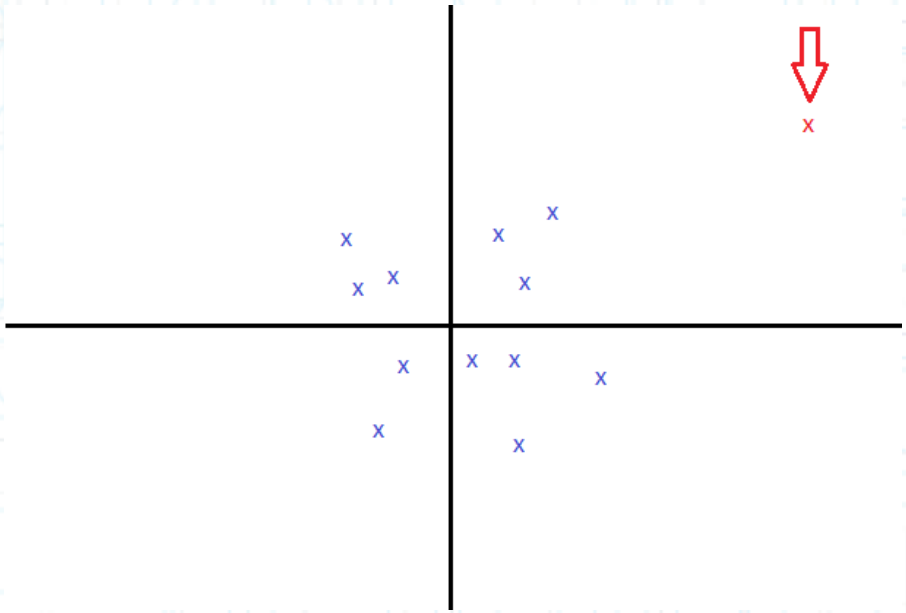
Anomalous Data Detection

Gary Steelman
University of Texas at Dallas
2012

Problem Definition

- Given a data matrix of m instances with n attributes per instance
  - Which instances aren't what we would call "normal"? Which ones are outliers?



Why do we care? How do we do it?

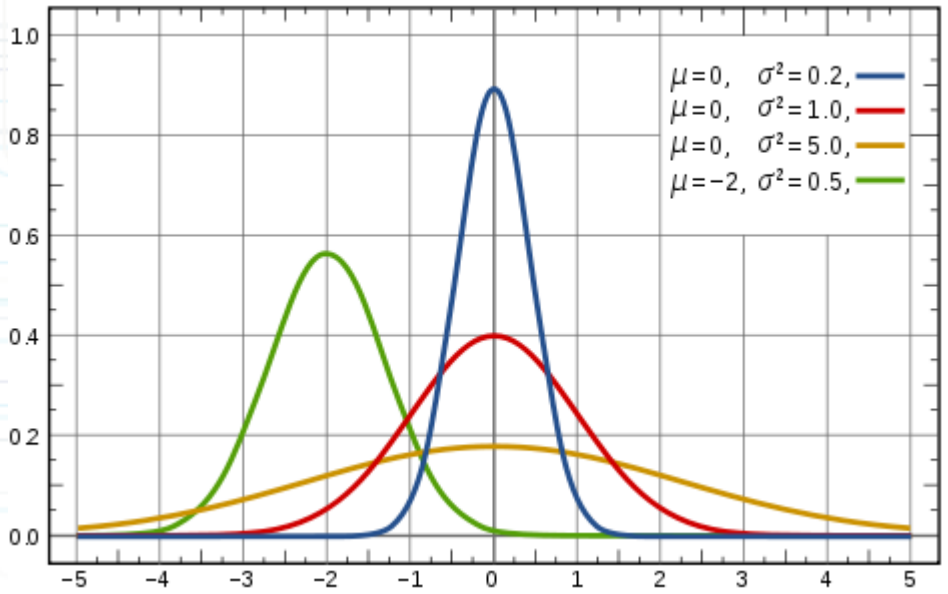
- Useful for detecting abnormal activity
  - Fraud detection/Identity theft
  - Hardware manufacturing quality control
  - Sensor data
  - Robotics
- We have to build a model of what is considered “normal” data (blue points)
  - Then compare any new instances (red point) against the model and see how closely it matches (or not!)

