Machine Learning for Cities
CUSP-GX 5006.001, Spring 2019

Lecture 11: 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 methodologies.

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 this week and next:

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.

Anomalies in urban systems

What sort of anomalies might we be most interested in detecting?

Safety/security/law enforcement: crime, terrorism, code violations…

Power grid: excessive energy use, blackouts/brownouts…

Sensors: Lighting, noise, temperature…

Transportation: accidents, traffic congestion, transit delays…

Events: severe weather, mass gatherings, …

Provision of city services: e.g., 311 calls and city responses

Many others…

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 we’ve learned (decision trees, random forests, SVMs, 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 we’ve learned (decision trees, random forests, SVMs, 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 we’ve learned (decision trees, random forests, SVMs, 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: Bayesian Network 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 we’ve learned (decision trees, random forests, SVMs, 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.

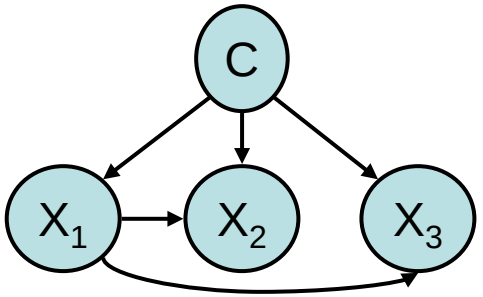
Ideally, we learn the model using "clean" training data (all examples known to be "normal").
In practice, we often have only an unlabeled dataset (assume anomalies are rare).

Naïve Bayes: assume all attributes independent.
Learn each distribution by maximum likelihood.
$Pr(X_1..X_M \mid C) = \prod_{i=1..M} Pr(X_i \mid C)$

Discrete attribute: learn probability of each value.

Real-valued attribute: learn μ and σ, assume Gaussian.

Bayesian network: specify or learn the structure.
Then learn each attribute's distribution, conditional on its parents' values, by maximum likelihood.
$Pr(X_1..X_M \mid C) = \prod_{i=1..M} Pr(X_i \mid Parents(X_i))$



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.

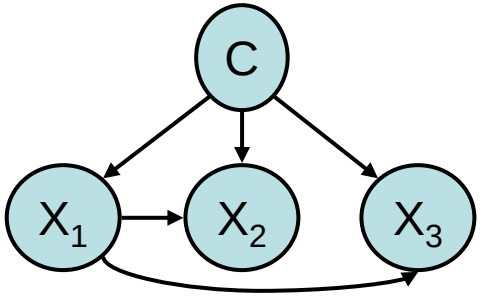
Ideally, we learn the model using "clean" training data (all examples known to be "normal").
In practice, we often have only an unlabeled dataset (assume anomalies are rare).

Naïve Bayes: assume all attributes independent.
Learn each distribution by maximum likelihood.
$Pr(X_1..X_M \mid C) = \prod_{i=1..M} Pr(X_i \mid C)$

Discrete attribute: learn probability of each value.

Real-valued attribute: learn μ and σ, assume Gaussian.

Bayesian network: specify or learn the structure.
Then learn each attribute's distribution, conditional on its parents' values, by maximum likelihood.
$Pr(X_1..X_M \mid C) = \prod_{i=1..M} Pr(X_i \mid Parents(X_i))$



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

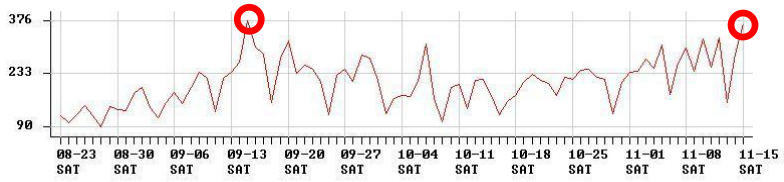
It does poorly in two cases: if the model representation is inadequate to describe the normal data, or if the model is corrupted by the unlabeled anomalies in the data.

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.

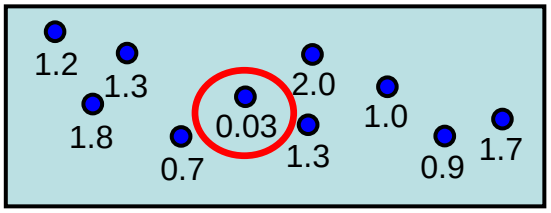
Time series data



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:
$E[x_t] = (\Sigma\ w_i x_i) / (\Sigma\ w_i)$, $w_i = e^{-(t-i)/b}$
where i = 1… 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:
$E[x_s] = (\Sigma\ w_i x_i) / (\Sigma\ w_i)$, $w_i = e^{-d(s,\ i)/b}$
where $i \neq s$ and d is Euclidean distance.

