Visualisation and Topological Aspects of Higher Dimensional Data

Final Report for CS39440 Major Project

Author: Samuel Jackson (slj11@aber.ac.uk)
Supervisor: Prof. Reyer Zwiggelaar (rrz@aber.ac.uk)

May 3, 2015
Version: 1.1 (Draft)

This report was submitted as partial fulfilment of a MEng degree in Software Engineering (G601)

Department of Computer Science
Aberystwyth University
Aberystwyth
Ceredigion
SY23 3DB
Wales, UK

Declaration of originality

In signing below, I confirm that:

- This submission is my own work, except where clearly indicated.
- I understand that there are severe penalties for plagiarism and other unfair practice, which can lead to loss of marks or even the withholding of a degree.
- I have read the sections on unfair practice in the Students' Examinations Handbook and the relevant sections of the current Student Handbook of the Department of Computer Science.
- I understand and agree to abide by the University's regulations governing these issues.

Signature .....

Date .....

Consent to share this work

In signing below, I hereby agree to this dissertation being made available to other students and academic staff of the Aberystwyth Computer Science Department.

Signature .....

Date .....

Ethics Form Application Number

The Ethics Form Application Number for this project is: 1120.

110036072

Acknowledgements

I am grateful to...

I'd like to thank...

Abstract

Include an abstract for your project. This should be no more than 300 words.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Background & Objectives

1.1 Mammography

Breast cancer is the leading cause of death among women and is the most common form of cancer found in women [59]. Early screening of breast cancer using mammography has been shown to reduce the mortality rate of women [45, 60].

Mammography is the analysis of female breast tissue through the use of X-ray radiology with the goal of producing high resolution images of the structure within the female breast. The com- position of the parenchymal patterns and tissue density revealed by in a mammographic evaluation can be used in the early detection of breast cancer.

Qualitatively speaking the composition of breast tissue can be split into four distinct categories. These are Nodular densities (corresponding to Terminal Ductal Lobular Units (TDLUs), linear densities (corresponding to ducts, vessels, and fibrous strands), homogeneous, structureless densities (corresponding to fibrous supporting tissue), and radiolucent areas (corresponding to adipose tissue) [63]. Typical markers used in the detection of cancer can are the presence of clusters of micro-calcifications, masses, architectural distortions, breast density and parenchymal patterns [40, 56].

There are two main projections used in X-ray mammography. These are known as craniocaudal (CC) and mediolateral oblique (MLO). Craniocaudal view is the ”bottom-up” view of the breast and is the best method for visualising the medial aspect of breast tissue [20]. The mediolateral oblique projection is a ”side-on” view of the breast that provides maximum visualisation of the breast tissue in its entirety but is limited in its ability to visualise the inner breast tissue [20].

1.1.1 Mammogram Acquisition

The difference between high risk and low risk breasts is inherently small in mammogram due to the low level tissue contrast between the two classes [34]. Because of this most essential part of mammographic imaging to ensure that a high contrast resolution is achieved in order to successfully capture the fine detail of the internal structures.

Mammopgraphic imaging has historically been carried using film but is now more commonly performed using digital mammography. Regardless of the medium the general technique for acquisition remains the same. The breast is placed on a plate between the X-ray tube and the detector. The breast is compressed from its normal conical shape onto the plate. This improves imaging by helping to ensure the X-ray attenuation through the breast tissue is as uniform as possible.

Traditional film acquisition is hampered due to the film’s sigmoid shaped response to x-ray exposure

[34]. This can lead to under or over exposure of the film which in turn leads to poor contrast. Digital mammography does not suffer from this issue because the response curve is essentially linear.

1.1.2 Risk Assessment

Mammograms provide a non-invasive means to assess the risk of a patient developing cancer. Several different systems have been developed to aid the classification of mammographic risk based on the parenchymal patterns visible through X-ray mammography.

1.1.2.1 Wolfe

The earliest attempt to classify mammographic risk using parenchymal patterns was suggested by Wolfe [72]. Wolfe proposed a classification system which split patients into four categories depending on the relative visible density of fat, ducts and connective tissue. The four categories are described, in order of lowest to highest risk, in ref. [72] as:

- **N1** - Breast is mostly composed of fat with no visible ducts and very little amounts of dysplasia present.
- **P1** - The parenchyma is primarily composed of fat with up to one quarter of the breast density being composed of visible ducts in the anterior position which may extend into a quadrant.
- **P2** - Breast indicates prominent duct pattern beyond one quarter of the breast that can occupy the entire parenchyma.
- **DY** - Breast is characterised by a severe increase in breast density and often appear as homogenous, missing the duct pattern present in P2 breasts.

1.1.2.2 Boyd

Boyd et al. [7] proposed a quantitive assessment of risk based on increasing classes of mammographic density, known as the six class categories (SCC). These classes are based on the proportion of dense tissue relative to the area of the breast. The six classes are:

- 0%
- >0 to $<10\%$
- 10 to $<25\%$
- 25 to $<50\%$
- 50 to $<75\%$
- $\geq 75\%$

1.1.2.3 Tabár

Tabár et al. [22] proposed a classification scheme which classifies a breast based on the percentage presence of the four building blocks of breast composition [22, 63]. The description of each of the five patterns is given as:

- **Pattern I** - Breast corresponding to pattern I exhibit scalloped contours and cooper's ligaments with evenly scattered TDLU's.
- **Pattern II** - Complete fatty replacement of both
- **Pattern III** - Prominent retroareolar duct pattern and fatty involution.
- **Pattern IV** - Extensive linear and nodular densities present throughout the parenchyma.
- **Pattern V** - Homogeneous, structureless fibrosis with a convex contour.

1.1.2.4 BI-RADS

The Breast Imaging Report and Data System (BI-RADS) [4,18] was developed by the American College of Radiology (ACR) in an attempt to standardise the lexicon used to describe mammography reports during standard screening. BI-RADS classifies the breast into four categories based on density [4].

1. Fatty Breast (<10% of dense tissue)
2. Fibroglandular (10 - 49% of dense tissue)
3. Heterogeneously dense (49 - 90% of dense tissue)
4. Homogeneously dense (>90% of dense tissue)

A radiologist will then classify the breast according to one of 7 categories after interpretation [4]. These are one of:

- Incomplete. Additional evaluation needed.
- Normal.
- Typically benign.
- Probably benign. A shorter interval follow-up is recommended.
- Suspicious Abnormality. Biopsy considered.
- Highly suggestive of malignancy. Biopsy should be performed.
- Histologically proven malignancy.

1.2 Features

Features are higher level descriptive abstractions computed from lower level structure such as areas of high intensity, edges, and corners present within an image. Features are the result of computing a descriptive property about an image from the intensity information contained within it. Hundreds of different types of features have been proposed [13, 21]. Broadly speaking, image features can be split into three distinct categories: shape (or morphological), intensity, and texture features. A brief review of some examples of each type are given in this section.

1.2.1 Intensity Features

Intensity features are possibly the simplest type of feature that can be detected from an image. Intensity features are simple descriptive statistics based on the grey-level histogram of an image or a patch of an image. Examples of such features are the mean, standard deviation/variance, skew, and kurtosis of the grey-level histogram of an ROI [13, 14].

1.2.2 Shape Features

Shape features are features which detected within an image based on the morphological properties of a region of interest (ROI) within an image. The ROI could either be a suspicious artefact present in the mammogram, such as a malignant or benign tumour or cluster of micro-calcifications [13], or it could simply be a property present in the parenchymal structure of the mammogram (which is the approach used in this project) such as regions of high density tissue [12] or prominent linear structures [75].

Typical example of features extracted from shapes of an ROI are the area covered by the shape, the circularity or rectangularity of the shape, and measures based on the perimeter of the shape [50]. Additional features based on the normalised radial length (NRL) [33] such as boundary roughness, mean, entropy and area ratio [50, 51], and statistics based on normalised chord length [73] such as the first four statistical moments of the resulting chord distribution [19].

1.2.3 Texture Features

Texture features are measurements of an image based on the repetition of patterns over an ROI [47]. Features based on the texture of an image are highly desirable because mammograms are obtained using a single medium of acquisition and the spatial distribution of features can be found in a single band [21]. In general, texture features aim to capture information about the spatial arrangement of pixels in an image.

A set of commonly used set of texture features can be derived from the grey-level co-occurrence matrices (GLCM) [24]. Grey-level co-occurrence matrices are used to describe the positions of pixels having similar grey-level values [47]. Pixel pairs within the GLCM are defined over a range of distances between pixels (often simply one) and orientations (often $0°$, $45°$, $90°$, and $135°$). From GLCMs computed for each direction and orientation features representing texture context information such as contrast, homogeneity, energy and entropy [13, 24, 47] can be calculated.

Another approach to texture features uses the grey level difference statistics of a of an image vector. This takes the form of the absolute difference between pairs of grey levels within an image or between the average of local neighbourhoods. Like GLCM features, this technique is parameterised by the distances between the two image patches and the orientation of the distances used as an offset. First order statistics and measures of spread (such as entropy, contrast and angular second moment) can then the derived [70].

Grey level run length statistics are another approach to texture analysis of mammographic images. This technique counts the number of consecutive occurrences of pixels with the same grey level value in a given orientation. Features are based on weighting longer or shorter runs as being more significant can then be created from the run length distribution [13, 70].

One final method of acquiring texture features is the via a family of techniques based on Gabor filters. A Gabor filter kernel is the product of a Gaussian kernel and a sinusoidal function. Gabor filters can be used to model the simple cells in the visual cortex and are therefore thought to be similar to the way humans perceive images [39]. Grigorescu et al [23] compared three different types of texture features derived from Gabor filters such as Gabor energy, complex moments and grating cell operator features.

1.3 Dimensionality Reduction

Collections of features extracted from a set of images form a feature matrix (also known as as feature space) where each row in the matrix corresponds to a single image and each column corresponds to a single feature extracted from that image. Each entry in the matrix contains the value of a specific feature detected for that image. The number of columns in a feature matrix m is known as the dimensionality of feature matrix. The goal of a feature matrix is that it encapsulates key information about what we are aiming to measure, allowing us to make inferences about what the data is telling us.

The field of dimensionality reduction is concerned with reducing the number of dimensions that a dataset has to only preserve the most important ones. In this way dimensionality reduction can be viewed as a feature selection method, discarding irrelevant or noisy dimensions in favour of those which best represent the data.

Dimensionality reduction techniques can be broadly split into two categories: linear and non-linear. As the name suggests linear approaches are restricted to representing data as a linear subspace while non-linear approaches are more versatile and can model more complex manifolds.

Techniques used to perform dimensionality reduction can also be divided into those which are based on the principle or eigendecomposition (sometimes spectral decomposition) [62]. Methods with rely on this principle include Principle Component Analysis (PCA) [58], Isomap [65], and Locally Linear Embedding (LLE) [55] . Dimensionality reduction techniques which rely on eigendecomposition aim to factorise the feature matrix into the form:

$$\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^T \tag{1}$$

Where $\boldsymbol{Q}$ is the the matrix of eigenvectors of $\boldsymbol{A}$ and $\boldsymbol{\Lambda}$ is a diagonal matrix containing the eigenvalues of $\boldsymbol{A}$. The lower dimensional representation of the feature matrix is then given by either the top or bottom eigenvectors in $\boldsymbol{Q}$.

Approaches not based on eigendecomposition include t-SNE and the SNE family [25, 68], Autoencoder based approaches [26], and Curvilinear Component Analysis [15]. SNE approaches focus on minimising the difference between probability distributions representing the higher and lower dimensional space. Autoencoders use a small number of middle nodes in a ANN to learn a compact representation of the data. Curvilinear component analysis builds on Sammon's mapping via minimisation of a cost function between distances in the representation but improves upon it by favouring local topology.

Another way of categorising dimensionality reduction techniques is whether they preserve local or global properties of the manifold. Since by definition of the problem a lower dimensional representation of a space cannot perfectly represent it there will always be a trade off in the accuracy of the mapping and number of discarded dimensions. Some approaches such as Isomap attempt to more accurately preserve the global properties of the data. Others such as t-SNE and LLE attempt to preserve the local neighbourhood structure at the expense of a less accurate representation of the global structure of the manifold.

1.3.1 Curse of Dimensionality

For some applications the dimensionality of a feature matrix may be quite small. Indeed it is trivial to ascertain the relationships between the features in a matrix where m is equal to 2 or 3 by pure visualisation. However this is not the case when the feature matrix contains higher dimensional data where m may be in the order of 100s or 1000s of features.

As the number of features (dimensionality) increases so does the volume of the feature space in which the data points are contained. This causes issues with any technique the requires statistical sig-

nificance because a fixed number of data points will indefinitely become sparsely distributed as the dimensionality of the space increases. This is known as the curse of dimensionality [5].

This causes havoc with algorithms (such as classifiers) that depend on distance metrics such as Euclidean distance. The effect is such that in a very high dimensional space the distance pairs of data points becomes negligible in computations such as the nearest neighbour algorithm. All data points are effectively equally far away so the choice essentially become one of randomness [16].

However, often it is not the case that all of dimensions of a feature space are required in order to obtain a meaningful representation. Quite often the the data points in a feature space correspond to a lower dimensional manifold that is embedded within the higher dimensional feature space. Dimensionality reduction (also referred to as feature selection) techniques can be used to disregard noisy or irrelevant dimensions while still retaining meaningful information present in the higher dimensional space.

1.3.2 Linear

Linear approaches to dimensionality reduction make the assumption that the data in question lies approximately on a linear subspace within the higher dimensional representation. The core idea behind linear dimensionality reduction algorithms is to find the basis vectors representing the lower dimensional subspace.

1.3.2.1 Principle Components Analysis

Principle components analysis aims to preserve the maximal covariance of the data [62]. First the covariance matrix of the feature space is computed via:

$$C = \frac{1}{n}\Sigma \mathbf{x}_i \mathbf{x}_i^T \tag{2}$$

Where C is the co-variance matrix and $\mathbf{x}$ is the feature matrix. Eigendecomposition is then performed on the covariance matrix and to obtain the basis vectors for the subspace. Taking the first n basis vectors creates a linear projection of the data onto a lower dimensional representation.

1.3.2.2 Classic Multidimensional Scaling

Classic multidimensional scaling (MDS) is another commonly used linear dimensionality reduction technique. In contrast to PCA, MDS aims to preserve pairwise distances between points in the lower dimensional representation [62]. Typically classic MDS begins by computing the distance matrix between points as given by

$$\boldsymbol{S}_{ij} = ||\boldsymbol{x}_i - \boldsymbol{x}_j||^2 \tag{3}$$

Since the output this matrix is not necessarily positive semi-definite the matrix first be first transformed into a Gram matrix using the following operation [62]

$$F = -\frac{1}{2}\boldsymbol{H}\boldsymbol{S}\boldsymbol{H} \tag{4}$$

Where $\boldsymbol{H}$ is a centring matrix with the value

$$H = \boldsymbol{I} - \frac{1}{n}\boldsymbol{e}\boldsymbol{e}^T \tag{5}$$

In which I is the identity matrix and $\boldsymbol{e}$ is a vector containing all ones. Performing eigendecomposition on the resulting matrix will result in a matrix of eigenvectors $\{\boldsymbol{q}_j\}_{j=1}^{d}$ and eigenvalues $\{\lambda_j\}_{j=1}^{d}$ as described in the first part of this section. The lower dimensional mapping produced by MDS is then given by

$$Y = \{\sqrt{\lambda_j}\boldsymbol{q}_{ji}\}_{j=1}^{n} \tag{6}$$

1.3.3 Non Linear

Often data points within the feature space do not lie on a linear subspace. Instead they are embedded in a non-linear manifold within a higher dimensional space. In this case taking a linear projection such as that produced by PCA will result in a distorted results with many parts of data overlapping with one another. However, linear techniques such as PCA are often useful as a preprocessing step to remove highly redundant or noisy dimensions before using a non-linear dimensionality reduction method.

1.3.3.1 t-Distributed Stochastic Neighbour Embedding

t-Distributed Stochastic Neighbour Embedding (t-SNE) [68] is a non-linear dimensionality reduction technique which aims is particularly well suited to the visualisation of higher dimensional spaces.

t-SNE is an adaption of the older stochastic neighbour embedding (SNE) algorithm [25]. Stochastic neighbour embedding models distances between points in the higher dimensional dataset as conditional probabilities. The conditional probability $p_{j|i}$ that point x_i would pick x_j as a neighbour if picked in proportion to their probability density under a Gaussian centred at x_i. Formally this is given by the equation

$$p_{j|i} = \frac{e^{-||x_i - x_j||^2/2\sigma^2}}{\sum_{k \neq j} e^{-||x_i - x_k||^2/2\sigma^2}} \tag{7}$$

Similarly the probability for the lower dimensional mapping is given by

$$q_{j|i} = \frac{e^{-||y_i - y_j||^2}}{\sum_{k \neq j} e^{-||x_i - x_k||^2}} \tag{8}$$

Where σ in this case is set to $\frac{1}{\sqrt{2}}$. In a perfect projection the point conditional probabilities p_{ij} and q_{ij} would be identical. SNE attempts to find the best possible mapping by minimising the different between the two conditional probability distributions given by the Kullback-Liebler divergence using gradient descent. The cost function used by gradient descent is given by

$$C = \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}} \tag{9}$$

The parameters of the algorithm are the learning rate for gradient descent and the choice of the initial sigma for the Gaussian. The choice of sigma is dependant on the density of the dataset being visualised and so is generated by means of a binary search for a sigma that produces a probability distribution P_i that matches a fixed perplexity term specified by the user and defined as:

$$Perp(P_i) = 2^{-\sum_i p_{j|i} \log_2 p_{j|i}} \tag{10}$$

t-SNE improves on SNE by solving two key issues with SNE. The first issue is that the cost function for SNE is difficult to optimise. Instead of minimising the sum of Kullback-Liebler divergences between the two conditional probabilities $p_{j|i}$ and $q_{j|i}$ the difference between the joint probability distributions P and Q are minimised instead. This is given by

$$C = \sum_i \sum_j P_{ij} \log \frac{p_{ij}}{q_{ij}} \tag{11}$$

Subsequently the definition for the pairwise similarities changes slightly to

$$p_{ij} = \frac{e^{-||x_i - x_j||^2/2\sigma^2}}{\sum_{k \neq l} e^{-||x_k - x_l||^2/2\sigma^2}} \tag{12}$$

$$q_{ij} = \frac{e^{-||y_i - y_j||^2}}{\sum_{k \neq l} e^{-||x_k - x_l||^2}} \tag{13}$$

To prevent outliers in the higher dimensional space from becoming extremely small and having little influence over the cost function p_{ij} is set to the symmetrised conditional probabilities via $p_{ij} = \frac{p_{j|i} = p_{i|j}}{2n}$.

The second improvement that t-SNE offers is a solution to the effects of the "crowding problem". This is where the area required to faithfully display the data points which are of similar distance from each other in higher dimensional space is much larger than the available space in the lower dimensional representation. t-SNE combats this problem by using the Student's t distribution instead of a Gaussian to convert low dimensional points to probabilities, changing the definition of q_{ij} to

$$q_{ij} = \frac{(1 + ||y_i - y_j||^2)^{-1}}{\sum_{k \neq l} (1 + ||x_k - x_l||^2)^{-1}} \tag{14}$$

The heavier tails present in the Student's t distribution allows for moderately distance points to be forced further out in the lower dimensional mapping.

1.3.3.2 Isomap

Isomap [65] is a fairly old dimensionality reduction algorithm. Isomap is a spectral dimensionality reduction technique that is based on computing the geodesic distances between points embedded in the higher dimensional manifold. Isomap is effectively an extension of metric multi-dimensional scaling (MDS).

Instead of using the euclidean distance metric (as in MDS) a weighted graph between all points is constructed. The distance between two points on the manifold is then given by the shortest path between two nodes in the graph. Typically this graph is constructed by finding the k nearest neighbours to the datapoint and then using Dijkstra's algorithm (or equivalent) to build the graph. One the matrix of geodesic distances has been computed Isomap performs the same steps as classic MDS to achieve an embedding based on geodesic rather than euclidean distance.

One of the biggest issues with Isomap is "short-circuiting". This is where the parameter k for the number of nearest neighbours is set to be too large. In this case a point's k largest neighbour may not be anywhere near the point's actual local neighbourhood resulting in a sub optimum lower dimensional representation.

1.3.4 Locally Linear Embedding

Locally linear embedding (LLE) [55] is another dimensionality reduction algorithm that is based on the principle of eigendecomposition but is substantially different in its approach. While Isomap focusses on the global structure of the embedded manifold locally linear embedding aims to preserve the local structure.

The fundamental assumption made by LLE is that the local neighbourhood surrounding a data point is roughly linear. Through this assumption LLE attempts to model each data point in terms of a matrix of weights required to "recreate" a data point from its neighbours [62].

LLE begins by finding each data point's k nearest neighbours. From the local neighbours a cost function can be defined that yields the reconstruction error of the weights given by

$$\epsilon(\boldsymbol{W}) = \sum_i ||\boldsymbol{x}_i - \boldsymbol{W}_{ij}\boldsymbol{x}_j||^2 \tag{15}$$

Where $\boldsymbol{W}_{ij} = 0$ if $\boldsymbol{x}_j$ is not in the local neighbourhood and $\sum_j \boldsymbol{W}_{ij} = 1$ if it is. Minimising the weight matrix yields a least squares problem described in [55]. After obtaining the optimal weights for reconstructing the data point the matrix $\boldsymbol{W}$ can be used for find the a lower dimensional representation by minimising the equivalent cost function for a lower dimensional representation given as

$$\Phi(\boldsymbol{Y}) = \sum_i ||\boldsymbol{y}_i - \boldsymbol{W}_{ij}\boldsymbol{y}_j||^2 \tag{16}$$

Subject to the constraints that the outputs are centred ($\sum_j \boldsymbol{y}_i = 0$) and the outputs have unit covariance [62]. Eigendecomposition of the feature matrix given by $\boldsymbol{F} = (\boldsymbol{I} - \boldsymbol{W})^T(\boldsymbol{I} - \boldsymbol{W})$ to obtain the bottom $d + 1$ eigenvectors produces the desired embedding.

LLE often produces better results than Isomap especially when the local structure of the embedded manifold is more important than the global structure. The optimisation of LLE is also quicker in comparison, largely due to it's ability to take advantage of sparse matrix algorithms. LLE can also have issues if the chosen neighbourhood is too large or if the density of the higher dimensional space is not uniform.

1.4 Quality Measures

Performing dimensionality reduction on a dataset by definition causes a loss of information. As such, as lower dimensional embedding with always contain differences compared to the higher dimensional representation. The quality of the mapping produced by a dimensionality reduction algorithm can be measured by examining home the neighbourhood of a point in the higher dimensional data changes relative to the lower dimensional representation. In a good representation the neighbours in the higher dimensional space should be almost the same in the lower dimensional space.

A common way of quantifying the quality of dimensionality reduction is by using measures derived from the co-ranking matrix [38]. Informally the co-ranking matrix is a square matrix representing the joint histogram of ranks between the two spaces. In this context the rank of data point j is a number representing its position in the ordered list of data points sorted by distance from point i.

More formally the co-ranking matrix is given by the following definition. Let the higher dimensional dataset be denoted by $X = \{x_1, ..., x_n\} \subset \mathbb{R}^H$ and the lower dimensional set by $Y = \{y_1, ..., y_n\} \subset$

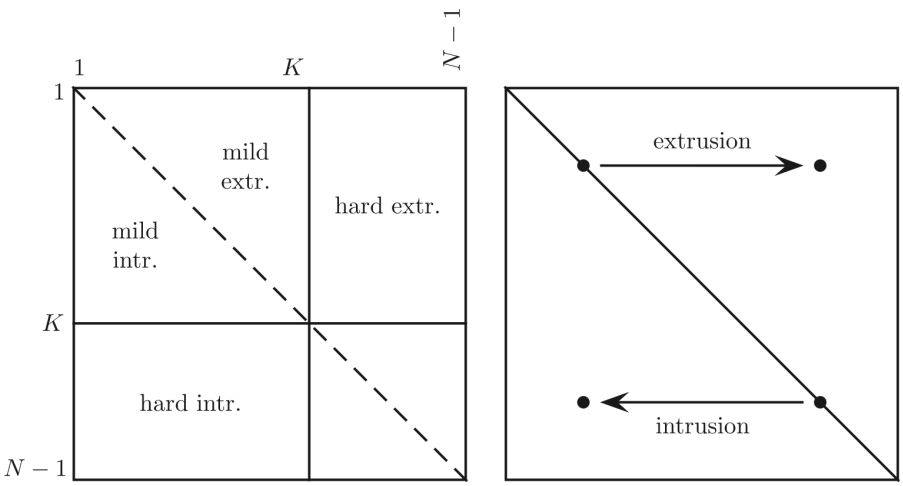


Figure 1.1: Global structure of the co-ranking matrix. Trustworthiness measures hard intrusions and continuity measures hard extrusions in the lower left and upper right submatrix respectively. The local continuity meta-criteria measures the mild intrusion and extrusions in the upper left submatrix. Image source: [41]

$\mathbb{R}^L$. Let δ_{ij} denote the distance between the data point i and j in $\mathbb{R}^H$ and d_{ij} the corresponding distance in $\mathbb{R}^L$. The rank of x_j with respect to x_i is given by

$$\rho_{ij} = |\{k|\delta_{ik} < \delta_{ij} \vee (\delta_{ik} = \delta_{ij} \wedge 1 \leq k < j \leq N\}| \tag{17}$$

Analogously the rank of y_j with respect to y_i is given by

$$rij = |\{k|d_{ik} < d_{ij} \vee (d_{ik} = d_{ij} \wedge 1 \leq k < j \leq N\}| \tag{18}$$

The co-ranking matrix is then defined as the histogram of all rank errors $\boldsymbol{Q} = [q_{kl}]_{1 \leq k,l \leq N-1}$ where

$$q_{kl} = |\{(i,j)|\rho_{ij} = k \wedge r_{ij} = l\}| \tag{19}$$

In an idealised, perfect projection the co-ranking matrix would be a square diagonal matrix. However, because pairs of points will inevitably change their rank between the original representation and the projection errors in ranking will show up as off-diagonal entries in the matrix. A point whose rank decreases ($\rho_{ij} > d_{ij}$) is called an intrusion. Conversely a point whose rank increases ($\rho_{ij} < d_{ij}$) is called an extrusion. A number of different weighted sums of the co-ranking matrix have been proposed aim to measure the properties of the rank errors over local neighbourhoods of size K.

The co-ranking matrix may be further subdivided into four blocks based on the first K rows and columns. Formally this is given by the cartesian product of the index sets $\mathbb{F}_K = \{1, .., K\}$ and $\mathbb{L}_K\{K + 1, ..., N - 1\}$ as follows:

$$\mathbb{UL}_K = \mathbb{F}_K \times \mathbb{F}_K, \mathbb{UR}_K = \mathbb{F}_K \times \mathbb{L}_K \tag{20}$$

$$\mathbb{LL}_K = \mathbb{L}_K \times \mathbb{F}_K, \mathbb{LR}_K = \mathbb{L}_K \times \mathbb{L}_K \tag{21}$$

One set of measures which can be derived from the co-ranking matrix was suggested by Venna and Kaski [32] measures the trustworthiness and continuity of the mapping. If a mapping is trustworthy the

k closest neighbours in the original space will be neighbours in the projected space. Specifically the trustworthiness metric measures how many points in the low dimensional neighbourhood are not in the high dimensional neighbourhood. In the context of the co-ranking matrix trustworthiness measures the "hard" intrusions into the local neighbourhood corresponding to the lower left portion of the co-ranking matrix given by $\mathbb{LL}_{\mathbb{K}}$. Another interpretation is to view this as measuring the number of true positives. The trustworthiness measure is defined in terms of the co-ranking matrix as [37]:

$$T = 1 - \frac{2}{G_K} \sum_{(k,l) \in \mathbb{LL}_K} (k - K) q_{kl} \tag{22}$$

Where G_K is a normalising constant defined as

$$G_K = \begin{cases} NK(2N - 3K - 1) & \text{if } K < N \\ N(N - K)(N - K - 1) & \text{if } K \geq N \end{cases} \tag{23}$$

On the other side of the coin the quality of a mapping can be degraded by discontinuity. Discontinuity is effectively the reverse of the trustworthiness measure. If the neighbours exist in the neighbourhood of the original space but not in the mapping then the nature of the original space may not be preserved due to discontinuities in the projection. In the context of the co-ranking matrix continuity measures the number of "hard" extrusions into the local neighbourhood which corresponds to the upper right proportion of the co-ranking matrix given by $\mathbb{UR}_{\mathbb{K}}$. The measure of continuity in terms of the co-ranking matrix is given by [37]:

$$C = 1 - \frac{2}{G_K} \sum_{(k,l) \in \mathbb{UR}_K} (l - K) q_{kl} \tag{24}$$

Where G_K is the same as in equation **??**.

Another measure proposed by Chen and Buja [10] is the local continuity meta-criteria (LCMC). In contrast to the measures of trustworthiness and continuity which attempts to quantify the error in the projection the LCMC attempts to quantify how well the dimensionality reduction algorithm performed. The LCMC can be viewed as a measure of the true positives. In the context of the co-ranking matrix the LCMC is given by

$$LCMC = \frac{K}{1 - N} + \frac{1}{NK} \sum_{(k,l) \in \mathbb{UL}_K} q_{kl} \tag{25}$$

One final measure that can be derived from the co-ranking matrix is the mean relative rank error (MMREs) proposed by Lee and Verleysen [36]. The MRREs are similar to the measures of trustworthiness and continuity. While trustworthiness and continuity measure the "hard"" intrusions and extrusions in a mapping, MRREs incorporate both "mild" and "hard" intrusions and extrusions. MRREs can be seen as quantifying both types of positives and negatives as opposed to trustworthiness and continuity only quantifying the false positives/negatives [37]. The MRREs are defined given in terms of the co-ranking matrix as

$$W_n(K) = \frac{1}{H_k} \sum_{(k,l) \in \mathbb{UL}_K \cup \mathbb{LL}_K} \frac{|k - l|}{l} q_{kl} \tag{26}$$

$$W_v(K) = \frac{1}{H_k} \sum_{(k,l) \in \mathbb{UL}_K \cup \mathbb{UR}_K} \frac{|k - l|}{l} q_{kl} \tag{27}$$

Where H_k is the normalising factor given by

$$H_k = N \sum_{k=1}^{K} \frac{|N - 2k - 1|}{k} \tag{28}$$

1.5 Visualisation

Traditionally plots of related low dimensional variables can be visualised directly using two or three dimensional scatterplots. However, direct visualisation of data with more than two or three dimensions is impossible with traditional methods. While dimensionality reduction techniques can be used to reduce the dimensionality of data down to two or three dimensions, it is unlikely that two or three dimensions are enough to accurately capture the structure of a complex higher dimensional manifold.

While a higher dimensional dataset cannot be directly visualised in the traditional sense, several different visualisation techniques have been developed that can be used to examine the structure present in higher dimensional data. This sections provides a brief review of several of these techniques.

1.5.1 Scatterplot matrix

A scatterplot matrices are possibly the simplest form of higher dimensional plot. Traditional scatterplots show the correlation between two variables. A scatterplot matrix is an $n \times n$ matrix of scatterplots where each individual plot shows the correlation between two variables i.e. the plot in row i and column j will show the plot of variables X_i and X_j

1.5.2 Parallel Coordinates

Parallel coordinate plots [28] show each variable as a single axis along the bottom of the graph. A single point on the plot is represented as a piecewise line across each of the axes. The position at which a point intersects an axis is the value of that the n-dimensional data point has for that variable.

There are some limitations to parallel coordinate plots. The principle limitation is that the effectiveness of the visualisation is dependant on the ordering of the axes. Different orderings will produce different visualisations. Sometimes the data naturally has a specific ordering (such as time-series data), however this is often not the case. Tatu et al. [64] experimented with using the Hough transform to automatically infer "good" axis orderings. Johansson et al. [30] investigated ranking axis orderings using a combination of clustering, correlation and outlier features on the data. The scaling of each axis is also an important factor in the visualisation, however this can be solved by rescaling the data before visualisation.

1.5.3 Andrews Plot

An Andrews plot defines each n-dimensional data point as a Fourier series with n terms. This visualisation is essentially the same as a parallel coordinates plot with each of the data points interpolated with Fourier interpolation. As such it suffers from the same limitations as parallel coordinates.

1.5.4 RadViz

RadViz [42] is a technique for visualising data points on a plane. RadViz models each n dimensional data point as a single point on a 2D plane surrounded by a unit circle on which each of the n features are placed equal distance apart from one another. Each data point is virtually "connected" to each of the points on the unit circle via a "spring". All of the springs are in equilibrium with one another and the final resting point for a point in the visualisation corresponds to the total strength exerted over point by each spring.

1.6 Analysis

This project aims to investigate the effect of mapping the feature space of both real and synthetic mammograms to a lower dimensional representation. This will involve the extraction of a variety of different shape, intensity, and texture features to produce a high dimensional feature space. We will then use dimensionality reduction techniques to produce a lower dimensional mapping and visualise the results using an appropriate technique.

The hypothesis that we are aiming to test in the project is: Does the lower dimensional mappings of the higher dimensional feature space of synthetic mammograms match those of real patients? The high level questions that this project aims to answer are:

- Are features which are important to real mammogram comparable to those of synthetic mammograms?
- If not, why are they different and does this relate to the author's own limitations?
- Can the knowledge gained be used to influence the perception of what features are important in a real mammogram?
- Could it be used to suggest how to build new mammogram models?

The technical work of the project will be concerned with implementing some existing approaches to feature extraction and dimensionality reduction in mammographic image analysis, but applying it to synthetic mammograms in order to evaluate the similarities and differences.

There are four core components of the methodology that need to be considered before implementation. The first is a decision on the number and type of features to be extracted from the images. Ideally the method used in this project should incorporate approaches that cover all three categories discussed in section **??**. This is so that a good variation in the discriminative properties of an image will be incorporated into the system. Focussing on only one technique for features considerably reduces the information that could be used to discriminate between images.

In this project, it is proposed that shape features are focussed on first. Due to the differences between real and synthetic mammograms, shape features are most likely to be the most successful for comparison between the two datasets. Then the focus will then sift to implementing intensity features and finally texture features. It is proposed that two shape features are used, one which detects blob features proposed by Chen et al. [12] and a linear structure feature proposed by Zwiggelaar et al. [75]. Regarding texture features those which are based on the GLCM [24] are well tested and should be comparatively easy to implement and interpret.

The second is the choice of dimensionality reduction techniques used to produce a lower dimensional mapping. Many different methods have been proposed for dimensionality reduction. It is highly likely relationship between variables in the feature space will not be a linear subspace. For this reason the

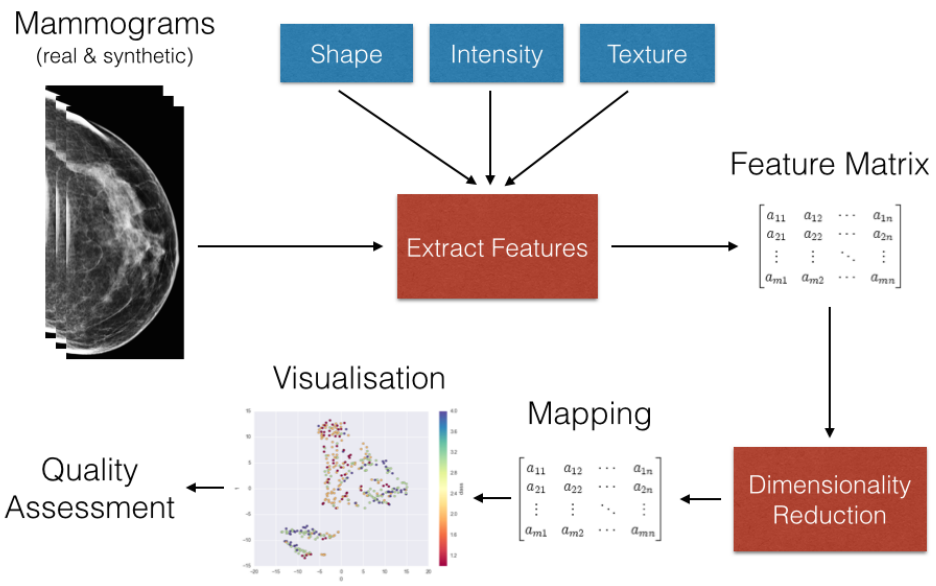


Figure 1.2: Conceptual overview of the image analysis pipeline to be produced as part of this project.

chosen technique will almost certainly be a non-linear dimensionality reduction technique. However, it may be the case that the a linear dimensionality reduction technique such as PCA can be used to remove extremely redundant dimensions and reduce noise as a preprocessing step to a non-linear dimensionality reduction technique.

t-SNE is a good choice for visualisation as it is specifically designed to find embedding which are produce good clusters and and preserve local neighbourhoods. However using just this one approach does not tell us anything about the global structure of the manifold on which the higher dimensional data lies. For understanding the global structure of the system it would be worth examining the projection created from a technique such as Isomap or locally linear embedding. All three of these are common dimensionality reduction approaches that are readily available in many languages and platforms (see section **??**).

