CSCI 6612

Visual Analytics

Fernando Paulovich

Alex Safatli

Wednesday, September 4, 2019

Contents

Introduction **3**
Course Information . . . 3
What Is Visual Analytics? . . . 3
Data Pre-Processing . . . 4
Visualization . . . 7

Going Further into Visualization **10**
Multidimensional Visualization . . . 10
Graph, Network, and Tree Visualization . . . 12

Data Mining **14**
Introduction to Data Mining . . . 14
Classification . . . 15
Clustering . . . 19

Visual Analytics **20**
Introduction to Visual Analytics . . . 20

Introduction

Course Information

General Information. Lectures take place at a single time every week on Wednesdays at 11:35 AM. There are two teaching assistants who can give support for programming; recommended languages are Python and Javascript. Prototype software will be created within groups. Evaluation will be fair - if the idea is feasible it will give a good mark. This course will provide tools to create depictions of data and avoid miscommunication. *Thinking, Fast and Slow* by Daniel Kahneman is a recommended read that is loosely relevant and is about making decisions. He writes that "[o]ur comforting conviction that the world makes sense rests on a secure foundation: our almost unlimited ability to ignore our ignorance."

Office hours are 15:00 to 17:00 on Wednesdays. Arrive around 3. Two textbooks are assigned to the course. All of the information regarding formatting and implementation will be available in the **syllabus**.

NOTE Brightspace will be used for submissions and will have the syllabus and slides.

Group Projects. Will be responsible for 50% of the course mark and will be in groups (three to four members) and will be composed of a development of a prototype, scientific report, written project proposal (and 3 minute presentation), and demo presentation (15 to 20 minutes). Successful groups find an interesting domain with rich data. It *can* be done alone if a thesis can be used as a backbone for the project. Group definition is next week (September 11, 2019). Proposals are expected October 2, 2019. Other dates are on the slides. Create something to place onto your CV.

Assignments. There are four, each worth 10% of the course mark. Late assignments will not be accepted. Assignments are submitted electronically. These are due September 22, October 13, November 3, and November 24 respectively.

Seminars. Group seminars and presentations related to the course topic will be made (15 to 20 minutes) and be responsible for 10% of the course mark.

What Is Visual Analytics?

Big data, or too much data, can sometimes just have **noise**. It is difficult to differentiate the two. Removing noise is something we strive to do. There are common problems that will be discussed and should be avoided. With too much data, you can have increasing and increasing uncertainty; you can start getting unusable answers. Sometimes "enhancing" data can just mean you're adding more uncertainty into a problem.

Clive Humby suggests that "[d]ata is the new oil. It's valuable, but if undefined it cannot really be used. It has to be changed... to create a valuable entity... so much data be broken down, analyzed for it to have value." That's what this course focuses on: finding the **information** in **data**. Data is continuously increasing. Not all of it can be stored on disk. There is a large, growing, gap between data analysis and data. How can we cope with this? How do we survive this "*data deluge*"?

There are schools of thought that create a taxonomy between machine learning and data minings. This course will consider them the same. We can talk about this as one set of techniques and we can consider another set of techniques as **visualization**. However, they are beginning to become more similar - if combined in a smart way to solve a problem we have **visual analytics**. With some given defined domain and problem with a series of techniques put together in an explainable way that makes sense, we can begin to define hypotheses and perform exploratory tasks. With the remaining data, verification can happen.

What Is Visual Analytics? continued on next page...

Course Expectations. The university offers machine learning and visual analytics courses. Machine learning will be discussed in a very shallow manner. Visual representations will be discussed with regards to how applicable they are to each technique. By the end of the course, the following two questions can be answered:

- Do I need Visual Analytics solutions?
- Where or how to start?

Learning Outcomes. These are present on the slides and should be reviewed (slide 12). Remember there *are* limits to machine learning techniques.

NOTE Slides will be opened and available for download.

Data Pre-Processing

Normally, you need to do something to data to make it useable. This is **data pre-processing**. Data can be *collected* or *produced* (by sensors or simulations). Data can be *raw* (not processed) or *derived* using a process such as noise removal, scale, or interpolation. A lot of visual analytics will be encompassed with pre-processing data. Often, filtering needs to be done because not everything in a dataset is relevant to a problem. Redundancy can and will happen. Missing values can be present. Different scales could be present. The world is not a spreadsheet. Sometimes some spreadsheets can be too difficult to handle.

Data is composed of n instances and each one contains m (1 or more) observations or attributes (these are vectors). Each attribute can be a number, symbol, string, or a more complex structure. An attribute can be *independent* or *dependent* (affected by the variation of one or more independent attributes). Each attribute can be *ordinal* (numeric) and *nominal* (non-numeric). Ordinal can be binary, discrete, or continuous. Nominal data can be categorical (finite list of values), ranked (categorical and ordered), or arbitrary (infinite values without order such as an address).

Metadata can help to interpret data, providing information such as:

- Reference points for measures,
- Employed units,
- Symbol employed as missing values, and
- Measurements resolution

NOTE See the UCI Machine Learning Repository for a source of open data (`http://archive.ics.uci.edu/ml/index.php` - the *Iris* dataset will be used very often in this course), Statistics Canada, ...

Remember, better accuracy doesn't mean it has semantic meaning or makes sense. If you change a value that isn't supposed to be numeric and make it numeric and get better accuracy, you're not really achieving comprehension. It's meaningless.

Sometimes it is prefereable to visualize raw data (medical applications for instance). For some applications it is *necessary* to have some kind of pre-processing. In general, machine learning and visualization algorithms require data in a pre-defined format with certain characteristics (e.g., a distribution on the data). However, changing a distrubtion means you're influencing a classifier as well. Pre-processing should not affect these sorts of patterns.

The **data mining process** is taing data, selecting from that data, pre-processing it, transforming it, mining it to determine patterns and interpreting or evaluating that data in order to acquire **knowledge**.

Data Pre-Processing continued on next page...

There are companies that will actually do only this: pre-process data or *wrangling* data. An example is *Trifacta*. It takes a large majority of the time for data analysis.

Many issues can affect **data quality**: outliers, redundancy, invalid values, missing values, and defective distributions. Normally, distributions are usally gaussian. In nature, a lot of distributions are gaussian. Values will have limits and restrictions. For example, age cannot be negative. You should not expect someone will be, for instance, 200 years old. These should be checked.

NOTE There will be an assignment regarding data that has many issues. It is due this month; basically just Python (`scikit`).

Statistical analysis can provide useful information; even simple measures are important (max, min, standard deviation, ...). **Histograms** can be used to analyze the distribution. For example, the Iris dataset has two variables with a great deal of intersection in their distribution - excluding them will produce better classifcation. Statistical analysis can also be useful on *outlier detection* (very important - these should be removed), identification of similar clusters, and identification of redundant variables or instances using correlation.

Simple Method for Outlier Detection. There are some sophisticated techniques. This is not one of those. Suppose j has a Gaussian distribution. An average, standard deviation is computed; if something is too far outside it will be considered an outlier. See the slides (16). A constant c is derived.