Model 1 – Univariate Gaussians

- Train a univariate Gaussian for each attribute of the data, to learn what each attribute *should* look like (if it follows a normal distribution)

$$\mu_{MLE} = \frac{1}{n}\sum_{i=0}^{n} x_i \qquad \sigma^2_{MLE} = \frac{1}{n}\sum_{i=0}^{n} (x_i - \mu_{MLE})^2$$

$$p(x|\mu_{MLE}, \sigma^2_{MLE})$$



Model 1 (continued)

- Sample each of the Gaussians by passing in the j^{th} value from *x* and get back the j^{th} Gaussian's probability $p_j(x_j)$
- Multiply all the different $p_j(x_j)$ together to get an estimate for p(x), p^(x)
- Compare p^(x) to some ε and if p^(x) < ε classify *x* as anomalous

$$\vec{x} = \langle 5.0, 12.1, -6.9, 16.0 \rangle$$
$$\downarrow \quad \downarrow \quad \downarrow \quad \downarrow$$
$$\vec{G} = [g_1, \; g_2, \; g_3, \; g_4]$$
$$\downarrow \quad \downarrow \quad \downarrow \quad \downarrow$$
$$\{p_j(x_j)\} = \underbrace{\{.3, \quad .7, \quad .12, \quad .55\}}_{\text{multiply together}}$$

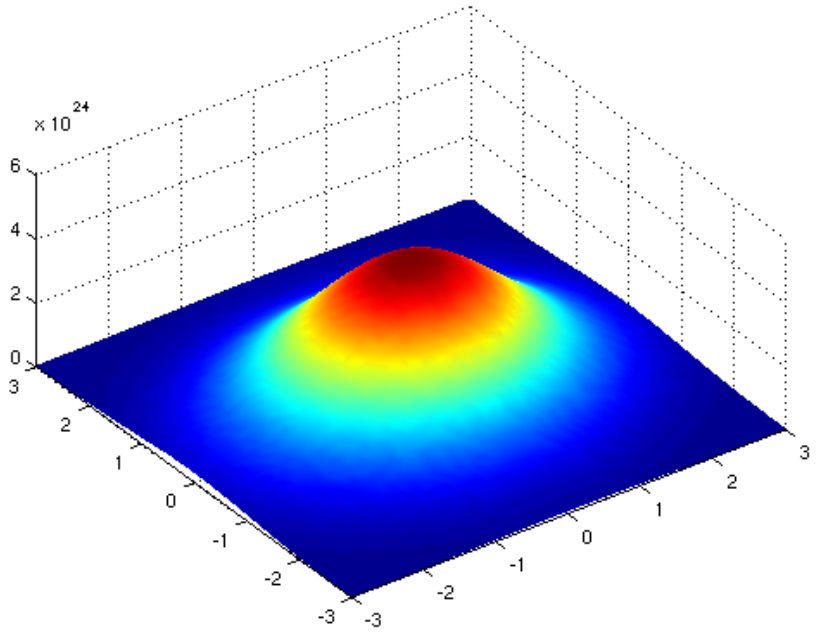
$$\hat{p}(\vec{x}) = .01386 = 1.386\%$$

$$\epsilon = .01 \Rightarrow \hat{p}(\vec{x}) > \epsilon$$

$\vec{x}$ is not anomalous

Model 2 – Multivariate Gaussian

- Instead of creating one model for each attribute, use a single model that encapsulates all data. Train a multivariate Gaussian to learn what all attributes *should* look like (if they follow a normal distribution)



$$\vec{\mu}_{MLE} = \frac{1}{n}\sum_{i=0}^{m}\vec{x}_i$$

$$\Sigma_{i,j} = \frac{\sum_{k=0}^{m}(D_{k,i}-\mu_{MLE\,i})(D_{k,j}-\mu_{MLE\,j})}{n-1}$$

$$where\ 0 \le i \le n,\ 0 \le j \le n$$

Model 2 (continued)

