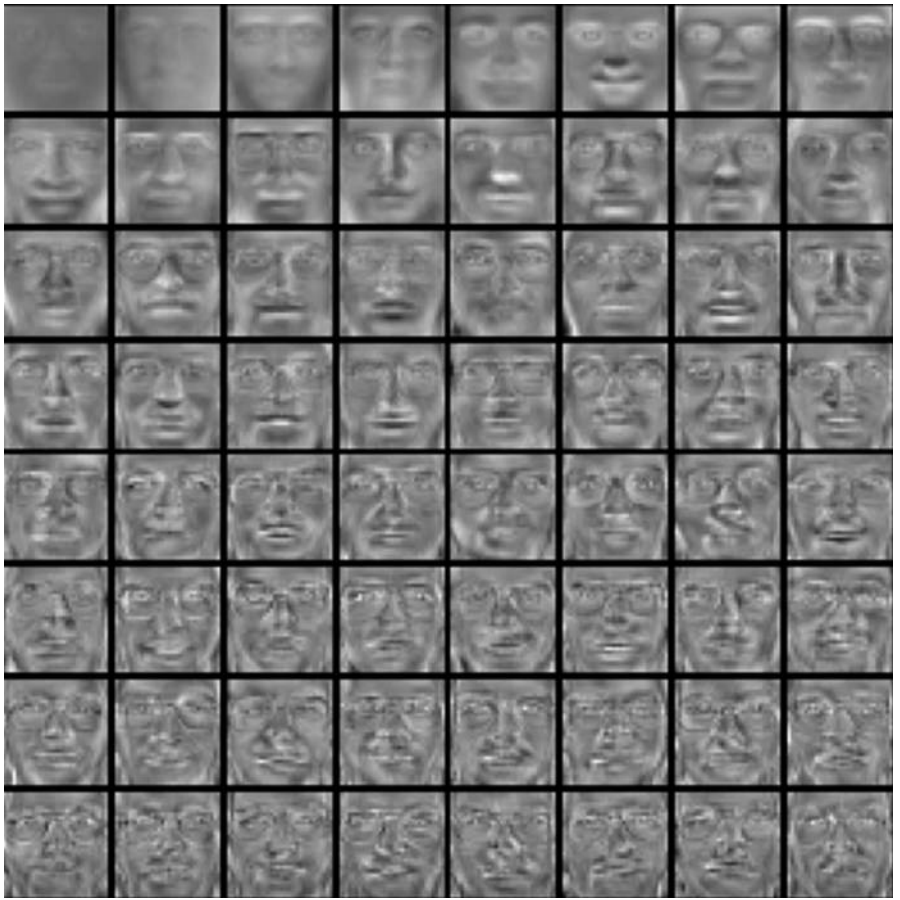
Example: Eigenfaces

Training examples
$x_1, \ldots x_m$

Example: Eigenfaces

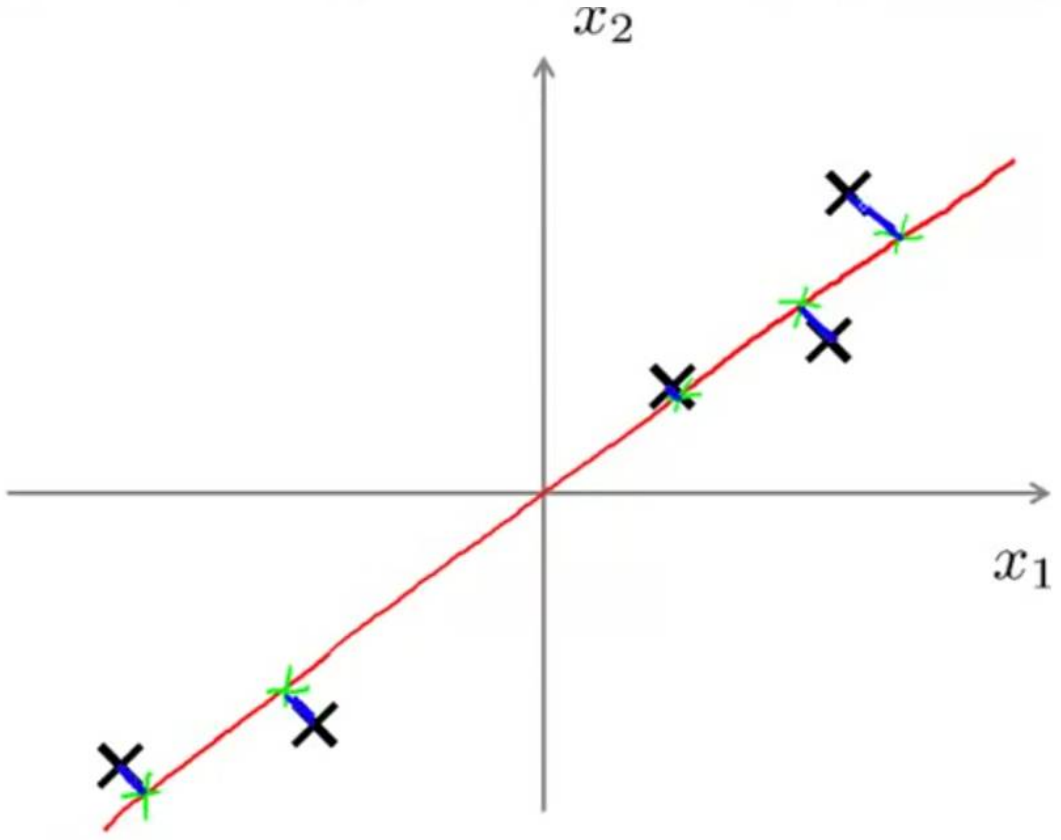
Mean: μ

Top eigenvectors u_1, ..., u_k



Principal Component Analysis (PCA)

