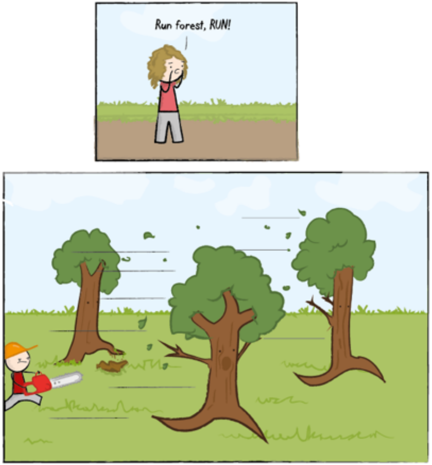
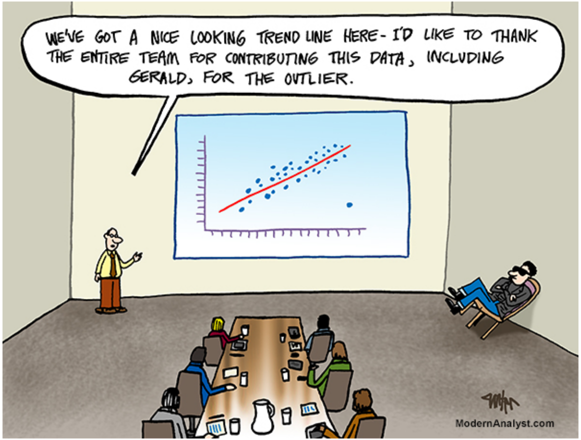
CSCI 4022 Fall 2021
UV Wrapup, Outliers Intro



nicartoons.com © Nick Rhyder


WE'VE GOT A NICE LOOKING TREND LINE HERE- I'D LIKE TO THANK THE ENTIRE TEAM FOR CONTRIBUTING THIS DATA, INCLUDING GERALD, FOR THE OUTLIER.
ModernAnalyst.com

Announcements and Reminders

M check-in

W

F exam

M HW9

W

final : proj stuff

1. HW8 tonight; extra OH per deadline-day usual.
2. Project check-in after break. All you need is pre-processing and preliminary results! I'll give feedback on the proposals during break, but please make Piazza posts if you want any advice on things you can do with your data sets.
3. **Last day** of theory lecture is today, with associated notebooks on Friday. We have an exam review, some practice/application lectures, and some project presentations for the 5 lecture days after break. Home stretch!!

UV Decomposition: Big Picture

Given an $n \times m$ utility or data matrix M, with some unknown/blank entries...

1. Preprocess M: e.g. normalize
2. Initialize: decompose with *many* different initial U and V because local minima lurk
3. Iterate: do multiple training epochs, looping over all elements in U and V each time
4. Converge: don't expect RMSE to reach 0, but once its epoch-to-epoch change is $<$ tolerance we can break. We can also stop once no component improves RMSE by more than some threshold (e.g. a max noem)
5. Would another choice of d work better? Try it, and recall **elbow** plots of d as the x-axis and $RMSE$ as the y-axis. More d is always (expected to be) less $RMSE$, but with diminishing returns.

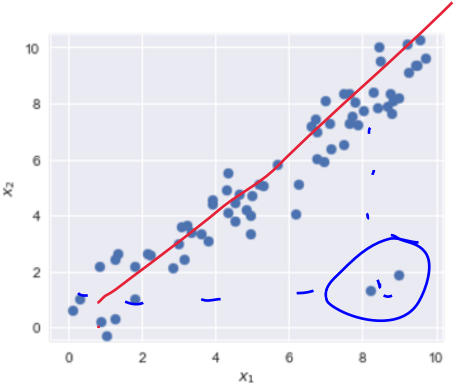
Outliers: Z-scores and Random Forests

Definition: An outlier is an observation which deviates so much from other observations as to arouse suspicion it was generated by a different mechanism. (from Hawkin, 1980)

Building a predictive model that includes outliers means it wasn't trained on data that came from the process we want to make predictions about

Example: Consider the linear model $Y = \alpha + \beta x + \varepsilon$, where $\varepsilon \sim N(0\sigma^2)$.

Does it look like the data set here was generated by this process?



Example: Consider a utility matrix, representing 100 users' ratings of 100,000 movies. S'pose a user has rated every single movie as a 5 (out of 5 stars).