Thirdly the lower dimensional mapping must be visualised. This may be done either directly if the mapping is to two or three dimensions. Alternatively the a higher dimensional plot such as those discussed in section **??** could be used to higher examine the dimensional spaces. The choice of visualisation techniques will depend heavily on the choice of the two prior components. It is likely that the project will use a combination of different visualisation strategies depending on what specific aspect of the dataset is being examined. One of the issues with higher dimensional data is that it is impossible to produce visualisation that accurately capture all aspects of the data at once. Whatever technique is used will only ever show a projection of the feature space at best.

The final component of the system will be a form of quality measurement. While visual examination of the mapping is a necessity for qualitatively interpreting how the mapping is derived from the feature space, it is important to gain a quantitive measurement of the mapping. Quantitive measures can be used to check parameters for the dimensionality reduction algorithm are well configured and compare can compare between different combinations of features.

There will be two main results produced by this project. The first is an analysis of how well each of the features extracted and subsequent projections produced the phantom dataset match with those of the real mammogram dataset. The second is an analysis of the quality of the projections produced. This is important as it measures the quality of the

1.6.1 Choice of Language

A technical decision must also be made regarding the choice of technologies and programming languages to be used as part of the project. All four components of the system are likely to be non-trivial to implement. Where possible existing implementations of parts of the four major components of this project should be used. The reasons for this are twofold. 1) using existing implementations should ensure that the development time is kept to a minimum, 2) they are likely to be better tested for validity.

The programming language used as part of the system must be capable of easily implementing high level operations. It would be useful if the language of choice readily includes high level libraries for image processing and dimensionality reduction as well as statistical packages. At the time of writing there are several contenders for potential language choices to be used in this project.

Python [52] is one of the more obvious choices. The language community has several useful libraries for image processing such as scikit-image [69] and OpenCV bindings [29] as well as dimensionality reduction such as scikit-learn [48]. It also has lots of general purpose high level libraries likely to be useful in this project such as NumPy [43], Scipy [57] and Pandas [35]. For visualisation the matplotlib library seems like an obvious choice. An additional strength of python it is a very high level language that is perfect for rapid prototyping. The programmer does not need to concern themselves with the manual memory management or type system as in C, C++ or Java. However, this does not necessarily come with a loss in performance. Libraries such as Numpy and Scipy provide extremely quick precompiled high level operations written in C, allowing us to join performance with simplicity.

A strong alternative contender to Python is the R language [67]. R was originally designed as a statistical programming language but has good support for many of the requirements of this project. Like python, R is a very high level scripting language which has many additional packages that can be installed such as packages for PCA [53], t-SNE [17], LLE [27] etc. and the RIPA (R Image Processing and Analysis module) [49]. For visualisation the excellent ggplot2 [71] library is a clear candidate for this language.

Alternative languages which could be appropriate for this project include Matlab and C++. Like R and Python Matlab [66] is a relatively high level language which has an extensive collection of libraries for data analysis and visualisation as well as support for image processing. However, the major downside to the Matlab application is that it is not free, even on a student license. The performance and object orientation offered by C++ are the major benefits offered and has direct support of the OpenCV library. However, as mentioned above the major downside C++ is the headaches of manual memory management and type system.s

1.7 Research Method

The research method used in this project will be an experimental one. There will be an initial exploratory phase in which features, dimensionality reduction algorithms, visualisations and quality measures are tried and tested. Modifications will be made to existing approaches if necessary. Once an adequate implementation of both features, dimensionality reduction techniques, visualisation, and quality analysis has been achieved I will carry out the experiment with a subset of the real and synthetic mammogram datasets. I will use the largest possible subset from the real and synthetic datasets that I can as part of the experiment in order ensure the best possible representation of the feature space without over representing the subjects.

1.8 Development Methodology

The methodology for the implementation of the project will draw core on ideas from the agile development methodology. This software development methodology is the only one which makes sense for a research orientated project. Research orientated projects are by definition exploratory in nature and the results of the project cannot be stated in concrete at the start of the project. This rules out methodologies like waterfall and feature driven because in both systems the end result of development must be clearly stated up front, with little room for manoeuvre.

Agile approaches embrace change. This is a desirable attribute in a project where we can define the high level end goals of development but where the specifics must be flexible towards finding an approach that works based on analysis of the results from initial prototyping.

As this project is an individual endeavour the full power of an agile approach cannot be fully realised because some of the core principles rely interactions between multiple individuals (such as daily stand-up meetings and paired programming from XP).

There are however many other benefits to an agile approach that are relevant to an individual project. Test driven development (TDD) is a concept that is highly relevant. In software development TDD is used to ensure that you have confidence to make changes and verification that what you have written works. This is especially relevant and desirable in research projects because it not only gives you verification that the software is working, but can also be used to verify that the output of the program and therefore results are correct.

Short iterations and small releases that incrementally add value is another concept that is relevant to an individual project. This provides a measurable indicator of progress throughout the project. In this project I will aim to produce weekly iterations where I will select several stories to work on throughout the week. I will time the start of an iteration to coincide with the meeting with my postdoctoral supervisor who will effectively act as a “customer” to the project. Work for the week will be decided based on discussions during these meetings so this naturally defines an anchor for iteration beginnings and endings.

Programmatic idioms associated with XP are also of value to an individual project. Simplicity (especially the concept of YAGNI) and heavy refactoring are likely to feature heavily in this project. Simplicity ensures that we keep the focus of the project limited to the goals of the research question without adding superfluous code that doesn’t contribute to answering the questions outlined the problem analysis. Refactoring will allow the design of the project to be incremental. We can start with an initial basic design and rewrite and modify the structure of the code when it is needed in order to accommodate problems encountered or other change. In this way a good design (where verification of correctness is controlled by TDD) becomes a natural byproduct of development.

Chapter 2

Experiment Methods

2.1 Overview

The technical outcome of this project was to produce an image analysis pipeline. Broadly speaking the pipeline can be broken into four distinct components. These are feature detection and extraction, dimensionality reduction, quality evaluation, and visualisation.

2.2 Techniques

2.2.1 Preprocessing

Very little preprocessing of the images has been used in this project. Each of the mammograms used has an accompanying binary mask which is used to remove the background and pectoral muscle (see section **??**). The skin around the edge of the breast can cause issues with the blob and line detection due to the intense response during image acquisition. As we are only interested in structure within the parenchyma binary erosion is performed on the breast masks using a disk shaped kernel with a radius of 30 pixels.

2.2.2 Features

In this project we have used three different types of image features to build a feature space from the mammogram datasets. We have used two different types of shape features one to detect blobs and one to detect linear structure. These features intrinsically define a ROI which has some parenchyma pattern of interest. From the ROIs defined by these features we can extract intensity features (based on the image histogram of the ROI) and texture features (in the case of this project GLCM features).

2.2.2.1 Blob Features

The blob detection was achieved by following an approach similar to that described by Chen et al. [11, 12]. This is a multi-scale approach based on building a Laplacian of Gaussian (LoG) pyramid over ten different scales.

To obtain a multi-scale view but retain comparative performance (due to the very large size of mammographic images) instead of increasing the size of the kernel the sigma of the Gaussian is fixed to 8.0 and the image is smoothed and downscaled by a factor or $\sqrt{2}$ instead. Once the image has been convolved with the LoG kernel, the resulting image is finally upscaled to full size before peak detection.

Each of the mammographic images in the real dataset come with a set of breast masks which segment the tissue of the breast from the result of the image (such as the pectoral muscle). This helps to ensure that we are detecting the only within the parenchyma but causes issues due to a large edge response produced from the LoG kernel around the edge of the breast. In Chen's thesis this effect was dealt with by means of a "deformable" convolution. The image is convolved with a standard image kernel when the area of the kernel is entirely within the mask area. When the kernel being convolved falls outside of the mask the filter kernel is modified so that it returns zero outside of the mask and the LoG response normalised by the number of nonzero pixels otherwise.

For each scale image produced the local maxima are detected using maximum filter and a conservative threshold. The resulting position of the peak defines the location of the blob detected in the image while the effective sigma of the Gaussian for the scale of the image σki (where i is the scale and k is the downscale factor equal to $\sqrt{2}$) is the radius of the blob.

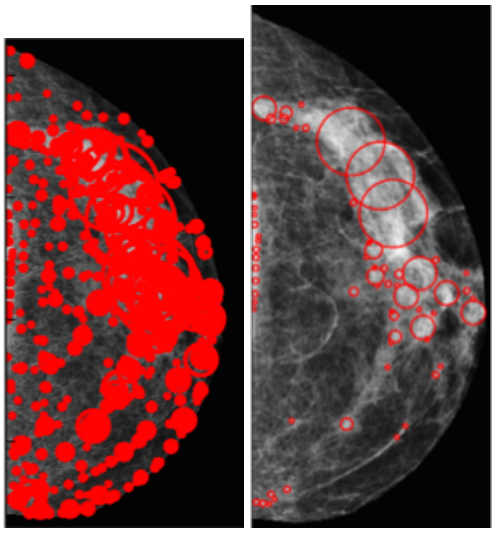


Figure 2.1: Example image showing the raw detections (left) and the results of merging the blobs (right).

This procedure returns a very large number of responses with many overlapping detections. Since we are only interested in blob that best characterises the detected peak, a blob merging strategy is employed to remove redundant blobs. As we are only concerned with blobs that characterise patterns within the parenchyma all blobs whose radius falls outside of the bounds of the image mask are removed.

Next a thresholding technique is used to remove blobs that fall below a certain level of intensity. The area of the image covered by the blob is categorised into 9 clusters. The average intensity of each cluster is computed and the top 5 clusters are selected. The threshold used is defined to be the average intensity of the top five most intense clusters across all blobs less the standard deviation of those same clusters.

In Chen's thesis the clustering is performed using the Fuzzy c-means algorithm while in my implementation I have used the k-means algorithm. The result of this is that the algorithm requires more clusters (5 compared to only the top 3 clusters in Chen's thesis) to achieve comparable results.

After these operations the number of blobs detected is significantly reduced but there are still a large number of blobs which significantly overlap one another in high density regions. To achieve a better representation of the distribution of high density tissue in the mammogram blobs are merged according to how they interact one another. The intersections are classified into one of three categories:

- External: $d \geq r_A + r_B$
- Intersection: $r_A - r_B < d < r_A + r_B$

- Internal: $d \leq r_A + r_B$

Where d is the distance between the two blobs and the r_A and r_B are the radii of blobs A and B. The above definitions assume that $r_A \geq r_B$.

Merging proceeds as follows: if a blob is external then it remains retained. If a blob is internal to a larger (coarser) blob it will be removed. If the blobs intersect one another and they are closely located ($d \leq r_A + \alpha r_B$ for $0 \leq \alpha \leq 1$, assuming $r_A \geq r_B$). In the experiments documented in the report overlap parameter α used was 0.01.

For each of the blobs detected, the coordinates of the detected blob and the radius (i.e. the sigma of the Gaussian associated with the scale the blob was detected at) are saved to a comma separated value (CSV) file. This file was then loaded into the IPython notebook system for analysis. Using the analysis python module developed as part of the general pipeline framework the following features were calculated from the radius and position of the blobs for each image:

- Number of blobs detected
- Average radius
- Standard deviation of the radius
- Min & max radius
- Small/medium/large radius count: For the radius was binned into three separate equally sized bins across the range scales used to detect blobs.
- Density: the average distance between this blob and the k nearest blobs. In all experiments k was set to 4.
- 25/50/75 percentiles
- Count of blobs above the mean.

2.2.2.2 Line Features

Along with shape features based on blobs of high intensity a shape feature based on finding linear structure within a mammogram was used. Linear structure aims to try and characterise the ductal shapes visible in a typical mammogram. For the implementation of this feature we follow the work of Zwiggelaar et al. [75] and use an orientated bins method to pick out ROIs which may otherwise be missed using just blob features alone.

The orientated bins method proposed in ref. [75] filters the local neighbourhood of an image by dividing the neighbourhood window into n angular bins with an angular resolution of $\frac{2\pi}{n}$. The line strength of the local neighbourhood is given by computing the difference between the maximum average intensity of opposing bins and the average intensity of the local neighbourhood as a whole. In the case of a well defined line only one set of opposing bins will give a high response. The orientation of the line can also be derived from the orientation of the bin with the largest intensity.

This operation results in two different filtered images. The first is the line strength image. Each pixel in the line strength image represents the average value from the maximum of the filtering. The second image represents the orientation and each pixels is a number representing the direction that the maximum bin was orientated in.

Once the line strength and orientation images have been generated the results can be enhanced by applying non-maximal suppression [61]. For each pixel in the line strength image non-maximal

suppression selects the two pixels perpendicular to the current pixel based on the corresponding pixel in the line orientation image. If either of the two perpendicular neighbouring pixels are more intense than the current pixel is ignored.

After suppression the image is once thresholded using a conservative value to remove noise and the image is converted to a binary image. A morphological dilation operation is used to improve the connectivity of the line shapes detected from the image by increasing the size very slightly to connect any disconnected components. Any points which are within an 8-connectivity of one another are counted as being part of the same structure.

The resulting shape features can be measured by standard first order statistics to produce descriptive features about the blobs and lines detected from the mammogram. As with blob features the detections were exported to a CSV file and then loaded into a IPython notebook for further analysis. Features created from the distribution of areas detected by the line features were:

- Number of lines detected
- Average area
- Standard deviation of the areas
- Min & max area
- 25/50/75 percentiles
- Count of areas above the mean.

2.2.2.3 Intensity

The shape features detected using orientated bins and multi-scale blobs define regions of interest across the breast. From these ROIs the patch of the image which is covered by the area or radius of the shape feature can be extracted. The histogram of the intensity values of this image patch provide can provide additional discriminative information about ROI. Descriptive statistics derived from the histogram of the ROI were computed. The final features derived were:

- Number of intensity values
- Mean
- Standard deviation.
- Min & max values
- 25/50/75 percentiles
- Skew
- Kurtosis

2.2.2.4 Texture

As with intensity features, texture features can be extracted from the patches defined by the shapes features as well. In this project we have only used texture features derived from the grey-level co-occurrence matrix [24]. The properties computed from the co-occurrence matrices were homogeneity, dissimilarity, energy, and contrast. The definitions of each are given as the following:

Homogeneity:

$$\sum_{i,j=0}^{levels-1} \frac{P_{i,j}}{1+(i-j)^2} \tag{1}$$

Dissimilarity:

$$\sum_{i,j=0}^{levels-1} P_{i,j}|i-j| \tag{2}$$

Energy (or the square root of the angular second moment):

$$\sqrt{\sum_{i,j=0}^{levels-1} P_{i,j}^2} \tag{3}$$

Contrast:

$$\sum_{i,j=0}^{levels-1} P_{i,j}(i-j)^2 \tag{4}$$

The experiments performed in chapter **??** all used a distance of 1 and a combination of eight different orientations of angles 0.0, 22.5, 45.0, 67.5, 90.0, 112.5, 135.0, 157.5 degrees.

Both intensity and texture features will produce some entires zero entries for blobs of larger scales because there were none detected in that image. This causes the projections produced to be split into cluster according to what the size of the largest blob detected was. To prevent this unwanted effect zero entires were replace by the mean value for the relevant scale.

2.2.3 Dimensionality Reduction

In this project the t-SNE as the dimensionality reduction algorithm with which to produce the lower dimensional representations from the higher dimensional feature spaces. The implementation we have used as part of this project is the standard algorithm available through the scikit learn library.

Before running the dimensionality reduction algorithm the input feature matrix is standardised by removing the mean from each feature and scaling to standard variance. This ensures that the all of the features will roughly on the same scale as one another and therefore that the distance metric used to compute neighbours shouldn't be heavily weighted in favour of feature orders of magnitude larger than the others.

t-SNE was chosen as the primary dimensionality reduction algorithm for use in this project because 1) it generally produces reasonably good visual representations when reducing data to 2 or 3 dimensions and 2) it aims to preserve the local neighbourhood to produce representations where points close in the higher dimensional space will be close in the visualisation. This is useful because it can be used to visually show whether the points corresponding to synthetic mammograms appear close to the real mammogram images.

Unfortunately, a major limitation of the t-SNE algorithm is that it make no attempt to preserve the global structure of the underlying manifold in higher dimensional space. For this reason the mappings produced for t-SNE cannot be used to infer anything about the global structure of the manifold.

2.2.4 Visualisation

The visualisation aspects of this project have largely been handled by the inbuilt functionality that available in the matplotlib and pandas Python libraries. However some additional visualisation techniques

have also been implemented for examining what the images look like for each point in the lower dimensional mapping.

2.2.4.1 Visualisation of Images from Mapping

In order to examine the how images change across the lower dimensional mapping and to try and understand why images are grouped closely together a small utility was created that would read in the mapping of the feature space and display a scatter plot of the mapping. When hovering over each point in the visualisation the image corresponding to that data point is displayed to the right of the scatter plot.

This is the only part of the project that was not implemented in python. Due to time constraints it was deemed not worth exerting an huge amount of time on developing this feature, but rather to just get something rapidly working. Ideally this would have been implemented in something like pyQt and utilised the matplotlib library. However, the actual implementation is written in JavaScript, HTML, and CSS. The main driving principle behind this is that it was quicker write some basic CSS to position the image and scatter plot next to one another than it was to write a proper Python GUI from scratch.

2.2.4.2 Median Image Plot

Similarly to the previous visualisation this visualisation takes a projection of the feature space and creates a two dimensional histogram of the points. From each of the resulting bins the median point in both the x and y directions if found. The image which corresponds to this point is selected to be used as part of the visualisation. Each of the images is stitched together to form a matrix of images in the same shape as the lower dimensional mapping. Due to the huge size of mammographic images the images used in this visualisation are downscaled by a factor of 8. This keeps the performance for creating the visualisation reasonable at the expense of loosing some of the finer detail in the images.

2.3 Datasets

In this project we are using two different datasets. One consisting of mammograms for taken from a collection of real patients. The other dataset is a collection of artificially generated breast phantoms generously created by the University of Pennsylvania.

2.3.1 Real Data

The dataset of real mammograms was taken from a private dataset captured using a Hologic full field digital mammography system. The dataset contains images of 90 unique patients each with a craniocaudal and mediolateral oblique view of both the left and right breasts, resulting in 360 total images. Each image in the dataset also has a corresponding binary breast mask. This mask is used to segment the background and the pectoral muscle from the breast parenchyma.

2.3.2 Synthetic Data

The synthetic breast phantoms were generated by the University of Pennsylvania using the techniques outlined by Bakic et al. [1–3]. Their simulation system consists of three major components: a breast tissue model, a compression model, and an acquisition model. Adipose tissue compartments within the breast are modelled using thin shells in areas of primarily adipose tissue and as blobs in areas of

predominantly fibroglandular tissue. Ductal lobes are also simulated by the model using a randomly generated tree.

As the phantoms have not been formally assigned a BI-RADS by an expert radiologist (as in the case of the real mammogram dataset) the ground truth for the risk associated with a particular mammogram is given by it's volumetric breast density (VBD). This information was supplied in the meta-data produced alongside the phantom mammograms.

None of the breast phantoms came with masks, but the masks are an essential part of blob detection. As none of the phantoms contain a pectoral muscle the masks were generated by simply using Otsu's method [46] to create a binary mask for each phantom.

2.4 Comparison of Features

A comparison of the features can be made using the Kolmogorov-Smirnov two sample test. This is a non parametric test used to compare the difference between two probability distributions. In the one sample version of the test an empirical distribution function is compared with a reference cumulative distribution function (such as the normal distribution). The test quantifies the distance between the two distributions. The two sample version of the test is used to measure if two empirical distribution functions differ. Formally, the two sample Kolmogorov-Smirnov test is given by [74]

$$D_{m,n} = \max_x |F(x) - G(x)| \tag{5}$$

Where F and G are cumulative distribution functions of size m and n respectively. The null hypothesis that both samples come from a population that is similarly distributed can be rejected if $D - m, n$ is greater than the critical value for the corresponding significance level $D_{m,n,a}$ which is given by

$$D_{m,n,a} = c(a)\sqrt{\frac{m+n}{mn}} \tag{6}$$

Where $c(a)$ is the inverse Kolmogorov distribution at a. The Kolmogorov-Smirnov two sample test can be applied each feature extracted for both real and phantom mammograms. This will be used to quantitively compare how well each of the features extracted compare to each other.

2.5 Evaluating the Quality of the Mapping

As discussed in section **??** there are several quality metrics that can be used to measure the quality of a mapping produced from a dimensionality reduction algorithm which are independent of the technique being used.

In this project two different types of quality measures derived from the co-ranking matrix are implemented. The chosen metrics are trustworthiness and continuity and the local continuity meta-criteria. These metrics were chosen because together they can be used to measure both the false positives/negatives present in the mapping (for trustworthiness and continuity) and the true positives/negatives (LCMC).

2.6 Implementation

This section provides a brief overview of the implementation details used in the project. All of the methods discussed in the preceding sections were implemented in Python. The technical output of the project is a Python library and command line tool which is used to create the pipeline discussed in the first part of this chapter.

2.6.1 Python Package

The majority of the components used in the project are built upon the top of the scipy stack [31]. Two major additional libraries (which also rely on the scipy stack) which have been used heavily in the project are the scikit image [69] and scikit learn [48] projects.

The library created as part of this project forms a complete python package. The package consists of six top level modules and a collection of submodules implementing specific functionality relating to the features detected by the system. The description of each of the modules is as follows:

- **reduction**: The reduction module implements functions performing multi-processed feature extraction from a dataset.
- **analysis**: The analysis module implements commonly used functions used to analyse the images after feature extraction has been performed. These functions are typically called directly from the python interpreter to in a IPython notebook session.
- **coranking** This module contains the code for computing the co-ranking matrix form two representations of the same dataset.
- **plotting**: This module provides a collection of custom, convenience plotting functions that largely depend on the matplotlib library.
- **io_tools**: The io_tools module implements functions for iterating over directories of images and there corresponding masks as well as image loadings and preprocessing functions.
- **utils**: The utils module contains a collection of miscellaneous helper functions used in various places throughout the package.

An additional sub-package contains the modules used by the reduction module to perform feature extraction these modules include:

- **blobs**: Contains the code implementing blob detection using the approach outlined in section **??**. This also uses an additional private module which provides the code for the graph built as part of the blob merging scheme.
- **linear_structure** Contains the code implementing line feature detection using the approach outlined in section **??**. This also uses a couple of additional private modules within the sub-package which provide the code for non-maximal suppression and the orientated bins feature.
- **texture** Contains the code for computing the GLCM and texture features from ROIs.
- **intensity** Contains the code for computing first order statistics from ROIs.

The last part of the library is the deformable convolution module. This implements the deformable convolution approach outlined by Chen et al. [12]. Due to the performance considerations associated with convolution I chose not to write this module directly in Python but to implement it as a C function which is compiled using the Python C API.

2.6.2 Command Line Interface

The command line tools offered by the program are built using the Click library [54]. The CLI provides a thin wrapper to the some of the higher level library functions available in the package. The most important commands offered by the CLI are those concerned with image processing to collect features from the images. These functions involve iterating over a folder of images and applying the feature extraction techniques outlined in section **??**. These operations can take in the order of several hours to complete and so it is useful to have them exposed on the command so they can be run and left until complete. These commands for image reduction also add the ability to automatically dump the output of the reduction to a CSV file named by the user upon completion.

Other useful commands included in the CLI interface are the ability to run and plot the output of shape feature detection from a single image and the running the t-SNE algorithm on a feature dataset from the command line.

2.6.3 Multiprocessing

The image processing component of the pipeline created for this project is extremely time consuming. The two shape features used in this project take up to a minute to process a single image. The biggest contribution to the running time is the filtering and convolution of the images. Sadly this is an operation that is intrinsically complex and therefore slow. However, since the processing of an image is completely separate from all other images multiprocessing support becomes an option.

The feature detection routines produces as part of the Python package all support multiprocessing through the Python multiprocessing module. Each routine has support to use an unlimited number of threads although in practice the physical limitations of the computer hardware used limited the number to just two. Despite only using one additional process this still managed to speed up the feature detection routines for the whole set of images by several hours.

Chapter 3

Results and Conclusions

This chapter outlines the results of the experiments performed using the methods outlined in the preceding chapter. This section is split into two sections. The first section examines the effect of mapping the higher dimensional feature space to a lower dimensional representation using the methods described in section **??**. The second section presents an investigation into the quality of each of the mappings produced in the first section.

For each of the experiments in this section the full set of 360 real images of both left, right and mediolateral oblique and craniocaudal views. The set of phantom mammograms used in this project consisted of 60 images but each image corresponds to only a single "case". Because of this limitation a single breast phantom from each case was selected at random to be included in the data used for the experiments, brining the total number of images used in all experiments to 366.

Features extracted from both sets of images are compared using a two sample Kolmogorov-Smirnov (KS) test and are reported in the relevant section. Three dimensionality reduction algorithms were tested with each of the different feature sets and visualisation of the feature space in both 2 and 3 dimensions are shown for each.

Two different kinds of quality measure derived from the co-ranking matrix were computed for each of the visualisations shown in this section. The quality metrics used were trustworthiness & continuity and the local continuity meta-criteria up to a neighbourhood of size k.

3.1 Blob features

The results for the two sample KS test between the real and synthetic datasets for blob features detected over 10 scales are shown in table **??**. An explanation of what each type of feature means can be found in section **??**. 2D projections of the blob feature space as produced by t-SNE, Isomap and locally linear embedding are shown in figures **??**, **??**, and **??** respectively.

In each of the figures the real mammograms are shown as coloured dots and the breast phantoms are shown as coloured triangles. The labels corresponding to the colours used in each visualisations are the the image's BIRADS class (in the case of real mammograms) or the volumetric breast density (VBD) (in the case of breast phantoms).

Generally speaking the phantom mammograms are shown to be very different from the real mammograms features used. The distribution of blobs detected in the phantom mammograms are generally larger and contain much less small scale structure when compare to real mammograms. This is reflected in the projections for the feature space.

In the t-SNE projections it can be seen that all of the phantoms blobs, regardless of VBD, are

grouped towards clusters of containing a relatively high proportion of BIRADS risk class 2 & 3 images. The projections produced by Isomap and LLE generally separate the breast phantoms for the original data more than with t-SNE. Several of real mammograms with features that match those of the breast phantoms are pushed also interspersed with the phantom mammograms. This can be seen best in the 3D projection produced by LLE in figure **??**. These real mammograms are ones which tend to have a higher number of larger blobs detected than real mammograms in general.

	KS	p-value
blob_count	0.341667	2.774753e-07
avg_radius	0.847222	1.360953e-42
std_radius	0.711111	3.929143e-30
max_radius	0.363889	3.327680e-08
small_radius_count	0.319444	2.024353e-06
med_radius_count	0.338889	3.583275e-07
large_radius_count	0.733333	5.112303e-32
density	0.169444	4.114705e-02
upper_dist_count	0.345833	1.883393e-07
25%	0.358333	5.726005e-08
50%	0.743056	7.334764e-33
75%	0.777778	5.796838e-36

Table 3.1: Comparison of the Kolmogorov-Smirnov test results for each feature generated from the radii of blobs detected in an image between real and phantom mammograms.

The quality evaluation for each of the visualisations are shown in figures **??**, **??**, and **??**. The quality metrics shown in these figures suggest that the 2D visualisation produced by t-SNE is the most reliable mapping in terms of both trustworthiness & continuity and LCMC. In the case of the 3D projections Isomap provides a better visualisation with it topping all both criteria.

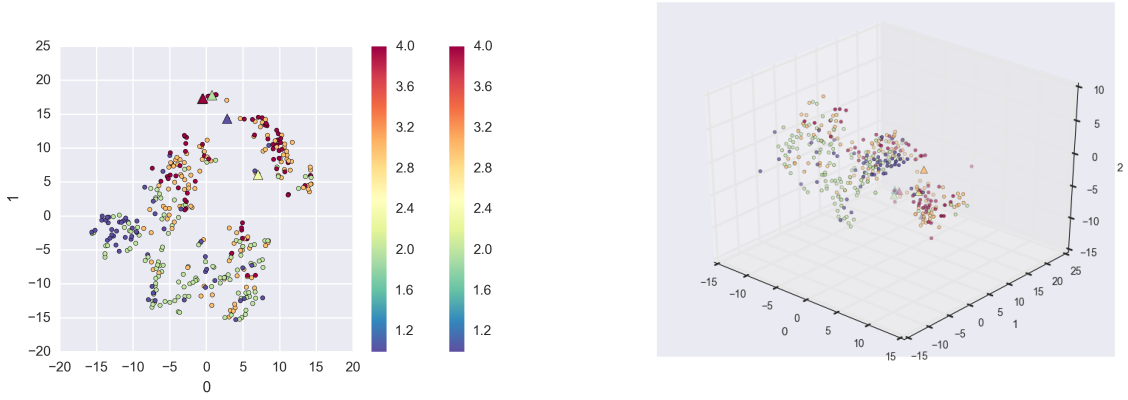


Figure 3.1: 2D & 3D projections of the blob feature space produced by the t-SNE algorithm with a learning rate of 300 and perplexity of 30.

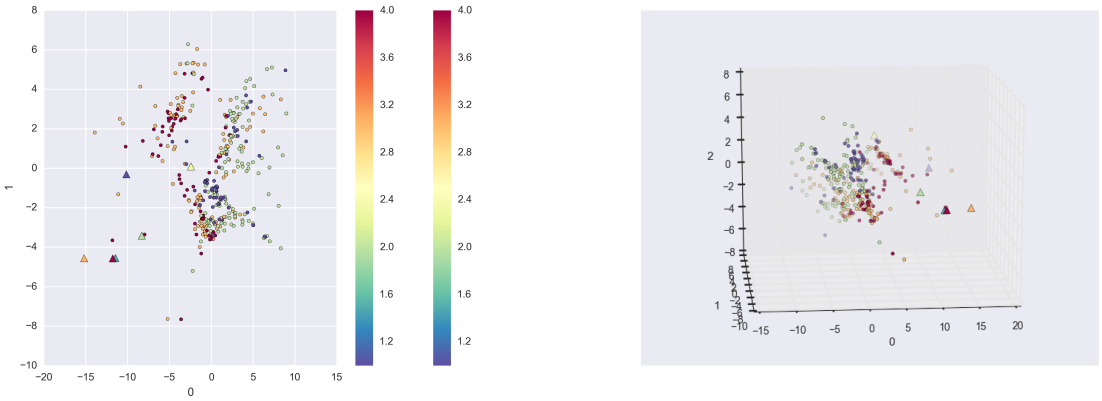


Figure 3.2: 2D & 3D projections of the blob feature space produced by the Isomap algorithm with 10 neighbours.

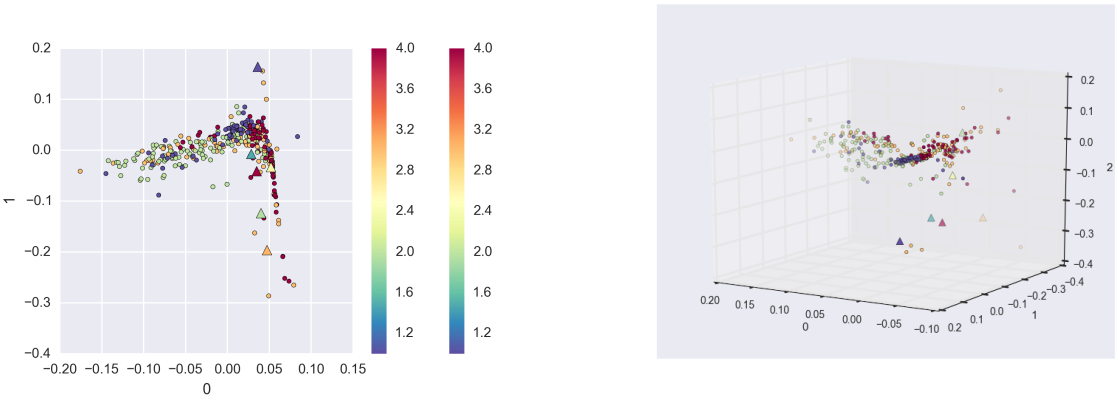


Figure 3.3: 2D & 3D projections of the blob feature space produced by the LLE algorithm with 10 neighbours.

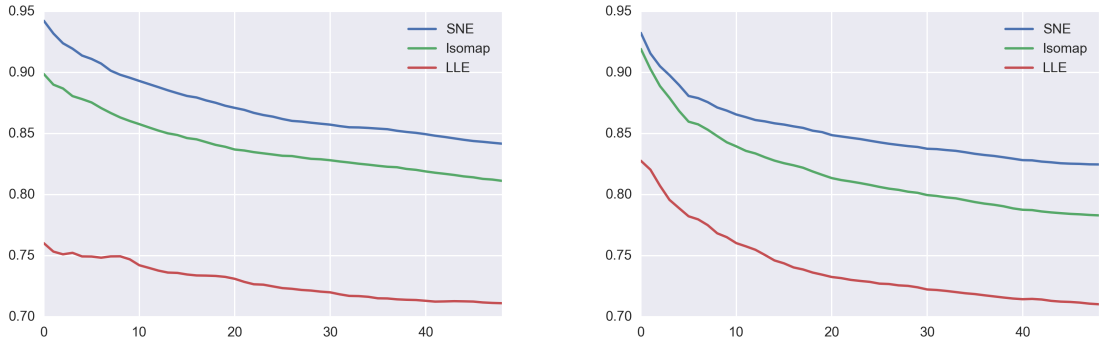


Figure 3.4: Trustworthiness and continuity of the 2D projections produced from blob features.

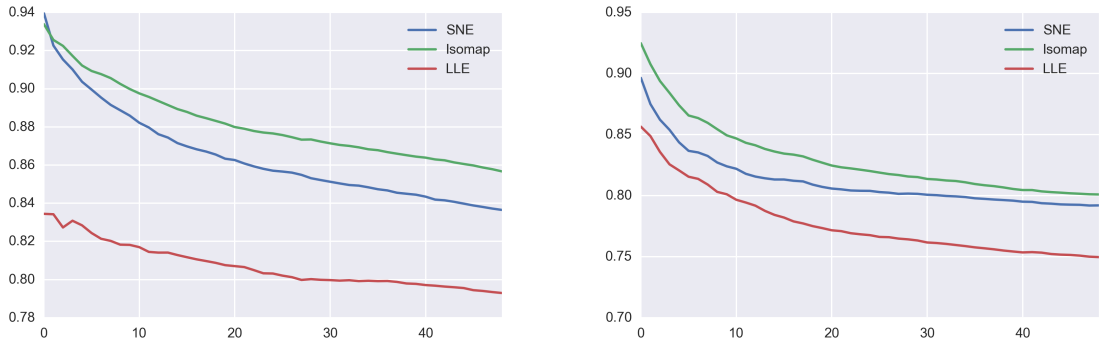


Figure 3.5: Trustworthiness and continuity of the 3D projections produced from blob features.

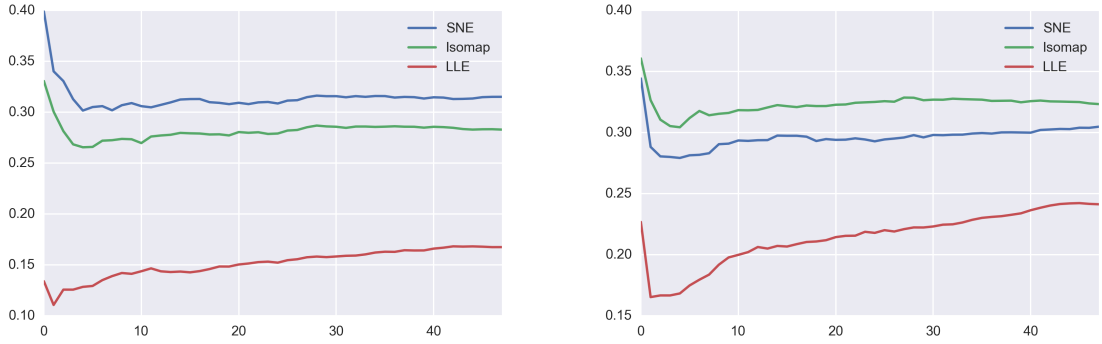


Figure 3.6: LCMC of both the 2D projection (left) and 3D projection (right) of the feature space for blobs.

3.1.1 Line features

Results for the KS two sample test between the two feature spaces can be seen in table **??**. Again, as with blobs, the distributions for real and synthetic mammograms are not in particularly good agreement according to the test. 2D and 3D Projections for the line feature space for each of the three different dimensionality reduction algorithms are shown in figures **??**, **??**, and **??** respectively.

Note that in these results the value of the min area feature was removed. When using the min area feature all projections would produce two clusters. One smaller cluster would be created containing which contained blobs where the min area was much larger than the rest of the dataset. This is most likely caused by a thresholding issue causing some unwanted smaller lines to be retained for some of the images.

	KS	p-value
count	0.933333	1.338265e-51
mean	0.593056	4.356206e-21
std	0.654167	1.450514e-25
25%	0.143056	1.255126e-01
50%	0.304167	7.344875e-06
75%	0.508333	1.320817e-15
max	0.737500	2.231475e-32
skew	0.480556	5.426886e-14
kurtosis	0.506944	1.598384e-15
upper_dist_count	0.913889	1.724254e-49

Table 3.2: Comparison of the Kolmogorov-Smirnov test results for each feature generated from the area of lines detected in an image between real and phantom mammograms.

The t-SNE projection shows that there is a general transition from images which contain a high mean area of linear structure on the left towards images which have a lower mean area on the right. In terms of BIRADS risk it can be seen that very low and very high risk classes are grouped towards the left, reflecting that less liner structure is detected using the orientated bins method in low risk class (because there's very little dense structure present) and high risk classes (because dense areas are mostly large, homogenous blobs not linear structure). As can be clearly seen from the visualisation the synthetic images are grouped towards the lower end reflecting the smaller size of the linear structures detected.