- Sample the Gaussian by passing in the entire instance vector *x* to get an estimate for p(x), p^(x)
- Compare p^(x) to some ε and if p^(x) < ε classify *x* as anomalous

$$\vec{x} = \langle 5.0, 12.1, -6.9, 16.0 \rangle$$
$$\downarrow$$
$$G$$
$$\downarrow$$
$$\hat{p}(\vec{x}) = .000169$$

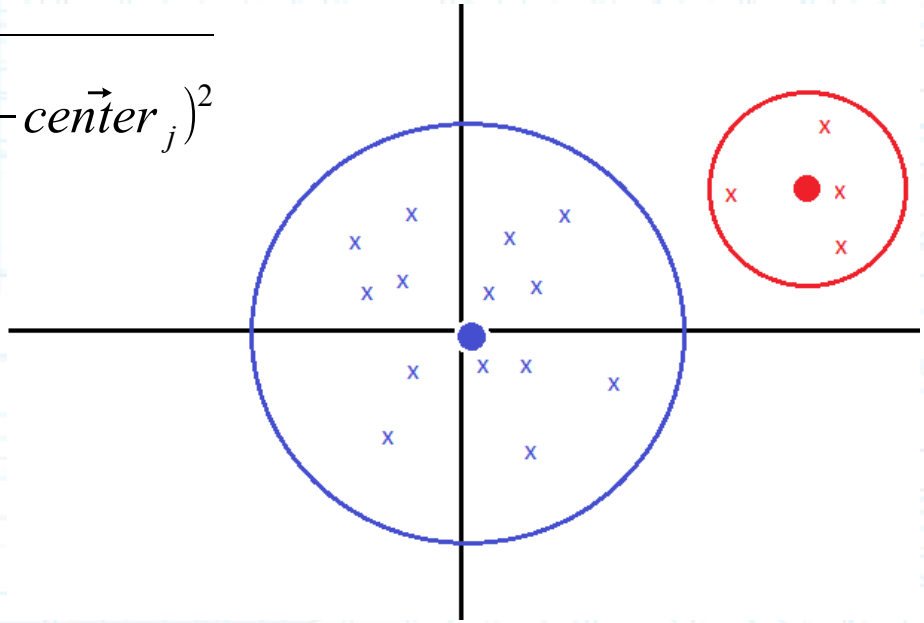
$$\epsilon = .0015 \Rightarrow \hat{p}(\vec{x}) < \epsilon$$

$\vec{x}$ is anomalous!

Model 3 – Array of KMeans

- Train an array of KMeans classifiers to group the values for each attribute into two clusters, normal and anomalous
  - Compute cluster centers for each attribute and then assign all values for an attribute to the closest cluster center

$$assignment(\vec{x}) = \underset{\{centers\}}{\operatorname{argmin}} \sqrt{\sum_{j=0}^{n} (\vec{x}_j - \vec{center}_j)^2}$$



Model 3 (continued)

