1 MAA507, seminar and project topics.

Below is a couple of possible topics you can choose for your seminar presentation and project report, feel free to choose another topic related to the course. However you should then ask first so that we know and can approve of the topic as well as see for which seminar occasion it would be good for you to present.

The the project and seminar presentations are to be done in groups of 1-2 people. They consists of first a $20 - 30$ min. presentation and then the acting opponents will have an opportunity to ask questions and give comments. Note that the time available for the presentations might change depending on the number of groups.

It is important to note that the project report should be submitted at the end of the course, regardless of when you did your seminar presentation. Depending on when you present, it might be worth it to first focus on the presentation, and then finish the report after.

You are required to choose group and topic at the latest 28 February, if you have problem finding a group (and want to work in a pair) make sure you contact any of the lecturers as soon as possible so that we can possibly find someone for you to work with.

1.1 Theme 1, graphs and networks

1.1.1 k-means and other related clustering algorithms

K-means and other clustering algorithms such as EM-algorithm and more advanced methods such as fuzzy C-means are heavily used in clustering of date.

- Look at one method such as k-means, describe the method as well as some additional modifications on the algorithm or more advanced versions of the method. It might also be interesting to implement the method and try it on some data (such as an image, or some other dataset you got).
- Look at and describe a couple common methods such as k-means, EM-algorithm and methods based on the Gaussian distribution or areas of high density. Try to compare the methods, what are their strengths and weaknesses?

Some of these are quite easy (such as K-means), in that case we expect you to not only describe how it works, but also something else such as looking at why it works, how efficient it is or look at at least one of the more advanced versions of your method.

There is a good page on K-means on Wikipedia where you can also find some links to pages on some related algorithms. There you can also find many references to other related articles. One article comparing some alternative related methods is Alternatives to the kmeans algorithm that find better clusterings by Greg Hamerly and CHarles Elkan.

1.1.2 Community detection in networks.

There have been many studies of social networks in recent years given the rise of sites such as twitter and facebook which gives access to a huge amount of data to work with. One interesting question is regarding how we can find communities within such a large network? To do this graphbased clustering algorithms are usually used, some of which are described in Community detection algorithms: a comparative analysis by A. Lancichinetti and S. Fortunato.

In this project we would like you to describe the problem as well as pick one (possibly two) of the algorithms described in their paper which you choose to work with. Describe the algorithm and if possible try to implement and test it on some standard datasets. For example you could choose to work with the Newman-Girvan algorithm or the Markov Cluster Algorithm.

At M. Newmans homepage you can find some standard datasets which are often used to evaluate and compare the results of these type of methods.

1.1.3 Traveling salesman problem

Consider a salesman that wants to visit a number of cities once, we are interested in how we can find the shortest path which visits all these cities once. For the seminar presentation we would like you to describe the problem, show how to solve it analytically and why this is generally not possible even for a moderate number of cities. Then it would be good to give an explanation of one or a few methods which can be used to give approximate solutions. There is a large number of different methods you can look at such as Monte-Carlo based methods, methods based on linear programming, Markov chains, genetic algorithms or ant colony methods.

For the project report you are encouraged to look more at one of the more advanced methods to solve this and related problems.

You can find some information on the problem on Traveling salesman problem on Wikipedia.

1.1.4 A comparison of different distance or similarity measurements on graphs.

While we in say $\mathcal{R}^n$ it is often natural to use the usual euclidean distance between points. In graph's the vertices is often not easily put in such a space, instead we have measurements of some kind of relation between vertices. This could be length of for example roads between vertices but it could also be something such as number of times two phrases represented by two vertices are used together.

In this topic we would like you to try a couple of different distance measurements and look at how the distance or similarity function used gives different results when doing some common things on graphs. You could for example randomize a couple of graphs and see how different distance functions give different clusters when trying to find clusters in the graph.

You can read some on clustering and different distances at Hierarchical clustering on Wikipedia. An example of a similarity measurement and application can be found at Conceptual Graph Matching for Semantic Search by Jiwei Zhong , Haiping Zhu , Jianming Li , Yong Yu.

