Monitoring Massive Network Traffic using Bayesian Inference

David Rodriguez
Cisco Systems, Inc.
Senior Research Engineer

November 2, 2018

Team

Dhia Mahjoub	Scott Sitar
Gilad Ranier	Matt Foley
Irwin Fule-Ver	Skyler Hawthorne
Thomas Matthew	

Table: We are the research-engineering team implementing algorithms and maintaining the DNS threat intelligence to the Cisco Umbrella product.

Table of contents

Plan

Observe Signals

Generate Signals

Fit Fast

Fit Many

Fit Over Time

Measure Risk

Signals of Threats

Phishing

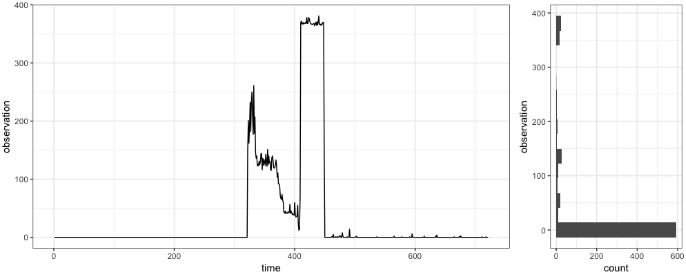


Figure: `071867.vps-10.com`

Heuristic Fallout

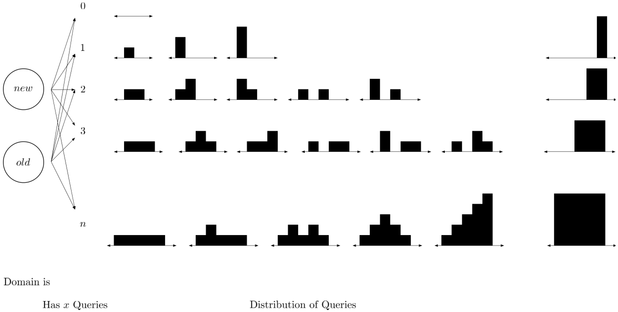


Figure: The combinatorial explosion of query patterns highlight patterns with *zero* queries. Also, notice, some patterns are similar if permuted.

Here's the Problem

Detecting anomalies associated with threats are hard to determine if[1]:

- the domain has previous query volume
- there is large variations in query volume
- there are gaps between periods with query volume

[1] we could also mention there are difficulties in modeling *non-stationary* time-series

Plan

Observe Signals

Generate Signals

Fit Fast

Fit Many

Fit Over Time

Measure Risk

Be the Adversary

Question

What if roles were reversed? Rather than observing, you were asked to generate *malicious* traffic.

Be the Adversary

Question

What if roles were reversed? Rather than observing, you were asked to generate *malicious* traffic.

You might need some tools, but that's not a problem.

Common Discrete Distributions

Observation

If you can generate a random number then you can definitely generate any one of these:

- $Geom(p)$ - the geometric
- $Pois(\lambda)$ - the poisson
- $Bin(n, p)$ - the binomial
- $NB(n, p)$ - the negative binomial

Common Discrete Distributions [2]

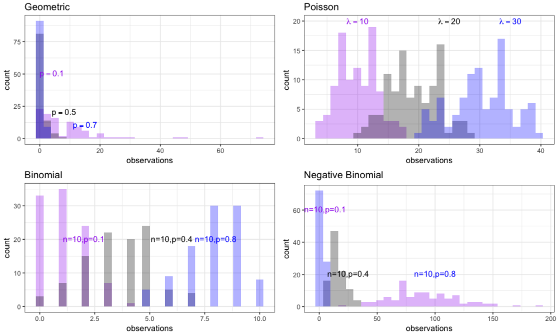


Figure: Clockwise starting top left: geometric, poisson, negative binomial, and binomial distributions. For given parameters 100 samples generated per distribution.

[2] likely not seen in the real traffic

Common Discrete Distributions [3]

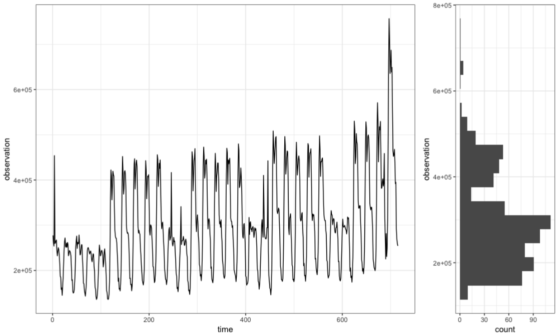


Figure: Example query volume to **jd.com** over the last 30 days is bimodal and therefore not one of the previous distributions.