The projections produced by LLE and Isomap produce mapping which contain far less visible structure in comparison to t-SNE. Both techniques produce a fairly structureless blob with some transition between the low & high risk to middle risk cases. This less obvious transition is again explained by the same relationship that is present in the t-SNE projection, but is just less prominent. The reduction in the quality of the projection could be down to the choice of k or the crowding problem.

Examination of the three quality criteria show that t-SNE does not perform so well here when compared to the blob feature space. In general, Isomap provides a "better" embedding than the one produced by t-SNE for both 2D and 3D projections. In terms of trustworthiness and continuity t-SNE produces embeddings that are better in 2 dimensions in terms of the immediate neighbourhood but quickly degrades when measured against larger neighbourhoods.

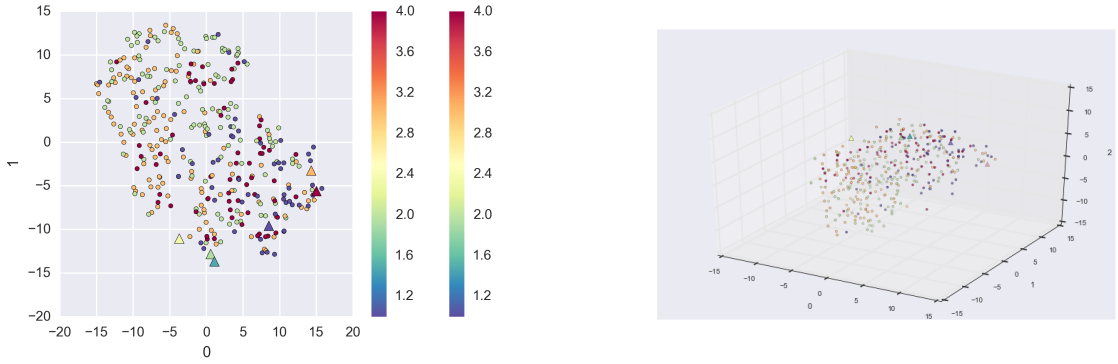


Figure 3.7: 2D & 3D projections of the line feature space produced by the t-SNE algorithm with a learning rate of 300 and perplexity of 30.

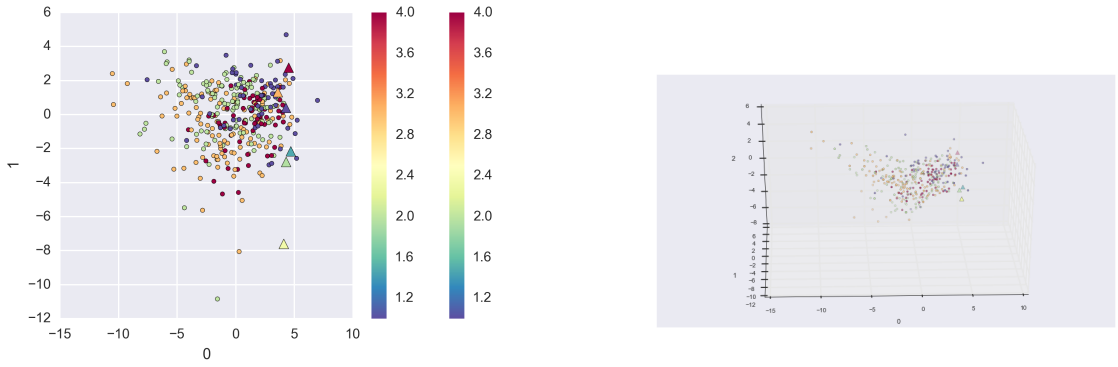


Figure 3.8: 2D & 3D projections of the line feature space produced by the Isomap algorithm with 10 neighbours.

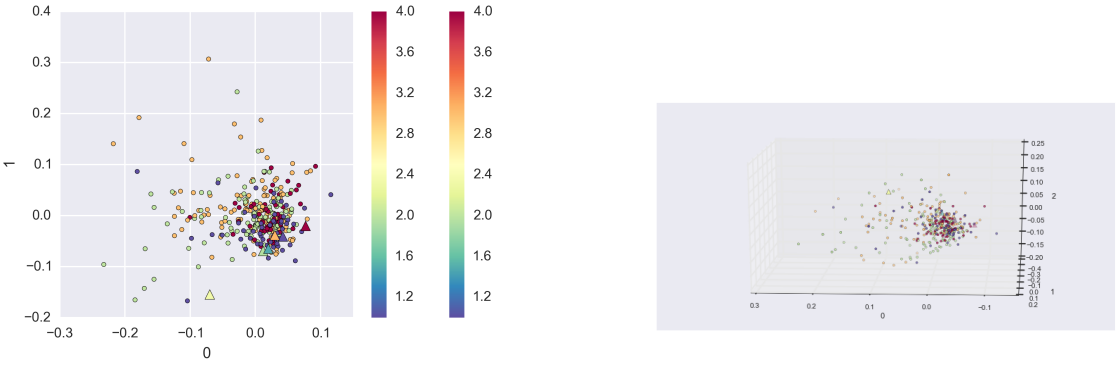


Figure 3.9: 2D & 3D projections of the line feature space produced by the LLE algorithm with 10 neighbours.

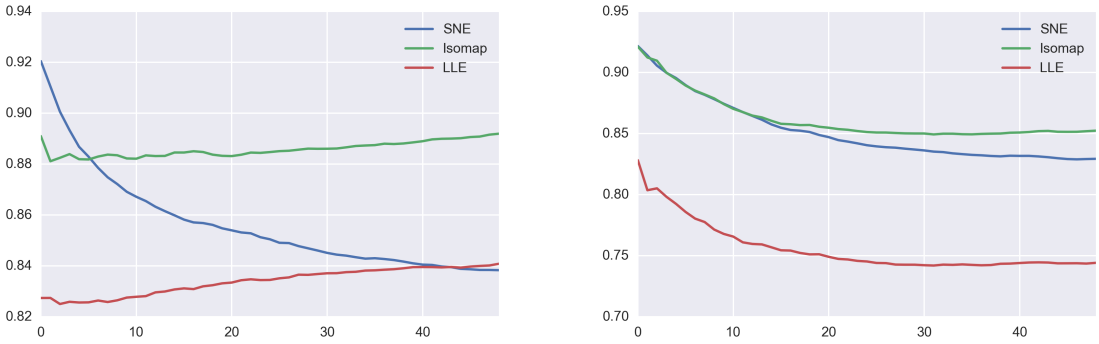


Figure 3.10: Trustworthiness and continuity of the 2D projections produced from line features.

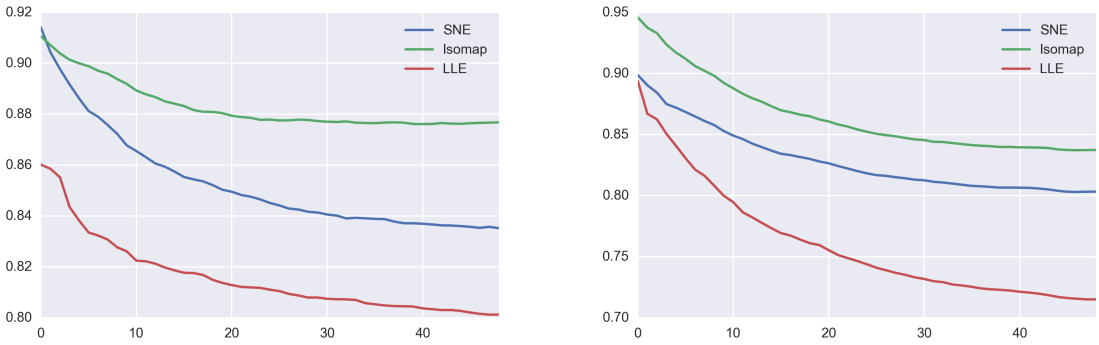


Figure 3.11: Trustworthiness and continuity of the 3D projections produced from line features.

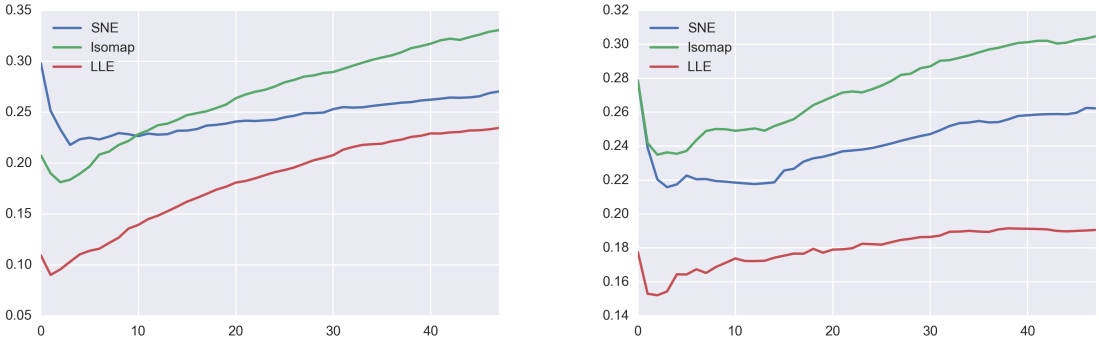


Figure 3.12: LCMC of both the 2D projection (left) and 3D projection (right) of the feature space for lines.

3.1.2 Intensity & Texture Features

Intensity and texture features showed the lowest similarity between the real and phantom datasets. As can be seen in figure **??** the intensity distribution of the phantom mammograms are nothing like a real mammogram. Because of this difference the results show that the they are clearly in different spaces. The results of the KS two sample test showed that the two distributions generated for both intensity and texture features are completely different. The results of the two sample Kolmogorov-Smirnov test are included in appendix **??** because the number of features for blobs is based on the number of scales used leading to the a large (up to 100) number of entries.

The feature spaces presented in this section were generated by taking ROIs defined by the blob and line features detected from each of the mammograms by computing intensity and texture features from each image patch. Because blobs are detected across a range of scales the average value for each scale is used leading to a feature space of size nk where n is the number of features and k is the number of scales (e.g. for texture: $n = 4$ and $k = 10$ so it has 40 features).

The results of performing dimensionality reduction on both of these feature spaces is that in all cases the synthetic mammograms are separated from the real mammograms into an isolated cluster on there own. This is because the values for both types of feature are on average much higher for the breast phantoms than the real mammograms.

For features derived from patches defined by blobs it can be seen that between the real mammograms there is a trend visible in both intensity and textural features which causes some transition between high to low risk. This is particularly visible for intensity features in the 2D plot produced by t-SNE shown in figure **??** and in the 2D plot in figure **??** for texture features.

In the case of intensity features this transition is caused by blobs that are composed of blobs which are of higher intensity. For texture features the trend is explained by a transition for blobs with low contrast and dissimilarity and high homogeneity and energy corresponding to high risk blobs transitioning through to low risk blobs where the reverse is true.

Intensity features derived from lines do show a degree of separation between high and low risk mammograms. High risk mammograms typically have a higher average intensity. This is most clearly seen in the 2D visualisations produced by t-SNE and Isomap in **??** and **??**. On the other hand the texture features do not provide any useful correlation between risk classes.

The quality measures for intensity features derived from blobs show that t-SNE produces the best visualisation in two dimensions but is again beaten by Isomap in three dimensions for anything larger than a small local neighbourhood. For texture features Isomap appears to out do t-SNE in all measures for neighbourhoods larger than 10.

Quality measures taken from the visualisations for intensity from lines show again that t-SNE is best for small values of k in 2 and 3 dimensions but is overtaken by Isomap and LLE for neighbourhoods of larger sizes. The results are similar for texture features derived from lines but in this case the performance of t-SNE is only very marginally better for small values of k. In almost all cases for texture from lines Isomap quickly overtakes t-SNE as having higher or similar quality projections.

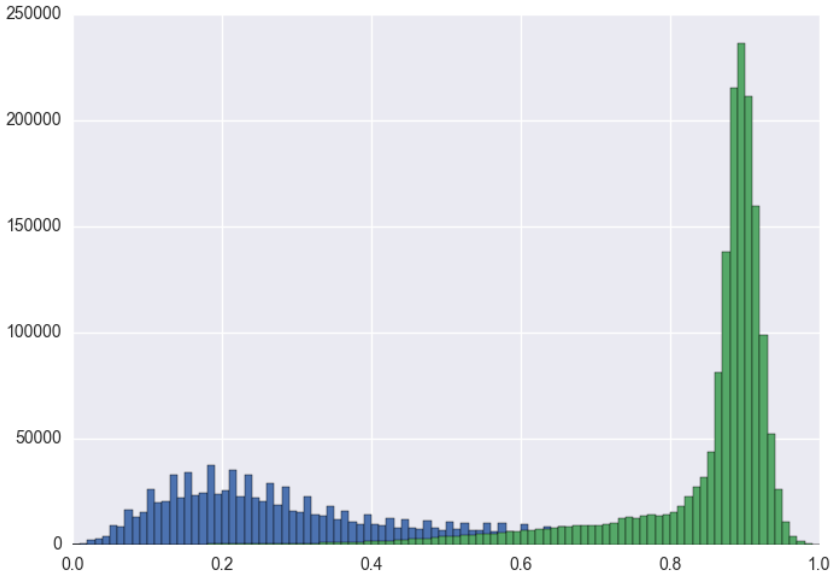


Figure 3.13: Comparison of the histogram of a real mammogram (blue) against a synthetic mammogram (green). The distribution of intensities are radically different from one another.

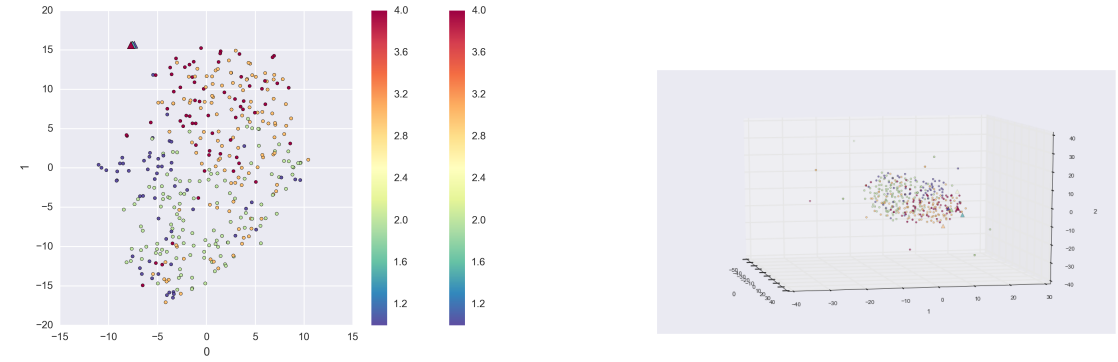


Figure 3.14: 2D & 3D projections of the intensity feature space produced by the t-SNE algorithm with a learning rate of 300 and perplexity of 30.

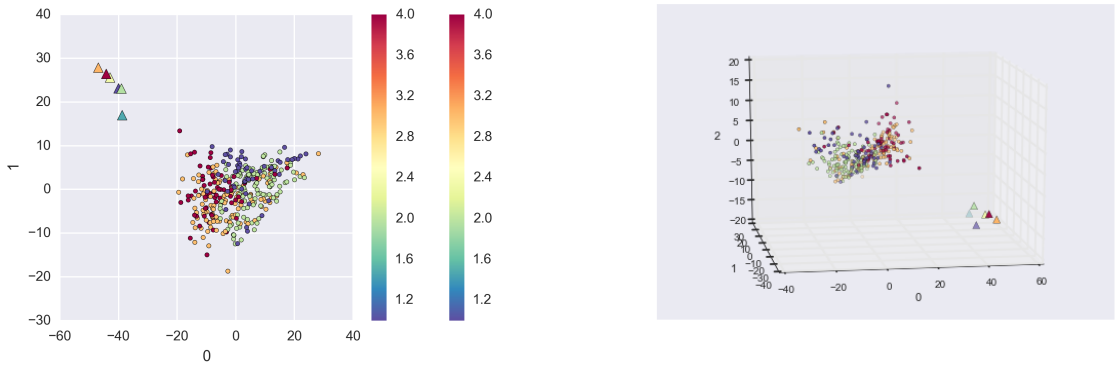


Figure 3.15: 2D & 3D projections of the intensity feature space generated from blobs produced by the Isomap algorithm with 50 neighbours.

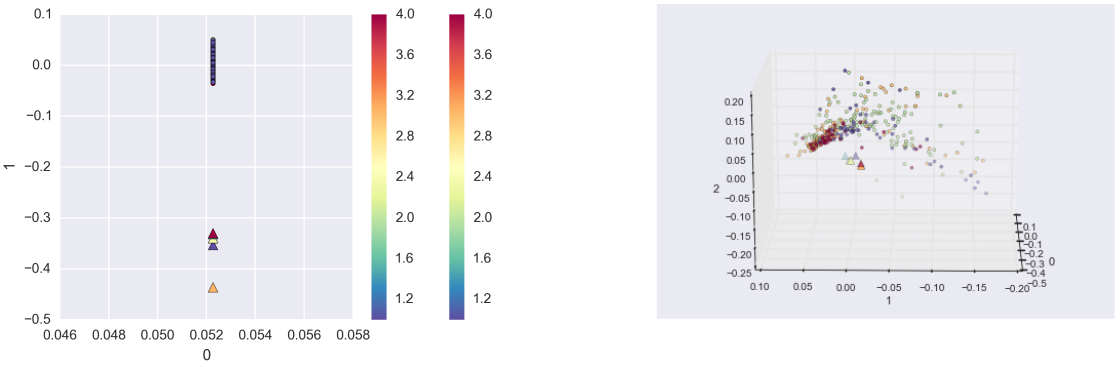


Figure 3.16: 2D & 3D projections of the intensity feature space generated from blobs produced by the LLE algorithm with 50 neighbours.

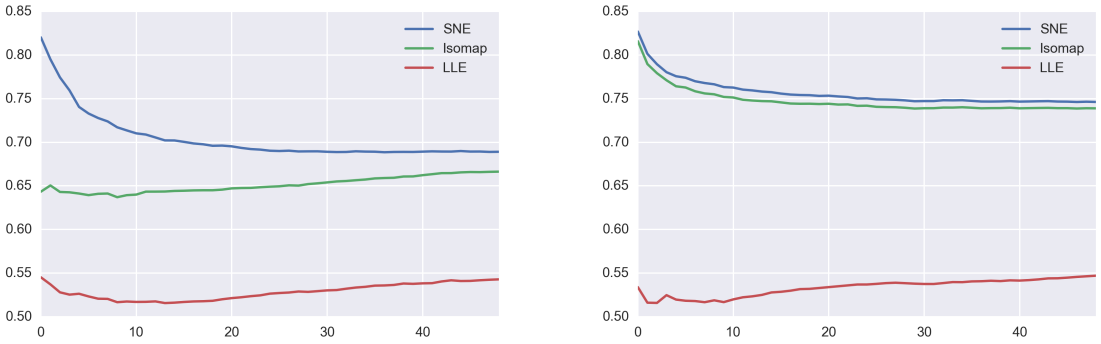


Figure 3.17: Trustworthiness and continuity of the 2D projections produced from intensity features from blobs.

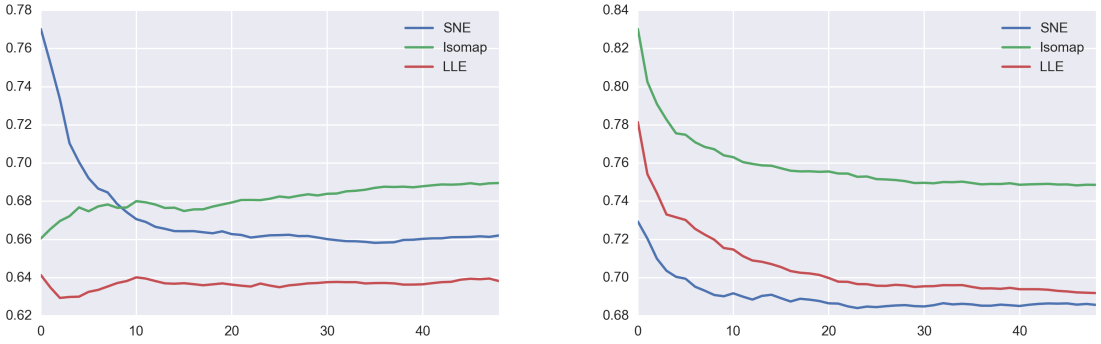


Figure 3.18: Trustworthiness and continuity of the 3D projections produced from intensity features from blobs.

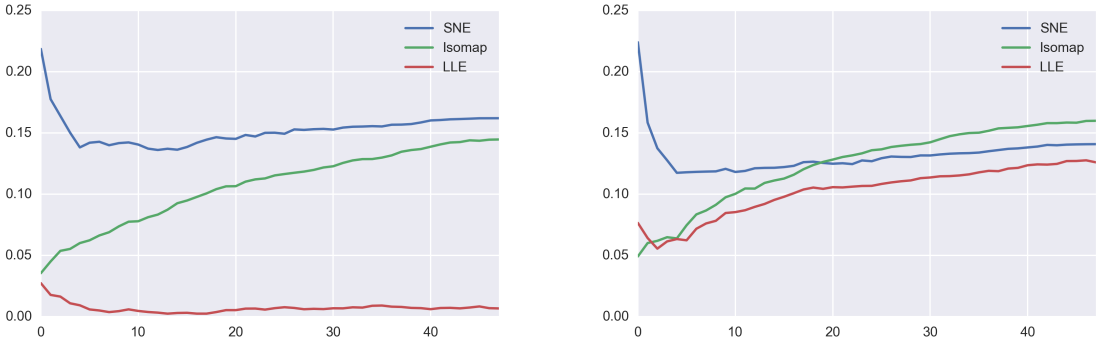


Figure 3.19: LCMC of both the 2D projection (left) and 3D projection (right) of the feature space for intensity from blobs.

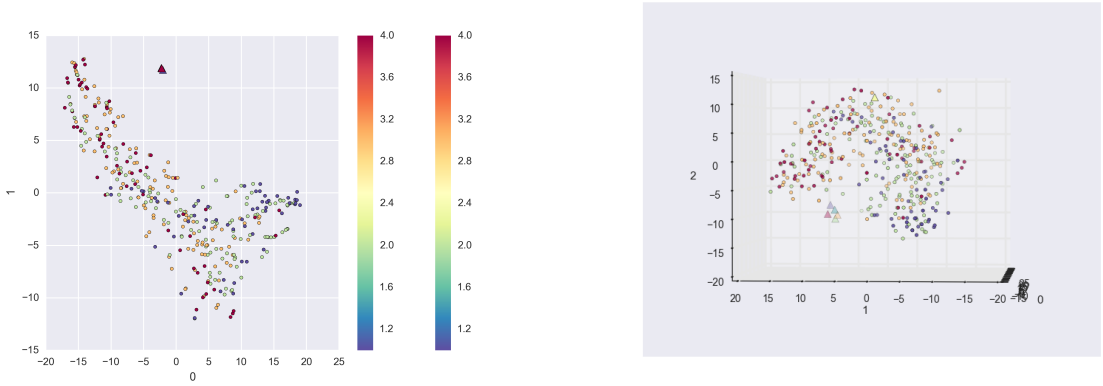


Figure 3.20: 2D & 3D projections of the texture feature space generated from blobs generated from blobs produced by the t-SNE algorithm with a learning rate of 300 and perplexity of 30.

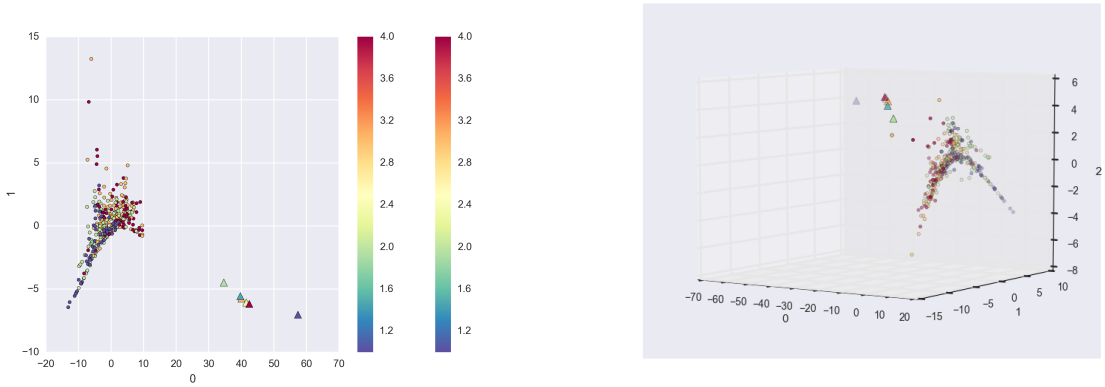


Figure 3.21: 2D & 3D projections of the texture feature space generated from blobs produced by the Isomap algorithm with 10 neighbours.

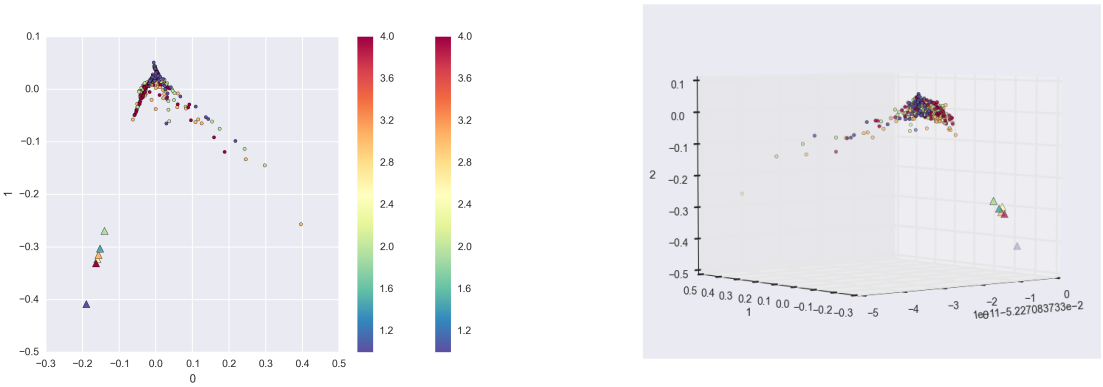


Figure 3.22: 2D & 3D projections of the texture feature space generated from blobs produced by the LLE algorithm with 10 neighbours.

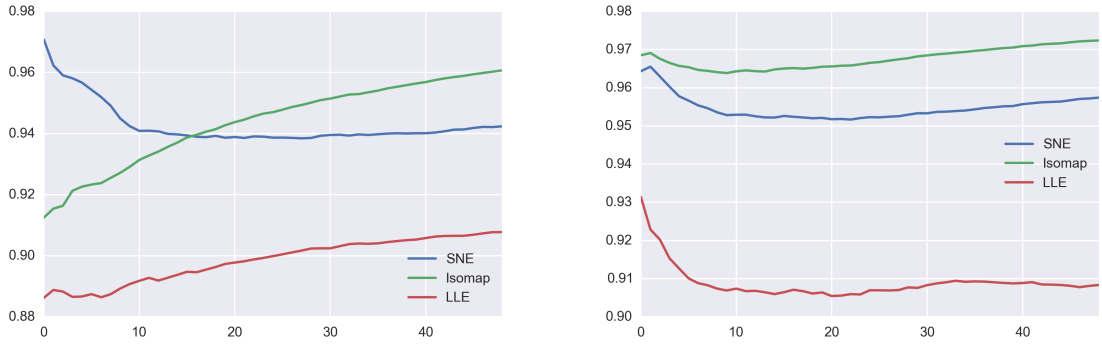


Figure 3.23: Trustworthiness and continuity of the 2D projections produced from texture features from blobs.

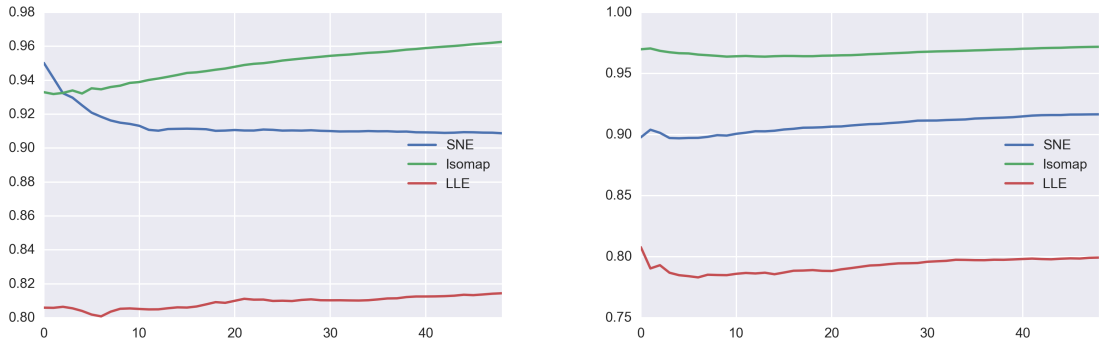


Figure 3.24: Trustworthiness and continuity of the 3D projections produced from texture features from blobs.

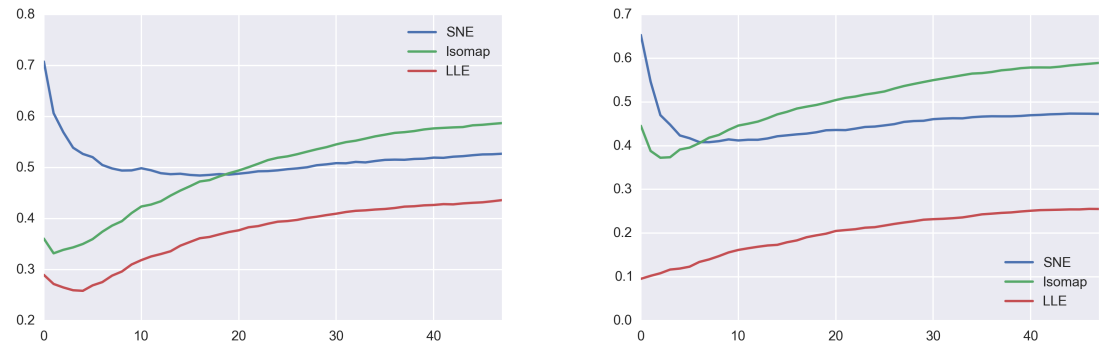


Figure 3.25: LCMC of both the 2D projection (left) and 3D projection (right) of the feature space for texture.

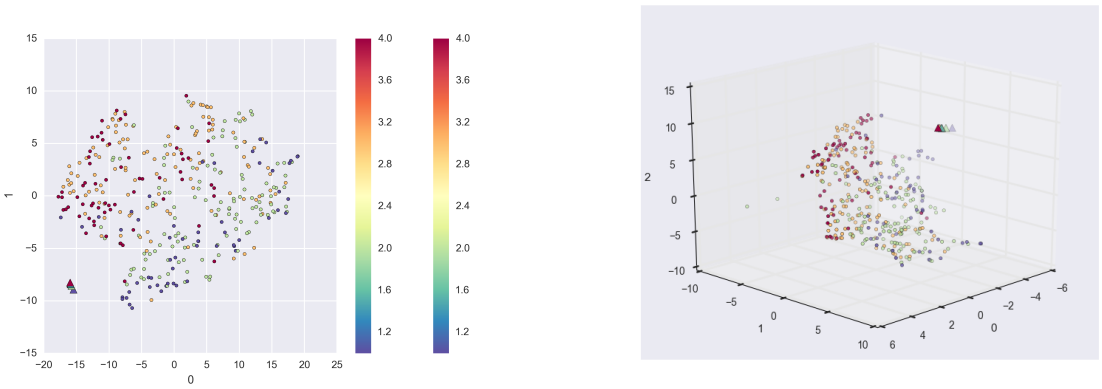


Figure 3.26: 2D & 3D projections of the intensity feature space for lines produced by the t-SNE algorithm with a learning rate of 300 and perplexity of 30.

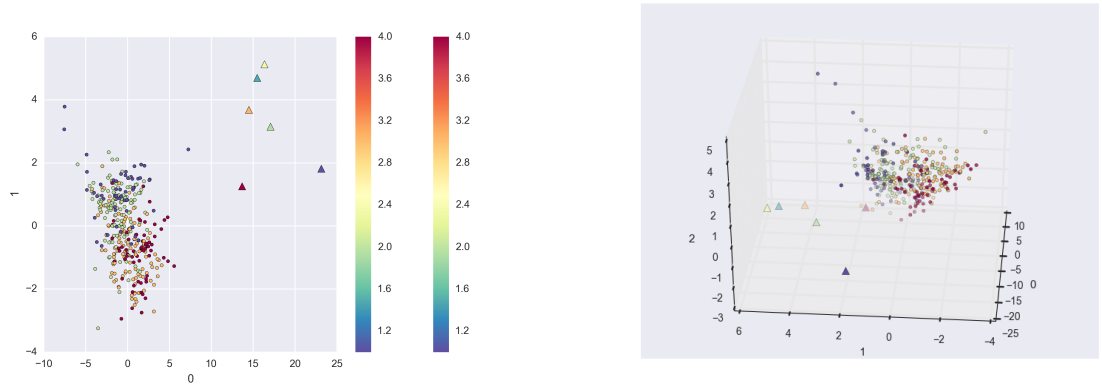


Figure 3.27: 2D & 3D projections of the intensity feature space generated from lines produced by the Isomap algorithm with 10 neighbours.

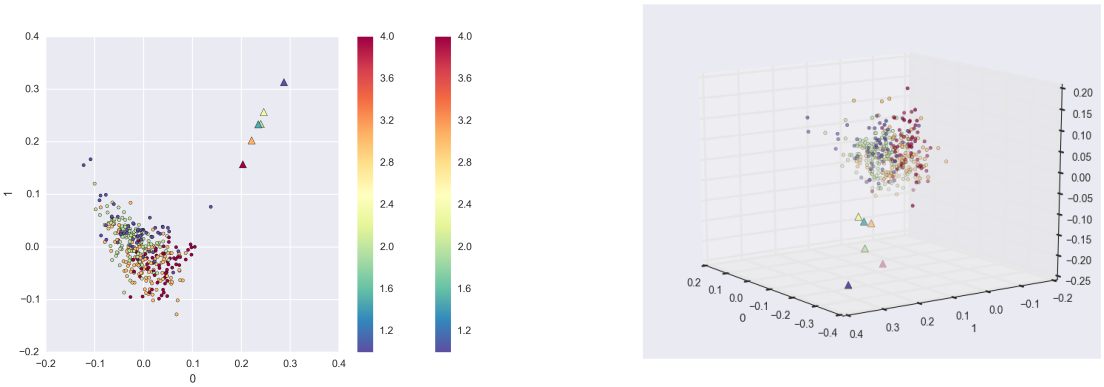


Figure 3.28: 2D & 3D projections of the intensity feature space for lines generated from blobs produced by the LLE algorithm with 50 neighbours.

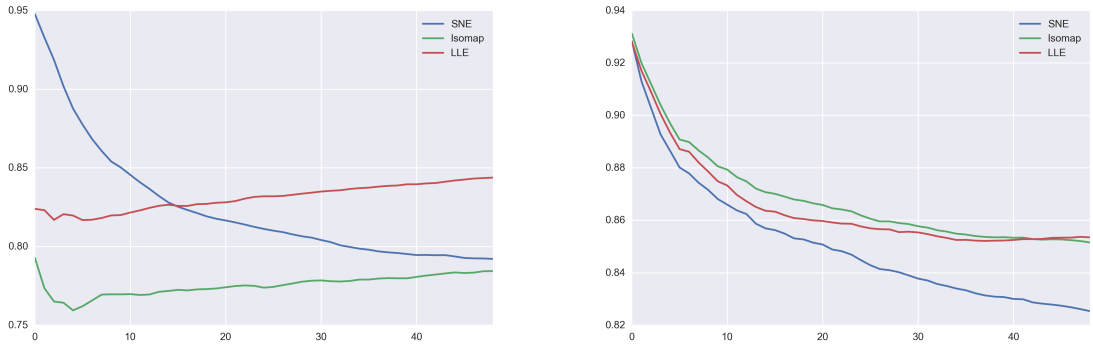


Figure 3.29: Trustworthiness and continuity of the 2D projections produced from intensity features from lines.

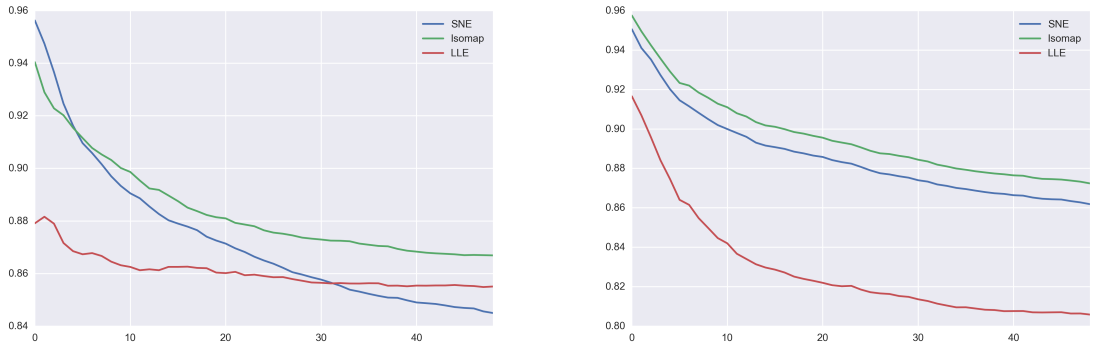


Figure 3.30: Trustworthiness and continuity of the 3D projections produced from intensity features from lines.

