Machine Learning Lecture 11
Features and dimensionality reduction

Dr. Ioannis Patras

Slides thanks: Tim Hospedales

1

Overview

- Feature Design
- Why Reduce Dimensions?
- Feature Selection and Methods
  - Filtering
  - Wrapper
  - Built-in
- Dimensionality Reduction
  - PCA
- Summary

2

Overview

- Feature Design ← How to choose/design your features
- Why Reduce Dimensions?
- Feature Selection and Methods
  - Filtering ← How to prune unhelpful features
  - Wrapper
  - Built-in
- Dimensionality Reduction ← How to to compress columns
  - PCA
- Summary

3

What do we mean by 'Features'

- Input database columns, aka Attributes, Dimensions.
- What data to include as the input to your learning procedure
  - Sometimes choice is out of your scope or obvious from domain/business setting
  - Sometimes choice is an opportunity for engineering / intuition
  - (Given that more attributes potentially provide more information, but potentially increase overfitting, memory and slow the computation)

4

Feature Choice

- E.g., You work at Zoopla, a real-estate web-site.
  - Your job is to add a feature that predicts prices for listed houses
  - What should you include?
  - Rooms, m^2 Area, Postcode, Distance to Transport/School, Crime…?
  - Floor type? Date of construction? Room shape? Wallpaper color…? Number of internal doors?
- Using linear/non-linear regression, you will find that every feature added increases R^2 / decreases RMSE.
  - How to decide when to stop?
  - Continuous increase of fit with irrelevant features due to over fitting
    - An option is to use cross-validation to decide if which features

5

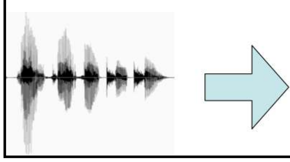
Feature Engineering

- Input and Output for Machine Learning Algorithm
  - $\{\mathbf{x}_i=[x_1,\ldots,x_d], y_i\}$
- How to convert real life data:
  - Into $\mathbf{x}_i=[x_1,\ldots,x_d]$, y_i ?
  - What to store in your database?
- Domain Specific
  - Human Expertise

6

Feature Engineering

- It may make sense for your input to be some transformation of the raw data
  - Because your data may not be fixed length
  - Because the right non-linear transformation can make learning easier
  - Because the right low-dimensional transform could help avoid over-fitting.
- Designing this transformation:
  - Feature engineering



“Danger”

7

Feature Engineering: Examples

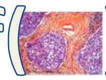
- E.g., Audio or accelerometer data
  - Full data is waveform
  - Probably don’t want to use directly: Variable and high dimensions
  - (E.g., 5sec/44KhZ/stereo: Half million columns.)
- Features:
  - High-amplitude count (1d) … Loud noise detector.
  - Zero-crossing count(1d) … Activity recognition.
  - Fourier Coefficients (e.g., 128d) … Music / Speech recognition
- Case Study @ EECS:
  - Activity + identity recognition on mobile phone accelerometer

8

Feature Engineering: Examples

- E.g., Images.
  - Full data is all the pixels
- Features:
  - Average brightness (1d) e.g., Day vs Night.
  - Color histogram (3 – 1000dims) e.g., Recognize objects
  - Histogram of Gradients (~128dim) e.g., Detect objects
  - Raw Pixels (10000dim+) e.g., Face recognition

f()=32yr

f()=cancer

f()=male

9

Feature Engineering: Examples

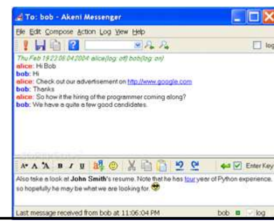
- E.g., Internet Advertising
  - Full data is everything on your facebook profile.
  - Should we show an add about premium baby-items?
  - Engineer a feature aggregating high income related likes+posts & a feature aggregating baby/mother related likes+posts
- Could result in a simpler & more efficient model than if you threw everything in

$\{\mathbf{x}_i=[x_1,\ldots,x_d], y_i\}$

10

Feature Engineering: Text

- Text is a heavily studied/mined domain
  - Naturally: High dimensional & variable length.
  - How to represent as fixed length and low(-ish) dimensional?
- One option is meta-features
  - Feature Engineering of Document:
    - Length, average word length, punctuation frequency, (in)correct spelling ratio, emoticon density, uppercase/lowercase ratio, etc.
    - (6 dims in this example)
- Case Study: Using meta-features:
  - IM/SMS authors can be recognised



11

Feature Engineering: Text

- The most common choice for text is "Bag of Words"
- E.g., Raw data:
  - "John likes to watch movies. Mary likes movies too."
    - John: 1, likes: 2, to: 1, ….
  - "John also likes to watch football games."
    - John: 1, also: 1, likes: 1,…
- Result is a "bag of words" vector f

```
[1, 2, 1, 1, 2, 0, 0, 0, 1, 1]
[1, 1, 1, 1, 0, 1, 1, 1, 0, 0]
```

  - Length: # of words in the dictionary.
  - Row: Frequency of each word in a document
  - Sum of a row: Number of words in corresponding document
- Can also use bi-grams, trigrams, n-grams

12

Case Studies

EECS Work ☺

- Using Bag of Words + Linear Classifiers, previous project students did….
  - Predict bill passage from bill text.
  - Predict politician identity from speeches.
  - Predict movie genre from script/subtitles
  - Predict TV show from script/subtitles
  - Predict actor/director/scriptwriter from script/subtitles
  - Predict amazon product review rating + helpfulness
  - Predict sales rank/price from product description

IMDb

Excellent product that I completely hate, Apr 1, 2013

By Thirsty - See all my reviews

This review is from: Strollmaster 3000 (Baby Product)

The Strollmaster 3000 is every parent's dream - roomy, durable, safe, and easy to fold, with a unique 17-point harness. Best yet, it weighs just 1.6 lbs. and sells for an unbelievable $17.99. Unfortunately, it has one fatal flaw - the cupholder can only handle beverages up to 64 oz. I was dumbstruck as well. Is this America? I was left holding my 128 oz. Big Gulp like some kind of sucker. So, if you're into amazing, durable products that are a steal and virtually idyllic, then, sure, buy it. If you want to down a bathtub of Dr. Pepper, though, I'd pass.

