Quantifying Similarity and Distance Measures for Vector-Based Datasets: Histograms, Signals, and Probability Distribution Functions

by Mark A Tschopp and Efraín Hernández-Rivera
NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer’s or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.
Quantifying Similarity and Distance Measures for Vector-Based Datasets: Histograms, Signals, and Probability Distribution Functions

by Mark A Tschopp
*Weapons and Materials Research Directorate, ARL*

Efraín Hernández–Rivera
*Oak Ridge Institute for Science & Technology (ORISE), Belcamp, MD*

Approved for public release; distribution is unlimited.
February 2017 Technical Note

Quantifying Similarity and Distance Measures for Vector-Based Datasets: Histograms, Signals, and Probability Distribution Functions

Mark A Tschopp and Efraín Hernández–Rivera

US Army Research Laboratory
ATTN: RDRL-WMM-F
Aberdeen Proving Ground, MD 21005-5069

Approved for public release; distribution is unlimited.

It is often important to characterize the similarity or dissimilarity (distance) between different measured or computed datasets. There are a large number of different possible similarity and distance measures that can be applied to different datasets. In this technical note, a number of different measures implemented in both MATLAB and Python as functions are used to quantify similarity/distance between 2 vector-based datasets. The scripts are attached as appendixes as is a description of their execution.
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>List of Figures</td>
<td>iv</td>
</tr>
<tr>
<td>List of Tables</td>
<td>v</td>
</tr>
<tr>
<td>1. Introduction</td>
<td>1</td>
</tr>
<tr>
<td>2. Function Description</td>
<td>2</td>
</tr>
<tr>
<td>3. Implementation and Usage</td>
<td>11</td>
</tr>
<tr>
<td>4. Examples</td>
<td>11</td>
</tr>
<tr>
<td>5. Summary</td>
<td>18</td>
</tr>
<tr>
<td>6. References</td>
<td>19</td>
</tr>
<tr>
<td>Appendix A. Python Function</td>
<td>21</td>
</tr>
<tr>
<td>Appendix B. MATLAB Function</td>
<td>25</td>
</tr>
<tr>
<td>Distribution List</td>
<td>31</td>
</tr>
</tbody>
</table>
List of Figures

Fig. 1  Example of how different families of metrics are influenced by minor deviations in peak position, peak broadening, peak splitting, and noise (modified peaks, from left to right) for 4 Gaussian curves. The original peaks (top, in green) are compared to the modified peaks (second from top, in green) for the $L_1$ norm metric, the intersection metric, the inner product metric, and the Shannon entropy metrics (in red). .......................... 10

Fig. 2  Example of 3 different XRD patterns, which are slightly offset to show the different peaks. The bottom plots show a magnified view of the master XRD patterns (above). In each of the cases, the intensity has been normalized such that the area is 1. .................................................. 12

Fig. 3  Distance metrics of the $L_p$ family computed for the 3 XRD patterns shown in Fig. 2. The diagonal is showing each XRD pattern compared against itself (i.e., the distance is zero). The remaining distance values have been normalized such that the maximum distance for each metric is equal to 1. ................................................................. 13

Fig. 4  Distance metrics of the $L_1$ family computed for the 3 XRD patterns in Fig. 2 ................................................................. 15

Fig. 5  Distance metrics of the intersection, inner product (cosine), fidelity, and squared euclidean families computed for the 3 regions in the 3 XRD patterns in Fig. 2. The left contour map is of the full XRD pattern and the other 2 are for the 2 magnified views of the peaks (bottom left and bottom right in Fig. 2)................................................................. 17
List of Tables

Table 1 L_p Minkowski family ...............................................................4
Table 2 L_1 family ..............................................................................4
Table 3 Intersection family .................................................................5
Table 4 Inner product family .................................................................6
Table 5 Fidelity family .........................................................................6
Table 6 The \chi^2 family ......................................................................7
Table 7 Shannon’s entropy family ........................................................8
Table 8 Combination family .................................................................8
Table 9 Vicissitude family ...................................................................9
INTENTIONALLY LEFT BLANK.
1. Introduction

As materials science and other communities are becoming increasingly more data-driven, it is important to be able to quantify the similarity or difference between different datasets. Datasets can take any number of different forms: scalar values, 1-dimensional (1-D) vectors, 2-dimensional (2-D) matrices, \(n\)-dimensional matrices, etc. Mathematically, there are a number of different measures for assessing the similarity or dissimilarity (distance) between the different forms of these datasets.\(^1\) For instance, given datasets \(X\) and \(Y\), one common form of representing their distance is the \(L_1\)-norm (i.e., the sum of the absolute difference between each corresponding data point \(\sum_i |X_i - Y_i|\), which is related to the mean absolute error). The Euclidean distance, or \(L_2\)-norm, is the square root of the sum of the squared difference between each corresponding data point \(\left(\sum_i (X_i - Y_i)^2\right)^{1/2}\), related to the root mean squared error. Yet another measure is the \(L_\infty\) norm, or the Chebyshev norm, which is the the maximum distance between the 2 datasets \(\left(\sum_i (X_i - Y_i)^\infty\right)^{1/\infty} \equiv \max|X_i, Y_i|\).

These are just a few distance examples related to the \(L_p\) Minkowski norms and are not indicative of the different ways to characterize the similarity or difference between 2 datasets. There are a large number of different possible similarity and distance measures that can be applied to different datasets.

