Data Visualization (CSCI 627/490)

Data Manipulation & Isosurfacing

Dr. David Koop
Overview: Reducing Items & Attributes

- **Filter**
  - Items
  - Attributes

- **Aggregate**
  - Items
  - Attributes

[Munzner (ill. Maguire), 2014]
Tasks in Understanding High-Dim. Data

Task 1
- **What?**
  - In High-dimensional data
  - Out 2D data
- **Why?**
  - In HD data
  - Out 2D data

Task 2
- **What?**
  - In 2D data
  - Out Scatterplot Clusters & points
- **Why?**
  - Discover
  - Explore
  - Identify
- **How?**
  - Encode
  - Navigate
  - Select

Task 3
- **What?**
  - In Scatterplot Clusters & points
  - Out Labels for clusters
- **Why?**
  - Produce
  - Annotate

[Munzner (ill. Maguire), 2014]
Principle Component Analysis (PCA)

original data space

Gene 2

Gene 3

Gene 1

component space

PC 1

PC 2

PC 1

PC 2

M. Scholz, CC-BY-SA 2.0

D. Koop, CSCI 627/490, Fall 2020
Probing Projections

Abstract
—We introduce a set of integrated interaction techniques to interpret and interrogate dimensionality-reduced data. Projection techniques generally aim to make a high-dimensional information space visible in form of a planar layout. However, the meaning of the resulting data projections can be hard to grasp. It is seldom clear why elements are placed far apart or close together and the inevitable approximation errors of any projection technique are not exposed to the viewer. Previous research on dimensionality reduction focuses on the efficient generation of data projections, interactive customisation of the model, and comparison of different projection techniques. There has been only little research on how the visualization resulting from data projection is interacted with. We contribute the concept of probing as an integrated approach to interpreting the meaning and quality of visualizations and propose a set of interactive methods to examine dimensionality-reduced data as well as the projection itself. The methods let viewers see approximation errors, question the positioning of elements, compare them to each other, and visualize the influence of data dimensions on the projection space. We created a web-based system implementing these methods, and report on findings from an evaluation with data analysts using the prototype to examine multidimensional datasets.

Index Terms
—Information visualization, interactivity, dimensionality reduction, multidimensional scaling.

1INTRODUCTION
A primary goal of information visualization is to find patterns and relationships in multivariate datasets. Many visualization techniques have been developed towards this goal such as multiple coordinated views [2], parallel coordinates [14], scatterplot matrices [28], and dimensionality reductions such as multidimensional scaling (MDS) [5]. Dimensionality reductions are a particular class of techniques that synthesise high-dimensional data spaces onto projection spaces with much fewer dimensions, typically the two dimensions of the plane. While most visualization techniques juxtapose the different data dimensions as matrices or columns, dimensionality reductions integrate them into a planar canvas. The projection results in a so-called spatialisation (i.e., embedding) of data elements that approximately represents similarity as proximity and in turn dissimilarity as distance. Considering that the human perceptional system comprises a well-developed capacity for spatial reasoning, the assumption is that spatialisation would be a more natural way [31] to analyse high-dimensional datasets since groupings, separations, and other patterns among data elements become immediately discernible.

However, there are two major caveats linked with dimensionality reduction: first, it can be challenging to interpret the positions of projected elements, and second, the errors that occur with any projection technique.

Probing Projections

[J. Stahnke et al., 2015]
Focus+Context Overview

- **Embed**
  - Elide Data
  - Superimpose Layer
  - Distort Geometry

- **Reduce**
  - Filter
  - Aggregate
  - Embed

[Munzner (ill. Maguire), 2014]
Elision & Degree of Interest Function

- DOI = I(x) - D(x,y)
  - I: interest function
  - D: distance (semantic or spatial)
  - x: location of item
  - y: current focus point
  - Interactive: y changes

[Heer and Card, 2004]
Superimposition with Interactive Lenses

(a) Alteration

(b) Suppression

[ChronoLenses and Sampling Lens in Tominski et al., 2014]
It can be difficult to observe micro and macro features simultaneously with complex graphs. If you zoom in for detail, the graph is too big to view in its entirety. If you zoom out to see the overall structure, small details are lost.

