NAVSEA WARFARE CENTERS
DAHLGREN

Some Statistical Applications in Cyber Security

David J. Marchette

October 17, 2017

Acknowledgment: This work funded in part by the NSWC

In-House Laboratory Independent Research (ILIR) program.

NSWCDD-PN-17-00417 Distribution A: Approved for Public Release

Topics

Topics

Take-Away Points

- Mathematics and statistics provide many tools for cyber security.
- Simple can be powerful.
    - Complicated models or algorithms are not always necessary.
    - Sometimes they are.
- “Complicated” things become “simple” with familiarity.

Outline

- Cyber security data.
  - Network data.
  - Malware.
- Probability density estimation.
  - Kernel estimators.
  - Streaming data.
- Machine learning.

Topics

Network Flows

As former senator Ted Stevens (R-Alaska) famously said: "... the Internet is not something that you just dump something on. It's not a big truck. It's a series of tubes."

- Network flows can be thought of as one-way "tubes".
- When you connect to a web site, you have:
  - a flow from your machine to the server (with your requests);
  - a flow from the server to your machine (with the responses).
- These "flows" consist of packets – the data is broken up into relatively small chunks, and these are sent along and reconstructed at your machine.

Network Flows

- Each "flow" pair have "ports" associated with your machine and the server – 16-bit integers (in the case of web traffic the server's port is likely 80 or 443).
- These ports (and some other information transfered on the packets) allow the respective machines to put together the flows properly and give them to the correct application.
- Even if two people use the same computer at the same time, they have their own port-pairing, so their flows don't get mixed up and sent to the wrong person.

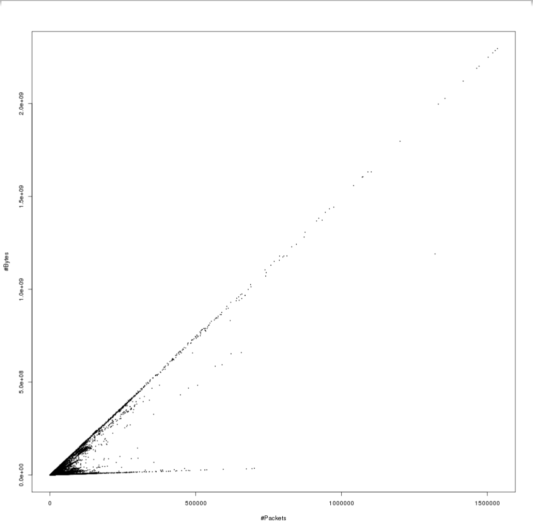
Network Flow Data

- One place to obtain some network flow data (as well as some other interesting cyber data, is `http://csr.lanl.gov/data/cyber1/`.
- The flows data consist of the following:
  - Time,
  - Source and destination IPs,
  - Source and destination ports,
  - protocol,
  - #packets,
  - #bytes.

Visualization

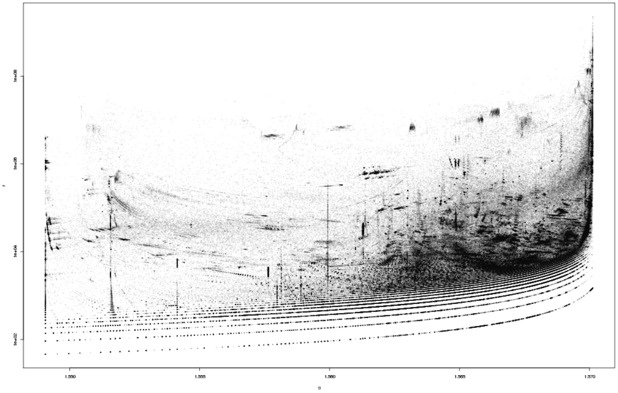
- It is important to understand the data, and to do this we need visualization tools.
- Simple is often better: complicated plots can be confusing and hard to interpret, unless great care is taken to ensure the information is correctly represented.
- The simplest plot is a two dimensional scatter plot.
- Let's look at the network flows by plotting number of packets against number of bytes.
- We might think the number of bytes should be a constant multiple of the number of packets and hence lie on $y = bx$; why not fill each byte as much as possible?