Simple Method for Detecting Redundant Variables. Covariance and variance can be used to detect these values. This can be done but depending on what is being done but some classifiers are immune to redundant variables and it will not affect results. Be cautious; CORRELATION DOES NOT MEAN CAUSALITY. Normal machine learning algorithms will remove these already.

Missing Data and Data Cleaning. On "real" data it is also normal that values can be *missing* or wrong. Common strategies to address this is to **discard** the data (can be important data loss), assign a **sentinel value** (cannot be used for calculations; a symbol, for instance), or to calculate a **replacement value** (**data imputation**). Most often, data imputation is used but it can change the data distribution so be cautious.

Data imputation has two simple methods: (1) assigning the *average value* (can hide outlier values and flatten the dataset), and (2) assigning a value based on the *nearest neighbors* (on the neighborhood calculation it is difficult to know what the relevant attributes are).

NOTE Something that will be mentioned a number of times this course is related to distances (such as k-means). Something that is often done is acquiring groups by clustering. How does one find similarity between students, for instance? This can be useful for marketing analysis. It is not a Euclidean distance. For clustering, specifically, there are limitations. Be careful with what you do with missing data. This is a frequent but old discussion - everything we do is an approximation. There is rarely an exact solution. The most important thing is being cognizant of error and uncertainty. Neural networks are statistical methods, too. Many papers and algorithms are for specific datasets and not problems.

Normalization. On applcations that involve instance comparisons, one scenario can *distort* the result and introduce **bias**. Normally you need to do normalization. This is for when the Eucliedean norm of the vectors (line or column) are too different. These are incredibly important for distance or dot product calculations. Depending on the technique this can change everything for you. One method involves dividing by the unit norm; another process consists of transforming the data so the values range between 0 and 1 using minimum and maximum values. Another known method is *standardization* - transforms the data so that the average is zero and standard deviation is 1. **Standardization is very useful for PCA**.

NOTE Reference what resources and libraries you are using on assignments.

NORMALIZATION DISTORTS THE DATA AND IN THE PRESENCE OF OUTLIERS DATA CAN BE FLATTENED. There is, however, a huge problem with normalization; this is usually fine but if there is at least one outlying element this can affect all other elements.

Interpolation. If your data is a signal, one thing you can do is this. Sometimes it is necessary to fill the "space" between samples - the *linear interpolation* can be computed using a formula $x = (1-\alpha)x_j + \alpha x_k$ where alpha ranges from 0 to 1 inclusive (slide 29) - this is just a line equation and is what is done with regression. Interpolation can be used for visualization often to show information. For machine learning, osme techniques require more refined information or this may need to be done for different levels of granularity between data.

Sampling. It is possible to *reduce* the data using a sampling strategy to preserve some data property such as a distribution. The problem with sampling is there is a danger of losing information and that it is expensive. Remember, the pre-processing stage is assisted by a user and for large amounts of data sampling is necessary. Depending on distribution, random sampling is enough (check the random distribution). **Clustering** techniques can also be used to get cluster representatives as samples.

Dimensionality Reduction. Sometimes this needs to be done for a great deal of variables. This should preserve the information contained in the data and such reduction can be done by hand, selecting attributes, or using a technique such as **Principle Component Analysis** (PCA) which preserves distance and is very fast to compute, **Multidimensional Scaling** (MDS), and **Self-organizing Maps** (SOM). If the covariance matrix is too expensive to calculate for PCA (a linear transformation using matrices; PCA is just a rotation in the n-dimensional space), you can just create a **random projection** (using a randomly generated matrix).

NOTE Be careful about "random". It is typically pseudo-random. A distribution is followed that could be linear, Gaussian, etc. Java and C are Gaussian. FORTRAN is linear.

Mapping Nominal Values to Numbers. In case of ranked nominal values (air quality), there is a straightforward mapping into consecutive integers. In case of categorical values (car type), these can be transformed by expanding into binary values, one column for each category. Assigning numbers implicitly assigns weight and affects distances. Binary values are not integers; no distances can be calculated. YOU CANNOT CALCULATE THE EUCLIDEAN DISTANCE ON BINARY NUMBERS but binary values can often or are often omitted for classifiers such as with SVM. For non-ranekd values, the problem is more complex (car name). If there is only one arbitrary nominal variable, we can use *correspondence analysis* - a numerical value can be assigned using the other variables to calculate a distance matrix, applying MDS to calculate unidimensional coordinates. This creates a one-dimensional representation to preserve distance. Depending on the domain, this could be marginally useful.

Summary. Sometimes it is necessary to transform or derive data (interpolation, sampling, dimensionality reduction, mapping nominal values to numbers). Sometimes raw data needs to be converted (text, images, audio). IF DATA WERE TRANSFORMED THROUGH SOME PROCESS; THIS NEEDS TO BE INFORMED TO THE USER OR ANALYST.

NOTE Neural networks, specifically DNNs, can process raw data, which is very amazing because that means the transformation is not necessary. You can just input images.

Distances. The way the distance between two objects is calculated is very important. A histogram (of the pairwise distances) can help to understand the distance nature. For sparse datasets, you usually have very small groups. Normally, Euclidean distance is what is in mind. To be a distance metrix, some properties need to be obeyed (non-negativity, identity, symmetry, triangular inequality - for instance, some classifiers cannot use cosine distance because it is not really one). See slides on family of distance. Not all dissimilarity is a distance metric - do not need

to obey the metric properties. A dissimilarity can be the inverse of a similarity. A well-known example is the cosine dissimilarity (in text mining, sparse spaces). For binary data, specific metrics should be used (simple matching, Jaccard's distance, Hamming distance, ...). Which one? No tried, true method but whichever one you choose, try to consider why it is relevant for the problem or domain.

NOTE It's becoming more and more easy to just use a library for data analysis and classification. But that sometimes hides techniques and decisions. For instance, what distance metric is used? You need control and awareness of the process. There is a new trend called "auto machine learning" that does all of this work for you.

Visualization

Visualization as a field was born 1994; it is a rather young field. For instance, word clouds was the first time a visualization was made with respect to text documents. Today, they are everywhere and is becoming more widespread. Bar graphs, line graphs, scatter plots were not created by computer scientists. They were there before. But, other techniques began to get introduced and are now more widespread and are in other fields as well. This part of the course will focus on a number of specific techniques and what they can do as well as their limits.

In the past, visualizations were used to support hypotheses. In fact, as datasets begin to grow, it is not only for this. It can be used in an exploratory phase and the hypothesis testing will involve statistical analysis and machine learning. This entire course involves itself with the notion that machine learning is incredibly difficult for creating hypotheses but visualization can allow one to create a hypothesis and use machine learning to support it.