One important property that the metrics discussed herein share is their high computational efficiency. These metrics perform a bin-to-bin comparison between datasets. Therefore, these metrics scale linearly, proportional to \(O(N)\). However, the bin-to-bin comparison does have some limitations, most notably these metrics do not take the local bin environment into account in quantifying the distance. Hence, more complex metrics have been developed to overcome bin-to-bin metric limitations, but these metrics often come with increased computational cost. Nonetheless, the bin-to-bin metrics described within this technical note are widely used and may have an important role when computing the distance and similarity of large datasets and when considering high-throughput processes.

In this technical note, a number of different measures implemented in both MATLAB and Python as functions are used to quantify similarity/distance between 2 vector-based datasets, which can be representative of vectors of values being compared (e.g., histograms, probability distribution functions, signals). The scripts are attached as appendixes as is a description of their execution.
2. Function Description

The functions used to compute distance or similarity measures require vectors $X$ and $Y$ and return the corresponding similarities or distances per the various measures given hereafter. There are a number of different families of distance and similarity functions, which are given in Tables 1–9 and are briefly discussed hereafter.

- The $L_p$ Minkowski family (Table 1) contains measures related to the generalized formula, $\sqrt[p]{\sum_i |X_i - Y_i|^p}$, where $p$ encompasses everything from the city block $L_1$ distance to the Chebyshev $L_\infty$ distance.

- The $L_1$ family (Table 2) contains measures related to the absolute difference $L_1$ distance (i.e., $\sum_i |X_i - Y_i|$).

- The intersection family (Table 3) contains measures related to the intersection of $X$ and $Y$. The $\min(X, Y)$ or $\max(X, Y)$ term appears in either the denominator or numerator for this family.

- The inner product family (Table 4) contains measures related to the summed inner product, or dot product, of $X$ and $Y$ (i.e., $\sum_i X_i Y_i$).

- The fidelity (or squared-chord) family (Table 5) contains measures related to the sum of the square root of the inner (dot) product (i.e., $\sum_i \sqrt{X_i Y_i}$), which is referred to as the Fidelity similarity.

- The squared $L_2$ (or $\chi^2$) family (Table 6) contains measures related to the square of the $L_2$ (Euclidean) norm (i.e., $\sum_i (X_i - Y_i)^2$). The denominator in some of these measures leads to an asymmetric response if $X$ and $Y$ are swapped.

- The Shannon’s entropy family (Table 7) contains measures related to Shannon’s concept of probabilistic uncertainty or entropy (e.g., $\sum_i \ln X_i$, $\sum_i \ln Y_i$, or some similar form).

- The combination family (Table 8) contains measures that have concepts from multiple families (e.g., the combined average of the $L_1$ and $L_\infty$ norms).

- The vicissitude family (Table 9) contains a number of measures introduced in Cha.²
There are some caveats to their implementation, most notably errors associated with division by zero or taking the log of zero.
### Table 1 \( L_p \) Minkowski family

<table>
<thead>
<tr>
<th>Family</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>City block, ( L_1 )-norm</td>
<td>( d_{\text{City}} = \sum</td>
</tr>
<tr>
<td>Euclidean, ( L_2 )-norm</td>
<td>( d_{\text{Eucl}} = \sqrt{\sum (X_i - Y_i)^2} ) (2)</td>
</tr>
<tr>
<td>Minkowski, ( L_p )-norm</td>
<td>( d_{\text{Mink}} = \sqrt{\sum</td>
</tr>
<tr>
<td>Chebyshev, ( L_{\infty} )-norm</td>
<td>( d_{\text{CV}} = \max_i</td>
</tr>
</tbody>
</table>

### Table 2 \( L_1 \) family

<table>
<thead>
<tr>
<th>Family</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sørensen</td>
<td>( d_{\text{Sø}} = \sum</td>
</tr>
<tr>
<td>Gower</td>
<td>( d_{\text{Gw}} = \sum</td>
</tr>
<tr>
<td>Soergel</td>
<td>( d_{\text{Sg}} = \sum</td>
</tr>
<tr>
<td>Kulczynski</td>
<td>( d_{\text{Kul}} = \sum</td>
</tr>
<tr>
<td>Canberra</td>
<td>( d_{\text{Canb}} = \sum</td>
</tr>
<tr>
<td>Lorentzian</td>
<td>( d_{\text{Lor}} = \sum \ln (1 +</td>
</tr>
<tr>
<td>Intersection</td>
<td>( d_{Is} = \sum</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------------------------------</td>
</tr>
<tr>
<td>Wave Hedges</td>
<td>( d_{WH} = \sum</td>
</tr>
<tr>
<td>Czekanowski</td>
<td>( d_{Cz} = 2 \sum \min (X_i, Y_i)/ \sum (X_i + Y_i) )</td>
</tr>
<tr>
<td>Motyka</td>
<td>( d_{Mo} = \sum \max (X_i, Y_i)/ \sum (X_i + Y_i) )</td>
</tr>
<tr>
<td>Tanimoto</td>
<td>( d_{Ta} = \sum [\max (X_i, Y_i) - \min (X_i, Y_i)]/ \sum \max (X_i, Y_i) )</td>
</tr>
<tr>
<td>Intersection</td>
<td>( s_{Is} = \sum \min (X_i, Y_i) )</td>
</tr>
<tr>
<td>Czekanowski</td>
<td>( s_{Cz} = 2 \sum \min (X_i, Y_i)/ \sum (X_i + Y_i) )</td>
</tr>
<tr>
<td>Motyka</td>
<td>( s_{Mo} = \sum \min (X_i, Y_i)/ \sum (X_i + Y_i) )</td>
</tr>
<tr>
<td>Ruzicka</td>
<td>( s_{Ru} = \sum \min (X_i, Y_i)/ \sum \max (X_i, Y_i) )</td>
</tr>
</tbody>
</table>