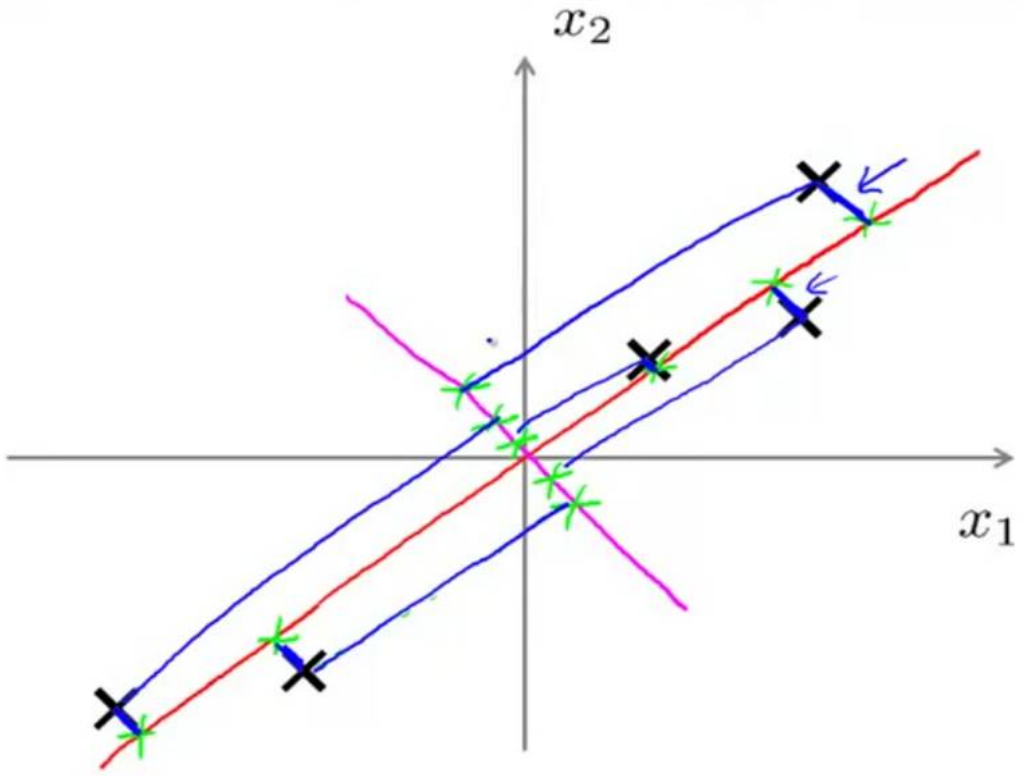
Principal Component Analysis (PCA) problem formulation



$x \in \mathbb{R}^2$

Minimizes the squared projection errors. (orthogonal distances of the points from the line).