A vizulization is the communication of information through graphics. A single image can contain a large amount of information and can be interpreted faster than text. Imagine interpretation is done in parallel on the perceptual system; text is sequential. Image is also language-independent. However, colors can depend on culture. Interpolation can be helpful. Reading spreadsheets alone without this support isn't enough sometimes. There are limits. Human **interaction** is *crucial* for:

- **Exploration** of data,
- The **generation** of hypotheses,
- **Interpretation** of results, and
- **Steering** of the analytics (e.g., relying solely on accuracy is not a good idea)

Visualizations are interactive. This is an important step - it allows for exploratory tasks. In practice, there is a large amount of data stored and it is hard to know where to start in domains. Exploratory tasks are a way to allow the user to check for patterns. Not every correlation is a valid one, though (remember causality). A visualization can be used for different reasons:

- **Presentation**: present concepts or facts to an audience,
- **Confirmation**: there is a hypothesis and the user wants to *confirm or refuse it* - should be a mix of visualization and machine learning, and
- **Exploration**: there is data and certain properties are sought out to be verified.

NOTE Do not use 3D pie charts.

A number of articles in the literature will try to use visualization for confirmation but statistics are far more relevant and useful.

Visualization continued on next page...

VISUALIZATION IS PARTICULARLY USEFUL FOR THE EXPLORATORY ANALYSIS.

The interpretation of results from machine learning has a number of techniques that help the process. Depending on the problem, explanation of the results is just as important if not more important than accuracy. Why does a patient have a high severity symptom, for example. Or, regions may have higher violence indexes - why? With deep neural networks, high accuracies can be obtained - but how do you interpret those results? It is not based on rules. Some rules *could* be extracted - and this is why trees are used so often.

Some algorithms have parameters and they need to be changed to observe what happens. For example, with clustering there are some parameters that can be adjusted that dramatically change the results.

NOTE The course is a combination of machine learning and visualization. There will often be different amounts/levels of both, particularly on projects. Some problems will have different requirements and there will consequently be different constraints.

When *analyzing a graph*, first detect groups of objects (pre-attentive), then categorize the groups (cognitive) and understand them. Finally, we analyze the special cases that were not grouped.

NOTE Gapminder is a nice tool to start doing something visual without programming. Allows questions to be answered based on data. Invites journalists to answer simple questions - the majority are wrong. The hypothesis for this is that these journalists have knowledge from 30 or more years ago which differ from current knowledge models. There are a number of things assumed to be true but are just hypotheses. For example, climate change and global warning involve hypotheses and evidence but not necessarily truth. CORRELATION IS NOT CASUALITY.

NOTE One concern for gathering knowledge is **data access**. This isn't enough. What should be available is data analysis.

The **visualization process** defines.a mapping between data and graphical elements displayed onto a screen. Interaction is also an important aspect of the visualization process and can be part of a larger process (*knowledge discovery*). *Visual mapping* involves the mapping of multiple attribtues into a visual representation and this involves clever thinking. KNOWLEDGE DISCOVERY IS THE ULTIMATE GOAL ENSURING CORRELATION IS NOT MISTAKEN FOR CAUSALITY. Human beings are pattern recognition machines.

A common way to translate data into visual representations is to map each data attribute into a different *visual marker*. It is possible to use up to eight visual variables:

1. Position,
2. Shape,
3. Size,
4. Brightness,
5. Color,
6. Orientation,
7. Texture, and
8. Movement

Position (in 1D, 2D, 3D) is the most important variable. Spatial arrangement is the first thing read on a visualization. Humans consider two things that are close to be similar. Shape is the next important and involves using any graphical element as a marker; these should be as different as possible but have similar areas and complexities.

Visualization continued on next page...

Using color is very bad when there are a large number of categories. After around seven colors, it is hard to distinguish the colors. Changing the way these are displayed can be done by, for instance, changing the size of the markers. Size (length, area, volume) can be used to map continuous and categorical variables. It should not be used for categorical variables. Brightness, again, is useful only for continuous variables. Values are mapped to colors normally using a colormap. There are specific colormaps for continuous and categorical values (only few classes). Rainbow scales are awful for continuous or categorical values (especially the former); the reason being is that it is hard to distinguish the small differences. Orientation, or direction, is pre-attentively processed. It cannot be used for all markers; it has problems with more than one natural axis. Texture used to be used in the past (when color was not used); it can be understood as a combination of other variables such as shape, color, and orientation. Movement is useful but mostly when to indicate temporal variation. There is a paper that strongly advises not to, however.

NOTE Colorbrewer is a very good resource for colormaps and provides good color advice using cartography as an example domain. D3 offers some. Diverging color scales are usually good for negative, neutral, and positive continuous values. Do not use color scales that you like most. It is not a competition for beauty. Follow some rules.

IMPORTANT: Any screen pattern should imply a *data pattern*. Otherwise, it is an artifact. This can be done very easily. If something is on the screen, it should be in the data. For example, three-dimensional pie charts can often hide numbers and add depth and perspective that hide similarity. It is often an indicator of trying to lie to deliver a different message without using false data. Any *order (magnitude) perceived* in the visuals should reflect some *data property*, as well. Similarity between data instances should also imply a similarity between the symbols representing them. *Difference choices of mapping can affect visualization effectiveness.*

NOTE 3D visualizations are typically bad because of occulusion in the mapping to 2D screens.

A lot of this can be intuition. User tests can be useful in order to determine what is best. At university, this involves a process of beauracracy with ethics.

NOTE The harm of visualization is to fool yourself or fool others by implying causality with correlation. See *Spurious Correlations*.

Shneiderman proposed a taxonomy according to data type and tasks. Keim created a classification based on three dimensions: data type, visualization techniques, and interaction methods and distortion. These taxonomies are not going to be discussed further and change over time. There are a number of data types such as unidimensional, bi-dimensional, algorithms and software (software, memory, etc.).

The steps for visualizing data include:

1. Find out what kind of data you have (*data type*),
2. Determine the *questions* that need to be answered by the data,
3. Derive *requirements* from these tasks (such as anonymizing individuals),
4. *Preprocess* the data,
5. *Design* the visualization, and
6. *Evaluate* and improve it (this doesn't really have to be done for a company but does for research)

Visualization continued on next page...

NOTE Geographical coordinates should always include a map.

Going Further into Visualization

Multidimensional Visualization

The techniques discussed here can only be used for data tables that do not have **spatial attributes** (geographical). You need to use maps for that case. It is common to classify multidimensional visualization techniques into:

- Point-based techniques,
- Line-based techniques, and
- Region-based techniques

POINT-BASED TECHNIQUES (like a scatterplot) are projections of m-dimensional instances into a p-dimensional visualization space ($p = 1,2,3$) where a glyph is associated to each point. Be aware it is hard to make pairwise comparisons between shape and color. Scatterplots are the most common of visual representations but as dimensionality increases new strategies must be sought using

- **Dimensional selection**: manually or with some algorithm
- **Incorporate dimensions**: map other dimensions to graphical elements
- **Multiple displays**: showing using superimposition or juxtaposition

Scatterplot matrices are grids of scatterplots showing all combinations of dimensions and pairwise graphs. Brushable scatterplot matrices are useful to link points between all dimensions and indicate their position when selected. Some papers are indicated for reading (slide 8). Scatterplots still only let you compare two variables at a time which is a large limitation. There are techniques to project multiple dimensions into a scatterplot.

