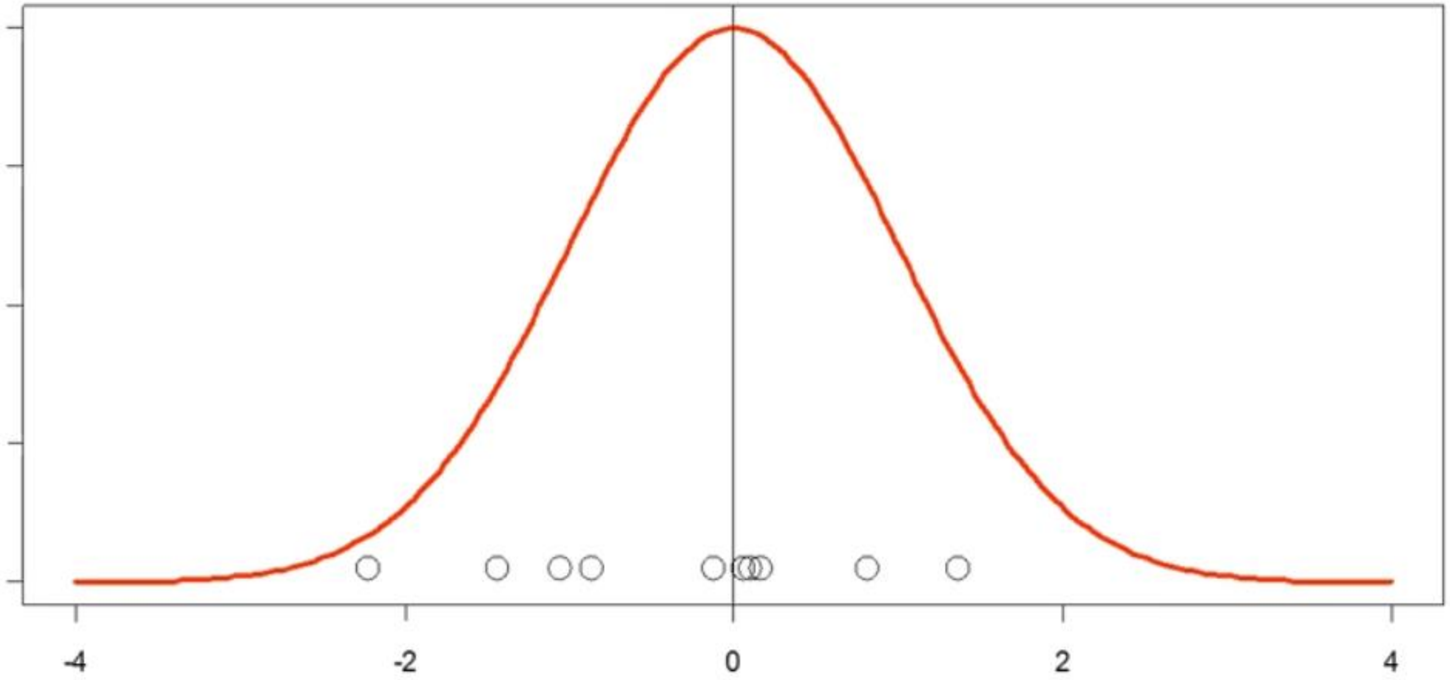
马氏距离

Mahalanobis distance

Measuring Distance

The distance of a point from the mean in univariate space is a simple measure: $x_i - \bar{x}$