13

Feature Engineering

- Input may be some transformation of the raw data
  - Because your data may not be fixed length
  1. Because the right non-linear transformation make the problem easier
  2. Because the right low-dimensional transform could help avoid over-fitting.
- Sometimes cleverly derived features can simplify learning.
  - E.g., House price database has length+width of room.
  - => Linear regressor on area=L*W, non-linear regressor on length & width.
- But can have more features than original data
- Dichotomy between designing exactly the right feature (#2) and designing very many features (#1) above.
  - If we include/make many features in the hope of finding a good one….
  - We may not know which are relevant
  - Risk of over-fitting

14

Overview

- Feature Design
- Why Reduce Dimensions?
- Feature Selection and Methods
  - Filtering
  - Wrapper
  - Built-in
- Dimensionality Reduction
  - PCA
- Summary

16

Many Dimensions

- Suppose we are given or have designed our features. Then…
- We often end up with many dimensions
  - Because we over-killed on feature engineering
  - Because it's a problem where we have very little prior knowledge, so had no choice but to include everything.
    - E.g., drug discovery, genome analysis

17

Why Reduce Dimensions?

- "Curse of Dimensionality"
- Irrelevant Data
- Computation Time
- Visualization
- Interpretation

- Many applications have $> 10^6$ features (columns)

f()=spam

f()=32yr

18

Why Reduce Dimensions? "Curse of Dimensionality"

- Human intuition breaks down in high dimensions
  - E.g., Gaussian, Cube, Sphere-Cube.
  - Everything is similarly far away
- So do many machine learning algorithms…
  - Slow
  - Inaccurate
  - Makes overfitting very easy, so poor generalization
- E.g.,
  - KNN: Not robust to high dimensions.
  - Linear Regression. Need data > dimensions.

19

Why Reduce Dimensions? Irrelevant Data

- Curse of Dimensionality especially dangerous if
  - Many weakly relevant dimensions
  - Some very relevant, but many irrelevant dimensions
- Irrelevant dimensions, e.g.,:
  - Given article content: Classify sports versus technology.
  - Given someone's facebook profile, what product should I advertise to them?

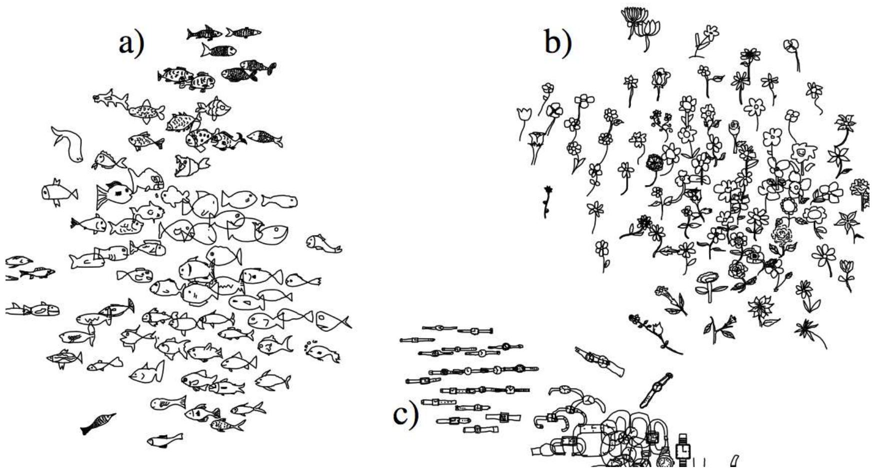
20

Why Reduce Dimensions? Computation Time

- We have seen:
  - Train: Regression $O(d^2n+d^3)$
  - Test: Regression, MaxEnt: O(nd), KNN O(nd).
  - (An $O(d^3)$ method will be 8x faster with ½ the features!)
- "Big Data" / Web-scale
  - N=10^6, d=10^6
  - => Reducing dimensions is critical
- Embedded systems & mobile apps
- Real-time apps

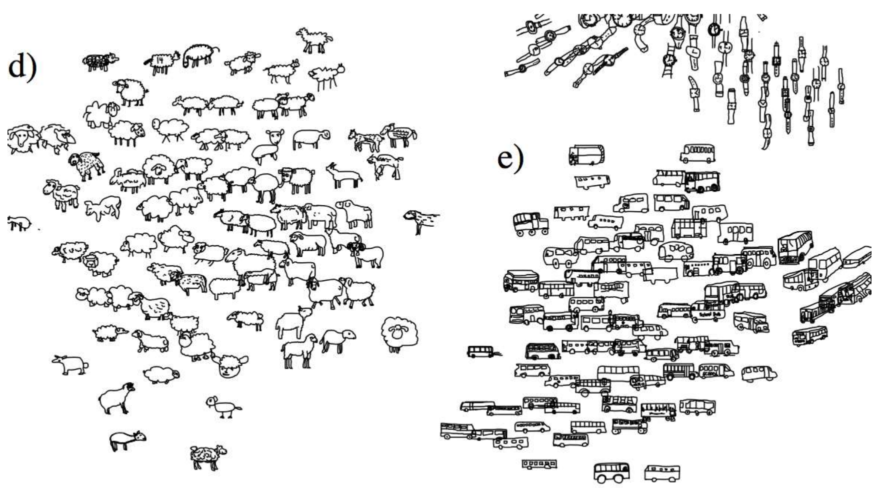
21

Why Reduce Dimensions? Visualization



22

Why Reduce Dimensions? Visualization



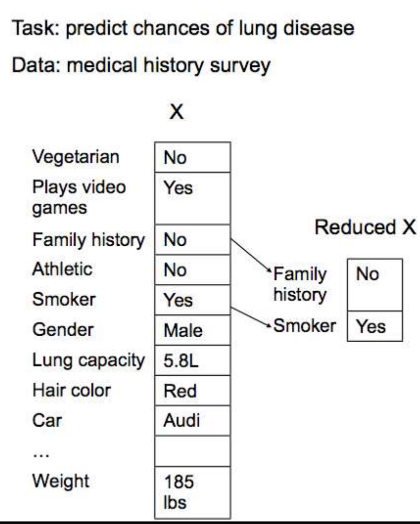
23

Why Reduce Dimensions? Interpretation

- Sometimes finding good dimensions is the fundamental aim
- E.g.: What causes a program to crash?
  - Features are aspects of a single program execution
    - Which branches were taken?
    - What values did functions return?
  - Classifier F(Trace): Crash or Not
  - Features that predict crashes well are probably bugs
- E.g.: What causes lung cancer?
  - Features are aspects of a patient's medical history
  - Binary response variable: did the patient develop lung cancer?
  - Want to legislate against features that predict lung cancer.

