Data Analysis and Application of Machine Learning

Dr. Heiko Niedermayer

Cornelius Diekmann, M.Sc.
Prof. Dr.-Ing. Georg Carle

Lehrstuhl für Netzarchitekturen und Netzdienste
Institut für Informatik
Technische Universität München

Version: July 7, 2014

Agenda

Slides: 35

Introduction

Introduction

- Networks change and their usage changes. As a consequence network algorithms and methods have to adapt, analysis as well.
- Consider TCP, TCP adapts to the currently possible data rate. The rate is limited due to overall bandwidth constraints, but mostly also due to other traffic crossing the same links and routers. This can be seen as a learning problem where the predictor learns the current rate online and adapts its value.
- The TCP congestion control is, of course, a control algorithm and not a machine learning method. However, there are also machine learning proposals for data rate adaptation.

Online vs Offline

- Note, the TCP example from before refers to *online learning* or data processing where the predictor or analyzer learns and adapts while it is used.
- The opposite to that are *offline* methods. Here, data is gathered before training or analysis. The data is then given to the algorithm.
- After the learning phase, learning usually stops and the learned functionality is then only used.
- Note, learning is a form of optimization (find parameters with minimum error). In online learning optimization happens over individual data items while in offline learning the idea is to optimize over all given data.

Supervised vs Unsupervised

- Again, in case of the TCP example, the algorithm itself derives its data rate recommendation from its data input (situation), previous recommendations and subsequent consequences (experience). Methods like these where the learner does all by itself are called *unsupervised learning*.
- Now consider the following: Engineers have recorded a number of situations and selected the best data rates they encountered via experiment as the data rate of choice in each case. This data set is now given to a machine learning algorithm to learn the correct data rate decisions from their decision examples. Here, we have the engineers as teachers who present correct input-output pairs to the algorithm and they subsequently evaluate if the algorithm learned the patterns they had given the algorithm and how it performs on similar patters they did not give the learner. This is an example for *supervised learning*.

Preparing Data

What to do with data?

- Data is produced by simulation, measurement, or other similar forms of data generation.
- The form in which the data then exists...
  - ... can be used directly for analysis or machine learning.
  - ... can be in a form that makes it hard to find properties of interest, however.
    - *Consider voice processing, the signal values from the samples themselves are not very helpful.*
  - ....can be transformed into in a semantically more useful form.
    - *Consider again the voice processing, transforming the data into frequency spectrum provides more useful information for voice analysis algorithms and machine learning.*

What to do with data?

- So, it is recommendable to consider transforming the data into a more useful form before applying other methods (Considering does not mean doing it always).
- Split the problem into smaller ones: Let specialized algorithms derive data properties, which are then used for further analysis.
  - As example: Instead of taking any possible graph as input, you take a fixed set of graph properties as input for your algorithm.

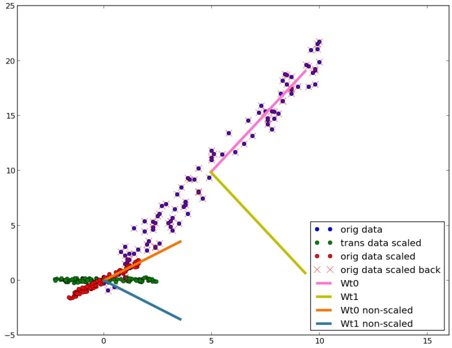
Methods

- Normalize data
- Transform (multi-dimensional) data so that some items (dimensions) become more significant (effect) and others less (noise)
  - Principle Component Analysis (basically an eigenvector analysis of the covariance matrix used to change to a ”better” basis)
- Remove unimportant or less important data items (Dimensionality Reduction)
- Noise Reduction, Outlier Removal
- Time Domain $\mapsto$ Frequency Domain:
  - Fourier Transform (FFT, DFT), e.g. in windowed form to cover changing of frequencies over time
  - Wavelet Transforms, Z-Transform, etc.
- ...

Principle Component Analysis (PCA)

- Principle Component Analysis (PCA, Hauptkomponentenanalyse) is used to:
  - Reduce number of dimensions (variables) while keeping most of the variance of the data and (hopefully) mainly losing noise.
  - Project data onto lower-dimensional space
  - Find important factors / effects in the data (assumes some linearity)
- Algorithm:
  - PCA considers data as m n-dimensional data points which form an m-x-n matrix.
  - Step0: Remove the mean from the data so that is centers around 0.
  - Step1: Compute n-x-n covariance matrix of the data.
  - Step2: Calculate eigenvectors of covariance matrix.
  - Step3: Select k (with $k \leq n$) eigenvectors with the largest eigenvalues to form the new basis.
  - Step4: Transform data with n-x-k matrix. Result is a m-x-k matrix in the new basis.