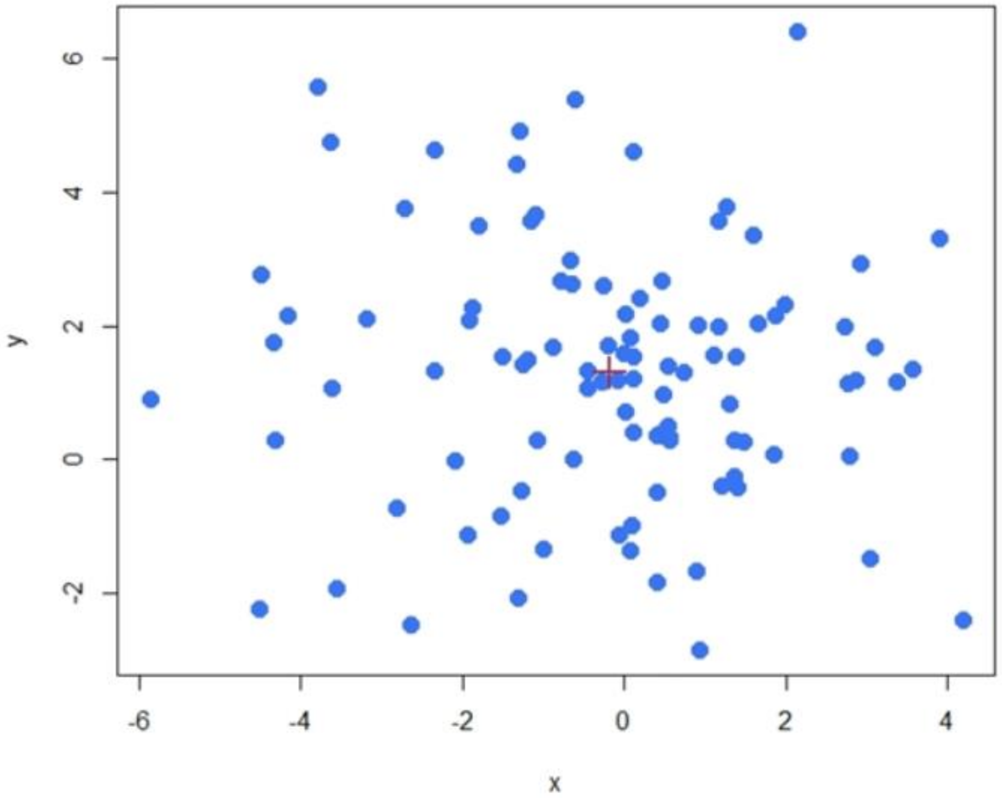
This type of distance is called Euclidean distance



Measuring Multivariate Distance

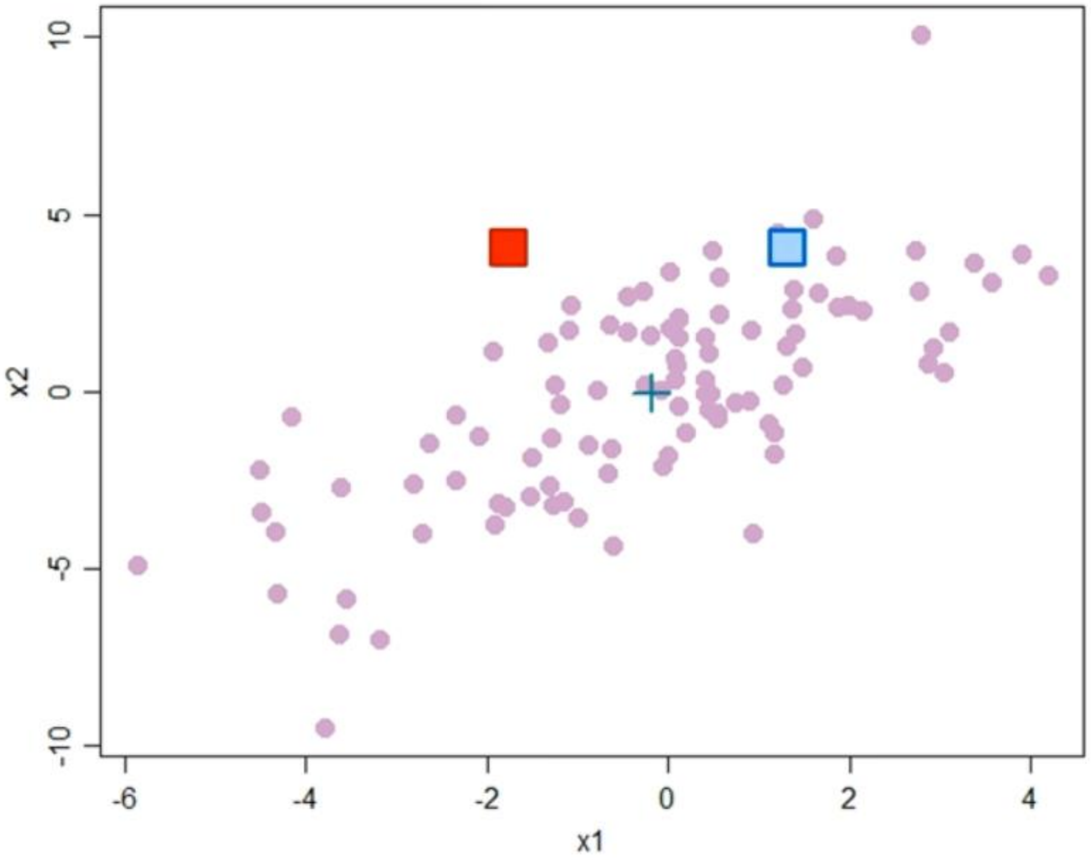
This can easily be extended to multivariate space by using the Euclidean distance of a point from the mean:

$$\sqrt{(x_i - \overline{x})^2 + (y_i - \overline{y})^2 + \cdots + (n_i - \overline{n})^2}$$