Multidimensional projections keep, as much as possible, the relationships found in m-dimensional space into a p-dimensional space. For instance, this would include similarity relationships, neighborhoods, etc. The result is a set of points on a plane where close points indicate related instances and distant ones are non-related (Euclidean distance). Finding the distance is a problem to also be considered in the original space; a function needs to be determined. What happens is an optimization of a function such that the difference of the distances in the original space and new space is incredibly small. More can be done to adjust this. PCA, LDA are examples of this technique. See the HDI index example.

NOTE Do you want to be very precise globally or locally? Local differences are more precisely determined in certain multidimensional projection techniques and likewise globally. One technique used very often creates groups but the groups are not useful relationships globally (TCD?).

Radviz is another point-based technique. For an m-dimensional dataset, m anchors are created and distributed over a circumference. See slide 18 for calculations on positions of anchors. It is relatively easy to implement. New instances are mapped onto a circle. If all attributes had the same value, their position in the circle will be the centre. The position of the anchors is very important. A data instance is mapped to a position v on the convex hull of the anchors (this relies on convex hull combinations). The order of the anchors can change the representation. All of them are valid. The best way to position anchors is to have the largest spread of points but the number of combinations is large.

NOTE An assignment will ask you to implement Radviz from scratch using D3.

LINE-BASED TECHNIQUES use line crossings, curvatures, etc. You can sometimes use visualizations to select features for classification. *Superimposition* involves putting lines on top of each other. *Stacking* positions them vertically further away from each other to avoid occlusion problems. But this keeps summing values; however, it is useful to see trends, drops and patterns, but not necessarily understanding the numbers. The problem with this first line-based technique is that it can only be used if the units of the dimensions are related. This is very useful for time series data.

One solution to different units is **parallel coordinates** which is a mathematical concept as well as a visual representation. The axes are parallel (non-orthogonal) and the data instances are represented as polylines that cross the axes on positions proportional to the value on the dimensions. Relationships can be spotted this way and multiple attributes can be considered without problem. Order of attributes can also be changed. Outliers can be relatively easy to spot, too. Very helpful for exploratory analysis. This technique is considered an incredibly powerful tool but does have some potential cluttering.

You have to ask yourself what sort of tasks this technique is useful for, though. Too many classes, is an issue, for instance. It is useful, though, for the overview of patterns or finding outliers but not individual instances. You don't have to show all of the lines though - you can also show *aggregations* after creating groups. This is not exclusive to this technique but is an exercise that can be done with any visual representation. There are a number of variants to this technique to try and overcome some of these limitations including clustering, compression of lines or finding correlation between attributes and ordering based on correlation.

NOTE Parallel coordinates very useful for multidimensional exploratory analysis.

For automatic *axes reordering*, there are a couple of papers that can be read (slide 32).

RADIAL AXES TECHNIQUES. For each technique with parallel axes there is a technique with **radial axes**. This technique is incredibly useful for comparing a small number of classes by creating shapes using polygons created by connecting attribtues. It is also available in D3. For layouts with more than one cycle, concentric circles or spirals can be used (particularly when concerning time).

REGION-BASED TECHNIQUES present data values using filled *polygons* varying the size, color, shape, and other visual attributes. Although it presents several cognitive problems, many techniques have been developed. One of the most common visual representations are *bar graphs* (stacked bar charts, grouped bar charts, 3D histograms).

NOTE Bring slides and present for 3-5 minutes next week - **proposals** are due. Feasibility does not matter. On the proposal, though, be sure to focus both on ML and visualization - they need to be linked in some way. They are not independent. Could visualize the probabilities of belonging to each class of a classifier. Allow the user to interact with the system/model. Labels are not always present in datasets.

3D versions of bar graphs are called **cityscapes** and are often used for georeferenced data. Remember the important rule of visualization: if there is a geographic information, use it and display it on a map. It is one of the most important types of information.

Tabular Displays. It is easy to generate visual representations of the data as tables. *Heatmaps* map data values into rectangles filled with colors given by a color scale. One problem with heatmaps is the question of order. Another tabular display are *correlation matrices* but order also is a question here.

Permutation and reordering is important and allows you to see patterns that you may not otherwise be able to do. There are several ways to perform this reordering. Creating dendrograms connecting the most similar elements

manifests as *hierarchical clustering* and generates similarity trees.

TableLens combines these ideas and adds mechanisms to visualize all the tabular data while providing a detailed view of the data. In order words, filling in the cells of a spreadsheet with bar data.

DENSE PIXEL DISPLAY TECHNIQUES map each data value to individual pixels, creating filled polygons to represent each dimension. The most basic approach creates an image per dimension. The images of each dimension can be placed in different manners on the screen. A projection method can also be used to place similar instances close together.

COMBINING TECHNIQUES. Some techniques combine the features of two or more classes of techniques. One way to do this is to create glyphs - images for instances that can be assigned with different types of mapping:

- One to one,
- One to many,
- Many to one

See slide 61 for more information and examples. Mapping more than four layers on a map is very difficult to do. Commercial tools have difficulty with this problem. There are also three different strategies to position glyphs:

- *Uniform*: glyphs are scaled and positioned with equal spacing between them to fill the entire screen - avoid overlaps,
- *Data-driven*: data values determine the position of glyphs, and
- *Structure-driven*: if there is any implicit structure such as cyclic or hierarchical, it can be used to position the glyphs

Graph, Network, and Tree Visualization

Visualization techniques may also represent relationships between hierarchical relationships (part/sub-part, parent/child), connections (i.e. cities connected by streets), and derivations (i.e. sequence of steps or stages). Relationships can be simple or complex (directed, weighted).

Hierarchical relationships (trees) are the most basic. We can split techniques for visualizing trees into (1) *space filling methods* and (2) *non-space filling methods*. The former use as much as possible the available visual space where juxtaposition is used to represent the connection between nodes.

The most common approaches for **space filling methods** are **rectangular** and **radial representations**. You do not need to use edges to indicate relationships. *Treemaps* are the most popuar rectangular representation where a rectangle is recursively divided into pieces alternating horizontal and vertical cuts based on sub-tree size at given levels. Recognize that the rectangles are proportional to something.

There are a number of different algorithms to implement treemaps. The most common is *slice-and-dice* and is incredibly simple, alternating vertical and horizontal slicing at each level and making the number of children the size of the elements. See slide 7 for other variations of this technique. The problem with treemaps is that it tends to create very thin rectangles and very strange shapes.

The most modern implementation is the *squarified treemap* which ensures the aspect ratio is approximately 1 (or as close as possible to squares). The problem with treemaps has been known for many years. But squarified treemaps comes with its own problems. Pushes everything towards a single corner with the smallest elements in that corner.