Overview of Outlier Methods: Many Options

1. Z-score/extreme value analysis (parametric) Also: Mahalanobis distance extension
2. Probabilistic/statistical modeling (parametric)
3. Linear regression models, PCA (parametric)
4. Proximity/density-based models (non-parametric)
5. Information theory models
6. Higher-dimensional outlier detection methods

Today: Z-score + Mahalanobis distance (parametric, simple, tough with higher dim'l data)
Next time: Isolation forests (iForests) (non-parametric, more sophisticated, works well with high dim'l data, parallelizes readily)

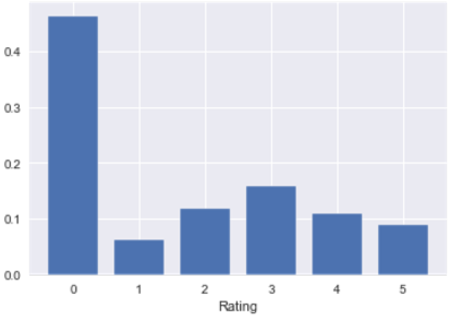
Z-scores

- Relies on assumption of an underlying normal distribution of your data
- If the data aren't (approx.) normal, a transformation to normal might be an important preliminary step.

Some simple transformations:

1. Removing extreme values.
2. Logarithm, Exponentiate.

Example: Consider a utility matrix users' movie ratings, where 0 denotes a user has not yet seen/rated a particular movie.



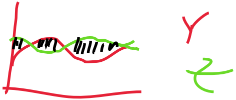
Normality

Goal: Make non-normal data look more normal. tools: some simple transformations:

1. Removing extreme values.
2. Logarithm, Exponentiate.
3. **Box-Cox**: For positive data Y, transform the data via

$$Y' = \begin{cases} \frac{Y^\lambda - 1}{\lambda} & \lambda \neq 0 \\ \ln Y & \lambda \neq 0 \end{cases}$$

where we *choose* the value of $\lambda \in [-5, 5]$ to maximize how close to normal the resulting Y' looks.



”Functional Distance”

Note: Ostensibly we need a measure for ”most normal.” While these are probability distributions and/or data sets, we can instead do a distance-between-functions on their *cdfs*: norms here tend to be of the form $d(Y, Z) = \int_\Omega |f_Y(x) - f_Z(x)| dx$; note that squaring or other powers inside the integral are also common, as in Pythagorean/Euclidean distance norms.

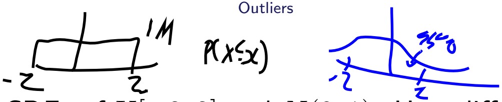
For comparing cdfs, we know the anti-derivatives of f and g (the two cdfs F and G!), so we often use an L^1 type difference. The statistic

$$d(F, G) = \max_x |F(x) - G(x)|$$

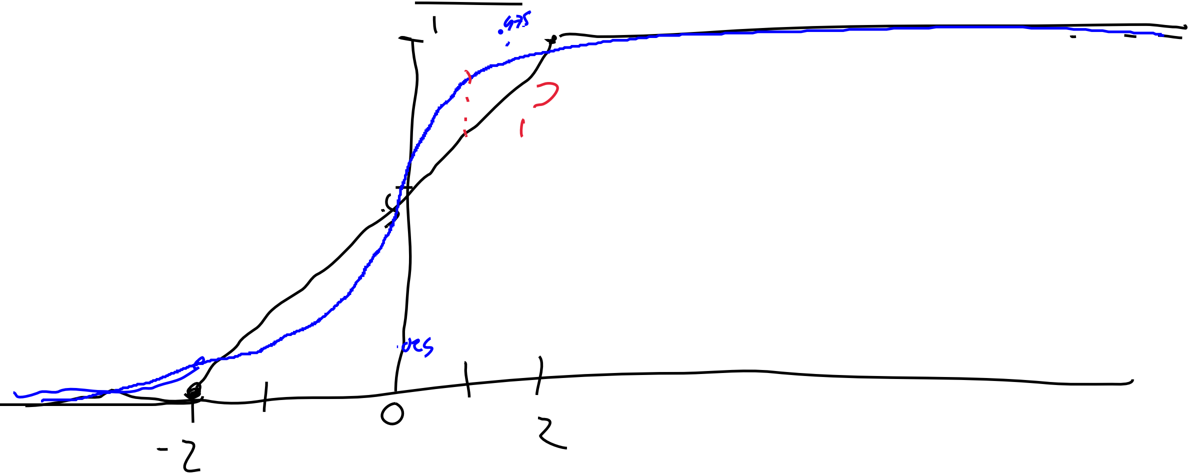
is actually the basis for the most common ”are these distributions equal” hypothesis test, called the *Kolmogorov–Smirnov* test.

There are other goodness-of-fit tests, including our good ol’ standby maximum likelihood, where we choose $\lambda \in [-5, 5]$ to make $P(Y' | Y' \sim N(\bar{Y'}, s^2_{Y'}))$ as large as possible! This one is the usual with Box-Cox.