Focus + context techniques allow interactive exploration of an area.
Distortion: Stretch and Squish Navigation

Figure 3. LiveRAC shows a full day of system management time-series data using a reorderable matrix of area-aware charts. Over 4000 devices are shown in rows, with 11 columns representing groups of monitored parameters. (a): The user has sorted by the maximum value in the CPU column. The first several dozen rows have been stretched to show sparklines for the devices, with the top 13 enlarged enough to display text labels. The time period of business hours has been selected, showing the increase in the In pkts parameter for many devices. (b): The top three rows have been further enlarged to show fully detailed charts in the CPU column and partially detailed ones in Swap and two other columns. The time marker (vertical black line on each chart) indicates the start of anomalous activity in several of spire’s parameters. Below the labeled rows, we see many blocks at the lowest semantic zoom level, and further below we see a compressed region of highly saturated blocks that aggregate information from many charts.

Principle: multiple views are most effective when coordinated through explicit linking. The principle of linked views [15] is that explicit coordination between views enhances their value. In LiveRAC, as the user moves the cursor within a chart, the same point in time is marked in all charts with a vertical line. Similarly, selecting a time segment in one chart shows a mark in all of them. This technique allows direct comparison between parameter values at the same time on different charts. In addition, people can easily correlate times between large charts with detailed axis labels, and smaller, more concise charts.

Assertion: showing several levels of detail simultaneously provides useful high information density in context. Several technique choices are based on this assertion. First, LiveRAC uses stretch and squish navigation, where expanding one or many regions compresses the rest of the view [11, 17]. The accompanying video shows the look and feel of this navigation technique. The stretching and squish- ing operates on rectangular regions, so expanding a single chart also magnifies the entire row for the device it represents, and the entire column for the parameters that it shows. The edges of the display are fixed so that all cells remain within the visible area, as opposed to conventional zooming where some regions are pushed off-screen. There are rapid navigation shortcuts to zoom a single cell, a column, an aggregated group of devices, the results of a search, or to zoom out to an overview. Users can also directly drag grid lines or resize freely drawn on-screen rectangles. Navigation shortcuts can also be created for any arbitrary grouping, whose cells do not need to be contiguous. This interaction mechanism affords multiple focus regions, supporting multiple levels of detail.

Second, charts in LiveRAC dynamically adapt to show visual representations adapted in each cell to the available screen space. This technique, called semantic zooming [13], allows a hierarchy of representations for a group of device-parameter time-series. In Figure 3, the largest charts have multiple overlaid curves and detailed axis and legend labels. Smaller charts show fewer curves and less labeling, and at smaller sizes only one curve is shown as a sparkline [24]. On each curve, the maximum value over the displayed time period is indicated with a red dot, the minimum with a blue dot, and the current value with a green one. All representation levels color code the background rectangle according to dynamically changeable thresholds of the minimum, maximum, or average values of the parameters within the current time window. The smallest view is a simple block, where this color coding is the only information shown.

Third, aggregation techniques achieve visual scalability by ensuring dense regions show meaningful visual representations. Given our target scale of dozens of parameters and thousands of devices, the size of the matrix could easily surpass 100,000 cells. Stretch and squish navigation allows users to quickly create a mosaic with cells of many different sizes.
H3 Layout

Large Graph Exploration with H3Viewer and Site Manager

(Demo)
H3 Layout

Large Graph Exploration with H3Viewer and Site Manager

(Demo)

[T. Munzner, 1998]
Focus+Context in Network Exploration

(a) Bring (step 1) – Selecting a node fades out (b) Bring (step 2) – Neighbor nodes are pulled (c) Go – After selecting a neighbor (the green node in Fig. 4(b)), a short animation brings the focus towards a new neighborhood.

[Lambert et al., 2010]
Focus+Context in Network Exploration

(a) Moderately large graph drawn with straight line edges. The graph nodes correspond to the USA major cities; edges show migration flows. The graph contains 1715 nodes and 9778 edges. Nodes are laid out according to geographical positions of cities, producing a drawing with poor readability, where edges mix in a totally unordered way and where some nodes are close to unnoticeable.

(b) The same graph as in Fig. 1(a) now drawn using edge bundling with edges rendered as Bézier curves.

Figure 1: Illustration of edge bundling.

(a) The fish-eye distorts a small region of the graph for local inspection.

(b) The magnifying lens shows a zoom on a local region.

