

# Large Scale Data Analysis for Policy 90-866, Fall 2010

## Lecture 9: Anomaly and Outlier Detection

Parts of this lecture were adapted from Banerjee et al., *Anomaly Detection: A Tutorial*, presented at SDM 2008. I recommend viewing their excellent presentation for a more detailed discussion of anomaly detection and the current state of the art.

# What is detection?

The goal of the **detection** task is to automatically identify relevant patterns in massive, complex data.

Main goal: focus the user's attention on a potentially relevant subset of the data.

- a) Automatically **detect** relevant individual records, or groups of records.
- b) **Characterize** and **explain** patterns: pattern type, affected subset, models of normal/abnormal data.
- c) Present the pattern to the user.

## Some common detection tasks

Detecting **anomalous** records or groups

Discovering **novelties** (e.g. new drugs)

Detecting **clusters** in space or time

Removing **noise** or **errors** in data

Detecting **specific patterns** (e.g. fraud)

Detecting emerging **events** which may require rapid responses.

We will discuss two main topics in this module:

Detecting **individual records** that are anomalous or interesting.

Detecting interesting **groups** or **patterns** of records.

# Anomaly detection

In **anomaly (or outlier) detection**, we attempt to detect individual data records that are anomalous or unexpected.

Example 1: Given a massive database of financial data, which transactions are suspicious and likely to be **fraudulent**?

Example 2: Given the huge number of container shipments arriving at our country's ports every day, which should be opened by customs (to prevent smuggling, terrorism, etc.)?

Example 3: Given a log of all the traffic on our computer network, which sessions represent (attempted) **intrusions**?

Example 4: Given a sky survey of astronomical objects, which are **novelties** that might represent new scientific discoveries?



Goal: differentiate “normal” from “abnormal” records.

Abnormal records may be useful (e.g. novelties) or harmful (requiring action).

Removing anomalies can also improve our models of the normal data.

# Intrusion detection

- **Intrusions** are defined as attempts to bypass the security mechanisms of a computer or network in order to gain unauthorized access or enable unauthorized activities (e.g. stealing personal information).
- **Intrusion detection** is the process of monitoring the events occurring in a computer system or network and analyzing them for intrusions or attempted intrusions.
- Many challenges:
  - Traditional intrusion detection methods are based on detecting signatures of known attacks.
  - Substantial latency in deployment of newly created signatures across the system prevents rapid responses to emerging attack types.
- Anomaly detection can alleviate these limitations by automatically detecting previously unknown cyber-attacks.



(Adapted from Banerjee et al., *Anomaly Detection: A Tutorial*)

# Fraud detection

- Fraud detection refers to the detection of criminal activities occurring in commercial organizations.
  - Malicious users might be the actual customers of the organization or might be posing as a customer (identity theft).
- Types of fraud
  - Credit card fraud
  - Insurance claim fraud
  - Mobile / cell phone fraud
  - Insider trading
  - Online transaction fraud (eBay)
- Challenges
  - Fast and accurate real-time detection.
  - Predicting fraud based on previous transactions.
  - Costs of false positives and false negatives can both be high.



# Anomaly detection = classification?

One option is to treat anomaly detection as a binary classification problem, identifying each record as “anomalous” or “normal”.

Advantage: we can use any of the classification techniques from Module I (decision trees, k-nearest neighbor, naïve Bayes, etc.)

How to learn and evaluate classifiers with a skewed class distribution (e.g. 99.9% normal, 0.1% anomalies)?

Define  $A$  = # of anomalies detected  
 $B$  = total # of anomalies in data  
 $C$  = total # of records detected

Consider tradeoffs between  
Precision =  $A / C$ , Recall =  $A / B$ .  
(Why not just use accuracy?)

Typically, anomaly detection systems report potential anomalies to a human user, who can then decide whether or not to act on each case.

In this case, we want high recall (i.e. if any anomalies are present, we are very likely to report them). Precision is often less important, but higher precision means fewer potential anomalies the user has to sift through.

# Anomaly detection = classification?

One option is to treat anomaly detection as a binary classification problem, identifying each record as “anomalous” or “normal”.

Advantage: we can use any of the classification techniques from Module I (decision trees, k-nearest neighbor, naïve Bayes, etc.)

How to learn and evaluate classifiers with a skewed class distribution (e.g. 99.9% normal, 0.1% anomalies)?

Define  $A$  = # of anomalies detected  
 $B$  = total # of anomalies in data  
 $C$  = total # of records detected

Consider tradeoffs between  
Precision =  $A / C$ , Recall =  $A / B$ .  
(Why not just use accuracy?)

Typically, anomaly detection systems report potential anomalies to a human user, who can then decide whether or not to act on each case.

Early warning systems: users are willing to tolerate the occasional false alarm (weekly, monthly, etc.) but may start ignoring the system if it alerts too often.

Scientific discovery: novelties may be very rare (1 / billion), so users may be delighted with one true anomaly per thousand reports.

# Anomaly detection = classification?

One option is to treat anomaly detection as a binary classification problem, identifying each record as “anomalous” or “normal”.

Advantage: we can use any of the classification techniques from Module I (decision trees, k-nearest neighbor, naïve Bayes, etc.)