Table 3 Intersection family
Table 4 Inner product family

<table>
<thead>
<tr>
<th>Inner product</th>
<th>Inner product family</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_{Ip}$</td>
<td>$s_{Ip} = \sum X_i Y_i$ (20)</td>
</tr>
<tr>
<td>Harmonic mean</td>
<td>$s_{Hm} = 2 \sum X_i Y_i/(X_i + Y_i)$ (21)</td>
</tr>
<tr>
<td>Cosine</td>
<td>$s_{Cos} = \sum X_i Y_i/\sqrt{\sum X_i^2 \sum Y_i^2}$ (22)</td>
</tr>
<tr>
<td>Kumar-Hassebrook</td>
<td>$s_{KH} = \sum X_i Y_i/\sum (X_i^2 + Y_i^2 - X_i Y_i)$ (23)</td>
</tr>
<tr>
<td>Jaccard</td>
<td>$s_{Ja} = \sum X_i Y_i/\sum (X_i^2 + Y_i^2 - X_i Y_i)$ (24)</td>
</tr>
<tr>
<td>Jaccard</td>
<td>$d_{Ja} = \sum (X_i - Y_i)^2/\sum (X_i^2 + Y_i^2 - X_i Y_i)$ (25)</td>
</tr>
<tr>
<td>Dice</td>
<td>$s_{Di} = 2 \sum X_i Y_i/\sum (X_i^2 + Y_i^2)$ (26)</td>
</tr>
<tr>
<td>Dice</td>
<td>$d_{Di} = \sum (X_i - Y_i)^2/\sum (X_i^2 + Y_i^2)$ (27)</td>
</tr>
</tbody>
</table>

Table 5 Fidelity family

<table>
<thead>
<tr>
<th>Fidelity</th>
<th>Fidelity family</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_{Fid}$</td>
<td>$s_{Fid} = \sum \sqrt{X_i Y_i}$ (28)</td>
</tr>
<tr>
<td>Bhattacharyya</td>
<td>$d_{Ba} = - \ln \sum \sqrt{X_i Y_i}$ (29)</td>
</tr>
<tr>
<td>Hellinger</td>
<td>$d_{He} = \sqrt{2} \sum (\sqrt{X_i} - \sqrt{Y_i})^2$ (30)</td>
</tr>
<tr>
<td>Matusita</td>
<td>$d_{Mat} = \sqrt{\sum (\sqrt{X_i} - \sqrt{Y_i})^2}$ (31)</td>
</tr>
<tr>
<td>Squared-chord</td>
<td>$d_{SC} = \sum (\sqrt{X_i} - \sqrt{Y_i})^2$ (32)</td>
</tr>
<tr>
<td>Squared-chord</td>
<td>$s_{SC} = 2 \sum \sqrt{X_i Y_i} - 1$ (33)</td>
</tr>
</tbody>
</table>
Table 6 The $\chi^2$ family

<table>
<thead>
<tr>
<th>Family</th>
<th>Formula (Equation)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Squared Euclidean</td>
<td>$d_{SE} = \sum (X_i - Y_i)^2$ (34)</td>
</tr>
<tr>
<td>Pearson $\chi^2$</td>
<td>$d_{Pea} = \sum (X_i - Y_i)^2 / Y_i$ (35)</td>
</tr>
<tr>
<td>Neyman $\chi^2$</td>
<td>$d_{Ney} = \sum (X_i - Y_i)^2 / X_i$ (36)</td>
</tr>
<tr>
<td>Squared $\chi^2$</td>
<td>$d_{Sq} = \sum (X_i - Y_i)^2 / (X_i + Y_i)$ (37)</td>
</tr>
<tr>
<td>Probabilistic Symmetric $\chi^2$</td>
<td>$d_{Sy} = 2 \sum (X_i - Y_i)^2 / (X_i + Y_i)$ (38)</td>
</tr>
<tr>
<td>Divergence</td>
<td>$d_{Div} = 2 \sum (X_i - Y_i)^2 / (X_i + Y_i)^2$ (39)</td>
</tr>
<tr>
<td>Clark</td>
<td>$d_{Cl} = \sqrt{\sum (</td>
</tr>
<tr>
<td>Additive Symmetric $\chi^2$</td>
<td>$d_{Ad} = \sum (X_i - Y_i)^2 (X_i + Y_i) / X_iY_i$ (41)</td>
</tr>
</tbody>
</table>
Table 7 Shannon’s entropy family

<table>
<thead>
<tr>
<th>Method</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kullback-Leibler</td>
<td>( d_{KL} = \sum X_i \ln (X_i/Y_i) )</td>
</tr>
<tr>
<td>Jeffreys</td>
<td>( d_{Jef} = \sum (X_i - Y_i) \ln (X_i/Y_i) )</td>
</tr>
<tr>
<td>K divergence</td>
<td>( d_{Kdv} = \sum X_i \ln (2X_i/(X_i + Y_i)) )</td>
</tr>
<tr>
<td>Topsøe</td>
<td>( d_{Top} = \sum (X_i \ln (2X_i/(X_i + Y_i)) + Y_i \ln (2Y_i/(X_i + Y_i))) )</td>
</tr>
<tr>
<td>Jensen-Shannon</td>
<td>( d_{JS} = \sum (X_i \ln (2X_i/(X_i + Y_i)) + Y_i \ln (2Y_i/(X_i + Y_i))) / 2 )</td>
</tr>
<tr>
<td>Jensen difference</td>
<td>( d_{Jdf} = \sum ((X_i \ln X_i + Y_i \ln Y_i) / 2 - (X_i + Y_i) / 2 \ln ((X_i + Y_i) / 2)) )</td>
</tr>
</tbody>
</table>