24

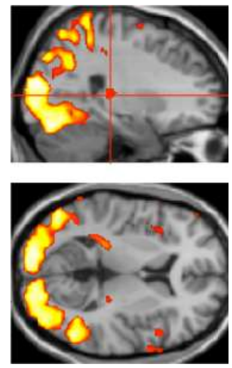
Why Reduce Dimensions? Interpretation

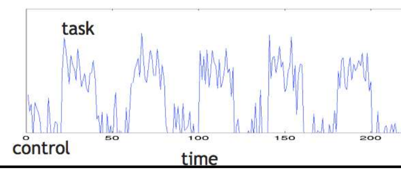


25

Case Study: fMRI Brain Imaging

- "Mind Reading" experiment
- Predict mental state from voxels
  - What are they seeing/reading/thinking?
- Interesting for both prediction and insight
  - Prediction: 'Mind reading'
    - E.g., Is the subject concealing information?
  - Insight: Which part of the brain does what
- Challenge:
  - 10-100 examples.
  - 1000k features
  - => d >> n
  - Most dimensions irrelevant.
  - Noise (scanner, body, subject)





26

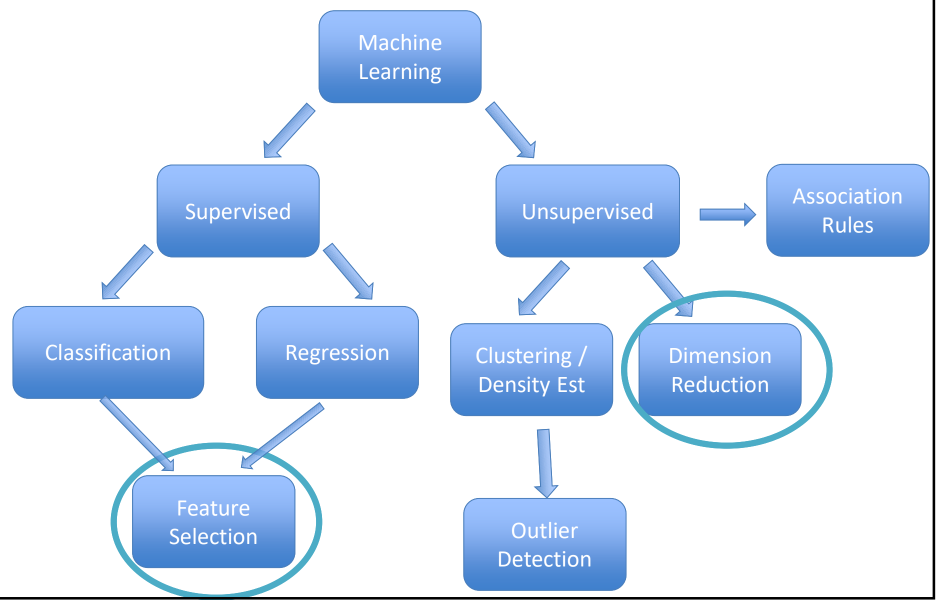
Reducing Dimension

Two categories of ways to reduce dimensions….

- Feature Selection
  - Pick a good subset of features (attributes, columns), ignore the others (typically supervised)
- Dimensionality Reduction by Linear Projection
  - Transform linear combination of all features to a smaller set of features (typically unsupervised)

27

A Taxonomy



28

Overview

- Feature Design
- Why Reduce Dimensions?
- Feature Selection and Methods
  - Filtering
  - Wrapper
  - Built-in
- Dimensionality Reduction
  - PCA
- Summary

29

Feature Selection Methods

- Three main types
  - Filtering
  - Wrapper
  - Embedded
- Formally:
- Want to learn y=f(**x**)
  - $\mathbf{x}=[x_1,..x_j,..x_d]$
  - Suspect not all x_j are relevant
  - Task: Find the relevant subset
  - Challenge: there are 2^d subsets!

30

Feature Selection Methods: Filtering

- Assign a score to each feature: s_j=score(j)
  - Sort features j by score, and pick the top K or top K%.
- Common scoring methods
  - Correlation between X_j and Y
  - Estimate the mutual information between X_j and Y

$$I(X;Y) = \sum_{y \in Y} \sum_{x \in X} p(x,y) \log\left(\frac{p(x,y)}{p(x)\,p(y)}\right)$$

  - χ^2 test of statistical independence between X_j and Y.
  - Domain Specific.
    - Text: Ignore words such as “the”, “it”.

31

Feature Selection Methods: Filtering

- Advantages:
  - Very fast
  - Simple to apply.
- Disadvantages?
  - Doesn't take into account feature interaction
  - => Apparently useless features can be useful when grouped together
    - It will miss these
- Practical:
  - Use light filtering as an initial step if training time is a big issue.

32

Feature Selection Methods: Wrapper

- Filter ignores features that can be useful in conjunction
  - Also doesn't account for limitations / power of learning algorithm.
- Wrapper methods:
  - For each subset of features:
    - Retrain learning algorithm on chosen subset
    - Evaluate learning algorithm on validation data
  - Pick the subset which has highest validation performance
- Issue:
  - Repeatedly retraining is costly
  - There are exponentially many (2^d) subsets of features.