Figure 2: Fisheye and magnifying lens.

Magnifying Lens and Fish-eye–The magnifying lens [3] and geometrical fish-eye [7] were also added to the system as basic interactors. They allow to get local details on an area of the graph without having to zoom in (see Fig. 2(a) and Fig. 2(b)). These techniques allow to get a rough estimate on the degree or number of edges that have been bundled together, and an idea on the spatial organization of neighborhoods.

Neighborhood highlighting–After edges have been bundled, the graph gains in overall readability at the loss of more local information. For instance, connections between any two particular nodes cannot be easily recovered and isolated out of a bundle. When designing the system and deciding on the interactions to implement and combine, we focused on the recovery of these local information. By hovering the mouse over any node in the graph drawing, the user can highlight its neighborhood. This is accomplished by showing a translucent circle over the immediate where a node sits while clearly displaying the neighborhood of the node (top of Fig. 3(a)). The circle fades off nodes not belonging to the selected neighborhood, temporarily providing a clear view of it. The size of the translucent circle is fitted as to enclose all immediate neighbors of the node in the graph. Using the mouse wheel, the user can select neighbors sitting at a bounded distance from the node. The size of the translucent circle adjusts accordingly (bottom of Fig. 3(b)).

Bring & Go–Now, neighborhoods in the graph don't always sit close. As a consequence, the translucent circle highlighting neighbors of a node can potentially be quite large. That is, the distance between nodes in the graph does not always match their Euclidean distance in the drawing –

Focus+Context in Network Exploration

[Lambert et al., 2010]
(a) Neighborhood highlighting – selecting a node brings up its neighbors, fading away all other graph elements.

(b) Using the mouse wheel, the neighborhood is extended to nodes sitting further away.

Figure 3: Illustration of the Neighborhood highlighting in interaction

This indeed is the challenge posed to all layout algorithms. The Bring & Go technique introduced by Tominski et al. [18] solves this paradox. The Bring operation pulls neighbors of a node to near proximity, temporarily resolving a situation where the layout algorithm had failed. Fig. 4(a) and Fig. 4(b) illustrates this situation – the passage from step 1 to step 2 being smoothly animated. Once the neighbors have been repositioned close to the node, the Go operation lets the user decide of a new direction to move to by selecting a neighbor. After clicking a neighbor node, the visualization is panned until re-centered around the target neighbor. The transition is performed by smoothly animating the pan (see Fig. 3). A recent user-study of this interaction technique has been made by Moscovich et al. [15]. When bringing neighbors close to the selected node, the edges abandon their curve shapes and are morphed to straight lines. This is done by modifying the control points coordinates of each curve so that they are all aligned.

Our system thus comprises a comprehensive palette of interactions focusing on adjacency or accessibility tasks (we borrow this terminology from Lee et al.'s [14] task taxonomy, itself referring to the work of Amar et al. [1]). That is, tasks such as exploring neighbor nodes, or counting them, finding how many nodes can be accessed from any given one, etc., can be easily done through direct manipulation of the graph using zoom, pan, neighborhood highlight or Bring & Go, for instance. All these interactions techniques have been implemented as interactor plugins for the Tulip graph visualization software [2] and are available through its plugin server.

4 Maintaining fluid interaction

The challenge we were faced with is that curves generation have a relatively high computational cost when it comes to interacting with bundles. Indeed, although the curves can be drawn in reasonable time for static drawings using standard rendering techniques, the problem becomes tedious when one wants to interact on bundles using any of the techniques described in the previous section. The curves' shapes must be continually transformed as the user moves the mouse and pilots interaction (geometrical fisheye or Bring & Go for instance).

Moreover, we did not want fluidity to impact on the quality of the curves and impose an upper bound on the number of control points used to compute the edge routes. Instead, we aimed at producing a system capable of dealing with an arbitrary number of control points. As a consequence, the computation of the points interpolating the curve itself puts a real burden on the system and calls for an extremely efficient approach. The solution we designed avoids performing computations on the CPU as far as possible, relying on the GPU for almost all curve related computations. The only computations that are potentially performed on the CPU are the original graph layout and the bundling part.