Network Flows



Network Flows: Polar Coordinates

$$(x, y) \to (\theta, r)$$



- This uses alpha blending to handle overplotting.
- We see the discretization of the data in the bottom curves.
- Vertical lines correspond to the lines we saw before.
- Why are there so many vertical lines?
- Perhaps some of the clumping we see is "interesting" and one might consider investigating these.

Observations

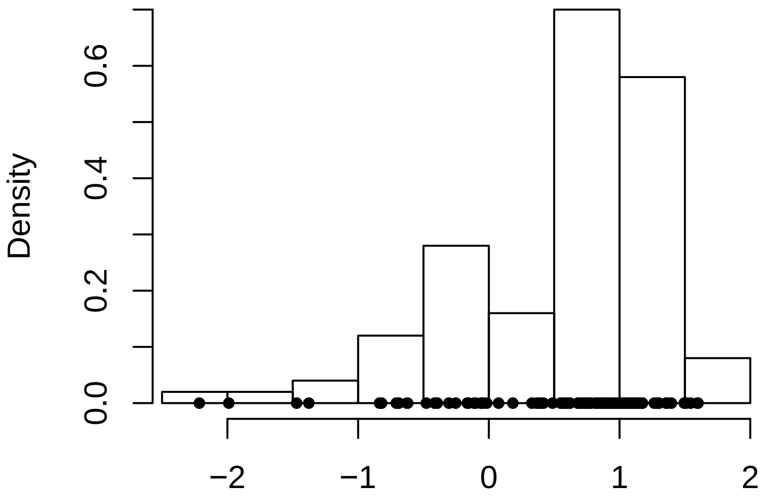
- Interactive flows (terminals such as ssh) send packets with each character typed and so will tend to have smaller sized packets.
- Data transfers (such as ftp or videos) tend to fill packets as full as possible.
- The different lines (angles – vertical lines in the polar coordinates plot) indicate different applications that make different trade-offs.
- If you expect a particular application to be served from a given port, you might use this information to see if the pattern you observe is consistent with the pattern you expect from that application.
- Alternatively, if you observe activity on a port, this analysis might suggest what type of activity is taking place.

Topics

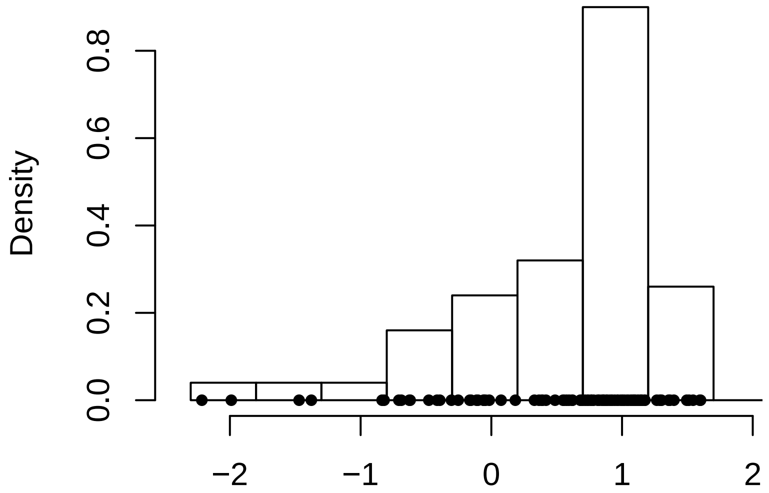
Distributions

- Consider the question of how the data you observe is distributed.
- Knowing this allows us to make inference about what is going on.
- We all know about histograms, and this is the starting point for understanding distributions.
- Given data, we lay out a set of bins, and count the proportion of observations in each bin.