Another set of space filling method include *radial space filling* techniques, also known as sunburst displays which places the root of the hierarchy in the center of a display and employs nested rings to represent the hierarchy.

Graph, Network, and Tree Visualization continued on next page...

The most common method for displaying hierarchical relationships with **non-space filling methods** is the **node-link diagram** - this is a very loose definition which can have many variations in how they are drawn. As a slight departure, recognize that in the design of algorithms for drawing diagrams some factors need to be taken into account.

- *Draw conventions*: straight edges, polygonal lines, or curves, position nodes in a grid, all siblings are in the same vertical position, etc.
- *Restrictions*: positioning a particular node in the centre of the drawing, positioning groups of nodes close to each other, etc., and
- *Aesthetics*: several rules that greatly influence the final interpretation.

Aesthetic rules can be to minimize cross-lines, keeping a nice aspect ratio, minimizing total drawing area, etc. (see slide 13). These are mostly for beauty. For trees it is relatively easy to respect these rules. A simple algorithm is described on slide 14.

A number of tree layouts exist including radial layouts and collapsible trees.

Trees are easy. They are a specific type of **graph** (connected, unweighted, acyclic). There are specific techniques for visualizing general graphs; two examples are **node-link diagrams** and **matrix representations**.

Force-based methods provide a way to transform the graph in real-time by enforcing physics and use a spring analogy to represent the edges. There are two forces between nodes (attraction/repulsion f_{ij} between connected nodes and repulsion g_{ij} avoiding nodes get too close).

NOTE Strange fact in research life: if your paper is too complicated, it will just be accepted.

The simplest model uses Hook's laws to represent the force of a spring and the inverse square law to represent the repulsion. So, for each vertex, you consider its neighbors and create vectors for these forces. As an approximation, you can walk the sum or each of the individual forces and their x-component magnitudes (slide 23) and frames an optimization problem where results converge to a local minimum (the system doesn't ever really fully stabilize because the forces do not decrease over time; if you don't stop it, it will just start rotating). Instead of a maximum number of rotations, you could enforce a decay on the forces (*simulated annealing*).

NOTE This is an expensive visualization technique ($O(n^2)$) - up to about 4000 nodes it is not really a problem, though.

Circular layouts layout nodes on the edge of a circle and has all its edges within the circle. However, recognize order is a problem again. A more recent technique is using *edge bundling* which groups/clusters edges to common paths based on proximity. It involves creating a tree and relating it to the graph.

Edge bundling can also be extended to a more novel technique called *force-based edge bundling* which is incredibly expensive (mostly GPU-bound). This is just one algorithm; there are a large number of algorithms for edge bundling.

In **matrix representations**, an adjacency matrix is visualized and there are specific strategies for reordering, from user-centric to automatic. Recognize, though, this is an NP-complete problem. The order of the matrix is important in depicting patterns.

A *hybrid representation* **NodeTrix** combines node linking with a matrix which replaces a set of nodes with a matrix. This is a very powerful tool for very large graphs and depict inter-relationships.

Data Mining

Introduction to Data Mining

Data Mining has many definitions. One is the non-trivial *extraction* of implicit, previously unknown and potentially useful information from data. It could also be the *exploration and analysis*, by automatic or semi-automatic means, of *large* quantities (in terms of complexity, not size; data mining is not only about large data sets) of data in order to discover *meaningful patterns*. And, finally, another definition is the *prediction* of outcomes of future observations. Trying to understand the results (*Interpretation/Evaluation* in the data workflow) can be done with visualization.

So what isn't data mining? Looking up a phone number in a phone directory, for example. Or querying a web search engine for information about "Amazon". This is **Information Retrieval**. However, identifying that certain names are more prevalent in certain US locations (O'Brien, ...) *is* data mining. So is grouping together similar documents returned by a search engine according to their context ("Amazon rainforest", "Amazon.com", ...).

NOTE The **StarQuest series** on YouTube provides a good breakdown of complicated statistics and machine learning methods as *video guides*.

Data mining originated from ideas of machine learning/artificial intelligence, pattern recognition, statistics, and database systems. Traditional techniques may be unsuitable due to the *enormity* of data (does not fit into memory — need to use distributed databases and/or online techniques), *high dimensionality* of data (statistics change with high dimensionality — remember that with some large number of dimensions, the distance between closer and further elements is very small), and the *hetereogeneous*, distributed nature of data (for example, health data can involve images, etc.).

What has changed over time? Better and cheaper computers.

NOTE Visualization has its own course in addition to Data Mining having a number of courses available. This course focuses on Visual Analytics. In this course, anything that is not a Visualization component will be referred to as Data Mining.

NOTE Increasing the number of attributes is not a good thing - this is the *curse of dimensionality*. Even if there is claims being made about acquiring more data to make meaningful decisions, too many attributes can also be a problem for techniques.

Data mining tasks include **Prediction Methods** (using some variables to predict unknown or future values of other variables) and **Description Methods** (finding human-interpretable patterns describing data). For example, prediction methods include **classification**, **regression**, and **anomaly detection**. Description methods include **clustering** and **association rule discovery**.

Classification is, given a collection of records (**training set** — need to trust this; not always trustworthy) where each record contains a set of attributes and one of them is the class, finding a model with class attribute *as a function* of the values of other attributes. The goal of classification is to assign a class, as accurately as possible, to previously unseen records. A **test set** is used to determine accuracy and, usually, the given training set is divided into training and test sets for validation. See *cross-validation*. It has limitations but nothing is better than Support Vector Machine (SVM).

The training set is not always completely reflective of truth. For example, imagine a *recommendation system* such as found on Netflix. What if you change your mind or tastes?

Introduction to Data Mining continued on next page...

One application for classification is direct marketing (targeting individuals likely to buy a new product by considering profiles and purchase histories — very powerful classification and demographic information is publicly available). Another application in South America involves credit (financing) and lending money and preventing a wait time, even for small purchases, but with an incredibly high interest rate (50%). This doesn't happen from banks, but stores.

Regression predicts a value of a continuous valued variable based on values of other variables assuming a linear or nonlinear model of dependency. This is greatly studied in statistics and neural network fields. An example includes stock market indices — this is a common use case but it is not trivial. It is very difficult to predict because other variables influence the stock market (such as political debates).

Clustering is, given a set of data instances each with a set of attributes, and a similarity measure, finding groups or clusters based on similarity — data instances in one cluster are more similar to one another and those in different clusters are less similar to one another. Similarity measures include Euclidean distance (if attributes are continuous) or other problem-specific measures. This is an optimization problem where intracluster distances are minimized and intercluster distances are maximized. In practice, this is not trivial and many points will exist between clusters.

One application for clustering is market segmentation (finding subdivisions of the market into distinct subsets of customers where any subset may conceivably be selected as a market target to be reached with a distinct marketing mix) which relies on having a good distance metric and knowing why and in what way clusters differ. This last step, *explaining the clustering*, is the most important step that is commonly missed.

