Machine Learning Tools for the Visualisation of BigData

We all love images!

Statutory warning: Has been produced in the same facility which produces maths and may contains traces of maths.

Amit Kumar Mishra
Kyle Harrison

University of Cape Town

Learning Machines and Data!

Linear and nonlinear methods to compress

Introduction

- Human beings are lazy and commit errors!

Introduction

- Human beings are lazy and commit errors!
- Human level intelligence will need human level error! (Turing)

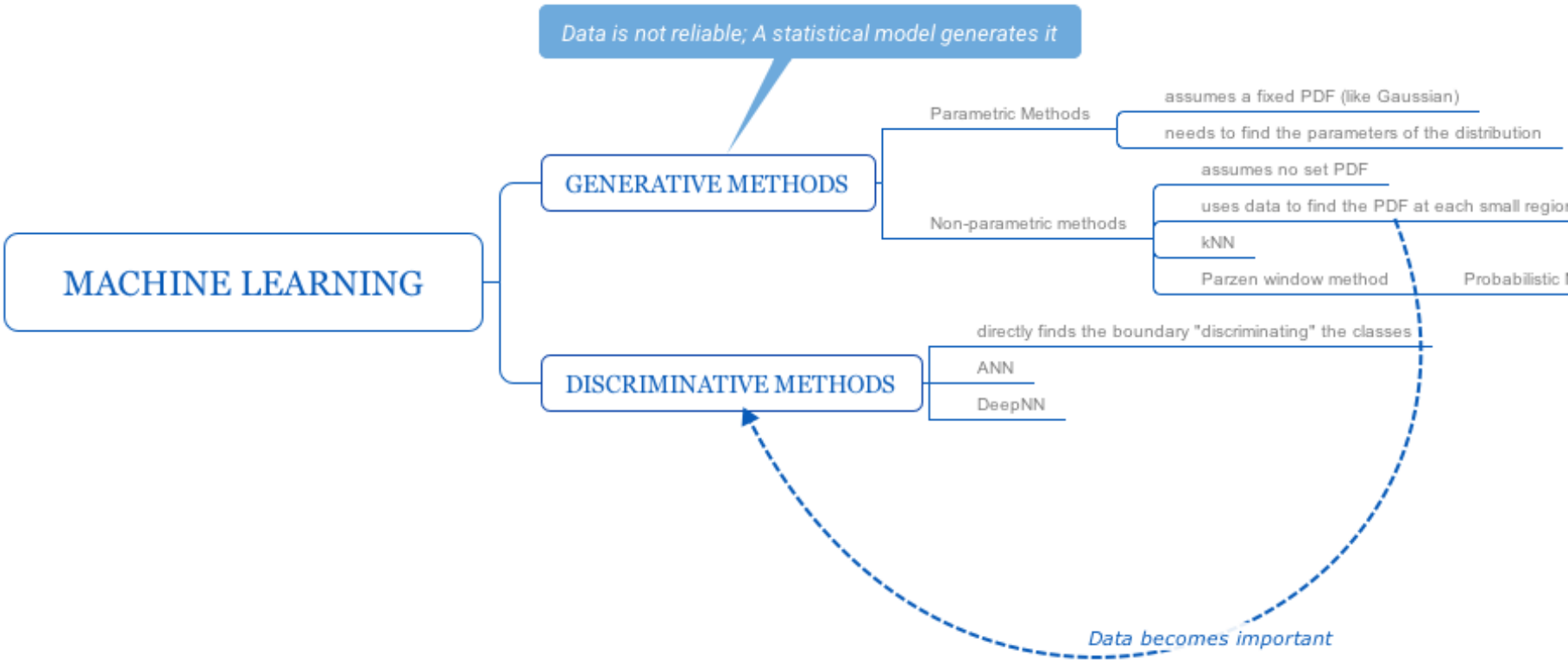
Introduction

- Human beings are lazy and commit errors!
- Human level intelligence will need human level error! (Turing)
- Machines are reliable and can have way more dimensions than "we can understand"

Introduction

- Human beings are lazy and commit errors!
- Human level intelligence will need human level error! (Turing)
- Machines are reliable and can have way more dimensions than "we can understand"
- We need to see things to understand (whether to extract new information or to classify known classes)
- We shall discuss some basic ways to make big-data small (!) enough to visualise them
- We shall present an open source tool we have developed which anyone can use to view large datasets