4.1 Introduction to spline rendering

Now, there are two major issues when rendering a parametric spline. Control points define the curve analytically described as a polynomial (see Eq. (1 for Bézier curves). Second, once the polynomial has been determined, it must be evaluated as many times as required in order to interpolate the curve itself. As a consequence, when interacting with the graph asking for local deformation of edges, bringing neighbors closer or following an edge, the curves must be re-computed on the fly.

A classical approach when rendering a curve is to compute the interpolation points on the CPU, then call appropriate graphics primitives and let the GPU render the curve.
Project Design

• Feedback:
  - Data Manipulation?
  - Questions lead, not technique!
  - Be creative! (interaction too) https://xeno.graphics

• Work on turning your visualization ideas into designs

• Turn in:
  - Two Design Sketches (like sheets 2-4 from 5 Sheet Design)
  - One Bad Design Sketch (like sheets 2-4: here, justify why bad)
  - Progress on Implementation

• Due Friday
Assignment 5

- Map of Citi Bike trips
  - Multiple Views
  - Linked Highlighting
  - Filtering
  - Aggregation

- Due Monday, Nov. 23
Data Wrangling

- Problem 1: Visualizations need data
  - Solution: The Web!
- Problem 2: Data has extra information I don't need
  - Solution: Filter it
- Problem 3: Data is dirty
  - Solution: Clean it up
- Problem 4: Data isn't in the same place
  - Solution: Combine data from different sources
- Problem 5: Data isn't structured correctly
  - Solution: Reorder, map, and nest it
Hosting data

- github.com
- gist.github.com
- figshare.com
- myjson.com
- Other services
Cross-origin resource sharing (CORS)

• Restricts where data can be loaded from
• If developing locally, can
  - Run a web server locally (python -m http.server or npm's http-server)
  - Put the data on a website (like github), make sure to use raw URLs
• If loading JavaScript, this sometimes requires more help
  - https://gitcdn.xyz
Filtering Data

- Often useful to filter data before loading into D3
Why JavaScript?

- Python and R have great support for this sort of processing
- Data comes from the Web, want to put visualizations on the Web
- Sometimes unnecessary to download, process, and upload!
- More tools are helping JavaScript become a better language
JavaScript Data Wrangling Resources

• Latest version: https://observablehq.com/@berkeleyvis/learn-js-data
• My old version: https://observablehq.com/@dakoop/learn-js-data
• Based on http://learnjsdata.com/
• Good coverage of data wrangling using JavaScript
Comma Separated Values (CSV)

• File structure:

cities.csv:

city, state, population, land area
seattle, WA, 652405, 83.9
new york, NY, 8405837, 302.6
boston, MA, 645966, 48.3
kansas city, MO, 467007, 315.0

• Loading using D3:

d3.csv("/data/cities.csv").then(function(data) {
    console.log(data[0]);
});

• Result:

=> {city: "seattle", state: "WA", population: 652405, land area: 83.9}

• Values are strings! Convert to numbers via the unary + operator:

- d.population => "652405"
- +d.population => 652405
Tab Separated Values (TSV)

• File structure:

```
animals.tsv:

name   type     avg_weight
---    ------    --------
tiger  mammal   260
hippo  mammal   3400
komodo dragon reptile  150
```

• Loading using D3:

```javascript
d3.tsv("/data/animals.tsv").then(function(data) {
    console.log(data[0]);
});
```

• Result:

`=> {name: "tiger", type: "mammal", avg_weight: "260"}`

• Can also have other delimiters (e.g. '|', ';')
JavaScript Object Notation (JSON)

• File Structure:

```javascript
employees.json:
[
  {
    "name": "Andy Hunt",
    "title": "Big Boss",
    "age": 68,
    "bonus": true
  },
  {
    "name": "Charles Mack",
    "title": "Jr Dev",
    "age": 24,
    "bonus": false
  }
]
```

• Loading using D3:

```javascript
d3.json("/data/employees.json".then(function(data) {
  console.log(data[0]);
});
```

• Result:

```javascript
=> {name: "Andy Hunt", title: "Big Boss", age: 68, bonus: true}
```
Loading Multiple Files

- Use Promise.all to load multiple files and then process them all

```javascript
Promise.all([d3.csv("/data/cities.csv"),
              d3.tsv("/data/animals.tsv")]).then(analyze);

function analyze(data) {
  cities = data[0]; animals = data[1];

  console.log(cities[0]);
  console.log(animals[0]);
}
```

```javascript
=> {city: "seattle", state: "WA", population: "652405", land area: "83.9"}
{name: "tiger", type: "mammal", avg_weight: "260"}
```
Combining Data