How to learn and evaluate classifiers with a skewed class distribution (e.g. 99.9% normal, 0.1% anomalies)?

Define  $A$  = # of anomalies detected  
 $B$  = total # of anomalies in data  
 $C$  = total # of records detected

Consider tradeoffs between  
Precision =  $A / C$ , Recall =  $A / B$ .  
(Why not just use accuracy?)

Cost-sensitive classification: penalize misclassification of anomalies more than misclassifying normal examples.

Simple example: Naïve Bayes classification gives posterior probability of each class.

For each example, choose the class with the highest value of (class probability x cost of misclassifying an example of that class).

# Anomaly detection = classification?

One option is to treat anomaly detection as a binary classification problem, identifying each record as “anomalous” or “normal”.

Advantage: we can use any of the classification techniques from Module I (decision trees, k-nearest neighbor, naïve Bayes, etc.)

In order to treat anomaly detection as (cost-sensitive) classification:

- 1) We need a large training dataset, with each record labeled “normal” or “anomaly”.
- 2) We need enough data to learn accurate models of both the normal and anomaly classes.
- 3) We can model only previously identified types of anomaly.

Real-world anomaly detection often fails to meet these criteria:

- 1) The training dataset may not have the anomalies labeled.
- 2) Anomalies are **rare**: few or no examples in training data.
- 3) We want to be able to detect any anomalies in the data, including anomaly types that we have never seen before.

Solution: Learn a model of the “normal” class only. Then detect any data records that are unlikely or unexpected given this model.

# Model-based anomaly detection

Our first step is to learn a model of the “normal” class C from data.

Ideally, we learn the model using “clean” training data (all examples known to be “normal”).  
In practice, we often have only an unlabeled dataset (assume anomalies are rare).

Naïve Bayes: assume all attributes independent.  
Learn each distribution by maximum likelihood.

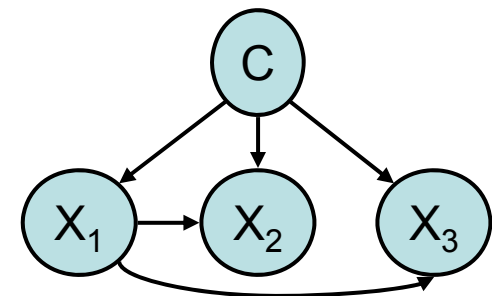
$$\Pr(X_1 \dots X_M | C) = \prod_{i=1..M} \Pr(X_i | C)$$

Discrete attribute: learn probability of each value.

Real-valued attribute: learn  $\mu$  and  $\sigma$ , assume Gaussian.

Bayesian network: specify or learn the structure.  
Then learn each attribute’s distribution, conditional on its parents’ values, by maximum likelihood.

$$\Pr(X_1 \dots X_M | C) = \prod_{i=1..M} \Pr(X_i | \text{Parents}(X_i))$$



If there are multiple “normal” classes and we have class labels or clusters, we can learn separate distributions for each class.

We can now compute the likelihood of each data record’s attribute values given the “normal” model(s), and report any records with likelihood below some threshold as anomalies.

# Model-based anomaly detection

Our first step is to learn a model of the “normal” class C from data.

Ideally, we learn the model using “clean” training data (all examples known to be “normal”). In practice, we often have only an unlabeled dataset (assume anomalies are rare).

Naïve Bayes: assume all attributes independent.  
Learn each distribution by maximum likelihood.

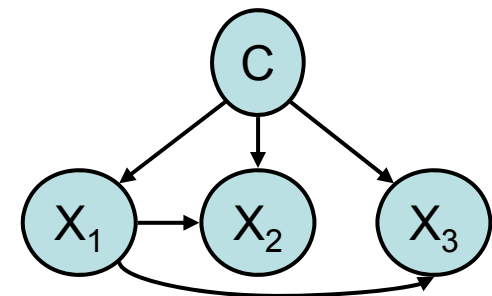
$$\Pr(X_1 \dots X_M | C) = \prod_{i=1..M} \Pr(X_i | C)$$

Discrete attribute: learn probability of each value.

Real-valued attribute: learn  $\mu$  and  $\sigma$ , assume Gaussian.

Bayesian network: specify or learn the structure.  
Then learn each attribute’s distribution, conditional on its parents’ values, by maximum likelihood.

$$\Pr(X_1 \dots X_M | C) = \prod_{i=1..M} \Pr(X_i | \text{Parents}(X_i))$$



The model-based approach to anomaly detection is often best if you can construct an accurate model for the normal data.

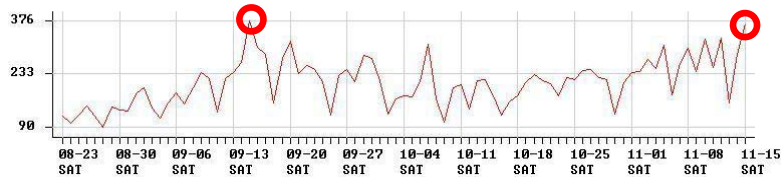
It does poorly in two cases: if the model representation is inadequate to describe the normal data, or if the model is corrupted by the unlabeled anomalies in the data.

