Mahalanobis Distance

James C Craven

Thursday, December 5th 2024

What?

- The Mahalanobis distance is a statistical method that detects outliers in a numerical, multivariate dataset.
- It was developed by P.C. Mahalanobis [mæ.ha.la.no'bis], an indian scientists and mathematician in the year 1933

So What?

The Mahalanobis distance is used for a variety of different applications:

- Chemometrics - identifying uncommon/unusual chemical samples
- Regression - identifying which data points have the highest influence in the regression
- Ecological Niche Modeling - Measuring deviation of environmental factors from typical conditions

How?

The Mahalanobis distance metric (w.r.t. a distribution Q) is calculated for one data point of the dataset using the point, the mean vector $\vec{\mu}$, and the inverse of the covariance matrix, S^{-1}:

$$d_M(\vec{x}, Q) = \sqrt{(\vec{x} - \vec{\mu})^T S^{-1} (\vec{x} - \vec{\mu})}$$

This is done for all data points in the dataset and then is compared against a Chi-squared metric. For a multivariate distribution $Q \sim \mathcal{N}$ with n features, the metric is equal to $\chi^2_{1-\alpha,n}$ where $1 - \alpha$ is the level of confidence we can assert that the point is an outlier.

What if my data is ”weird”?

Even for data that is not normally distributed, the Mahalanobis Distance metric is useful due to the **Central Limit Theorem**:

Theorem

If X_1, X_2,. . . , X_n constitute a random sample from an infinite population with mean μ, the variance σ^2, and the moment-generating function $M_X(t)$, then the limiting distribution of

$$Z = \frac{\bar{X} - \mu}{\sigma/\sqrt{n}}$$

as $n \to \infty$ is the standard normal distribution, $\mathcal{N}(X; 0, 1)$

On to more interesting things...

```
import numpy as np
from scipy.spatial.distance import mahalanobis
from scipy.stats import chi2

# read in data
data = ...

alpha = 0.01
mu = np.mean(data, axis=0)
cov_inv = np.linalg.inv(np.cov(data, rowvar=Falase))

distances = [mahalanobis(point, mu, cov_inv)
             for point in data]

threshold = np.sqrt(chi2.ppf(1 - alpha, data.shape[1]))

outliers = data[np.where(distances > threshold)]
print(outliers)
```

I don’t like that though . . .

- It requires two obscure imports from scipy
- It requires you to remember how to calculate the threshold
- Boilerplate is gross, and users shouldn’t have to write their own function for this