IAML: Dimensionality Reduction

Victor Lavrenko and Charles Sutton
School of Informatics

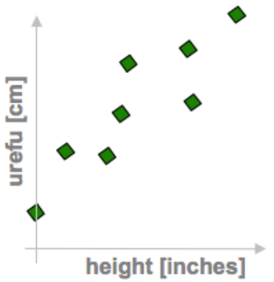
Semester 1

Overview

- Curse of dimensionality
- Different ways to reduce dimensionality
- Principal Components Analysis (PCA)
- Examples: Eigen Faces, Topics in Text
- PCA for classification
- Witten & Frank section 7.3
  - only the PCA section required

True vs. observed dimensionality

- Get a population, predict some property
  - instances represented as {urefu, height} pairs
  - what is the dimensionality of this data?



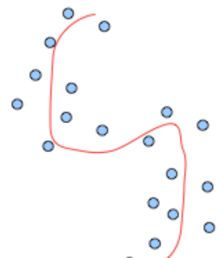
"height" = "urefu" in Swahili

- Data points over time from different geographic areas over time:
  - X_1: # of traffic accidents
  - X_2: # of burst water pipes
  - X_3: snow-plow expenditures
  - X_4: # of forest fires
  - X_5: # patients with heat stroke

Temperature below freezing?

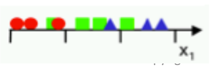
Curse of dimensionality

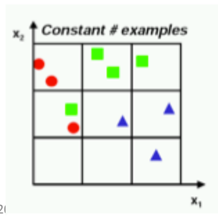
- Datasets typically high dimensional
  - vision: 10^4 pixels, text: 10^6 words
    - the way we observe / record them
  - true dimensionality often much lower
    - a manifold (sheet) in a high-d space
- Example: handwritten digits
  - 20 x 20 bitmap: $\{0,1\}^{400}$ possible events
    - will never see most of these events
    - actual digits: tiny fraction of events
  - true dimensionality:
    - possible variations of the pen-stroke



Curse of dimensionality (2)

- Machine learning methods are statistical by nature
  - count observations in various regions of some space
  - use counts to construct the predictor f(x)
  - e.g. decision trees: p_+/p_- in {o=rain,w=strong,T>28°}
  - text: #documents in {"hp" and "3d" and not "$" and …)
- As dimensionality grows: fewer observations per region
  - 1d: 3 regions, 2d: 3^2 regions, 1000d – hopeless
  - statistics need repetition
    - flip a coin once → head
    - P(head) = 100%?





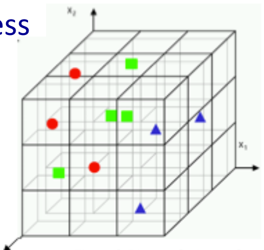
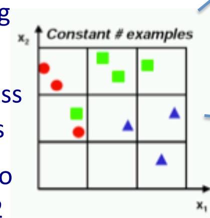
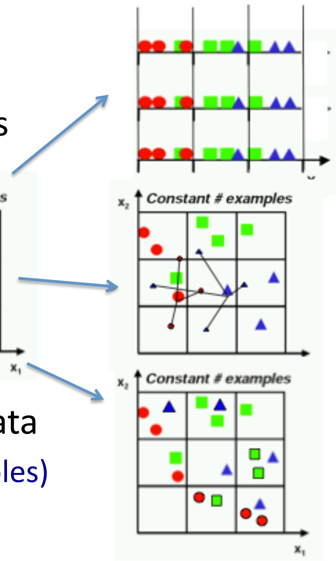


Figure © Ricardo Gutierrez-Osuna

Dealing with high dimensionality

- Use domain knowledge
  - feature engineering: SIFT, MFCC
- Make assumption about dimensions
  - independence: count along each dimension separately
  - smoothness: propagate class counts to neighboring regions
  - symmetry: e.g. invariance to order of dimensions: x1 ⇔ x2
- Reduce the dimensionality of the data
  - create a new set of dimensions (variables)