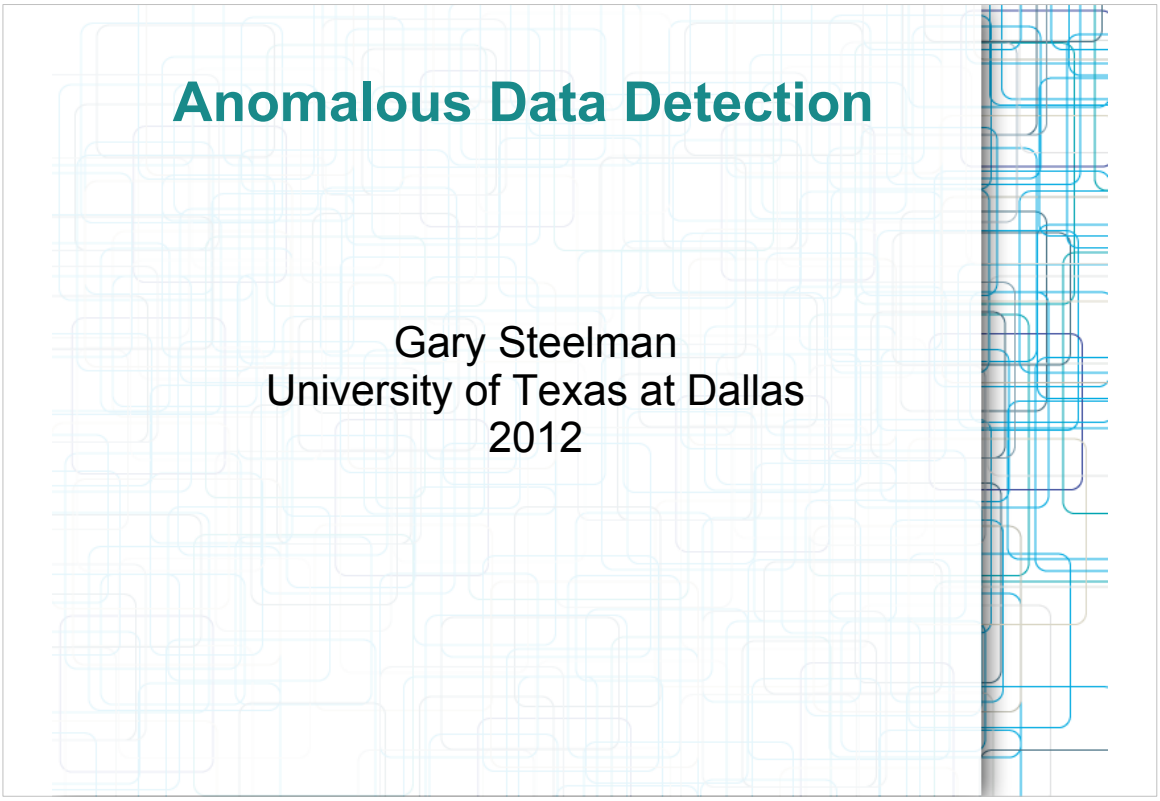
- Sample each KMeans by passing in the j[th] value from *x* and getting back true (anomalous) or false (normal) value based on distance to centers
- Logical OR together the passed back values
- If any of the KMeans says true, classify *x* as anomalous

$$\vec{x} = \langle 5.0, 12.0 \rangle$$
$$\downarrow \quad \downarrow$$
$$\begin{bmatrix} K_{normal\ center} \\ K_{anom\ center} \end{bmatrix} = \begin{bmatrix} 6.5 \\ 3.0 \end{bmatrix} \begin{bmatrix} 1.5 \\ 3.0 \end{bmatrix}$$
$$\downarrow \quad \downarrow$$
$$R = F \ \vee \ T$$
$$\downarrow$$
$$R = T$$

$\vec{x}$ is anomalous!

Conclusions

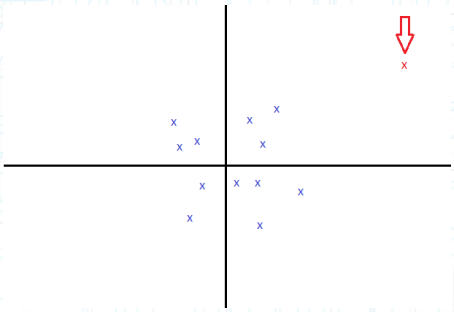
- Given a set of data instances we can model the behavior/pattern of the data using a variety of methods
- Univariate Gaussian, Multivariate Gaussian, and KMeans
- Some models faster than others, some more accurate than others, some don't work in certain situations



Introduce self, state name and purpose of presentation.