1.1.5 Hierarchical clustering

Hierarchical clustering is a clustering algorithm which not only gives a final clustering (like for example K-means), but instead gives a hierarchy of clusters. This gives it more flexibility in the size and amounts of clusters than many other clustering algorithms. Hierarchical clustering begins by defining some kind of distance (or similarity) between clusters and a linkage criterion used to divide or combine clusters. After this clusters are created using a greedy approach by either starting with one big cluster and dividing this untill every observation is in their own cluster or the other way around by letting every observation start in their own cluster and succesively combine clusters.

In this project we would like you to explain how the different types of hierarchical clustering works and what are the strengths and weaknesses of the different versions (as well as if there is any common for all of them). It would also be a good idea if you either implement some hierarchical clustering method and try it on some example data, or look some more at some of the underlying mathematical concepts such as greedy algorithms: what is it, why is it used here, does it give the öptimalsolution? or the time complexity of the method or different variations of the method.

You can read an introduction on hierarchical clustering on Hierarchical clustering on Wikipedia.

1.1.6 Minimum spanning tree and Image segmentation.

Minimum spanning trees forms an alternative method in image segmentation by looking at the image as a graph with the pixels forming the vertices. In this topic we would like you to describe one such method and show how it can be used.

You can find a short introduction at Minimum spanning tree-based segmentation on Wikipedia. A good article is Efficient Graph-Based Image Segmentation by Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Another article on a later method can be seen in An Efficient Parallel Algorithm for Graph-Based Image Segmentation

1.1.7 Expanders

When designing communication networks, it is of interest that any pair of nodes have a short path connecting them, and also that there are no bottlenecks—that there are multiple independent paths connecting any two sets of nodes, to allow working around traffic congestion and other problems that may arise at intermediate nodes. One of the early highlighted features of the Internet was its ability to reroute traffic around problem areas, but that still requires that the underlying hardware network provides plenty of alternative routes, and designing a network so that it does is not as easy as one might think.
The trivial mathematical solution would be to connect every node directly to every other node, but that is typically much too expensive and technically infeasible; rather, one should expect that there is some fixed number k which is the maximal number of neighbours allowed for any single node. Given such a bound, it is not too hard to see that the worst-case path length must grow at least logarithmically with the total number of nodes, which is not too bad and also achievable. The tricky thing is that it is difficult to come up with a deterministic construction of a network topology that achieves this for arbitrarily large networks; examples are known, but they are not easy. On the other hand, it turns out that random graphs provide plenty of examples!
What this project would involve is:

1. Presenting the concepts of (n, k, c)-expander, (n, k, c)-magnifier, and possibly also n-superconcentrator; explain how one can obtain one from the other. Analyse the various conditions involved and explain their technical implications, if there are any.

2. Prove the theorem that one can use the two largest eigenvalues of a graph to verify that it is a magnifier. Generate a couple of random graphs and use this to check how good magnifiers they are.

3. A possible extension would be to use probabilistic methods to prove that linear size expanders exist, but this is requires more extensive background in graph theory and probability theory, so it is not expected to be done as part of this project.

One source about this is West: *Introduction to Graph Theory*. Selected excerpts will be posted on blackboard.

1.2 Theme 2, sparse matrices and eigenvalue computations

1.2.1 Solving linear systems

Often when trying to solve a linear system you might end up with an extremely large but very sparse system. In this case trying to find the inverse to solve the problem might not be feasible, instead we need specialized methods. You can either:

- Look at some special types of matrices for which there are specialized methods to solve the problem, for example band matrices. Describe these types of problems and the method used to solve them.
- Look at a more general method for solving sparse linear systems such as conjugate gradient or GMRES.

An introduction to Levinson recursion used to solve linear systems of toeplitz matrices can be found at Levinson recursion on Wikipedia. There you can also find an introduction to iterative methods based on the Krylov subspace such as Conjugate gradient method. Another method to look at could be how to use the Cholesky decomposition for positive definite matrices.

1.2.2 Iterative methods based on the Krylov subspace for calculating eigenvalues

The Krylov subspace is the space generated by repeatedly multiply the initial matrix A with a a vector b.

$$K_r(A, b) = span\{b, Ab, A^2b, \ldots, A^{r-1}b\}$$

This subspace is used by many modern iterative methods in order to find eigenvalues of large sparse matrices. For the seminar presentation we recommend that you show why we need specialized methods for large (sparse) matrices, a short description of the Krylov subspace and briefly outline how the Krylov subspace can be used in order to find eigenvalues of A.