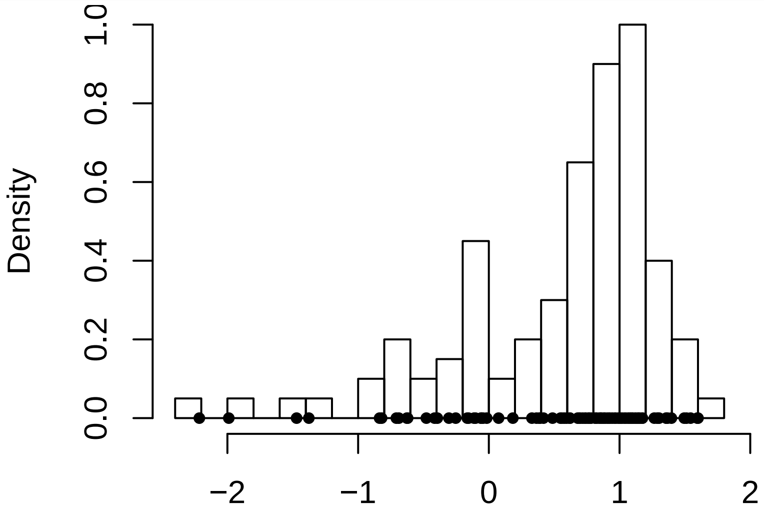
The Histogram



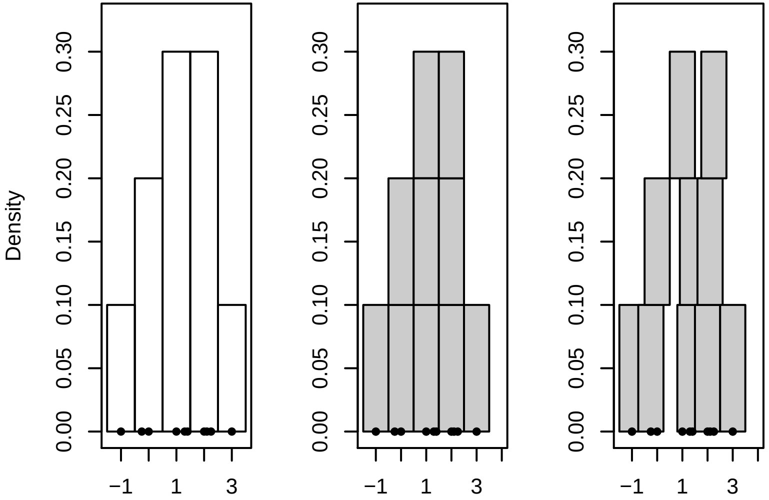
The Histogram: Same Data, Different Bins



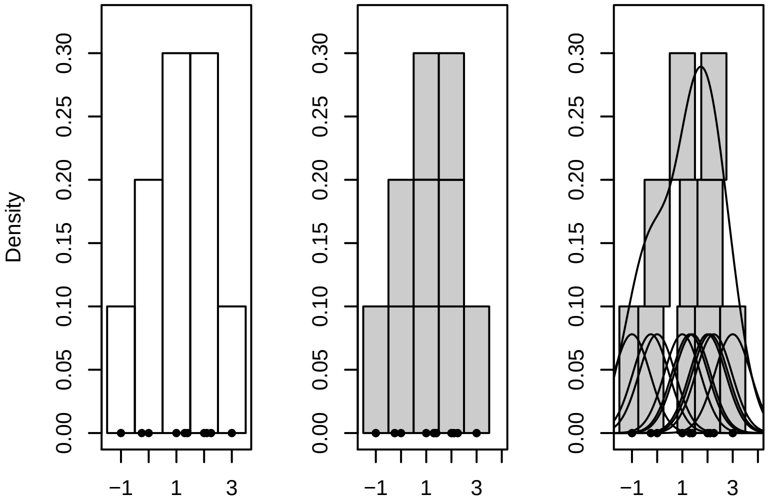
The Histogram: More Bins



Data Centered Bins



The Histogram – The Kernel Estimator

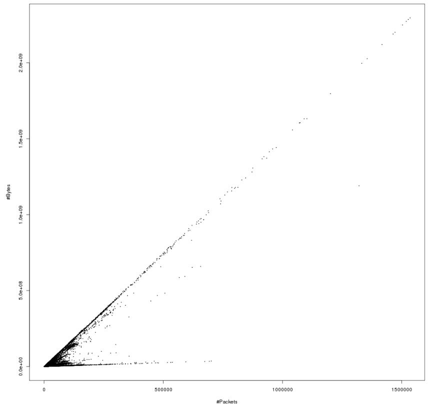


The Kernel Estimator

$$f(x) = \frac{1}{n}\sum_{i=1}^{n} K_h(x-x_i) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{h}\phi\left(\frac{x-x_i}{h}\right) = \frac{1}{nh}\sum_{i=1}^{n}\phi\left(\frac{x-x_i}{h}\right).$$

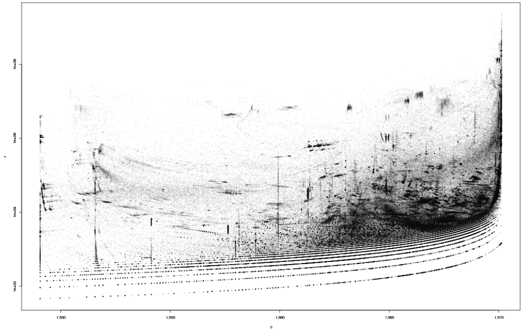
- Easily extended to multivariate versions.
- Note that this is an “average”.
- Also note that this is a mixture of (in this case) normals.