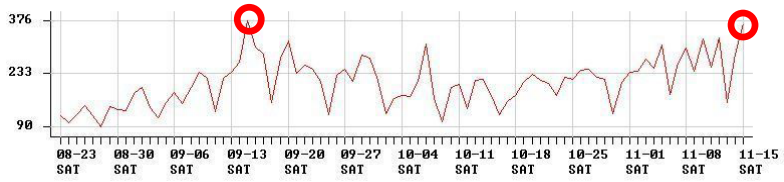
Gaussian processes could also be used to identify anomalies in these cases.

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.

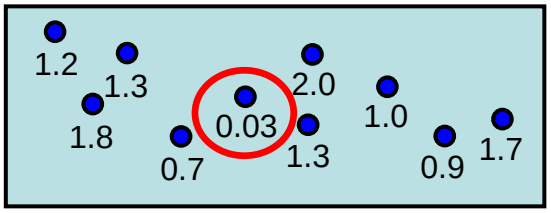
Time series data



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:
$E[x_t] = (\Sigma\ w_i x_i) / (\Sigma\ w_i)$, $w_i = e^{-(t-i)/b}$
where i = 1… 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:
$E[x_s] = (\Sigma\ w_i x_i) / (\Sigma\ w_i)$, $w_i = e^{-d(s,\ i)/b}$
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 kth 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 ≤ 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.

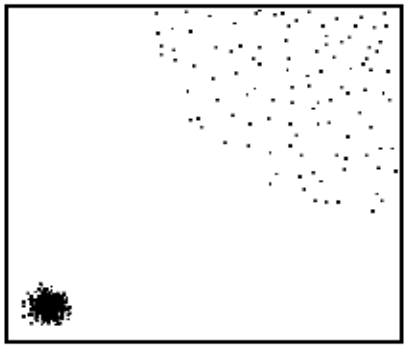
Approach 2 is similar to distance-based anomaly detection, but uses a metric that takes cluster structure into account.

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:

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

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.

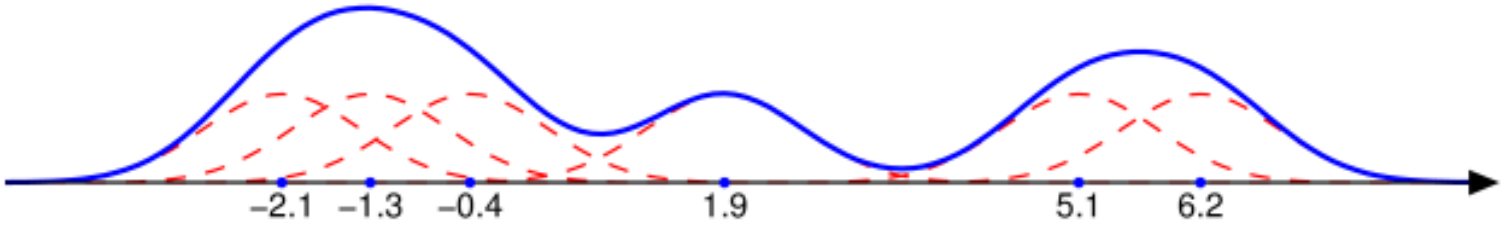
"Local" approaches compare each point's density to the densities of nearby points, and report points x_i with lowest ratio.

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.

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



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

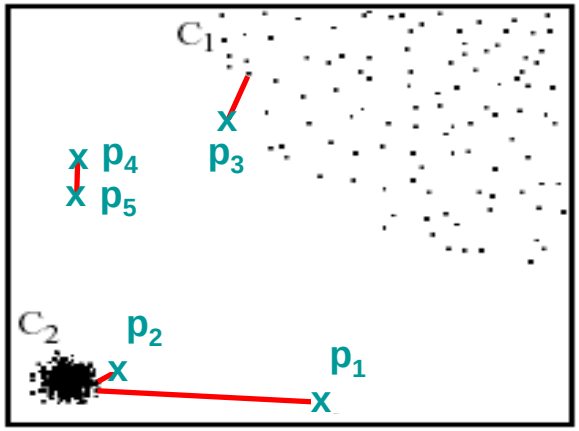
Consider a probability density function f(x), e.g. Gaussian, centered at each data point x_j.