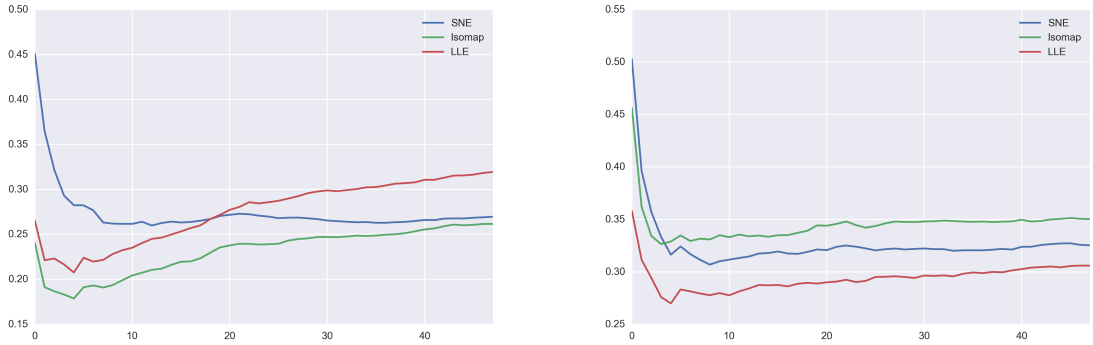


Figure 3.31: LCMC of both the 2D projection (left) and 3D projection (right) of the feature space for intensity from lines.

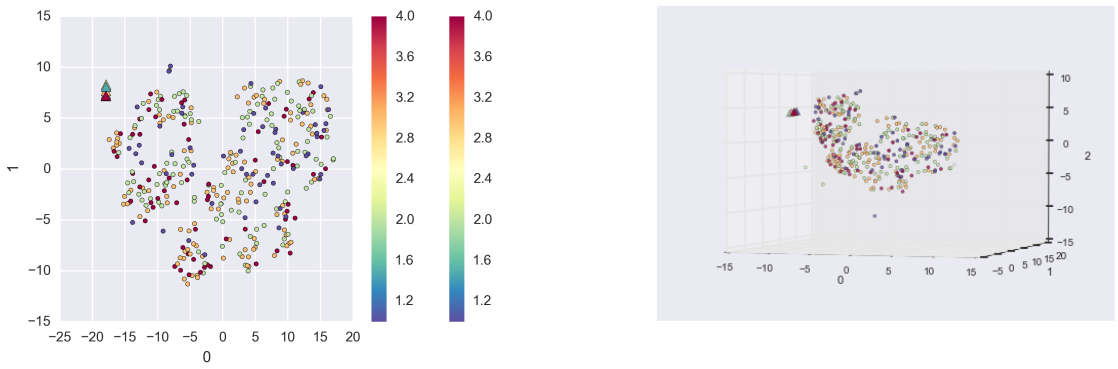


Figure 3.32: 2D & 3D projections of the texture feature space generated generated from lines produced by the t-SNE algorithm with a learning rate of 300 and perplexity of 30.

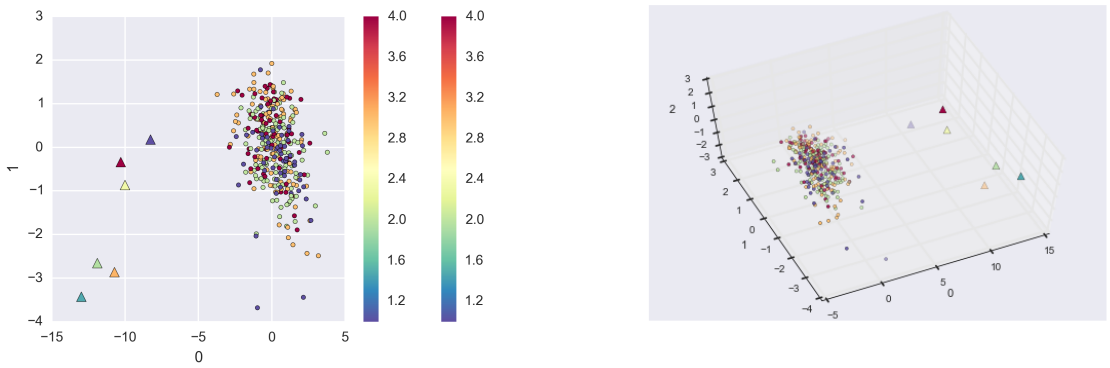


Figure 3.33: 2D & 3D projections of the texture feature space generated from lines produced by the Isomap algorithm with 10 neighbours.

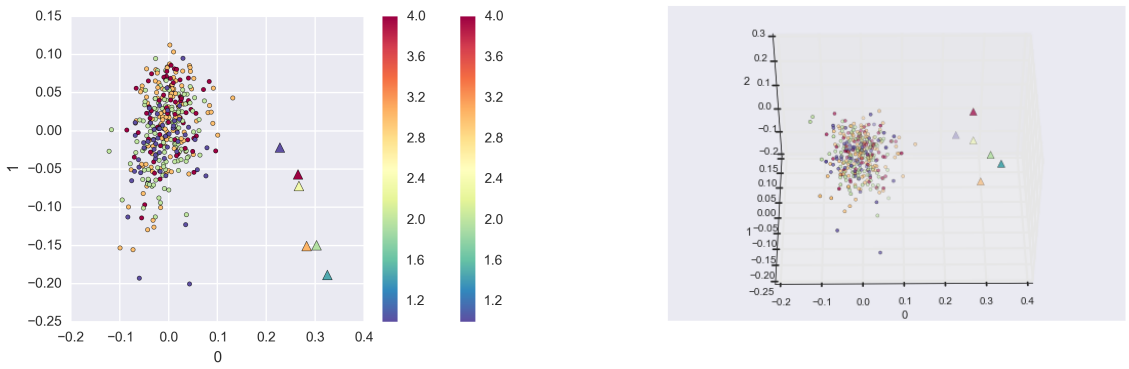


Figure 3.34: 2D & 3D projections of the texture feature space generated from lines produced by the LLE algorithm with 10 neighbours.

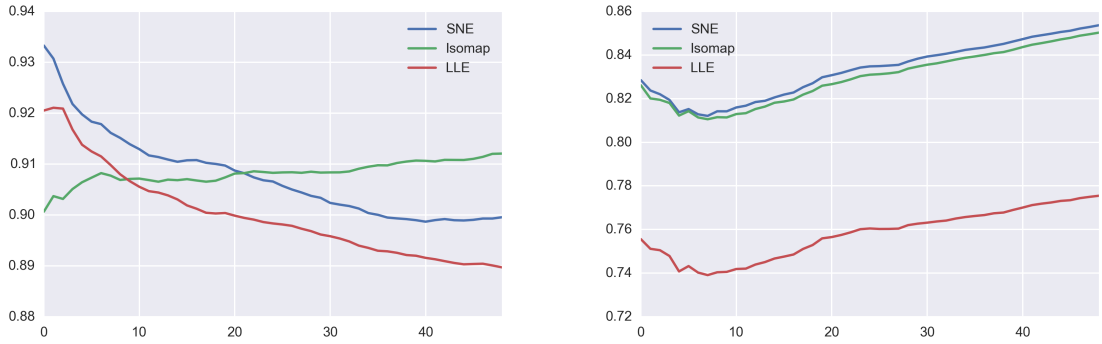


Figure 3.35: Trustworthiness and continuity of the 2D projections produced from texture features from lines.

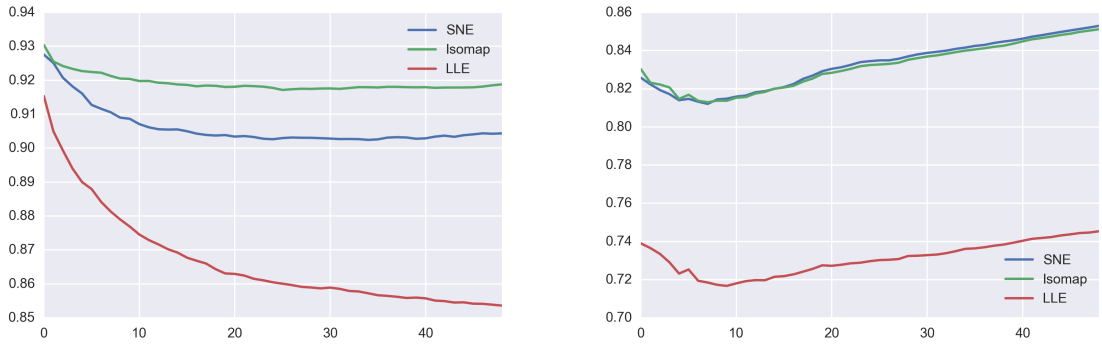


Figure 3.36: Trustworthiness and continuity of the 3D projections produced from texture features from lines.

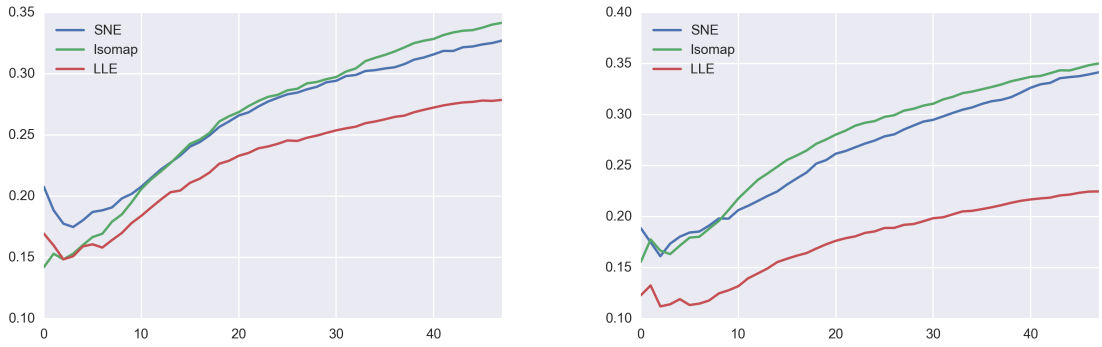


Figure 3.37: LCMC of both the 2D projection (left) and 3D projection (right) of the feature space from lines for texture.

3.2 Conclusions

In summary, it can be concluded that the synthetic mammograms used as part of this experiment are not significantly related to real mammograms. The synthetic mammograms appear to be closest to real mammograms in terms of the shape of the structures present in the image. The best results from this experiment were derived from the feature space of multi-scale blobs. Line features also seemed to positively show the the real and synthetic mammograms are in the same space in terms of shape. However, texture and intensity features derived from either features clearly show that the two datasets are not in the same intensity space. The two sample KS test confirms that the feature created from both datasets are statistically not drawn from the same distribution and therefore must conclude that they are, for all features presented here, effectively different.

This conclusion roughly correlates with the limitations discussed by the authors of the synthetic data [1–3]. They state that the synthetic mammograms are closest to real mammograms in terms of shape but are not so close in terms of intensity and texture. In the experiments with shape features presented here the major difference between the real and synthetic mammograms was caused by a lack of small scale structure being detected in the synthetic mammograms. This causes them to be grouped in with mammograms which are deemed to be of higher risk, regardless of their effective risk, because of the lack of small blob counts detected. The same can be said for line features where the number and size of lines is generally smaller, regardless of the risk.

The quality evaluation of the visualisations produced suggests that t-SNE produces visualisations which best capture the local neighbourhood structure in for a small local neighbourhood, but that Isomap generally performs better across most feature spaces as k increases. This does appear to coincide with expectations about what the two algorithms aim to preserve. t-SNE is typically better at preserving the local relationship of a neighbourhood and Isomap better for the global structure. LLE is shown to generally produce better visualisations when k is small and degrades as the neighbourhood becomes larger. However it was typically shown to produce a lower quality of embedding in comparison to t-SNE.

Chapter 4

Critical Evaluation

This final chapter presents my evaluation of the project as a whole, outlines directions for future work in this area and provides a personal reflection on how I have handled and executed this project.

4.1 Evaluation of the Project

Overall this project has managed to answer some of the original questions laid out during the problem analysis and I feel that while the results of this project are not necessarily as good as I was hoping for this has still been a fun project and excellent learning experience. I started this project with very minimal knowledge of image processing and analysis techniques (other than what had been taught in other university modules) and no knowledge of dimensionality reduction algorithms. I also started with module with very little prior knowledge linear algebra, something which I would call a prerequisite for properly understanding many dimensionality reduction algorithms.

There were several goals that I set out to achieve as part of this project. This section will discuss them one by one and provide a review of what was achieved, what went well, and what went wrong.

The first goal was to image features from both real and synthetic mammogram images based on a variety of common techniques used with mammographic images. This required an extensive review of existing literature on the subject and the selection of a few techniques from each case to be used as part of the final system pipeline.

I feel that the features chosen for use in this project were entirely appropriate. I feel that starting with shape based features was a good idea and that the two techniques chosen produced reasonable results in practice. Blob features proved more complicated to implement than line features overall, but blob features end up producing better results and so I felt the extra effort involved was justified.

After extracting shape features I wanted to also try and focus on extracting other types of feature from the images. An difficult issue with this the time complexity required to extract more complicated features (i.e. texture features) from an image. For this reason I chose to use the patch of image defined by shape features as way narrow down the amount of processing required on each image. The reasoning behind this was that because a ROI has already been defined, extracting texture and intensity features from that region might yield additional useful information.

My biggest regret with feature extraction is that I didn't realise until after the extracting texture and intensity features that the intensity distribution between real and synthetic images is very different, leading to the clear separation between the two datasets that can be viewed in all of the visualisations of these spaces. While this proved to be a disappointing result it does correlate with the some of the limitations proposed by the authors of the synthetic dataset in refs. [1–3]. In terms of the shape features,

the synthetic mammograms at least appeared to be in the same space as their real counterparts however but the extracted features were still unable to properly align the synthetics under projection according to their associated level of risk.

The second goal was to perform dimensionality reduction on the feature space generated by each of these techniques to produce two and three dimensional visual representations. A variety of different techniques were investigated as part of the initial stage of this project and a few fundamentally different approaches were selected to be used in the results of this project.

As I have briefly mentioning in the first paragraph of this section, one of the major challenges that I felt I faced with this project was trying fully understand the core principles and mathematics behind how dimensionality reduction algorithms worked. This was an area in which I had zero knowledge of before I began this project and subsequently I spent a significant proportion of the early part of the project trying researching how some of the more common techniques work along with the strengths and weaknesses of each.

There was very little technical effort involved with this component of the project. All of the techniques used in the project were already implemented as part of the scikit-learn python library. I felt that there was no need to reimplement complicated dimensionality reduction algorithms when they were already readily available in a well known and well tested package. I feel that the choice of dimensionality reduction techniques used was justified and that a small yet diverse mix of approaches were presented along with the results.

The third goal was to produce a way of evaluating the quality of the visualisations produced by dimensionality reduction. Once again this was investigated as part of the initial stage of the project and quality metrics derived from co-ranking matrices were eventually selected as the best choice due to the large number of measures that can be derived from them.

Again, like dimensionality reduction, this was an area that required more background reading that actual implementation. The implementation of the co-ranking matrix and the measures derived from it are actually only a few hundred lines of code, but understanding how the co-ranking matrix is derived and what the metrics subsequently derived from it measure took the majority of the effort. I felt that this was a justified approach to assess the quality of the mappings produced but that there was much more that could have been done in this area, for example in visualising the quality of the mapping in a similar way as suggested by Mokbel et al [41].

The third aim was to integrate these separate components into an image analysis pipeline. This goal was achieved as part of a Python image analysis package produced during this project. While the implementation offered in this package is far from perfect I feel that it provides a good basic implementation of a image analysis pipeline. I feel that the most well developed part of the system is the image processing component. This provides reusable way to implement new feature detection routines through an interface that provides support for multiprocessing on a per image basis.

There are, however, some parts of the system I am not entirely happy with. The most notable of these is the section analysis module. This has mostly been developed on an ad-hoc basis depending on the functions I required to perform analysis. This module has ended up being a collection of the more common operations used when processing the features using IPython notebooks. It as particularly difficult deciding which functions to include in this module. The problem is that the operations you need to perform on the data largely depends on the patterns you see in the data itself. On one hand obviously not every single thing should be included, but on the other the aim of good software is to produce reusable components. Based on this I chose to work on the code in an IPython notebook and if I required the function in more than one place, move it into the analysis module.

The final aim of this project was to try and examine the difference between projections of the feature space of real and synthetic mammogram datasets. In this goal I feel that I have only been partially successful. While I feel that the results of the experiments performed allow us to draw some valid

conclusions about the the differences between the two datasets I also feel that I could have spent more time constructing system to evaluate the results.

I feel that by far my most major weakness in the project was failing to full develop how to formally compare the feature spaces generated by the two datasets. Throughout the project my analysis of the dimensionality reduction was mostly based on visual examination of the mapping and trying to relate back to the original data. Looking at how images change across the lower dimensional embedding and trying to relate this to why the synthetics images show up as different works fine, but does not provide quantitive evaluation of how the feature spaces overlap.

I believe that this weakness was mostly caused by my own failure to plan this section of the project correctly ahead of time. This was the first research style project I have undertaken and in reflection I should have spent more time planning how to formally compare the two datasets. In reflection I feel that I perhaps jumped into development and implementation too soon without giving enough thought as to how to formally evaluate my results.

In reflation on the choice of language used I feel that Python was an excellent choice for this project. The fact that Python is a very high level language but with the capability of producing decent performance when needed. The scipy and numpy libraries in particular saved me from having to reinvent the wheel on a lot of things. The general design of the language allowed me to work at a higher level compared to other languages. I found that I was easily able to focus on actually implementing what was required for the project than spending my time writing boilerplate code. Python's C API also came in handy as predicted. My implementation of the deformable convolution would have been much, much slower if it had been implemented in pure Python code.

My approach to the management of the project seemed to work reasonably well. The agile based process I used in this project allowed me all of the flexibility I needed to work on what needed to be done rather than focussing on rigid requirements. As predicted I found that it really helped to be adaptive the project as new issues and challenges arose.

One thing that I found worked particularly well was the having a weekly iteration which was timed to start and finish with my weekly dissertation meeting. The discussion in these meetings allowed me to decide what "stories" should be worked on throughout the week. During the first few iterations I tended to find that I would be overly ambitious about the number of task I could complete in a week. In the later iterations I found that this started to balance out more as I became more aware of how much I could feasibly achieve in a week.

I found the use of test driven development to be difficult to adhere to at times. I also often found it challenging to write decent tests for the software I was producing. Many of the functions in the package either return output that is difficult to evaluate for correctness. Another issue with testing was the fact that a number routines take a quite a long time to execute (i.e. greater than a minute per image). As this execution time makes it infeasible for unit testing I found that regression testing of the larger components of the system often proved to be more useful than the smaller unit tests. I did, however, find that both types of testing used in the project were extremely useful debugging the program, especially in the later stages of development as the system grew larger.

I also had a positive experience with using continuous integration throughout my project. My primary workflow involved doing most of my work on separate git branches and merging the finished work into the main branch upon completion. I found that it provided a useful safety net for checking if I broke something while working on code in two different branches in parallel.

In conclusion I found this project to be a mostly enjoyable and rewarding experience. I found the most challenging and rewarding aspect to be trying to understand the concepts that I have been trying to make use. I strongly feel that while many of aspects of this project weren't a success I have learned a great deal how to better structure a research based project like this. I have also been left with a much stronger knowledge and personal interest in the image analysis and dimensionality reduction domains.

4.2 Future Work

There are a multitude of ways in which this project could be extended. This section lists of of the ways in which I believe the project could be expanded along with a justification as to why each point would be worth investigating.

One piece of additional work I would have liked to of investigated is to run the projections produced through a couple of common classifiers (such as, for example, a non-linear SVM). Alongside the quality metrics for visualisation, this would give me some raw numbers quantifying how well a classifier performs on the feature spaces produced by my implementations. While this would have little benefit to the projects main goals of examining the differences between synthetic and real feature spaces, it would have given me . This would have the additional advantage of providing a quantitive way of adjusting the parameters for the feature extraction and dimensionality reduction components of the system.

Another improvement I would have liked to have made to the system is the to speed up the performance of the deformable convolution module. Currently this module provides a very slow implementation of the convolution operator. While this is necessarily slow around the edge of mask being convolved with the image because of the nature of modifying the kernel with every convolution this does not need to be the case across the main portion of the image. For most of the breast two 1D second derivative Gaussian kernel in vertical and horizontal directions could be used and simply summed together to produce the same response but reducing the overall complexity.

I believe that the best feature used in this project was the multi-scale blob feature. I think that if I were to continue developing this project I would have experimented with making the linear structure feature multi-scale. I think that this feature misses some of the very small and very large scale linear structures because the algorithm has trouble detecting features across all scales with the same parameters (bins size, threshold etc.). For smaller lines the kernel proves to be too large and the line response does not show through strongly enough to avoid getting removed by thresholding. Likewise for larger lines the opposite is true; the response from the kernel in all directions appears uniform. Using this, duplicated lines detected across scales could be removed using a merging method like blob features, but it would obviously have to be more complicated because of the inconsistent size and shape of line features.

With regards to the visualisation aspect of this project I feel that there are several areas for further expansion. One method that captures my attention was a point wise measure of quality for lower dimensional mappings derived from the co-ranking matrix [41]. This can be used to colour the data points in a scatterplot of the lower dimensional mapping of a feature space. I feel that an approach based on a technique such as this provide a much easier visual interpretation of the quality of the visualisation in comparison of the measures of trustworthiness, continuity, and LCMC.

During this project I have not focussed too much on the use of higher dimensional representations of the feature space. While these are often not as easy to interpret as a 2D or 3D scatterplot, they can provide useful additional insight. One thing which I would have liked to have tried in my project is to implement a method of ordering features according to there importance such as proposed by many methods listed in ref. [6]. In particular my imagination was captured by the discussion of ordering parallel coordinates axes using the Hough transform [64] and interactive feature selection by information loss [30].

One aspect that I did not focus on much during the course of this project was “topological” aspect in the project title. Mid way through the course of this project I began to read into the field of topological data analysis [9]. I think while this area is not necessarily directly related to the examining the goals of this specific project it would be an area I feel would be worth examining. I think that exploring the topology of the mammographic feature space might yield some interesting relationships and I would be excited to explore this area further.

Appendices

Appendix A

Third-Party Code and Libraries

name	description
click	Command line interface library used to create the various CLI tools supplied with the package created for this project.
coverage	Python library used to measure the coverage provided by the unit tests. This was used both by the developer and automatically run by the build server
matplotlib	Matplotlib is an general purpose 2D and 3D plotting library. This library has been heavily used both as a component of the pandas library, and in its own right to generate most of the plots shown in this document.
medpy	Library which provides functions for reading DICOM format images
nose	Python unit testing tool. This provides a suite of test helpers and assertion functions as well as a command line program used to run the project's unit & regression tests
numpy	General purpose, fast and efficient array manipulation library. This is a core dependancy of scipy, pandas, and the scikit libraries.
pandas	Pandas provides high level data analysis and manipulation tools. This project is heavily dependant on pandas for its database-esque operations, plotting routines, and I/O routines.
scikit-image	scikit-image is built on top of the scipy library and provides many of the higher level image analysis functions such as loading and transforming images, as well as the implementation of the GLCM matrices used in the project.
scikit-learn	The scikit-learn library provided implementations of the manifold learning algorithms used in the project. This includes the implementations of t-SNE, LLE and Isomap. Other uses of the library were for pairwise distance computations and k-means clustering.
scipy	Scipy is a collection of mathematical, scientific, and engieering packages. Scipy is a core dependancy of many of the other third-party libraries used in this project. It has also been heavily used within the project itself.
seaborn	Seaborn is another plotting library built on top of matplotlib. It provides some additional plotting functionality not present in matplotlib itself.
sphinx	Sphinx is a library which can be used to automatically generate documentation from the docstrings of Python code.

Table A.1: List of third-party libraries used in this project

Appendix B

Code samples

Appendix C

IPython Notebooks

Blob Analysis

```
In [121]: %matplotlib inline
          import pandas as pd
          import numpy as np
          import scipy.stats as stats
          import matplotlib.pyplot as plt
          import mia
```

```
Warning: Cannot change to a different GUI toolkit: qt. Using osx instead.
```

3.1 Loading and Preprocessing

Loading the hologic and synthetic datasets.

```
In [83]: hologic = pd.DataFrame.from_csv("hologic_blobs.csv")
         phantom = pd.DataFrame.from_csv("synthetic_blobs.csv")
```

Loading the meta data for the real and synthetic datasets.

```
In [84]: hologic_meta = mia.analysis.create_hologic_meta_data(hologic, "meta_data/real_meta.csv")
         phantom_meta = mia.analysis.create_synthetic_meta_data(phantom,
                                                                "meta_data/synthetic_meta.csv")
         phantom_meta.index.name = 'img_name'
```

Prepare the BI-RADS/VBD labels for both datasets.

```
In [85]: hologic_labels = hologic_meta.drop_duplicates().BIRADS
         phantom_labels = phantom_meta['VBD.1']

         class_labels = pd.concat([hologic_labels, phantom_labels])
         class_labels.index.name = "img_name"
         labels = mia.analysis.remove_duplicate_index(class_labels)[0]
```

3.2 Creating Features

Create blob features from distribution of blobs

```
In [86]: hologic_blob_features = mia.analysis.features_from_blobs(hologic)
         phantom_blob_features = mia.analysis.features_from_blobs(phantom)
```

Take a random subset of the phantom mammograms. This is important so that each case is not over represented.

```
In [87]: syn_feature_meta = mia.analysis.remove_duplicate_index(phantom_meta)
         phantom_blob_features['phantom_name'] = syn_feature_meta.phantom_name.tolist()
         phantom_blob_features_subset = mia.analysis.create_random_subset(phantom_blob_features,
                                                                          'phantom_name')
```

Combine the features from both datasets.

```
In [88]: features = pd.concat([hologic_blob_features, phantom_blob_features_subset])
         assert features.shape[0] == 366
         features.head()
```

Out[88]:

	blob_count	avg_radius	std_radius	min_radius	\
p214-010-60001-cl.png	56	22.121831	22.923389	8	
p214-010-60001-cr.png	78	19.054538	17.506086	8	
p214-010-60001-ml.png	98	20.011191	21.876304	8	
p214-010-60001-mr.png	139	15.309764	15.307860	8	
p214-010-60005-cl.png	97	20.132590	23.255605	8	

	max_radius	small_radius_count	med_radius_count	\
p214-010-60001-cl.png	128.000000	52	1	
p214-010-60001-cr.png	90.509668	68	4	
p214-010-60001-ml.png	128.000000	90	3	
p214-010-60001-mr.png	128.000000	136	1	
p214-010-60005-cl.png	181.019336	94	2	

	large_radius_count	density	upper_dist_count	25%	\
p214-010-60001-cl.png	3	52.940812	21	8	
p214-010-60001-cr.png	6	40.749811	22	8	
p214-010-60001-ml.png	5	42.644057	27	8	
p214-010-60001-mr.png	2	38.287439	40	8	
p214-010-60005-cl.png	1	41.456308	27	8	

	50%	75%
p214-010-60001-cl.png	11.313708	22.627417
p214-010-60001-cr.png	11.313708	22.627417
p214-010-60001-ml.png	11.313708	22.627417
p214-010-60001-mr.png	11.313708	16.000000
p214-010-60005-cl.png	11.313708	22.627417

Filter some features, such as the min, to remove noise.

```
In [89]: selected_features = features.drop(['min_radius'], axis=1)
```

3.3 Compare Real and Synthetic Features

Compare the distributions of features detected from the real mammograms and the phantoms using the Kolmogorov-Smirnov two sample test.

```
In [90]: ks_stats = [list(stats.ks_2samp(hologic_blob_features[col],
                                         phantom_blob_features[col]))
                                         for col in selected_features.columns]

         ks_test = pd.DataFrame(ks_stats, columns=['KS', 'p-value'], index=selected_features.columns)
         ks_test.to_latex("tables/blob_features_ks.tex")
         ks_test
```

```
Out[90]:                          KS       p-value
         blob_count          0.341667  2.774753e-07
         avg_radius          0.847222  1.360953e-42
         std_radius          0.711111  3.929143e-30
         max_radius          0.363889  3.327680e-08
         small_radius_count  0.319444  2.024353e-06
         med_radius_count    0.338889  3.583275e-07
         large_radius_count  0.733333  5.112303e-32
         density             0.169444  4.114705e-02
         upper_dist_count    0.345833  1.883393e-07
         25%                 0.358333  5.726005e-08
         50%                 0.743056  7.334764e-33
         75%                 0.777778  5.796838e-36
```

3.4 Dimensionality Reduction

3.4.1 t-SNE

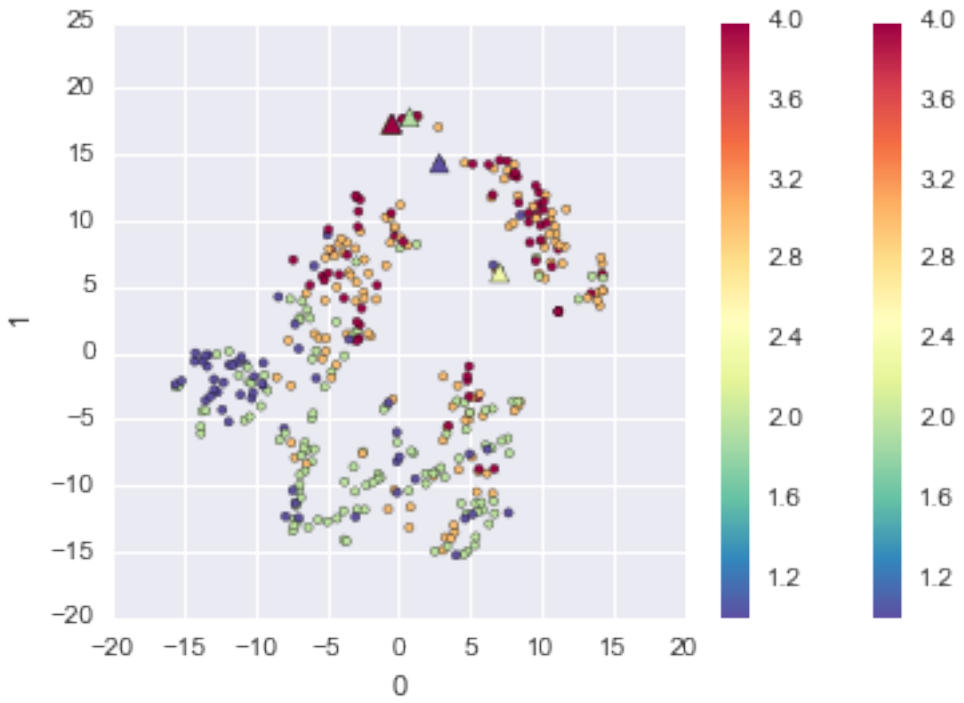
Running t-SNE to obtain a two dimensional representation.

```
In [91]: kwargs = {
             'learning_rate': 300,
             'perplexity': 30,
             'verbose': 1
         }
```

```
In [92]: SNE_mapping_2d = mia.analysis.tSNE(selected_features, n_components=2, **kwargs)
```

```
[t-SNE] Computing pairwise distances...
[t-SNE] Computed conditional probabilities for sample 366 / 366
[t-SNE] Mean sigma: 0.989818
[t-SNE] Error after 83 iterations with early exaggeration: 12.840095
[t-SNE] Error after 284 iterations: 0.384253
```

```
In [93]: mia.plotting.plot_mapping_2d(SNE_mapping_2d, hologic_blob_features.index,
                                     phantom_blob_features_subset.index, labels)
         plt.savefig('figures/mappings/blob_SNE_mapping_2d.png', dpi=300)
```



Running t-SNE to obtain a 3 dimensional mapping

```
In [94]: SNE_mapping_3d = mia.analysis.tSNE(selected_features, n_components=3, **kwargs)
```

```
[t-SNE] Computing pairwise distances...
[t-SNE] Computed conditional probabilities for sample 366 / 366
[t-SNE] Mean sigma: 0.989818
[t-SNE] Error after 83 iterations with early exaggeration: 13.395568
[t-SNE] Error after 228 iterations: 0.412479
```

```
In [126]: mia.plotting.plot_mapping_3d(SNE_mapping_3d, hologic_blob_features.index,
                                       phantom_blob_features_subset.index, labels)
```

```
Out[126]: <matplotlib.axes._subplots.Axes3DSubplot at 0x11472cdd0>
```

3.4.2 Isomap

Running Isomap to obtain a 2 dimensional mapping

```
In [96]: iso_kwargs = {
             'n_neighbors': 10,
         }
```

```
In [97]: iso_mapping_2d = mia.analysis.isomap(selected_features, n_components=2, **iso_kwargs)
```

```
5.68455464542
```

```
In [132]: mia.plotting.plot_mapping_2d(iso_mapping_2d, hologic_blob_features.index,
                                       phantom_blob_features_subset.index, labels)
          mia.io_tools.dump_mapping_to_json(iso_mapping_2d.loc[hologic_blob_features.index],
                                            [0,1], labels, 'data.json')
          plt.savefig('figures/mappings/blob_iso_mapping_2d.png', dpi=300)
```

```
In [99]: iso_mapping_3d = mia.analysis.isomap(selected_features, n_components=3, **iso_kwargs)
```

```
4.26343423263
```

```
In [123]: mia.plotting.plot_mapping_3d(iso_mapping_3d, hologic_blob_features.index,
                                       phantom_blob_features_subset.index, labels)
```

```
Out[123]: <matplotlib.axes._subplots.Axes3DSubplot at 0x11230ecd0>
```

3.4.3 Locally Linear Embedding

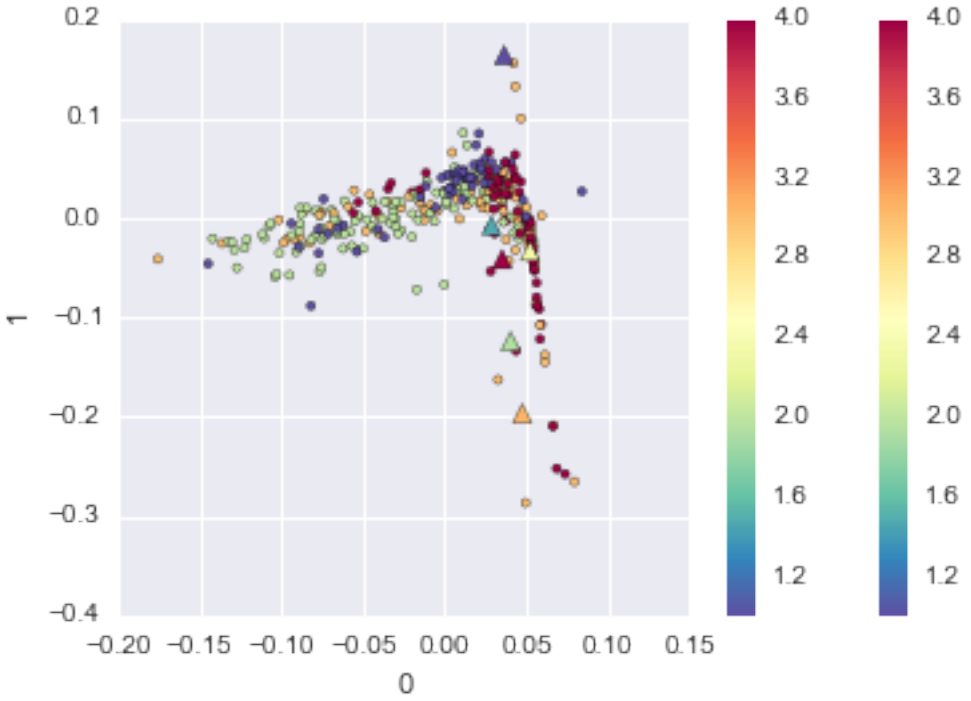
Running locally linear embedding to obtain 2d mapping

```
In [101]: lle_kwargs = {
              'n_neighbors': 10,
          }
```

```
In [102]: lle_mapping_2d = mia.analysis.lle(selected_features, n_components=2, **lle_kwargs)
```

```
1.17807316961e-07
```

```
In [103]: mia.plotting.plot_mapping_2d(lle_mapping_2d, hologic_blob_features.index,
                                       phantom_blob_features_subset.index, labels)
          plt.savefig('figures/mappings/blob_lle_mapping_2d.png', dpi=300)
```




```
In [104]: lle_mapping_3d = mia.analysis.lle(selected_features, n_components=3, **lle_kwargs)
```

```
In [124]: mia.plotting.plot_mapping_3d(lle_mapping_3d, hologic_blob_features.index,
                                       phantom_blob_features_subset.index, labels)
```

```
Out[124]: <matplotlib.axes._subplots.Axes3DSubplot at 0x1124fee10>
```

3.4.4 Quality Assessment of Dimensionality Reduction

Assess the quality of the DR against measurements from the co-ranking matrices. First create co-ranking matrices for each of the dimensionality reduction mappings

```
In [106]: max_k = 50
```