33

Wrapper Methods: Greedy Search

Forward Selection

- Initialize no feats: fs={}
- Do:
  - Try all unused features s
  - Find s^* to add that improves performance the most
  - Add feature s* to fs.
- While: performance improving

34

Wrapper Methods: Greedy Search

Forward Selection

- Initialize no feats: fs={}
- Do:
  - Try all unused features s
  - Find s^* to add that improves performance the most
  - Add feature s* to fs.
- While: performance improving

Backward Selection

- Initialize fs={1,..,d}
- Do:
  - Try removing each feature in fs
  - Find s^* to remove which improves performance the most
  - Remove s^* from fs.
- While: performance improving.

Backward:
- Tends to find better models (interaction)
- Frequently too expensive

Both can be too greedy

35

Feature Selection Methods: Embedded

- Wrapper methods:
  - Advantage: can be applied to any model (model agnostic)
  - Disadvantage: suffer from repeated re-train cost and sub-optimality (greedy).
- In some special cases, feature selection can be built into a particular learning algorithm
  - Model specific
  - May be more efficient / optimal

36

Embedded Methods

- We have seen regularization, e.g., MaxEnt and regression
  - Find w=argmin E(w,D)
  - This is known as L2 regularization because it penalizes the squared weights

$$E_{MCLR}(\mathbf{w},D) = -\sum \log p(y_i \mid \mathbf{x}_i) + \lambda \mathbf{w}^T \mathbf{w} \qquad E_{MSER}(\mathbf{w},D) = \sum (y_i - f(\mathbf{x}_i))^2 + \lambda \mathbf{w}^T \mathbf{w}$$

- Suppose some dimension j of **x** is totally irrelevant
  - Suppose we remove it (setting w_j=0)
  - No effect on the first term
  - Improves the second term
- => Good regularization can help with feature selection.
  - But how to achieve it?

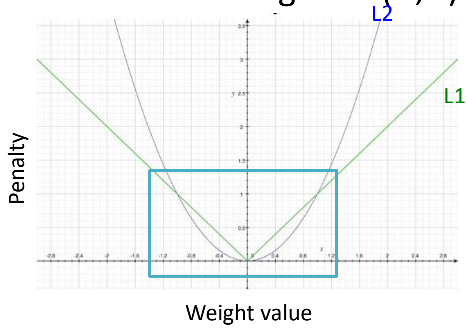
38

Embedded Methods

- We have seen regularization, e.g., MaxEnt and regression

$$E_{MCLR}(\mathbf{w}, D) = -\sum \log p(y_i \mid \mathbf{x}_i) + \lambda R(\mathbf{w}) \qquad E_{MSER}(\mathbf{w}, D) = \sum_i (y_i - f(\mathbf{x}_i))^2 + \lambda R(\mathbf{w})$$

  - Find w=argmin E(w,D)



$$R_2(\mathbf{w}) = \mathbf{w}^T\mathbf{w} = \sum w_j^2 = w_1^2 + \ldots + w_d^2$$

$$R_0(\mathbf{w}) = \sum I(w_j \neq 0)$$

$$R_1(\mathbf{w}) = \sum |w_j| = |w_1| + \ldots + |w_d|$$

39

Embedded Methods

- We have seen regularization, e.g., MaxEnt and regression
  - Find w=argmin E(w,D)

$$E_{MCLR}(\mathbf{w}, D) = -\sum \log p(y_i \mid \mathbf{x}_i) + R(\mathbf{w}) \qquad E_{MSER}(\mathbf{w}, D) = \sum_i (y_i - f(\mathbf{x}_i))^2 + R(\mathbf{w})$$

- L2 regularizer: “Ridge”
  - Fast and easy, but weak feature selection
- L0 regularizer is ideal
  - But very slow optimise (NP hard)
    - (because not differentiable)
- L1 regularizer: “Lasso”
  - Commonly chosen tradeoff.
  - Reasonably easy, reasonably quick

$$R_2(w) = \sum w_j^2 = w_1^2 + \ldots + w_d^2$$

$$R_0(w) = \sum I(w_j \neq 0)$$

$$R_1(w) = \sum |w_j| = |w_1| + \ldots + |w_d|$$

40

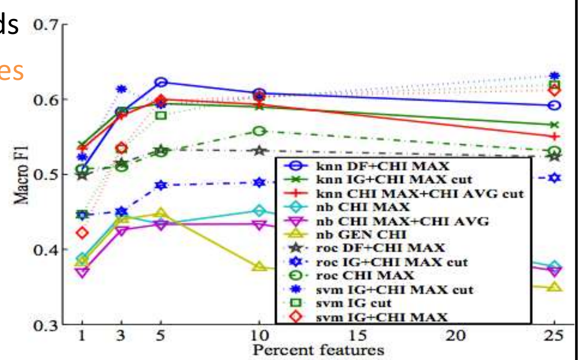
Feature Selection: When Can it Hurt?

- E.g., Fat tail in NLP
  - Many n-grams are seen only once in training.
  - 8-gram "Today I give a lecture on feature selection" only one in the mailbox, but a good predictor of WORK email.

41

Case Study: Classifying News Articles

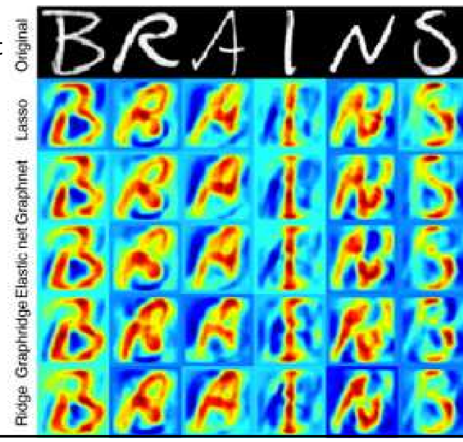
- Approximately 10^5 English words.
  - (10^{10} bigrams, etc.)
  - Reuters article benchmark: 1500
- Rogati et al, CIKM 2002
  - Study filter methods
- Best @ 3-5% of features