Table 8 Combination family

<table>
<thead>
<tr>
<th>Method</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>Taneja</td>
<td>( d_{Tan} = \sum (X_i + Y_i)/2 \ln((X_i + Y_i)/(2\sqrt{X_iY_i})) )</td>
</tr>
<tr>
<td>Kumar–Johnson</td>
<td>( d_{KJ} = \sum (X_i^2 - Y_i^2)^2/2(X_iY_i)^{3/2} )</td>
</tr>
<tr>
<td>Average((L_1, L_\infty))</td>
<td>( d_{avL} = \sum(</td>
</tr>
</tbody>
</table>
Table 9 Vicissitude family

\[
\begin{align*}
\text{Vicis-Wave Hedges} & & d_{Vwh} & = & \sum |X_i - Y_i|/ \min(X_i, Y_i) & (51) \\
\text{Vicis-Symmetric } \chi^2 & & d_{v\chi_1} & = & \sum (X_i - Y_i)^2/ \min(X_i, Y_i)^2 & (52) \\
\text{Vicis-Symmetric } \chi^2 & & d_{v\chi_2} & = & \sum (X_i - Y_i)^2/ \min(X_i, Y_i) & (53) \\
\text{Vicis-Symmetric } \chi^2 & & d_{v\chi_3} & = & \sum (X_i - Y_i)^2/ \max(X_i, Y_i) & (54) \\
\text{max-Symmetric } \chi^2 & & d_{\text{MaxS}} & = & \max(\sum (X_i - Y_i)^2/ X_i, \sum (X_i - Y_i)^2/ Y_i) & (55) \\
\text{min-Symmetric } \chi^2 & & d_{\text{MinS}} & = & \min(\sum (X_i - Y_i)^2/ X_i, \sum (X_i - Y_i)^2/ Y_i) & (56)
\end{align*}
\]
Figure 1 shows an example of the general form of the similarity/distance measures as applied to Gaussian peaks. In this case, the original (Gaussian) peaks $X$ are compared with the modified peaks $Y$ using 4 different forms within the similarity metrics: $L_1$-norm, intersection, inner product, and Shannon entropy. First, the similarity scalar value $s_i$ is computed as a function of bin value for the 2 top peaks ($X$ and $Y$, respectively). Next (not shown), the individual $s_i$ are summed over all $i$ to produce a single scalar value of similarity, $s = \sum_i s_i (X_i, Y_i)$. Interestingly, different similarity metrics may accentuate different characteristics within the data signal (i.e., peaks in this case). For example, notice how the Shannon entropy accentuates peak shift and broadening but is comparatively not very sensitive to the other peak modifications shown in Fig. 1.

Fig. 1 Example of how different families of metrics are influenced by minor deviations in peak position, peak broadening, peak splitting, and noise (modified peaks, from left to right) for 4 Gaussian curves. The original peaks (top, in green) are compared to the modified peaks (second from top, in green) for the $L_1$ norm metric, the intersection metric, the inner product metric, and the Shannon entropy metrics (in red).
3. Implementation and Usage

The implementation of the 1-D dataset distance/similarity measurements were implemented in MATLAB and Python. The respective function/class can be found in Appendix A and Appendix B, respectively. The MATLAB function compares two 1-D vectors of equal size and returns structured variables with the various similarity and distance metrics. The Python class provides a framework that compares two 1-D vectors of equal size and returns the measured distances per family.

The attached MATLAB function has been tested on MATLAB R2014 and R2015 on a Windows operating system. The Python class has been tested with Python 2.7.* and numpy 1.11.* versions on RHEL (6.8) and MacOSX (El Capitan). The MATLAB code can be executed by completing the following:

- Download the various scripts into the same directory:
  - compute_metrics.m
- Open the script in MATLAB
- Type ‘compute_metrics(X,Y,3)’ at the command prompt to run. The third argument is used for the Minkowski metric in the $L_p$ family (i.e., $p = 3$)

The Python class can be imported as a module (e.g., import PyDIST as dists) and used as instructed within the class “DESCRIPTION”. The following example was generated by using the MATLAB scripts. A detailed analysis on the sensitivity of these metrics and their applicability for quantifying differences in X-ray diffraction (XRD) features is presented by Hernández–Rivera et al.3

4. Examples

As an example, 3 different XRD patterns (Fig. 2) are compared using the different distance and similarity metrics. Three different $2\theta$ ranges (full XRD pattern, $18^\circ$–$25^\circ$ range, and $36^\circ$–$40^\circ$ range) are used to assess the quantitative difference between the 3 XRD patterns and to show how much the metrics change as a function of the range used for the 3 patterns.
Fig. 2 Example of 3 different XRD patterns, which are slightly offset to show the different peaks. The bottom plots show a magnified view of the master XRD patterns (above). In each of the cases, the intensity has been normalized such that the area is 1.