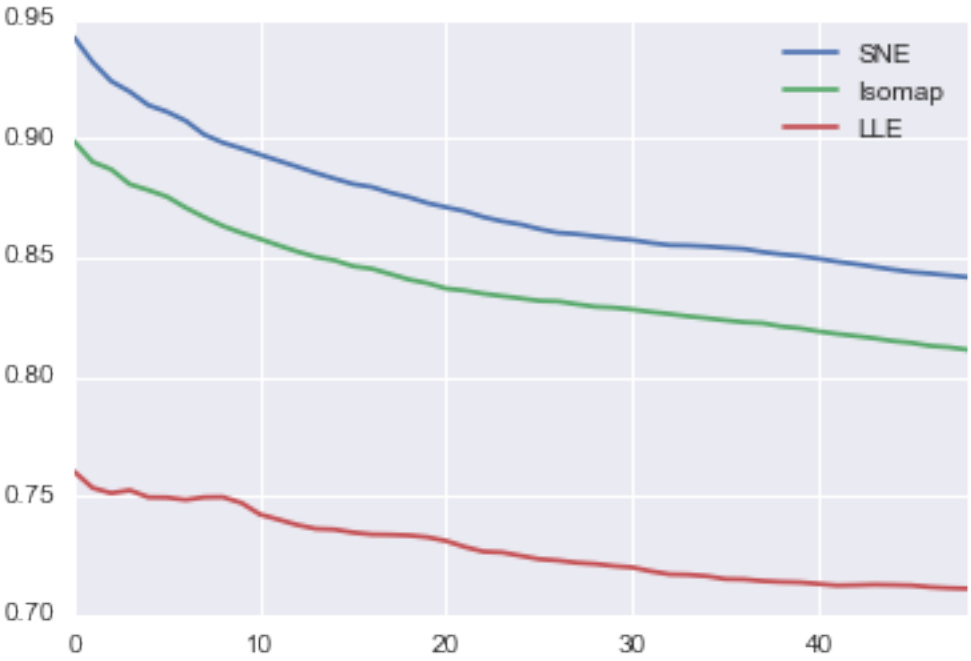
```
In [107]: SNE_mapping_2d_cm = mia.coranking.coranking_matrix(selected_features,
                                                             SNE_mapping_2d)
          iso_mapping_2d_cm = mia.coranking.coranking_matrix(selected_features,
                                                             iso_mapping_2d)
          lle_mapping_2d_cm = mia.coranking.coranking_matrix(selected_features,
                                                             lle_mapping_2d)

          SNE_mapping_3d_cm = mia.coranking.coranking_matrix(selected_features,
                                                             SNE_mapping_3d)
          iso_mapping_3d_cm = mia.coranking.coranking_matrix(selected_features,
                                                             iso_mapping_3d)
          lle_mapping_3d_cm = mia.coranking.coranking_matrix(selected_features,
                                                             lle_mapping_3d)
```

3.4.4.1 2D Mappings

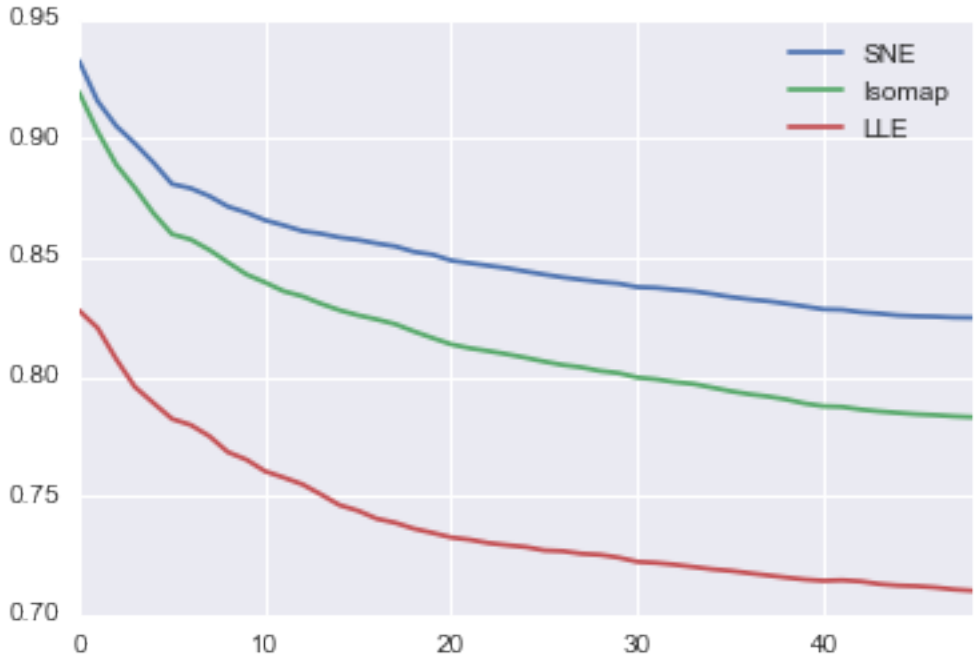
```
In [108]: SNE_trustworthiness_2d = [mia.coranking.trustworthiness(SNE_mapping_2d_cm, k)
                                    for k in range(1, max_k)]
          iso_trustworthiness_2d = [mia.coranking.trustworthiness(iso_mapping_2d_cm, k)
                                    for k in range(1, max_k)]
          lle_trustworthiness_2d = [mia.coranking.trustworthiness(lle_mapping_2d_cm, k)
                                    for k in range(1, max_k)]
```

```
In [109]: trustworthiness_df = pd.DataFrame([SNE_trustworthiness_2d,
                                             iso_trustworthiness_2d,
                                             lle_trustworthiness_2d],
                                             index=['SNE', 'Isomap', 'LLE']).T
          trustworthiness_df.plot()
          plt.savefig('figures/quality_measures/blob_trustworthiness_2d.png', dpi=300)
```



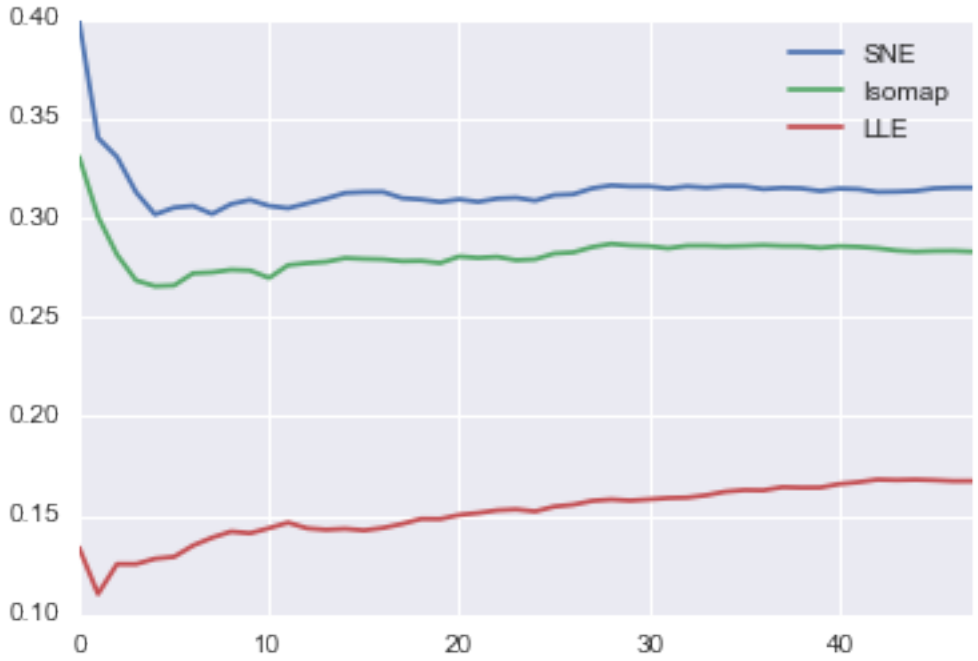

```
In [110]: SNE_continuity_2d = [mia.coranking.continuity(SNE_mapping_2d_cm, k)
                              for k in range(1, max_k)]
          iso_continuity_2d = [mia.coranking.continuity(iso_mapping_2d_cm, k)
                              for k in range(1, max_k)]
          lle_continuity_2d = [mia.coranking.continuity(lle_mapping_2d_cm, k)
                              for k in range(1, max_k)]
```

```
In [111]: continuity_df = pd.DataFrame([SNE_continuity_2d,
                                        iso_continuity_2d,
                                        lle_continuity_2d],
                                        index=['SNE', 'Isomap', 'LLE']).T
          continuity_df.plot()
          plt.savefig('figures/quality_measures/blob_continuity_2d.png', dpi=300)
```




```
In [112]: SNE_lcmc_2d = [mia.coranking.LCMC(SNE_mapping_2d_cm, k)
                         for k in range(2, max_k)]
          iso_lcmc_2d = [mia.coranking.LCMC(iso_mapping_2d_cm, k)
                         for k in range(2, max_k)]
          lle_lcmc_2d = [mia.coranking.LCMC(lle_mapping_2d_cm, k)
                         for k in range(2, max_k)]
```

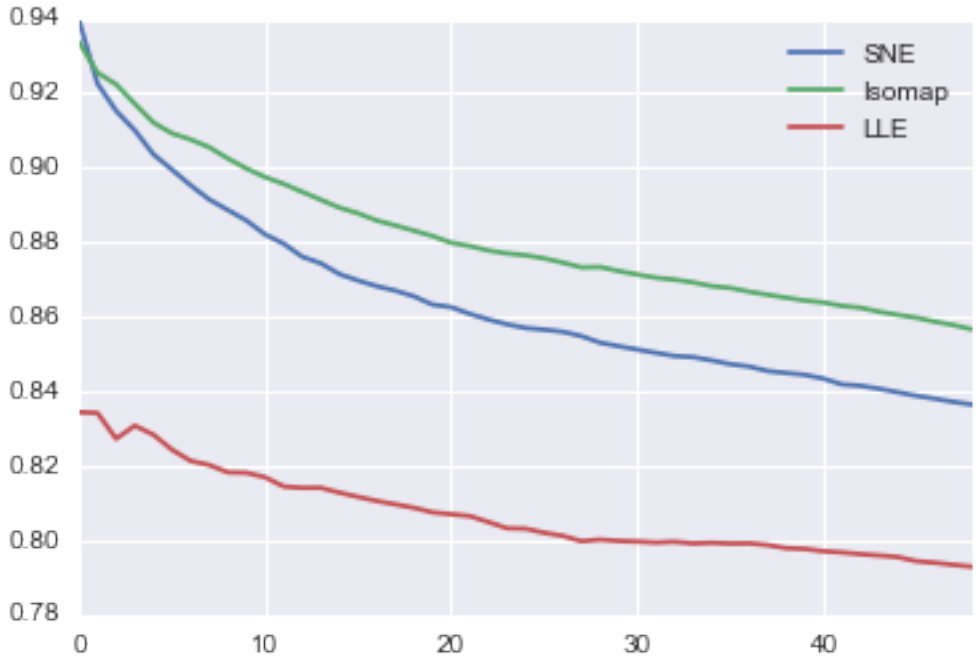
```
In [113]: lcmc_df = pd.DataFrame([SNE_lcmc_2d,
                                 iso_lcmc_2d,
                                 lle_lcmc_2d],
                                 index=['SNE', 'Isomap', 'LLE']).T
          lcmc_df.plot()
          plt.savefig('figures/quality_measures/blob_lcmc_2d.png', dpi=300)
```



3.4.4.2 3D Mappings

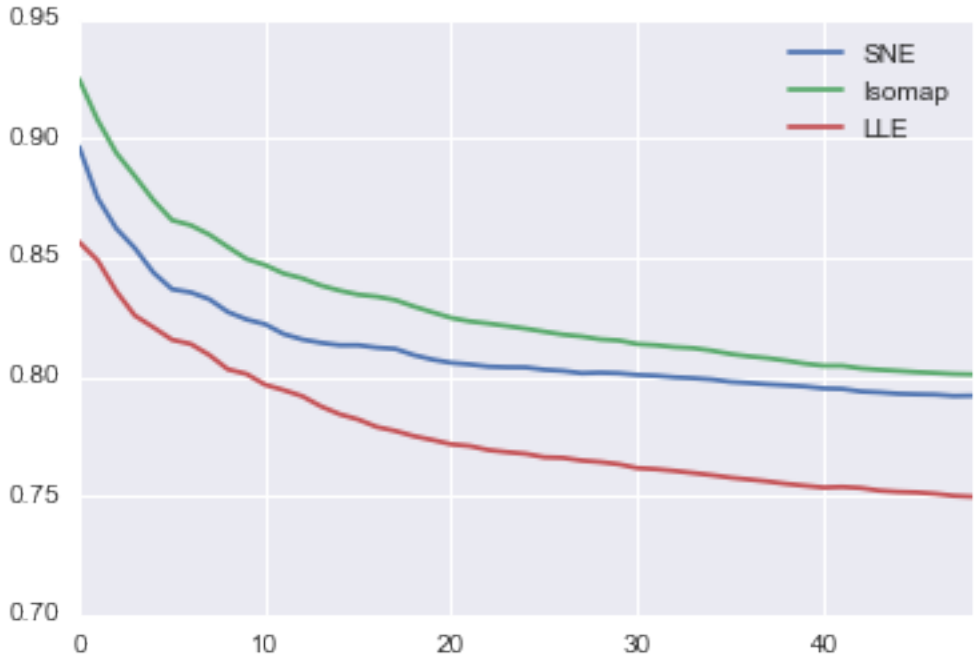
```
In [114]: SNE_trustworthiness_3d = [mia.coranking.trustworthiness(SNE_mapping_3d_cm, k)
                                    for k in range(1, max_k)]
          iso_trustworthiness_3d = [mia.coranking.trustworthiness(iso_mapping_3d_cm, k)
                                    for k in range(1, max_k)]
          lle_trustworthiness_3d = [mia.coranking.trustworthiness(lle_mapping_3d_cm, k)
                                    for k in range(1, max_k)]


In [115]: trustworthiness3d_df = pd.DataFrame([SNE_trustworthiness_3d,
                                                iso_trustworthiness_3d,
                                                lle_trustworthiness_3d],
                                                index=['SNE', 'Isomap', 'LLE']).T
          trustworthiness3d_df.plot()
          plt.savefig('figures/quality_measures/blob_trustworthiness_3d.png', dpi=300)
```



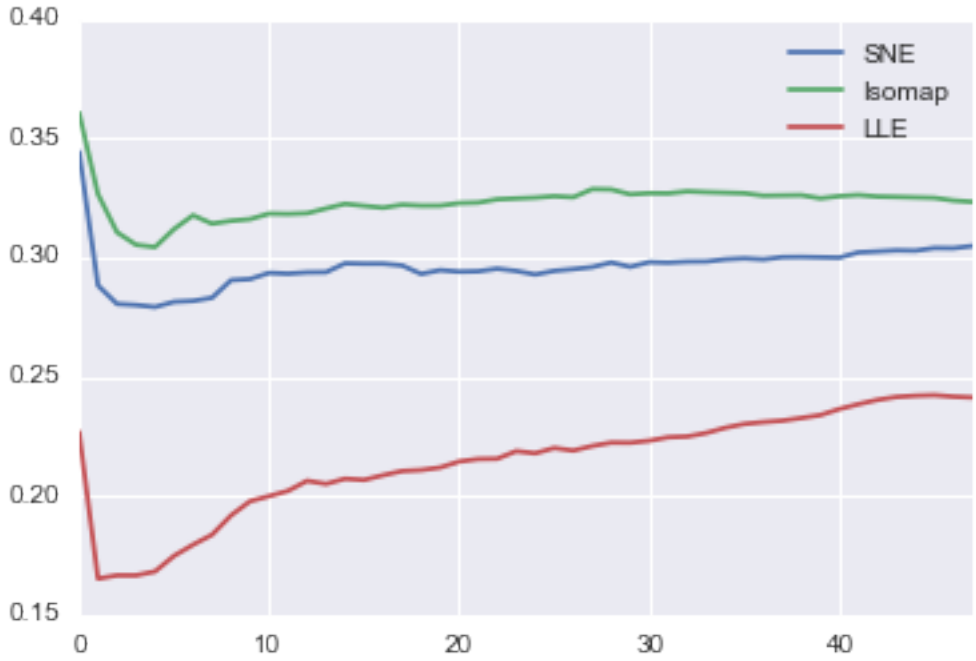

```
In [116]: SNE_continuity_3d = [mia.coranking.continuity(SNE_mapping_3d_cm, k)
                               for k in range(1, max_k)]
          iso_continuity_3d = [mia.coranking.continuity(iso_mapping_3d_cm, k)
                               for k in range(1, max_k)]
          lle_continuity_3d = [mia.coranking.continuity(lle_mapping_3d_cm, k)
                               for k in range(1, max_k)]
```

```
In [117]: continuity3d_df = pd.DataFrame([SNE_continuity_3d,
                                          iso_continuity_3d,
                                          lle_continuity_3d],
                                          index=['SNE', 'Isomap', 'LLE']).T
          continuity3d_df.plot()
          plt.savefig('figures/quality_measures/blob_continuity_3d.png', dpi=300)
```




```
In [118]: SNE_lcmc_3d = [mia.coranking.LCMC(SNE_mapping_3d_cm, k)
                         for k in range(2, max_k)]
          iso_lcmc_3d = [mia.coranking.LCMC(iso_mapping_3d_cm, k)
                         for k in range(2, max_k)]
          lle_lcmc_3d = [mia.coranking.LCMC(lle_mapping_3d_cm, k)
                         for k in range(2, max_k)]
```

```
In [119]: lcmc3d_df = pd.DataFrame([SNE_lcmc_3d,
                                   iso_lcmc_3d,
                                   lle_lcmc_3d],
                                   index=['SNE', 'Isomap', 'LLE']).T
          lcmc3d_df.plot()
          plt.savefig('figures/quality_measures/blob_lcmc_3d.png', dpi=300)
```



Blob Intensity Analysis

```
In [98]: %matplotlib qt
         import pandas as pd
         import numpy as np
         import scipy.stats as stats
         import matplotlib.pyplot as plt
         import mia
```

```
Warning: Cannot change to a different GUI toolkit: qt. Using osx instead.
```

3.5 Loading and Preprocessing

Loading the hologic and synthetic datasets.

```
In [57]: hologic = pd.DataFrame.from_csv("real_intensity.csv")
         hologic.drop(hologic.columns[:2], axis=1, inplace=True)
         hologic.drop('breast_area', axis=1, inplace=True)

         phantom = pd.DataFrame.from_csv("synthetic_intensity.csv")
         phantom.drop(phantom.columns[:2], axis=1, inplace=True)
         phantom.drop('breast_area', axis=1, inplace=True)
```

Loading the meta data for the real and synthetic datasets.

```
In [58]: hologic_meta = mia.analysis.create_hologic_meta_data(hologic, "meta_data/real_meta.csv")
         phantom_meta = mia.analysis.create_synthetic_meta_data(phantom,
                                                                "meta_data/synthetic_meta.csv")
         phantom_meta.index.name = 'img_name'
```

Prepare the BI-RADS/VBD labels for both datasets.

```
In [59]: hologic_labels = hologic_meta.drop_duplicates().BIRADS
         phantom_labels = phantom_meta['VBD.1']

         class_labels = pd.concat([hologic_labels, phantom_labels])
         class_labels.index.name = "img_name"
         labels = mia.analysis.remove_duplicate_index(class_labels)[0]
```

3.6 Creating Features

Create blob features from distribution of blobs

```
In [60]: hologic_intensity_features = mia.analysis.group_by_scale_space(hologic)
         phantom_intensity_features = mia.analysis.group_by_scale_space(phantom)
```

Take a random subset of the phantom mammograms. This is important so that each case is not over represented.

```
In [61]: syn_feature_meta = mia.analysis.remove_duplicate_index(phantom_meta)
         phantom_intensity_features['phantom_name'] = syn_feature_meta.phantom_name.tolist()
         phantom_intensity_features_subset \
             = mia.analysis.create_random_subset(phantom_intensity_features, 'phantom_name')
```

Combine the features from both datasets.

```
In [62]: features = pd.concat([hologic_intensity_features, phantom_intensity_features_subset])
         assert features.shape[0] == 366
         features.head()
```

```
Out[62]:                         count      mean       std       min       25%  \
         p214-010-60001-cl.png     256  0.610387  0.076183  0.411502  0.555738
         p214-010-60001-cr.png     256  0.558904  0.087279  0.328763  0.510450
         p214-010-60001-ml.png     256  0.566832  0.079640  0.349452  0.515433
         p214-010-60001-mr.png     256  0.534146  0.093002  0.287792  0.476245
         p214-010-60005-cl.png     256  0.613984  0.074093  0.396148  0.565591

                                    50%       75%       max      skew  kurtosis  \
         p214-010-60001-cl.png  0.613839  0.667148  0.781775 -0.176690 -0.383312
         p214-010-60001-cr.png  0.570363  0.622144  0.724731 -0.406150 -0.127657
         p214-010-60001-ml.png  0.572314  0.625658  0.741338 -0.329743 -0.087993
         p214-010-60001-mr.png  0.543159  0.600705  0.728595 -0.344968  0.028718
         p214-010-60005-cl.png  0.620705  0.668532  0.766788 -0.386427 -0.065749

                                   ...      count_9    mean_9     std_9     min_9  \
         p214-010-60001-cl.png     ...       131044  0.541212  0.118465  0.141845
         p214-010-60001-cr.png     ...       131044  0.541212  0.118465  0.141845
         p214-010-60001-ml.png     ...       131044  0.541212  0.118465  0.141845
         p214-010-60001-mr.png     ...       131044  0.541212  0.118465  0.141845
         p214-010-60005-cl.png     ...       131044  0.683325  0.142954  0.079646

                                  25%_9    50%_9     75%_9     max_9    skew_9  \
         p214-010-60001-cl.png  0.462988  0.54614  0.624509  0.921247 -0.150691
         p214-010-60001-cr.png  0.462988  0.54614  0.624509  0.921247 -0.150691
         p214-010-60001-ml.png  0.462988  0.54614  0.624509  0.921247 -0.150691
         p214-010-60001-mr.png  0.462988  0.54614  0.624509  0.921247 -0.150691
         p214-010-60005-cl.png  0.597345  0.70354  0.792035  0.995575 -0.848023

                                kurtosis_9
         p214-010-60001-cl.png    0.148522
         p214-010-60001-cr.png    0.148522
         p214-010-60001-ml.png    0.148522
         p214-010-60001-mr.png    0.148522
         p214-010-60005-cl.png    0.758542

         [5 rows x 100 columns]
```

Filter some features, such as the min, to remove noise.

```
In [63]: selected_features = features.copy()
```

3.7 Compare Real and Synthetic Features

Compare the distributions of features detected from the real mammograms and the phantoms using the Kolmogorov-Smirnov two sample test.

```
In [106]: ks_stats = [list(stats.ks_2samp(hologic_intensity_features[col],
                                          phantom_intensity_features[col]))
                                          for col in selected_features.columns]

          ks_test = pd.DataFrame(ks_stats, columns=['KS', 'p-value'], index=selected_features.columns)
          ks_test.to_latex("tables/intensity_features_ks.tex", longtable=True)
          ks_test
```

Out[106]:

	KS	p-value
count	0.000000	1.000000e+00
mean	1.000000	3.587622e-59
std	0.876389	1.515491e-45
min	1.000000	3.587622e-59
25%	1.000000	3.587622e-59
50%	1.000000	3.587622e-59
75%	1.000000	3.587622e-59
max	1.000000	3.587622e-59
skew	0.891667	3.923764e-47
kurtosis	0.568056	2.209941e-19
count_1	0.000000	1.000000e+00
mean_1	1.000000	3.587622e-59
std_1	0.981944	4.539903e-57
min_1	1.000000	3.587622e-59
25%_1	1.000000	3.587622e-59
50%_1	1.000000	3.587622e-59
75%_1	1.000000	3.587622e-59
max_1	0.997222	7.598385e-59
skew_1	0.784722	1.335838e-36
kurtosis_1	0.466667	3.216681e-13
count_2	0.000000	1.000000e+00
mean_2	1.000000	3.587622e-59
std_2	1.000000	3.587622e-59
min_2	1.000000	3.587622e-59
25%_2	1.000000	3.587622e-59
50%_2	1.000000	3.587622e-59
75%_2	1.000000	3.587622e-59
max_2	0.994444	1.605940e-58
skew_2	0.501389	3.410114e-15
kurtosis_2	0.436111	1.342503e-11
...	...	...
count_7	0.000000	1.000000e+00
mean_7	1.000000	3.587622e-59
std_7	0.955556	4.577742e-54
min_7	1.000000	3.587622e-59
25%_7	1.000000	3.587622e-59
50%_7	1.000000	3.587622e-59
75%_7	1.000000	3.587622e-59
max_7	0.740278	1.280685e-32
skew_7	0.319444	2.024353e-06
kurtosis_7	0.304167	7.344875e-06
count_8	0.000000	1.000000e+00
mean_8	1.000000	3.587622e-59
std_8	0.929167	3.823290e-51
min_8	1.000000	3.587622e-59
25%_8	1.000000	3.587622e-59
50%_8	1.000000	3.587622e-59
75%_8	1.000000	3.587622e-59
max_8	0.736111	2.943248e-32

```
skew_8      0.547222  5.120902e-18
kurtosis_8  0.395833  1.248634e-09
count_9     0.000000  1.000000e+00
mean_9      1.000000  3.587622e-59
std_9       0.956944  3.195999e-54
min_9       1.000000  3.587622e-59
25%_9       1.000000  3.587622e-59
50%_9       1.000000  3.587622e-59
75%_9       0.997222  7.598385e-59
max_9       0.794444  1.674259e-37
skew_9      0.702778  1.934059e-29
kurtosis_9  0.891667  3.923764e-47

[100 rows x 2 columns]
```

3.8 Dimensionality Reduction

3.8.1 t-SNE

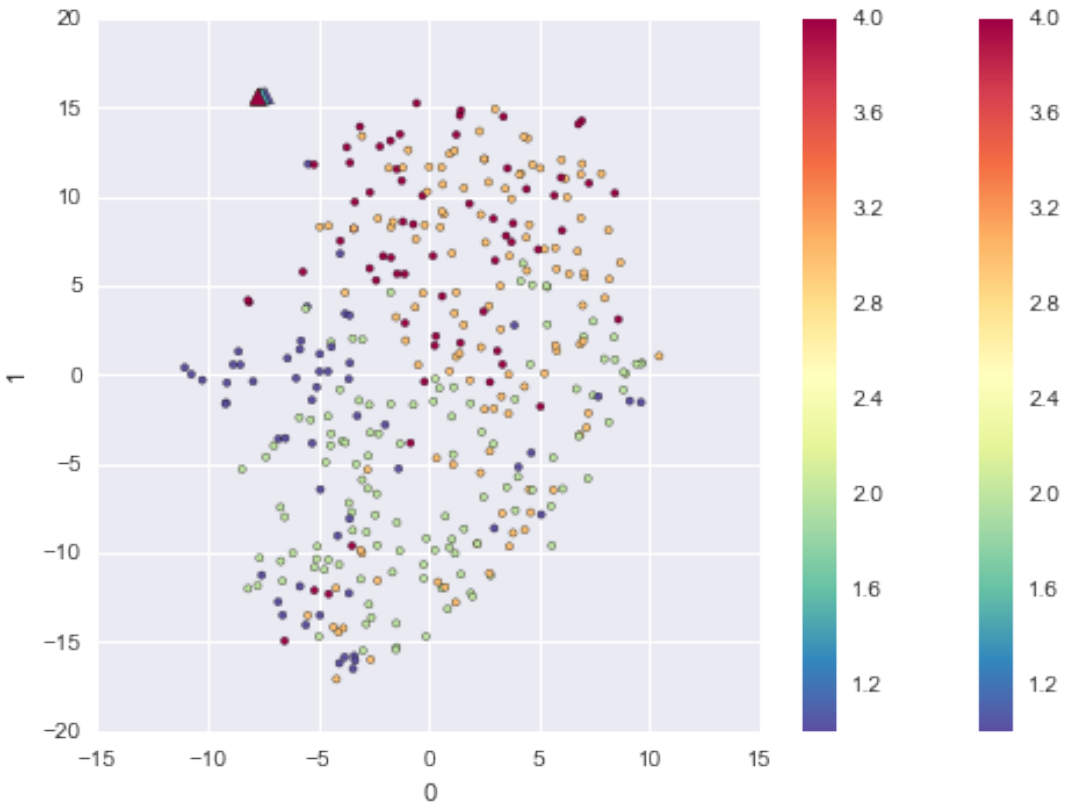
Running t-SNE to obtain a two dimensional representation.

```
In [65]: kwargs = {
             'learning_rate': 300,
             'perplexity': 30,
             'verbose': 1
         }
```

```
In [66]: SNE_mapping_2d = mia.analysis.tSNE(selected_features, n_components=2, **kwargs)
```

```
[t-SNE] Computing pairwise distances...
[t-SNE] Computed conditional probabilities for sample 366 / 366
[t-SNE] Mean sigma: 2.401870
[t-SNE] Error after 83 iterations with early exaggeration: 14.124506
[t-SNE] Error after 276 iterations: 0.673903
```

```
In [95]: mia.plotting.plot_mapping_2d(SNE_mapping_2d, hologic_intensity_features.index,
                                     phantom_intensity_features_subset.index, labels)
         plt.savefig('figures/mappings/intensity_SNE_mapping_2d.png', dpi=300)
```



Running t-SNE to obtain a 3 dimensional mapping

```
In [68]: SNE_mapping_3d = mia.analysis.tSNE(selected_features, n_components=3, **kwargs)
```

```
[t-SNE] Computing pairwise distances...
[t-SNE] Computed conditional probabilities for sample 366 / 366
[t-SNE] Mean sigma: 2.401870
[t-SNE] Error after 100 iterations with early exaggeration: 17.879216
[t-SNE] Error after 346 iterations: 1.711401
```

```
In [99]: mia.plotting.plot_mapping_3d(SNE_mapping_3d, hologic_intensity_features.index,
                                      phantom_intensity_features_subset.index, labels)
```

```
Out[99]: <matplotlib.axes._subplots.Axes3DSubplot at 0x10f3a8690>
```

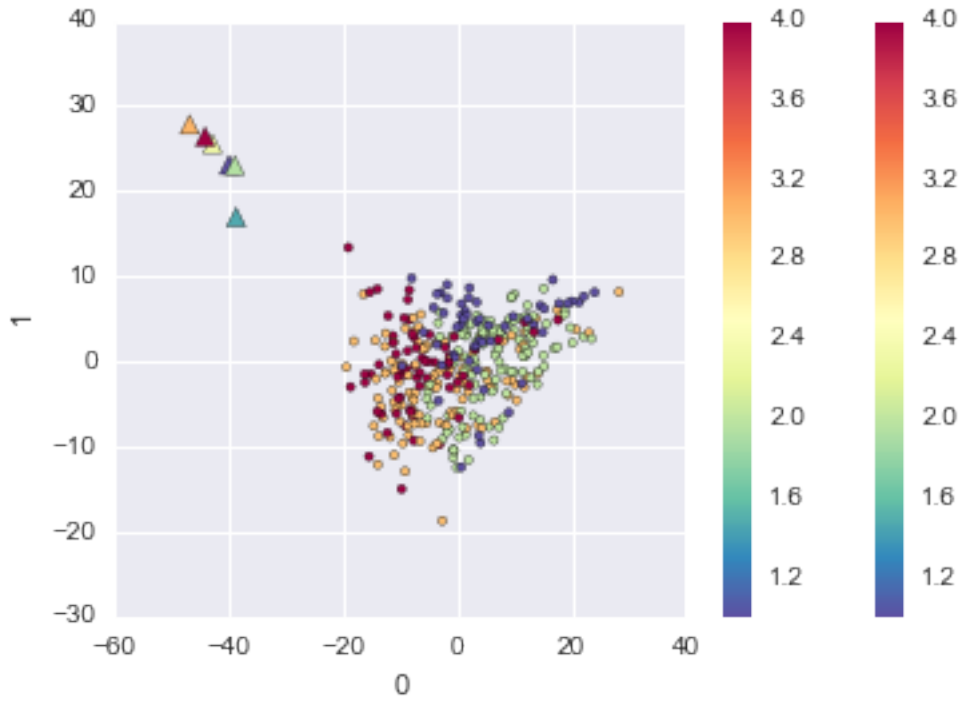
3.8.2 Isomap

Running Isomap to obtain a 2 dimensional mapping

```
In [70]: iso_kwargs = {
             'n_neighbors': 10,
         }
```

```
In [71]: iso_mapping_2d = mia.analysis.isomap(selected_features, n_components=2, **iso_kwargs)
```

```
In [72]: mia.plotting.plot_mapping_2d(iso_mapping_2d, hologic_intensity_features.index,
                                      phantom_intensity_features_subset.index, labels)
         plt.savefig('figures/mappings/intensity_iso_mapping_2d.png', dpi=300)
```




```
In [73]: iso_mapping_3d = mia.analysis.isomap(selected_features, n_components=3, **iso_kwargs)
```

```
In [101]: mia.plotting.plot_mapping_3d(iso_mapping_3d, hologic_intensity_features.index,
                                       phantom_intensity_features_subset.index, labels)
```

```
Out[101]: <matplotlib.axes._subplots.Axes3DSubplot at 0x10e572ed0>
```

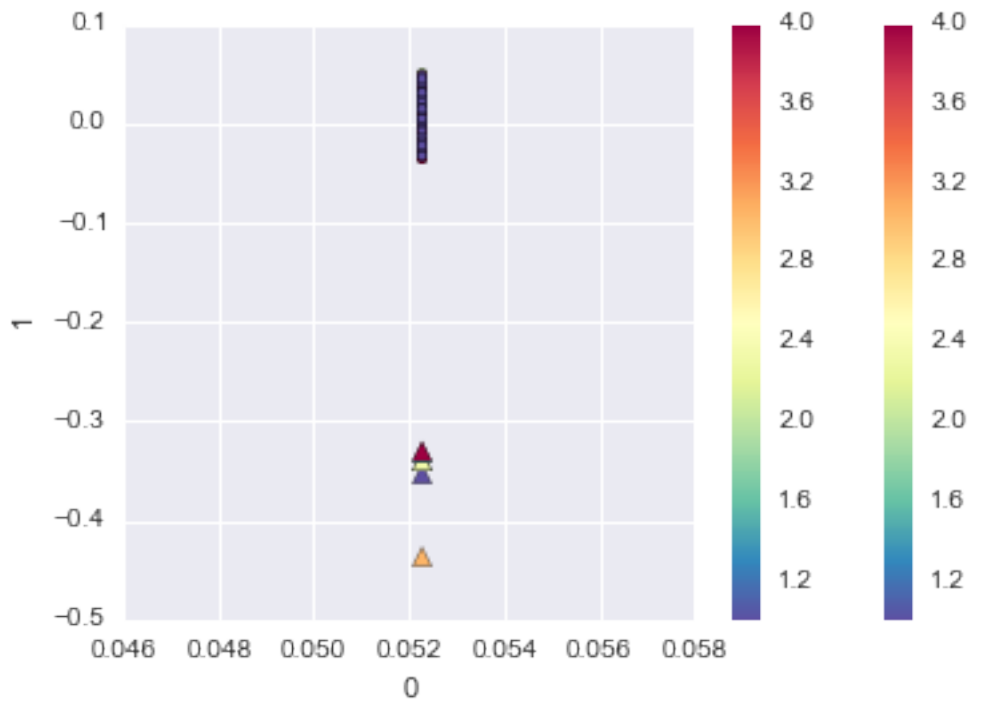
3.8.3 Locally Linear Embedding

Running locally linear embedding to obtain 2d mapping

```
In [75]: lle_kwargs = {
             'n_neighbors': 10,
         }
```

```
In [76]: lle_mapping_2d = mia.analysis.lle(selected_features, n_components=2, **lle_kwargs)
```

```
In [77]: mia.plotting.plot_mapping_2d(lle_mapping_2d, hologic_intensity_features.index,
                                      phantom_intensity_features_subset.index, labels)
         plt.savefig('figures/mappings/intensity_lle_mapping_2d.png', dpi=300)
```




```
In [78]: lle_mapping_3d = mia.analysis.lle(selected_features, n_components=3, **lle_kwargs)
```

```
In [102]: mia.plotting.plot_mapping_3d(lle_mapping_3d, hologic_intensity_features.index,
                                       phantom_intensity_features_subset.index, labels)
```

```
Out[102]: <matplotlib.axes._subplots.Axes3DSubplot at 0x107926a50>
```

3.8.4 Quality Assessment of Dimensionality Reduction

Assess the quality of the DR against measurements from the co-ranking matrices. First create co-ranking matrices for each of the dimensionality reduction mappings

```
In [80]: max_k = 50
```