Problem Definition

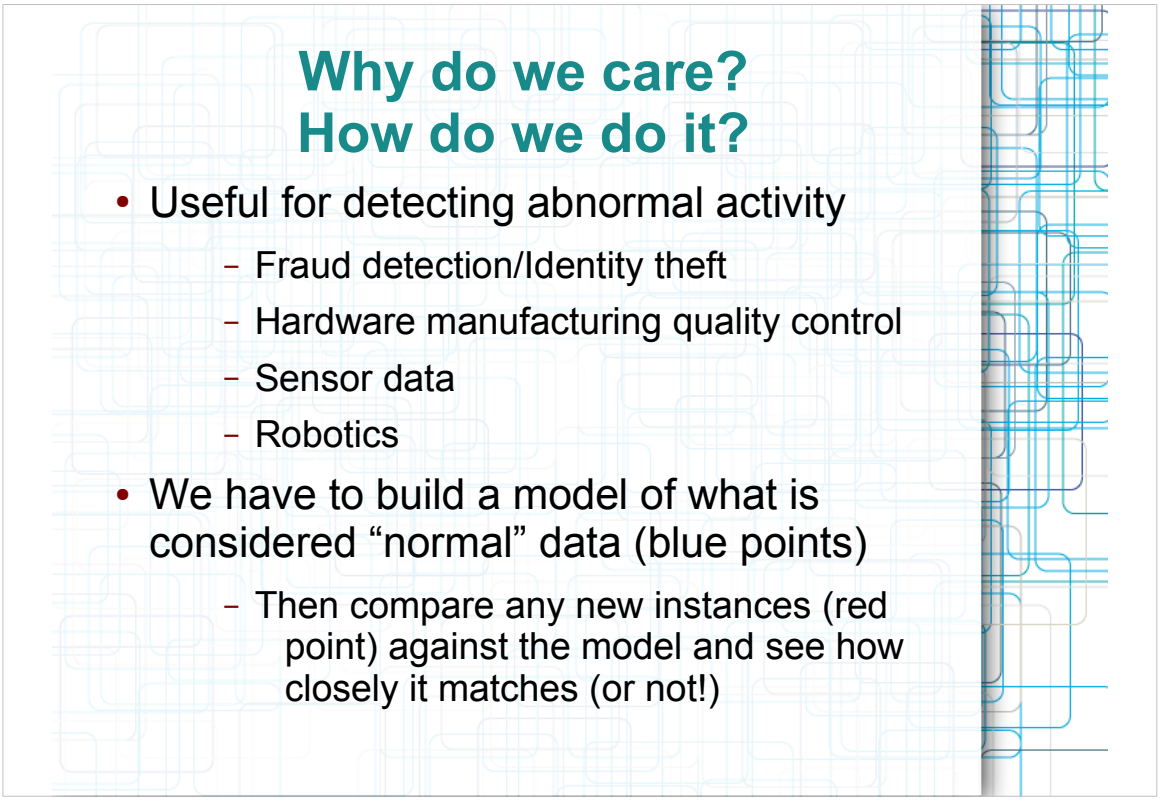
- Given a data matrix of m instances with n attributes per instance
  - Which instances aren't what we would call "normal"? Which ones are outliers?



An instance = a single row of observed values from the data.

An attribute = one dimension of an instance. 2D points have 2 attributes (x, y). Attribute values are the actual values of the attributes (2.5, 5.0).

Clearly the red instance is far away from where the majority of the instances are. This means that the red instance is anomalous.



The listed activities can all be caught with anomalous data detection. Fraud / identity theft can ruin a person's life or a business' future. Hardware manufacturing defects can cause errors in airplane engines and cause deaths. Outlier sensor readings can trigger alarms that shouldn't be triggered. Robotics can try to not overreact to outlier data, which would cause the robot to break itself or think the world is exploding around it.

A model is a way of compactly representing a large amount of data.