Principal Component Analysis (PCA) problem formulation

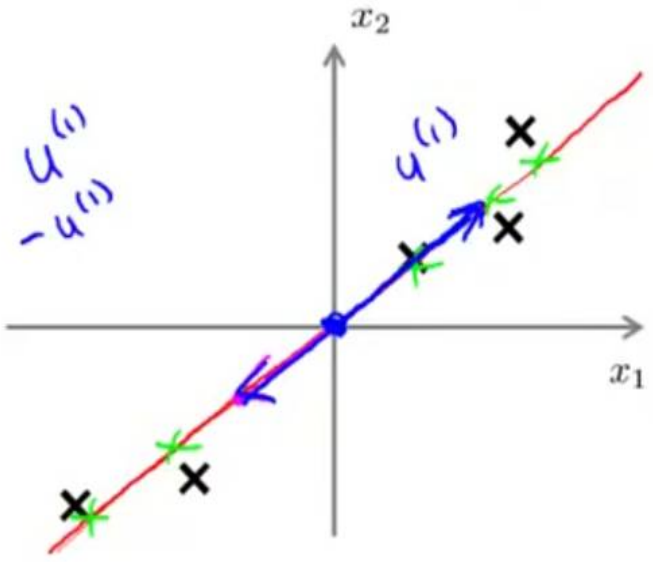


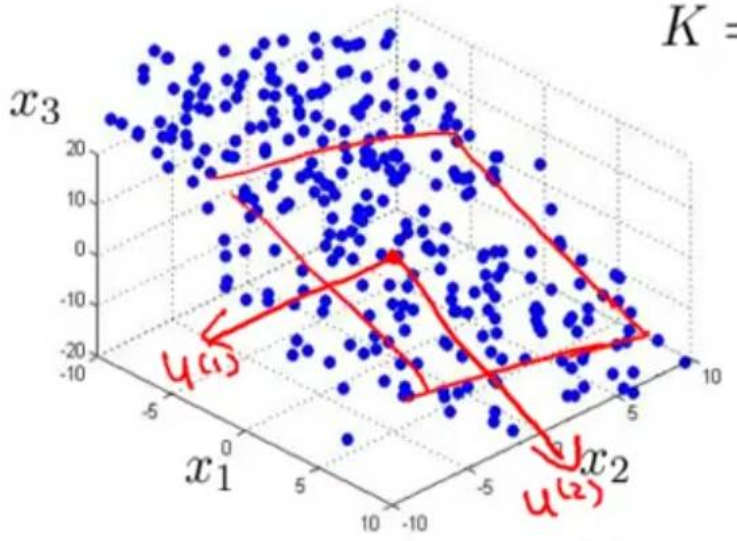
$x \in \mathbb{R}^2$

Projecting the points on the red line is better than projecting on the magenta line, since the points better retain their spread-out nature (variance). Projection errors are more for magenta line. PCA will choose the red line instead of magenta line.