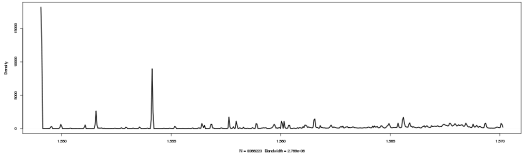
Network Flows



http://csr.lanl.gov/data/cyber1/

Network Flows





Big Data

- In computer security, we often have huge amounts of data, and it is often coming at us very fast.
  - Network packets at a firewall can be many thousands a second.
  - Emails at a mail server for a large company can be many thousands a day.
  - Virus scanners may scan hundreds of files on each computer every day.
  - Log files can have hundreds of entries every day – on a large network this can be many thousands of entries.
- Sometimes we can't (or choose not to, for privacy or security reasons) store all the data.
- Even if we could, we usually don't want to have to read through all the historical data every time we want to update our algorithms.

Streaming Data

- Streaming data refers to data that is observed in time.
- Generally we can only store a small window of the data for any processing we wish to do.
- Whether the data is archived somewhere or not, the idea is that we can't afford to look at more than a small window of data at any time.
- We need methods to design our algorithms ("learn") without looking at all the archived data – we need to update the algorithm's parameters as we observe new data.
- Note that the kernel estimator uses all the data: how can we build a density estimate like this without using all the data each time we want to evaluate it?

Streaming Data

Averages can be computed in a streaming fashion:

$$\widehat{X}_n = \frac{n-1}{n}\widehat{X}_{n-1} + \frac{1}{n}X_n.$$

We can implement an exponential window:

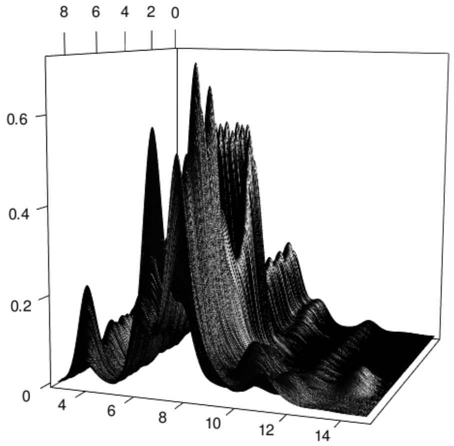
$$\widetilde{X}_n = \frac{N-1}{N}\widetilde{X}_{n-1} + \frac{1}{N}X_n = \theta\widetilde{X}_{n-1} + (1-\theta)X_n,$$

and apply this idea to the kernel estimator:

$$\widetilde{f}_n(x) = \theta\widetilde{f}_{n-1}(x) + (1-\theta)\phi\left(\frac{x-X_n}{h}\right).$$

θ controls how much of the past we "remember". Note that we have to set a grid of x points at which we want to compute $\widetilde{f}$.

Streaming Network Flows: log(#bytes) in a Flow



Mixture Models

- A mixture of normals is defined as:

$$f_m(x) = \sum_{i=1}^{m} \pi_i \phi(x; \mu, \Sigma).$$

- The $\pi_i \in [0, 1]$ are the mixing proportions and sum to 1.
- Note that all the parameters can be estimated via "averages", and hence can be done in a streaming fashion. See Priebe, JASA, 1994.

Topics

Machine Learning: Classification

- Given $\{(x_i, y_i)\}_{i=1,\dots,n} \subset X \times Y$ with x_i corresponding to observations (flows, programs, email, system calls, log files: "features"), and y_i corresponding to class labels (e.g. "malware", "benign").
- A classifier is a mapping $g : X \to Y$.
- Machine learning (pattern recognition, classification) is designing a function g from "training data" $\{(x_i, y_i)\}_{i=1,\dots,n}$ for which "truth" is known.