Figure 3 shows the different $L_p$-norm metrics for the full range of the 3 XRD patterns. The first row/column is the bottom pattern in Fig. 2, the second row/column is the middle pattern, and the third row/column is the top pattern. The intersection of the different patterns in the contour plot is the distance between the 2 XRD patterns. Notice that the intersection of each pattern with itself has a distance of zero and the maximum distance is normalized to a value of one, helping to show the comparison between different metrics. For instance, in Fig. 3, all metrics suggest that XRD patterns 1 and 2 are furthest apart (i.e., dissimilar), while patterns 1 and 3 are consistently the most similar.
Fig. 3 Distance metrics of the $L_p$ family computed for the 3 XRD patterns shown in Fig. 2. The diagonal is showing each XRD pattern compared against itself (i.e., the distance is zero). The remaining distance values have been normalized such that the maximum distance for each metric is equal to 1.
Figure 4 shows several different metrics from the $L_1$ family for the 3 XRD patterns. Again, comparing with the distance values from the $L_p$ family in Fig. 3, these metrics also agree in terms of indicating patterns 1 and 2 are the most dissimilar as well as showing that patterns 1 and 3 are the most similar. Interestingly, it can be seen that certain metrics, such as the Kulczynski distance, are more sensitive to changes between the 3 patterns (i.e., the lowest normalized distance of 0.64 is much lower than the other metrics).
<table>
<thead>
<tr>
<th></th>
<th>Sorensen</th>
<th></th>
<th></th>
<th>Gower</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>0.87</td>
<td></td>
<td>1</td>
<td>0.87</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0.96</td>
<td></td>
<td>1</td>
<td>0.96</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.87</td>
<td>0.96</td>
<td></td>
<td>0.87</td>
<td>0.96</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Soergel</th>
<th></th>
<th></th>
<th>Kulczynski</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>0.92</td>
<td></td>
<td>1</td>
<td>0.64</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0.98</td>
<td></td>
<td>1</td>
<td>0.86</td>
</tr>
<tr>
<td></td>
<td>0.92</td>
<td>0.98</td>
<td></td>
<td>0.64</td>
<td>0.86</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Canberra</th>
<th></th>
<th></th>
<th>Lorentzian</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>0.89</td>
<td></td>
<td>1</td>
<td>0.87</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0.97</td>
<td></td>
<td>1</td>
<td>0.96</td>
</tr>
<tr>
<td></td>
<td>0.89</td>
<td>0.97</td>
<td></td>
<td>0.87</td>
<td>0.96</td>
</tr>
</tbody>
</table>

**Fig. 4** Distance metrics of the L₁ family computed for the 3 XRD patterns in Fig. 2
Figure 5 shows 4 distance metrics from 4 other families of metrics: intersection, inner product, fidelity, and squared euclidean. The different contour plots are for the 3 different ranges depicted in Fig. 2, with the leftmost plots being for the full-range XRD patterns. First, the order of similarity/distance between the full-range patterns is identical for the metrics shown. However, for the 18°–25° range, patterns 2 and 3 are computed to be most dissimilar by the different metrics. Furthermore, the 36°–40° range has mixed results in terms of quantifying the patterns that are most dissimilar. In terms of the most similar XRD patterns, though, the restricted 2θ ranges agree with the computed distances for the full range (i.e., patterns 2 and 3 are consistently calculated as the most similar).
Intersection

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.87</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0.97</td>
<td>0.86</td>
</tr>
<tr>
<td>0.87</td>
<td>0.96</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Cosine

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.8</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0.96</td>
<td>0.8</td>
</tr>
<tr>
<td>0.8</td>
<td>0.94</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fidelity

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.78</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0.95</td>
<td>0.77</td>
</tr>
<tr>
<td>0.78</td>
<td>0.95</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Squared Euclidean

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.75</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0.87</td>
<td>0.7</td>
</tr>
<tr>
<td>0.75</td>
<td>0.9</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fig. 5 Distance metrics of the intersection, inner product (cosine), fidelity, and squared euclidean families computed for the 3 regions in the 3 XRD patterns in Fig. 2. The left contour map is of the full XRD pattern and the other 2 are for the 2 magnified views of the peaks (bottom left and bottom right in Fig. 2).
5. **Summary**

It is often important to characterize the similarity or dissimilarity (distance) between different measured or computed datasets. There are a large number of different possible similarity and distance measures that can be applied to different datasets. In this technical note, a number of different measures implemented in both MATLAB and Python as functions are used to quantify similarity/distance between 2 vector-based datasets. The scripts are attached as appendixes as is a description of their execution.

The *PyDIST.py* code can be downloaded by clicking here.

The *compute_metrics.m* code can be downloaded by clicking here.
6. References


Appendix A. Python Function

This appendix appears in its original form, without editorial change.

Approved for public release; distribution is unlimited.
This module provides a framework to calculate several types of distance metrics to compare two (N,) arrays.

.. Copyright 2016 Efrain Hernandez-Rivera
   Last updated: 2016-09-12 by E. Hernandez-Rivera

Funding Acknowledgement:
.. This research was supported in part by an appointment to the Postgraduate
.. Research Participation Program at the U.S. Army Research Laboratory
.. administered by the Oak Ridge Institute for Science and Education (ORISE)
.. through an interagency agreement between the U.S. Department of Energy
.. and USARL.

```python
import numpy as np
from math import sqrt, log

class Distances(object):
    """ Python distance/similarity module. Currently, includes distances
    from Cha [1].

    Parameters
    ----------
    family: name of the distance family
        minkowski
        L1
        inner
        fidelity
        squaredL2
        shannon
        combination
        vicissitude

    X: array_like
        Reference histogram/distribution

    Y: array_like
        Histogram/distribution to measure distance from X

    Returns
    -------
    distances: dict
        Dictionary containing distances for all the family members as
        outlined by Cha

    Usage
    -----

    >>> import PyDIST as distance
```

Approved for public release; distribution is unlimited.
$$
>>> \text{dist} = \text{distance.Distances([1,2,3],[4,6,8])}

>>> \text{mink} = \text{dist.minkowski()}

References
---------

***

def __init__(self, P, Q):
    if sum(P) < 1e-20 or sum(Q) < 1e-20:
        raise "One or both vector are zero (empty)...
    if len(P) != len(Q):
        raise "Arrays need to be of equal sizes...