Association Rule Discovery is, given a set of records containing some number of items from a collection, producing *dependency rules* which will predict occurences of an item based on the occurences of other items. These records are typically transactions and datasets can be transformed into these sorts of records. These rules are *causality* and not correlation. Some parameters can be controlled (indicating how trustworthy the rules can be).

One application for association rule discovery is marketing and sales promotion. Each rule has a *consequent* (what should be done to boost sales) and an *antecedent* (shows what products would be affected if any of these are discontinued) and knowledge can be obtained by considering the antedecent and consequent together. Another application is supermarket shelf management — this is a very classical example involving identifying items bought together by sufficiently many customers. The approach is processing point-of-sale (POS) data collected with barcode scanners to find dependencies among items. The classic rule is that if a customer buys diaper and milk, he is likely to buy beer.

The problem with association rules is that it creates a tremendous amount of rules. A better visual representation is needed.

Anomaly detection detects significant deviations from normal behavior by constructing a profile and considering instances with respect to that profile. Sometimes this is considered classification but it is not. One application is credit card fraud detection which predicts fradulent cases in credit card transactions. Another is network intrusion detection (the typical network traffic at the University level may reach over 100 million connections per day).

Classification

Classification only makes sense if you have new data incoming whereas clustering involves not having classes. There are many different classification techniques but only up to three will be discussed. For successful classification, the accuracy of both the training and test sets needs to be high. The model that is created *cannot generalize* — it only predicts what it has been trained for. Because of this, there are problems of overfitting and underfitting. The process of learning with a training set is called *induction* and applying the model on a test set is *deduction*.

Techniques include:

Classification continued on next page...

- Decision Tree-based methods,
- Instance-based method (k-NN),
- Naive Bayes and Bayesian Belief Networks,
- Neural Networks (do not create groups; only deals with probabilities), and
- Support Vector Machines (SVM).

For the purpose of visualization, only the first three are really relevant and produce very clear predictions. It is known that neural networks are incredibly good, but not only do they need to be good, they also need to be *explainable*. This is incredibly relevant today in the European Union (EU) that requires corporations to explain the results of their computers. Trees are very powerful for this — they are very explainable. Be careful with trees, though.

Decision Tree-based methods involve a *tree induction algorithm*. Trees are correct approximations, but there could be more than one tree that fits the same data. It is possible to describe the best tree but to find that is an NP-Complete problem. So, heuristics are required. Applying these models to test data is trivial and fast. With a good data structure, this can be done almost instantaneously. There are many tree induction algorithms: Hunt's Algorithm (one of the earliest), CART, ID3 or C4.5, and SLIQ or SPRINT. See slides for general structures. *Hunt's Algorithm* is a recursive algorithm that splits on classes (and creates *binary trees*). Order of attributes matters.

In practice, these algorithms are greedy and optimize certain criterion. Issues include determining how to split records and when to stop splitting. Creating a tree algorithm can sometimes be a good idea depending on your data. Specifying test conditions depends on attribute types (nominal, ordinal, continuous) and depends on number of ways to split (2-way, multi-way). With ordinal attributes, binary splits become troublesome — you cannot really split larger and smaller ordinal values on one side and need to control splits. Continous attributes can be handled with discretization or binary decisions (the latter is more computationally intensive but this is only for model creation and not classification). Determining the best split often comes down to *purity* (ideally, a single class decision and finding the shallowest possible tree). So, in the greedy approach, nodes with homogeneous class distributions are preferred — need a measure of node impurity (examples include GINI Index, entropy, and misclassification error). In practice, GINI and entropy create very similar trees. The problem with these definitions is that they create many partitions but we need to avoid this (it's not just depth but also number of nodes). Another technique to combat this is using information gain (used in ID3 and C4.5 — the latter is the most used algorithm).

In summary, these algorithms have *no prior assumption on class or attribtue distribution*. Constructing the optimal tree is NP-complete. They are inexpensive to construct and extremely fast to classify with. They are also easy to interpret (for small trees). The accuracy for these techniques are comparable to other classification techniques for many simple data sets. For more complex data sets, it doesn't really work (but **random forests** can help). It is also robust to the presence of noise and attribute redundancy. Irrelevant attributes, however, can impair the result (preprocessing is essential).

NOTE Given demographics, it has not been found for there to be a correlation between demographics and GPA in college students.

NOTE Install `scikit-learn` and it can be used for classification.

What is a good accuracy? It depends on how much a drop in precision matters. Anything below 60% is random for binary classification. For the EU, a drop in precision is okay as long as the model is explainable. Also, knowing the probability for an instance can be important too because it can tell you how trustworthy a result is.

Classification continued on next page...

Underfitting and **overfitting** is important to consider. There are two kinds of classification errors: misclassification on the training records (training errors) or generalization (test) errors which is misclassification on unseen records. Good models must not only fit the training but have to accurately classify unseen records — you want to account for both training and generalization errors. Underfitting is when training and generalization errors are high. Overfitting is when the training error is low but the generalization is high. It is possible to get very high accuracy for a training set but then you will have very high overfitting. Avoiding overfitting is more important than underfitting.

NOTE Research typically has benchmark data sets but, in practice, a lot of problems exist with data. So existing paper techniques might not work in practice.

Occam's Razor states that given two models of similar generalization errors, one should prefer the simpler model — this avoids overfitting. Model complexity is important when evaluating a model. The best model is a tradeoff between complexity and performance. **Random forests** are another good way to solve this problem — subsampling of attributes is done to create tree with the maximum depth and consequently varrying complexities. A voting system is used to determine the best tree. This is a brilliant technique because it reduces complexity and attribtue numbers at the same time *and* many different decisions are being made at the same time (it is more generic than a single tree).

NOTE Next week there will be a tutorial on frontend and backend integration.

Bayes Classifiers are *probabilistic frameworks* for solving classification problems and model *probabilistic relationships* between the attributes set and classes. There are different implementations including Naive Bayes and Bayesian Belief Networks (BBN). Recognize there is always a tradeoff between complexity of a model, and its accuracy, and interpretibility. Bayes is relatively easy to understand and extract information from - it is particularly useful for text.

Bayes Theorem relies on conditional probability. See slides. Furthermore, this formulation also indicates you can understand the importance of each respective attribute. Bayesian classifiers consider each attribute and class label as *random variables*. Given a record with attributes $(A_1, A_2, ..., A_n)$ the goal is to predict a class C that maximizes $P(C|A_1, ..., A_n)$. Can this be estimated directly from data? The approach of the classifier is to compute the posterior probability for all calues of C using the theorem.

The probability of $P(A_1, ...A_n|C)$ cannot be computed from the data but we can take a *naive* approach. Naive Bayes assumes CONDITIONAL INDEPENDENCE among attributes when a class is given - $P(A_1, ...A_n|C) = P(A_1|C)...P(A_n|C)$. We can estimate $P(A_i|C_j)$ in all cases, particularly for discrete attributes (for continuous attributes, we use a probability density estimation function and assume attributes follow normal distributions). Be careful though - if there are relationships of dependence among attributes this assumption *does not hold*. You *can* calculate the correlation between attributes to mitigate this.