A model of “normal” data roughly reflects some form of the average of “normal” data.

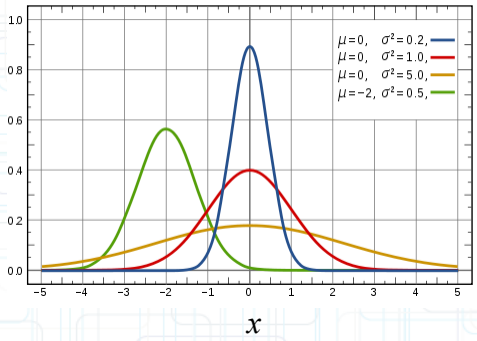
We create a model because storing all the data and using it at run-time is very slow and memory intensive.

Model 1 – Univariate Gaussians

- Train a univariate Gaussian for each attribute of the data, to learn what each attribute *should* look like (if it follows a normal distribution)

$$\mu_{MLE} = \frac{1}{n}\sum_{i=0}^{n} x_i \qquad \sigma^2_{MLE} = \frac{1}{n}\sum_{i=0}^{n} (x_i - \mu_{MLE})^2$$

$p(x|\mu_{MLE}, \sigma^2_{MLE})$



A univariate Gaussian can tell us about the distribution of a variable – that is, given a random variable X, how likely it is to take on a given value.

A normal distribution is shown – blue curve shows x is very likely to equal 0 and very unlikely to = 5.

The mean μ of the data is just the average of all the values. This the tallest point on the graph, meaning it is the mostly likely value for X to take on. Makes sense because it is right in the middle of all the observed values.

The variance σ^2 of the data represents how varied the values the variable can take are. High variance means that there's a wide range of possibilities for the value to take on and the graph would be a wider curve. Low variance means the graph is more skinny.

Model 1 (continued)

- Sample each of the Gaussians by passing in the j[th] value from *x* and get back the j[th] Gaussian's probability $p_j(x_j)$
- Multiply all the different $p_j(x_j)$ together to get an estimate for p(x), p^(x)
- Compare p^(x) to some ε and if p^(x) < ε classify *x* as anomalous

$$\vec{x} = \langle 5.0, 12.1, -6.9, 16.0 \rangle$$
$$\downarrow \quad \downarrow \quad \downarrow \quad \downarrow$$
$$\vec{G} = [g_1, \ g_2, \ g_3, \ g_4]$$
$$\downarrow \quad \downarrow \quad \downarrow \quad \downarrow$$
$$\{p_j(x_j)\} = \underbrace{\{.3, \quad .7, \quad .12, \quad .55\}}_{\text{multiply together}}$$

$$\hat{p}(\vec{x}) = .01386 = 1.386\%$$

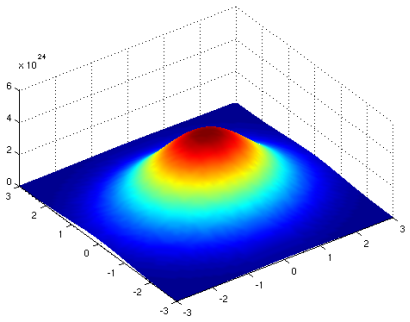
$$\epsilon = .01 \Rightarrow \hat{p}(\vec{x}) > \epsilon$$

$\vec{x}$ is not anomalous

Create one univariate Gaussian for each attribute of *x*.
Send each attribute in *x* to the corresponding Gaussian.
Multiply the results together to obtain an approximation for p(x) across all attributes.
Calculate ε as some approximate probability barrier. If the approximated p(x) is below the ε probability, it means that there is just too low of a chance of seeing *x* for it to be normal. So then classify *x* as anomalous.
An example execution is shown on the right side.