Principal Component Analysis (PCA) problem formulation

$3D \to 2D$
$K = 2$

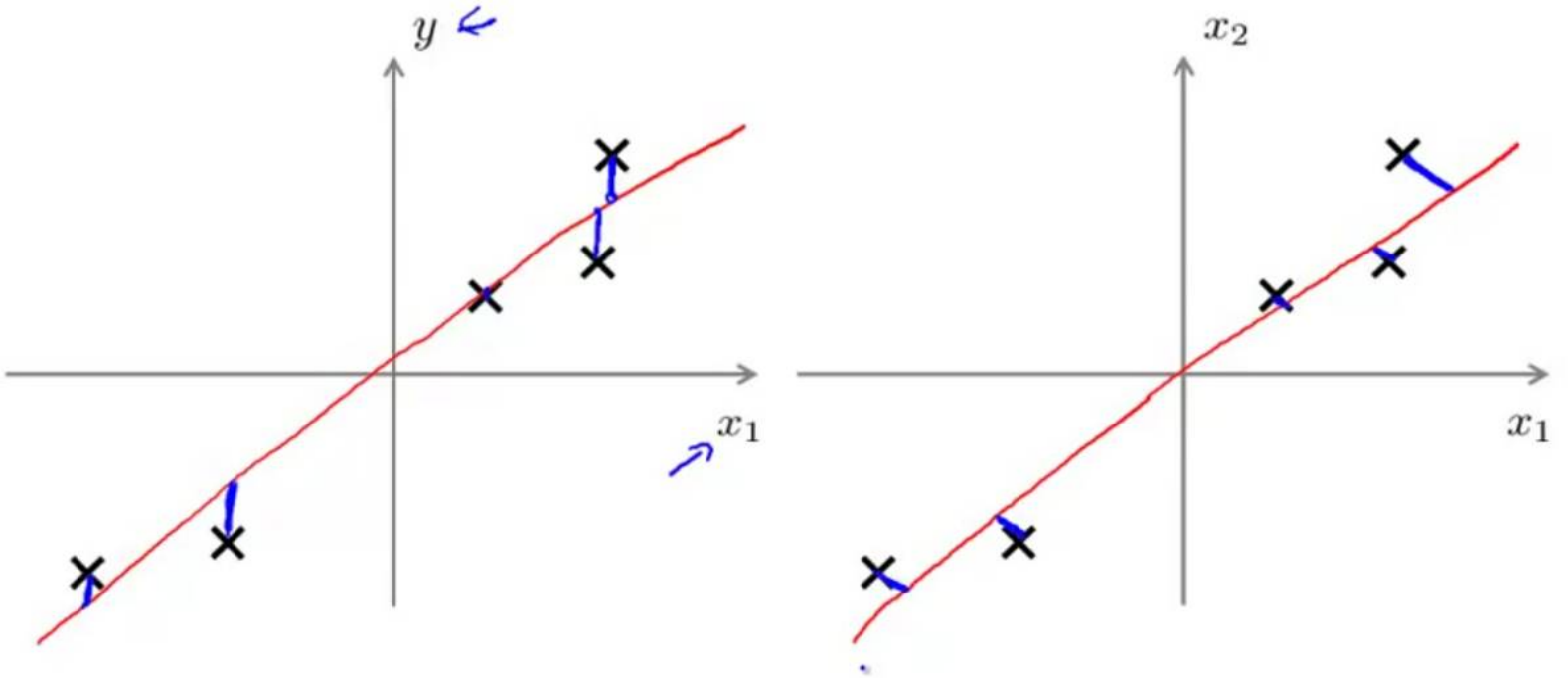




Reduce from 2-dimension to 1-dimension: Find a direction (a vector $u^{(1)} \in \mathbb{R}^n$) onto which to project the data so as to minimize the projection error.

Reduce from n-dimension to k-dimension: Find k vectors $u^{(1)}, u^{(2)}, \ldots, u^{(k)}$ onto which to project the data, so as to minimize the projection error.

PCA is not linear regression



Linear Regression minimizes the squared errors from the point and the straight line (Vertical Distances). PCA minimizes the shortest orthogonal distance of the point from the red line.

Also, LR has label Y (supervised), while PCA has only the features (x_i)

Data preprocessing

Training set: $x^{(1)}, x^{(2)}, \ldots, x^{(m)}$

Preprocessing (feature scaling/mean normalization):

$$\mu_j = \frac{1}{m} \sum_{i=1}^{m} x_j^{(i)}$$

Replace each $x_j^{(i)}$ with $x_j - \mu_j$.

If different features on different scales (e.g., $x_1 =$ size of house, $x_2 =$ number of bedrooms), scale features to have comparable range of values.

$$x_j^{(i)} \leftarrow \frac{x_j^{(i)} - \mu_j}{s_j}$$

Principal Component Analysis (PCA) algorithm

Reduce data from n-dimensions to k-dimensions

Compute "covariance matrix":

$$\Sigma = \frac{1}{m} \sum_{i=1}^{m} (x^{(i)})(x^{(i)})^T$$

$n \times 1$ $\quad$ $1 \times n$ $\quad$ $n \times n$ $\quad$ Sigma

Compute "eigenvectors" of matrix Σ:

```
[U,S,V] = svd(Sigma);
```

→ Singular value decomposition

eig (Sigma)

n×n matrix.

$$U = \begin{bmatrix} | & | & | & & | \\ u^{(1)} & u^{(2)} & u^{(3)} & \cdots & u^{(n)} \\ | & | & | & & | \end{bmatrix}$$

k

$U \in \mathbb{R}^{n \times n}$

$u^{(1)}, \ldots, u^{(k)}$

Covariance matrix is always symmetric positive semi-definite. Eigen Values will always be real.

numpy.linalg.svd

Take k out of n vectors that will be the k directions on which to project the data.