42

Case Study: fMRI Brain Imaging

- Schoenmakers et al, NeuroImage 2013
  - MRI Voxel => Pixel Regression.
    - (Attempt to “mind-read” what you see from your brain activity)
    - Input: Brain voxels
    - Output: Your visual field
  - Feature selection:
    - Most brain voxels irrelevant
    - But we don’t know which…
  - Best result with L1 “Lasso” regularization



43

Case Study: fMRI Brain Imaging

- Nishimoto et al, Curent Biology, 2011

Presented clip

Clip reconstructed from brain activity

44

Case Study: Regression for Super Resolution (EECS work! ☺)

- Super Resolution: Regression problem with:
  - Input: Low resolution image
  - Output: high-resolution image
- L1 "Sparse Coding" Feature Selection

45

Case Study: View Direction Classification (EECS Work! ☺)

- Random Forest
  - Built in feat selection

46

Summary

- Sometimes we want to engineer new features:
  - Raw data may not be suitable (e.g., variable length)
  - A good derived feature may simplify the problem.
  - A suitable set of features may be lower dimensional than raw data
- Sometimes we want to select features:
  - When there are many potential inputs, and little domain knowledge to select/engineer them
  - When there are resource constraints (large scale/embedded)
  - When we engineered many features in the hope of finding a good one
  - When the feature selection is itself the goal

49

Overview

- Feature Design
- Why Reduce Dimensions?
- Feature Selection and Methods
  - Filtering
  - Wrapper
  - Built-in
- Dimensionality Reduction
  - PCA
- Summary

50

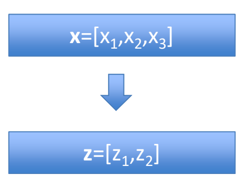
From Feature Selection to Dimensionality Reduction

- So far we selected a subset of good columns (feature selection)
  - We loose everything in the discarded columns.
- Sometimes we want to “compress” all the columns into a smaller number, but loosing the least possible information

51

Dimensionality Reduction: Overview

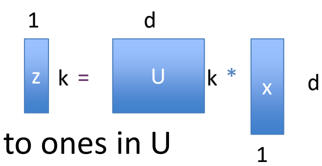
- Data **x**, |**x**|=d
- Derived Features **z**, |**z**|=k, k<d
  - $z_1=F_1(x) = x_1+x_2$
  - $z_2=F_2(x) = 2x_3-x_1-x_2$
- Feature Selection
  - $z_1=x_1$
  - $z_2=x_3$
- Dimensionality Reduction:
  - How to find a good linear combination of features?
  - Restrict ourselves to linear combinations for now
  - Supervised: Find a linear combination that helps achieve a task.
  - Unsupervised: Find a linear combination according to some other criteria



52

Dimensionality Reduction: Linear

- Linear combinations of features can be expressed as a matrix multiply
  - **z**=U**x**

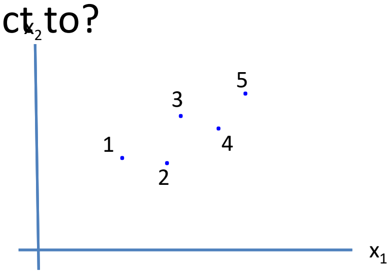


- E.g., U is a binary row
  - **z** is a subset of **x** according to ones in U
- E.g., U is a list of 1s
  - **z** is the sum of the elements in **x**
- Lots more options…
  - So how to find a "good" matrix U? Ideas?
  - Pick U that explains the data well / looses little information

53

Dimensionality Reduction: Geometric Intuition

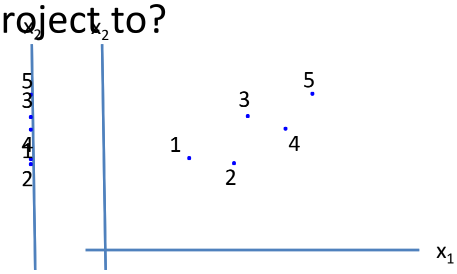
- d=2, k=1
- Which axis do we project to?
  - $z=x_1$?
  - $z=x_2$?

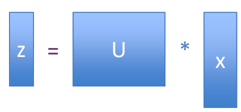


54

Dimensionality Reduction: Geometric Intuition

- d=2, k=1
- Which axis do we project to?
  - $z=x_1$?
  - $z=x_2$?

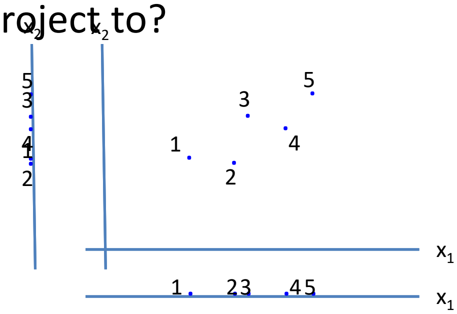




55

Dimensionality Reduction: Geometric Intuition

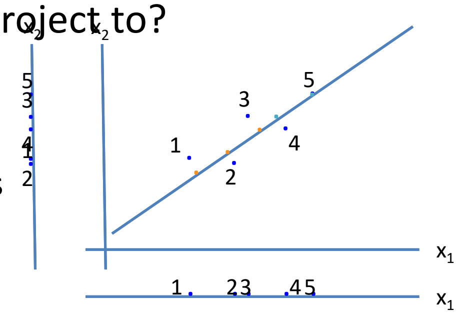
- d=2, k=1
- Which axis do we project to?
  - $z=x_1$?
  - $z=x_2$?