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

Many simpler density estimators also exist, for example the inverse of the distance to the kth 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.

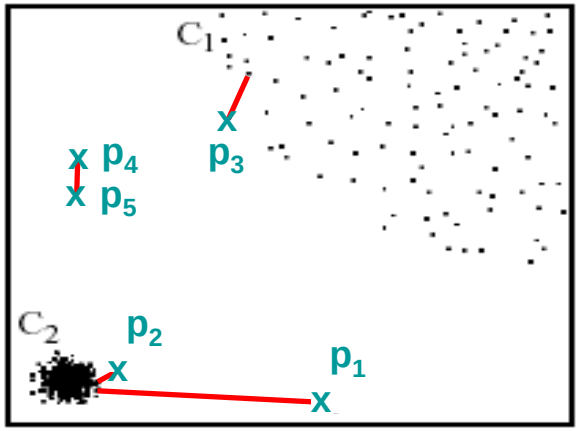
For **global density-based** anomaly detection, point p_2 is less anomalous than points in C_1 since its density is higher.

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 σ, 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, but also useful for urban data.
- 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 (rarely happens!) → 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.

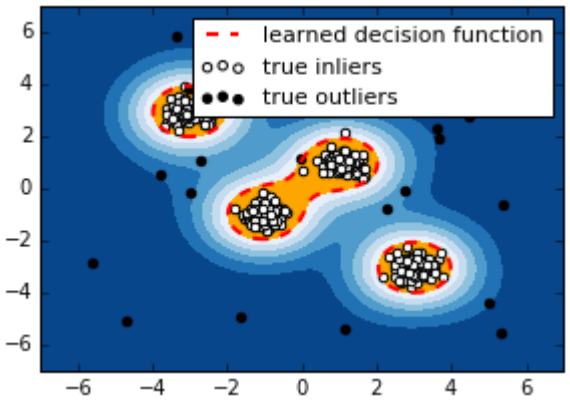
Two additional methods (in sklearn)

One-class SVM (Scholkopf et al., 2001):

Given “normal” training data, one-class SVM learns a non-linear decision boundary surrounding the normal class.

Kernel trick: find hyperplane in high dimensional feature space separating normal data points from the origin (also maximize distance from plane to origin).

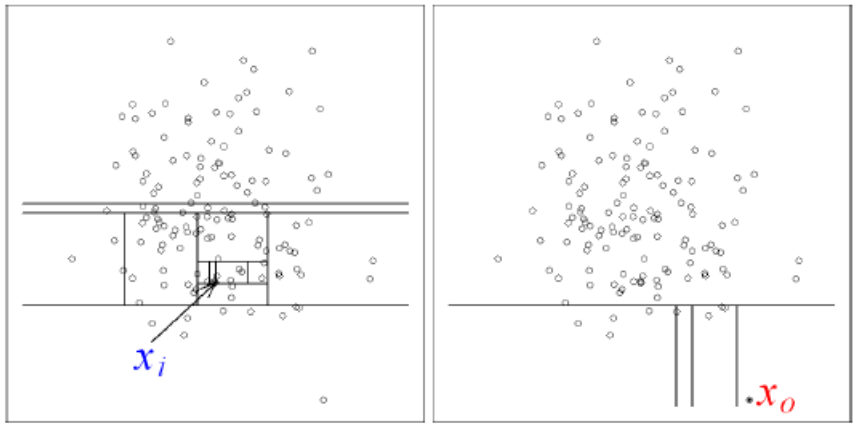
Identify a subset S of input space such that the probability of a test point lying outside S is upper-bounded by constant.



Isolation forest (Liu et al., 2008):

Across multiple trees in a “totally random” forest (at each split, randomly select split attribute and value), measure the average number of splits needed to isolate each data point from the rest of the data.

Fast, scalable, linear-time algorithm: does not have to compute pairwise distances.



(a) Isolating x_i (b) Isolating x_o

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.
- One-class SVM: B. Scholkopf et al., “Estimating the support of a high-dimensional distribution”, *Neural Computation* 13(7), 1443-71, 2001.
- Isolation forest: F.T. Liu et al., in *Proc. ICDM 2008*.
- Sklearn includes: local density-based anomaly detection (LOF), one-class SVM, isolation forest.