09.01 - Anomaly Detection

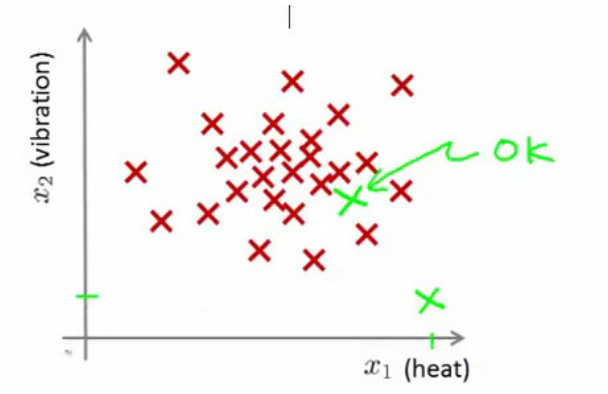

Figure 1: Anomaly detection in aircraft engines

1 Problem Motivation

Example:
Say that we have some aircraft engine features:

- x_1 = heat generated
- x_2 = vibration intensity

In this type of problem, we want to know if a new engine is "wrong", meaning that it might not function properly and we need to further test it.

As we can see in fig. 1, if the new engine is inside of the "normal" motors, then we assume it's also normal. On the other hand, if the engine falls in a region that is outside of the normal one, we consider it as being "wrong". The exact problem, then, is defining what is this normal region and how to decided whether a given point is inside it or not.

More formally, we're given a dataset $\{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}$ of normal data points and we need to decide whether x_{test} is anomalous.

We use probability models that estimate $P(x_{test} = \text{normal})$.

2 Gaussian Distribution

We say that $x \sim \mathcal{N}(\mu, \sigma^2)$ (x is distributed as Normal), with mean μ and variance σ^2 if:

$$p(x; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} \exp(-\frac{(x-\mu)^2}{2\sigma^2})$$

Which can be modelled as a bell curve, as seen in fig. 2. In simple terms, μ is where the center of the curve is and σ^2 is how wide the curve is.

3 Algorithm

Training set: $\{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}$.
Each example is $x \in \mathbb{R}^n$.

We're going to model the probabilities as such:

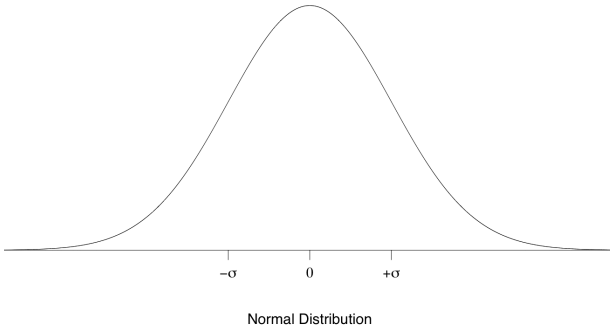

Figure 2: Normal distribution

$$p(x) = p(x_1; \mu_1, \sigma_1^2)\, p(x_2; \mu_2, \sigma_2^2) \dots p(x_n; \mu_n, \sigma_n^2) = \prod_{j=1}^{n} p(x_j; \mu_j, \sigma_j^2)$$

Which is basically to estimate each feature as a normal distribution and join them together. This formula assumes independent features, but it works well even though the assumption might be wrong.

Putting it all together, we have algorithm 1.

Algorithm 1 Anomaly detection algorithm using gaussians

1: Choose features x_i that might be indicative of anomalous examples.
2: Fit parameters $\mu_1, \dots, \mu_n$, $\sigma_1^2, \dots, \sigma_n^2$.
3: $\mu_j = \frac{1}{m} \sum_{i=1}^{m} x_j^{(i)}$
4: $\sigma_j^2 = \frac{1}{m} \sum_{i=1}^{m} (x_j^{(i)} - \mu_j)^2$
5: Given a new example x, compute:
6: $p(x) = \prod_{j=1}^{n} p(x_j; \mu_j, \sigma_j^2) = \prod_{j=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} \exp(-\frac{(x_j - \mu_j)^2}{2\sigma_j^2})$
7: Anomaly if $p(x) < \epsilon$

4 Developing and Evaluating and Anomaly Detection System

When developing a learning algorithm making decisions is much easier if we have a way of evaluating our learning algorithm.

To evaluate our anomaly detection system we need some labeled data, of anomalous and non-anomalous examples.

We still train with only normal/non-anomalous examples. The cv set and the test set contain anomalous data points as well as non-anomalous. Then we just fit the model and evaluate it as normal supervised problem (f1-score, recall, etc.). We can also use cross validation set to choose the threshold ϵ.

5 Anomaly Detection vs. Supervised Learning

If we have labeled data, why don't we just use a supervised learning straight away? One of the reasons is that anomaly detection works better with very small anomalies (positive examples) whereas supervised learning works better when we have a large number of positive and negative examples.

Also, anomaly detection works better when we have many differet types of anomalies, so that supervised learning doesn't learn from positive examples what the anomalies look like. Future anomalies might look nothing like any of the anomalous examples we've seen so far.

6 Choosing What Features to Use

Before running algorithm plot histogram of the data. If the result is a non gaussian plot, try to transform it (use log(x), log(x+constant), x^c, etc..) and, if after transforming it it looks gausian, substitute the whole feature for its transformation.

A good way to get new features is to look at anamolous examples missclassified manually to search what differences they have compared to the normal examples.

Another good idea is to choose features that might take on unusually large or small values in the event of an anormally.

7 Multivariate Gaussian Distribution

Problem of the anterior algorithm is that it does not take into account the other features. It evaluates every feature differently.

This algorithm, on the other hand, doesn't model $p(x_1), p(x_2) \dots$ separetly, it models $p(x)$ in one go. This means that, to model a feature, it uses inormation from the rest, so it might work better. Parameters: $\mu \in \mathbb{R}^n, \Sigma \in \mathbb{R}^{n \times n}$ (covariance matrix).

$$p(x; \mu, \Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu))$$

8 Anomaly Detection using the Multivariate Gaussian Distribution

Parameter fitting:

$$\mu = \frac{1}{m}\sum_{i=1}^{m} x^{(i)}$$

$$\Sigma = \frac{1}{m}\sum_{i=1}^{m}(x^{(i)}-\mu)(x^{(i)}-\mu)^T$$

After we have done this, we do the same as the normal anomaly detection algorithm but we calculate $p(x)$ with the new formula.

This new model captures correlations between features automatically. It's also computationally more expensive. Furthermore, it needs $m > n$ (more samples than features), otherwise Σ is non-invertible.