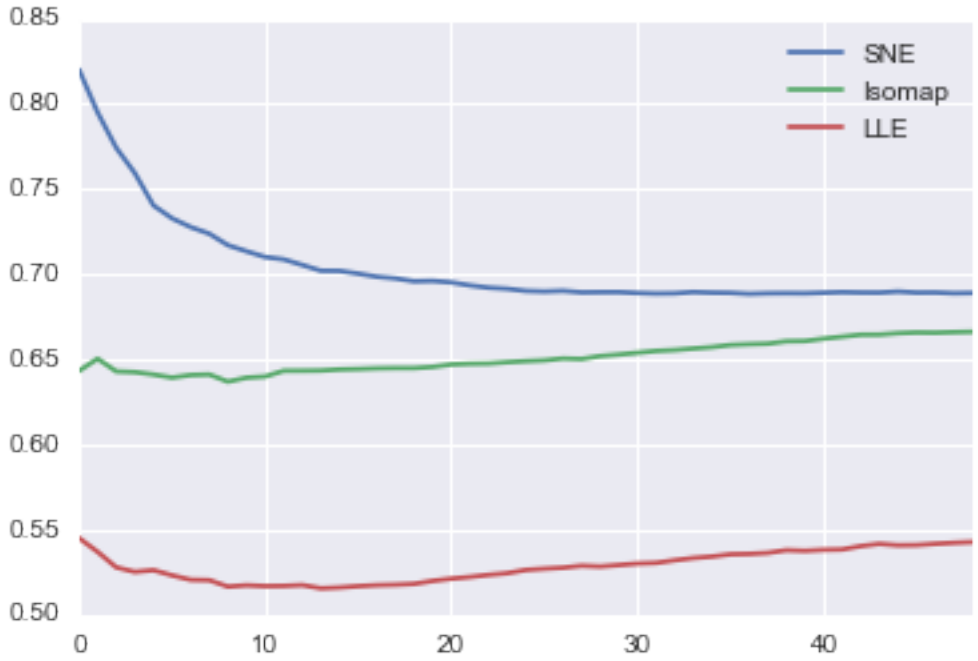
```
In [81]: SNE_mapping_2d_cm = mia.coranking.coranking_matrix(selected_features,
                                                            SNE_mapping_2d)
         iso_mapping_2d_cm = mia.coranking.coranking_matrix(selected_features,
                                                            iso_mapping_2d)
         lle_mapping_2d_cm = mia.coranking.coranking_matrix(selected_features,
                                                            lle_mapping_2d)

         SNE_mapping_3d_cm = mia.coranking.coranking_matrix(selected_features,
                                                            SNE_mapping_3d)
         iso_mapping_3d_cm = mia.coranking.coranking_matrix(selected_features,
                                                            iso_mapping_3d)
         lle_mapping_3d_cm = mia.coranking.coranking_matrix(selected_features,
                                                            lle_mapping_3d)
```

3.8.4.1 2D Mappings

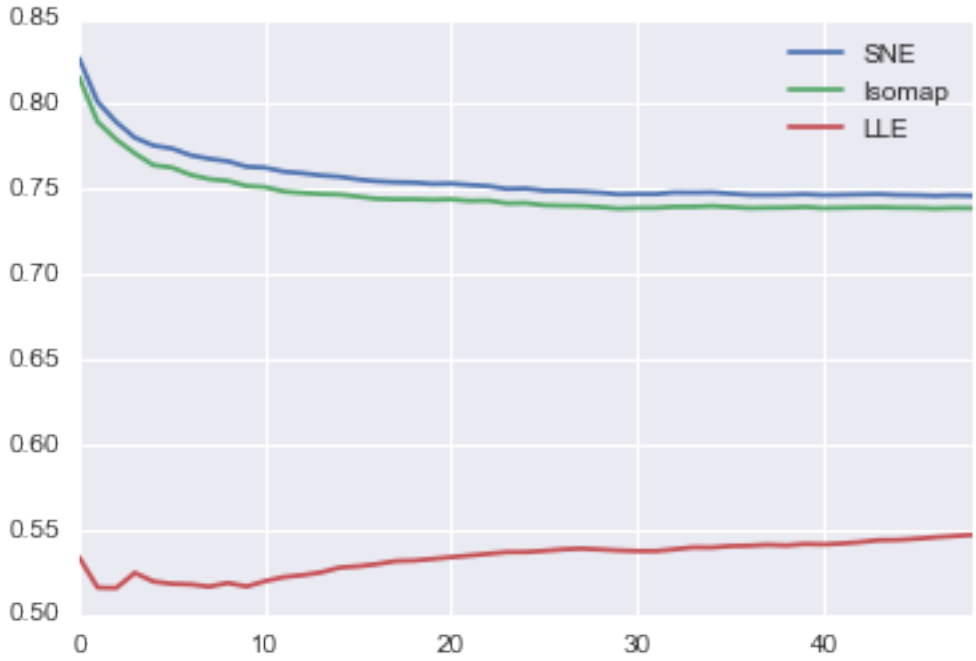
```
In [82]: SNE_trustworthiness_2d = [mia.coranking.trustworthiness(SNE_mapping_2d_cm, k)
                                   for k in range(1, max_k)]
         iso_trustworthiness_2d = [mia.coranking.trustworthiness(iso_mapping_2d_cm, k)
                                   for k in range(1, max_k)]
         lle_trustworthiness_2d = [mia.coranking.trustworthiness(lle_mapping_2d_cm, k)
                                   for k in range(1, max_k)]
```

```
In [83]: trustworthiness_df = pd.DataFrame([SNE_trustworthiness_2d,
                                            iso_trustworthiness_2d,
                                            lle_trustworthiness_2d],
                                            index=['SNE', 'Isomap', 'LLE']).T
         trustworthiness_df.plot()
         plt.savefig('figures/quality_measures/intensity_trustworthiness_2d.png', dpi=300)
```



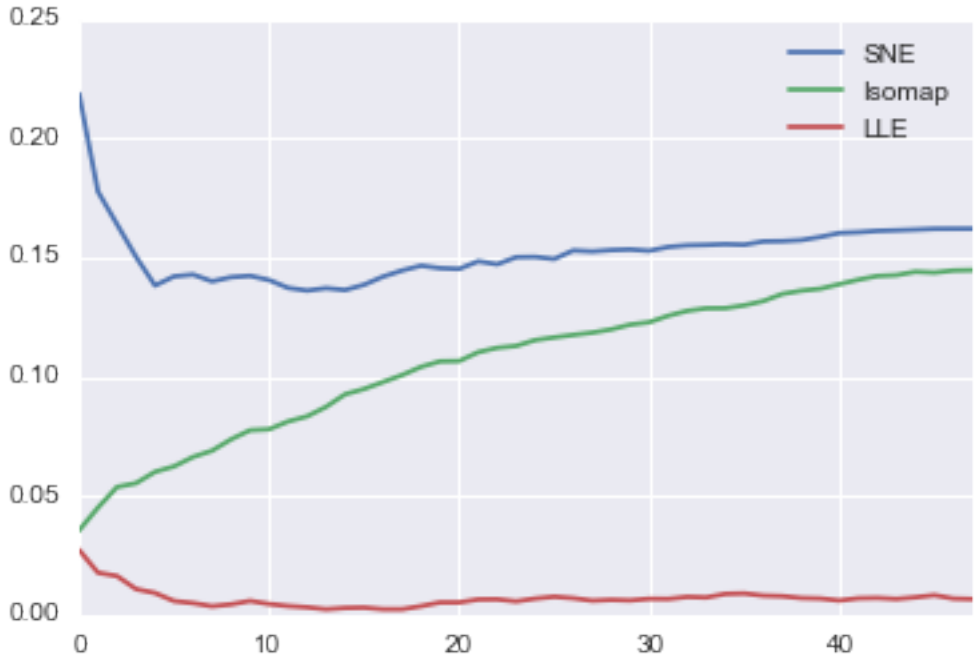

```
In [84]: SNE_continuity_2d = [mia.coranking.continuity(SNE_mapping_2d_cm, k)
                              for k in range(1, max_k)]
         iso_continuity_2d = [mia.coranking.continuity(iso_mapping_2d_cm, k)
                              for k in range(1, max_k)]
         lle_continuity_2d = [mia.coranking.continuity(lle_mapping_2d_cm, k)
                              for k in range(1, max_k)]
```

```
In [85]: continuity_df = pd.DataFrame([SNE_continuity_2d,
                                       iso_continuity_2d,
                                       lle_continuity_2d],
                                       index=['SNE', 'Isomap', 'LLE']).T
         continuity_df.plot()
         plt.savefig('figures/quality_measures/intensity_continuity_2d.png', dpi=300)
```




```
In [86]: SNE_lcmc_2d = [mia.coranking.LCMC(SNE_mapping_2d_cm, k)
                        for k in range(2, max_k)]
         iso_lcmc_2d = [mia.coranking.LCMC(iso_mapping_2d_cm, k)
                        for k in range(2, max_k)]
         lle_lcmc_2d = [mia.coranking.LCMC(lle_mapping_2d_cm, k)
                        for k in range(2, max_k)]
```

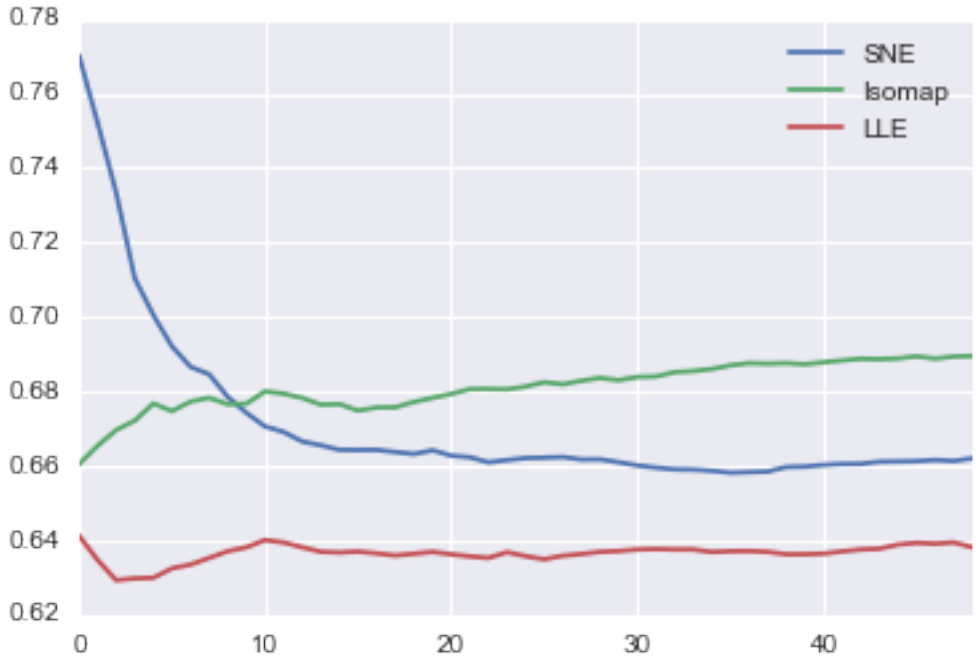
```
In [87]: lcmc_df = pd.DataFrame([SNE_lcmc_2d,
                                iso_lcmc_2d,
                                lle_lcmc_2d],
                                index=['SNE', 'Isomap', 'LLE']).T
         lcmc_df.plot()
         plt.savefig('figures/quality_measures/intensity_lcmc_2d.png', dpi=300)
```



3.8.4.2 3D Mappings

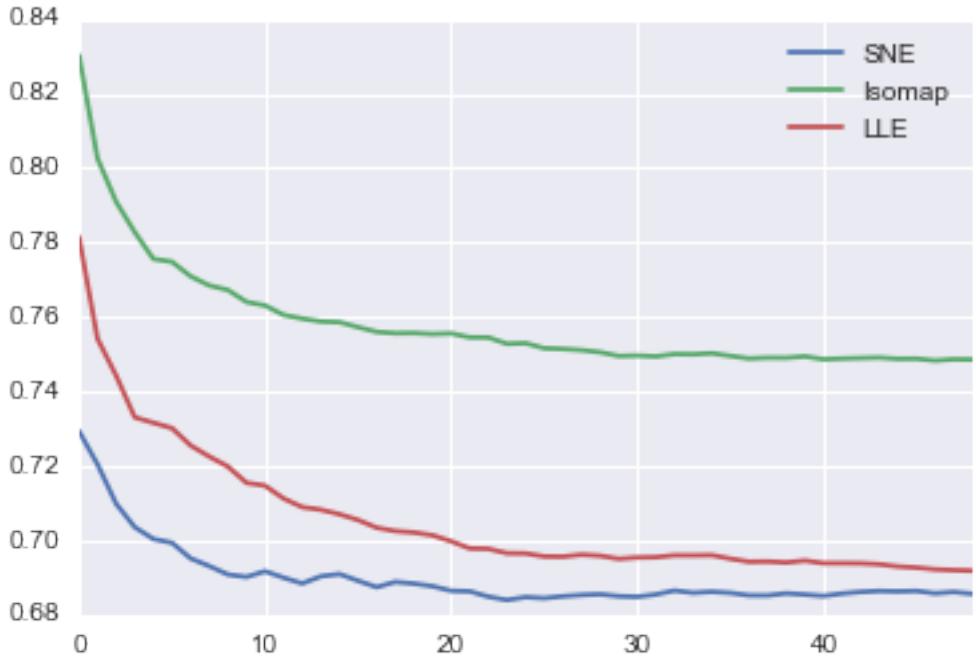
```
In [88]: SNE_trustworthiness_3d = [mia.coranking.trustworthiness(SNE_mapping_3d_cm, k)
                                   for k in range(1, max_k)]
         iso_trustworthiness_3d = [mia.coranking.trustworthiness(iso_mapping_3d_cm, k)
                                   for k in range(1, max_k)]
         lle_trustworthiness_3d = [mia.coranking.trustworthiness(lle_mapping_3d_cm, k)
                                   for k in range(1, max_k)]


In [89]: trustworthiness3d_df = pd.DataFrame([SNE_trustworthiness_3d,
                                               iso_trustworthiness_3d,
                                               lle_trustworthiness_3d],
                                               index=['SNE', 'Isomap', 'LLE']).T
         trustworthiness3d_df.plot()
         plt.savefig('figures/quality_measures/intensity_trustworthiness_3d.png', dpi=300)
```



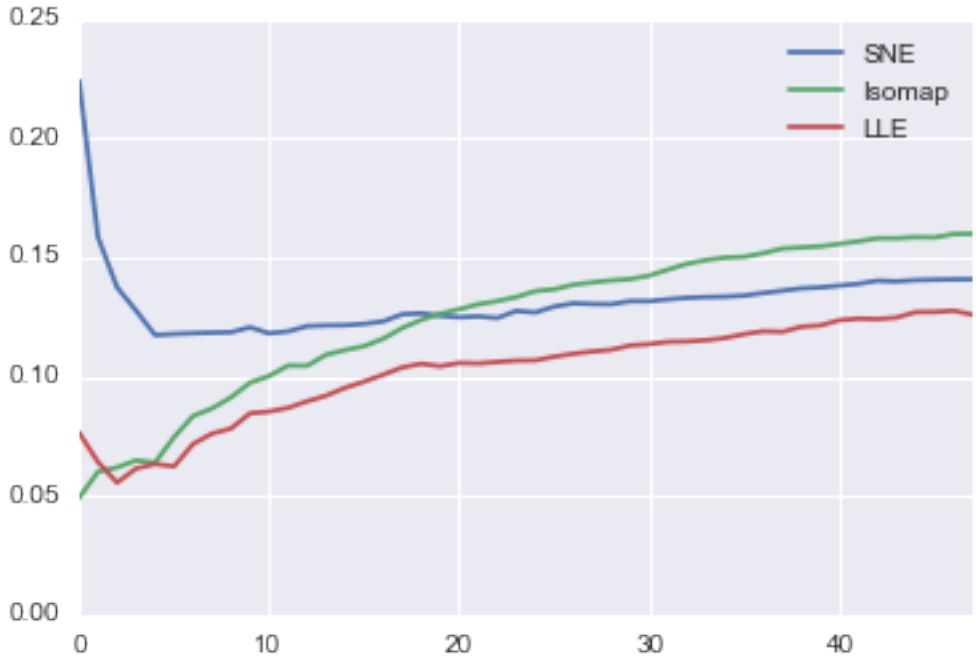

```
In [90]: SNE_continuity_3d = [mia.coranking.continuity(SNE_mapping_3d_cm, k)
                              for k in range(1, max_k)]
         iso_continuity_3d = [mia.coranking.continuity(iso_mapping_3d_cm, k)
                              for k in range(1, max_k)]
         lle_continuity_3d = [mia.coranking.continuity(lle_mapping_3d_cm, k)
                              for k in range(1, max_k)]
```

```
In [91]: continuity3d_df = pd.DataFrame([SNE_continuity_3d,
                                         iso_continuity_3d,
                                         lle_continuity_3d],
                                         index=['SNE', 'Isomap', 'LLE']).T
         continuity3d_df.plot()
         plt.savefig('figures/quality_measures/intensity_continuity_3d.png', dpi=300)
```




```
In [92]: SNE_lcmc_3d = [mia.coranking.LCMC(SNE_mapping_3d_cm, k)
                        for k in range(2, max_k)]
         iso_lcmc_3d = [mia.coranking.LCMC(iso_mapping_3d_cm, k)
                        for k in range(2, max_k)]
         lle_lcmc_3d = [mia.coranking.LCMC(lle_mapping_3d_cm, k)
                        for k in range(2, max_k)]
```

```
In [93]: lcmc3d_df = pd.DataFrame([SNE_lcmc_3d,
                                  iso_lcmc_3d,
                                  lle_lcmc_3d],
                                  index=['SNE', 'Isomap', 'LLE']).T
         lcmc3d_df.plot()
         plt.savefig('figures/quality_measures/intensity_lcmc_3d.png', dpi=300)
```



Blob Texture Analysis

```
In [41]: %matplotlib qt
         import pandas as pd
         import numpy as np
         import scipy.stats as stats
         import matplotlib.pyplot as plt
         import mia
```

```
Warning: Cannot change to a different GUI toolkit: qt. Using osx instead.
```

3.9 Loading and Preprocessing

Loading the hologic and synthetic datasets.

```
In [3]: hologic = pd.DataFrame.from_csv("real_texture.csv")
        hologic.drop(hologic.columns[:2], axis=1, inplace=True)
        hologic.drop('breast_area', axis=1, inplace=True)

        phantom = pd.DataFrame.from_csv("synthetic_texture.csv")
        phantom.drop(phantom.columns[:2], axis=1, inplace=True)
        phantom.drop('breast_area', axis=1, inplace=True)
```

Loading the meta data for the real and synthetic datasets.

```
In [4]: hologic_meta = mia.analysis.create_hologic_meta_data(hologic, "meta_data/real_meta.csv")
        phantom_meta = mia.analysis.create_synthetic_meta_data(phantom,
                                                               "meta_data/synthetic_meta.csv")
        phantom_meta.index.name = 'img_name'
```

Prepare the BI-RADS/VBD labels for both datasets.

```
In [5]: hologic_labels = hologic_meta.drop_duplicates().BIRADS
        phantom_labels = phantom_meta['VBD.1']

        class_labels = pd.concat([hologic_labels, phantom_labels])
        class_labels.index.name = "img_name"
        labels = mia.analysis.remove_duplicate_index(class_labels)[0]
```

3.10 Creating Features

Create blob features from distribution of blobs

```
In [6]: hologic_texture_features = mia.analysis.group_by_scale_space(hologic)
        phantom_texture_features = mia.analysis.group_by_scale_space(phantom)
```

Take a random subset of the phantom mammograms. This is important so that each case is not over represented.

```
In [7]: syn_feature_meta = mia.analysis.remove_duplicate_index(phantom_meta)
        phantom_texture_features['phantom_name'] = syn_feature_meta.phantom_name.tolist()
        phantom_texture_features_subset = mia.analysis.create_random_subset(phantom_texture_features,
                                                                            'phantom_name')
```

Combine the features from both datasets.

```
In [8]: features = pd.concat([hologic_texture_features, phantom_texture_features_subset])
        assert features.shape[0] == 366
        features.head()
```

Out[8]:

	contrast	dissimilarity	homogeneity	energy	\
p214-010-60001-cl.png	180.015284	10.623496	0.092774	0.068239	
p214-010-60001-cr.png	241.273806	10.449699	0.101719	0.073124	
p214-010-60001-ml.png	217.490041	11.182093	0.089156	0.068559	
p214-010-60001-mr.png	288.353799	11.644971	0.093407	0.073823	
p214-010-60005-cl.png	160.227140	10.034384	0.099470	0.069002	

	contrast_1	dissimilarity_1	homogeneity_1	energy_1	\
p214-010-60001-cl.png	281.402566	10.773226	0.103179	0.056349	
p214-010-60001-cr.png	257.052029	10.725951	0.098164	0.051589	
p214-010-60001-ml.png	255.275936	11.383161	0.091460	0.052581	
p214-010-60001-mr.png	237.048490	11.298087	0.089295	0.050686	
p214-010-60005-cl.png	157.603883	9.991076	0.096144	0.051515	

	contrast_2	dissimilarity_2	...	homogeneity_7	\
p214-010-60001-cl.png	166.662740	10.250118	...	0.099589	
p214-010-60001-cr.png	236.447806	10.048496	...	0.104395	
p214-010-60001-ml.png	223.895510	10.859934	...	0.094498	
p214-010-60001-mr.png	173.248493	10.351740	...	0.099926	
p214-010-60005-cl.png	148.042716	9.647045	...	0.106255	

	energy_7	contrast_8	dissimilarity_8	homogeneity_8	\
p214-010-60001-cl.png	0.016556	163.937053	10.084599	0.099519	
p214-010-60001-cr.png	0.015297	166.288393	9.918993	0.107002	
p214-010-60001-ml.png	0.014957	171.293827	10.346745	0.094648	
p214-010-60001-mr.png	0.015496	166.981656	10.147627	0.096699	
p214-010-60005-cl.png	0.017021	166.288393	9.918993	0.107002	

	energy_8	contrast_9	dissimilarity_9	homogeneity_9	\
p214-010-60001-cl.png	0.013505	157.454233	9.781551	0.105151	
p214-010-60001-cr.png	0.022819	157.454233	9.781551	0.105151	
p214-010-60001-ml.png	0.014245	157.454233	9.781551	0.105151	
p214-010-60001-mr.png	0.013841	157.454233	9.781551	0.105151	

```
p214-010-60005-cl.png  0.022819  127.449202         8.913421       0.110486

                       energy_9
p214-010-60001-cl.png  0.019975
p214-010-60001-cr.png  0.019975
p214-010-60001-ml.png  0.019975
p214-010-60001-mr.png  0.019975
p214-010-60005-cl.png  0.016995

[5 rows x 40 columns]
```

Filter some features, such as the min, to remove noise.

```
In [9]: selected_features = features.copy()
```

3.11 Compare Real and Synthetic Features

Compare the distributions of features detected from the real mammograms and the phantoms using the Kolmogorov-Smirnov two sample test.

```
In [80]: ks_stats = [list(stats.ks_2samp(hologic_texture_features[col],
                                        phantom_texture_features[col]))
                                        for col in hologic_texture_features.columns]

         ks_test = pd.DataFrame(ks_stats, columns=['KS', 'p-value'],
                                index=hologic_texture_features.columns)
         ks_test.to_latex("tables/texture_features_ks.tex")
         ks_test
```

Out[80]:

	KS	p-value
contrast	0.381944	5.383186e-09
dissimilarity	1.000000	3.587622e-59
homogeneity	1.000000	3.587622e-59
energy	1.000000	3.587622e-59
contrast_1	0.586111	1.318746e-20
dissimilarity_1	1.000000	3.587622e-59
homogeneity_1	1.000000	3.587622e-59
energy_1	1.000000	3.587622e-59
contrast_2	0.863889	2.874007e-44
dissimilarity_2	1.000000	3.587622e-59
homogeneity_2	1.000000	3.587622e-59
energy_2	1.000000	3.587622e-59
contrast_3	0.923611	1.538596e-50
dissimilarity_3	1.000000	3.587622e-59
homogeneity_3	1.000000	3.587622e-59
energy_3	1.000000	3.587622e-59
contrast_4	0.845833	1.870608e-42
dissimilarity_4	1.000000	3.587622e-59
homogeneity_4	1.000000	3.587622e-59
energy_4	1.000000	3.587622e-59
contrast_5	0.979167	9.485680e-57
dissimilarity_5	1.000000	3.587622e-59
homogeneity_5	1.000000	3.587622e-59
energy_5	1.000000	3.587622e-59
contrast_6	1.000000	3.587622e-59
dissimilarity_6	1.000000	3.587622e-59
homogeneity_6	1.000000	3.587622e-59
energy_6	0.994444	1.605940e-58
contrast_7	0.969444	1.230285e-55
dissimilarity_7	1.000000	3.587622e-59
homogeneity_7	1.000000	3.587622e-59
energy_7	1.000000	3.587622e-59
contrast_8	0.986111	1.497318e-57
dissimilarity_8	1.000000	3.587622e-59
homogeneity_8	1.000000	3.587622e-59
energy_8	0.997222	7.598385e-59

```
contrast_9       1.000000  3.587622e-59
dissimilarity_9  1.000000  3.587622e-59
homogeneity_9    1.000000  3.587622e-59
energy_9         1.000000  3.587622e-59
```

3.12 Dimensionality Reduction

3.12.1 t-SNE

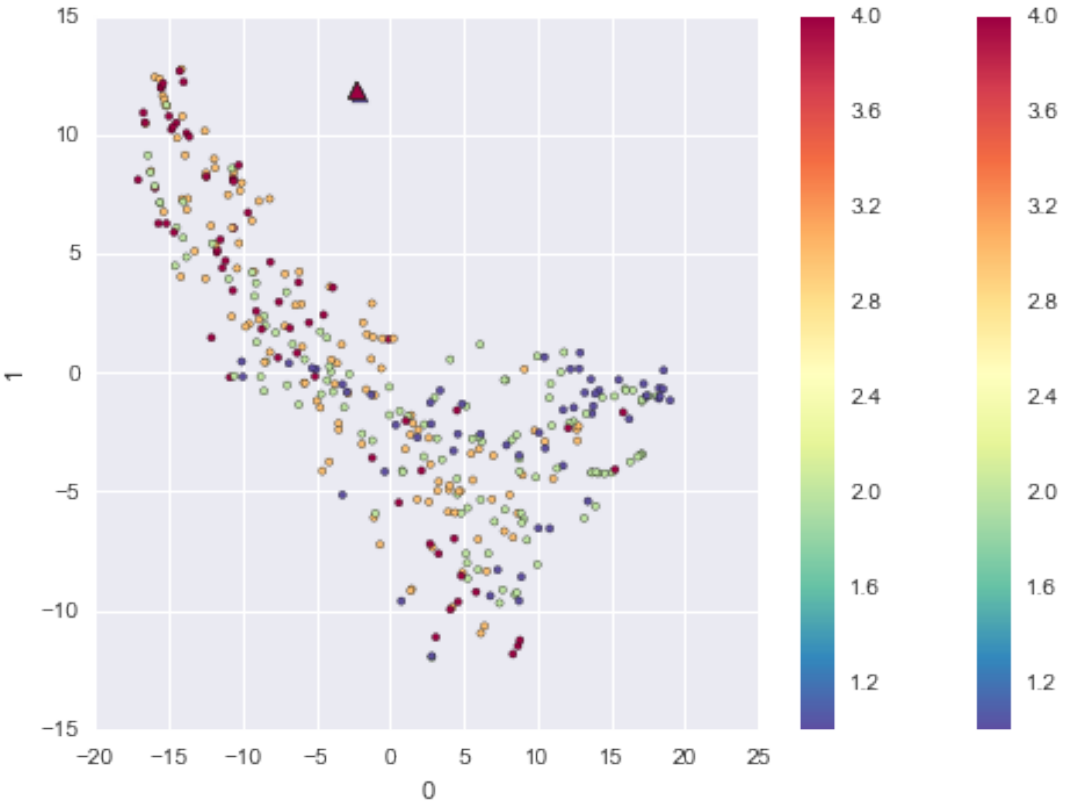
Running t-SNE to obtain a two dimensional representation.

```
In [11]: kwargs = {
             'learning_rate': 300,
             'perplexity': 30,
             'verbose': 1
         }
```

```
In [12]: SNE_mapping_2d = mia.analysis.tSNE(selected_features, n_components=2, **kwargs)
```

```
[t-SNE] Computing pairwise distances...
[t-SNE] Computed conditional probabilities for sample 366 / 366
[t-SNE] Mean sigma: 0.834920
[t-SNE] Error after 83 iterations with early exaggeration: 12.847253
[t-SNE] Error after 296 iterations: 0.467811
```

```
In [13]: mia.plotting.plot_mapping_2d(SNE_mapping_2d, hologic_texture_features.index,
                                     phantom_texture_features_subset.index, labels)
         plt.savefig('figures/mappings/texture_SNE_mapping_2d.png', dpi=300)
```



Running t-SNE to obtain a 3 dimensional mapping

```
In [14]: SNE_mapping_3d = mia.analysis.tSNE(selected_features, n_components=3, **kwargs)
```

```
[t-SNE] Computing pairwise distances...
[t-SNE] Computed conditional probabilities for sample 366 / 366
[t-SNE] Mean sigma: 0.834920
[t-SNE] Error after 100 iterations with early exaggeration: 14.777595
[t-SNE] Error after 271 iterations: 0.948866
```

```
In [42]: mia.plotting.plot_mapping_3d(SNE_mapping_3d, hologic_texture_features.index,
                                     phantom_texture_features_subset.index, labels)
```

```
Out[42]: <matplotlib.axes._subplots.Axes3DSubplot at 0x110926f90>
```

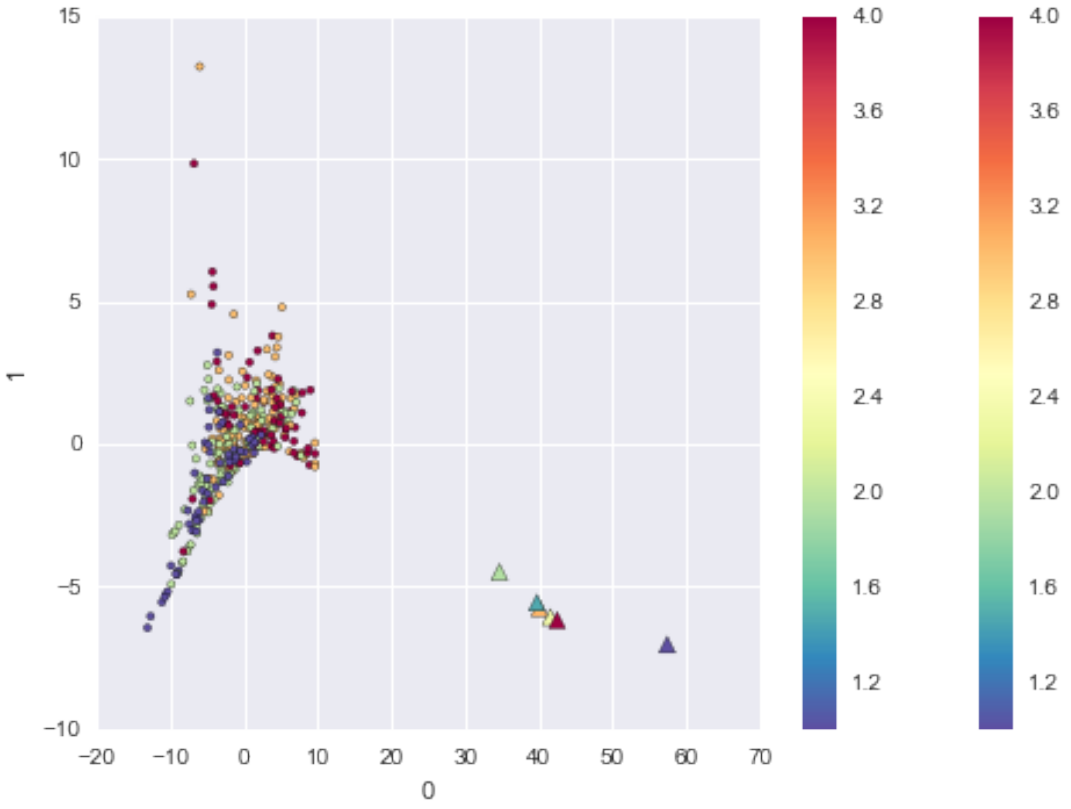
3.12.2 Isomap

Running Isomap to obtain a 2 dimensional mapping

```
In [16]: iso_kwargs = {
             'n_neighbors': 10,
         }
```

```
In [17]: iso_mapping_2d = mia.analysis.isomap(selected_features, n_components=2, **iso_kwargs)
```

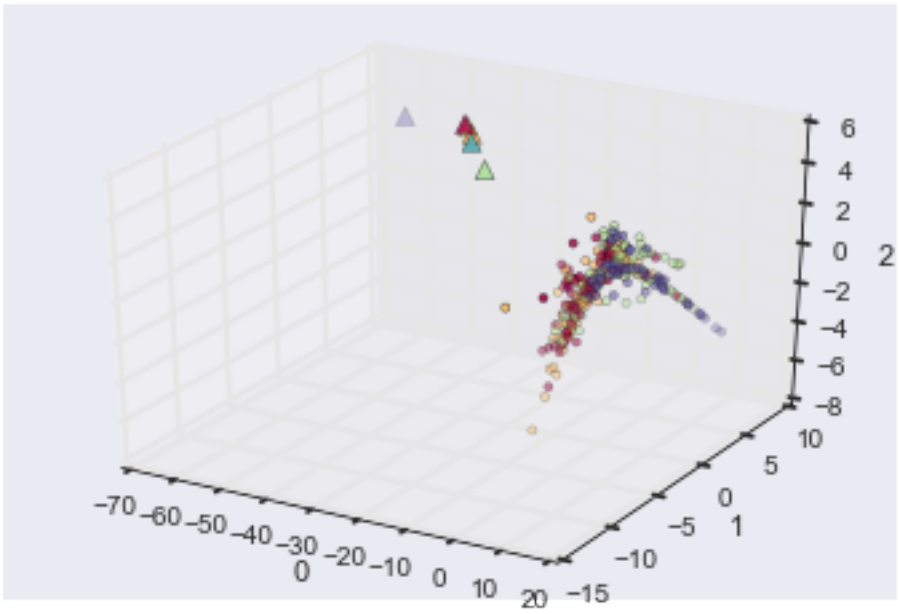
```
In [18]: mia.plotting.plot_mapping_2d(iso_mapping_2d, hologic_texture_features.index,
                                     phantom_texture_features_subset.index, labels)
         plt.savefig('figures/mappings/texture_iso_mapping_2d.png', dpi=300)
```



```
In [19]: iso_mapping_3d = mia.analysis.isomap(selected_features, n_components=3, **iso_kwargs)
```

```
In [81]: mia.plotting.plot_mapping_3d(iso_mapping_3d, hologic_texture_features.index,
                                      phantom_texture_features_subset.index, labels)
```

```
Out[81]: <matplotlib.axes._subplots.Axes3DSubplot at 0x118f53690>
```




```
<matplotlib.figure.Figure at 0x118f3ec10>
```

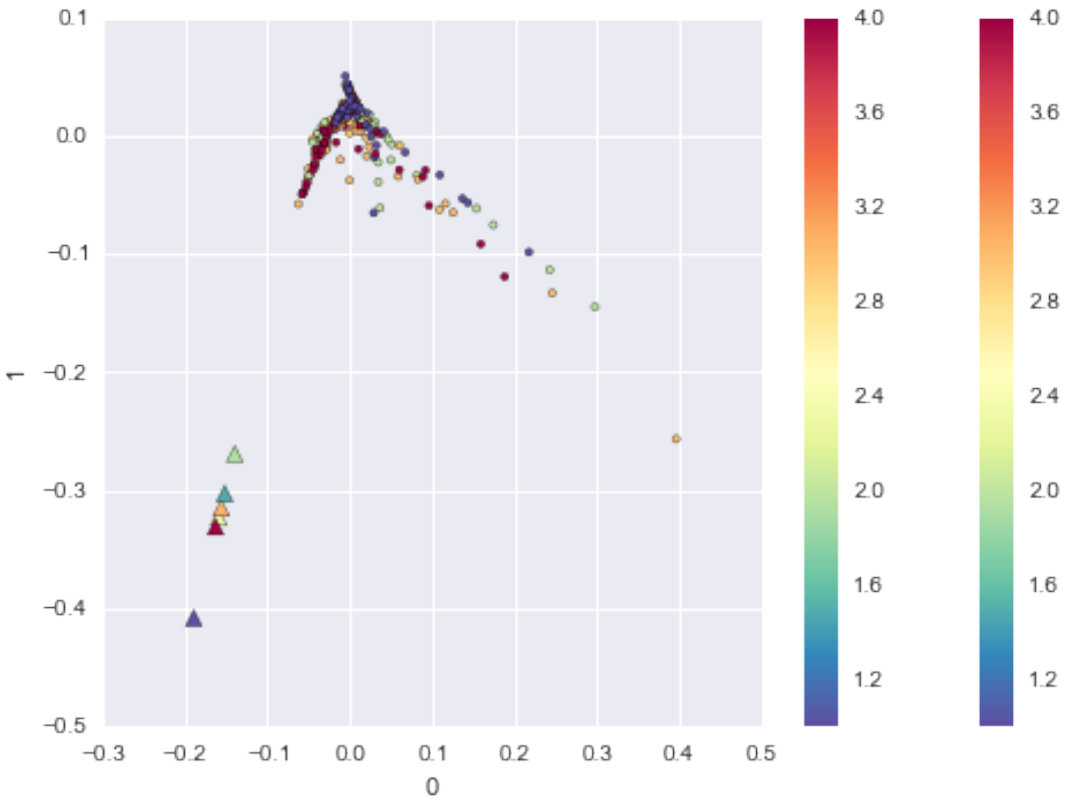
3.12.3 Locally Linear Embedding

Running locally linear embedding to obtain 2d mapping

```
In [21]: lle_kwargs = {
             'n_neighbors': 10,
         }
```

```
In [22]: lle_mapping_2d = mia.analysis.lle(selected_features, n_components=2, **lle_kwargs)
```

```
In [23]: mia.plotting.plot_mapping_2d(lle_mapping_2d, hologic_texture_features.index,
                                      phantom_texture_features_subset.index, labels)
         plt.savefig('figures/mappings/texture_lle_mapping_2d.png', dpi=300)
```




```
In [24]: lle_mapping_3d = mia.analysis.lle(selected_features, n_components=3, **lle_kwargs)
```

```
In [44]: mia.plotting.plot_mapping_3d(lle_mapping_3d, hologic_texture_features.index,
                                     phantom_texture_features_subset.index, labels)
```

```
Out[44]: <matplotlib.axes._subplots.Axes3DSubplot at 0x1105098d0>
```