Measuring Multivariate Distance

However, Euclidean distance has limitations in real datasets,which often have some degree of covariance



Covariance

$$
\begin{aligned}
Cov\left(X,Y\right) &= E\left[\left(X-E\left[X\right]\right)\left(Y-E\left[Y\right]\right)\right] \\
&= E\left[XY\right]-2E\left[Y\right]E\left[X\right]+E\left[X\right]E\left[Y\right] \\
&= E\left[XY\right]-E\left[X\right]E\left[Y\right]
\end{aligned}
$$

Covariance Matrix

Square matrix of covariances between all pairs of variables

$$
\begin{array}{cc}
 & \begin{array}{ccc} \boldsymbol{x1} & \boldsymbol{x2} & \boldsymbol{x3} \end{array} \\
\begin{array}{c} \boldsymbol{x1} \\ \boldsymbol{x2} \\ \boldsymbol{x3} \end{array} &
\begin{bmatrix}
cov(x1,x1) & cov(x1,x2) & cov(x1,x3) \\
cov(x2,x1) & cov(x2,x2) & cov(x2,x3) \\
cov(x3,x1) & cov(x3,x2) & cov(x3,x3)
\end{bmatrix}
\end{array}
$$

The covariance of a variable with itself is just its variance, so the matrix diagonals contain the variances.

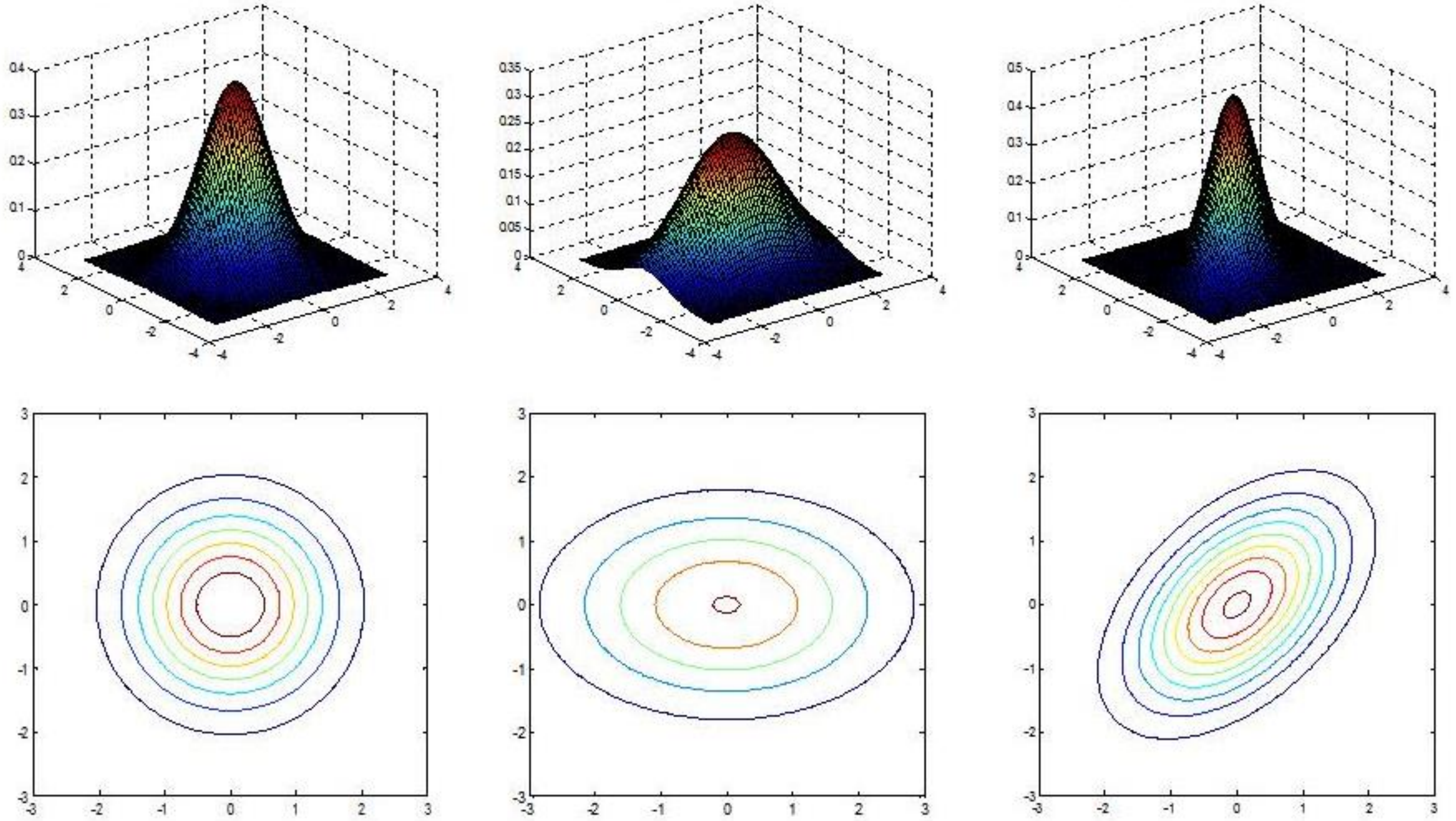
Matrix is also symmetrical because cov(x1, x2) = cov(x2, x1).