[3] likely not seen in the real traffic

Mixtures of Discrete Distributions

We can *mix* distributions.[4]

Zero Inflated Distributions

$$f(x;\theta) = \psi \mathcal{I}_0 + (1-\psi) g(x;\theta) \tag{1}$$

where $\mathcal{I}_0$ is an indicator variable at zero, $\psi \in [0,1]$, and $g(x;\theta)$ is any discrete distribution from the previous slide.

[4] be careful to maintain the properties of a probability distribution

Spam Filtering as Mixtures of Distributions[5]

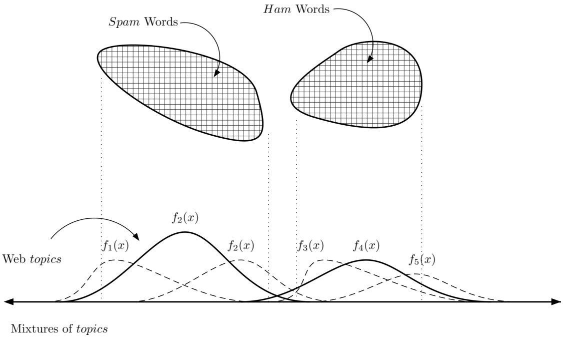


Figure: Other applications using mixtures of distributions are spam filters where *spam* and *ham* can be seen a web *topics*. Certain words appear more frequently within topics. [2]

[5]Think of an equation like this: $f(x) = \sum_i^n \psi_i f_i(x)$ where $\sum_i \psi_i = 1$

Zero Inflated Simulations

Puzzle

Pick an urn with probability p. If you pick urn A draw 0. If you pick urn B draw a number from a negative binomial distribution. Start over.

Zero Inflated Simulations

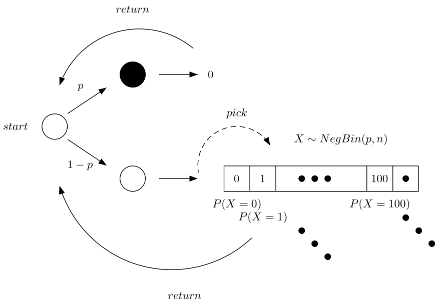


Figure: Picking a zero with probability p otherwise picking a number from a negative binomial.

24 Hour Simulations

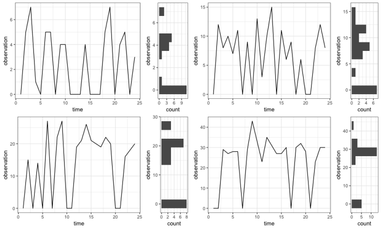


Figure: Zero-Inflated Poissons (*Zip*) with $\psi = .30$ along with $\lambda = 5, 10, 20, 30$

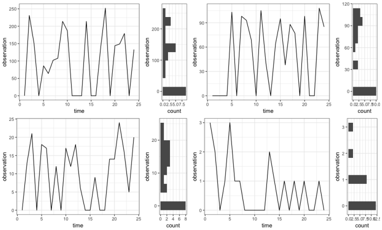


Figure: Zero-Inflated Negative Binomials (*Zinb*): $\psi = .3, n = 10, p = .01, .3, .4, .6$

Real World versus Simulations

Admittedly, our little game has limitations.

Puzzle

Consider hourly counts from one day to known botnets, phishing, dns-tunneling. Suppose, the order of the hours don't matter, can we simulate daily traffic with a $Zinb(\psi, p, n)$?[6]

[6] for some ψ, p, n that we can choose.

