Manifold Learning to Detect Changes in Networks

Kenneth Heafield
Richard and Dena Krown SURF Fellow
Mentor: Steven Low
Problem

- Monitor systems and watch for changes
- Unsupervised
  - Computer must be able to learn patterns
  - Automatically determine if deviation is significant
- Fast
  - Test for anomalies as data comes in
  - Incorporate new data into model
- Non-linear
  - Algorithm needs to work in many environments
Applications to Networking

- Monitor network packets and streams
  - Collect header information, particularly port numbers
- Security
  - Detect worms by large, structural changes
  - Detect viruses by small numbers of deviations from fit
- Optimization
  - Automatically learn traffic patterns and react to them
  - Anticipate traffic
How to phrase the problem mathematically

Linear regression in multiple dimensions with Principal Component Analysis (PCA)

Extending PCA to estimate errors in principal components
  • How to use the errors

Kernel PCA adds non-linearity

Future
  • Implementation
Thinking Geometrically

- Each packet is a data point with coordinates equal to its information.
- Fit a manifold to find patterns:
  - Compare with previous fits by storing manifold parameters.
  - Structure of manifold can tell us about underlying processes.
- Distance from manifold indicates deviation.
Principal Component Analysis

● Choose directions of greatest variance
  ● These are the eigenvectors of the covariance matrix
  ● Called Principal Components

● Widespread use in science

● Linear
  ● Many non-linear extensions—we will focus on kernel PCA later
  ● Equivalent to least-squares

● Jolliffe 2002
Goal: Find errors in Principal Components.
- Assume uncorrelated, multivariate normal distribution
- Find out how much each component contributes to estimating each point
- Get error of estimate in terms of (unknown) errors in components.
- Use residual to approximate error
- Out pops a regression problem which we can solve
Finding the Nearest Point

- Principal Component Analysis defines a subspace
  - Example: Linear regression finds a one-dimensional subspace of the two-dimensional input
  - Components are orthonormal
- Project data point into subspace
  - Data point $X_i$
  - Components $C_k$
  - Nearest point $N_i = \sum_{k=1}^{m} (X_i \cdot C_k) C_k$
Error in Nearest Point

- $N_i$ is the closest point to data $X_i$
  - Residual is $X_i - N_i$
- What is the error in this estimate?
  - Predictor $N_i$ variance $\rho_i^2$
  - Component $C_k$ variance $\sigma_k^2$
  - Symmetric about component, spread evenly in the $p-1$ possible dimensions
- Propagate the error:
  \[
  \rho_i^2 = \frac{1}{p-1} \sum_{k=1}^{m} \sigma_k^2 (X_i \cdot X_i - 2X_i \cdot N_i + p(X_i \cdot C_k)^2)
  \]
Idea: Regression Problem

- Use squared residual length $\|X_i - N_i\|^2$
  - This should, on average, equal predictor variance $\rho_i^2$
- Goal: Find $\sigma_k$
  - This is a linear regression problem:
    $$\|X_i - N_i\|^2 \approx \frac{1}{p-1} \sum_{k=1}^{m} \sigma_k^2 (X_i \cdot X_i - 2X_i \cdot N_i + p (X_i \cdot C_k)^2)$$
  - Subject to constraints
    - To be a variance, $0 \leq \sigma_k^2 \leq 1$
What All That Math Just Meant

- We did linear regression in multiple dimensions
- Found the point closest to each data point
- The residuals estimate error present
- Error is allocated to the contributing components
Using the Errors

- Recall assumptions about error
- Compare time slices to find structural changes
  - Match up components then test for similarity
- Measure distances to anomalous points
  - We can find the standard deviation at any point on the manifold
  - Compare residual to standard deviation and test
Kernel Principal Component Analysis

- Non-linear manifold fitting algorithm
- Conceptually uses Principal Component Analysis (PCA) as a subroutine
  - Non-linearly maps data points (linearizes) into an abstract feature space
  - Performs PCA in feature space
- Errors
  - Error computation is conceptually the same
- Schölkopf et al. 1996
Kernels

- Feature space can be high or even infinite dimensional
  - Avoid computing in feature space
- Map two points into feature space and compute dot product simultaneously
  - Kernel function takes two data points and computes their dot products in feature space
  - Non-data points are expressed as linear combinations
- Example: polynomials of degree \( d \)
  \[
  k(x, y) = (x \cdot y + 1)^d
  \]
Future

Implementation
- Working kernel PCA implementation
- Hungarian algorithm for matching components
- Use constrained least-squares regression algorithm

Use
- Time slice incoming network data
- Compare fits between slices
- Classify regions of manifold as potential problems
Summary

➲ Problem arising from computer networks
➲ Application of Principal Component Analysis (PCA)
➲ Extensions to PCA
  ● Accounting for and using error
  ● Kernel PCA
➲ Future of project
Acknowledgements

- Richard and Dena Krown SURF Fellow
- SURF Office