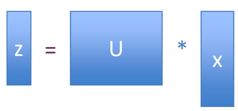


56

Dimensionality Reduction: Geometric Intuition

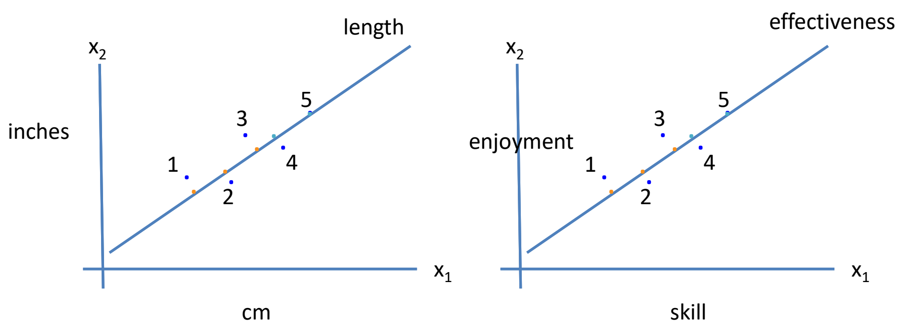
- d=2, k=1
- Which axis do we project to?
  - $z=x_1$?
  - $z=x_2$?
  - New combined axis





57

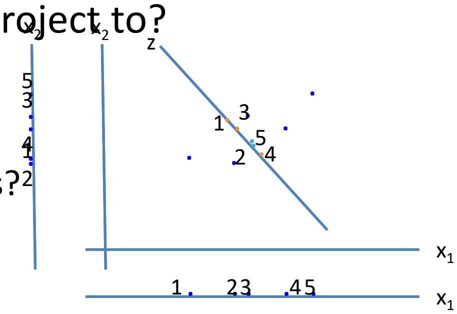
Dimensionality Reduction (Aside: Specific Example)



58

Dimensionality Reduction: Geometric Intuition

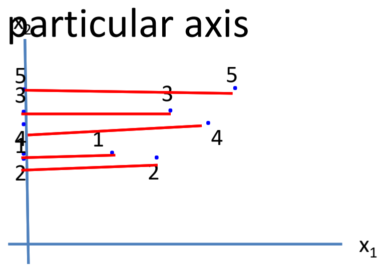
- d=2, k=1
- Which axis do we project to?
  - $z=x_1$?
  - $z=x_2$?
  - New combined axis?

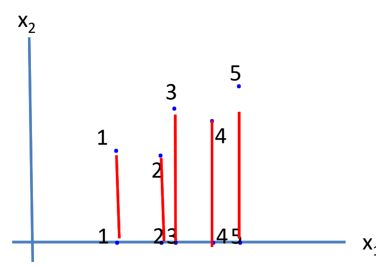


59

Dimensionality Reduction: Residual Error

- Can measure the residual error of projecting to a particular axis

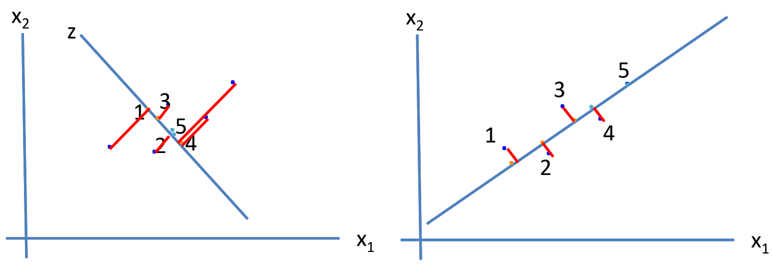




60

Dimensionality Reduction: Residual Error

- Can measure the residual error of projecting to a particular axis



61

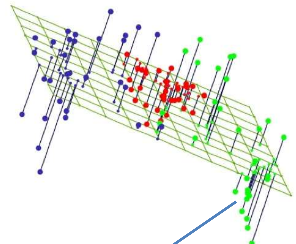
Overview

- Feature Design
- Why Reduce Dimensions?
- Feature Selection and Methods
  - Filtering
  - Wrapper
  - Built-in
- Dimensionality Reduction
  - PCA
- Examples

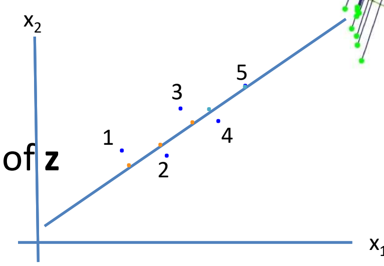
62

Dimensionality Reduction: Principal Components Analysis (PCA)

- PCA Objective
  - Project to the axis with minimum residual error
- Encoder: $\mathbf{z}=U^T\mathbf{x}$
- Decoder: $\underline{\mathbf{x}}=U\mathbf{z}$
- Find matrix U for minimum error



$$E(U)=\sum_i(\underline{x}_i-x_i)^2$$
$$=\sum_i(UU^Tx_i-x_i)^2$$

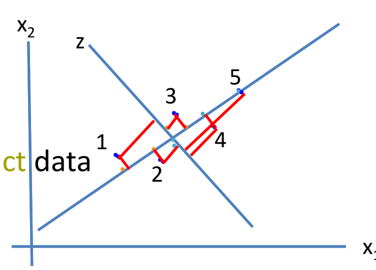


- Same as maximize variance of **z**

63

Dimensionality Reduction: Principal Components Analysis (PCA)

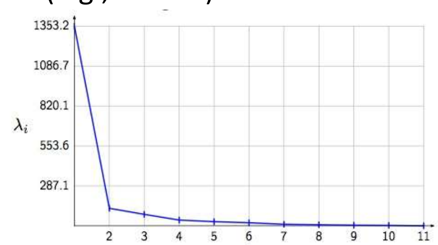
- How to solve PCA?
  - Turns out the right basis is given by the eigenvectors of the covariance matrix. So: $[U,V]=\mathrm{eig}(XX^T)$
  - Faster version without explicit covariance: [U,S,V]=svd(X)
    - Rows of U are the basis
    - Diagonal of S are the eigenvalues
    - Pick the first k rows
- Encode $\mathbf{z}=U(1{:}k,:)\mathbf{x}$
- Decode $\mathbf{x}=U(1{:}k,:)^T\mathbf{z}$
- Note:
  - Using all the Evs will store the exact data



64

PCA: How to choose the number of dimensions

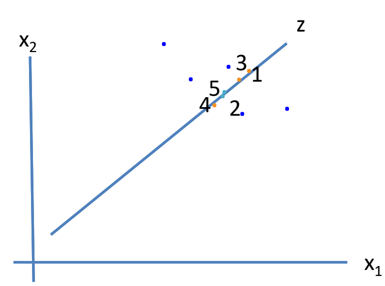
- How to choose?
- Each eigenvalue tells you what fraction of the variance/error is accounted for.
  - Strategy 1: Pick k dimensions.
- If you plot the eigenvalues, you typically get
  - Strategy 2: Take a number of eigenvalues such that you account for P% of the variance (E.g., P=99%)



65

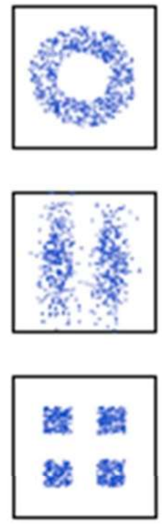
PCA: Pitfalls

- You must subtract the mean of the data first
  - Why?