Principal Component Analysis (PCA) algorithm

From `[U,S,V] = svd(Sigma)`, we get:

$$\rightarrow U = \underbrace{\begin{bmatrix} | & | & & | \\ u^{(1)} & u^{(2)} & \cdots & u^{(n)} \\ | & | & & | \end{bmatrix}}_{k} \in \mathbb{R}^{n \times n}$$

$x \in \mathbb{R}^n \longrightarrow z \in \mathbb{R}^k$

$$z = \underbrace{\begin{bmatrix} | & | & & | \\ u^{(1)} & u^{(2)} & \cdots & u^{(k)} \\ | & | & & | \end{bmatrix}^T}_{n \times k} \quad x = \underbrace{\begin{bmatrix} — (u^{(1)})^T — \\ \vdots \\ — (u^{(k)})^T — \end{bmatrix}}_{k \times n} \underbrace{x}_{n \times 1}$$

$z \in \mathbb{R}^k$

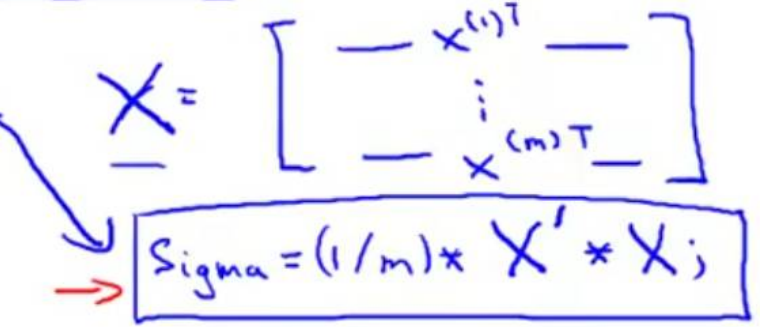
U_{reduce}

$k \times 1$

Principal Component Analysis (PCA) algorithm summary

After mean normalization (ensure every feature has zero mean) and optionally feature scaling:

Sigma = $\frac{1}{m}\sum_{i=1}^{m}(x^{(i)})(x^{(i)})^T$




```
[U,S,V] = svd(Sigma);
Ureduce = U(:,1:k);
z = Ureduce'*x;
```

Choosing k (number of principal components)

Average squared projection error: $\frac{1}{m}\sum_{i=1}^{m} \|x^{(i)} - x_{approx}^{(i)}\|^2$