    # use numpy arrays for efficient coding
    P = np.array(P, dtype=float); Q = np.array(Q, dtype=float)

    # Correct for zero values
    P[np.where(P<1e-20)] = 1e-20
    Q[np.where(Q<1e-20)] = 1e-20

    self.P = P
    self.Q = Q

def minkowski(self, n=1):
    P = self.P; Q = self.Q
    return {'Euclidean': sqrt(sum((P-Q)*(P-Q))),
            'City Block': sum(abs(P-Q)),
            'Minkowski': (sum(abs(P-Q)**n))**(1./n),
            'Chebyshev': max(abs(P-Q))}

def L1(self):
    P = self.P; Q = self.Q
    A = sum(abs(P-Q)); d = len(P)
    return {'Sorensen': A/sum(P+Q),
            'Gower': A/d,
            'Sorgel': A/sum(np.maximum(P,Q)),
            'Kulczynski': A/sum(np.minimum(P,Q)),
            'Canberra': sum(abs(P-Q)/(P+Q)),
            'Lorentzian': sum(np.log(1+abs(P-Q)))}

def inter(self):
    P = self.P; Q = self.Q
    A = sum(abs(P-Q)); maxPQ = sum(np.maximum(P,Q))
    return {'Intersection': 0.5*A,
            'Wave Hedges': sum(abs(P-Q)/np.maximum(P,Q)),
            'Czekanowski': A/sum(P+Q),
            'Motyka': maxPQ/sum(P+Q),
            'Ruzicka': 1-sum(np.minimum(P,Q))/maxPQ,
            'Tanimoto': sum(np.maximum(P,Q)-np.minimum(P,Q))/maxPQ}

def inner(self):
    P = self.P; Q = self.Q
    ip = sum(P*Q); p2 = sum(P*P); q2 = sum(Q*Q)
    d = len(P)
    return {'Inner Product': 1-ip, ...
'Harmonic Mean':1-2*\sum(P*Q/(P+Q)),
'Cosine' :1-ip/(\sqrt{p2} * \sqrt{q2}),
'Jaccard' :\sum((P-Q) *(P-Q))/(p2+q2-ip),
'Dice' :\sum((P-Q)*(P-Q))/(p2+q2)}

def fidelity(self):
P=self.P; Q=self.Q; fid=sum(np.sqrt(P*Q))
return {'Fidelity' :1-fid,
'Bhattacharyya':-log(fid),
'Hellinger' :2*sqrt(1-fid),
'Matusita' :sqrt(2-2*fid),
'Squared-Chord':\sum((np.sqrt(P)-np.sqrt(Q))**2))}

def squaredL2(self):
P=self.P; Q=self.Q; d=len(P)
return {'Squared Euclidean':\sum((P-Q)**2),
'Pearson Chi':\sum((P-Q)**2/P),
'Neyman Chi' :\sum((P-Q)**2/(P+Q)),
'Squared Chi':\sum((P-Q)**2/(P+Q)**2),
'Divergence' :2*\sum((P-Q)**2/(P+Q)**2),
'Clark' :sqrt(\sum{(abs(P-Q)/(P+Q)))**2})}

def shannon(self):
P=self.P; Q=self.Q
return {'Kull-Leiber':\sum(P*np.log(P/Q)),
'Jeffreys' :\sum((P-Q)*np.log(P/Q)),
'Kdivergence':\sum(P*np.log(2*P/(P+Q))),
'Topsoe' :\sum(P*np.log(2*P/(P+Q))*Q*np.log(2*Q/(P+Q))),
'Jensen-Shan':0.5*\sum(P*np.log(2*P/(P+Q))
+Q*np.log(2*Q/(P+Q))),
'Jensen-Diff':0.5*\sum(P*np.log(P)+Q*np.log(Q)
-(P+Q)*np.log((P+Q)/2.))}

def combination(self):
P=self.P; Q=self.Q
return {'Taneja' :0.5*\sum((P+Q)*np.log((P+Q)/(2.*np.sqrt(P*Q)))),
'Kumar-John':\sum((P+Q)**2/(2.*np.sqrt(P+Q))),
'AverageL' :0.5*{\sum(abs(P-Q)))+max(abs(P-Q))}

def vicissitude(self):
P=self.P; Q=self.Q; p=sum((P-Q)*(P-Q)/P); q=sum((P-Q)*(P-Q)/Q)
pqmin=np.minimum(P,Q)
return {'Vicis-Wave Hedge':\sum(abs(P-Q)/pqmin),
'Vicis-Symm Chi1' :\sum((P-Q)*(P-Q)/pqmin**2),
'Vicis-Symm Chi2' :\sum((P-Q)*(P-Q)/pqmin),
'Vicis-Symm Chi3' :\sum((P-Q)*(P-Q)/np.maximum(P,Q)),
'Max-Symm Chi' :\max{p,q},
'Min-Symm Chi' :\min{p,q}
Appendix B. MATLAB Function

This appendix appears in its original form, without editorial change.