Simulating Malicious Traffic[7]

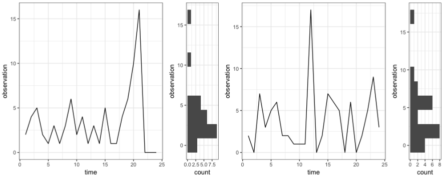


Figure: Botnet domain `a1a79b359237e.hosting` with $Zinb(0.13, 0.45, 3.24)$

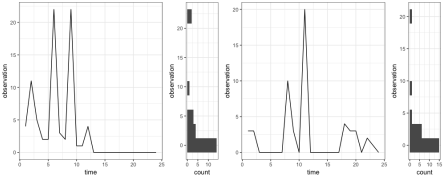


Figure: Phishing domain `support-globomail.com` with $Zinb(0.50, 0.25, 2.01)$

[7] Images on left *real* the right *simulated*

Simulating Malicious Traffic[8]

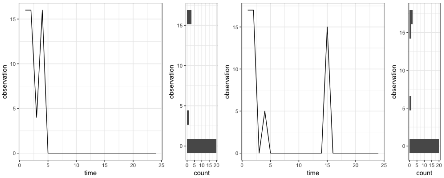


Figure: Phishing domain `universal-ads.com` with *Zinb*$(0.83, 0.39, 9.07)$

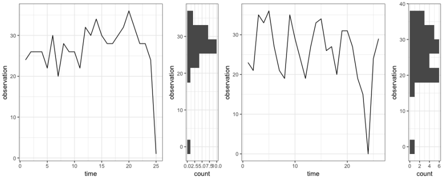


Figure: Phishing domain `clientes-moopixel.com` with *Zinb*$(0.10, 0.41, 17.81)$

[8] Image on left *real* the right *simulated*

Simulation Fit

Note

Be skeptical, just because a simulation looked good once, it might have been rare.

Measure of Fit to Malicious Traffic

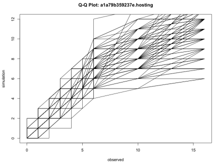


Figure: `a1a79b359237e.hosting`

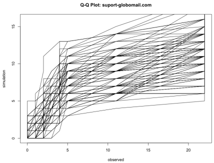


Figure: `support-globomail.com`

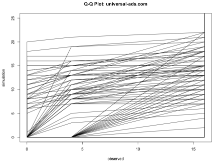


Figure: `universal-ads.com`

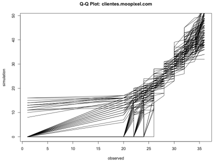


Figure: `clientes-moopixel.com`

Figure: *QQ*-Plots where tighter bands provide evidence the simulated data agrees with the observed. Wider bands, show more uncertainty.

Plan

Observe Signals

Generate Signals

Fit Fast

Fit Many

Fit Over Time

Measure Risk

Rainier supported by Stripe, Inc.[9] and authored by Avi Bryant[10] is an open-source Bayesian Inference project written in Scala.
The appeal of this project is:

- functional API with higher order function abstractions
- efficient hierarchical model fitting for datasets fitting in memory
- community of collaborators working on problems related to predictive modeling and risk and fraud detection

[9] https://stripe.com
[10] https://twitter.com/avibryant

Bayesian Inference and Monte Carlo Simulations

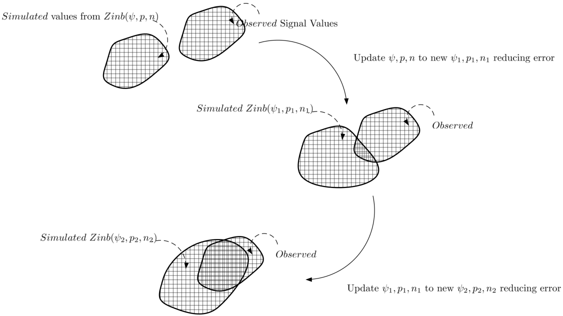


Figure: Bayesian inference is iterative process of drawing samples from prior distributions and comparing to observed data and updating those priors based on the error.

Example Bayesian Sampling[1] via Gibbs sampling

Bayesian Sampling with data-augmentation

```
procedure GIBBS SAMPLER                                  ▷ Estimating ψ and θ
    ψ^(0) ← u_0                                          ▷ u_0 ~ Uniform(0,1)
    θ^(0) ← θ_0                                          ▷ random θ_0
    for t ← 1, ... do
        Generate z_i^(t) (i = 1, ..., n) from (j = 1, ..., k)
            P(z_i^(t) = j | ψ_j^(t-1), θ_j^(t-1), x_i) ∝ ψ_j^(t-1) f(x_i | θ_j^(t-1))
        Generate ψ^(t) from π(ψ | z^(t))
        Generate θ^(t) from π(θ | z^(t), x)
    end for
    return ψ^(n), θ^(n)
end procedure
```

Sampling from Mixtures

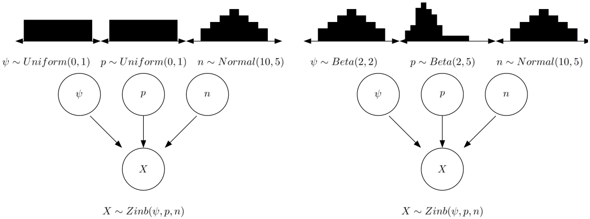


Figure: Two $Zinb(\psi, p, n)$ where the parameters ψ, p, n have different prior distributions.Some priors are considered *non-informative* and should be handled carefully.