Taxonomy of Machine Learning



Squeezing the information out!-I

- E.g. smart conference hall heating (lets say we use 20 heat-sensors)

Squeezing the information out!-I

- E.g. smart conference hall heating (lets say we use 20 heat-sensors)
- What is the source of the 20-D data?
- "may be" the sources are pockets of attendees

Squeezing the information out!-I

- E.g. smart conference hall heating (lets say we use 20 heat-sensors)
- What is the source of the 20-D data?
- "may be" the sources are pockets of attendees
- few underlying "factor" determine the large dimension data
- Factor analysis (as old as statistics!)

Discrete Karhunen-Loeve expansion

- Let $\mathbf{x}$ be n dimensional random vector (of data from sensors); represented in terms of n independent vectors:
  $\mathbf{x} = y_1\phi_1 + y_2\phi_2 + ... = \sum_i y_i\phi_i$ (like coordinate systems)
- Lets calculate only m $(m < n)$ y_is, and use constants for the rest
- $\hat{\mathbf{X}}(m) = \sum_{i=1}^{m} \mathbf{y}_i\phi_i + \sum_{i=m+1}^{n} b_i\phi_i$

Discrete Karhunen-Loeve expansion

- Let $\mathbf{x}$ be n dimensional random vector (of data from sensors); represented in terms of n independent vectors:
  $\mathbf{x} = y_1\phi_1 + y_2\phi_2 + ... = \sum_i y_i\phi_i$ (like coordinate systems)
- Lets calculate only m $(m < n)$ y_is, and use constants for the rest
- $\hat{\mathbf{X}}(m) = \sum_{i=1}^{m} \mathbf{y}_i\phi_i + \sum_{i=m+1}^{n} b_i\phi_i$
- Error: $\Delta\mathbf{X}(m) = \sum_{i=m+1}^{n}(\mathbf{y}_i - b_i)\phi_i$ (prove this)
- Minimize the mean-squared error, i.e. $E(||\Delta\mathbf{X}(m)||^2)$ (WHY?)

Discrete Karhunen-Loeve expansion

- Let $\mathbf{x}$ be n dimensional random vector (of data from sensors); represented in terms of n independent vectors: $\mathbf{x} = y_1\phi_1 + y_2\phi_2 + ... = \sum_i y_i\phi_i$ (like coordinate systems)
- Lets calculate only m $(m < n)$ y_is, and use constants for the rest
- $\hat{\mathbf{X}}(m) = \sum_{i=1}^{m} \mathbf{y}_i\phi_i + \sum_{i=m+1}^{n} b_i\phi_i$
- Error: $\Delta\mathbf{X}(m) = \sum_{i=m+1}^{n}(\mathbf{y}_i - b_i)\phi_i$ (prove this)
- Minimize the mean-squared error, i.e. $E(||\Delta\mathbf{X}(m)||^2)$ (WHY?)
- Optimal choice of ϕ_is, satisfy $\Sigma_X\phi_i = \lambda_i\phi_i$, i.e. eigenvectors of the covariance matrix Σ_X (KL transformation or PCA)

Linear Discriminant Analysis (LDA)

- Intuition: for classification clusters of different classes should be far off and clusters as such should be crowded (Just MSE is not enough)

Linear Discriminant Analysis (LDA)

- Intuition: for classification clusters of different classes should be far off and clusters as such should be crowded (Just MSE is not enough)
- Interclass (betweenclass) scattering:
  $\mathbf{S}_b = \sum_{j=1}^{c} (\mathbf{m}_j - \mathbf{m})(\mathbf{m}_j - \mathbf{m})^T$
- Intraclass (withinclass) covariance matrix :
  $\mathbf{S}_w = \sum_{j=1}^{c} \sum_{i=1}^{N_j} (\mathbf{x}_i^j - \mathbf{m}_j)(x_i^j - \mathbf{m}_j)^T$
- $\mathbf{m}_j = \frac{1}{N_j} \sum_{i=1}^{N_j} \mathbf{x}_i^j$

Linear Discriminant Analysis (LDA)