Example



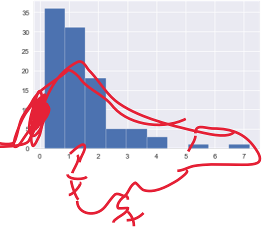
Example: Sketch the CDFs of $U[-2,2]$ and $N(0,1)$. How different are they?



Box Cox in Action

Goal: Make non-normal data look more normal.

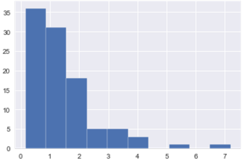
Example: Consider the data at top-left. What do different values of λ do?



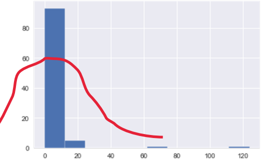
Box Cox in Action

Goal: Make non-normal data look more normal.

Example: Consider the data at top-left. What do different values of λ do?



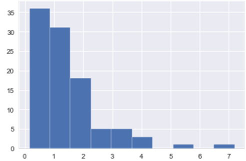
First try: $\lambda = 3$



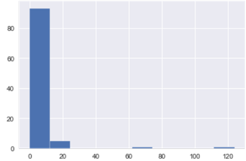
Box Cox in Action

Goal: Make non-normal data look more normal.

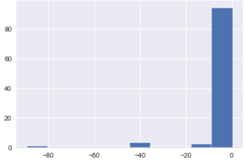
Example: Consider the data at top-left. What do different values of λ do?



First try: $\lambda = 3$



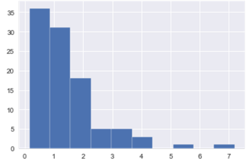
Second try: $\lambda = -3$



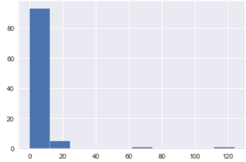
Box Cox in Action

Goal: Make non-normal data look more normal.

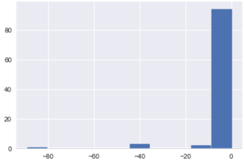
Example: Consider the data at top-left. What do different values of λ do?



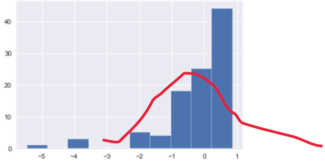
First try: $\lambda = 3$



Second try: $\lambda = -3$



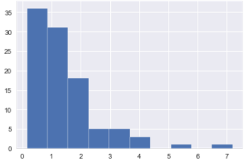
Third try: $\lambda = -1$



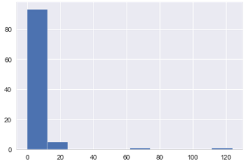
Box Cox in Action

Goal: Make non-normal data look more normal.

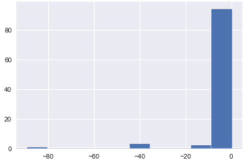
Example: Consider the data at top-left. What do different values of λ do?



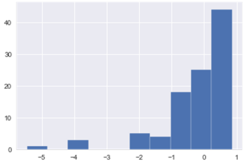
First try: $\lambda = 3$



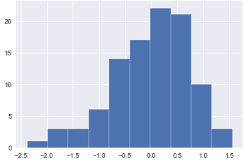
Second try: $\lambda = -3$



Third try: $\lambda = -1$



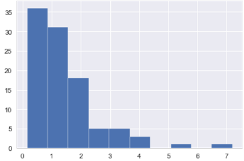
Fourth try: $\lambda = -0.25$



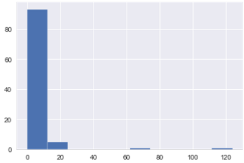
Box Cox in Action

Goal: Make non-normal data look more normal.

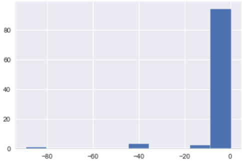
Example: Consider the data at top-left. What do different values of λ do?