3.12.4 Quality Assessment of Dimensionality Reduction

Assess the quality of the DR against measurements from the co-ranking matrices. First create co-ranking matrices for each of the dimensionality reduction mappings

```
In [26]: max_k = 10
```

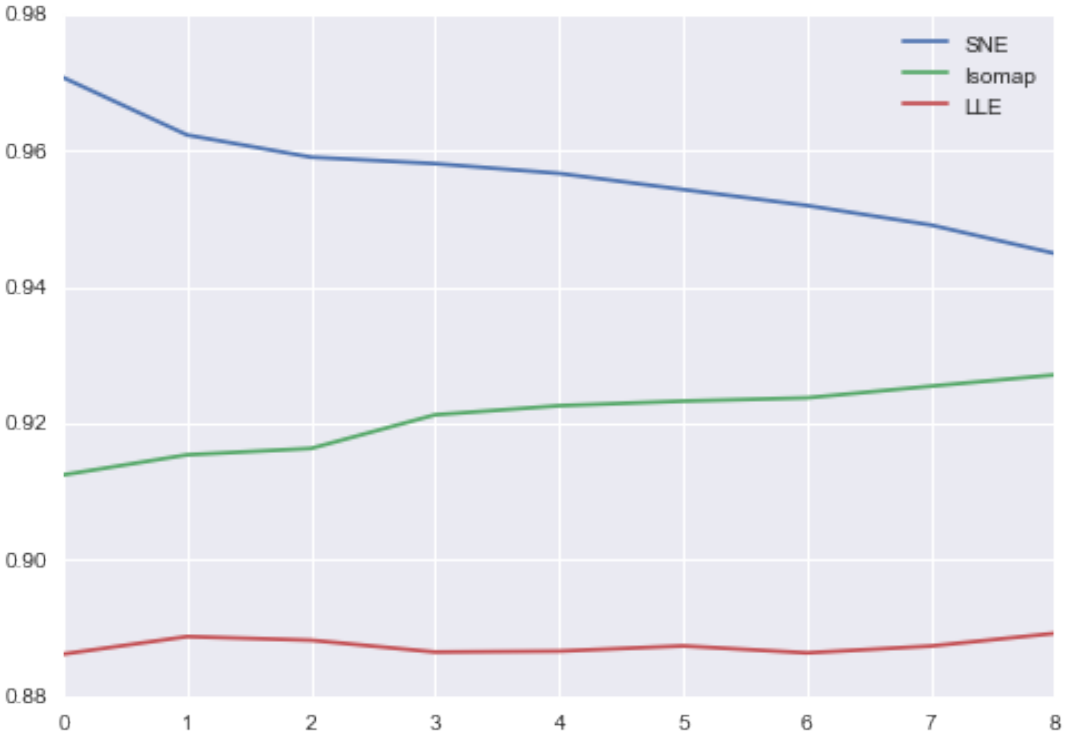
```
In [27]: SNE_mapping_2d_cm = mia.coranking.coranking_matrix(selected_features,
                                                           SNE_mapping_2d)
         iso_mapping_2d_cm = mia.coranking.coranking_matrix(selected_features,
                                                           iso_mapping_2d)
         lle_mapping_2d_cm = mia.coranking.coranking_matrix(selected_features,
                                                           lle_mapping_2d)

         SNE_mapping_3d_cm = mia.coranking.coranking_matrix(selected_features,
                                                           SNE_mapping_3d)
         iso_mapping_3d_cm = mia.coranking.coranking_matrix(selected_features,
                                                           iso_mapping_3d)
         lle_mapping_3d_cm = mia.coranking.coranking_matrix(selected_features,
                                                           lle_mapping_3d)
```

3.12.4.1 2D Mappings

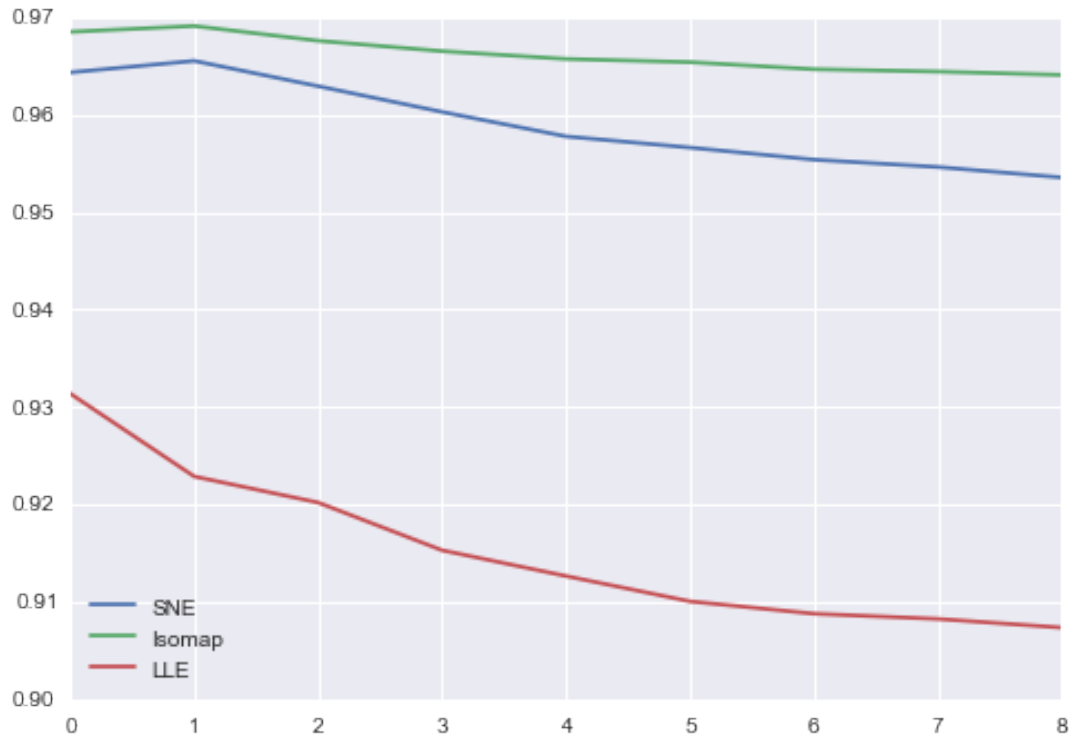
```
In [28]: SNE_trustworthiness_2d = [mia.coranking.trustworthiness(SNE_mapping_2d_cm, k)
                                   for k in range(1, max_k)]
         iso_trustworthiness_2d = [mia.coranking.trustworthiness(iso_mapping_2d_cm, k)
                                   for k in range(1, max_k)]
         lle_trustworthiness_2d = [mia.coranking.trustworthiness(lle_mapping_2d_cm, k)
                                   for k in range(1, max_k)]
```

```
In [29]: trustworthiness_df = pd.DataFrame([SNE_trustworthiness_2d,
                                            iso_trustworthiness_2d,
                                            lle_trustworthiness_2d],
                                            index=['SNE', 'Isomap', 'LLE']).T
         trustworthiness_df.plot()
         plt.savefig('figures/quality_measures/texture_trustworthiness_2d.png', dpi=300)
```



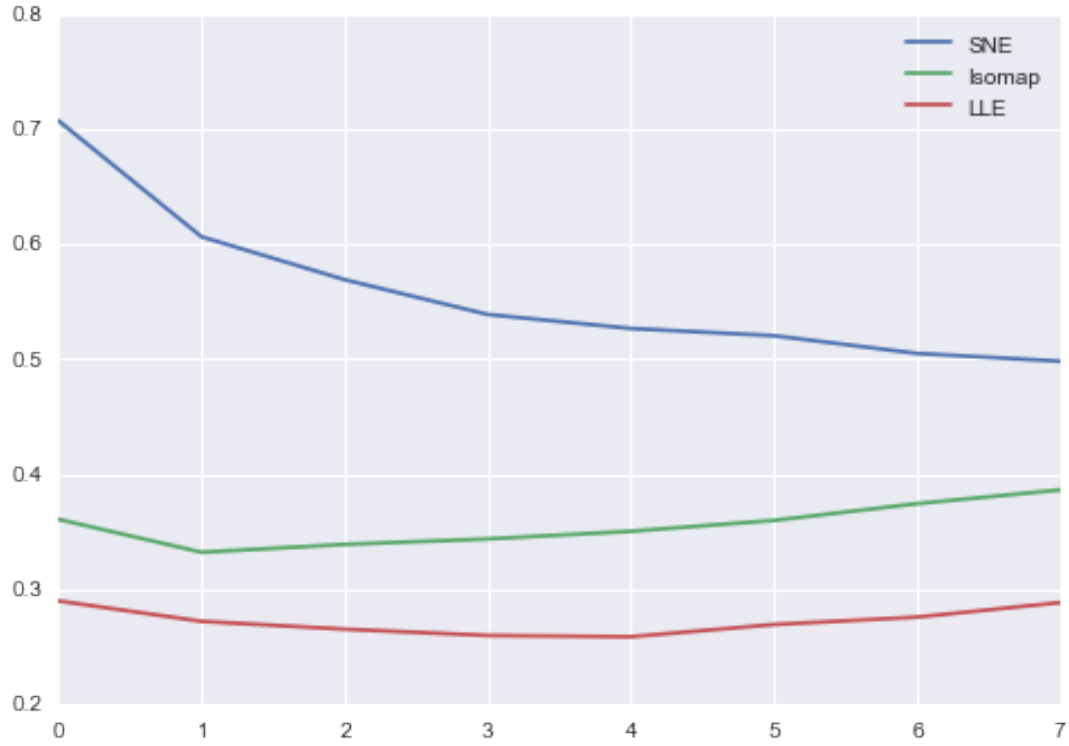

```
In [30]: SNE_continuity_2d = [mia.coranking.continuity(SNE_mapping_2d_cm, k)
                              for k in range(1, max_k)]
         iso_continuity_2d = [mia.coranking.continuity(iso_mapping_2d_cm, k)
                              for k in range(1, max_k)]
         lle_continuity_2d = [mia.coranking.continuity(lle_mapping_2d_cm, k)
                              for k in range(1, max_k)]
```

```
In [31]: continuity_df = pd.DataFrame([SNE_continuity_2d,
                                        iso_continuity_2d,
                                        lle_continuity_2d],
                                        index=['SNE', 'Isomap', 'LLE']).T
         continuity_df.plot()
         plt.savefig('figures/quality_measures/texture_continuity_2d.png', dpi=300)
```




```
In [32]: SNE_lcmc_2d = [mia.coranking.LCMC(SNE_mapping_2d_cm, k)
                        for k in range(2, max_k)]
         iso_lcmc_2d = [mia.coranking.LCMC(iso_mapping_2d_cm, k)
                        for k in range(2, max_k)]
         lle_lcmc_2d = [mia.coranking.LCMC(lle_mapping_2d_cm, k)
                        for k in range(2, max_k)]
```

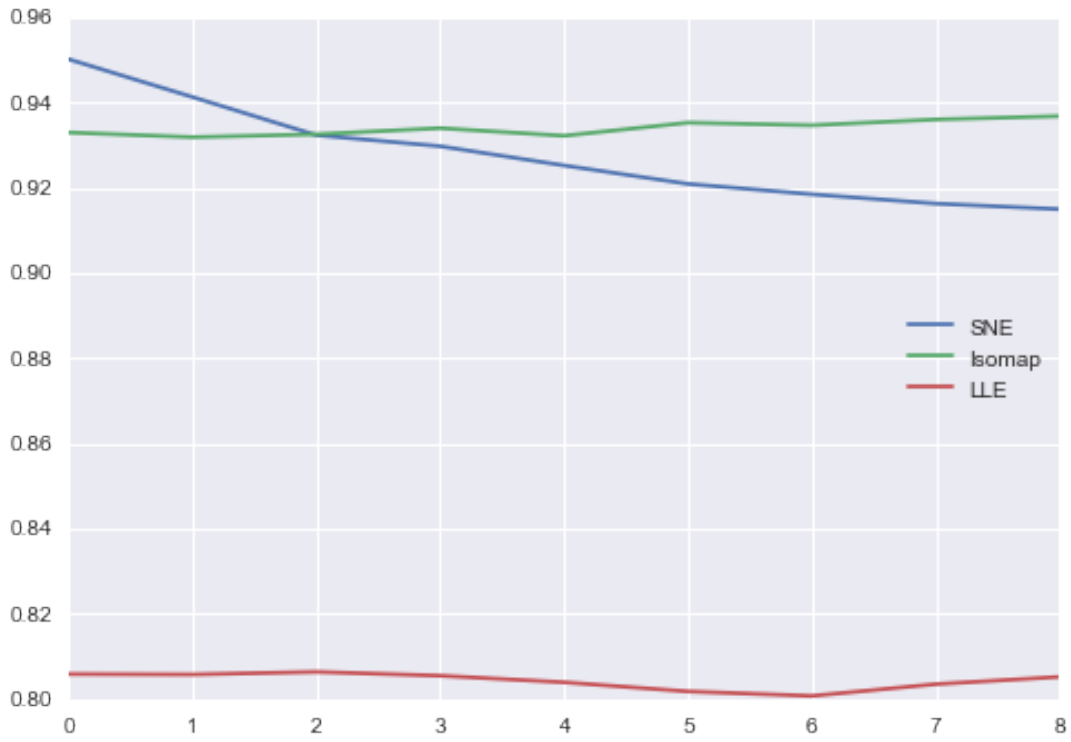
```
In [33]: lcmc_df = pd.DataFrame([SNE_lcmc_2d,
                                 iso_lcmc_2d,
                                 lle_lcmc_2d],
                                 index=['SNE', 'Isomap', 'LLE']).T
         lcmc_df.plot()
         plt.savefig('figures/quality_measures/texture_lcmc_2d.png', dpi=300)
```



3.12.4.2 3D Mappings

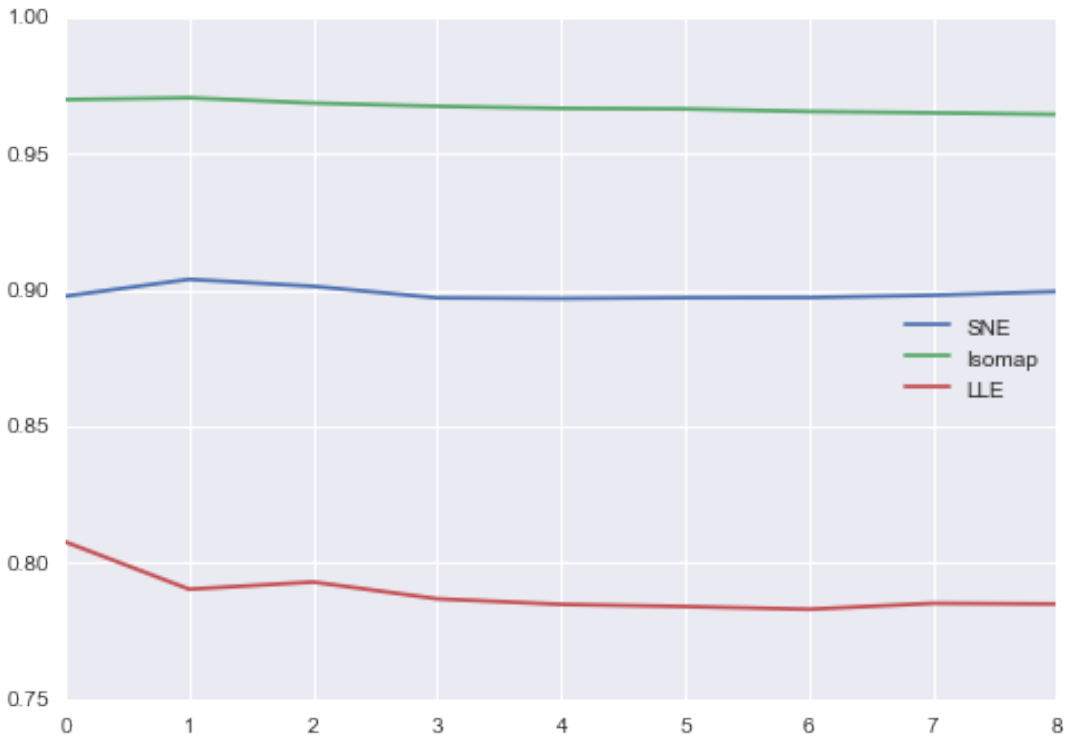
```
In [34]: SNE_trustworthiness_3d = [mia.coranking.trustworthiness(SNE_mapping_3d_cm, k)
                                   for k in range(1, max_k)]
         iso_trustworthiness_3d = [mia.coranking.trustworthiness(iso_mapping_3d_cm, k)
                                   for k in range(1, max_k)]
         lle_trustworthiness_3d = [mia.coranking.trustworthiness(lle_mapping_3d_cm, k)
                                   for k in range(1, max_k)]
```

```
In [35]: trustworthiness3d_df = pd.DataFrame([SNE_trustworthiness_3d,
                                              iso_trustworthiness_3d,
                                              lle_trustworthiness_3d],
                                              index=['SNE', 'Isomap', 'LLE']).T
         trustworthiness3d_df.plot()
         plt.savefig('figures/quality_measures/texture_trustworthiness_3d.png', dpi=300)
```



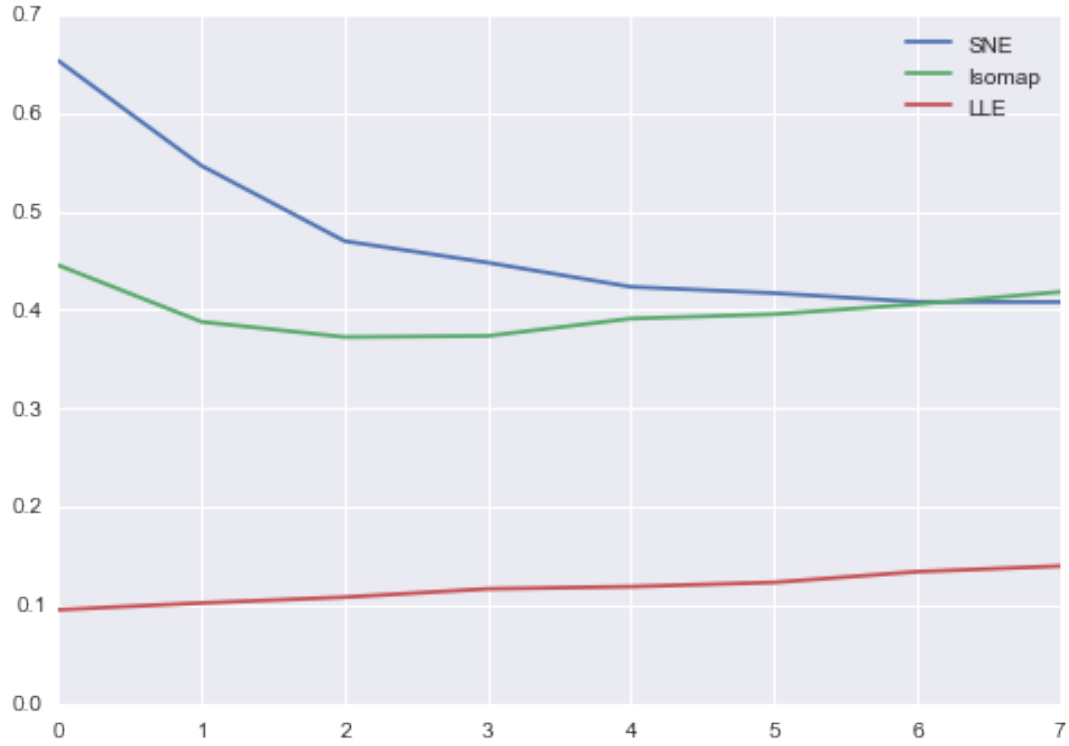

```
In [36]: SNE_continuity_3d = [mia.coranking.continuity(SNE_mapping_3d_cm, k)
                             for k in range(1, max_k)]
         iso_continuity_3d = [mia.coranking.continuity(iso_mapping_3d_cm, k)
                             for k in range(1, max_k)]
         lle_continuity_3d = [mia.coranking.continuity(lle_mapping_3d_cm, k)
                             for k in range(1, max_k)]
```

```
In [37]: continuity3d_df = pd.DataFrame([SNE_continuity_3d,
                                         iso_continuity_3d,
                                         lle_continuity_3d],
                                         index=['SNE', 'Isomap', 'LLE']).T
         continuity3d_df.plot()
         plt.savefig('figures/quality_measures/texture_continuity_3d.png', dpi=300)
```




```
In [38]: SNE_lcmc_3d = [mia.coranking.LCMC(SNE_mapping_3d_cm, k)
                        for k in range(2, max_k)]
         iso_lcmc_3d = [mia.coranking.LCMC(iso_mapping_3d_cm, k)
                        for k in range(2, max_k)]
         lle_lcmc_3d = [mia.coranking.LCMC(lle_mapping_3d_cm, k)
                        for k in range(2, max_k)]
```

```
In [39]: lcmc3d_df = pd.DataFrame([SNE_lcmc_3d,
                                  iso_lcmc_3d,
                                  lle_lcmc_3d],
                                  index=['SNE', 'Isomap', 'LLE']).T
         lcmc3d_df.plot()
         plt.savefig('figures/quality_measures/texture_lcmc_3d.png', dpi=300)
```



Line Analysis

```
In [159]: %matplotlib inline
          import pandas as pd
          import numpy as np
          import scipy.stats as stats
          import matplotlib.pyplot as plt
          import mia
```

```
Warning: Cannot change to a different GUI toolkit: qt. Using osx instead.
```

3.13 Loading and Preprocessing

Loading the hologic and synthetic datasets.

```
In [113]: hologic = pd.DataFrame.from_csv("real-lines.csv")
          phantom = pd.DataFrame.from_csv("phantom-lines.csv")
```

Loading the meta data for the real and synthetic datasets.

```
In [114]: hologic_meta = mia.analysis.create_hologic_meta_data(hologic, "meta_data/real_meta.csv")
          phantom_meta = mia.analysis.create_synthetic_meta_data(phantom,
                                                                 "meta_data/synthetic_meta.csv")
          phantom_meta.index.name = 'img_name'
```

Prepare the BI-RADS/VBD labels for both datasets.

```
In [115]: hologic_labels = hologic_meta.drop_duplicates().BIRADS
          phantom_labels = phantom_meta['VBD.1']

          class_labels = pd.concat([hologic_labels, phantom_labels])
          class_labels.index.name = "img_name"
          labels = mia.analysis.remove_duplicate_index(class_labels)[0]
```

3.14 Creating Features

Create blob features from distribution of blobs

```
In [116]: hologic_line_features = mia.analysis.features_from_lines(hologic)
          phantom_line_features = mia.analysis.features_from_lines(phantom)
```

Take a random subset of the phantom mammograms. This is important so that each case is not over represented.

```
In [117]: syn_feature_meta = mia.analysis.remove_duplicate_index(phantom_meta)
          phantom_line_features['phantom_name'] = syn_feature_meta.phantom_name.tolist()
          phantom_line_features_subset = mia.analysis.create_random_subset(phantom_line_features,
                                                                           'phantom_name')
```

Combine the features from both datasets.

```
In [118]: features = pd.concat([hologic_line_features, phantom_line_features_subset])
          assert features.shape[0] == 366
          features.head()
```

Out[118]:

	count	mean	std	min	25%	50%	\
p214-010-60001-cl.png	72	161.791667	245.194659	1	61.50	94.5	
p214-010-60001-cr.png	57	176.421053	460.921203	1	59.00	77.0	
p214-010-60001-ml.png	72	200.708333	296.524822	1	62.00	95.5	
p214-010-60001-mr.png	74	203.459459	371.688380	1	52.25	81.0	
p214-010-60005-cl.png	114	143.184211	152.914376	1	58.25	89.5	

	75%	max	skew	kurtosis	upper_dist_count
p214-010-60001-cl.png	137.25	1744	4.786025	26.793023	16
p214-010-60001-cr.png	152.00	3494	6.905092	50.202367	13
p214-010-60001-ml.png	210.50	1662	3.484285	13.937538	19
p214-010-60001-mr.png	192.25	2522	4.309621	22.429202	16
p214-010-60005-cl.png	170.75	911	2.667624	9.028130	36

Filter some features, such as the min, to remove noise.

```
In [119]: selected_features = features.drop(['min'], axis=1)
          selected_features.fillna(0, inplace=True)
```

3.15 Compare Real and Synthetic Features

Compare the distributions of features detected from the real mammograms and the phantoms using the Kolmogorov-Smirnov two sample test.

```
In [120]: ks_stats = [list(stats.ks_2samp(hologic_line_features[col],
                                          phantom_line_features[col]))
                                          for col in selected_features.columns]

          ks_test = pd.DataFrame(ks_stats, columns=['KS', 'p-value'], index=selected_features.columns)
          ks_test.to_latex("tables/line_features_ks.tex")
          ks_test
```

Out[120]:

	KS	p-value
count	0.933333	1.338265e-51
mean	0.593056	4.356206e-21
std	0.654167	1.450514e-25
25%	0.143056	1.255126e-01
50%	0.304167	7.344875e-06
75%	0.508333	1.320817e-15
max	0.737500	2.231475e-32
skew	0.480556	5.426886e-14
kurtosis	0.506944	1.598384e-15
upper_dist_count	0.913889	1.724254e-49

3.16 Dimensionality Reduction

3.16.1 t-SNE

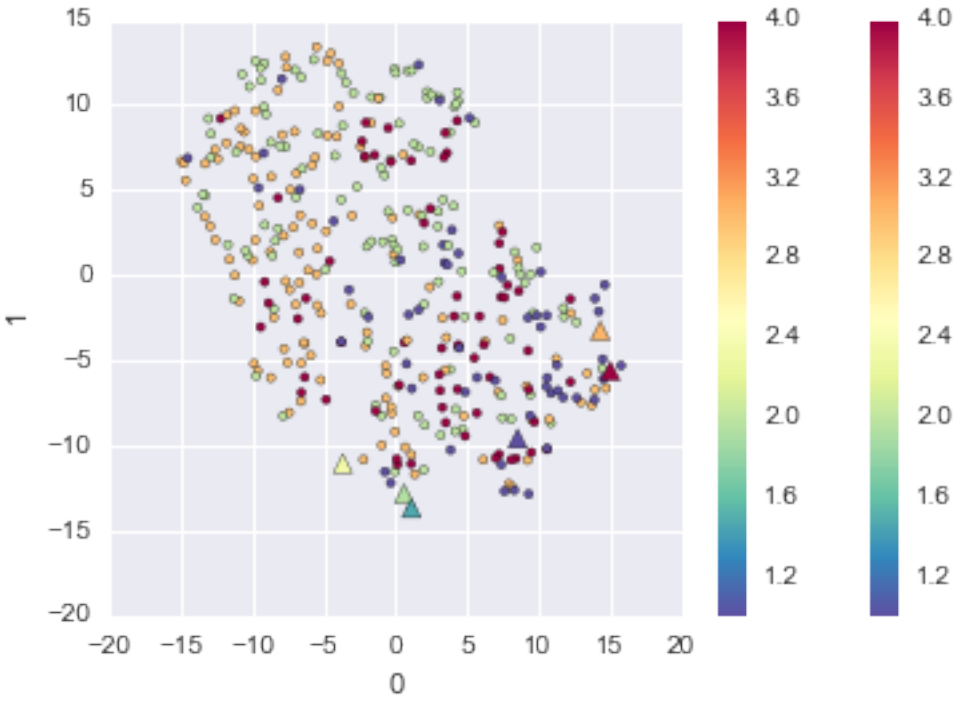
Running t-SNE to obtain a two dimensional representation.

In [121]:
```
kwargs = {
    'learning_rate': 300,
    'perplexity': 30,
    'verbose': 1
}
```

In [122]:
```
SNE_mapping_2d = mia.analysis.tSNE(selected_features, n_components=2, **kwargs)
```

```
[t-SNE] Computing pairwise distances...
[t-SNE] Computed conditional probabilities for sample 366 / 366
[t-SNE] Mean sigma: 0.935505
[t-SNE] Error after 83 iterations with early exaggeration: 14.319531
[t-SNE] Error after 292 iterations: 0.654946
```

In [123]:
```
mia.plotting.plot_mapping_2d(SNE_mapping_2d, hologic_line_features.index,
                             phantom_line_features_subset.index, labels)
plt.savefig('figures/mappings/line_SNE_mapping_2d.png', dpi=300)
```



Running t-SNE to obtain a 3 dimensional mapping

```
In [124]: SNE_mapping_3d = mia.analysis.tSNE(selected_features, n_components=3, **kwargs)
```

```
[t-SNE] Computing pairwise distances...
[t-SNE] Computed conditional probabilities for sample 366 / 366
[t-SNE] Mean sigma: 0.935505
[t-SNE] Error after 92 iterations with early exaggeration: 14.468912
[t-SNE] Error after 333 iterations: 0.664658
```

```
In [160]: mia.plotting.plot_mapping_3d(SNE_mapping_3d, hologic_line_features.index,
                                       phantom_line_features_subset.index, labels)
          # plt.savefig('figures/mappings/line_SNE_mapping_3d.png', dpi=300)
```

```
Out[160]: <matplotlib.axes._subplots.Axes3DSubplot at 0x110d3dd50>
```

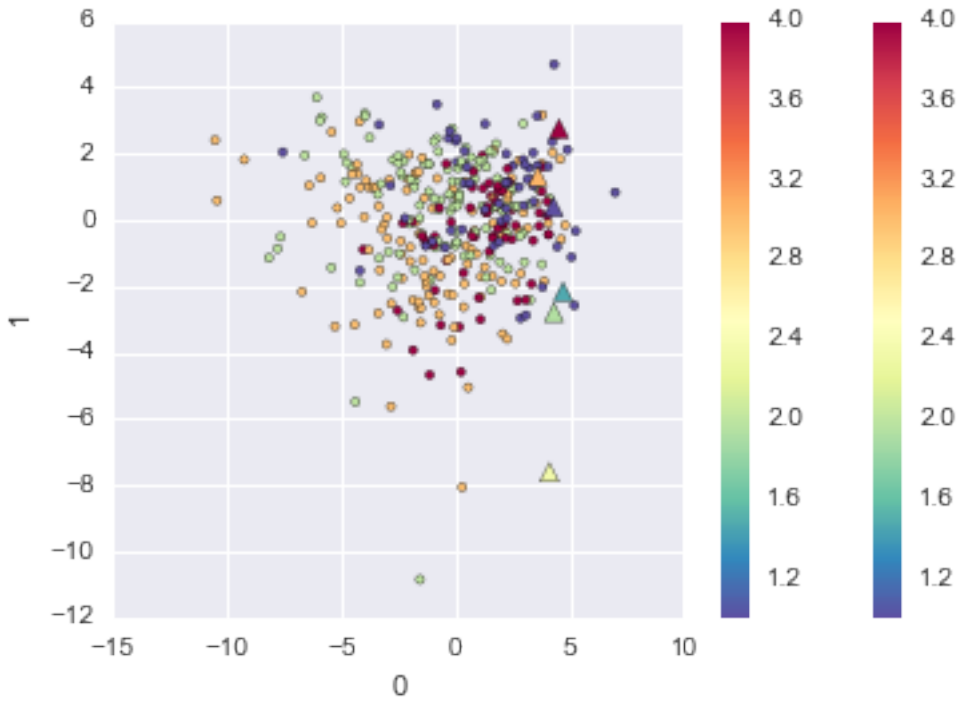
3.16.2 Isomap

Running Isomap to obtain a 2 dimensional mapping

```
In [126]: iso_kwargs = {
              'n_neighbors': 10,
          }
```

```
In [127]: iso_mapping_2d = mia.analysis.isomap(selected_features, n_components=2, **iso_kwargs)
```

```
In [128]: mia.plotting.plot_mapping_2d(iso_mapping_2d, hologic_line_features.index,
                                       phantom_line_features_subset.index, labels)
          plt.savefig('figures/mappings/line_iso_mapping_2d.png', dpi=300)
```



```
In [129]: iso_mapping_3d = mia.analysis.isomap(selected_features, n_components=3, **iso_kwargs)

In [163]: mia.plotting.plot_mapping_3d(iso_mapping_3d, hologic_line_features.index,
                                       phantom_line_features_subset.index, labels)
          # plt.savefig('figures/mappings/line_iso_mapping_3d.png', dpi=300)

Out[163]: <matplotlib.axes._subplots.Axes3DSubplot at 0x113f47b90>
```

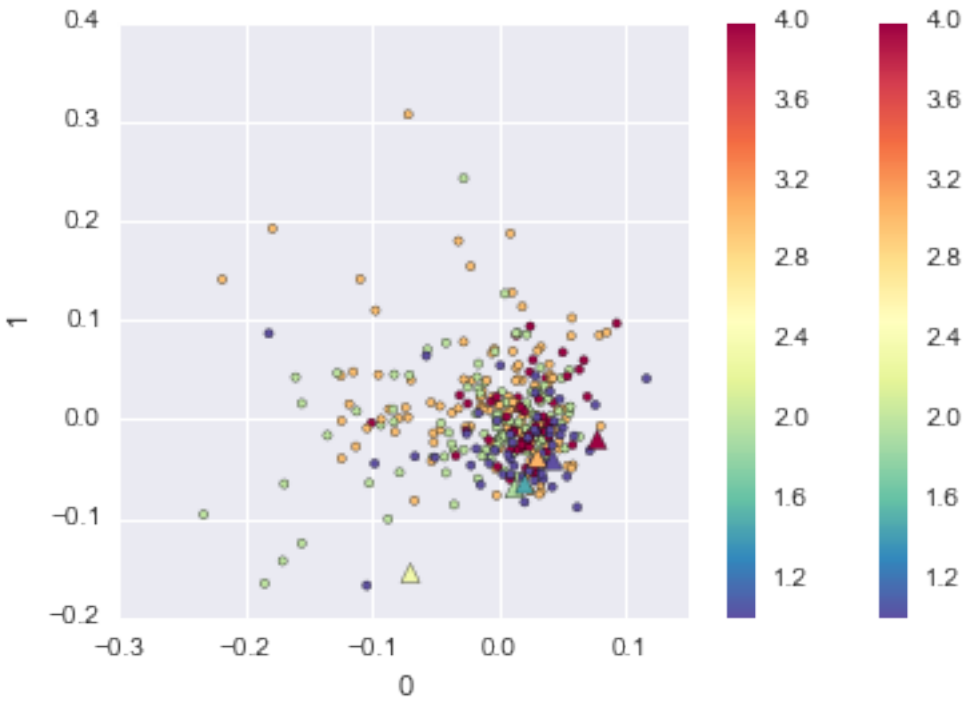
3.16.3 Locally Linear Embedding

Running locally linear embedding to obtain 2d mapping

```
In [131]: lle_kwargs = {
              'n_neighbors': 10,
          }

In [132]: lle_mapping_2d = mia.analysis.lle(selected_features, n_components=2, **lle_kwargs)

In [133]: mia.plotting.plot_mapping_2d(lle_mapping_2d, hologic_line_features.index,
                                       phantom_line_features_subset.index, labels)
          plt.savefig('figures/mappings/line_lle_mapping_2d.png', dpi=300)
```




```
In [134]: lle_mapping_3d = mia.analysis.lle(selected_features, n_components=3, **lle_kwargs)

In [162]: mia.plotting.plot_mapping_3d(lle_mapping_3d, hologic_line_features.index,
                                       phantom_line_features_subset.index, labels)
          # plt.savefig('figures/mappings/line_lle_mapping_3d.png', dpi=300)

Out[162]: <matplotlib.axes._subplots.Axes3DSubplot at 0x113799550>
```

3.16.4 Quality Assessment of Dimensionality Reduction

Assess the quality of the DR against measurements from the co-ranking matrices. First create co-ranking matrices for each of the dimensionality reduction mappings

```
In [136]: max_k = 50
```