- Intuition: for classification clusters of different classes should be far off and clusters as such should be crowded (Just MSE is not enough)
- Interclass (betweenclass) scattering:
  $\mathbf{S}_b = \sum_{j=1}^{c} (\mathbf{m}_j - \mathbf{m})(\mathbf{m}_j - \mathbf{m})^T$
- Intraclass (withinclass) covariance matrix :
  $\mathbf{S}_w = \sum_{j=1}^{c} \sum_{i=1}^{N_j} (\mathbf{x}_i^j - \mathbf{m}_j)(x_i^j - \mathbf{m}_j)^T$
- $\mathbf{m}_j = \frac{1}{N_j} \sum_{i=1}^{N_j} \mathbf{x}_i^j$
- LDA maximizes the ratio: $\frac{det[\mathbf{S}_b]}{det[\mathbf{S}_w]}$
- Solution: Eigen vectors of $\mathbf{S}_w^{-1}\mathbf{S}_b$

Squeezing the information out!-II

- Look into tweeter feed to predict flood intensity! (A crazy hypothesis)

Squeezing the information out!-II

- Look into tweeter feed to predict flood intensity! (A crazy hypothesis)
- Not easy to know what controls what: can we reduce the dimension keeping as much fidelity as possible?
- We want to "see" the data the way it looked in N-dimension (how?)

Squeezing the information out!-II

- Look into tweeter feed to predict flood intensity! (A crazy hypothesis)
- Not easy to know what controls what: can we reduce the dimension keeping as much fidelity as possible?
- We want to "see" the data the way it looked in N-dimension (how?)
- Lets try to preserve the "structure"

Sammon's Mapping

- Structure (in Shannon's sense): geometrical relationships among subsets of the data vectors
- Original data: N data points of L dimension
- Problem: to find best N points in a lower d (2 or 3) dimension, s.t. their interpoint distances approximate the corresponding interpoint distance in the original L-dimensional space

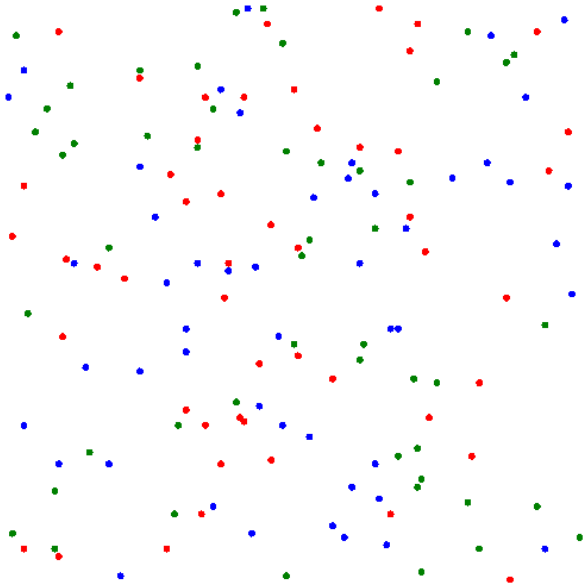
Sammon's Mapping

- Structure (in Shannon's sense): geometrical relationships among subsets of the data vectors
- Original data: N data points of L dimension
- Problem: to find best N points in a lower d (2 or 3) dimension, s.t. their interpoint distances approximate the corresponding interpoint distance in the original L-dimensional space
- Distance between two points in the original space: $d_{ij}^{*} \equiv dist[X_i, X_j]$
- Distance between two points in the mapped space: $d_{ij} \equiv dist[Y_i, Y_j]$
- Minimize interpoint distance; define the error function as: $E = \frac{1}{\sum_{i<j}[d_{ij}^{*}]} \sum_{i<j}^{N} \frac{[d_{ij}^{*} - d_{ij}]^2}{d_{ij}^{*}}$