Model 2 – Multivariate Gaussian

- Instead of creating one model for each attribute, use a single model that encapsulates all data. Train a multivariate Gaussian to learn what all attributes *should* look like (if they follow a normal distribution)



$$\vec{\mu}_{MLE} = \frac{1}{n}\sum_{i=0}^{m}\vec{x}_i$$

$$\Sigma_{i,j} = \frac{\sum_{k=0}^{m}(D_{k,i}-\mu_{MLE\,i})(D_{k,j}-\mu_{MLE\,j})}{n-1}$$

$$where\ 0 \le i \le n,\ 0 \le j \le n$$

A multivariate Gaussian can tell us about the distribution of a set variables together – that is, given a random vector variable X, how likely it is to take on a vector of assignments.

A normal distribution of two variables x and y is shown. The height of the surface represents how likely the data will be equal to (x,y) as a pair.

The means vector μ of the data is just the average of all the vectors observed. This the tallest point on the graph, meaning it is the mostly likely value for (x,y) to take on. Makes sense because it is right in the middle of all the observed values.

The covariance matrix Σ of the data represents how varied the values the variables can take together are. High variance means that there's a wide range of possibilities for the value to take on and the graph would be a wider surface. Low variance means the graph is more skinny.

Model 2 (continued)

- Sample the Gaussian by passing in the entire instance vector *x* to get an estimate for p(x), p^(x)
- Compare p^(x) to some ε and if p^(x) < ε classify *x* as anomalous

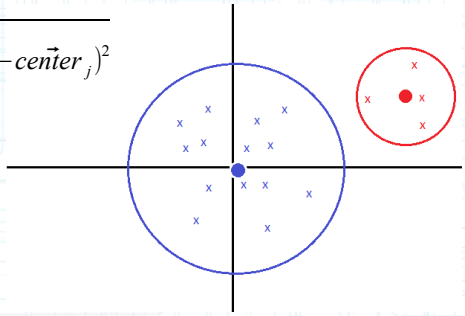
$$\vec{x} = \langle 5.0, 12.1, -6.9, 16.0 \rangle$$
$$\downarrow$$
$$G$$
$$\downarrow$$
$$\hat{p}(\vec{x}) = .000169$$
$$\epsilon = .0015 \Rightarrow \hat{p}(\vec{x}) < \epsilon$$
$$\vec{x} \text{ is anomalous!}$$

Create one multivariate Gaussian.
Send a data instance vector *x* to the Gaussian.
Obtain an approximated p(x) from the Gaussian.
Calculate ε as some approximate probability barrier. If the approximated p(x) is below the ε probability, it means that there is just too low of a chance of seeing the instance for it to be normal. So then classify *x* as anomalous.
An example execution is shown on the right side.

Model 3 – Array of KMeans

- Train an array of KMeans classifiers to group the values for each attribute into two clusters, normal and anomalous
  - Compute cluster centers for each attribute and then assign all values for an attribute to the closest cluster center

$$assignment(\vec{x}) = \underset{\{centers\}}{\operatorname{argmin}} \sqrt{\sum_{j=0}^{n} (\vec{x}_j - \vec{center}_j)^2}$$



Kmeans classifiers are constructed iteratively rather than immediately as with Gaussians.
For this we generate 2 random points to begin with.
Then we repeat this process:
1) Assign all data instances to their closest center (via computing the distances)
2) Update the centers to the average value of all their assigned instances
Until the point assignments do not change.
This leaves us with the large red and large blue dots as the centers and something like the red and blue circles representing the reach of each of the centers. Any points within the circles are assigned to the center of the circle.
This allows us to cluster the data instances into groups, and we can see which group is the anomalous group and which is the normal group.

