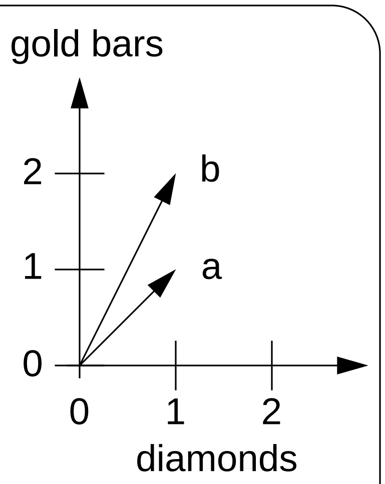
Distance Measures



What is the distance between pt. a and pt. b?

The N-dimensional real Cartesian space, denoted $\Re^N$ is the collection of all N-dimensional vectors with real elements. A metric, or distance measure, is a real-valued function with three properties:

$\forall \bar{x}, \bar{y}, \bar{z} \in \Re^N$:

1. $d(\bar{x}, \bar{y}) \geq 0$.
2. $d(\bar{x}, \bar{y}) = 0$ if and only if $\bar{x} = \bar{y}$
3. $d(\bar{x}, \bar{y}) \leq d(\bar{x}, \bar{z}) + d(\bar{z}, \bar{y})$

The Minkowski metric of order s, or the l_s metric, between $\bar{x}$ and $\bar{y}$ is:

$$d_s(\bar{x}, \bar{y}) \equiv \sqrt[s]{\sum_{k=1}^{N} \left|x_k - y_k\right|^s} = \|\bar{x} - \bar{y}\|_s$$

(the norm of the difference vector).

Important cases are:

1. l_1 or city block metric (sum of absolute values),

$$d_1(\bar{x}, \bar{y}) = \sum_{k=1}^{N} \left|x_k - y_k\right|$$

2. l_2, or Euclidean metric (mean-squared error),

$$d_2(\bar{x}, \bar{y}) = \sqrt{\sum_{k=1}^{N} \left|x_k - y_k\right|^2}$$

3. l_∞ or Chebyshev metric,

$$d_\infty(\bar{x}, \bar{y}) = \max_k \left|x_k - y_k\right|$$

We can similarly define a weighted Euclidean distance metric:

$$d_{2w}(\bar{x}, \bar{y}) = \sqrt{|\bar{x} - \bar{y}|^T \underline{W} |\bar{x} - \bar{y}|}$$

where:

$$\bar{x} = \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_k \end{bmatrix}, \bar{y} = \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_k \end{bmatrix}, \text{ and } \underline{W} = \begin{bmatrix} w_{11} & w_{12} & \dots & w_{1k} \\ w_{21} & w_{22} & \dots & w_{2k} \\ \dots & \dots & \dots & \dots \\ w_{k1} & w_{k2} & \dots & w_{kk} \end{bmatrix}.$$

Why are Euclidean distances so popular?

One reason is efficient computation. Suppose we are given a set of M reference vectors, $\bar{x}_m$, a measurement, $\bar{y}$, and we want to find the nearest neighbor:

$$NN = \underset{m}{min}\ d_2(\bar{x}_m, \bar{y})$$

This can be simplified as follows:

We note the minimum of a square root is the same as the minimum of a square (both are monotonically increasing functions):

$$d_2(\bar{x}_m, \bar{y})^2 = \sum_{j=1}^{k} (x_{m_j} - y_j)^2 = \sum_{j=1}^{k} x_{m_j}{}^2 - 2x_{m_j} y_j + y_j^2$$

$$= \left\|\bar{x}_m\right\|^2 - 2\bar{x}_m \bullet \bar{y} + \|\bar{y}\|^2$$

$$= C_m + C_y - 2\bar{x}_m \bullet \bar{y}$$

Therefore,

$$NN = \underset{m}{min}\, d_2(\bar{x}_m, \bar{y}) = C_m - 2\bar{x}_m \bullet \bar{y}$$

Thus, a Euclidean distance is virtually equivalent to a dot product (which can be computed very quickly on a vector processor). In fact, if all reference vectors have the same magnitude, C_m can be ignored (normalized codebook).