$$
\begin{array}{cc}
 & \begin{array}{ccc} \boldsymbol{x1} & \boldsymbol{x2} & \boldsymbol{x3} \end{array} \\
\begin{array}{c} \boldsymbol{x1} \\ \boldsymbol{x2} \\ \boldsymbol{x3} \end{array} &
\begin{bmatrix}
var(x1) & cov(x1,x2) & cov(x1,x3) \\
cov(x2,x1) & var(x2) & cov(x2,x3) \\
cov(x3,x1) & cov(x3,x2) & var(x3)
\end{bmatrix}
\end{array}
$$

$$\Sigma = \begin{bmatrix} 1.0 & 0 \\ 0 & 1.0 \end{bmatrix} \quad \mu = [0 \quad 0]$$

$$\Sigma = \begin{bmatrix} 2.5 & 0 \\ 0 & 1.0 \end{bmatrix} \quad \mu = [0 \quad 0]$$

$$\Sigma = \begin{bmatrix} 1.0 & 0.5 \\ 0.5 & 1.0 \end{bmatrix} \quad \mu = [0 \quad 0]$$



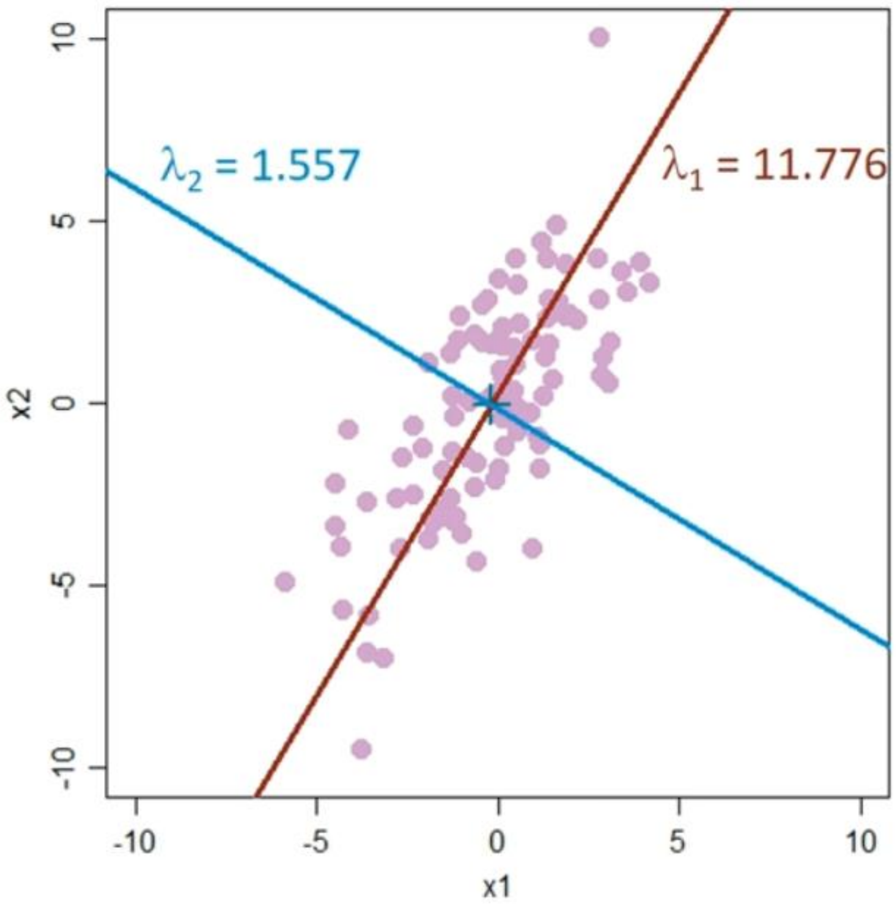
Rescaling to Remove Covariance

$$A\mathbf{x} = \lambda\mathbf{x}$$

Remove covariance by treating each eigenvector as a new axis, shrink axis by $\sqrt{\lambda_i}$, then calculate distance between points

Mahalanobis distance measure does the following:

- it transforms the variables into uncorrelated variables
- and makes their variances equal to 1
- then calculates simple Euclidean distance.



与欧氏距离不同的是，它考虑到各种特性之间的联系（例如：一条关于身高的信息会带来一条关于体重的信息，因为两者是有关联的），并且是尺度无关的(scale-invariant)，即独立于测量尺度。

(squared) Mahalanobis Distance

T indicates a transposed matrix

$$D^2 = (x - \overline{x})^T S^{-1} (x - \overline{x})$$

Matrix of distances from mean

Inverse of covariance matrix

Matrix of:

$(x_1, x_2, \dots, x_n)$ - $(\overline{x}_1, \overline{x}_2, \dots, \overline{x}_n)$

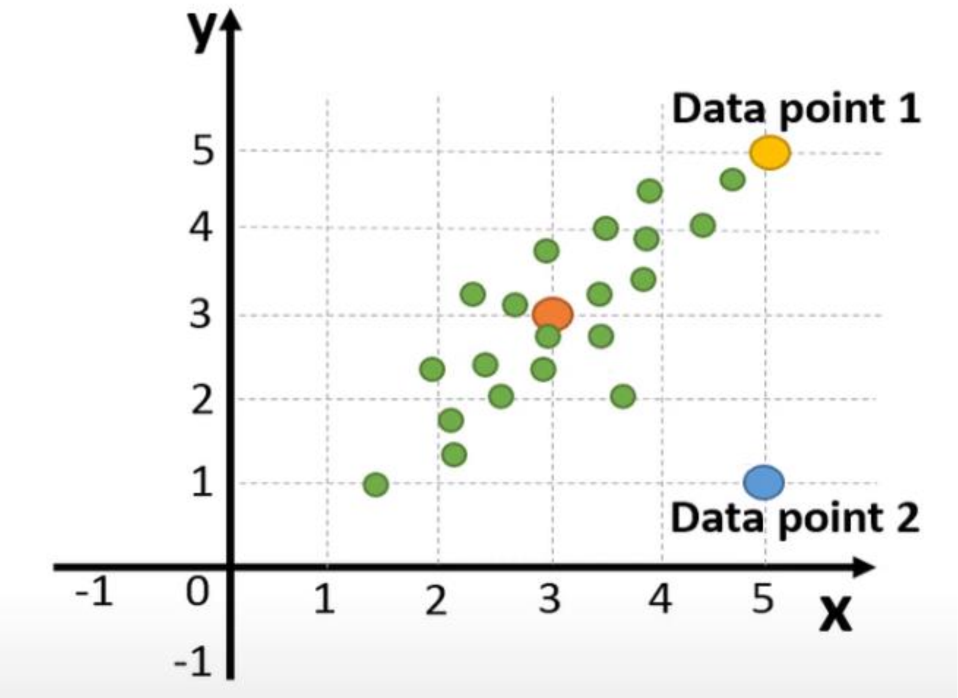
or

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} - \begin{bmatrix} \overline{x}_1 \\ \overline{x}_2 \\ \vdots \\ \overline{x}_n \end{bmatrix}$$

$$\begin{bmatrix} s_1^2 & \cdots & \mathrm{Cov}(s_n, s_1) \\ \vdots & \ddots & \vdots \\ \mathrm{Cov}(s_1, s_n) & \cdots & s_n^2 \end{bmatrix}$$