A very short page on the Krylov subspace with some links to relevant information can be found at Krylof subspace on Wikipedia. You can for example choose to look more at the Lanczos iteration or Arnoldi iteration and how they can be used to find eigenvalues.

1.2.3 PageRank and search engines

PageRank is a ranking algorithm used to rank homepages in order of importance depending on the link structure of the Internet. This or some variation of is often used by search engines to decide which of many found pages should be showed first. Since there is a huge amount of homepages on the Internet the ranking algorithm needs to be extremely fast, but it also needs to be very accurate since most people don't look at more than a couple of results of a search query, regardless of if there was a large number of hits or not.
You can choose either to work with the PageRank algorithm specifically, explain the mathematics behind it and how it can be calculated effectively and what it is used for. For the project report we would recommend that you write a little more about something related to the algorithm. You could see how PageRank changes as you do some small changes to a graph structure or you could take a look at how the constant c in the algorithm changes the result, could another c be used somewhere else? Alternatively you could look at some ways to calculate PageRank even faster and describe these.

Alternatively you could choose to look more at the overall structure of search engines, describe the essential steps for a search query as well as the overall components needed for a search engine.

You can read more on the mathematics behind PageRank at The $25.000.000.000 Eigenvector, The linear algebra behind Google by K. Bryan and T. Leise, there is also a good page on pagerank and the power method on wikipedia. Another good article if you want to read more is Inside PageRank by M. Bianchini, M. Gori, and F. Scarselli and The Mathematics of Internet Search Engines by F. Andersson and S. Silvestrov.

1.2.4 EigenTrust and peer-to-peer networks

Peer-to-peer file sharing networks have many advantages over a standard server-client approach in that it is much more scalable (the would be clients help other clients with their download, easing the load on the would be server). It is also more robust in that for example even if the original distributor failsthe file could still be available. However since anyone can upload files there is also the risk of malicious individuals to send dummy files (which doesn't work) or outright harmful files.

EigenTrust is a method similar to PageRank used to combat the spread of inauthentic files in a peer-to-peer file sharing networks. It works by giving a rank"(trust) to each individual on the network, with the aim of isolating those with a to small trust".

In this project you should explain the problem and how EigenTrust (or another similar method) is used and computed. For the project you could look at either describing the method (both the simple and more advanced versions) as good as possible. Alternatively if you want you could describe it a bit more briefly and try implementing it and try on some randomly created network.

The original article describing the Eigentrust algorithm is The EigenTrust Algorithm for Reputation Management in P2P Network.

1.2.5 DebtRank and systemic risk in financial networks

When a part of a financial network fails, it runs the risk of severly impacting other parts of the network as well (in the worst case creating a chain reaction). Especially the failure of a very large actors can cause many other actors to run into problems or fail themselves as well.

DebtRank is a way to measure the risk in such a financial network to determine the "importanceöf a single actor, not only depending on size but also their position"in the network. DebtRank works on a similar principle to PageRank but is calculated a bit differently.

In the project you should try to explain the problem and how DebtRank (or another similar method) can be used to evaluate the risk in the network. If you want it could also be interesting to try and implement the method on some network and try it yourself.

You can read of DebtRank in for example DebtRank: Too Central to Fail? Financial Networks, the FED and Systemic Risk

1.3 Theme 3, Information and Internet security

1.3.1 RSA cryptography

Being able to send information securely without anyone except the receiver being able to read it is very important. One of the most important cryptographic methods is the RSA algorithm. In the seminar presentation we would like you to describe the method, how it is used and briefly why it works. The discussion part could then be made discussing questions such as: If it's unfeasible to mathematically breakthe code, what other methods could be used to attack something using RSA-cryptography. What could happen if a way to more efficiently compute factorizations of large integer values (such as efficient accessible quantum computers).

For the project report it would be a good idea to look either more into why it's so hard to mathematically break something using RSA cryptography and/or look more into what you can do more to further increase the security using for example padding.

You can find a good introduction at RSA algorithm on Wikipedia, there are also some additional links you can look at.