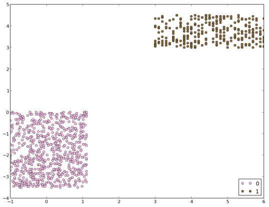
PCA in Python



- In Python: res= matplotlib.mlab.PCA(data).
- The Figure shows the original data (blue), the scaled version is the same data centered and divided by standard deviation in each axis (red). The green plot is the transformed data according to PCA. Wt0 is the eigenvector with highest eigenvalue.

Clustering

Clustering



- Clustering is an example of unsupervised learning.
- Input data is a multidimensional data set where the goal of the clustering is to find areas of data points that belong together due to some kind of proximity.
- The picture above shows a scatter plot of a data set. Visually, we may identify two clusters of points. The goal is now to find them algorithmically.
- Note that the choice of clustering algorithm depends on your application domain and what makes two points similar.

k-means

k-means Clustering

- k-means is probably the most-known clustering algorithm. (Python: in scipy.cluster.vq).
- Assumptions:
  - The data set contains *k* clusters.
  - The clusters are convex or can be approximated by a convex splitting of space.
- Idea:
  - Lets have *k* points, each defining a cluster as its mean.
  - For each point, the closest mean defines the cluster membership.

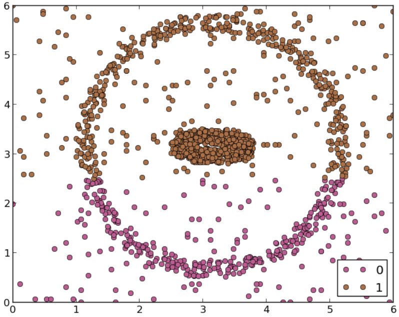
k-means Clustering

- Algorithm:
  - Input: *data*, *k*
  - Output: a list of *k* mean points defining the *k* clusters
  - Position the *k* points via some strategy over the data space.
    - Random point, random data point, given point, ...
  - Determine the cluster membership for all points.
  - Reposition cluster mean by computing the mean over all points of the cluster.
  - Repeat this process for some round or until convergence is achieved.

k-means Issues

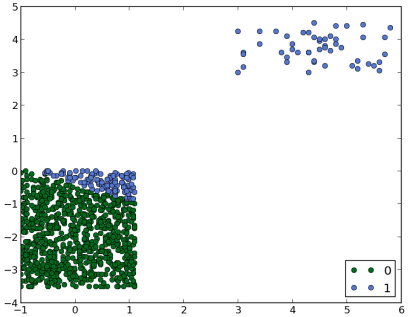
- How to know and chose k?
  - Depends on the problem and purpose of the clustering.
    - If the actual clusters have to be found, domain knowledge of the problem has to be used to carefully determine the value.
    - Selection is less critical if clustering is only applied to organize data (e.g. all As may be in cluster A, but also all Bs and Cs).
  - Some k-means variants try to find the right k, usually by trying different k.
- Difficult situations (Examples on next slides):
  - k-means splits the vector space into convex subspaces. However, clusters may have other geometric forms, e.g. circle.
  - The computing of the cluster mean favors clusters with a large number of points. Smaller clusters may not be able to move the mean of a cluster to its center as neighboring clusters with more points attract the mean more.

k-menas - 2 circles with noise



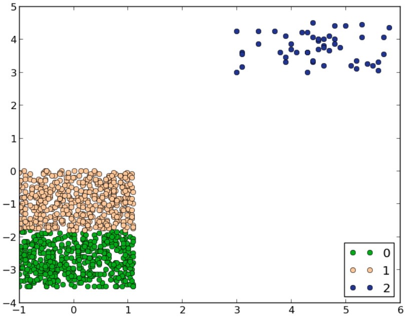
- Here, k-means fails to spot the circles as such clusters are not within its assumptions. Note that points on circles may be very far away from some other cluster members which may also not fit to the idea of nodes in a cluster being similar.

k-means - asymmetric number of points per cluster



- Here, k-means splits the lower-left cluster with a lot of points into two clusters, one of them including the other cluster. The smaller cluster has only small chance of attracting the mean (unless in case of good initial conditions).

k-means - asymmetric number of points per cluster



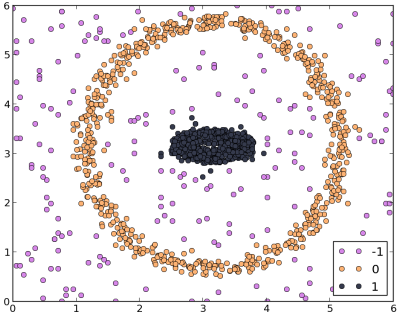
- Here, we set k=3. With two clusters and their means fighting for the main cluster, the top-right cluster can finally attract a cluster.

DBSCAN

DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

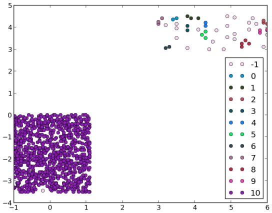
- Idea:
  - Clusters form a dense area of points and area between clusters is less dense.
  - Thus, use distance *eps* between points to decide if they are in the same cluster.
  - Points in less dense areas are considered noise and not put into a cluster.
- Algorithm:
  - Input: data, eps, minpts
  - Output: a list with the cluster membership of each point
  - Take a node that is not yet part of a cluster or considered noise.
  - Determine all points closer than *eps* distance. If the number of poins in this area is smaller than the constant *minpts*, consider the node noise.
  - Else, expand the cluster of the node as long as the new nodes also have *minpts* points in their *eps* distance area.