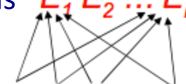




Dimensionality reduction

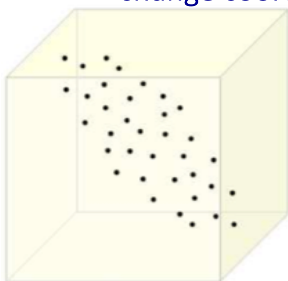
- Goal: represent instances with fewer variables
  - try to preserve as much structure in the data as possible
  - discriminative: only structure that affects class separability
- Feature selection
  - pick a subset of the original dimensions X_1 X_2 X_3 … X_{d-1} X_d
  - discriminative: pick good class "predictors" (e.g. gain)
- Feature extraction
  - construct a new set of dimensions E_1 E_2 … E_m

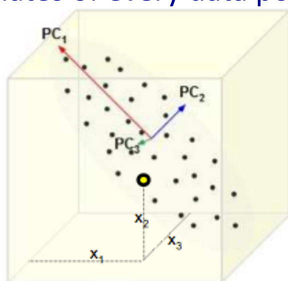
    $E_i = f(X_1 \ldots X_d)$
  - (linear) combinations of original X_1 X_2 X_3 … X_d

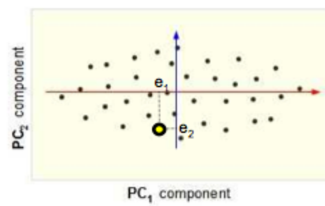


Principal Components Analysis

- Defines a set of principal components
  - 1st: direction of the greatest variability in the data
  - 2nd: perpendicular to 1st, greatest variability of what's left
  - ... and so on until d (original dimensionality)
- First *m* components become *m* new dimensions
  - change coordinates of every data point to these dimensions

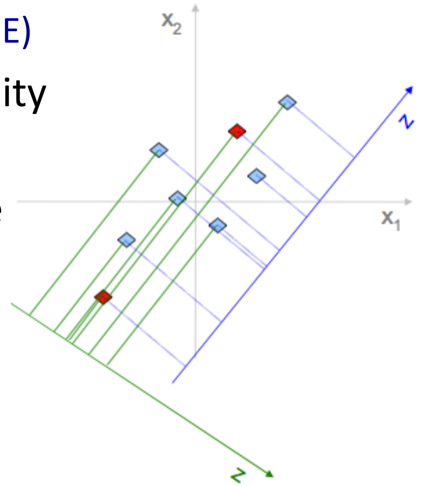




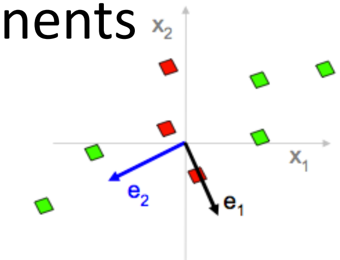


Why greatest variability?

- Example: reduce 2-dimensional data to 1-d
  - $\{x_1,x_2\} \rightarrow$ e (along new axis E)
- Pick E to maximize variability
- Reduces cases when two points are close in e-space but very far in (x,y)-space
- Minimizes distances between original points and their projections



Principal components



- Compute covariance matrix Σ
  - covariance of dimensions x_1 and x_2:
    - do x_1 and x_2 tend to increase together?
    - or does x_2 decrease as x_1 increases?
  - covariance: measure of variability

	x_1	x_2
x_1	2.0	0.8
x_2	0.8	0.6

$$\mathrm{var}(x_2) = \frac{1}{N}\sum_{i=1}^{n}(x_{2,i}-\mu_2)^2$$

$$\mathrm{cov}(x_1,x_2) = \frac{1}{N}\sum_{i=1}^{n}(x_{1,i}-\mu_1)(x_{2,i}-\mu_2)$$

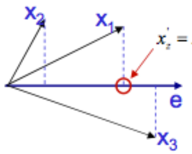
- Find the basis of Σ
  - find vectors e_i which aren't turned by Σ
    - $\Sigma\, e_i = \lambda_i\, e_i$: eigenvalue / eigenvector
  - 1^{st} PC: "longest" e_i (has largest λ_i), 2^{nd} PC: next longest, …