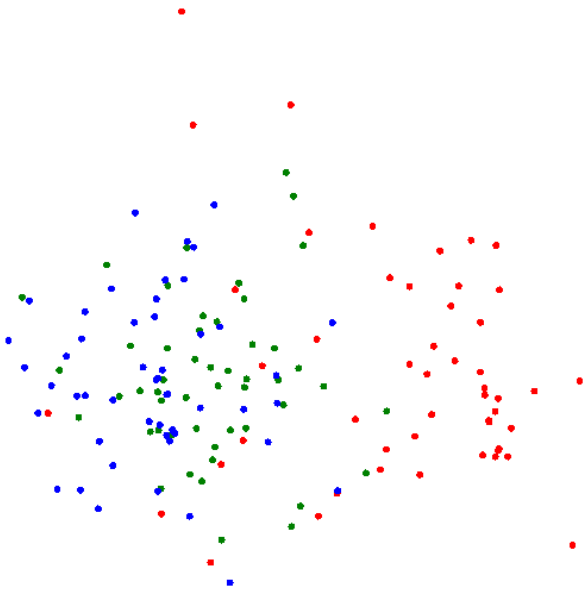
Sammon's Mapping

- Structure (in Shannon's sense): geometrical relationships among subsets of the data vectors
- Original data: N data points of L dimension
- Problem: to find best N points in a lower d (2 or 3) dimension, s.t. their interpoint distances approximate the corresponding interpoint distance in the original L-dimensional space
- Distance between two points in the original space: $d_{ij}^{*} \equiv dist[X_i, X_j]$
- Distance between two points in the mapped space: $d_{ij} \equiv dist[Y_i, Y_j]$
- Minimize interpoint distance; define the error function as: $E = \frac{1}{\sum_{i<j}[d_{ij}^{*}]} \sum_{i<j}^{N} \frac{[d_{ij}^{*} - d_{ij}]^2}{d_{ij}^{*}}$
- Good news: Its convex