# Spatial and temporal anomaly detection

One simple case of model-based anomaly detection is when we are monitoring a single real-valued quantity over time and/or space.

In this case, we typically want to report any observed value that is more than  $k$  standard deviations above or below its expected value.

## Time series data



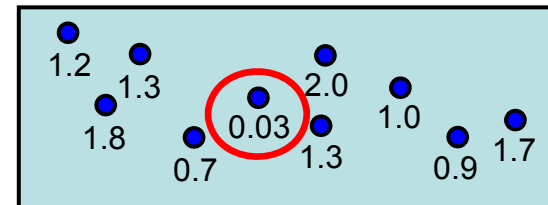
Time series analysis: the expected value for time step  $t$  is a function of the values for time steps 1 through  $t - 1$ .

Exponentially weighted averaging:

$$E[x_t] = (\sum w_i x_i) / (\sum w_i), w_i = e^{-(t-i)/b}$$

where  $i = 1 \dots t - 1$ .

## Spatially distributed data



Spatial regression: the expected value for location  $s$  is a function of the values for all other locations.

Kernel regression, exponential kernel:

$$E[x_s] = (\sum w_i x_i) / (\sum w_i), w_i = e^{-d(s, i)/b}$$

where  $i \neq s$  and  $d$  is Euclidean distance.

In the next lecture, we will discuss how to find anomalous **patterns** in space-time data.

# Distance-based anomaly detection

Given an unlabeled dataset  $x_1..x_N$  and a distance metric  $d(x_i, x_j)$ , we can use pairwise distances between records to detect anomalies.

Key assumption: normal records are similar to many other records, while anomalies are very different from most other records.

Approach 1: Choose a threshold distance  $D$ . For each record  $x_i$ , compute the fraction  $f_D(x_i)$  of other records with  $d(x_i, x_j) < D$ . The records with the lowest values of  $f_D$  are most anomalous.

Approach 2: Choose a number of neighbors  $k$ . For each record  $x_i$ , compute the distance  $d_k(x_i)$  to its  $k$ th nearest neighbor. The records with the highest values of  $d_k$  are most anomalous.

The advantages and disadvantages of these methods are similar to instance-based learning:

- No assumptions about distribution of the normal data; can model complex distributions.
- Computationally expensive (run time increases quadratically with size of the dataset).
- Difficult to choose a good value of the threshold  $D$  or number of neighbors  $k$ .
- Curse of dimensionality: distance between points becomes uniform in high dimensions.
- Poor performance when data is of variable density or has underlying cluster structure.

# Cluster-based anomaly detection

Given a clustering of an unlabeled dataset (e.g. by k-means), we can use the resulting clusters for anomaly detection in various ways.

Key assumption: normal records belong to large, dense clusters. Anomalies do not fit the clusters, or form their own tiny clusters.

Approach 1: Given a separate “clean” training set, cluster the normal data, and optionally learn a model for each cluster. If a test record is far from all cluster centers (or has low likelihood given any model), label it an anomaly.

Approach 2: Given an unlabeled dataset in which we wish to find anomalies, cluster the data, and then report any tiny clusters (# of records  $\leq k$ ) as anomalies. Also report any records that are far from all cluster centers.

Approach 1 is similar to model-based detection, but does not assume that different “normal” classes are labeled.

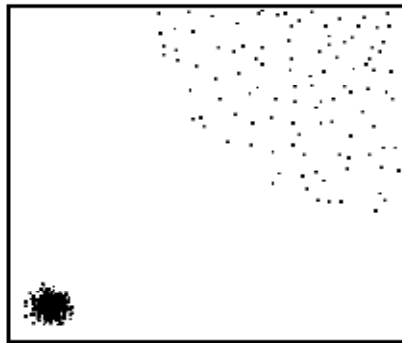
Approach 2 is similar to distance-based anomaly detection, but uses a metric that takes cluster structure into account.

When will this work better (or worse) than distance-based detection?

# Density-based anomaly detection

Density-based anomaly detection approaches perform density estimation at each data point, and identify records in low-density regions as potential anomalies.

“Global” approaches compare each point’s density to the densities of all other points, and report points  $x_i$  with lowest density.



In this example, the density of points is high in the lower left corner, lower in the upper right corner, and very low in between.

“Local” approaches compare each point’s density to the densities of nearby points, and report points  $x_i$  with lowest ratio:

$$\frac{\text{density}(x_i)}{\text{avg density of } k\text{-NN}(x_i)}$$

Distance-based and “global” density-based methods would consider points in the upper right corner to be more anomalous (lower density, larger distance between points).

“Local” density-based methods would not consider these points as more anomalous, since their neighbors also have low density.

# Density-based anomaly detection

Density-based anomaly detection approaches perform density estimation at each data point, and identify records in low-density regions as potential anomalies.

“Global” approaches compare each point’s density to the densities of all other points, and report points  $x_i$  with lowest density.