We are given training data $\{(x_i, y_i)\}_{i=1,\dots,n} \subset X \times Y$, and will be presented with a new $x \in X$ for which the label is unknown. We wish to infer the y associated with x.

Nearest Neighbors

We are given training data $\{(x_i, y_i)\}_{i=1,\dots,n} \subset X \times Y$, and a new $x \in X$ for which the label is unknown.

1. Find the closest x_i to x:

   $$\widehat{y} = y_{\arg\min d(x,x_i)}.$$

   That is, we give x_i the class label of the closest observation in our training data.
2. We must select an appropriate distance (dissimilarity) d.
3. Alternative: We can compute the k closest, and vote: take the majority class.

Kaggle Malware

- $10,868$ examples of malware grouped into 9 malware "families".*
- Each file has been byte-dumped and tabulated:
  - We are using the frequency of times each value $0, \dots, 255$ occurs in the file.
  - This seems really dumb (computer scientists laugh when I tell this story).
- We'll look at the nearest neighbor classifier on these data. 100 observations of each family are used for training (21 observations from the family containing only 42 observations). Test on the remaining.
- Remember: sometimes simple is good.

*https://www.kaggle.com/c/malware-classification

Kaggle Malware: 1NN Performance

True Class								
1291	116		6	2	53	12	75	128
5	1525		1		4	1	6	5
	33	2807	3		1	2	2	
1	63	29	352		12	2	7	10
27	89			19	13	8	23	21
55	166	2	10		554	2	32	28
30	6	4	3		4	244	45	
28	243				3	15	920	12
4	137				7	12	18	709

Error: 16.2%. That is, 84% of the observations are correctly classified.

Kaggle Malware: 1NN Performance – Why???

- Text analogy: byte-count histogram is analogous to the word-count histogram used in text analysis.
    - Maybe this is more like a morpheme-count histogram.
- Intuitively, a “family” shares a core of code (they are modifications of the “mother” malware).
- The bytes correspond to machine instructions – or at least they would if we were counting words instead of bytes.

Detecting Malicious Programs

- The Kaggle data is interesting, but for most of us the bottom line is more important: is this file the nice man in Nigeria sent me a virus?
- To investigate, I'll use the same basic approach as on the Kaggle data, but using benign/malicious data collected for Windows.
- In addition to the nearest neighbor classifier, I'll use a slightly more complicated algorithm: random forests.

Random Forests

We are given training data $\{(x_i, y_i)\}_{i=1,\dots,n} \subset X \times Y$, and a new $x \in X$ for which the label is unknown.

- The random forest is an ensemble of decision trees:
    1. Sample (with replacement) from the training data. Sample a subset of the variables.
    2. Build a decision tree using the two samples – don't bother with any optimization or pruning.
    3. Repeat.
- With a new observation, vote the trees.

Benign vs Malicious

- 7738 observations of windows binaries: 2054 benign, 5684 malicious.
- Random forest performance: 0.65% error.
    - 1.6% of benign misclassified.
    - 0.3% of malicious misclassified.
- Nearest neighbor classifier is a little worse: overall error of 1.1%.

Know Your Data

- The results demonstrate that there is something going on with this byte-count approach.
- Logically, the performance seems too good to be true, and yet it does seem to work.
- The data are high dimensional (256), so maybe there is a "curse of dimensionality" thing going on here.
- Perhaps we are finding OS-specific things:
  - The data collected for the benign files may be a different version of the operating system than the malicious.
  - We don't have version information about the data (beyond these are Windows files).
- Worrisome fact: there are several different sets of benign (or malicious) data. A classifier can be built to tell which set – which of the benign collections a file belongs to.