λ_1	0.26
λ_2	2.42

	e_1	e_2
x	0.4	-0.9
y	-0.9	-0.4

Direction of greatest variability

- Select dimension e which maximizes the variance
- Points x "projected" onto vector e:



$$x_z' = x^T e = \sum_{j=1}^{d} x_{ij} e_j$$

- Variance of projections: $\frac{1}{n}\sum_{i=1}^{n}\left(\sum_{j=1}^{d} x_{ij}e_j - \mu\right)^2 = \frac{1}{n}\sum_{i=1}^{n}\left(\sum_{j=1}^{d} x_{ij}e_j\right)^2$
- Maximize variance
  - want unit length: ||e||=1
  - add Lagrange multiplier

$$L = \frac{1}{n}\sum_{i=1}^{n}\left(\sum_{j=1}^{d} x_{ij}e_j\right)^2 - \lambda\left(\left(\sum_{k=1}^{d} e_j^2\right) - 1\right)$$

$$\frac{\partial L}{\partial e_a} = \frac{2}{n}\sum_{i=1}^{n}\left(\sum_{j=1}^{d} x_{ij}e_j\right)x_{ia} - 2\lambda e_a = 0$$

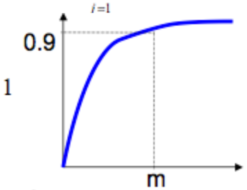
$$0 = 2\sum_{j=1}^{d} e_j \underbrace{\left(\frac{1}{n}\sum_{i=1}^{n} x_{ia}x_{ij}\right)}_{\text{covariance}} - 2\lambda e_a$$

$$\sum_{j=1}^{d}\mathrm{cov}(a,j)e_j = \lambda e_a \quad \text{for } a = 1\ldots d$$

$\Sigma e = \lambda e$ → e must be an eigenvector

Projecting to new dimensions

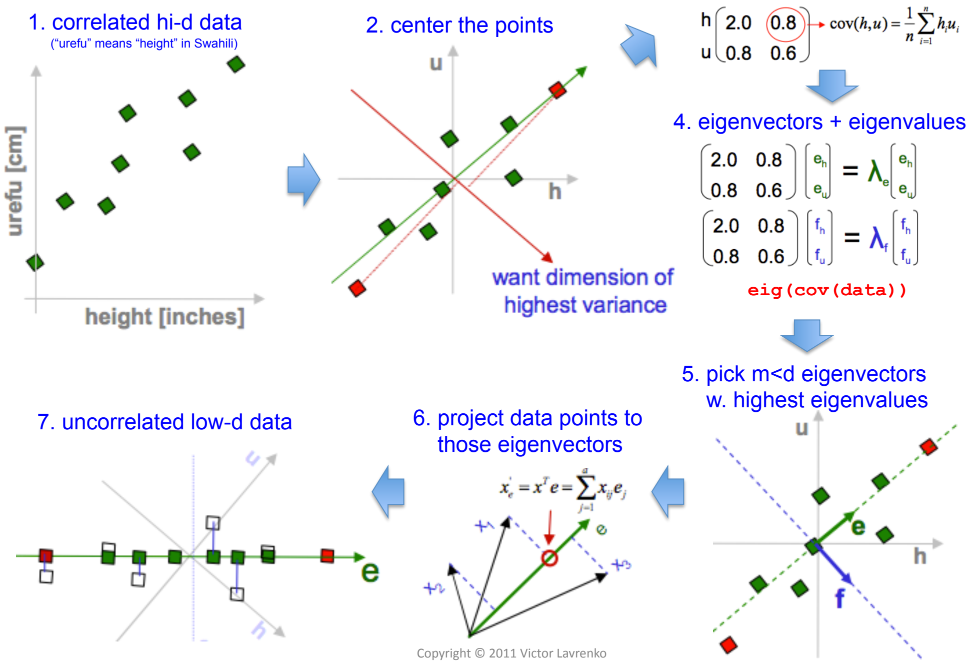
- Got a set of principal components $e_1 \ldots e_d$
  - orthogonal, unit length
  - corresponding eigenvalues $\lambda_1 \ldots \lambda_d$: $\sum_{i=1}^{d}\lambda_i = trace(\Sigma) = \sum_{i=1}^{d}\mathrm{cov}(X_j, X_j)$
  - fraction of variation explained by first *m* principal components $\frac{\sum_{i=1}^{m}\lambda_i}{\sum_{i=1}^{d}\lambda_i} \le 1$
    - typical threshold values: 0.9 or 0.95