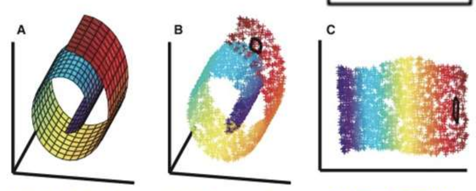


66

PCA: Pitfalls



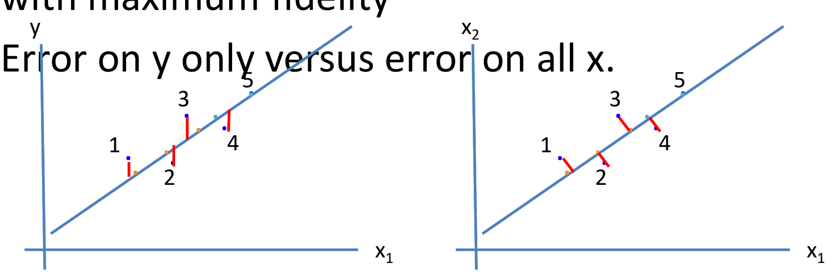
- You must subtract the mean of the data first
- NOT scale invariant, need to rescale first.
- 'Traditional' implementation computes $Cov(X)=XX^T$ which is already $O(nd^2)$
  - SVD can compute top k singulars in O(ndk)
- Second order statistics / Gaussianity assumption. This can hide many interesting patterns
- Non-linear manifolds are not covered
- Not discriminative
  - (The information your problem needs could be in a low-variance dimension)



67

PCA: Versus Linear Regression

- Regression: Predict a special output variable (y) given others (x)
- PCA: No special variable, model all the data (x) with maximum fidelity
- Error on y only versus error on all x.



68

Overview

- Feature Design
- Why Reduce Dimensions?
- Feature Selection and Methods
  - Filtering
  - Wrapper
  - Built-in
- Dimensionality Reduction
  - PCA
- Examples

69

PCA Example: Economics

- Many Economic Statistics….

Country	GDP (trillions of US$)	Per capita GDP (thousands of intl. $)	Human Develop-ment Index	Life expectancy	Poverty Index (Gini as percentage)	Mean household income (thousands of US$)	…
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	…
…	…	…	…	…	…	…	

70

PCA Example: Economics

- Many Economic Statistics
  - What are the underlying factors?
  - Reduce to 2 dimensions and plot…

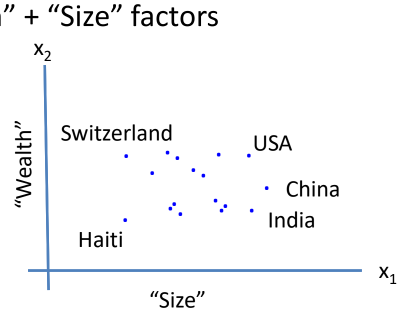
Country	GDP (trillions of US$)	Per capita GDP (thousands of intl. $)	Human Develop-ment Index	Life expectancy	Poverty Index (Gini as percentage)	Mean household income (thousands of US$)	…
Canada	1.577	39.17	0.908	80.7	32.6	67.293	…
China	5.878	7.54	0.687	73	46.9	10.22	…
India	1.632	3.41	0.547	64.7	36.8	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	…
…	…	…	…	…	…	…	

71

PCA Example: Economics

- Many Economic Statistics
  - What are the underlying factors?
  - Reduce to 2 dimensions and plot…
    - Reveals aggregate “Wealth” + “Size” factors

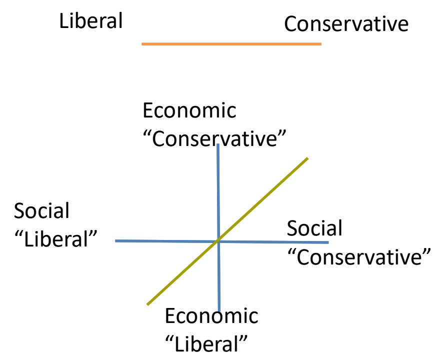
Country	GDP (trillions of US$)	Per capita GDP (thousands of intl. $)	Human Develop-ment Index	Life expectancy	Poverty Index (Gini as percentage)	Mean household income (thousands of US$)	…
Canada	1.577	39.17	0.908	80.7	32.6	67.293	…
China	5.878	7.54	0.687	73	46.9	10.22	…
India	1.632	3.41	0.547	64.7	36.8	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	…
…	…	…	…	…	…	…	



72

PCA Example: Politics

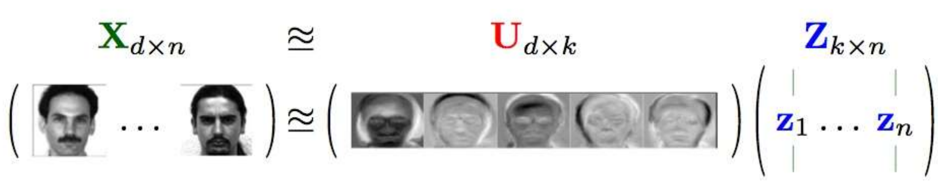
- Opinion database
  - Rows: People
  - Columns: Opinions (Immigration, Crime, Tax, Welfare, Drugs, etc)
- PCA -> 1D
  - Rows: People
  - Column: Left<->Right:
    - "Lib Dem <-> Conservative"
- PCA -> 2D
  - Rows: People
  - Columns:
    - Economic & Social views