Prewhitening of Features

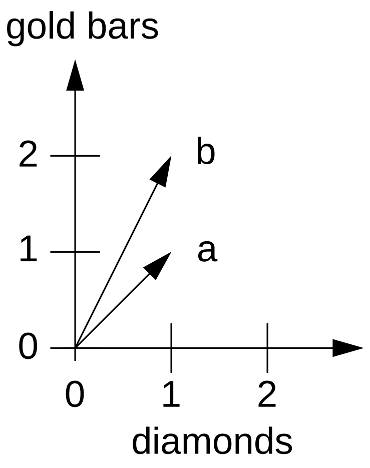
Consider the problem of comparing features of different scales:

Suppose we represent these points in space in two coordinate systems using the transformation:

$$\bar{z} = \underline{V}\bar{x}$$

System 1:

$$\beta_1 = 1\hat{i} + 0\hat{j} \text{ and } \beta_2 = 0\hat{i} + 1\hat{j}$$

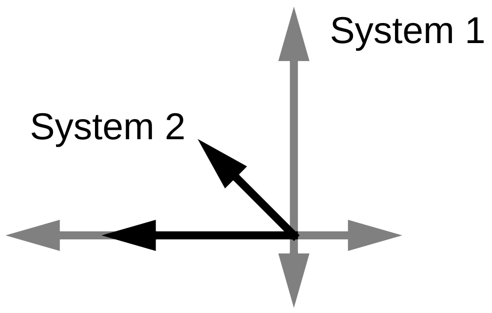


$$\bar{a} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \qquad \bar{b} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

$$d_2(\bar{a}, \bar{b}) = \sqrt{0^2 + 1^2} = 1$$

System 2:

$$\gamma_1 = -2\hat{i} + 0\hat{j} \text{ and } \gamma_1 = -1\hat{i} + 1\hat{j}$$



$$\bar{a} = \begin{bmatrix} -2 & 0 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} -1 \\ 1 \end{bmatrix} \qquad \bar{b} = \begin{bmatrix} -2 & 0 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} -\frac{3}{2} \\ 2 \end{bmatrix}$$

$$d_2(\bar{a}, \bar{b}) = \sqrt{\left(-1 - \left(-\frac{3}{2}\right)\right)^2 + (1-2)^2} = \sqrt{\frac{5}{4}}$$

The magnitude of the distance has changed. Though the rank-ordering of distances under such linear transformations won’t change, the cumulative effects of such changes in distances can be damaging in pattern recognition. Why?

We can simplify the distance calculation in the transformed space:

$$\begin{aligned} d_2(\underline{V}\bar{x}, \underline{V}\bar{y}) &= \sqrt{[\underline{V}\bar{x} - \underline{V}\bar{y}]^T [\underline{V}\bar{x} - \underline{V}\bar{y}]} \\ &= \sqrt{[\bar{x} - \bar{y}]^T \underline{V}^T \underline{V} [\bar{x} - \bar{y}]} \\ &= d_{2W}(\bar{x}, \bar{y}) \end{aligned}$$

This is just a weighted Euclidean distance.

Suppose all dimensions of the vector are not equal in importance. For example, suppose one dimension has virtually no variation, while another is very reliable. Suppose two dimensions are statistically correlated. What is a statistically optimal transformation?

Consider a decomposition of the covariance matrix (which is symmetric):

$$\underline{C} = \underline{\Phi}\underline{\Lambda}\underline{\Phi}^T$$

where Φ denotes a matrix of eigenvectors of $\underline{C}$ and $\underline{\Lambda}$ denotes a diagonal matrix whose elements are the eigenvalues of $\underline{C}$. Consider:

$$\bar{z} = \underline{\Lambda}^{-1/2}\underline{\Phi}\bar{x}$$

The covariance of $\bar{z}$, $\underline{C}_{\bar{z}}$ is easily shown to be an identity matrix (prove this!) We can also show that:

$$d_2(\bar{z}_1, \bar{z}_2) = \sqrt{[\bar{x}_1 - \bar{x}_2]^T \underline{C}_{\bar{x}}^{-1} [\bar{x}_1 - \bar{x}_2]}$$

Again, just a weighted Euclidean distance.