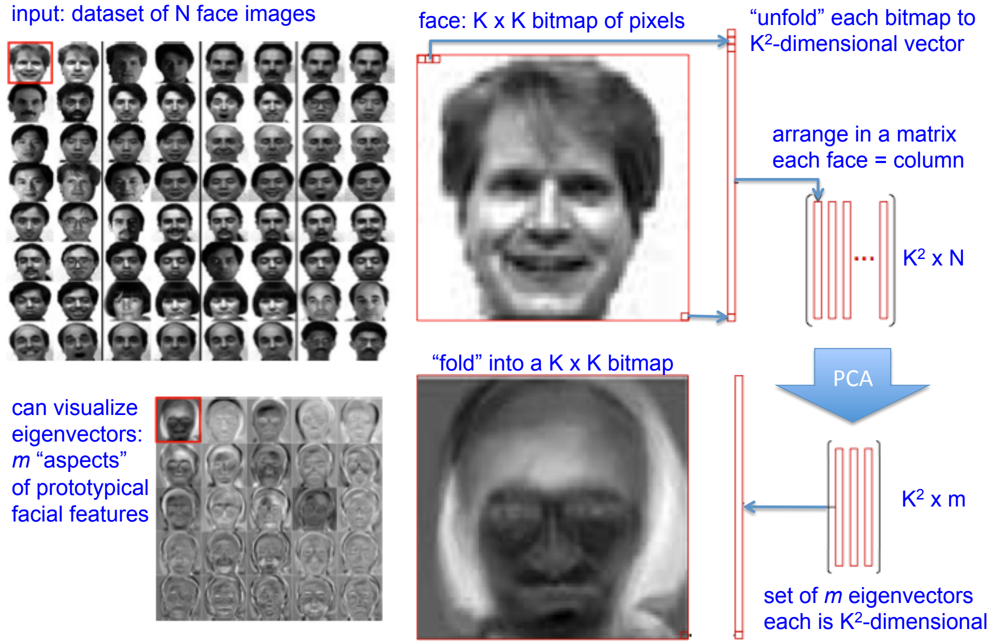
- $e_1 \ldots e_m$ are new dimension vectors
- Change coordinates: $x_{1..d} \rightarrow x'_{1..m}$
  - subtract mean from old dimensions
  - dot product each dimension with $e_1 \ldots e_m$

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \\ \vdots \\ x_d \end{bmatrix} \Rightarrow \begin{bmatrix} x_1' = \sum_{j=1}^{d} e_{1,j}x_j \\ x_2' = \sum_{j=1}^{d} e_{2,j}x_j \\ \vdots \\ x_m' = \sum_{j=1}^{d} e_{m,j}x_j \end{bmatrix}$$

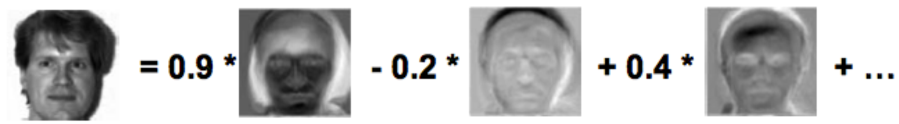
PCA in a nutshell



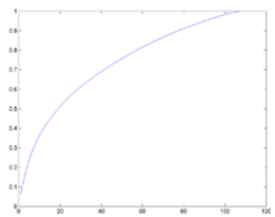
PCA example: Eigen Faces



Eigen Faces: Projection



- Project new face to space of eigen-faces
- Represent vector as a linear combination of principal components
- How many do we need?