• Suppose given products and brands
• Brands have an id and products have a `brand_id` that matches a brand
• Want to join these two datasets together
  - `Product.brand_id` => `Brand.id`
• Use a nested `forEach/filter`
• Use a native join command
Summarizing Data

- d3 has min, max, and extent functions of the form
  - 1st argument: dataset
  - 2nd argument: accessor function

Example:
```
var landExtent = d3.extent(data, function(d) {
  return d.land_area;
});
console.log(landExtent);
=> [48.3, 315]
```

- Summary statistics, e.g. mean, median, deviation → same format

Median Example:
```
var landMed = d3.median(data, function(d) {
  return d.land_area;
});
console.log(landMed);
=> 193.25
```
Grouping Data

• Take a flat structure and turn it into a (potentially nested) map
• Similar to a groupby in databases
• Data

```javascript
var expenses = [{
    "name": "jim", "amount": 34, "date": "11/12/2015"},
    {"name": "carl", "amount": 120.11, "date": "11/12/2015"},
    {"name": "jim", "amount": 45, "date": "12/01/2015"},
    {"name": "stacy", "amount": 12.00, "date": "01/04/2016"},
    {"name": "stacy", "amount": 34.10, "date": "01/04/2016"},
    {"name": "stacy", "amount": 44.80, "date": "01/05/2016"}
];
```

• Grouping:

```
expensesByName = d3.group(expenses, d => d.name)
```

• Results:

```
Map(3) { "jim" => Array(2) [Object, Object],
    "carl" => Array(1) [Object],
    "stacy" => Array(3) [Object, Object, Object] }
```
Rollup Data

• Data

```javascript
var expenses = [{"name":"jim","amount":34,"date":"11/12/2015"},
    {"name":"carl","amount":120.11,"date":"11/12/2015"},
    {"name":"jim","amount":45,"date":"12/01/2015"},
    {"name":"stacy","amount":12.00,"date":"01/04/2016"},
    {"name":"stacy","amount":34.10,"date":"01/04/2016"},
    {"name":"stacy","amount":44.80,"date":"01/05/2016"}
];
```

• Using d3.rollup:

```javascript
expensesAvgAmount = d3.rollup(    expenses,
    v => d3.mean(v, d => d.amount), // aggregate by the mean of amount    d => d.name // group by name
)
```

• Result:

```javascript
Map(3) {
    "jim" => 39.5
    "carl" => 120.11
    "stacy" => 30.3
}
```

[http://learnjsdata.com](http://learnjsdata.com)
groups and rollups

- Both group and rollup return Map objects
- groups and rollups are the same functions but return nested arrays
arquero

- New library for query processing and transformation of array-backed data tables:
- https://observablehq.com/@uwdata/arquero?collection=@uwdata/arquero
Scivis and Infovis

- Two subfields of visualization
- **Scivis** deals with data where the spatial position is given with data
  - Usually continuous data
  - Often displaying physical phenomena
  - Techniques like isosurfacing, volume rendering, vector field vis
- In **Infovis**, the data has no set spatial representation, designer chooses how to visually represent data
SciVis

[Google Image Search for “scientific visualization”, 2017]
InfoVis

[Google Image Search for “information visualization”, 2017]
- Values come from a **continuous** domain, infinitely many values
- **Sampled** at certain positions to approximate the entire domain
- Positions are often aligned in **grids**
- Often measurements of natural or simulated phenomena
- Examples: temperature, wind speed, tissue density, pressure, speed, electrical conductance
Fields in Visualization

Scalar Fields
(Order-0 Tensor Fields)

Vector Fields
(Order-1 Tensor Fields)

Tensor Fields
(Order-2+)

Each point in space has an associated...

Scalar

$\mathbf{s}$

Vector

$\begin{bmatrix}
  v_0 \\
  v_1 \\
  v_2
\end{bmatrix}$

Tensor

$\begin{bmatrix}
  \sigma_{00} & \sigma_{01} & \sigma_{02} \\
  \sigma_{10} & \sigma_{11} & \sigma_{12} \\
  \sigma_{20} & \sigma_{21} & \sigma_{22}
\end{bmatrix}$