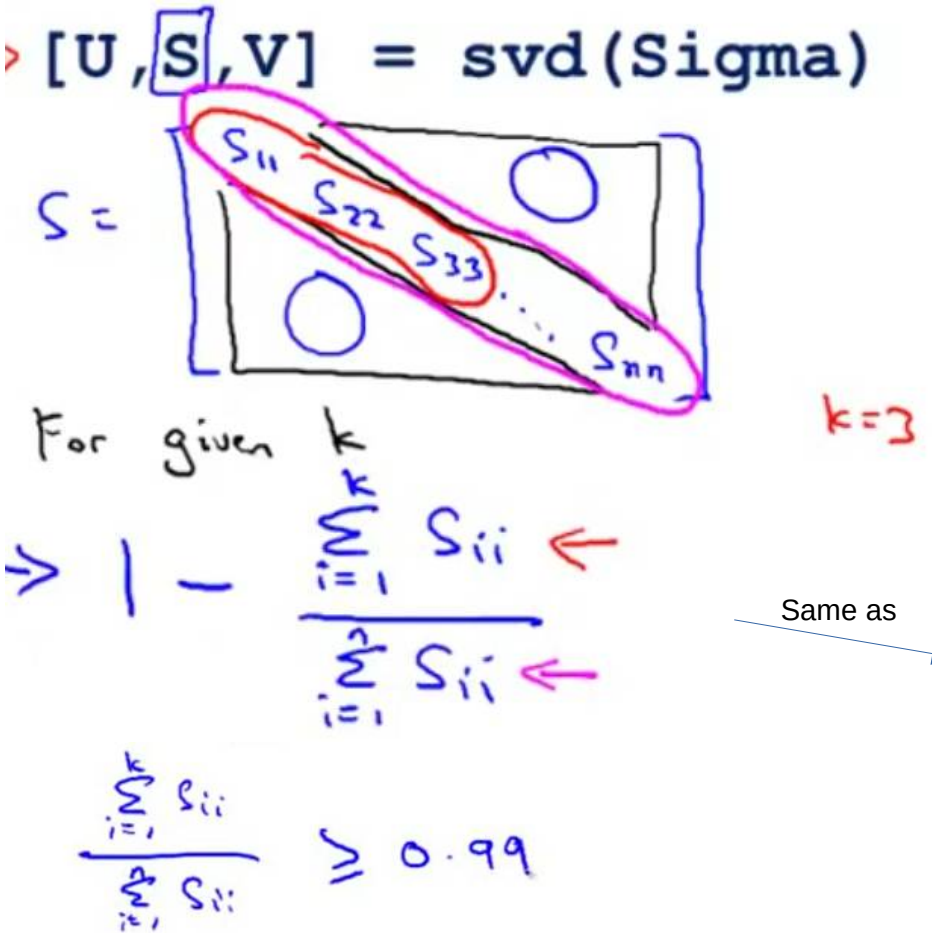
Total variation in the data: $\frac{1}{m}\sum_{i=1}^{m} \|x^{(i)}\|^2$

Typically, choose k to be smallest value so that

$$\frac{\frac{1}{m}\sum_{i=1}^{m} \|x^{(i)} - x_{approx}^{(i)}\|^2}{\frac{1}{m}\sum_{i=1}^{m} \|x^{(i)}\|^2} \leq 0.01 \quad (1\%)$$

"99% of variance is retained"

Alternative: Using singular values



Same as

$$\frac{\frac{1}{m}\sum_{i=1}^{m} \|x^{(i)} - x_{approx}^{(i)}\|^2}{\frac{1}{m}\sum_{i=1}^{m} \|x^{(i)}\|^2}$$

Alternative: Using singular values

Choosing k (number of principal components)

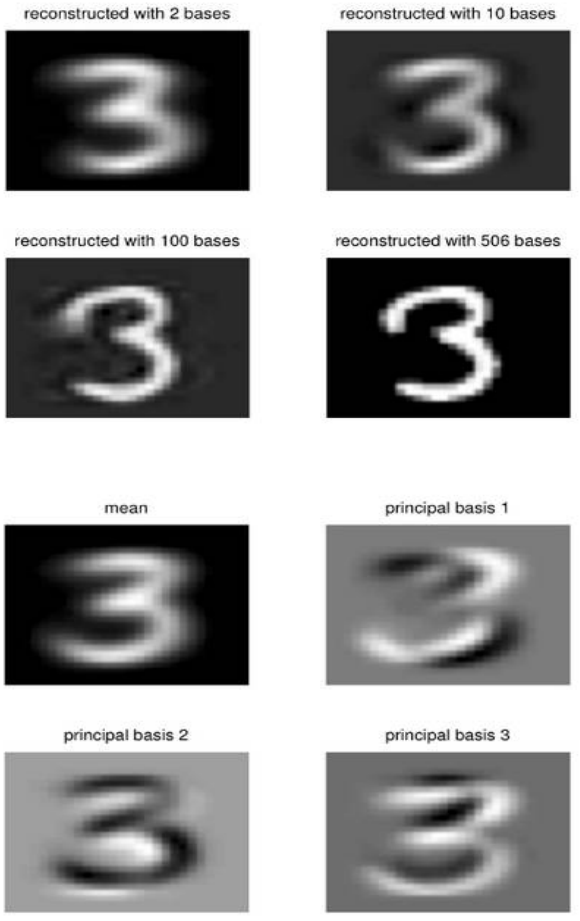
- `[U,S,V] = svd(Sigma)`

Pick smallest value of k for which

$$\frac{\sum_{i=1}^{k} S_{ii}}{\sum_{i=1}^{n} S_{ii}} \geq 0.99$$

(99% of variance retained)

Applying PCA to Digits



t-SNE

t-SNE : t-distributed Stochastic Neighbour Embedding

- One of the best dimensionality reduction technique, especially for visualization.
- Published in 2008 in JMLR.

