COMPSCI762: Foundations of Machine Learning
Data Preprocessing

Katerina Taskova and Jörg Simon Wicker

The University of Auckland

THE UNIVERSITY OF AUCKLAND
NEW ZEALAND

SCIENCE
SCHOOL OF COMPUTER SCIENCE

Data Preprocessing

This lecture will cover

Data Preprocessing
　　Data Reduction
　　　　Dimensionality Reduction
　　　　Principal Components Analysis
　　　　Feature Selection

Data Reduction

Data Reduction

- Data reduction: Obtain a reduced representation of the data set that is much smaller in volume but yet produces the same (or almost the same) analytical results
- Why data reduction? – A database may store terabytes of data, complex data analysis may take a very long time to run on the complete data set
- Data reduction strategies
  - Dimensionality reduction
    - Wavelet transforms
    - Principal Components Analysis (PCA)
    - Feature selection
  - Numerosity reduction
    - Regression and Log-Linear Models
    - Histograms, clustering, sampling
  - Data compression

Dimensionality Reduction

- Curse of dimensionality

| x_1	x_2	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	x_n
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7	11	12	22	24
0	0	0	0	0	0	0	0	0	0	0	74	38	99	2	4	0	0	0	0	0
0	0	0	0	0	0	0	0	84	69	55	0	0	0	0	0	0	0	0	0	0
0	0	0	0	66	35	14	62	0	0	0	0	0	0	0	0	0	0	0	0	0
32	48	54	21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

- When dimensionality increases, data becomes increasingly sparse
- Density and distance between points, which is critical to clustering, outlier analysis, classification, regression becomes less meaningful

Dimensionality Reduction

- Why dimensionality reduction?
    - Avoid the curse of dimensionality
    - Help eliminate irrelevant/redundant features and reduce noise
    - Reduce computational resources (memory and time)
    - Allow easier visualization
- Dimensionality reduction techniques
    - Unsupervised linear method: Principal Component Analysis
    - Supervised method: Feature selection

Principal Component Analysis – PCA

- Find a projection that captures the largest amount of variation in data
- The original data are projected onto a much smaller space
- How? – We find the eigenvectors and eigenvalues of the covariance matrix of the input attributes,
  - Eigenvectors: the directions that data variances occur, and they define the new attribute space
  - Eigenvalues: the amount of variance along the corresponding eigenvector

Demo: `https://setosa.io/ev/principal-component-analysis/`
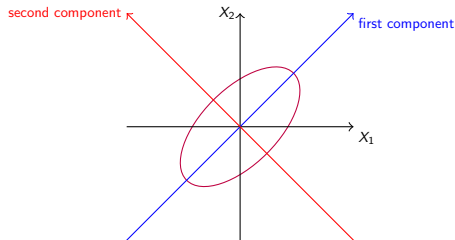
PCA – steps

- Given N data vectors from n-dimensions, find $k \leq n$ principal components that can be best used to represent the data
    - Normalize input data: Each attribute falls within the same range
    - Compute k orthonormal (unit) vectors, i.e. principal components
    - The input data is a linear combination of the k principal component vectors
    - The principal components are sorted in order of decreasing "significance" or strength
    - Since the components are sorted, the size of the data can be reduced by eliminating the $n - k$ weak components, i.e. those with low variance.
- The resulting vectors are orthogonal, are they correlated?
- Can you use PCA on categorical data?

```
https://scikit-learn.org/stable/modules/generated/sklearn.
decomposition.PCA.html
```

Feature or Attribute Selection

Reduce dimensionality by removing set of attributes

- **Redundant attributes**
    - Duplicate much or all of the information contained in one or more other attributes
      e.g. purchase price of a product and the amount of sales tax paid
- **Irrelevant attributes**
    - Contain no information that is useful for the data mining task at hand
      e.g. students' ID is often irrelevant to the task of predicting students' GPA

Two types of methods: Filters (fast) and Wrappers (high accuracy, expensive)

- Filters separate feature selection from classifier learning. No bias toward any learning algorithm

Feature Selection using Correlation

- For nominal data, given two attributes A and B with values $a_1, \ldots, a_c$ and $b_1, \ldots, b_r$ the correlation can be calculated using the χ^2 test:

$$\chi^2 = \sum_{i=1}^{c} \sum_{j=1}^{r} \frac{(o_{ij} - e_{ij})^2}{e_{ij}}$$

  - With o_{ij} being the actual frequency of the event (a_i, b_j)
  - And e_{ij} the expected frequency (n is the number of instances)

$$e_{ij} = \frac{count(A = a_i) \times count(B = b_j)}{n}$$

Feature Selection using Correlation

- Numerical data can be compared using Pearson's correlation coefficient

$$r_{A,B} = \frac{\sum_{i=1}^{n}(a_i - \bar{A})(b_i - \bar{B})}{n\sigma_A\sigma_B} = \frac{\sum_{i=1}^{n}(a_ib_i) - n\bar{A}\bar{B}}{n\sigma_A\sigma_B}$$

- With means $\bar{A}$ and $\bar{B}$, number of instances n, and standard deviations σ_A and σ_B
- So what does the correlation measure?
- How can it be used to remove redundant or unimportant features?

