share.coursera.org

ML:Anomaly Detection - Coursera

Just like in other learning problems, we are given a dataset $x^{(1)}, x^{(2)}, ..., x^{(m)}$.

We are then given a new example, x_{test}, and we want to know whether this new example is abnormal/anomalous.

We define a "model" $p(x)$ that tells us the probability the example is not anomalous. We also use a threshold ϵ (epsilon) as a dividing line so we can say which examples are anomalous and which are not.

A very common application of anomaly detection is detecting fraud:

> $x^{(i)} =$ features of user i's activities
>
> Model $p(x)$ from the data.
>
> Identify unusual users by checking which have $p(x) < \epsilon$.

If our anomaly detector is flagging **too many** anomalous examples, then we need to **decrease** our threshold ϵ

The Gaussian Distribution is a familiar bell-shaped curve that can be described by a function $\square(\mu, \sigma^2)$

Let $x \in \mathbb{R}$. If the probability distribution of x is Gaussian with mean μ, variance σ^2, then:

> $x \sim \square(\mu, \sigma^2)$

The little $\sim$ or 'tilde' can be read as "distributed as."

The Gaussian Distribution is parameterized by a mean and a variance.

Mu, or μ, describes the center of the curve, called the mean. The width of the curve is described by sigma, or σ, called the standard deviation.

The full function is as follows:

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

We can estimate the parameter μ from a given dataset by simply taking the average of all the examples:

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

We can estimate the other parameter, σ^2, with our familiar squared error formula:

$$\sigma^2 = \frac{1}{m}\sum_{i=1}^{m} (x^{(i)} - \mu)^2$$

Given a training set of examples, $\{x^{(1)}, ..., x^{(m)}\}$ where each example is a vector, $x \in \mathbb{R}^n$.

$$p(x) = p(x_1;\mu_1,\sigma_1^2)p(x_2;\mu_2,\sigma_2^2)\cdots p(x_n;\mu_n,\sigma_n^2)$$

In statistics, this is called an "independence assumption" on the values of the features inside training example x.

More compactly, the above expression can be written as follows:

$$= \prod_{j=1}^{n} p(x_j;\mu_j,\sigma_j^2)$$

The algorithm

Choose features x_i that you think might be indicative of anomalous examples.

Fit parameters $\mu_1, ..., \mu_n, \sigma_1^2, ..., \sigma_n^2$.

Calculate $\mu_j = \frac{1}{m} \sum_{i=1}^{m} x_j^{(i)}$

Calculate $\sigma_j^2 = \frac{1}{m} \sum_{i=1}^{m} (x_j^{(i)} - \mu_j)^2$

Given a new example x, compute $p(x)$:

$$p(x) = \prod_{j=1}^{n} p(x_j; \mu_j, \sigma_j^2) = \prod_{j=1}^{n} \frac{1}{\sqrt{2\pi}\sigma_j} exp(-\frac{(x_j - \mu_j)^2}{2\sigma_j^2})$$

Anomaly if $p(x) < \epsilon$

A vectorized version of the calculation for μ is $\mu = \frac{1}{m} \sum_{i=1}^{m} x^{(i)}$. You can vectorize σ^2 similarly.

To evaluate our learning algorithm, we take some labeled data, categorized into anomalous and non-anomalous examples ($y = 0$ if normal, $y = 1$ if anomalous).

Among that data, take a large proportion of **good**, non-anomalous data for the training set on which to train $p(x)$.

Then, take a smaller proportion of mixed anomalous and non-anomalous examples (you will usually have many more non-anomalous examples) for your cross-validation and test sets.

For example, we may have a set where 0.2% of the data is anomalous. We take 60% of those examples, all of which are good ($y = 0$) for the training set. We then take 20% of the examples for the cross-validation set (with 0.1% of the anomalous examples) and another 20% from the test set (with another 0.1% of the anomalous).

In other words, we split the data 60/20/20 training/CV/test and then split the anomalous examples 50/50 between the CV and test sets.

Algorithm evaluation:

Fit model $p(x)$ on training set $\{x^{(1)}, ..., x^{(m)}\}$

On a cross validation/test example x, predict:

If $p(x) < \epsilon$ (**anomaly**), then $y = 1$

If $p(x) \geq \epsilon$ (**normal**), then $y = 0$

Possible evaluation metrics (see "Machine Learning System Design" section):

- True positive, false positive, false negative, true negative.
- Precision/recall
- F_1 score

Note that we use the cross-validation set to choose parameter ϵ

When do we use anomaly detection and when do we use supervised learning?

Use anomaly detection when...

- We have a very small number of positive examples ($y = 1$ ... 0-20 examples is common) and a large number of negative ($y = 0$) examples.
- We have many different "types" of anomalies and it is hard for any algorithm to learn from positive examples what the anomalies look like; future anomalies may look nothing like any of the anomalous examples we've seen so far.

Use supervised learning when...

- We have a large number of both positive and negative examples. In other words, the training set is more evenly divided into classes.

- We have enough positive examples for the algorithm to get a sense of what new positives examples look like. The future positive examples are likely similar to the ones in the training set.

The features will greatly affect how well your anomaly detection algorithm works.

We can check that our features are **gaussian** by plotting a histogram of our data and checking for the bell-shaped curve.

Some **transforms** we can try on an example feature x that does not have the bell-shaped curve are:

- $log(x)$

- $log(x+1)$

- $log(x+c)$ for some constant

- $\sqrt{x}$

- $x^{1/3}$

We can play with each of these to try and achieve the gaussian shape in our data.

There is an **error analysis procedure** for anomaly detection that is very similar to the one in supervised learning.

Our goal is for $p(x)$ to be large for normal examples and small for anomalous examples.

One common problem is when $p(x)$ is similar for both types of examples. In this case, you need to examine the anomalous examples that are giving high probability in detail and try to figure out new features that will better distinguish the data.

In general, choose features that might take on unusually large or small values in the event of an anomaly.

The multivariate gaussian distribution is an extension of anomaly detection and may (or may not) catch more anomalies.

Instead of modeling $p(x_1), p(x_2), \ldots$ separately, we will model $p(x)$ all in one go. Our parameters will be: $\mu \in \mathbb{R}^n$ and $\Sigma \in \mathbb{R}^{n \times n}$

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

The important effect is that we can model oblong gaussian contours, allowing us to better fit data that might not fit into the normal circular contours.

Varying Σ changes the shape, width, and orientation of the contours. Changing μ will move the center of the distribution.

Check also:

- The Multivariate Gaussian Distribution (pdf), Chuong B. Do, October 10, 2008.

When doing anomaly detection with multivariate gaussian distribution, we compute μ and Σ normally. We then compute $p(x)$ using the new formula in the previous section and flag an anomaly if $p(x) < \epsilon$.

The original model for $p(x)$ corresponds to a multivariate Gaussian where the contours of $p(x; \mu, \Sigma)$ are axis-aligned.

The multivariate Gaussian model can automatically capture correlations between different features of x.

However, the original model maintains some advantages: it is computationally cheaper (no matrix to invert, which is costly for large number of features) and it performs well even with small training set size (in multivariate Gaussian model, it should be greater than the number of features for Σ to be invertible).

Next: Recommender Systems Back to Index: Main