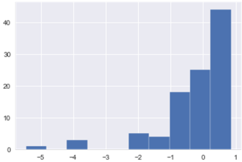
First try: $\lambda = 3$



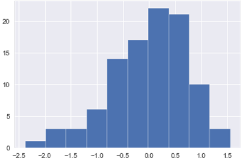
Second try: $\lambda = -3$



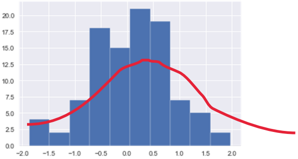
Third try: $\lambda = -1$



Fourth try: $\lambda = -0.25$



Last try: $\lambda = 0; Y' = \ln Y$

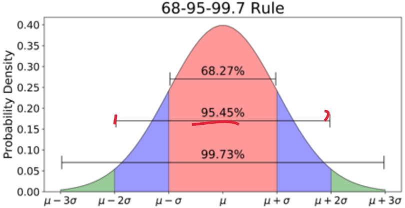


Z-Scores

Each data point can then be converted into a **Z-score** via the usual standardization transform:

1. X = original data (or transformed-to-normal data)
2. $\bar{X}$= mean of data X
3. s = sample standard deviation of X
4. Z = # standard deviations that a data point is away from the mean

$$\boxed{Z = \frac{X - \bar{X}}{s}}$$



Classification: Label data points as outliers if their (absolute) Z-score is greater than some threshold. Typical choices: Z = 2.5, 3 or 3.5, motivated by the "68-95-99.7 rule"

Mahalanobis distance

Con of Z-Scores:
Z-score becomes difficult in higher dimensions: we can compute how far a given datum $\vec{x}$ is from the average datum $\bar{X}$, but now each of these are d-dimensional.

We definitely *don't* want to just take a usual distance $x - \bar{x}$ and hit it with a norm (like L_2 distance). Why not? The data in x might be *correlated:* some dimensions might be basically redundant, so being an outlier in one would probabilistically make you also an outlier in other dimensions.

So what do we do?

Mahalanobis distance is a measure of how far points are from *the general structure of the data*.

Recall: (from GMMs) that the higher-dimensional normal distribution functions like an *ellipse* of points (or more like a rugby-ball shaped point cloud)

Mahalanobis distance

The resulting distance is a multivariate form of Z-scores. **Definition**: The *Mahalanobis* distances of a datum (row vector) $\vec{x}$ is given by:

like: var: $\sum(x-\bar{x})^2$

$$D_M(\vec{x}) = \sqrt{\underbrace{(\vec{x}-\bar{x})^T}_{x-\bar{x}} C^{-1} \underbrace{(\vec{x}-\bar{x})}_{x-\bar{x}}}$$

where:

1. $\bar{x}$ is the d-dimensional mean of the data. Component i is the mean of all the data along dimension i.
2. C is the sample covariance matrix, given by $C = \frac{1}{n-1}\sum_{i=1}^{n}(x_i-\bar{x})(x_j-\bar{x})^T$ Recall that the covariance between columns i and j is a measure of whether they shrink and grow in unison: does knowledge of one help (linearly) predict the other.

$C_{2,3}$: covariance of column 2 of x w/ col 3 of X

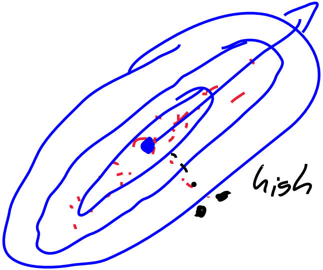
Mahalanobis distance

$$D_M(\vec{x}) = \sqrt{(\vec{x} - \bar{x})^T \, C^{-1} \, (\vec{x} - \bar{x})}$$

Note: This is *very closely related* to PCA and SVD! The pieces of the covariance matrix $\underline{(x_i - \bar{x})\,(x_j - \bar{x})^T}$ should look a lot like the $M^T M$ matrix we compute eigen-pairs from. In fact, when we standardize the data first, the matrix we're finding eigenpairs of *is the covariance*. This is why those methods are often explained in the **units** of variance: amount of variance in the data explained.

For the multivariate normal, the pdf of f depends solely on the Mahalanobis distance of the datum. Visually, points of equivalent Mahalanobis distance along a data set are on shared elliptical contours, where the ellipse is centered at $\bar{x}$ and oriented along its covariance.

Mahalanobis distance visualized



low probability from Normal ellipse

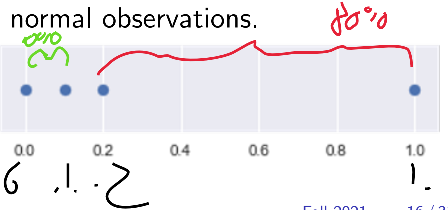
high Mahalanobis distance

Isolation Forests (iForest)

1. Non-parametric
2. Can handle large data and high-dimensional data (readily parallelizes)
3. Conceptually appealing because it locates outliers and doesn't waste time identifying normal data points
4. Used frequently in fraud detection
5. Relatively recent idea (OG: Liu et al, 2012)

Built on the idea that it is easier to isolate an anomalous observation because fewer conditions are needed in order to separate those data points from the normal observations.

Example: Using a series of conjunctions/disjunctions, isolate (a) the data point at 1, and (b) the data point at 0.1.