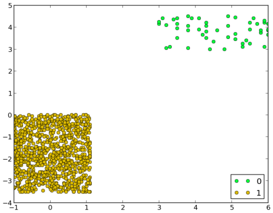
DBSCAN - 2 circles with noise



- Here, DBSCAN classified the two circles into two clusters and assigned other points in-between as noise.

DBSCAN - asymmetric number of points per cluster





- Here, the density of the clusters is different. Considering a valid eps distance for the lower-left cluster will lead to many found clusters for the top-right cluster. However, as the distance between the two clusters is large, a large eps distance will manage to find both.

Clustering - Conclusions

- We have seen both algorithms find the right clusters and also we have seen examples for both where they fail.
- The algorithm of choice should fit to your metric.
- What could this mean? If you look for similarities in the sense of all members being close in ID space (e.g. nodes should be close to certain AS routers in order to predict similar hop distance to target locations), then algorithms that anticipate that seem preferable. If you look for the diversity of data items with some similarity (e.g. all German nodes on the Internet to study their overall connectivity), this membership may not be defined by maximum distance to all nodes, but maybe only require closeness to some other member nodes.

Learning and Testing

Did we find the right clusters? Did we learn?

- In the previous section, we used all data points to generate clusters. How can we tell if the result is good?
- There are multiple answers to the question raised. We cannot deal with the application domain-specific ones.
- Methodologically:
  - Learning is tested best on yet unseen but similar data. It is usually about finding the essence of something.
    - Training Set: Set of data to train the algorithm.
    - Test Set: Set of data to see if what was learned also applies to other data.
  - Cross-Validation and similar aspects
    - Lets apply the clustering algorithms on subsets of the data. Will the results be similar or does it significantly change?
    - The main idea is again to generate multiple data sets. Each data set can be given to a learning algorithm. Their output is then compared or tested.
    - Furthermore, one may then continue the learning on another set.

How to generate the data sets?

- Splitting
  - Split the data into training sets and test sets. Traditional way in machine learning.
- Resampling
  - Each set is generated from the complete data set, e.g. with probability p in the set or not.
  - Bootstrap concept: statistical resampling concept with good properties, items are drawn with uniform random distribution always from the complete data set, same item can be drawn multiple times (for one data set).
  - Resampling allows to generate as many data sets as needed even if there are fewer actual data items.
- Note: The machine learning as well as these concepts assume that the data items are independent of each other. If subsequent data items depend on each other, consider concepts like time-series analysis or merge these items to larger data items.

Overfitting

- The performance of an algorithm on trained data only has the limit of 100 %. Memorizing all data items and their output is a way to achieve this. While this may show how capable an algorithm is in adapting to complex and strange data, it does tell anything useful about its real performance on real data.
- Overfitting refers to the fear that the algorithm learns the noise of the particular data set, but not the general aspects of the function that should be learned. This can worsen the performance on test set data.
- An indicator for running into overfitting is when only the performance on the training set increases while learning, but the performance on the test set remains at a certain level.

Supervised Learning

Supervised Learning

- As mentioned in the introduction, in supervised learning there is a teacher component that for each data item also provided the correct output. The goal is now again to learn the function from the examples, and generalize to cases that were not in the training set.
- Examples for supervised learning approaches:
  - Neural Networks, Support Vector Machines, Decision Trees, . . .
- The meta algorithm for supervised learning is that an optimization function adapts certain parameters of the algorithm(e.g. weights of inputs to a neuron) so that performance on the training set is maximized. The result is then tested on the test set to check actual performance.

Decision Tree

- Usually used for classification.
- Data items are evaluated from the root of the tree to a leaf.
- Each internal node of the tree contains a rule which child to select given a data item.
  - Male? Yes (child 1), No (child 2)
  - Height larger than 1,70m? Yes (child 1), No (child 2)
  - Weight? 0 - 60 kg (child 1), 60 - 80 kg (child 2), $>$ 80 kg (child 3)
- Each leaf node is linked with a class. Items ending at it are assigned the same class. The class is the class of the majority of items ending at the leaf node in the training set.
- Learning: start with the root node. This initial graph assigns the majority class of the items in the training set to all data items. Then add more nodes, preferable with rules that make items from one class follow similar paths and items from other classes follow other paths.

Conclusions

- Machine Learning provides interesting tools for data analysis and for making algorithms and systems adaptive.
- Select the method suitable for your problem. Do not simply take the one method everyone knows.
- Do not forget to prepare the data to provide analysis and learning with meaningful and standardized inputs.

Thanks for your attention!

Questions?