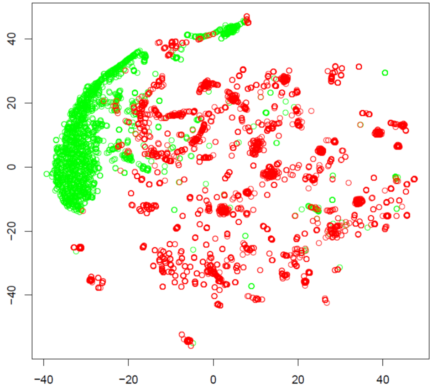
Too Good to be True? The Tank Parable

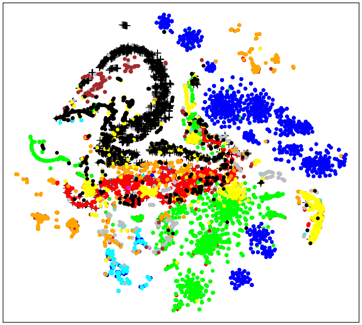
- A (possibly apocryphal) story tells of an attempt to build an algorithm to determine whether an image contained a (camouflaged) tank, or not.
- Data was collected, a neural network was built, and it was nearly perfect! It got excellent results on a set of witheld data!
- Unfortunately, when it was tested on an entirely new set of data, it was no better than chance.
- How can this be?

Too Good to be True? The Tank Parable

- It turns out, it costs a lot of money to go out and collect pictures of tanks.
- The easiest thing to do is to go somewhere (like an army base) that has a lot of tanks, and take as many pictures as you can of the tanks.
- Non-tank pictures are easy – you can take them at your leisure.
- Unfortunately, it was a cloudy day when the "tank" pictures were taken, and a sunny day when the "non-tank" pictures were taken.
- The algorithm was a cloudy-day detector.
- **Know your data!**

Know Your Data?

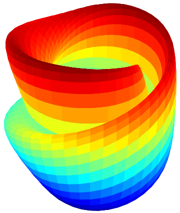




Maaten & Hinton (2008). Visualizing data using t-SNE. Journal of Machine Learning Research, 9, 2579-2605.

Manifold Learning

- Hypothesis: high dimensional data "lives" on a lower dimensional structure.
- Manifold learning is a set of techniques to infer this structure, or to embed the data from the high dimensional space into a lower dimensional space that respects the local structure.



Multidimensional Scaling

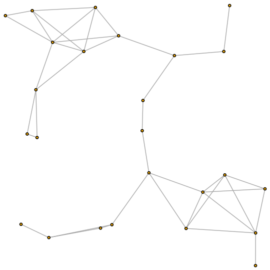
- Problem: Given a distance matrix (or dissimilarity matrix) D, find a set of points $X \in \mathbb{R}^d$ whose distance $d(X)$ best approximates D.
- This is the problem solved by multidimensional scaling (MDS).
- Different definitions of “best approximates” lead to different algorithms.
- Classical MDS utilizes the eigenvector decomposition of (a modified version of) the distance matrix.
- Some manifold learning algorithms compute a local distance and use MDS, others computer eigenvectors of related matrices. These are the algorithms I use most often.

Basic Graph Theory

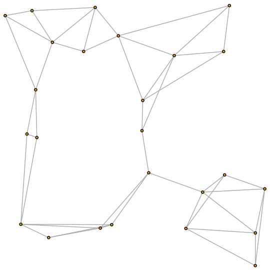
- A graph is a set V of vertices, and E of pairs of vertices (edges).
- The edges can be directed or undirected, and can have weights.
- In this talk they will be undirected.
- The (graph) distance between two vertices is the length of the shortest path between them in the graph.
- The adjacency matrix of a graph on n vertices is the $n \times n$ binary matrix with a 1 in those positions corresponding to the edges of the graph.
- The spectrum of a graph is the eigen decomposition of the adjacency matrix A, or more generally, of some function $f(A)$.

Graph Examples

ϵ-ball graph with $\epsilon = 0.25$.



3-nearest neighbor graph.



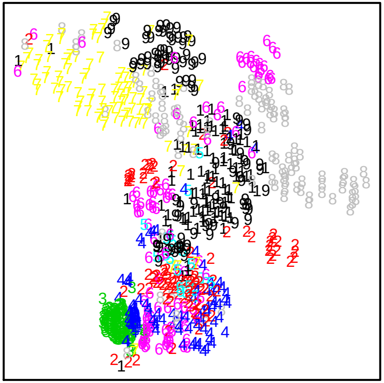
Basic Steps of Manifold Learning

- Given data $\{x_1, \ldots, x_n\} \in \mathbb{R}^p$:
  1. Construct a graph whose vertices are the x_i with edges between "near" points.
     - k-nearest neighbor graph.
     - ϵ-ball graph.
     - Variations.
  2. Compute the eigenvectors of:
     - The adjacency matrix.
     - The Laplacian of the adjacency matrix.
     - Scaled or modified versions of the above.
  3. Set Z to the matrix with columns corresponding to the main eigenvectors. That is, the rows $\{z_1, \ldots, z_n\}$ are the "embedded" data.
- Perform inference on Z.