Model 3 (continued)

- Sample each KMeans by passing in the j[th] value from *x* and getting back true (anomalous) or false (normal) value based on distance to centers
- Logical OR together the passed back values
- If any of the KMeans says true, classify *x* as anomalous

$$\vec{x} = \langle 5.0, 12.0 \rangle$$
$$\downarrow \quad \downarrow$$
$$\begin{bmatrix} K_{normal\ center} \\ K_{anom\ center} \end{bmatrix} = \begin{bmatrix} 6.5 \\ 3.0 \end{bmatrix} \begin{bmatrix} 1.5 \\ 3.0 \end{bmatrix}$$
$$\downarrow \quad \downarrow$$
$$R = F \ \vee \ T$$
$$\downarrow$$
$$R = T$$

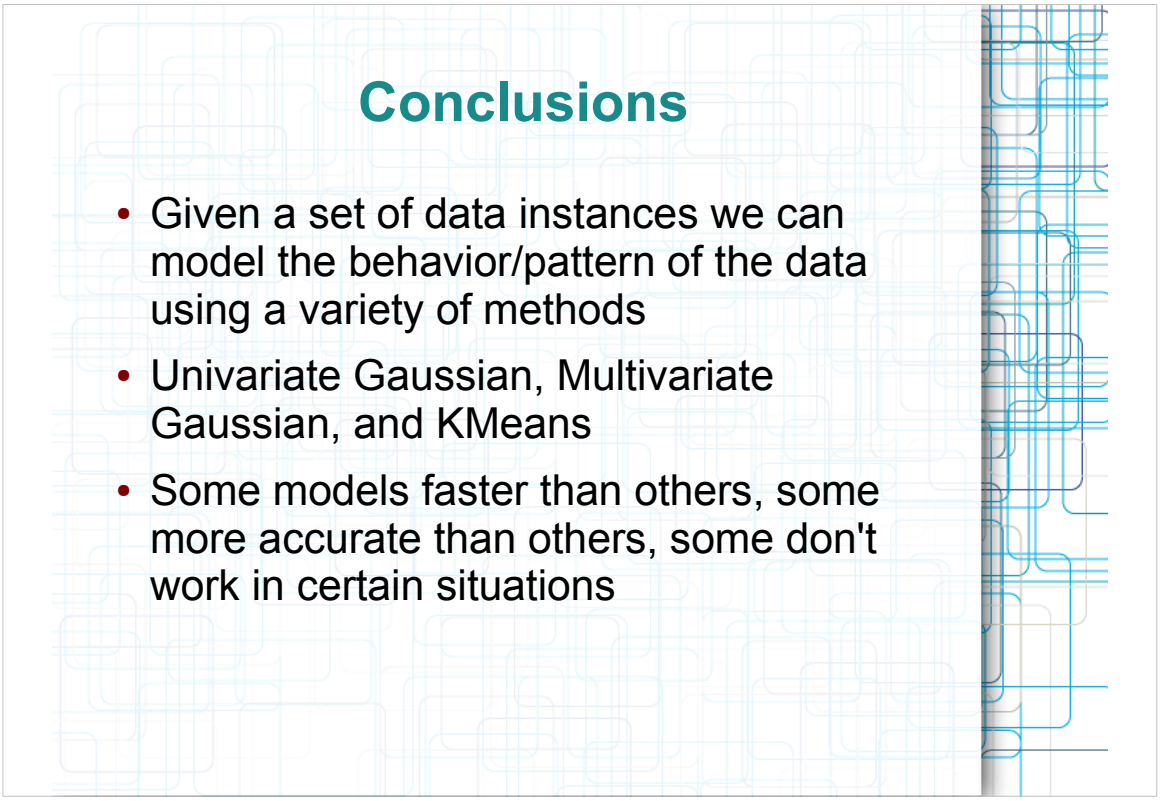
$\vec{x}$ is anomalous!

When a new data instance is received, compute the distance from each of its attributes to the corresponding KMeans centers.

If any of the attributes are closest to the anomalous center, that classifier return true, otherwise it returns false.

In the end, if any classifier said the attribute value was anomalous, then classify *x* as anomalous.

An example execution is shown on the right side.



Reiterate the idea of anomalous data detection and how it can be useful.

Reiterate the general idea: construct a model of normal data and then compare new instances against the model. If the model says the new instance is too different, flag the new instance as anomalous.

Reiterate the three methods covered.