Hello Rainier

Listing 1: Fitting Zero Inflated Negative Binomial in Rainier

```
import com.stripe.rainier.core.{NegativeBinomial, LogNormal, Beta}
import com.stripe.rainier.sampler.{RNG, ScalaRNG}

case class Zinb(psi: Double, p: Double, n: Double)

object ZinbMCMC extends Serializable {
  implicit val rng: RNG = ScalaRNG(1527608515939L)

  def fit(data: Seq[Int]): Zinb = {
    val priors = for {
      p <- Beta(2, 5).param
      n <- LogNormal(0, 1).param
    } yield (p, n)

    val psi = for {
      (p, n) <- priors
      psi <- Beta(2, 2).param
      fit <- NegativeBinomial(p, n).zeroInflated(psi).fit(data)
    } yield psi

    // ... your decide
    // ... call priors.sample() or psi.sample() for sequence of values

    Zinb(fitPsi, fitP, fitN)
  }
}
```

Plan

Observe Signals

Generate Signals

Fit Fast

Fit Many

Fit Over Time

Measure Risk

Massive Parallelization

Trick

Using Apache Spark we can distribute our simulations and run as many as we would like in parallel.[11]

[11] http://spark.apache.org

Massive Parallelization

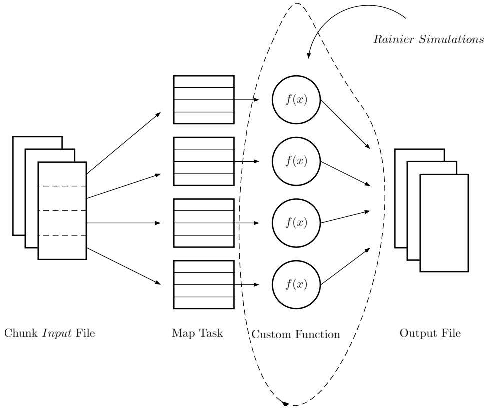


Figure: Passing chunks of the file(s) into rdd partitions, in Spark, distributes the Rainier simulations.

Puzzle

Given a file where each row contains a **(domain, day, Seq[Int])** write a program using Rainier to fit a zero inflated negative binomial distribution.

Hello Spark and Rainier [12]

Listing 2: Dispatching the Zinb simulation (to days worth simulating).

```
trait Event {
  val name: String
  val time: String
}

case class Dormant(name: String, time: String) extends Event
case class Singleton(name: String, time: String, value: Int) extends Event
case class MultiState(name: String, time: String, values: Seq[Int]) extends Event

def zinbDispatcher(event: Event): Zinb = {
    event match {
     case Dormant(_, _) => Zinb(0.0, 0.0, 0.0)
     case Singleton(_, _, value) => Zinb(1/2.40, 1/2, value*2)
     case MultiState(_, _, values) => ZinbMCMC.fit(values)
    }
}
```

[12] Completing the example: `sc.textFile(pathToFile).map(assignState).map(zinbDispatcher)`

Gotcha

Common errors occur with serialization of the rainier simulations. The previous example, not by accident, wrapped the *Zinb* simulation in a `Serializable` object. Another possibility, is to use:

```
com.twitter.chill.Meatlocker(f)
```

`chill` is shipped with `Spark`.

Plan

Observe Signals

Generate Signals

Fit Fast

Fit Many

Fit Over Time

Measure Risk

Scheduling the Processing

Major challenges in deciding:

- How many minutes/hours/days should be *fit*.
- How long between *fitting* each signal.