(Eigen) Face Recognition

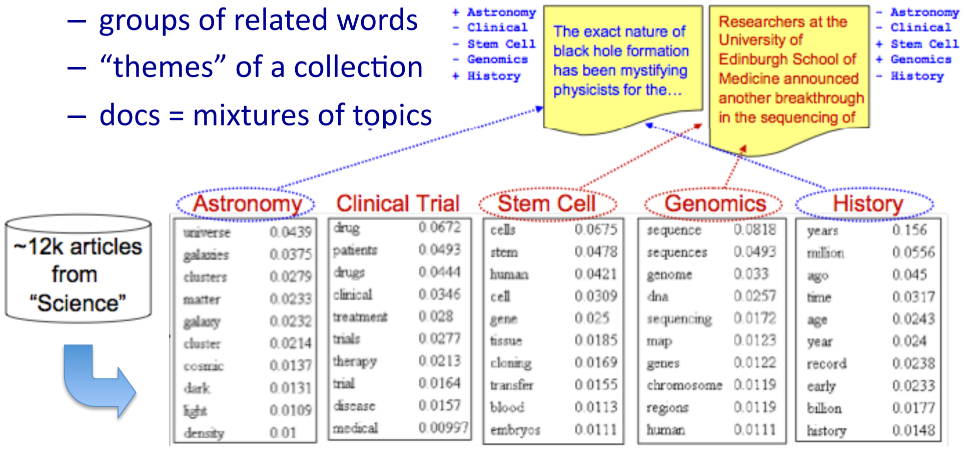
- Face similarity
  - in the reduced space
  - insensitive to lighting expression, orientation
- Projecting new "faces"
  - everything is a face

new face (not in training)

projected to eigenfaces

PCA example: Topics in Text

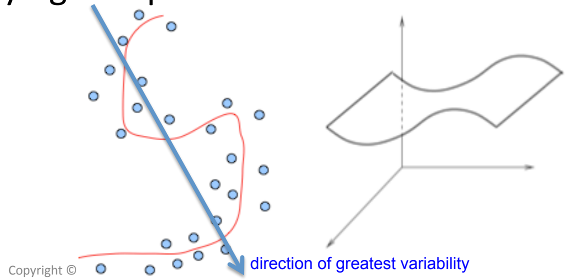
- Can run variants of PCA on news, scientific papers
- Eigenvectors can be interpreted as “topics”
  - groups of related words
  - “themes” of a collection
  - docs = mixtures of topics



Astronomy		Clinical Trial		Stem Cell		Genomics		History	
universe	0.0439	drug	0.0672	cells	0.0675	sequence	0.0818	years	0.156
galaxies	0.0375	patients	0.0493	stem	0.0478	sequences	0.0493	million	0.0556
clusters	0.0279	drugs	0.0444	human	0.0421	genome	0.033	ago	0.045
matter	0.0233	clinical	0.0346	cell	0.0309	dna	0.0257	time	0.0317
galaxy	0.0232	treatment	0.028	gene	0.025	sequencing	0.0172	age	0.0243
cluster	0.0214	trials	0.0277	tissue	0.0185	map	0.0123	year	0.024
cosmic	0.0137	therapy	0.0213	cloning	0.0169	genes	0.0122	record	0.0238
dark	0.0131	trial	0.0164	transfer	0.0155	chromosome	0.0119	early	0.0233
light	0.0109	disease	0.0157	blood	0.0113	regions	0.0119	billion	0.0177
density	0.01	medical	0.00997	embryos	0.0111	human	0.0111	history	0.0148