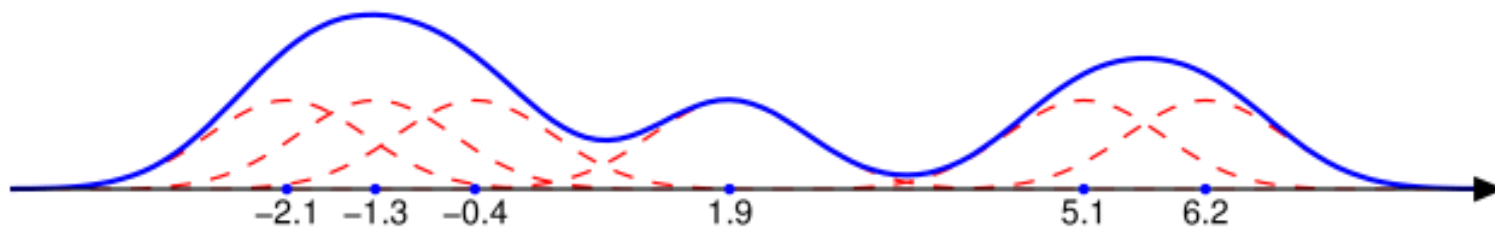
“Local” approaches compare each point’s density to the densities of nearby points, and report points  $x_i$  with lowest ratio.

There are many ways to compute the density at a point  $x_i$ .

## Kernel density estimation

Consider a probability density function  $f(x)$ , e.g. Gaussian, centered at each data point  $x_j$ .

$$\text{density}(x_i) = \frac{\sum_{j=1..N} f(x_i | \mu = x_j, \sigma)}{N}$$



# Density-based anomaly detection

Density-based anomaly detection approaches perform density estimation at each data point, and identify records in low-density regions as potential anomalies.

“Global” approaches compare each point’s density to the densities of all other points, and report points  $x_i$  with lowest density.

“Local” approaches compare each point’s density to the densities of nearby points, and report points  $x_i$  with lowest ratio.

There are many ways to compute the density at a point  $x_i$ .

## Kernel density estimation

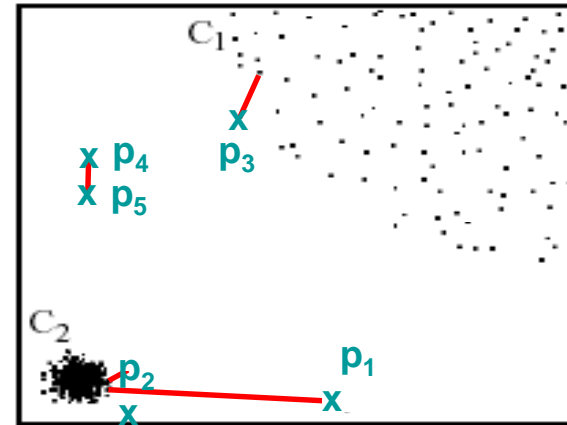
Consider a probability density function  $f(x)$ , e.g. Gaussian, centered at each data point  $x_j$ .

$$\text{density}(x_i) = \frac{\sum_{j=1..N} f(x_i | \mu = x_j, \sigma)}{N}$$

Many simpler density estimators also exist, for example the inverse of the distance to the  $k$ th nearest neighbor, or the inverse of the average distance to the  $k$ -NN.

# Comparison of detection methods

Let us assume that we have an unlabeled dataset with two real-valued attributes. Which anomaly detection methods will classify each point  $p_i$  as an anomaly?



For **distance-based** anomaly detection using 1-nearest neighbor,  $p_1$  and  $p_3$  are the two most anomalous points.

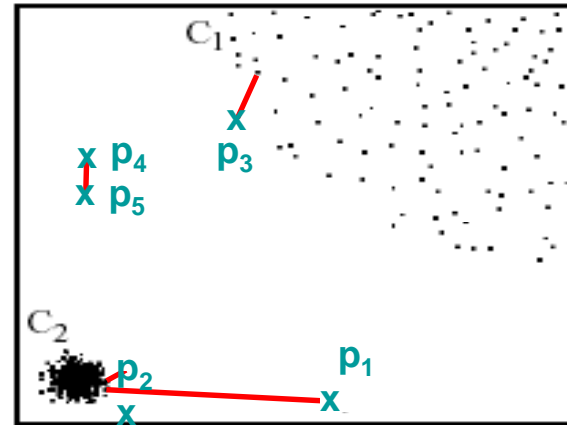
For **global density-based** anomaly detection, point  $p_2$  is less anomalous than points in  $C_1$  since its density is higher.

For two or more neighbors, points  $p_1$ ,  $p_4$ , and  $p_5$  are most anomalous.  $p_2$  is close to many points in  $C_2$ , so it is less anomalous than points in  $C_1$ .

For **local density-based** detection,  $p_2$  is more anomalous than points in  $C_1$ .  $p_2$  has lower density than its neighbors, while  $C_1$  has nearly uniform density.

# Comparison of detection methods

Let us assume that we have an unlabeled dataset with two real-valued attributes. Which anomaly detection methods will classify each point  $p_i$  as an anomaly?



Most **cluster-based** anomaly detection methods, assuming  $k = 2$  clusters, would form clusters roughly corresponding to  $C_1$  and  $C_2$ .