Scheduling Windows

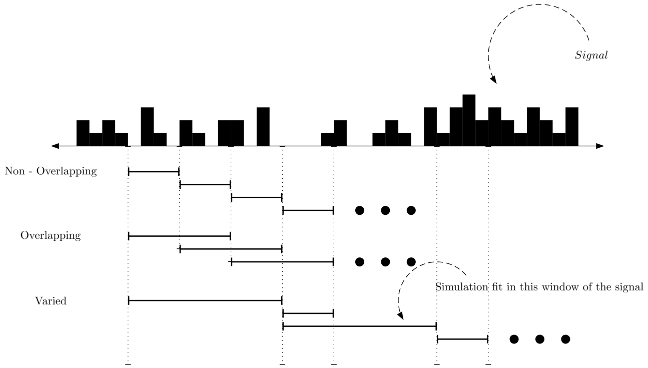


Figure: Some simulations can be run at non-overlapping intervals, overlapping intervals, and varied time windows.

Notes on Aggregation and Disaggregation

Note

The idea of *aggreagation* over a *large* window of time that is subsequently compared to an aggregation over a *small* window of time has been studied in problems related to **itermittant** demand. [4]

Plan

Observe Signals

Generate Signals

Fit Fast

Fit Many

Fit Over Time

Measure Risk

Measuring Risk

Goal

Exploit the parameterization of the fitted models to define a statistical measure of *rarity*.

Examples of common statistical tests:

- Given a data point x_i and a probability distribution $f(x;\theta)$ compute the p-value.
- Given data points: $x_1, \ldots, x_n$ and two models $f(x;\theta_1), g(x;\theta_2)$ compute the likelihood that the points are from one distribution rather than another.

Two Risk Measures

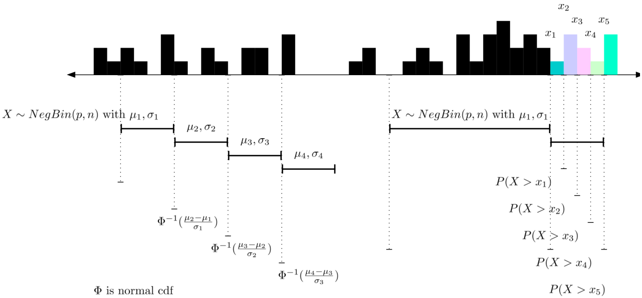


Figure: Parameters fit to previous observations of a signal can be used to analyze new observations in batch or streaming ways.

Risk Scores

Observation

We also want to *accumulate* risk-measures over time where more recent events contribute *more* to the score than older events. We can do this using exponential moving averages.

Given a new risk measure X_i at time i then update a time-dependent risk score S as follow:

$$S_i = X_i w + (1 - w) S_{i-1} \tag{2}$$

with $w \in [0, 1]$.

Time-Dependent Risk Measures

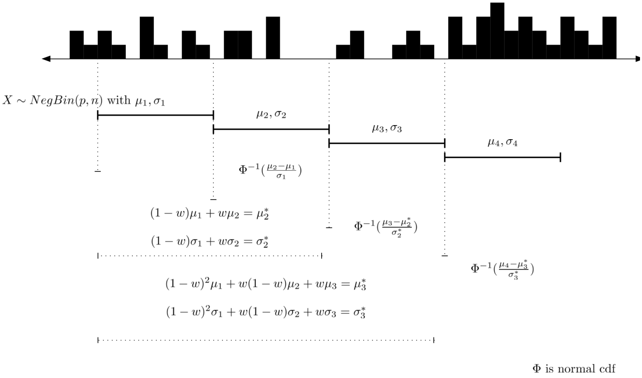


Figure: Example of trending historical μ_i, σ_i where more recent values contribute more.

Sample Pipeline

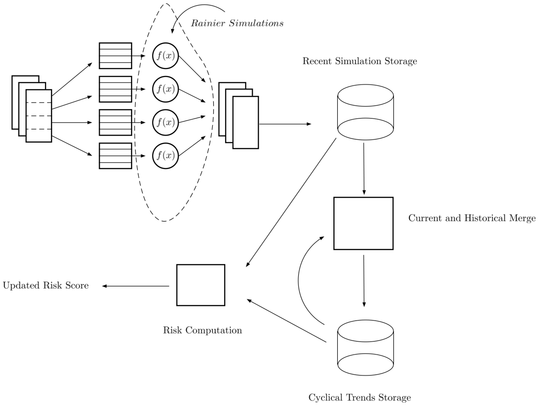


Figure: Example data pipeline where the most recent simulations are input to a historical database containing previous fitted parameters. Then, finally, a risk-score job fires off by reconciling the historical with the most recent simulation updating a chosen risk score.