Creating an iForest

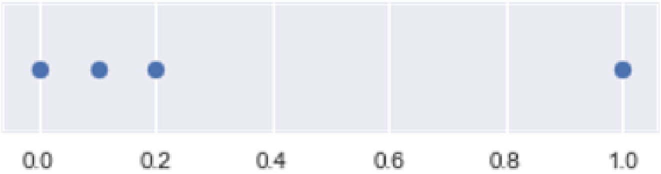
We create an iForest by creating lots of random trees: *isolation trees* (iTrees)

- Each tree has access to some sub-sample of the data, and is a full binary tree.
- Recursively create branches through the following process:
  1. Randomly select a feature/dimension, q
  2. Randomly select a split point, $p \in [\min(q), \max(q)]$ (within the limits of q's values in the sub-sample of data)
  3. The left branch (child node) gets the data points with $q < p$
  4. The right branch (child node) gets the data points with $q \geq p$
- Continue creating branches until one of:
  1. The tree reaches a height limit L
  2. There is a single data point associated with some branch (isolated a point!)
  3. All data left have the same values

iForest intuition

Example, and Intuition: for the tree below...

1. 80% probability we will isolate the data point at 1 with the first split.
2. 10% probability we will isolate the data point at 0 with the first split.
3. The two "internal" points would require 2 splits to isolate.



To build up an "anomaly score" for our data points, we want it to incorporate these ideas:
How likely points are to be isolated on average (expected value)
How many splits were necessary to isolate the points (ev of path length)

iForest Formalism

Definition:
An *isolation tree* (iTree) is a full binary tree whose nodes are either an external node with no children (leaf) or an internal node with a test and two child nodes (T_L and T_R). A test consists of an attribute q and split value p such that the test $q < p$ divides the data into T_L and T_R.

Definition:
The *path length* $h(x)$ of a data point x in an iTree T is measured by the number of edges traversed in T starting from the root node until the traversal is terminated at an external node.

Two cases:

1. If traversal terminates because T hit the height limit L, this is an underestimate of $h(x)$, so we augment by a small amount (revisit by how much later)
2. If traversal terminates "properly", then the size of the tree at/below x is 1

Isolation Trees

To build up an “anomaly score” for our data points, we want it to incorporate these ideas:
How likely points are to be isolated on average (expected value)
How many splits were necessary to isolate the points (ev of path length)

Game plan so far:

1. Create a bunch of iTrees, and see which data points become isolated
2. And for those data points, look at $E[h(x)]$, the average # splits to isolate them
3. High values of $E[h(x)] \implies$ probably normal data points
4. Low values $\implies$ probably outliers

...but how can we tell how likely data points are to be normal/outliers? Need a relative measure

Anomaly Score

Borrow from binary search trees the average path length of unsuccessful search:

$$\underline{c(n)} = \underline{2H(n-1) - 2(n-1)/n}$$

expected depth of an unfound point in a tree of size n.

where:
– $n :=$ size of data set
– $H(k) :=$ Harmonic number $= ln(k) + \gamma$, $\gamma =$ Euler-Mascheroni constant ≈ 0.57722

And we define our **iForest anomaly score** of data point x within data sample of n points as:

$$s(\boxed{x}, n) = 2^{-\underline{E[h(x)]}/c(n)}$$

if x "easy to find"

Sanity check: we're comparing $E[h(x)]$ (how many cuts needed to isolate a point) against $c(n)$ average number of cuts needed to find *any* point.

$\rightarrow 2^{-[0/n]} \sim 1$

if $E[h(x)] > c(n)$

$2^{-\text{large \#}} \rightarrow 0.$

1. What happens as datum x looks "more normal"?
2. What happens as it becomes more extreme of an outlier?

Anomaly Score

Borrow from binary search trees the average path length of unsuccessful search:

$$c(n) = 2H(n-1) - 2(n-1)/n$$

where:
– $n :=$ size of data set
– $H(k) :=$ Harmonic number $= ln(k) + \gamma$, $\gamma =$ Euler-Mascheroni constant ≈ 0.57722

And we define our **iForest anomaly score** of data point x within data sample of n points as:

$$s(x, n) = 2^{-E[h(x)]/c(n)}$$

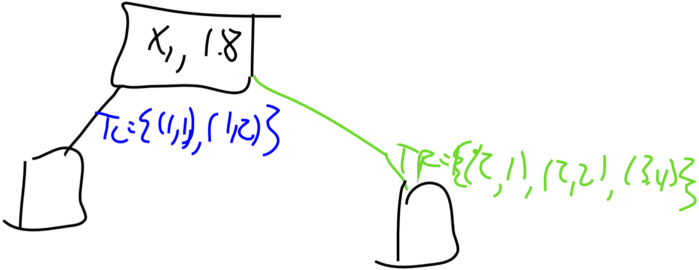
We can also use $c(n)$ to **augment** search for x in the iTree's calculation of path length! It approximates how many more splits we would expect to need beyond L to isolate x.