Points  $p_1$ ,  $p_4$ , and  $p_5$  are far from the clusters, and would be detected as anomalies. Point  $p_2$  would probably be detected if we modeled each cluster's standard deviation  $\sigma$ , while it would not be detected if we measured distance to the cluster center.

If we used  $k > 2$  clusters, point  $p_1$  and points  $p_4$ - $p_5$  might form separate clusters. This is why we also detect small clusters!

# Summary of anomaly detection

- Given a massive dataset, anomaly detection can be used to find individual records with surprising combinations of attribute values.
- Common applications include security (detection of intrusions, fraud, smuggling, etc.), scientific discovery, and data cleaning.
- Which anomaly detection method to use depends on the type of data available, and also how we expect anomalies to differ from normal data.
  - Given sufficient labeled data for each possible class of anomalies → use cost-sensitive classification.
  - Given a separate “clean” dataset representing normal data behavior → use model-based anomaly detection.
  - Detecting anomalies in an unlabeled dataset → use distance-based, density-based, or cluster-based anomaly detection.

# References

- A. Banerjee, V. Chandola, V. Kumar, J. Srivastava, and A. Lazarevic. *Anomaly Detection: A Tutorial*. Available online at: <http://www.siam.org/meetings/sdm08/TS2.ppt>
- Distance-based anomaly detection: E. Knorr and R. Ng, “Algorithms for mining distance-based outliers in large datasets.” In *Proc. VLDB*, 1998.
- Density-based anomaly detection: M.M. Breunig et al., “LOF: Identifying density-based local outliers.” In *Proc. KDD*, 2000.

# Large Scale Data Analysis for Policy

## 90-866, Fall 2010

### Lecture 10: Anomalous Pattern Detection

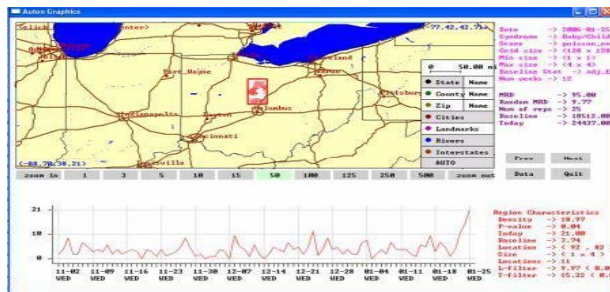
# Anomalous pattern detection

Main goal of pattern detection: to **identify** and **characterize** relevant subsets of a massive dataset, i.e. groups of records that differ from the rest of the data in an interesting way.

Question 1: Are any relevant patterns present in the data, or is the entire dataset “normal”?

Example: outbreak detection

Are there any emerging outbreaks of disease? If so, what type of outbreak, and what areas are affected?



Question 2: If there are any patterns, identify the pattern type and the affected subset of data records for each.

Example: intelligence analysis

Can we deduce the membership and structure of terrorist groups based on known links between suspected individuals?



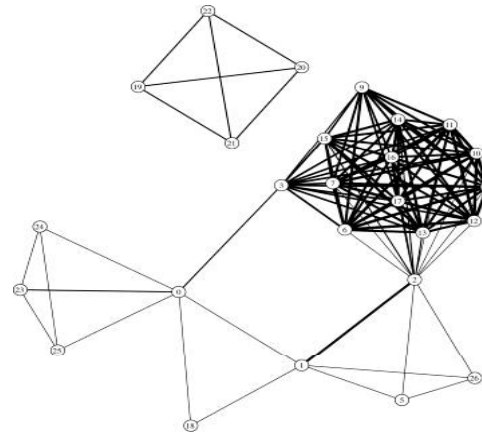
# Anomalous pattern detection

Main goal of pattern detection: to **identify** and **characterize** relevant subsets of a massive dataset, i.e. groups of records that differ from the rest of the data in an interesting way.

What makes a group of records “relevant”?

1. Matching some known pattern or structure.

Group detection: given a social network, find highly connected sets of individuals.



Many efficient algorithms have been developed to find dense subgraphs or other structures in network data.

# Anomalous pattern detection

Main goal of pattern detection: to **identify** and **characterize** relevant subsets of a massive dataset, i.e. groups of records that differ from the rest of the data in an interesting way.

What makes a group of records “relevant”?

1. Matching some known pattern or structure.
2. Multiple related records that are individually anomalous.

Fraud detection: look for individuals with a history of suspicious transactions.

Network intrusion detection: look for suspicious combinations of activities (e.g. port scanning).

In these domains, multiple “slightly anomalous” behaviors may together provide evidence of a major deviation from normal.

# Anomalous pattern detection

Main goal of pattern detection: to **identify** and **characterize** relevant subsets of a massive dataset, i.e. groups of records that differ from the rest of the data in an interesting way.

What makes a group of records “relevant”?

1. Matching some known pattern or structure.
2. Multiple related records that are individually anomalous.
3. Higher (or lower) than expected number of records with some combination of attributes.
4. Change in data distribution as compared to the rest of the dataset.

Cluster detection: find spatial areas or periods of time with more records than expected.

Event detection: is the recent data differently distributed than the past?

Key concept: A group of records may be highly anomalous or interesting even if none of the individual records is itself anomalous.

# Pattern detection = classification?

“Does the dataset contain a pattern?”  
This question could be treated as  
a problem of classifying datasets.