NOTE Remember for the project, do not go strictly by accuracy. Go for useful knowledge when choosing a classifier.

Naive Bayes is *robust* to isolated noise points and to irrelevant attributes. However, the conditional independence assumption may not hold for some attributes (problems with correlated attributes; use other techniques such as BBN in that case). Because you can attribute probabilities to the results of this classifier, this technique can be very defensible.

Artificial Neural Networks (ANN) are incredibly complicated. Something important to know though is that they are *just equations* that classifies weights for combinations. It is a mathematical function that produces an out-

Classification continued on next page...

put that can be transformed into a classification. Commonly people call them "black boxes" but this is arguable because you get weights back. We also cannot quite understand their rules, though.

The model of an ANN is an assembly of inter-connected nodes and weighted links. Output node sums up each of its input values according to weights and the output node is compared against some threshold. You can also stack layers, but this increases its complexity. When these weights are estimated, this can be very expensive for larger and larger networks. Specific hardware can be purchased, such as GPUs, to accelerate these calculations, but in a lot of ways this can be argued as a way of throwing hardware at a problem that may be solved with better techniques.

Training ANNs means learning the weights of neurons. For some applications, these are incredible - such as for self-driving cars. Or for recognizing faces. For other applications, though, when the answer needs to be explained, this technique is not very defensible. ANNs are difficult to use for visualizations. Some approaches involve creating trees to estimate the answer of a complex neural network as a means to explain results. But, the vital question is: if you can have an explainable model to estimate a complex model, why was that not obtained in the first place with a different technique? See `playground.tensorflow.org` for a way to visualize neural networks in practice and real-time.

Training data for neural networks need to be proportional to the complexity of the model, defined by the number of nodes. With insufficient amount of instances of data, they are not as reliable or trustworthy. Therefore, Deep Neural Networks require tremendous amounts of training data. For example, all of Wikipedia was thrown at a number of networks in order to ascertain the distances between terms such as "king" and "queen". See `teachablemachine.withgoogle.com` for a hands-on real-time neural network classifier using a webcam.

NOTE Active learning is an approach to problems where training data is not available which is based on SVMs.

NOTE Recognize that when you perform literature surveys, you start with a great deal of papers down to a subset that identifies the state of the art (a matter of identifying relevance). An SVM can help but it lacks rich features.

With small training data counts, it is better to use something like **Support Vector Machines** (SVM). It is difficult to beat them for small amounts of training data. The purpose of an SVM is to find a *linear hyperplane* (decision boundary) that will separate the data (there can be more than one possible solution but there is a solution that is the best possible - lines should separate as best as possible by maximizing distances). It is very expensive and requires rich features. It is a constrained optimization problem that can be solved using quadratic programming.

NOTE There are ways to project nonlinear datasets into infinite dimensions where the data is linearly seperable. This requires preprocessing to ultimately separate the space.

What if the decision boundary is not linear? We need NONLINEAR SUPPORT VECTOR MACHINES which transforms data into higher dimensional spaces. This requires the use of a kernel (a kernel is a function to transform data that replaces the dot product - the SVM does not really change but there are multiple kernels).

Model Evaluation. How do we evaluate models? What metrics do we use? The focus is on predictive capabilities of models rather than how fast it takes to classify or build models, scalability, etc. We commonly use a *confusion matrix*. The most widely-used metric is accuracy; but accuracy is misleading because you could easily ignore single classes of minority representations. Therefore, we can use precision, recall, and F-measure. The larger the precision, the lower the number of false positives. The larger the recall, the lower the number of false negatives.

Clustering

Clustering has multiple solutions that are acceptable; it is subjective. The notion of a cluster can be ambiguous. **Cluster Analysis** is the idea of finding groups of objects such that groups are separated (intra-cluster distances are minimized and inter-cluster distances are maximized) or *segmented*. It is not a supervised classification (training data, ...) - groups are not classes. It is not simple segmentation. It is not the results of a query. It is also not graph partitioning (some mutual relevance and synergy, but areas not identical).

Graphical representations can control, up to a degree, subjectivity and can force the perception of clustered groups.

There are different types of clustering. **Partitional clustering** (such as k-means) divides data into non-overlapping subsets (clusters). **Hierarchical clustering** creates a set of nested clusters organized as a hierarchical tree (*dendrograms* that can be cut at different levels to form clusters).

We can also talk about **exclusive** versus **non-exclusive clustering**, as well as **fuzzy** versus **non-fuzzy clustering**. The latter involves membership weights that sum to 1. Probabilistic clustering has similar characteristics. We can also talk about **partial** versus **complete clusters**; in some cases, we only want to cluster some of the data. There is also **heterogenous** versus **homogeneous clustering** (clusters of widely different sizes, shapes, and densities). For example, k-means requires uniform shapes (realize that if you increase the number of dimensions, everything can lie on a sphere so k-means works very well).

Clusters can also be characterized as being *well-separated* (combinatorial), *prototype-based* (such as k-means, more common - looks at the center of a cluster; the **centroid**), *contiguity-based* (nearest neighbor or transitive), and *density-based clusters* (dense regions vs low-density; ignores noise).

The most common algorithms are k-means and variants - the basic algorithm of which is very simple and converges to a local optimum, hierarchical clustering (agglomerative), and density-based clustering such as DBSCAN (very popular but there are newer versions).

k-means has often randomly chosen initial centroids (so clusters can vary from one run to another but the probability of selecting one centroid from each cluster randomly is infinitsimally small). *Initial seeding is vital for choosing the final result.* Usually converges within a few iterations. Clusters are typically evluated using Sum of Squared Error (SSE); given two clusterings, we can choose the one with the smallest error - want to minimize and converge to a local minimum. The basic algorithm can also yield empty clusters which can be solved by choosing a point to create a new centroid (and split). Recognize despite the simple premise, a lot of heuristics and decision-making is involved. DO NOT TRUST CLUSTERING RESULTS.

A variant of k-means, **Bisecting K-means**, can produce partitional or hierarchical clusters. It is less sensitive to initial seed, and faster because it tests locally. If the final goal is to segment data, this algorithm is very good for sampling.

Overcoming the limitations of the algorithm and its variants involve increasing the number of clusters and combining them. Deciding the k in k-means is difficult to do, in practice, but with greater computational power you can just test everything. And heuristics have been made; traditionally the $\sqrt{n}$ has been designated as the acceptable maximum number of k.

Hiearchical clustering produces a set of nested clusters visualized as a dendrogram (a tree-like diagram recording the sequences of merges or splits and the distances between them) (see neighbor-joining in bioinformatics for phylogenetic trees). With these techniques, you do not have to assume any particular number of clusters (any desired number can be obtained with cutting).