  Van der Maaten, Laurens, and Geoffrey Hinton. **"Visualizing data using t-SNE."** Journal of machine learning research 9.11 (2008).

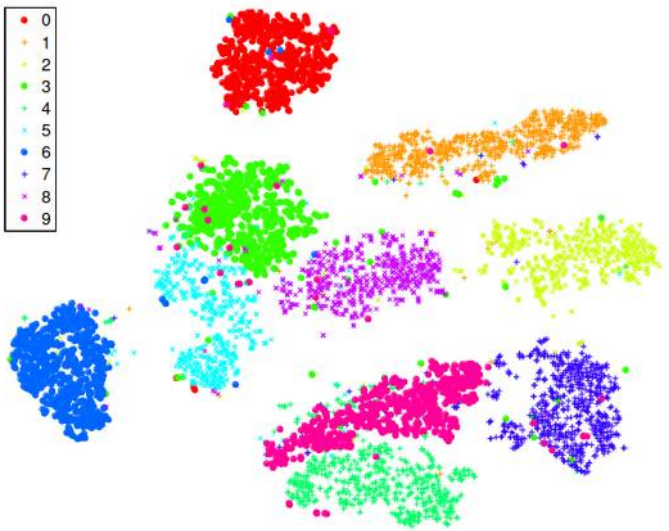
  https://jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf

- https://colah.github.io/posts/2014-10-Visualizing-MNIST/

t-SNE : t-distributed Stochastic Neighbour Embedding

- PCA tries to preserve the global shape (structure) of data. Maximizing variance achieves this.
- But many times the “local” structure is lost. i.e., the neighbourhood information may be lost or the distances information between points may be lost.
- t-SNE is flexible enough to preserve local structure. It may also be modified to preserve global structure.

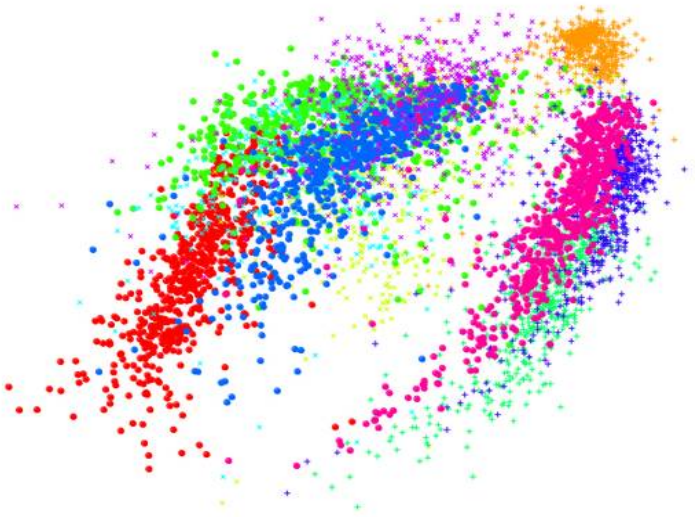
Visualizing MNIST



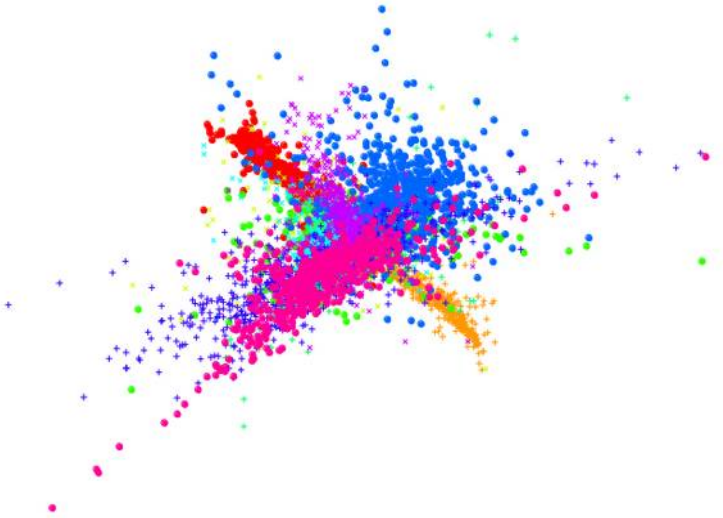
(a) Visualization by t-SNE.