Dataset  $D \rightarrow \{\text{normal, contains a pattern}\}$   
Or  $D \rightarrow \{\text{normal, pattern 1, pattern 2, ...}\}$

“Which records are affected?”  
This question could be treated as  
classification or anomaly detection.

Record  $x_i \rightarrow \{\text{normal, is part of a pattern}\}$   
Or  $x_i \rightarrow \{\text{normal, pattern 1, pattern 2, ...}\}$

For the first approach, what features of the dataset should we use?

Most relevant patterns only affect a small proportion of the dataset,  
and would not be visible looking only at summary statistics.

On the other hand, data sparsity prevents using each record-  
attribute combination as a different attribute of the dataset.

For the second approach, how can we combine data from multiple records?

This is essential since none of the records may be  
individually sufficient to detect the anomalous pattern.

Both “top-down” and “bottom-up” greedy approaches to detection fail.

# Subset scanning

We can scan over subsets of the dataset in order to find those groups of records that correspond to a pattern.

Step 1: Compute **score**  $F(S, P)$  for each subset  $S = \{x_i\}$  and for each pattern type  $P$ , where higher score means more likely to be a pattern.

Step 2: Consider the highest scoring potential patterns  $(S, P)$  and decide whether each actually represents a pattern.

There are many options for computing the score of a subset  $S$ .

# What's Strange About Recent Events?

We can scan over subsets of the dataset in order to find those groups of records that correspond to a pattern.

Step 1: Compute **score**  $F(S, P)$  for each subset  $S = \{x_i\}$  and for each pattern type  $P$ , where higher score means more likely to be a pattern.

Step 2: Consider the highest scoring potential patterns  $(S, P)$  and decide whether each actually represents a pattern.

There are many options for computing the score of a subset  $S$ .

In the WSARE method (“What’s Strange About Recent Events”), we consider the subsets of the data defined by a one- or two-component rule  $R$ , and find rules where the current data is significantly different than the past.

For each rule, we create a 2x2 contingency table comparing current and past data:

	<u>Current</u>	<u>Past</u>
# records satisfying $R$	48	45
# records satisfying $\sim R$	86	220



Compute p-value using a statistical test ( $X^2$  or Fisher’s Exact). Lower p-value = higher score.

# What's Strange About Recent Events?

We can scan over subsets of the dataset in order to find those groups of records that correspond to a pattern.

Step 1: Compute **score**  $F(S, P)$  for each subset  $S = \{x_i\}$  and for each pattern type  $P$ , where higher score means more likely to be a pattern.

Step 2: Consider the highest scoring potential patterns  $(S, P)$  and decide whether each actually represents a pattern.

There are many options for computing the score of a subset  $S$ .

In the WSARE method (“What’s Strange About Recent Events”), we consider the subsets of the data defined by a one- or two-component rule  $R$ , and find rules where the current data is significantly different than the past.

For example, using WSARE for hospital Emergency Department surveillance resulted in finding the following significant rule, corresponding to an outbreak of respiratory illness on 9/6/2000.

```
### Rule 3: Wed 09-06-2000 (daynum 36774, dayindex 131)
SCORE = -0.00000000 PVALUE = 0.00000000
 17.16% ( 23/134) of today's cases have Prodrome = Respiratory
and age2 less than 40
  4.53% ( 12/265) of other cases have Prodrome = Respiratory
and age2 less than 40
```

# Anomaly pattern detection

We can scan over subsets of the dataset in order to find those groups of records that correspond to a pattern.

Step 1: Compute **score**  $F(S, P)$  for each subset  $S = \{x_i\}$  and for each pattern type  $P$ , where higher score means more likely to be a pattern.

Step 2: Consider the highest scoring potential patterns  $(S, P)$  and decide whether each actually represents a pattern.

There are many options for computing the score of a subset  $S$ .

If we do not have access to past data, but we have access to the output of an anomaly detector, we can modify WSARE to detect rules  $R$  that correspond to a higher than expected number of anomalous records.

Now we create tables comparing the numbers of anomalous and normal records:

	<u>Anomalous</u>	<u>Normal</u>
# records satisfying $R$	17	5
# records satisfying $\sim R$	93	400



Compute p-value using a statistical test ( $X^2$  or Fisher's Exact). Lower p-value = higher score.

# Model-based pattern detection

We can scan over subsets of the dataset in order to find those groups of records that correspond to a pattern.

Step 1: Compute **score**  $F(S, P)$  for each subset  $S = \{x_i\}$  and for each pattern type  $P$ , where higher score means more likely to be a pattern.

Step 2: Consider the highest scoring potential patterns  $(S, P)$  and decide whether each actually represents a pattern.

There are many options for computing the score of a subset  $S$ .

In the **model-based** anomalous pattern detection approach, we model the effects of each pattern type  $P$  on the affected subset of the data  $S$ .

We then compute the **likelihood ratio statistic**  
 $\Pr(\text{Data} \mid H_1(S, P)) / \Pr(\text{Data} \mid H_0)$  for each  $(S, P)$ .

In **event detection**, we model the null hypothesis  $H_0$  by estimating expected counts for each data stream assuming no events.