The two main types of implementations are **agglomerative** and **divisive**. The former is more popular.

Deciding the proximity between clusters forms different techniques: `MIN`, `MAX`, and group average. Group average is a compromise between the former two. There is also a method using *Ward's method* – the similarity of two

Clustering continued on next page...

clusters is based on the increase in SSE error; highly analogous to k-means. These methods require $O(n^2)$ space (proximity matrix) and is $O(n^3)$ time in many cases; the complexity can be reduced to $O(n^2 \log n)$ in some cases. Recognize that no objective function is directly minimized; measuring the quality is difficult.

NOTE In practice, do not trust training data in classifications. It is a common issue that the classifications are *wrong* in training. Also create mechanisms to check the training data as well. Thankfully, in general, academia is now beginning to self-evaluate. You could think of classification as a black box, but it is more and more common for people to ask why decisions are being made and how to understand the classification process (**interpretability**). It's predicted that unless DNNs can be explained well, it might be discarded. See `distil.pub/2017/feature-visualization/`.

NOTE There are no solutions to understand or explain clustering. However, people still continue using them and rely on the clustering results.

DBSCAN is a density-based algorithm and is recommended for use on the project or assignments. You do not need to specify the number of clusters; it finds it for you. It has one or two parameters to set. In DBSCAN, *density* refers to the number of points within a specified radius (Eps). A point is a *core point* if it has more than a specified number of points (MinPts) within Eps. These are points at the interior of a cluster. A *border point* has fewer than MinPts within Eps but is in the neighborhood of a core point. A *noise point* is any point that is not a core point nor a border point. See figure on slides. The algorithm also involves eliminating noise points and performing clustering on the remaining points. Empirically, MinPts is good to be set as 4 for better results. DBSCAN does not work well when there are varying densities, and thus for high-dimensional data.

Cluster Validity. For *supervised classification* we have a variety of measures to evaluate how good our model is (accuracy, precision, recall). For *cluster analysis*, the analogous question is how to evaluate the "goodness" of the resulting clusters. You need to determine the clustering tendency of a set of data (distinguishing whether non-random structure exists). One could also compare the results of a cluster analysis to externally known results (externally given class labels). You could also compare the results of two different sets of analyses and determine which is better (what is the "correct" number of clusters?). Numerical measures to judge cluster validity can be external or internal (see slides). Another method involves looking at similarity matrices and identifying if there are clear clusters or not. The validation of clustering structures is the most difficult and frustrating part of cluster analysis. Without a strong effort, it will remain a "black art".

Visual Analytics

Introduction to Visual Analytics

NOTE People are being replaced by machines. It's not far off to have self-driving cars in popular use. Automation has become a new part of our lives. All of us as computer scientists will be part of this. Computers can often do better than humans and can replace physicians, doctors, etc. The revolution is not because we have technology; it's because it's cheaper. It's about money. Computational power is becoming less and less a problem. Another example is "auto machine learning".

Is data mining or machine learning the answer for *everything*?

If the answer is yes, there are some consequences. When does machine learning fail? There are things that data cannot describe (weather, for example, is very difficult to predict – there may be some latent variables we have not

Introduction to Visual Analytics continued on next page...

defined). There can be a lack of data for things. There is a "human element" to things; people do not necessarily trust machines. The answer so far right now is no. Fully automated methods only work under certain preconditions: (1) data is clearly structured, (2) data semantics are well-defined, (3) the data is complete and correct (the latter is difficult to prove), and (4) the problem is well-defined. Human beings can solve problems that are not well-defined; they can solve problems they have not been trained for.

There is also a notion of **algorithm aversion**: people erroneously avoid algorithms after seeing them err. There is a people on this. The idea is simple: people will often place their bets on humans over machine predictors, even if the accuracy is in favor of the latter. This is incredibly important — if you do not create a system that shows a person is in charge, that system may just fail. PEOPLE PREFER TO BELIEVE IN PEOPLE. Visualizations are useful to produce results and give users confidence in a result (the illusion they are in control).

VISUALIZATION PUTS USERS IN THE LOOP. Tweaking parameters is not the solution. It's more than that. Selecting attributes and performing classifications, each producing results, give users more confidence. People do not understand parameters. So, explanations and interactivity are key. In other words, some visualization should be on top of a data mining model. The intersection of the two is **Visual Analytics**. It was coined in 2004 and came about as a reaction to September 11, 2001. A good resource is *Illuminating the Path*, a government agenda from the United Stated of America.

NOTE The project should have *preprocessing* done on the data. Were the histograms looked at? Were some variables considered irrelevant? Are there outliers? This is part of the project; it should be commented on during the presentation.

In 2001, the term **Visual Data Mining** was coined, aiming at the incorporation of *domain knowledge* and human perception into the data mining process. This is the projection of visualization on the mining process. The combination of visualization and mining can occur in three ways:

1. Assist mining algorithms before data is processed (*pre-processing task* – for example, a histogram),
2. Interpretation of the data mining results (*patterns* – even a confusion matrix is a kind of visualization), and
3. Supervise and steer the run of a data mining algorithm (*during the data mining process* – this is more or less visual analytics)

Recognize the last way can occur in a cycle. Through the aid of a visualization, multiple results can be produced and analyzed. How does the user identify what has changed with each result? Remember this – a user needs to know what they are doing. Visual analytics came about from this idea.

In 2003, the notion of **Interactive Machine Learning Models** was introduced. This is one of the predecessors of visual analytics *steered by a designer* (and not for a final user).

Realize that you often want to generate the best model possible to show to the user without any intervention possible (this is auto machine learning). Look into data science democratization. This is about having public access to the analysis of data (not just the data) – doing everything automatically.

In 2006, we have the idea of using visual analytics models to generate knowledge (**Knowledge Generation**). Some process is involved in order to generate knowledge iteratively via three loops: exploration, verification, and knowledge generation. First, a **finding** can be a pattern of an unusual behaviour of the system (exploration) that can then be used to change the visual mapping or parameter for model building. Second, analysts gain **insights** when they interpret findings. Insights may lead to new **hypotheses**.

In terms of taxonomy, machine learning has been integrated into VA applications from two transversal perspectives: the types of ML algorithms and interaction intent (modifying parameters, measures and algorithm).

Dimension Reduction or multidimensional projections maps points in a multidimensional space to points on a visual space which optimizes some distance metric. This does not look for clusters. If there *are* clusters, they may manifest. One method that is often used is t-SNE. It will show clusters if your data has clusters. If it does not, it will still show clusters. The problem with this method is that it has two stages – the first stage clusters, more or less, and the second stage spreads them. Why do people like this? They might not know what they're doing. But it always shows something — but these groups may not exist or may not be pure. Therefore, hypotheses that may arise from these projections will be inherently wrong.

These techniques try to approximate distance between elements. By reducing the number of points, there is a better approximation. Projclus leverages this to join together clusters each respectively projected. This does not mean there is any relationship between clusters.

Sensors and Biosensors Analysis.