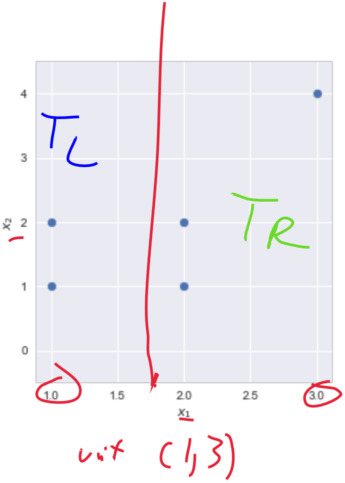
So: Given an iTree T and a data point x, if the splitting process terminates before x is isolated, we augment the path length as: $h(x) = \underset{ho = L}{h(x)} + c(n_{remaining})$.

iForest Example

Example: Build an iTree given the data shown here. Use height limit $L = 2$.

1. Pick a feature q: say, $x1$
2. Pick a split point p in $[\min(q), \max(q)]$: say, 1.8

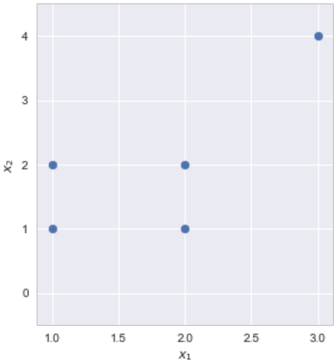




iForest Example

Example: Build an iTree given the data shown here. Use height limit $L = 2$.

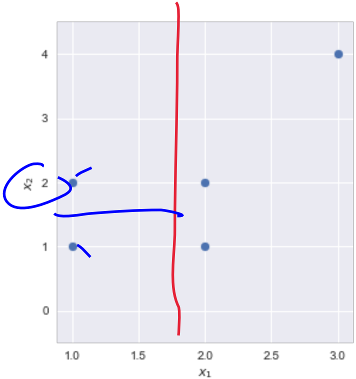
1. Pick a feature q: say, $x1$
2. Pick a split point p in $[\min(q), \max(q)]$: say, 1.8
3. $T_L = (1,1),(1,2), \quad T_R = (2,1),(2,2),(3,4)$
   So: Root node T_1 is internal node with attributes: feature $q = x_1$, split value $p = 1.8$, children T_L and T_R



iForest Example

Example: Build an iTree given the data shown here. Use height limit $L = 2$.

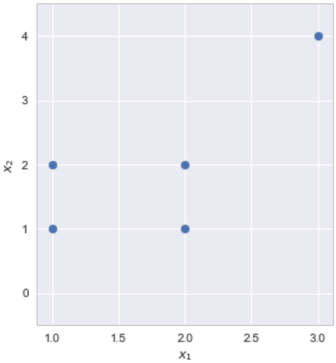
1. Pick a feature q: say, $x1$
2. Pick a split point p in $[\min(q), \max(q)]$: say, 1.8
3. $T_L = (1,1),(1,2), \quad T_R = (2,1),(2,2),(3,4)$
   So: Root node T_1 is internal node with attributes: feature $q = x_1$, split value $p = 1.8$, children T_L and T_R
   And continue: Work on left tree first. (Note: we need to do both sides! This isn't search, as we might query $h(y)$ where y is a data point in T_R).)
4. Pick a feature q: say, x_2
5. Pick a split point p : say, 1.5



iForest Example

Example: Build an iTree given the data shown here. Use height limit $L = 2$.

1. Pick a feature q: say, $x1$
2. Pick a split point p in $[\min(q), \max(q)]$: say, 1.8
3. $T_L = (1,1),(1,2), \quad T_R = (2,1),(2,2),(3,4)$
   So: Root node T_1 is internal node with attributes: feature $q = x_1$, split value $p = 1.8$, children T_L and T_R
   And continue: Work on left tree first. (Note: we need to do both sides! This isn't search, as we might query $h(y)$ where y is a data point in T_R).)
4. Pick a feature q: say, x_2
5. Pick a split point p : say, 1.5
6. $T_L = (1,1), \quad T_R = (1,2)$ Now: left child is internal node with attributes: $q = x_2$, $p = 1.5$, children T_L and T_R

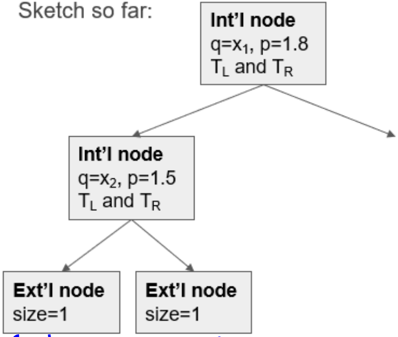


iForest Example

Example: Build an iTree given the data shown here. Use height limit $L = 2$.