Approved for public release; distribution is unlimited.
function [D,S,Error] = compute_metrics(P,Q,n)

% MA Tschopp
% Purpose: Implementation of Cha (2007) similarity/distance metrics
% Date: Sept 2016

{%
% For test purposes:
P = rand(1,10); P = P/sum(P);
Q = rand(1,10); Q = Q/sum(Q);
P(1) = 0;
Q(2) = 0;
Q(3)=P(3);
P = P/sum(P);
Q = Q/sum(Q);
Q(3)=P(3);
n = 3;
[D1,S1,~] = compute_metrics(P,Q,n)
P(P==0)=1e-20;
Q(Q==0)=1e-20;
P(P==Q)=P(P==Q)+1e-20;
P = P/sum(P);
Q = Q/sum(Q);
[D,S,~] = compute_metrics(P,Q,n)
%
% Correct for divide by zero
P1 = P; Q1 = Q;
P1(P==0)=1e-20;
Q1(Q==0)=1e-20;
P1(P==Q)=P1(P==Q)+1e-20;
D=[]; S=[]; Error = [];
%
% Lp Minkowski family
D.euclidean = sqrt(sum((P-Q).^2)); %sqrt(dot(P-Q,P-Q));
D.cityblock = sum(abs(P-Q));
D.minkowski = sum(abs(P-Q).^n)^(1/n);
D.chebyshev = max(abs(P-Q));

% L1 family
D.sorensen = sum(abs(P-Q))/sum(P+Q);
D.gower = sum(abs(P-Q))/length(P);
D.soergel = sum(abs(P-Q))/sum(max(P,Q));
if sum(min(P,Q))~=0
    D.kulczynski_d = sum(abs(P-Q))/sum(min(P,Q));
else
    D.kulczynski_d = sum(abs(P1-Q1))/sum(min(P1,Q1));
    Error.kulczynski_d = 'Divide by Zero Error';
end
if min(P+Q)==0
    D.canberra = sum(abs(P-Q)./(P+Q));
else
    D.canberra = sum(abs(P1-Q1)./(P1+Q1));
end
%
%}

Approved for public release; distribution is unlimited.
Error. canberra = 'Divide by Zero Error';
end
D.lorentzian = sum(log(1+abs(P-Q)));

% Intersection family
S.intersection = sum(min(P,Q));
D.intersection = 1-sum(min(P,Q));
if min(max(P,Q))~=0
    D.wavehedges = sum(1-min(P,Q)./max(P,Q));
else
    D.wavehedges = sum(1-min(P,Q1)./max(P1,Q1));
    Error.wavehedges = 'Divide by Zero Error';
end
S.czekanowski = 2*sum(min(P,Q))/sum(P+Q);
D.czekanowski = 1-2.*sum(min(P,Q))/sum(P+Q);
S.motyka = sum(min(P,Q))/sum(P+Q);
D.motyka = 1-sum(min(P,Q))/sum(P+Q);
if sum(abs(P-Q))~=0
    S.kulczynski_s = sum(min(P,Q))/sum(abs(P-Q));
else
    S.kulczynski_s = sum(min(P,Q1))/sum(abs(P1-Q1));
    Error.kulczynski_s = 'Divide by Zero Error';
end
if sum(min(P,Q))~=0
    D.kulczynski_s = 1/S.kulczynski_s;
else
    D.kulczynski_s = 1/(sum(min(P1,Q1))/sum(abs(P1-Q1)));
    Error.kulczynski_s = 'Divide by Zero Error';
end
S.ruzicka = sum(min(P,Q))/sum(max(P,Q));
D.ruzicka = 1-S.ruzicka;
D.tanimoto = (sum(P)+sum(Q)-2*sum(min(P,Q)))/(sum(P)+sum(Q)-sum(min(P,Q)));

% Inner product family
S.inner = dot(P,Q);
D.inner = 1-S.inner;
if min(P+Q)~=0
    S.harmonic = 2*sum((P.*Q)./(P+Q));
    D.harmonic = 1-S.harmonic;
else
    S.harmonic = 2*sum((P1.*Q1)./(P1+Q1));
    D.harmonic = 1-S.harmonic;
    Error.harmonic = 'Divide by Zero Error';
end
S.cosine = dot(P,Q)/sqrt(dot(P,P)*dot(Q,Q));
D.cosine = 1-S.cosine;
S.kumarh = dot(P,Q)/(dot(P,P)+dot(Q,Q)-dot(P,Q));
D.kumarh = 1-S.kumarh;
S.jaccard = dot(P,Q)/(dot(P,P)+dot(Q,Q)-dot(P,Q));
D.jaccard = dot(P-Q,P-Q)/(dot(P,P)+dot(Q,Q)-dot(P,Q));
S.dice = 2*dot(P,Q)/(dot(P,P)+dot(Q,Q));
D.dice = 1-2*dot(P,Q)/(dot(P,P)+dot(Q,Q));

% Fidelity (square chord) family

Approved for public release; distribution is unlimited.
S.fidelity = sum(sqrt(P.*Q));
D.fidelity = 1-S.fidelity;
if sum(sqrt(P.*Q))~=0
    D.bhattacharrya = -log(sum(sqrt(P.*Q)));%log(0) Error
else
    D.bhattacharrya = -log(sum(sqrt(P1.*Q1)));%log(0) Error
end
D.hellinger = sqrt(2*sum((sqrt(P)-sqrt(Q)).^2));
D.squaredchord = dot(sqrt(P)-sqrt(Q),sqrt(P)-sqrt(Q));
S.squaredchord = 2*sum(sqrt(P.*Q))-1;
D.matusita = sqrt(dot(sqrt(P)-sqrt(Q),sqrt(P)-sqrt(Q)));% Squared L2 (chi-squared) family
D.squaredeuclidean = dot(P-Q,P-Q);
if min(Q) ~= 0
    D.pearsonchi = sum((P-Q).^2./Q); %=pearsonchi(Q,P)
else
    D.pearsonchi = sum((P1-Q1).^2./Q1); %Divide by Zero Error
    Error.pearsonchi = 'Divide by Zero Error';
end
if min(P) ~= 0
    D.neymanchi = sum((P-Q).^2./P); %=pearsonchi(Q,P)
else
    D.neymanchi = sum((P1-Q1).^2./P1); %Divide by Zero Error
    Error.neymanchi = 'Divide by Zero Error';
end
if min(P+Q)~=0
    D.squaredchi = sum((P-Q).^2./(P+Q));%Divide by Zero Error
else
    D.squaredchi = sum((P1-Q1).^2./(P1+Q1));%Divide by Zero Error
end
D.probsymm = 2*D.squaredchi;
if min(P+Q)~=0
    D.divergence = 2*sum((P-Q).^2./(P+Q).^2));
    D.clark = sqrt(sum((abs(P-Q)./(P+Q)).^2));%Divide by Zero Error
else
    D.divergence = 2*sum((P1-Q1).^2./(P1+Q1).^2));
    D.clark = sqrt(sum((abs(P1-Q1)./(P1+Q1)).^2));%Divide by Zero Error
    Error.divergence = 'Divide by Zero Error';
    Error.clark = 'Divide by Zero Error';
end
if min(P.*Q)~=0
    D.additivesymm = sum((P-Q).^2.*(P+Q)./(P.*Q));%Divide by Zero Error
else
    D.additivesymm = sum((P1-Q1).^2.*(P1+Q1)./(P1.*Q1));%Divide by Zero Error
end
% Shannon's entropy family