(b) Visualization by Sammon mapping.

Visualizing MNIST



(a) Visualization by Isomap.



(b) Visualization by LLE.

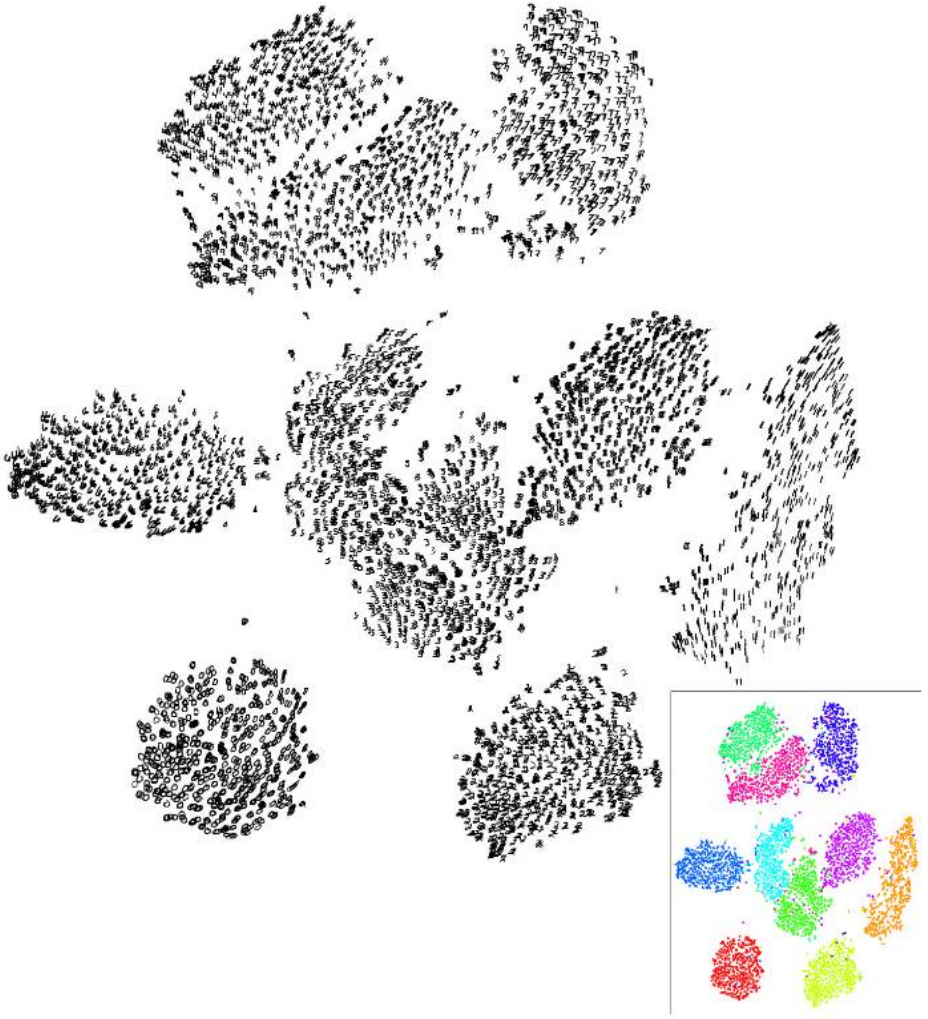


Figure 7: Visualization of 6,000 digits from the MNIST data set produced by the random walk version of t-SNE (employing all 60,000 digit images).

References

- Andrew Ng – CS229 Stanford Machine Learning course
- Mining Massive Datasets – Jure Lescovec – Stanford.
- t – SNE : Applied AI Course (YouTube)
  - https://www.youtube.com/watch?v=FQmCzpKWD48&list=PLupD_xFct8mHqCkuaXmeXhe0ajNDu0mhZ&index=2
- Generalized Principal Component Analysis Course - John Hopkins – by Rene Vidal. - Introductory Lecture.

End of Lecture