7. Since the left child's children T_L and T_R each consist of a single data point, they are both external nodes with size $= 1$.

Sketch so far:



Int'l node
$q = x_1$, $p = 1.8$
T_L and T_R

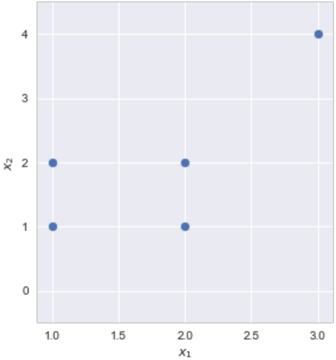
Int'l node
$q = x_2$, $p = 1.5$
T_L and T_R

Ext'l node
size=1

Ext'l node
size=1

(1,1)

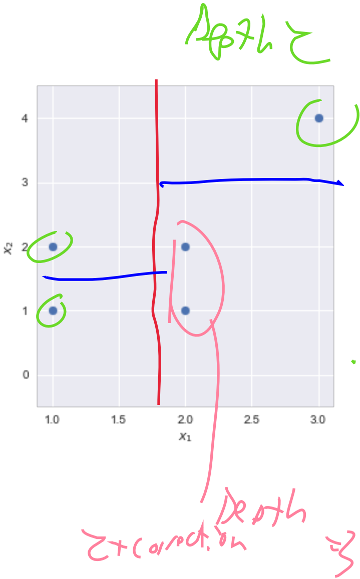
(1,2)



iForest Example

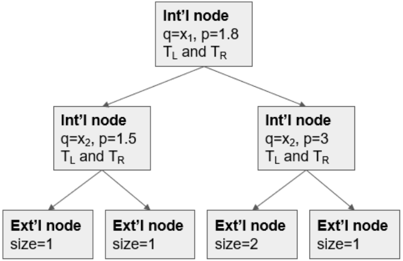
Example: Build an iTree given the data shown here. Use height limit $L = 2$.

7. Since the left child's children T_L and T_R each consist of a single data point, they are both external nodes with size $= 1$. Work on right tree now: (need to do both! This isn't search)
8. Pick a feature q: say, x_2
9. Pick a split point p: say 3
10. $T_L = (2, 1), (2, 2)$, $T_R = (3, 4)$, or right child is internal node with: $q = x_2, p = 3$, children T_L and T_R
11. We have reached the height limit $L = 2$, so both T_L and T_R are external nodes. Final sizes: T_L has size $= 2$, T_R has size $= 1$



iForest Example

Example: Build an iTree given the data shown here. Use height limit $L = 2$. Final depiction:



Int'l node
q=x_1, p=1.8
T_L and T_R

Int'l node
q=x_2, p=1.5
T_L and T_R

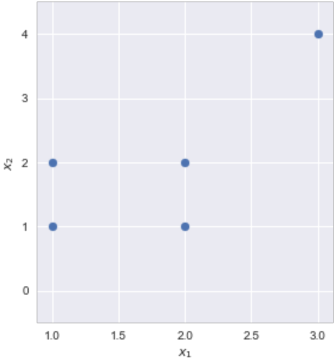
Int'l node
q=x_2, p=3
T_L and T_R

Ext'l node
size=1

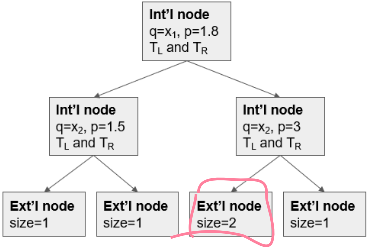
Ext'l node
size=1

Ext'l node
size=2

Ext'l node
size=1



iForest Computations



Int'l node
$q=x_1$, $p=1.8$
T_L and T_R

Int'l node
$q=x_2$, $p=1.5$
T_L and T_R

Int'l node
$q=x_2$, $p=3$
T_L and T_R

Ext'l node
size=1

Ext'l node
size=1

Ext'l node
size=2

Ext'l node
size=1

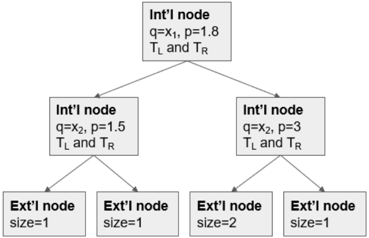
Example: Compute $s((2,2),5)$ and $s((3,4),5)$ using $s(x,n) = 2^{-E[h(x)]/c(n)}$. Recall that this compares the average height h to the expected $c(n) = 2(\ln(n-1) + 0.57722) - 2(n-1)/n$ with $n = 5$, or $c(5) = \underline{2.327.}$