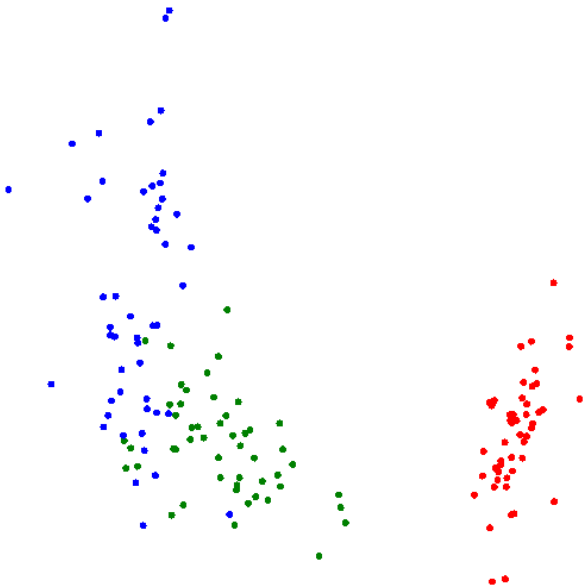
Sammon mapping: 0-iteration



Sammon mapping: 1-iteration



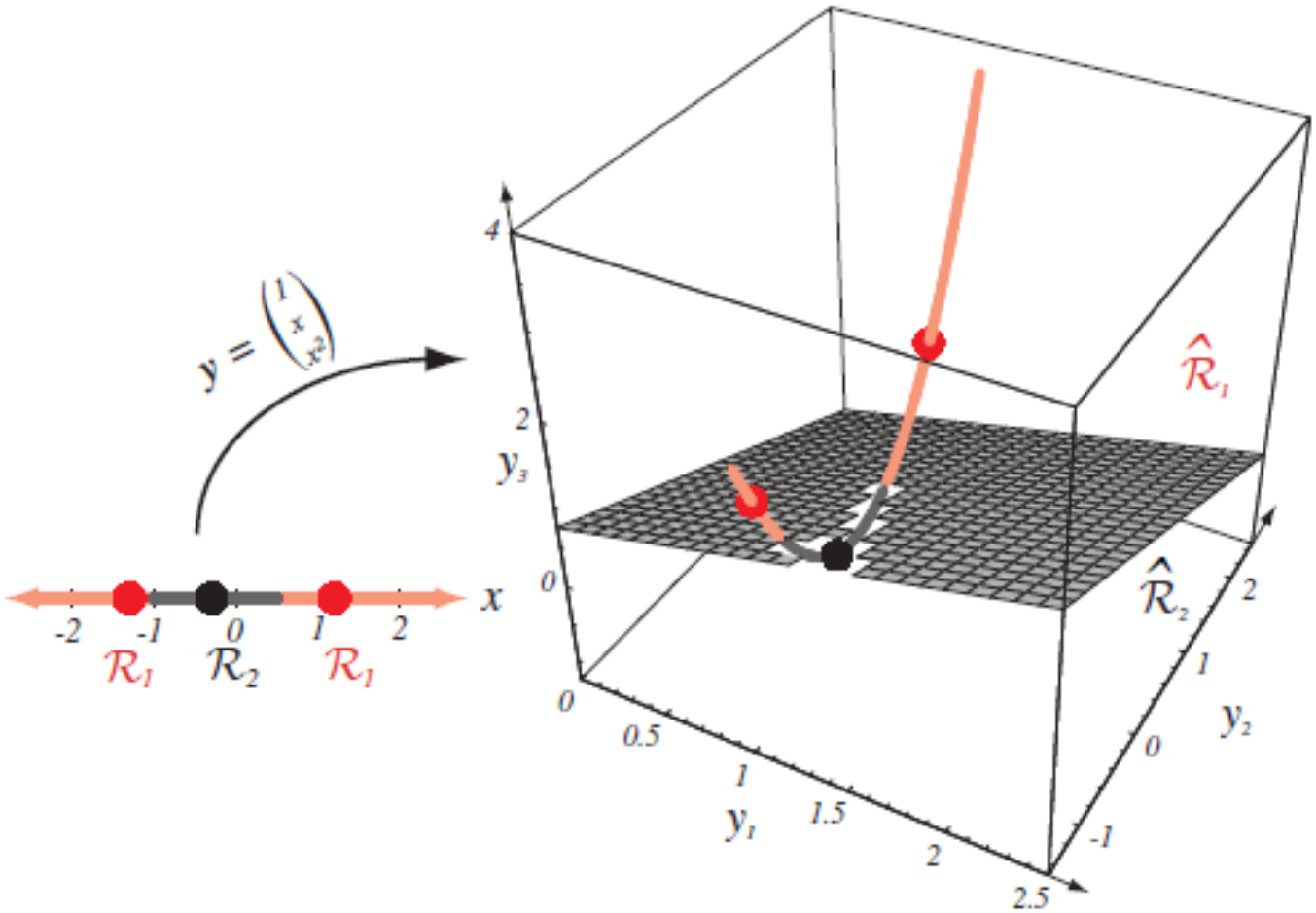
Sammon mapping: 10-iteration



Squeezing the information out!-III

- What if the original data is ill-behaved?
- Boundaries with curves (curves are not pretty here!)

Nonlinear turning into linear in higher dimensional space



kernel+PCA

- Lets non-linearize our data: x becomes $\phi(x)$ and y becomes $\phi(y)$

kernel+PCA

- Lets non-linearize our data: x becomes $\phi(x)$ and y becomes $\phi(y)$
- Designer choice of $\phi()$ s.t. $\phi(x)^T\phi(y) = k(x, y)$
- How we perform PCA?

kernel+PCA

- Lets non-linearize our data: x becomes $\phi(x)$ and y becomes $\phi(y)$
- Designer choice of $\phi()$ s.t. $\phi(x)^T\phi(y) = k(x, y)$
- How we perform PCA?
- Eigen values and functions of $\Sigma_X = \frac{1}{M-1} X.X^T$; M data points, each of size $N \times 1$
- $\Sigma = \frac{1}{M-1} \sum_{i=1}^{M} X_i X_i^T$ (any ideas?)

kernel+PCA

- Lets non-linearize our data: x becomes $\phi(x)$ and y becomes $\phi(y)$
- Designer choice of $\phi()$ s.t. $\phi(x)^T\phi(y) = k(x, y)$
- How we perform PCA?
- Eigen values and functions of $\Sigma_X = \frac{1}{M-1} X.X^T$; M data points, each of size $N \times 1$
- $\Sigma = \frac{1}{M-1} \sum_{i=1}^{M} X_i X_i^T$ (any ideas?)
- Lets pass the data through a nonlinear mapping ϕ; then the new $\Sigma_{X,\phi} = \frac{1}{M-1} \sum_{i=1}^{M} \phi(X_i)\phi(X_i)^T = \frac{1}{M-1} \sum_{i=1}^{M} k(X, X)$

Take-homes

- Data visualisation is extremely essential before we design any algorithm (we only have our eyes)

Take-homes

- Data visualisation is extremely essential before we design any algorithm (we only have our eyes)
- A range of tools to do so
- We give you a free suite of tools