Manifold Learning

Compute ϵ-ball graph on the Kaggle training data.

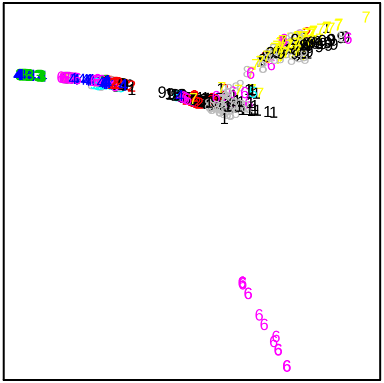
- **Layout the graph.**
- Embed using scaled Laplacian.
- Embed using adjacency matrix.
- Embed using MDS on graph distance.



Manifold Learning

Compute ϵ-ball graph on the Kaggle training data.

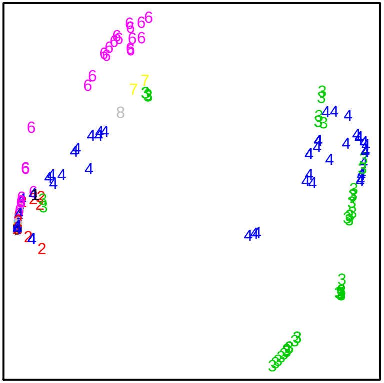
- Layout the graph.
- **Embed using scaled Laplacian.**
- Embed using adjacency matrix.
- Embed using MDS on graph distance.



Manifold Learning

Compute ϵ-ball graph on the Kaggle training data.

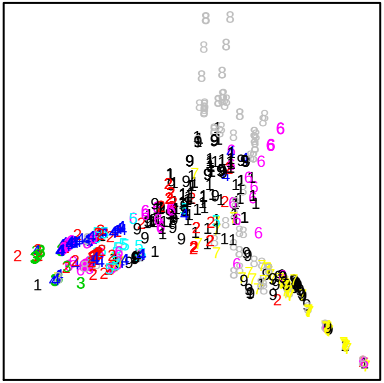
- Layout the graph.
- Embed using scaled Laplacian.
- **Embed using adjacency matrix.**
- Embed using MDS on graph distance.



Manifold Learning

Compute ϵ-ball graph on the Kaggle training data.

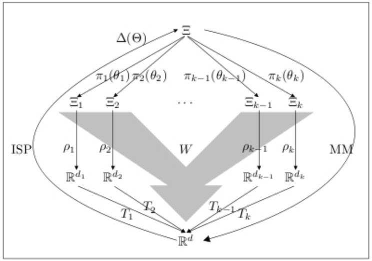
- Layout the graph.
- Embed using scaled Laplacian.
- Embed using adjacency matrix.
- **Embed using MDS on graph distance.**



Manifold Learning Discussion

- Different embedding methods extract different information about the data.
- These two dimensional plots are misleading in that there is no reason to assume the intrinsic dimensionality is 2.
- Some care must be taken to ensure that the embedding method can be applied to new data.

Joint Embedding



$$D_W = \begin{pmatrix} D_1 & W \\ W & D_2 \end{pmatrix}$$

Jointly embed D_1 and D_2 using D_W, where $W = \lambda D_1 + (1-\lambda) D_2$.

Discussion

- Mathematics has many tools for the data analyst, in particular for the analysis of cyber data.
- These tools include:
  - Computational statistics.
  - Machine learning.
  - Graph theory.
  - Manifold learning.
  - Topological data analysis.
- New applications of “pure” mathematics to data analysis are developed every day, and these areas are all huge growth areas for applied mathematicians.

Some Resources for Data

- http://csr.lanl.gov/data/cyber1.
- https://www.kaggle.com/c/malware-classification.
- http://www.secrepo.com.
- http://www.usma.edu/crc/sitepages/datasets.aspx.
- http://networkdata.ics.uci.edu/data.php?id=110.
- https://stratosphereips.org/category/dataset.html.
- http://csmining.org/index.php/malicious-software-datasets-.html.
- http://vizsec.org/data.
- https://zeltser.com/malware-sample-sources.

Questions?

dmarchette@gmail.com
david.marchette@navy.mil