if min(Q)~=0 && min(P./Q)~=0
    D.kullback_PQ = sum(P.*log(P./Q));
else
    D.kullback_PQ = sum(P1.*log(P1./Q1));
end

Approved for public release; distribution is unlimited.
```matlab
D.kullback_PQ = sum(P1.*log(P1./Q1));
Error.kullback_PQ = 'Divide by Zero or log(0) Error';
end
if min(P) == 0 && min(Q./P) == 0
    D.kullback_QP = sum(Q.*log(Q./P));
else
    D.kullback_QP = sum(Q1.*log(Q1./P1));
    Error.kullback_QP = 'Divide by Zero or log(0) Error';
end
if min(Q) == 0 && min(P./Q) == 0
    D.jeffreys = sum((P-Q).*log(P./Q));
else
    D.jeffreys = sum((P1-Q1).*log(P1./Q1));
    Error.jeffreys = 'Divide by Zero or log(0) Error';
end
if min(P+Q) == 0 && min(P./(P+Q)) == 0
    D.kdivergence = sum(P.*log(2*P./(P+Q)));
else
    D.kdivergence = sum(P1.*log(2*P1./(P1+Q1)));
    Error.kdivergence = 'Divide by Zero or log(0) Error';
end
if min(Q) == 0 && min(P./Q) == 0 && min(P) == 0 && min(Q./P) == 0
    D.jensen_s = 0.5*(D.kullback_PQ+D.kullback_QP);
else
    D.jensen_s = 0.5*(D.kullback_PQ+D.kullback_QP);
    Error.jensen_s = 'Divide by Zero or log(0) Error';
end
if min(P+Q) == 0 && min(P./(P+Q)) == 0 && min(Q./(P+Q)) == 0
    D.topsoe = sum(P.*log(2*P./(P+Q)));
else
    D.topsoe = sum(P1.*log(2*P1./(P1+Q1)));
    Error.topsoe = 'Divide by Zero or log(0) Error';
end
% Combinations
if min(dot(P,Q)) == 0 && min((P+Q)./sqrt(dot(P,Q))) == 0
    D.taneja = sum(0.5*(P.*log(P)+Q.*log(Q)-(P+Q).*log(0.5*(P+Q))));
else
    D.taneja = sum(0.5*(P1.*log(P1)+Q1.*log(Q1)-(P1+Q1).*log(0.5*(P1+Q1))));
    Error.taneja = 'Divide by Zero or log(0) Error';
end
if min(P) == 0 && min(Q) == 0
    D.kumarj = 0.5*sum((P.^2-Q.^2).^2./(P.*Q).^(3/2));
else
    D.kumarj = 0.5*sum((P1.^2-Q1.^2).^2./(P1.*Q1).^(3/2));
    Error.kumarj = 'Divide by Zero Error';
end
```

Approved for public release; distribution is unlimited.
D.avgL = 0.5*(sum(abs(P-Q))+max(abs(P-Q)));

% Vicissitude
if min(min(P,Q)) ~= 0
    D.viciswave = sum(abs(P-Q)./min(P,Q));
    D.vicissymm1 = sum((P-Q).^2./min(P,Q).^2);
    D.vicissymm2 = sum((P-Q).^2./min(P,Q));
else
    D.viciswave = sum(abs(P1-Q1)./min(P1,Q1));
    D.vicissymm1 = sum((P1-Q1).^2./min(P1,Q1).^2);
    D.vicissymm2 = sum((P1-Q1).^2./min(P1,Q1));
    Error.viciswave = 'Divide by Zero Error';
    Error.vicissymm1 = 'Divide by Zero Error';
    Error.vicissymm2 = 'Divide by Zero Error';
end

if min(max(P,Q)) ~= 0
    D.vicissymm3 = sum((P-Q).^2./max(P,Q));
else
    D.vicissymm3 = sum((P1-Q1).^2./max(P1,Q1));
    Error.vicissymm3 = 'Divide by Zero Error';
end

if min(Q) ~= 0 && min(P) ~= 0
    D.maxsymm = max(D.pearsonchi,D.neymanchi);
    D.minsymm = min(D.pearsonchi,D.neymanchi);
else
    D.maxsymm = max(D.pearsonchi,D.neymanchi);
    D.minsymm = min(D.pearsonchi,D.neymanchi);
    Error.maxsymm = 'Divide by Zero Error';
    Error.minsymm = 'Divide by Zero Error';
end

end
INTENTIONALLY LEFT BLANK.