Each pattern  $P$  is assumed to increase the counts for some data streams in the affected set of spatial locations  $S$ .

# Anomalous group detection

We can scan over subsets of the dataset in order to find those groups of records that correspond to a pattern.

Step 1: Compute **score**  $F(S, P)$  for each subset  $S = \{x_i\}$  and for each pattern type  $P$ , where higher score means more likely to be a pattern.

Step 2: Consider the highest scoring potential patterns  $(S, P)$  and decide whether each actually represents a pattern.

There are many options for computing the score of a subset  $S$ .

In the **model-based** anomalous pattern detection approach, we model the effects of each pattern type  $P$  on the affected subset of the data  $S$ .

We then compute the **likelihood ratio statistic**  
 $\Pr(\text{Data} \mid H_1(S, P)) / \Pr(\text{Data} \mid H_0)$  for each  $(S, P)$ .

In our **AGD** (“Anomalous Group Detection”) approach, we model the null hypothesis by learning a Bayes Net from training data.

Under the alternative hypothesis  $H_1(S)$ , we assume that records in  $S$  are drawn from a different Bayes Net.

# Conditional pattern detection

We can scan over subsets of the dataset in order to find those groups of records that correspond to a pattern.

Step 1: Compute **score**  $F(S, P)$  for each subset  $S = \{x_i\}$  and for each pattern type  $P$ , where higher score means more likely to be a pattern.

Step 2: Consider the highest scoring potential patterns  $(S, P)$  and decide whether each actually represents a pattern.

There are many options for computing the score of a subset  $S$ .

In the **model-based** anomalous pattern detection approach, we model the effects of each pattern type  $P$  on the affected subset of the data  $S$ .

We then compute the **likelihood ratio statistic**  
 $\Pr(\text{Data} \mid H_1(S, P)) / \Pr(\text{Data} \mid H_0)$  for each  $(S, P)$ .

Finally, rather than computing the entire data likelihood, we can compute the likelihood of some “output attributes” conditional on other “input attributes”.

If we don't know the input and output attributes, we can scan over subsets of input and output attributes as well!

# Which patterns to report?

We can scan over subsets of the dataset in order to find those groups of records that correspond to a pattern.

Step 1: Compute **score**  $F(S, P)$  for each subset  $S = \{x_i\}$  and for each pattern type  $P$ , where higher score means more likely to be a pattern.

Step 2: Consider the highest scoring potential patterns  $(S, P)$  and decide whether each actually represents a pattern.

Option 1: Report the  $k$  highest scoring subsets, ordered by score.

The disadvantage of this approach is that the user is not informed whether any of the discovered patterns are likely to be relevant.

However, this may be acceptable in monitoring systems or scientific discovery applications where the user is willing to evaluate a fixed number of potential patterns.

# Which patterns to report?

We can scan over subsets of the dataset in order to find those groups of records that correspond to a pattern.

Step 1: Compute **score**  $F(S, P)$  for each subset  $S = \{x_i\}$  and for each pattern type  $P$ , where higher score means more likely to be a pattern.

Step 2: Consider the highest scoring potential patterns  $(S, P)$  and decide whether each actually represents a pattern.

Option 2: Perform hypothesis tests, and report all **significant** patterns  $(S, P)$ .

In the hypothesis testing framework, we must adjust for the fact that we're performing so many tests. Otherwise we will report too many false positives!

In model-based approaches, one way to do this is **randomization**: we generate a large number of simulated datasets assuming the null model, and compare the scores of the potential patterns in the real dataset to the highest scoring patterns in the simulated data.

An alternative is to adjust the p-value threshold for each test based on the number of tests performed (e.g. Bonferroni threshold =  $.05 / \# \text{ tests}$ )

# Which patterns to report?

We can scan over subsets of the dataset in order to find those groups of records that correspond to a pattern.

Step 1: Compute **score**  $F(S, P)$  for each subset  $S = \{x_i\}$  and for each pattern type  $P$ , where higher score means more likely to be a pattern.

Step 2: Consider the highest scoring potential patterns  $(S, P)$  and decide whether each actually represents a pattern.

Option 3: Compute the **posterior probability** of each hypothesis  $H_1(S, P)$ .

In a Bayesian framework, we must spread the prior probability of a pattern over all possible hypotheses  $H_1(S, P)$ .

We then compute the likelihood of the data given each hypothesis  $H_1(S, P)$ , as well as the null hypothesis of no patterns,  $H_0$ .

We can then compute the posterior probability of each hypothesis by Bayes' Theorem:

$$\Pr(H | D) = \Pr(D | H) \Pr(H) / \Pr(D)$$

Best for known pattern types!

Option 2 better for anomalies!

# Which subsets to scan?

Since there are exponentially many subsets of the data, it is often computationally infeasible to search all of them.

The most common approach is to use domain knowledge to restrict our search space: for example, in spatial cluster detection, we assume that a pattern will affect a spatially localized group of records, and often further restrict the cluster size and shape.

e.g. “search over circular regions centered at a data point” → only  $N^2$  regions instead of  $2^N$ .

Another common approach is to perform a greedy search. For example, we grow subsets starting from each record, repeatedly adding the additional record that gives the highest scoring subset.