Risk All Wrong[13]

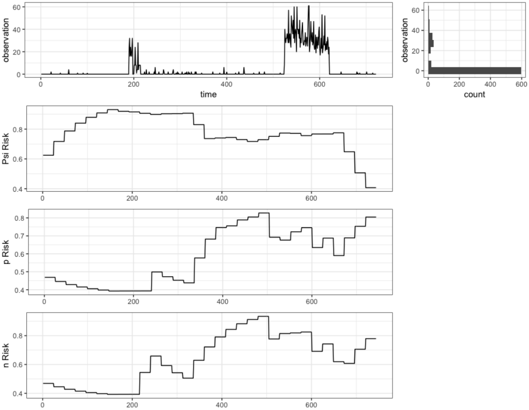


Figure: How not to create a risk score. Here the the risk-score per parameter is trended per weekday causing inappropriate correlations

[13] Additionally, there are good reasons why not to trend the parameters of a model.

Diebolt, J., and Robert, C. P.
Estimation of finite mixture distributions through bayesian sampling.
Journal of the Royal Statistical Society. Series B (Methodological) 56, 2 (1994), 363–375.

Jordan, M. I.
Graphical models.
Statistical Science 19, 1 (2004), 140–155.

Lambert, D., and Liu, C.
Adaptive thresholds.
Journal of the American Statistical Association 101, 473 (Mar 2006), 78–88.

Nikolopoulos, K., Syntetos, A. A., Boylan, J. E., Petropoulos, F., and Assimakopoulos, V.
An aggregate–disaggregate intermittent demand approach (adida) to forecasting: an empirical proposition and analysis.
Journal of the Operational Research Society 62, 3 (Mar 2011), 544–554.

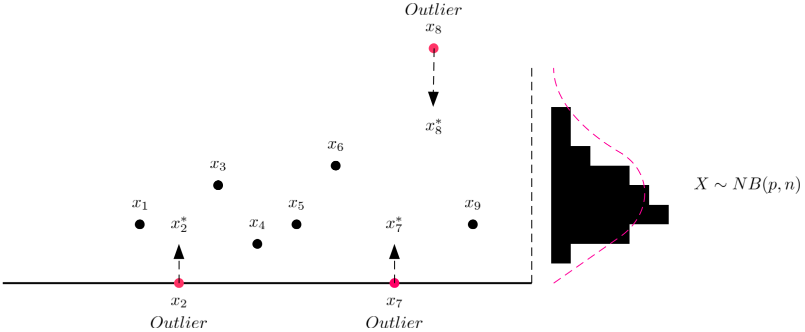
Discrete Pobability Distribution Updates: Adaptive Thresholding

Puzzle (Lambert, et al [3])

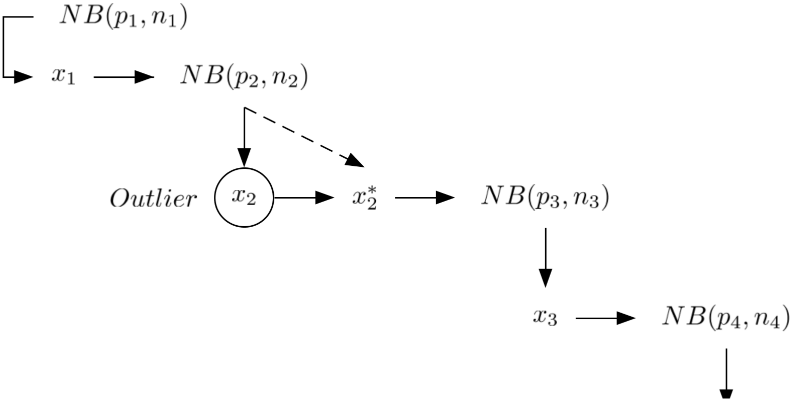
Update a negative binomial distribution $NB(p, n)$ from a stream of counts: $x_1, x_2, x_3, \ldots$.

Adaptive Thresholding

The **trick** is that not all values should contribute to updating the underlying parameters to $NB(p, n)$. In other words, outliers should be corrected or handled *robustly*.



Adaptive Thresholding



Two points worth exploring in the methods we've discussed are:

- Updating the distributions $NB(p_i, n_i)$ over time
- Tracking outlier significance