Solution: With only one trees there's no *average* for $E[h(x)]$, so we just take the height where we found things. This gives:

1. For (2,2): go *right* at the first $q = x_1, p = 1.8$, since $2 > 1.8$. Increment LEN by 1. (Now 1)
2. Then go *left* at the second split: $q = x_2, p = 3$, since $2 < 3$. Increment LEN by 1. (Now 2)
3. **Stop** since we're at an external node. This is an object of size 2, so increment LEN by an additional $c(2) = 1$.

Result: $h((2,2)) = 3$.

iForest Computations



Example: Compute $s((2,2),5)$ and $s((3,4),5)$ using $s(x,n) = 2^{-E[h(x)]/c(n)}$. Recall that this compares the average height h to the expected $c(n) = 2(\ln(n-1) + 0.57722) - 2(n-1)/n$ with $n = 5$, or $c(5) = 2.327$.

Solution: With only one trees there's no *average* for $E[h(x)]$, so we just take the height where we found things. This gives:

1. For (3,4): go *right* at the first $q = x_1, p = 1.8$, since $3 > 1.8$. Increment LEN by 1. (Now 1)
2. Then go *right* at the second split: $q = x_2, p = 3$, since $4 > 3$. Increment LEN by 1. (Now 2)
3. **Stop** since we're at an external node. This is an object of size 1, so we're done!

Result: $h((3,4)) = 2$.

iForest Computations

We found that $h((3,4)) = 2$, $h((2,2)) = 3$. Finally we'd make similarity scores using $s(x,n) = 2^{-E[h(x)]/c(n)}$, so:

$c(n)$

$2^{-\left(\frac{\text{Avg depth}}{\text{n iters}}\right)} / \text{avg search depth}$

$$s((2,2),5) = 2^{-3/2.327}$$
$$s((3,4),5) = 2^{-2/2.327} = \underline{0.551}$$

Result: (3,4) has a higher anomaly score than (2,2).
Of course, out in the wild we would create many trees – our lone tree is a pretty sorry forest!

iForest implementation notes

Guidelines:

- If some instances of s are very close to 1, then they are almost certainly anomalies
- If instances have s much smaller than 0.5, they can safely be assumed to be normal data points
- If all instances have s near 0.5, then the entire sample does not have any clear anomalies
- Pick height limit $L = \underline{log_2(n)}$ (an average tree height)
  Works great because we don't care about long paths!
  So stop building the tree once it's taller than average.
- Path length typically converge well with $\#$ trees around 100
- **Efficiency**: can sub-sample the full data set to build forest.
  Makes iForests easily parallelizable
  $n = 256$ usually enough, per literature

iForest Results

An added twist to an iForest may "chop" the data using lines or planes with slopes. What differs?

1. pick a random slope/coefficients and then define the minimal and maximal intercepts that would make it pass through data values.
2. Then pick a random intercept within that range, proceed as before
3. **Consider:** what's the upside of this?

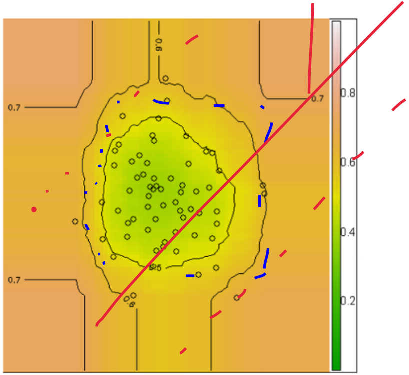


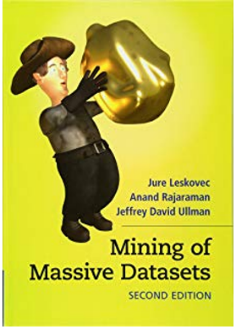
Figure 3. Anomaly score contour of iForest for a Gaussian distribution of sixty-four points. Contour lines for $s = 0.5, 0.6, 0.7$ are illustrated. Potential anomalies can be identified as points where $s \geq 0.6$.

from Liu, et. al (2012).

Acknowledgments

Next time: Notebook day... then an Exam Review!

Some material is adapted/adopted from Mining of Massive Data Sets, by Jure Leskovec, Anand Rajaraman, Jeff Ullman (Stanford University) `http://www.mmds.org`



Special thanks to Tony Wong for sharing his original adaptation and adoption of slide material.