Tradeoff: much more efficient than naïve search, but not guaranteed to find highest scoring region.

In some cases, we can find the highest-scoring subsets without actually computing the scores of all possible subsets!

# Example: fast spatial scan

In **spatial cluster detection**, we search over sets of adjacent locations and find spatial regions with significantly higher than expected counts.

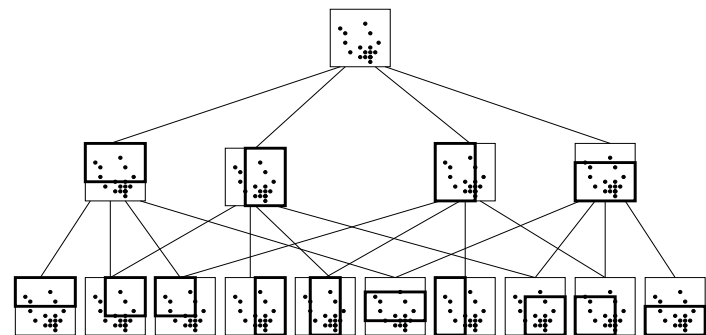
We restrict our search to rectangular regions for computational efficiency. This gives us high detection power for both compact and elongated clusters.

For massive datasets (e.g. disease surveillance for nationwide health data), we have to search over billions of possible regions, which could take weeks.

We can find the highest scoring clusters without an exhaustive search using **branch and bound**: we keep track of the highest region score that we have found so far, and prune sets of regions with provably lower scores.

A new multi-resolution data structure, the overlap-kd tree, enables us to make this search efficient.

We can now monitor nationwide health data in 20 minutes (vs. 1 week).



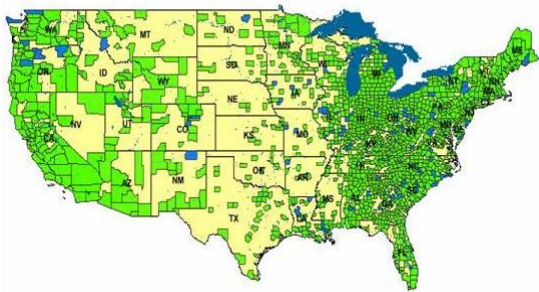
# Linear-time subset scanning

Given a score function  $F(S)$  which satisfies the **linear-time subset scanning** property, we can optimize  $F(S)$  over the exponentially many subsets of data records, while evaluating only  $O(N)$  regions instead of  $O(2^N)$ .

Just sort the locations from highest to lowest **priority** according to some function, then search over groups consisting of the top- $k$  highest priority locations ( $k = 1..N$ ). The highest scoring subset will be one of these!

Many useful score functions satisfy the LTSS property. In the spatial cluster detection setting, we can efficiently optimize a **spatial scan statistic** over subsets of locations to find the most interesting spatial region.

This works both for **univariate data**, monitoring a single data stream across time and space, and **multivariate data**, monitoring multiple data streams.



We can also incorporate relevant constraints such as spatial proximity, temporal consistency, or graph connectivity into the detection process.

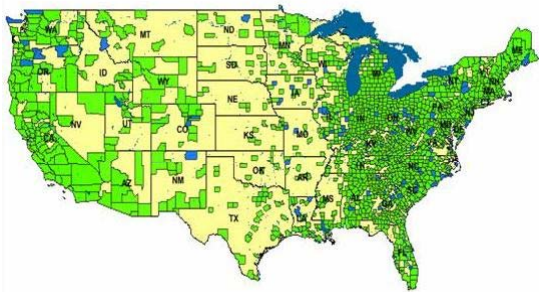
# Linear-time subset scanning

Given a score function  $F(S)$  which satisfies the **linear-time subset scanning** property, we can optimize  $F(S)$  over the exponentially many subsets of data records, while evaluating only  $O(N)$  regions instead of  $O(2^N)$ .

Just sort the locations from highest to lowest **priority** according to some function, then search over groups consisting of the top- $k$  highest priority locations ( $k = 1..N$ ). The highest scoring subset will be one of these!

Many useful score functions satisfy the LTSS property. In the spatial cluster detection setting, we can efficiently optimize a **spatial scan statistic** over subsets of locations to find the most interesting spatial region.

This works both for **univariate data**, monitoring a single data stream across time and space, and **multivariate data**, monitoring multiple data streams.



LTSS allows us to solve problems in milliseconds that would previously have required hundreds of millions of years!

# References

- A coherent text on anomalous pattern detection has yet to be written, but many methods have been proposed and are becoming common:
  - WSARE: W.-K. Wong et al., “Rule-based anomaly pattern detection for detecting disease outbreaks,” *Proc. 18th Natl. Conf. on Artificial Intelligence*, 2002.
  - APD (“Anomaly Pattern Detection”). K. Das, J. Schneider, and D.B. Neill, *Proc. KDD 2008*.
  - D.B. Neill and W.-K. Wong, “A Tutorial on Event Detection,” presented at *KDD 2009* conference.
- Software for spatial cluster detection and for WSARE is available on the Auton Laboratory web page, <http://www.autonlab.org>.