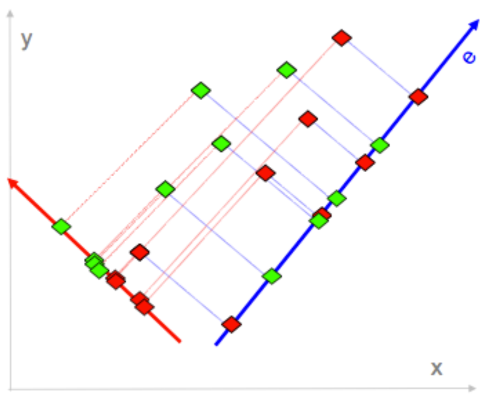
PCA: practical issues

- Covariance extremely sensitive to large values
  - multiply some dimension by 1000
    - dominates covariance
    - becomes a principal component
  - normalize each dimension to zero mean and unit variance:
    $x' = (x - \text{mean}) / \text{st.dev}$
- PCA assumes underlying subspace is linear
  - 1d: straight line
    2d: flat sheet
  - transform to handle non-linear spaces (manifolds)



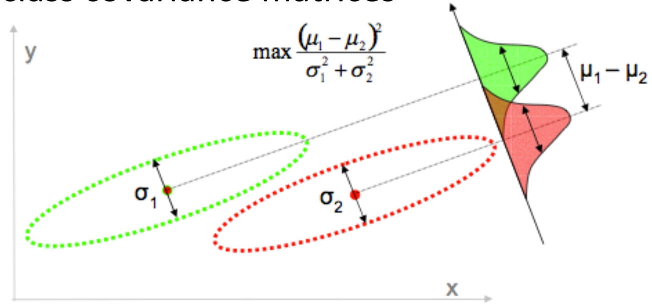
PCA and classification

- PCA is unsupervised
  - maximizes overall variance of the data along a small set of directions
  - does not know anything about class labels
  - can pick direction that makes it hard to separate classes
- Discriminative approach
  - look for a dimension that makes it easy to separate classes



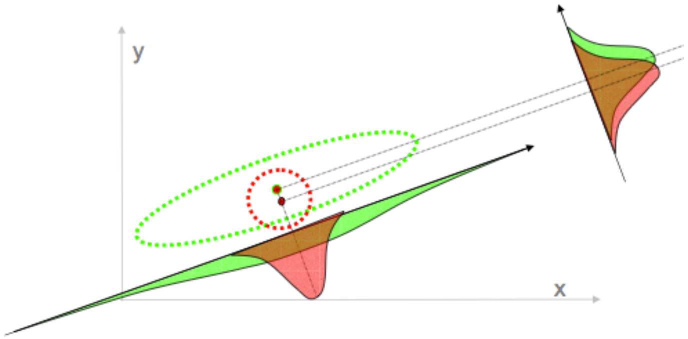
Linear Discriminant Analysis

- LDA: pick a new dimension that gives:
  - maximum separation between means of projected classes
  - minimum variance within each projected class
- Solution: eigenvectors based on between-class and within-class covariance matrices



PCA vs. LDA

- LDA not always good for classification
  - assumes classes are unimodal Gaussians
  - fails when discriminatory information is not in the mean, but in the variance of the data
- PCA may give a more discriminating projection



Dimensionality reduction

- Pros
  - reflects our intuitions about the data
  - allows estimating probabilities in high-dimensional data
    - no need to assume independence etc.
  - dramatic reduction in size of data
    - faster processing (as long as reduction is fast), smaller storage
- Cons
  - too expensive for many applications (Twitter, web)
  - disastrous for tasks with fine-grained classes
  - understand assumptions behind the methods (linearity etc.)
    - there may be better ways to deal with sparseness

Summary

- True dimensionality << observed dimensionality
- High dimensionality ➔ sparse, unstable estimates
- Dealing with high dimensionality:
  - use domain knowledge
  - make an assumption: independence / smoothness / symmetry
  - dimensionality reduction: feature selection / feature extraction
- Principal Components Analysis (PCA)
  - picks dimensions that maximize variability
    - eigenvectors of the covariance matrix
  - examples: Eigen Faces, Topics in Text
  - variant for classification: Linear Discriminant Analysis