1.3.2 Diffie-Hellman key exchange

The majority of all encryption of internet communication uses symmetric-key encryption schemes, such as AES, 3DES, and Twofish, where sender and receiver both use the same encryption key. This creates the problem of how the two can both obtain the key (a shared secret) when any communication channel between the two may be subject to eavesdropping. On the internet, it is frequently the case that the two parties have never before communicated with each other, so one cannot assume that they have any shared secret in common from which an encryption key could be derived.
The first published solution to this was the Diffie-Hellman key exchange procedure (wikipedia), which forms the basis of the Internet Key Exchange protocol (first codified in RFC 2409, later revised). This makes use of large primes (at least as large as the size of the shared secret; additional primes can be found in RFC 3526, which are published beforehand.
It should be noted that the Diffie-Hellman key exchange procedure resembles the RSA asymmetric-key encryption scheme, but has some fundamental differences. In RSA, one does calculations modulo a product of two large primes, whereas in Diffie-Hellman one does calculations modulo a prime. The strength of RSA is related to the difficulty of the integer factorisation problem, whereas the strength of DiffieâĂŞHellman is related to the difficulty of the discrete logarithm problem.

1.3.3 Primality testing and verification

Many encryption schemes need large (say about a thousand digits long, or more) primes as parameters. This leads to two algorithmic problems: 1. Find a prime that is that large. (This may be part of choosing one's encryption key.) 2. Verify that an integer that large really is a prime. (This may be part of checking that you're not being fed bogus data by an attacker.)
Useful literature on how to solve those problems are Sections 4.1-4.7 (particularly 4.4, 4.6, 4.7) and 4.10 of Herbert S. Wilf's Algorithms and Complexity.
One point of note is that since this book was published, the AKS primality test has established that there is a polynomial algorithm for deterministically testing primality. In one sense, this solves both problems 1 and 2 above, but it is still too slow (too high degree polynomial) for most practical purposes. Therefore the algorithms described in Wilf remain of practical importance.

1.3.4 Linear codes

When sending information over a channel errors often occur in the retrieved information. Linear codes is used to find and correct the most common errors. In this topic we would like you to describe the problem and where it occurs as well as describe one such linear code, for example Hamming codes or Hadamard codes. How it is used, what type of errors it can fix, and maybe something on the difference to other codes.
There is a page on on Wikipedia on Linear codes you can use with links to some of the most common methods.

1.3.5 Dimension reduction of large datasets

Often you might end up with data of high dimensionality and a large number of measurementsfor example temperature at a large number of measuring stations over a long period of time. A useful technique to make the date more easily manageable is to use a dimension reduction technique such as Principal component analysis (PCA), Partial least square regression (PLS-R) or discriminant analysis.

For the project report it would be a good idea to look briefly at a more advanced method based on your chosen method or look at some small variations of the method. For example how does computing PCA using singular value decomposition (SVD) differ from using the covariance method?

Two good sources on PCA in general is A Tutorial on Principal Component Analysis by Jonathon Shlens and chapter 8 in *Applied Multivariate Statistics* by Richard A. Johnson and Dean W. Wichern (available at the MDH library). Some ideas for applications and suggestions for sources on data for a specific field can be found using the Principal component analysis page on Wikipedia.

A good introduction to PLS-R can be found in Partial Least Square Regression by Hervé Abdi.

An introduction to Discriminant analysis (LDA/QDA) can be found on Wikipedia.

1.3.6 Implementing a spellchecker

Spellcheckers are used extensively nowadays in everything from word processors to search engines. One of the most common ways for a word processor to dentify a misspelled word is by using a lexicon. If some word is not found in the lexicon it is flagged as possibly misspelled after which typically one or several candidate words are proposed by calculating some distance measure to words in the lexicon and proposing those that are the "closestto the possibly misspelled word.

On of the most common (and simple) ways to measure distance between words is using the Levenshtein distance although other more advanced distance measures can also be used either by themselves or in combination with others. If the lexicon is large, calculating the distance from one new word to every word in the lexicon can be quite time consuming, one way to handle this is by first clustering the lexicon into different clusters using for example PAM (partition around medoids) and instead first calculate the distance to one element representing each cluster first and then later only calculating the distance to those words belonging to the closest clusters.

In this project we would like you to describe the problem, Levenshtein distance and how it can be used to construct a spellchecker as well as how clustering can be used to aid in the process. Preferably you should try and implement a simple spellchecker and try it on some test data as well.

You can read more on spellcheckers and how PAM can be used in the spellchecker in for example 'Effective Spell Checking Methods Using Clustering Algorithms' by R. C. de Amorim and M. Zampieri (you can ask the teachers for a copy). For an example of a simple implementation of a spellchecker as well as lexical data to get you started can be found at Peter Norvig's site.