- If the covariance matrix of the transformed vector is a diagonal matrix, the transformation is said to be an orthogonal transform.

- If the covariance matrix is an identity matrix, the transform is said to be an orthonormal transform.

- A common approximation to this procedure is to assume the dimensions of $\bar{x}$ are uncorrelated but of unequal variances, and to approximate $\underline{C}$ by a diagonal matrix, $\underline{\Lambda}$. Why? This is known as variance-weighting.

“Noise-Reduction”

The prewhitening transform, $\bar{z} = \underline{\Lambda}^{-1/2}\underline{\Phi}\bar{x}$, is normally created as a $k \times k$ matrix in which the eigenvalues are ordered from largest to smallest:

$$\begin{bmatrix} z_1 \\ z_2 \\ \dots \\ z_k \end{bmatrix} = \begin{bmatrix} \lambda_1^{-1/2} & & & \\ & \lambda_2^{-1/2} & & \\ & & \dots & \\ & & & \lambda_k^{-1/2} \end{bmatrix} \begin{bmatrix} v_{11} & v_{12} & \dots & v_{13} \\ v_{21} & v_{22} & \dots & v_{2k} \\ \dots & \dots & \dots & \dots \\ v_{k1} & v_{k2} & \dots & v_{kk} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_k \end{bmatrix}$$

where

$$\lambda_1 > \lambda_2 > \dots > \lambda_k .$$

In this case, a new feature vector can be formed by truncating the transformation matrix to $l < k$ rows. This is essentially discarding the least important features.

A measure of the amount of discriminatory power contained in a feature, or a set of features, can be defined as follows:

$$\%\ \mathrm{var} = \frac{\sum_{j=1}^{l} \lambda_j}{\sum_{j=1}^{k} \lambda_j}$$

This is the percent of the variance accounted for by the first l features.

Similarly, the coefficients of the eigenvectors tell us which dimensions of the input feature vector contribute most heavily to a dimension of the output feature vector. This is useful in determining the “meaning” of a particular feature (for example, the first decorrelated feature often is correlated with the overall spectral slope in a speech recognition system — this is sometimes an indication of the type of microphone).

Computational Procedures

Computing a “noise-free” covariance matrix is often difficult. One might attempt to do something simple, such as:

$$c_{ij} = \sum_{n=0}^{N-1} (x_i - \mu_i)(x_j - \mu_j) \text{ and } \mu_i = \sum_{n=0}^{N-1} x_i$$

On paper, this appears reasonable. However, often, the complete set of feature vectors contains valid data (speech signals) and noise (nonspeech signals). Hence, we will often compute the covariance matrix across a subset of the data, such as the particular acoustic event (a phoneme or word) we are interested in.

Second, the covariance matrix is often ill-conditioned. Stabilization procedures are used in which the elements of the covariance matrix are limited by some minimum value (a noise-floor or minimum SNR) so that the covariance matrix is better conditioned.

But how do we compute eigenvalues and eigenvectors on a computer?
One of the hardest things to do numerically! Why?

Suggestion: use a canned routine (see Numerical Recipes in C).

The definitive source is EISPACK (originally implemented in Fortran, now available in C). A simple method for symmetric matrices is known as the Jacobi transformation. In this method, a sequence of transformations are applied that set one off-diagonal element to zero at a time. The product of the subsequent transformations is the eigenvector matrix.

Another method, known as the QR decomposition, factors the covariance matrix into a series of transformations:

$$\underline{C} = \underline{Q}\underline{R}$$

where $\underline{Q}$ is orthogonal and $\underline{R}$ is upper diagonal. This is based on a transformation known as the Householder transform that reduces columns of a matrix below the diagonal to zero.