73

PCA Examples: Eigen-faces

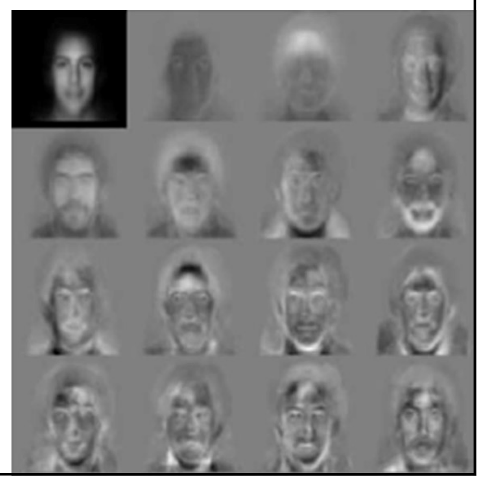
- Each face image is a database row
  - E.g. 100x100=10000 columns.
  - What if we require to store each face image in only



74

PCA Examples: Eigen-faces

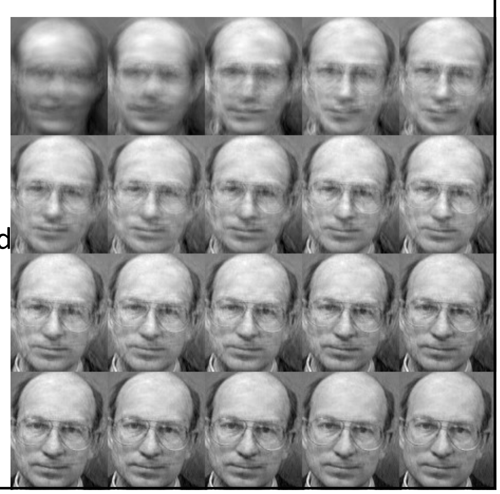
- Each face image is a database row
- First dimensions correspond to lighting, thereafter face, hair structure
- Extensively used for face recogntion
  - Speed, memory, accuracy
  - Tuck away lighting…



75

PCA Examples: Eigen-faces

- Example of reconstruction with increasing number of PCs
- Connection to general Image Compression
  - Linear vs Non-linear (eg DCT)
  - Not perceptually motivated
  - So works, but not great
- But good for revealing structure



76

PCA Examples: Other Vision.. (EECS Work ☺ - IEEE Trans KDE'11)

- Handwriting Recognition
- Gait-based person recognition
  - Recognize identity without subject cooperation (CF: fingerprint / iris)

1. Feature Engineer Background Subtraction
2. PCA to reduce dimensions

77

PCA Examples: De-anonymization (EECS Work – Student Projects ☺)

- Recognizing People by Social Network Preferences
  - (E.g. Facebook Likes)
- Vast data matrix:
  - Rows: Persons (millions)
  - Columns:
    - Likes (binary) (billions)
- Goal:
  - Predict identity from public likes.
- Too many dimensions for most algorithms. PCA made it work.

78

PCA Examples: Text Document Classification (EECS Projects ☺)

- Category of text document?
  - Who is the author of a text document?
    - Computational Forensics
  - Who is the director/script-writer of a movie?
- Vast data matrix:
  - Rows: Documents
  - Columns:
    - English dictionary (100k)
- Too many dimensions for most algorithms. PCA made it work
  - (But there are much better ways to do this, more later…)

79

PCA Example: Personality

- Private traits and attributes are predictable from digital records of human behavior, Kosinski et al, PNAS 2013
- Dataset 1: (PCA!)
  - Rows: Facebook Users, Columns: Likes
- Dataset 2: (Linear regression map PCA likes => personality and demographics)
  - Rows: facebook users, Columns: Profile details & personality test.
- Outcome: **Public likes give away**:
  - Relationship status, smoking, drugs, ethnicity, voting, religion, sexual orientation, parental divorce status…
  - Intelligence, Age, Emotional Stability, Extraversion etc…

80

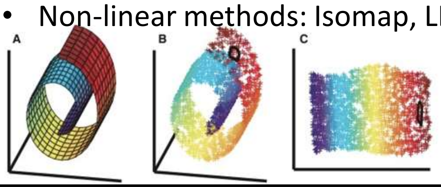
PCA Recap

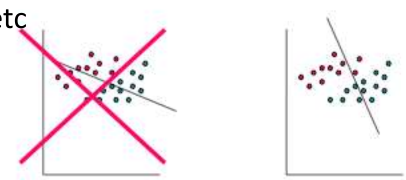
- Plus
  - Standard tool.
  - Available in most languages / toolkits.
  - Fairly Fast and robust.
- Minus
  - Assumes linear
  - Assumes orthogonal dimensions
  - Assumes Gaussian
  - Not "discriminatively trained"

81

Beyond PCA

- Non-negative matrix factorization
  - Bases have to be positive
- Linear Discriminant Analysis & Partial Least Squares
  - Find a lower-dimension projection that help separate classes
- Factor Analysis
  - Don't assume the dimensions are orthogonal
- Independent Component Analysis
  - Look for the most independent basis (e.g., blind-source separation)
- Non-linear methods: Isomap, LLE, etc





82

Summary

- Feature Engineering
  - Design a small set of informative features
- Feature Selection
  - Prune irrelevant features to increase speed, reduce overfitting
  - …and improve domain knowledge
  - E.g., Filter, Wrapper, Embedded methods
- Linear Dimensionality Reduction
  - Compress all features into a smaller number
  - E.g., PCA

83

You Should Know

- Feature Engineering
  - Some reasonable features to try in different domains
  - Especially text
- Reasons why it's useful to reduce data dimensionality
- Explain relative merits of feature selection methods
  - Filter
  - Wrapper
  - Embedded
- How feature selection applies in real-life data mining problems

84

You Should Know

- Difference between feature selection and linear dimensionality reduction.
- Linear Dimensionality Reduction
  - Explain linear dimensionality reduction as a matrix multiply
  - Encoding and decoding (compression / decompression)
- Practical
  - How to choose number of dimensions
  - Practical pitfalls to avoid
  - Some examples about how PCA can be used in real life data mining

85