Heuristic Search in Attribute Selection

- There are 2^{d-1} possible attribute combinations of d attributes
  $\rightarrow$ exhaustive search is not feasible (e.g. $d = 300$, $2.04 x 10^{90}$ combinations)
- Typical heuristic attribute selection methods:
    - Best single attribute under the attribute independence assumption
    - Best step-wise feature selection:
        - The best single-attribute is picked first
        - Then next best attribute condition to the first, ...
    - Step-wise attribute elimination:
        - Repeatedly eliminate the worst attribute
    - Best combined attribute selection and elimination
    - Optimal branch and bound:
        - Use attribute elimination and backtracking

Relief - Instance-based heuristic for feature selection

Input: Data set with N_a attributes and N_i instances that bellong to one of two classes, and number of randomly selected insatnces $N_r < N_i$

First normalize the data

Create a weight vector W with one weight $w_i \in W$ for each attribute

Initialize the weights $W = 0$

for $j \in 1 \ldots N_r$ **do**

     Randomly select instance $R = [r_1, \ldots, r_n]$

     Choose instance $H = [h_1, \ldots, h_n]$ as the closest neighbour of R in the same class (*nearHit*)

     Choose instance $M = [m_1, \ldots, m_n]$ as the closest neighbour of R in the other class (*nearMiss*)

     for $i \in 1 \ldots N_a$ **do**

         $w_i = w_i + (r_i - m_i)^2 - (r_i - h_i)^2$

     end

end

for $i \in 1 \ldots N_a$ **do**

     $w_i = \dfrac{w_i}{N_r}$

end

Relief

- Relief takes into account **all** attributes
- Result is a weight vector that represents the importance of each feature
- Features are then selected based on a threshold τ or ranked
- The algorithm above is the basic version of Relief, there are various extensions (ReliefF, RReliefF,...)

Wrappers

- The correlation method and Relief are filters
- Main idea of wrappers:
  - Generate a subset of the features and evaluate the performance of the classifier on the subset
  - Add or remove attributes from the subset and see if the performance of the classifier improves
  - Risk of overfitting, especially if choosing the same classifier as for the main learning task

Subset selection
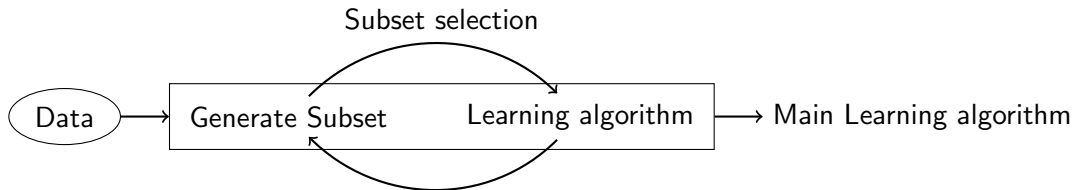
Preprocessing

- So, to summarize...
    - When are preprocessing approaches useful?
    - When should you avoid them?
    - How about specific cases
        - Many correlated features?
        - Many independent features?
        - Which algorithms you know already would need preprocessing?
        - How about Decision trees? Why?
        - How about Regression? Why?
    - Are we cheating in preprocessing: for example by creating new examples?

Conclusion

- Preprocessing is an important part in machine learning and data analysis
- Missing values can be caused by various reasons depending on what the reasons are, they must be addressed differently
- Various imputation approaches exist, they use the information of other instances and values to impute the missing values
- Noisy data can be addressed for example by binning, clustering, or regression
- Feature selection can be used to reduce the number of redundant and unimportant features
- Imbalanced data sets can be a problem for evaluation and classifiers
- Sampling can be used to overcome class imbalance problems

Literature & other resources

- Material in Chapter 3 in Han's *Data Mining*
- Detailed math and application of PCA: Chapter 12.1 and Appendix C Bishop's *Pattern Recognition and Machine Learning*
- Step by step code on PCA calculation (Python notebook on Canvas)
- Example code on using sckit-learn PCA implementation
  `https://colab.research.google.com/github/cpearce/`
  `PythonDataScienceHandbook/blob/master/notebooks/05.`
  `09-Principal-Component-Analysis.ipynb`

Thank you for your attention!

https://ml.acukland.ac.nz