Matrix with diagonals = variance of samples 1 … *n* and cells = covariance of samples (1,2) … (1,*n*)

如果协方差矩阵为单位矩阵，那么马氏距离就简化为欧氏距离



$$\text{Centroid} = \begin{pmatrix} \overline{X} \\ \overline{Y} \end{pmatrix} = \begin{pmatrix} 3.1 \\ 3.0 \end{pmatrix}$$

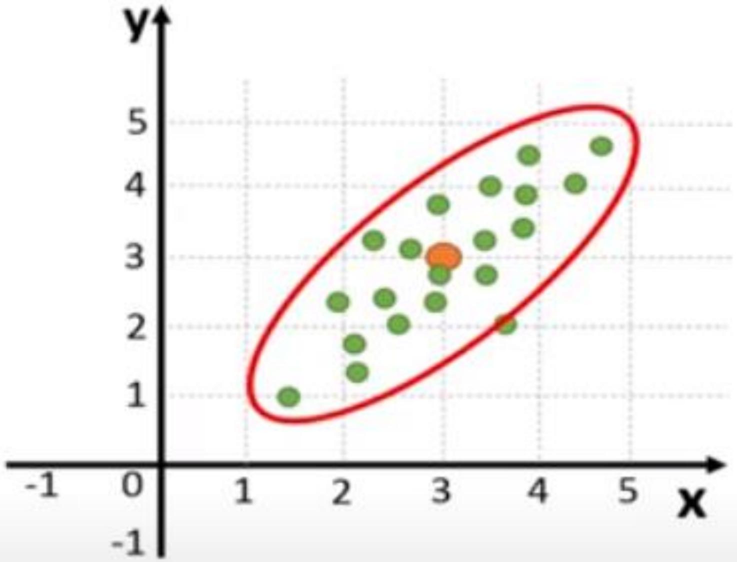
$$d = \sqrt{(5-3.1)^2 + (5-3.0)^2} = 2.76$$

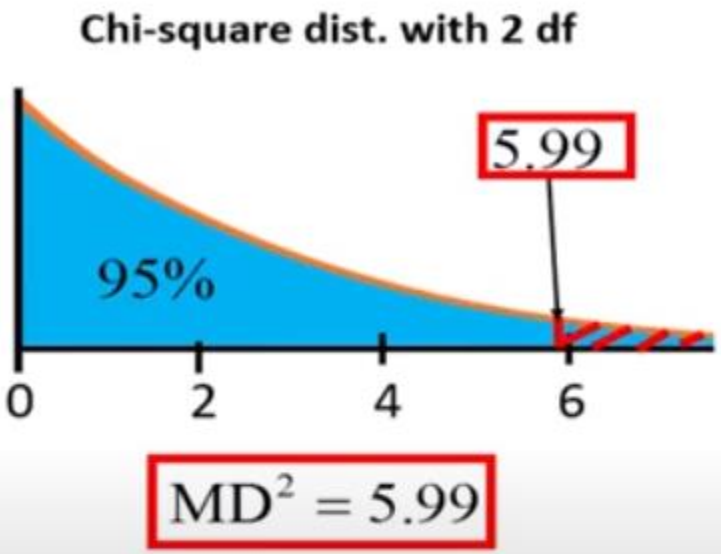
$$d = \sqrt{(5-3.1)^2 + (1-3.0)^2} = 2.76$$

$$\text{MD}_1 = 2.26$$

$$\text{MD}_2 = 6.45$$

Error ellipse





This ellipse can be calculated based on the assumption that the data is multivariate normally distributed.

The squared Mahalanobis distance should then follow a Chi-Square distribution with p degrees of freedom.

计算方法

It seems like maybe you're getting S and then inverting it.

You shouldn't do that; it's slow and numerically inaccurate.

Instead, you should get the Cholesky factor L of S so that $S = L\, L$^T; then

```
M^2(x, y; L L^T)
  = (x - y)^T (L L^T)^-1 (x - y)
  = (x - y)^T L^-T L^-1 (x - y)
  = || L^-1 (x - y) ||^2,
```

and since L is triangular L^-1 (x - y) can be computed efficiently.

As it turns out, scipy.linalg.solve_triangular will happily do a bunch of these at once if you reshape it properly.