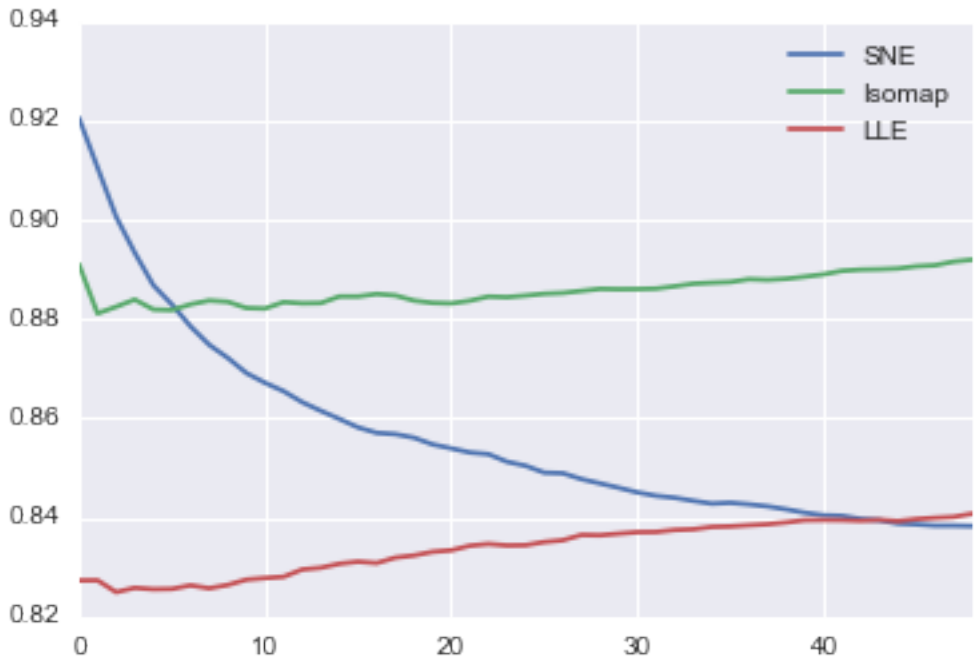
```
In [137]: SNE_mapping_2d_cm = mia.coranking.coranking_matrix(selected_features,
                                                            SNE_mapping_2d)
          iso_mapping_2d_cm = mia.coranking.coranking_matrix(selected_features,
                                                            iso_mapping_2d)
          lle_mapping_2d_cm = mia.coranking.coranking_matrix(selected_features,
                                                            lle_mapping_2d)

          SNE_mapping_3d_cm = mia.coranking.coranking_matrix(selected_features,
                                                            SNE_mapping_3d)
          iso_mapping_3d_cm = mia.coranking.coranking_matrix(selected_features,
                                                            iso_mapping_3d)
          lle_mapping_3d_cm = mia.coranking.coranking_matrix(selected_features,
                                                            lle_mapping_3d)
```

3.16.4.1 2D Mappings

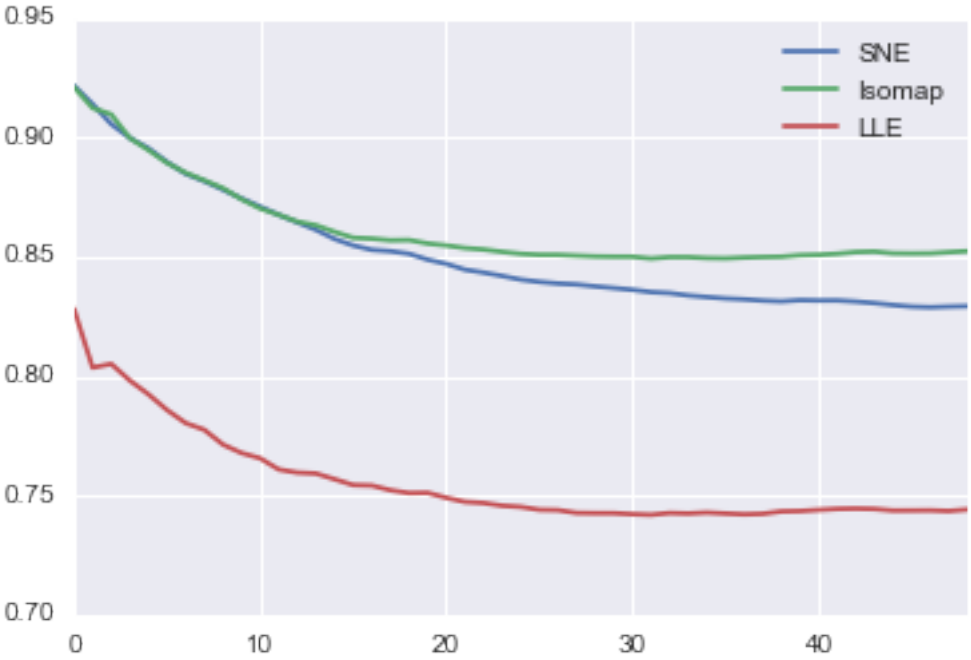
```
In [138]: SNE_trustworthiness_2d = [mia.coranking.trustworthiness(SNE_mapping_2d_cm, k)
                                    for k in range(1, max_k)]
          iso_trustworthiness_2d = [mia.coranking.trustworthiness(iso_mapping_2d_cm, k)
                                    for k in range(1, max_k)]
          lle_trustworthiness_2d = [mia.coranking.trustworthiness(lle_mapping_2d_cm, k)
                                    for k in range(1, max_k)]
```

```
In [139]: trustworthiness_df = pd.DataFrame([SNE_trustworthiness_2d,
                                             iso_trustworthiness_2d,
                                             lle_trustworthiness_2d],
                                             index=['SNE', 'Isomap', 'LLE']).T
          trustworthiness_df.plot()
          plt.savefig('figures/quality_measures/line_trustworthiness_2d.png', dpi=300)
```



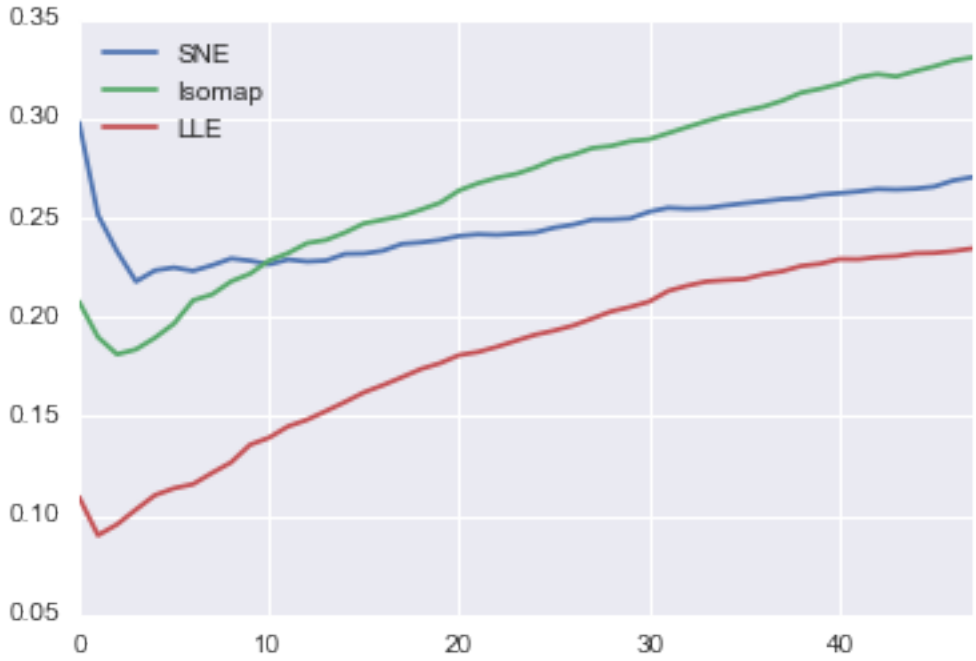
```
In [140]: SNE_continuity_2d = [mia.coranking.continuity(SNE_mapping_2d_cm, k)
                               for k in range(1, max_k)]
          iso_continuity_2d = [mia.coranking.continuity(iso_mapping_2d_cm, k)
                               for k in range(1, max_k)]
          lle_continuity_2d = [mia.coranking.continuity(lle_mapping_2d_cm, k)
                               for k in range(1, max_k)]
```

```
In [141]: continuity_df = pd.DataFrame([SNE_continuity_2d,
                                        iso_continuity_2d,
                                        lle_continuity_2d],
                                        index=['SNE', 'Isomap', 'LLE']).T
          continuity_df.plot()
          plt.savefig('figures/quality_measures/line_continuity_2d.png', dpi=300)
```




```
In [142]: SNE_lcmc_2d = [mia.coranking.LCMC(SNE_mapping_2d_cm, k)
                         for k in range(2, max_k)]
          iso_lcmc_2d = [mia.coranking.LCMC(iso_mapping_2d_cm, k)
                         for k in range(2, max_k)]
          lle_lcmc_2d = [mia.coranking.LCMC(lle_mapping_2d_cm, k)
                         for k in range(2, max_k)]
```

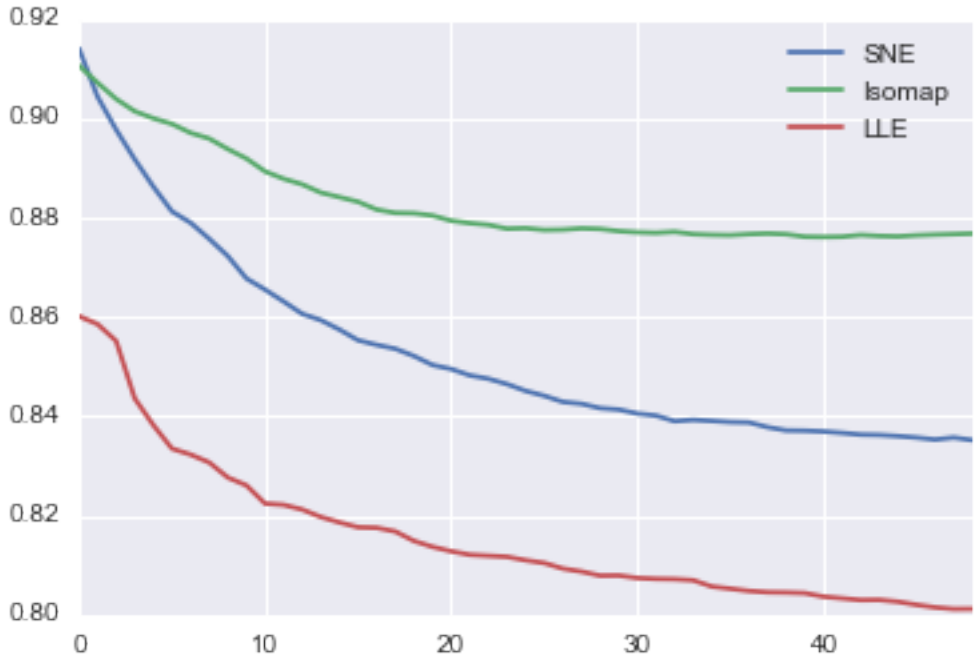
```
In [143]: lcmc_df = pd.DataFrame([SNE_lcmc_2d,
                                  iso_lcmc_2d,
                                  lle_lcmc_2d],
                                  index=['SNE', 'Isomap', 'LLE']).T
          lcmc_df.plot()
          plt.savefig('figures/quality_measures/line_lcmc_2d.png', dpi=300)
```



3.16.4.2 3D Mappings

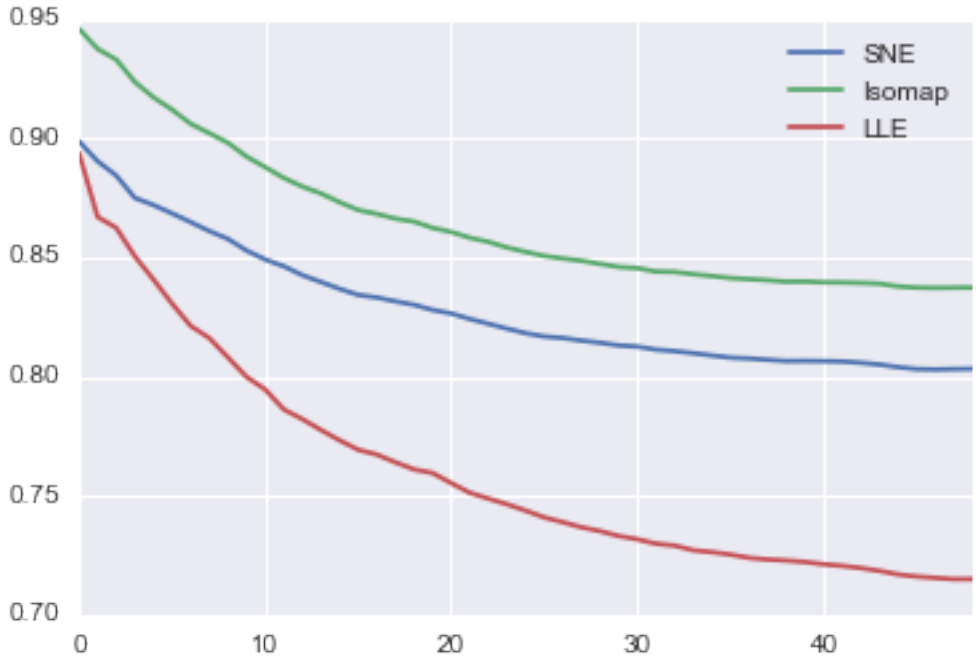
```
In [144]: SNE_trustworthiness_3d = [mia.coranking.trustworthiness(SNE_mapping_3d_cm, k)
                                    for k in range(1, max_k)]
          iso_trustworthiness_3d = [mia.coranking.trustworthiness(iso_mapping_3d_cm, k)
                                    for k in range(1, max_k)]
          lle_trustworthiness_3d = [mia.coranking.trustworthiness(lle_mapping_3d_cm, k)
                                    for k in range(1, max_k)]


In [145]: trustworthiness3d_df = pd.DataFrame([SNE_trustworthiness_3d,
                                                iso_trustworthiness_3d,
                                                lle_trustworthiness_3d],
                                                index=['SNE', 'Isomap', 'LLE']).T
          trustworthiness3d_df.plot()
          plt.savefig('figures/quality_measures/line_trustworthiness_3d.png', dpi=300)
```



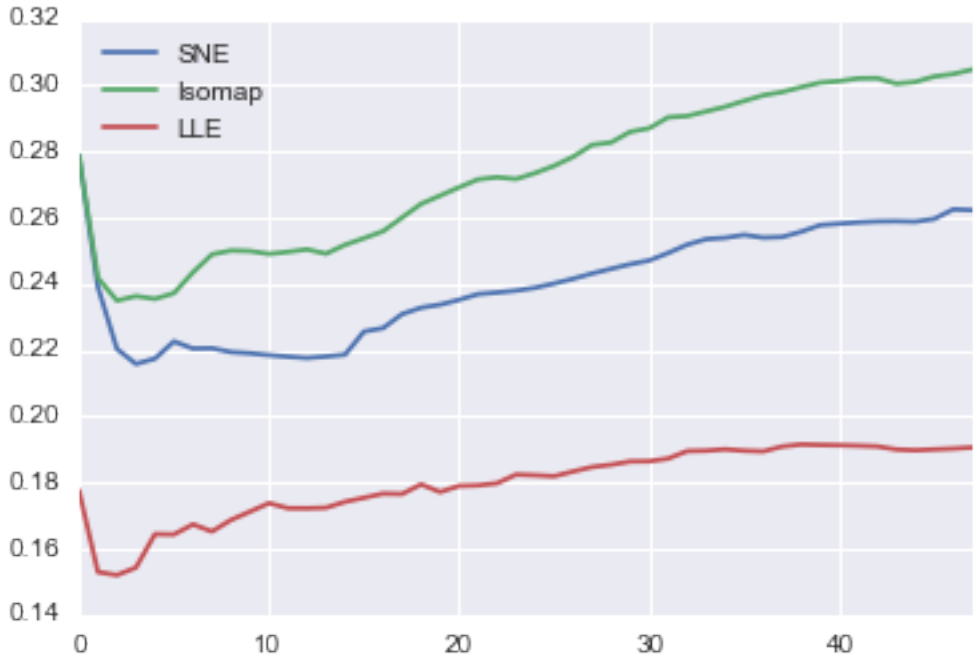

```
In [146]: SNE_continuity_3d = [mia.coranking.continuity(SNE_mapping_3d_cm, k)
                               for k in range(1, max_k)]
          iso_continuity_3d = [mia.coranking.continuity(iso_mapping_3d_cm, k)
                               for k in range(1, max_k)]
          lle_continuity_3d = [mia.coranking.continuity(lle_mapping_3d_cm, k)
                               for k in range(1, max_k)]
```

```
In [147]: continuity3d_df = pd.DataFrame([SNE_continuity_3d,
                                          iso_continuity_3d,
                                          lle_continuity_3d],
                                          index=['SNE', 'Isomap', 'LLE']).T
          continuity3d_df.plot()
          plt.savefig('figures/quality_measures/line_continuity_3d.png', dpi=300)
```




```
In [148]: SNE_lcmc_3d = [mia.coranking.LCMC(SNE_mapping_3d_cm, k)
                         for k in range(2, max_k)]
          iso_lcmc_3d = [mia.coranking.LCMC(iso_mapping_3d_cm, k)
                         for k in range(2, max_k)]
          lle_lcmc_3d = [mia.coranking.LCMC(lle_mapping_3d_cm, k)
                         for k in range(2, max_k)]
```

```
In [149]: lcmc3d_df = pd.DataFrame([SNE_lcmc_3d,
                                   iso_lcmc_3d,
                                   lle_lcmc_3d],
                                   index=['SNE', 'Isomap', 'LLE']).T
          lcmc3d_df.plot()
          plt.savefig('figures/quality_measures/line_lcmc_3d.png', dpi=300)
```



Line Intensity Analysis

```
In [48]: %matplotlib inline
         import pandas as pd
         import numpy as np
         import scipy.stats as stats
         import matplotlib.pyplot as plt
         import mia
```

3.17 Loading and Preprocessing

Loading the hologic and synthetic datasets.

```
In [2]: hologic = pd.DataFrame.from_csv("real_intensity_lines.csv")
        hologic.drop(hologic.columns[:2], axis=1, inplace=True)
        hologic.drop('breast_area', axis=1, inplace=True)

        phantom = pd.DataFrame.from_csv("synthetic_intensity_lines.csv")
        phantom.drop(phantom.columns[:2], axis=1, inplace=True)
        phantom.drop('breast_area', axis=1, inplace=True)
```

Loading the meta data for the real and synthetic datasets.

```
In [3]: hologic_meta = mia.analysis.create_hologic_meta_data(hologic, "meta_data/real_meta.csv")
        phantom_meta = mia.analysis.create_synthetic_meta_data(phantom,
                                                               "meta_data/synthetic_meta.csv")
        phantom_meta.index.name = 'img_name'
```

Prepare the BI-RADS/VBD labels for both datasets.

```
In [4]: hologic_labels = hologic_meta.drop_duplicates().BIRADS
        phantom_labels = phantom_meta['VBD.1']

        class_labels = pd.concat([hologic_labels, phantom_labels])
        class_labels.index.name = "img_name"
        labels = mia.analysis.remove_duplicate_index(class_labels)[0]
```

3.18 Creating Features

Create blob features from distribution of blobs

```
In [55]: hologic_intensity_features = hologic[hologic.columns[4:]]
         hologic_intensity_features = hologic_intensity_features.groupby(hologic.index).agg(np.mean)
         phantom_intensity_features = phantom[phantom.columns[4:]]
         phantom_intensity_features = phantom_intensity_features.groupby(phantom.index).agg(np.mean)
```

Take a random subset of the phantom mammograms. This is important so that each case is not over represented.

```
In [6]: syn_feature_meta = mia.analysis.remove_duplicate_index(phantom_meta)
        phantom_intensity_features['phantom_name'] = syn_feature_meta.phantom_name.tolist()
        phantom_intensity_features_subset \
            = mia.analysis.create_random_subset(phantom_intensity_features, 'phantom_name')
```

Combine the features from both datasets.

```
In [7]: features = pd.concat([hologic_intensity_features, phantom_intensity_features_subset])
        assert features.shape[0] == 366
        features.head()
```

```
Out[7]:                              mean       std       min       25%       50%  \
        p214-010-60001-cl.png  0.362584  0.099563  0.176311  0.290825  0.350695
        p214-010-60001-cr.png  0.326219  0.102222  0.153672  0.250758  0.317584
        p214-010-60001-ml.png  0.394061  0.102072  0.194879  0.324951  0.387695
        p214-010-60001-mr.png  0.346520  0.092847  0.181811  0.286208  0.339103
        p214-010-60005-cl.png  0.406239  0.110257  0.187572  0.328258  0.400346

                                   75%       max      skew  kurtosis
        p214-010-60001-cl.png  0.425325  0.651941  0.554375  0.564813
        p214-010-60001-cr.png  0.391641  0.590179  0.510056  0.047178
        p214-010-60001-ml.png  0.456746  0.673763  0.354676  0.266671
        p214-010-60001-mr.png  0.398680  0.605823  0.455182  0.255500
        p214-010-60005-cl.png  0.478228  0.672765  0.219887 -0.055904
```

Filter some features, such as the min, to remove noise.

```
In [8]: selected_features = features.copy()
```

3.19 Compare Real and Synthetic Features

Compare the distributions of features detected from the real mammograms and the phantoms using the Kolmogorov-Smirnov two sample test.

```
In [56]: ks_stats = [list(stats.ks_2samp(hologic_intensity_features[col],
                                         phantom_intensity_features[col]))
                                         for col in selected_features.columns]

         ks_test = pd.DataFrame(ks_stats, columns=['KS', 'p-value'], index=selected_features.columns)
         ks_test.to_latex("tables/line_intensity_features_ks.tex")
         ks_test
```

```
Out[56]:                KS       p-value
         mean      1.000000  3.587622e-59
         std       0.966667  2.546525e-55
         min       1.000000  3.587622e-59
         25%       1.000000  3.587622e-59
         50%       1.000000  3.587622e-59
         75%       1.000000  3.587622e-59
         max       1.000000  3.587622e-59
         skew      1.000000  3.587622e-59
         kurtosis  0.213889  4.106586e-03
```

3.20 Dimensionality Reduction

3.20.1 t-SNE

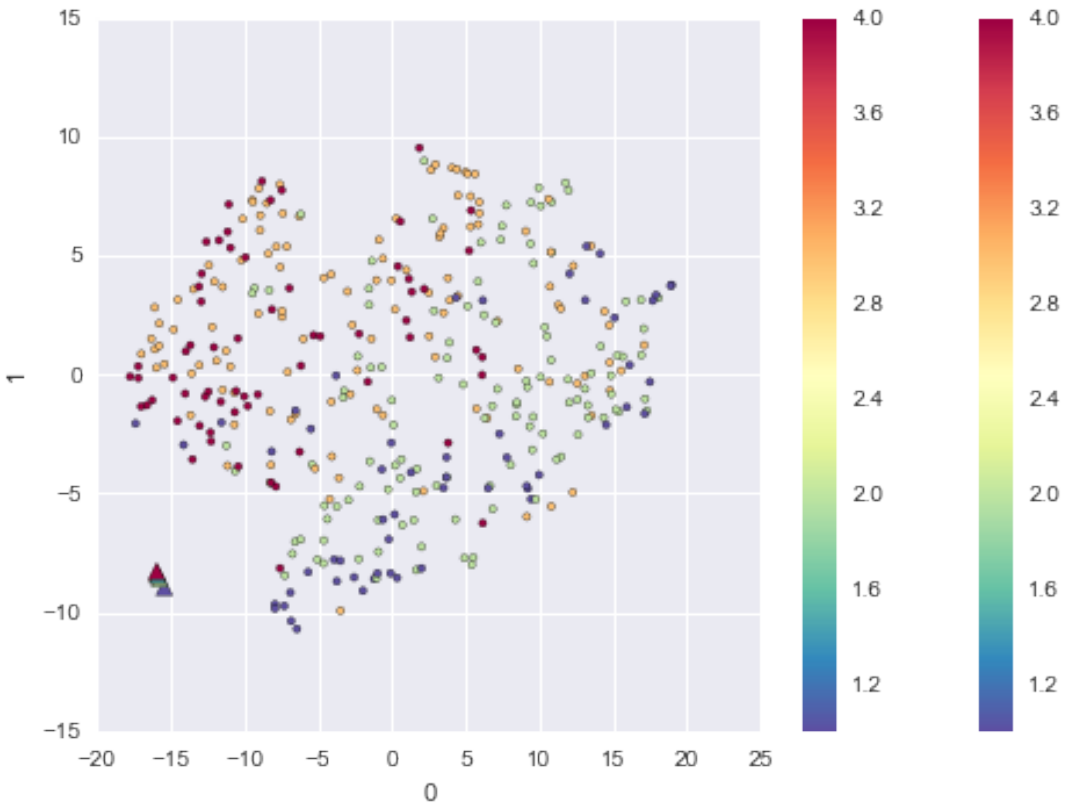
Running t-SNE to obtain a two dimensional representation.

```
In [10]: kwargs = {
             'learning_rate': 300,
             'perplexity': 30,
             'verbose': 1
         }
```

```
In [11]: SNE_mapping_2d = mia.analysis.tSNE(selected_features, n_components=2, **kwargs)
```

```
[t-SNE] Computing pairwise distances...
[t-SNE] Computed conditional probabilities for sample 366 / 366
[t-SNE] Mean sigma: 0.599228
[t-SNE] Error after 83 iterations with early exaggeration: 13.517177
[t-SNE] Error after 280 iterations: 0.549352
```

```
In [12]: mia.plotting.plot_mapping_2d(SNE_mapping_2d, hologic_intensity_features.index,
                                      phantom_intensity_features_subset.index, labels)
         plt.savefig('figures/mappings/line_intensity_SNE_mapping_2d.png', dpi=300)
```



Running t-SNE to obtain a 3 dimensional mapping

```
In [13]: SNE_mapping_3d = mia.analysis.tSNE(selected_features, n_components=3, **kwargs)
```

```
[t-SNE] Computing pairwise distances...
[t-SNE] Computed conditional probabilities for sample 366 / 366
[t-SNE] Mean sigma: 0.599228
[t-SNE] Error after 83 iterations with early exaggeration: 13.324412
[t-SNE] Error after 173 iterations: 0.359292
```

```
In [43]: mia.plotting.plot_mapping_3d(SNE_mapping_3d, hologic_intensity_features.index,
                                     phantom_intensity_features_subset.index, labels)
```

```
Out[43]: <matplotlib.axes._subplots.Axes3DSubplot at 0x101fef190>
```

3.20.2 Isomap

Running Isomap to obtain a 2 dimensional mapping

```
In [15]: iso_kwargs = {
             'n_neighbors': 10,
         }
```

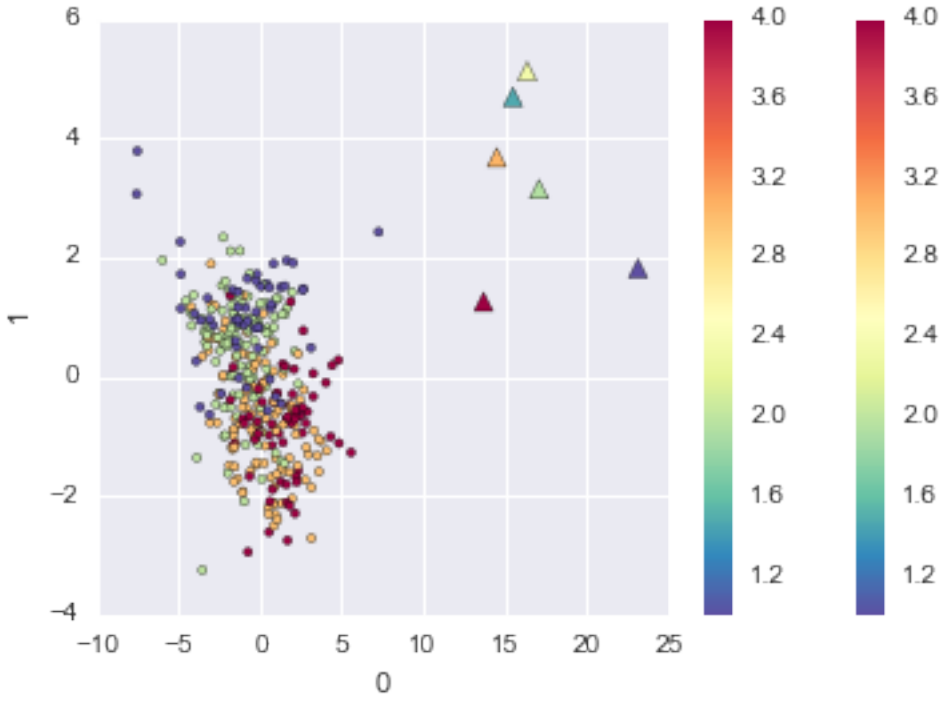
```
In [16]: iso_mapping_2d = mia.analysis.isomap(selected_features, n_components=2, **iso_kwargs)
```

```
In [58]: mia.plotting.plot_mapping_2d(iso_mapping_2d, hologic_intensity_features.index,
                                     phantom_intensity_features_subset.index, labels)
         plt.savefig('figures/mappings/line_intensity_iso_mapping_2d.png', dpi=300)
         features[iso_mapping_2d[1] > 0].describe() - features[iso_mapping_2d[1] <= 0].describe()
```

Out[58]:

	mean	std	min	25%	50%	75%	max
count	-2.000000	-2.000000	-2.000000	-2.000000	-2.000000	-2.000000	-2.000000
mean	-0.003513	-0.021104	0.036702	0.011160	-0.003569	-0.018553	-0.039642
std	0.061442	0.001805	0.071537	0.066832	0.061887	0.056479	0.040878
min	-0.006950	-0.069210	0.024354	0.007557	-0.005128	-0.026422	-0.070078
25%	-0.021067	-0.018684	0.012954	-0.012430	-0.020751	-0.031740	-0.059877
50%	-0.025319	-0.018390	0.019640	-0.008693	-0.024878	-0.037709	-0.054216
75%	-0.014675	-0.021351	0.020545	-0.000035	-0.016425	-0.033499	-0.039873
max	0.459748	-0.029071	0.634287	0.525722	0.458126	0.396945	0.236608

	skew	kurtosis
count	-2.000000	-2.000000
mean	0.120643	0.351995
std	0.069032	0.354517
min	-0.634472	0.076846
25%	0.173825	0.253910
50%	0.155946	0.251659
75%	0.116317	0.313281
max	0.183788	3.838945




```
In [18]: iso_mapping_3d = mia.analysis.isomap(selected_features, n_components=3, **iso_kwargs)
```

```
In [45]: mia.plotting.plot_mapping_3d(iso_mapping_3d, hologic_intensity_features.index,
                                     phantom_intensity_features_subset.index, labels)
```

```
Out[45]: <matplotlib.axes._subplots.Axes3DSubplot at 0x10204ae10>
```

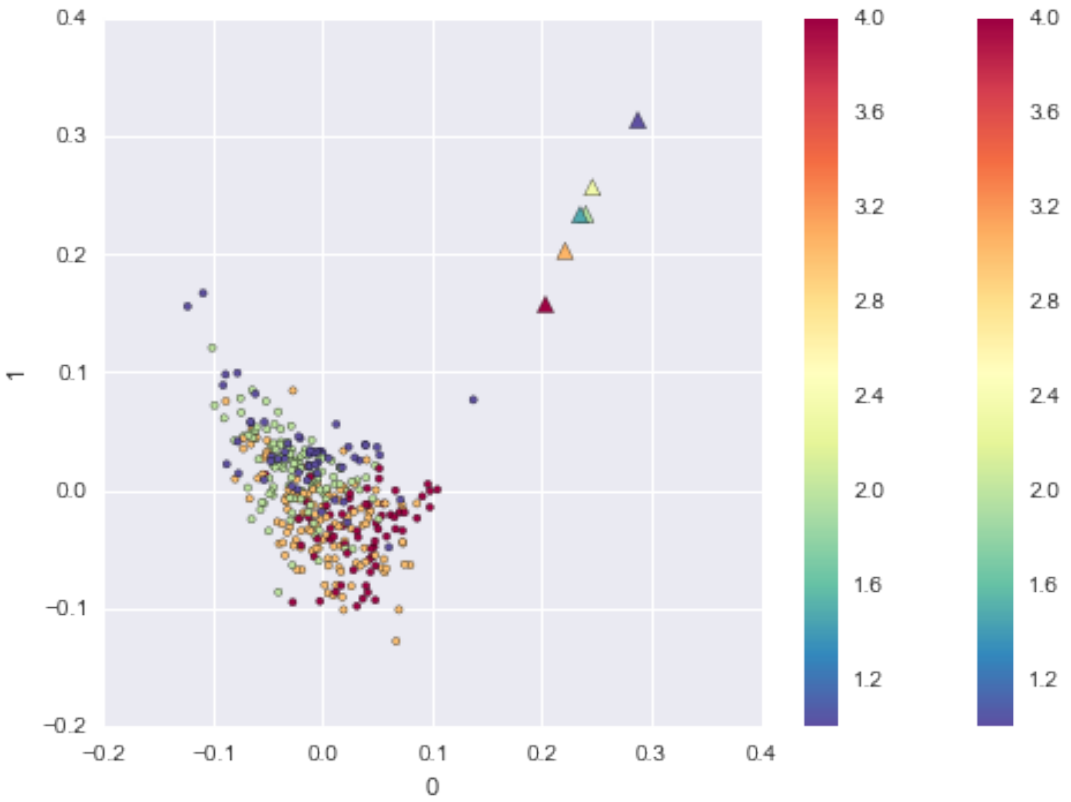
3.20.3 Locally Linear Embedding

Running locally linear embedding to obtain 2d mapping

```
In [20]: lle_kwargs = {
             'n_neighbors': 10,
         }
```

```
In [21]: lle_mapping_2d = mia.analysis.lle(selected_features, n_components=2, **lle_kwargs)
```

```
In [22]: mia.plotting.plot_mapping_2d(lle_mapping_2d, hologic_intensity_features.index,
                                      phantom_intensity_features_subset.index, labels)
         plt.savefig('figures/mappings/line_intensity_lle_mapping_2d.png', dpi=300)
```




```
In [23]: lle_mapping_3d = mia.analysis.lle(selected_features, n_components=3, **lle_kwargs)
```

```
In [46]: mia.plotting.plot_mapping_3d(lle_mapping_3d, hologic_intensity_features.index,
                                      phantom_intensity_features_subset.index, labels)
```

```
Out[46]: <matplotlib.axes._subplots.Axes3DSubplot at 0x108282a50>
```

3.20.4 Quality Assessment of Dimensionality Reduction

Assess the quality of the DR against measurements from the co-ranking matrices. First create co-ranking matrices for each of the dimensionality reduction mappings

```
In [27]: max_k = 50
```

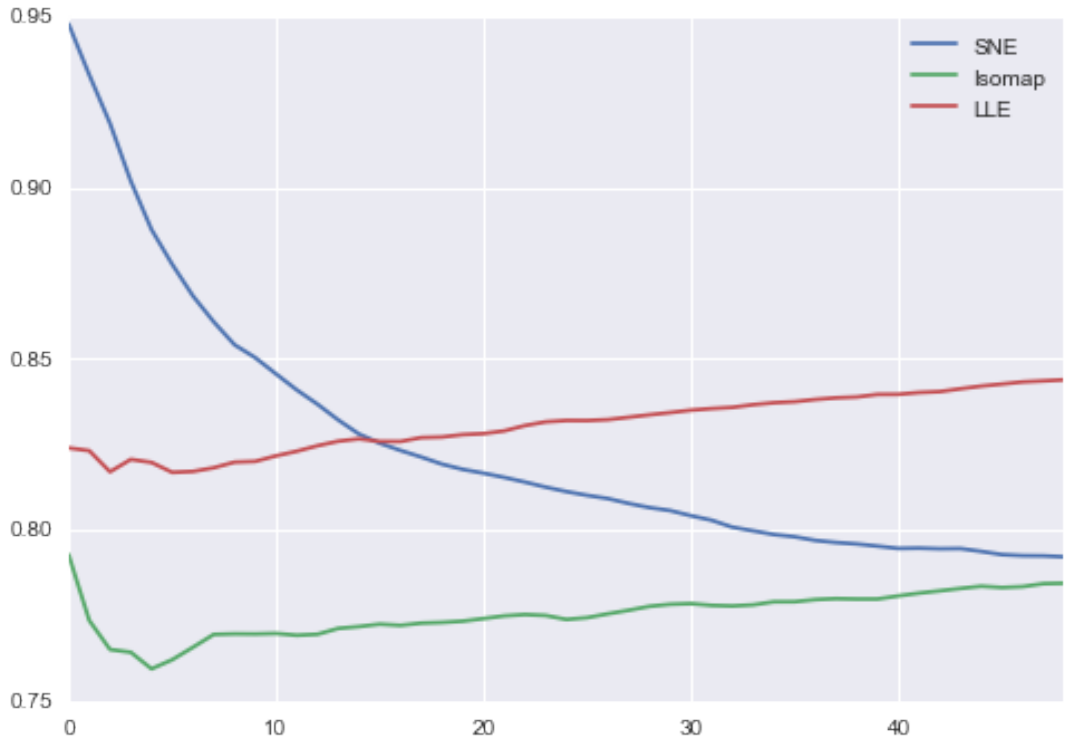
```
In [28]: SNE_mapping_2d_cm = mia.coranking.coranking_matrix(selected_features,
                                                           SNE_mapping_2d)
         iso_mapping_2d_cm = mia.coranking.coranking_matrix(selected_features,
                                                           iso_mapping_2d)
         lle_mapping_2d_cm = mia.coranking.coranking_matrix(selected_features,
                                                           lle_mapping_2d)
```

```
SNE_mapping_3d_cm = mia.coranking.coranking_matrix(selected_features,
                                                   SNE_mapping_3d)
iso_mapping_3d_cm = mia.coranking.coranking_matrix(selected_features,
                                                   iso_mapping_3d)
lle_mapping_3d_cm = mia.coranking.coranking_matrix(selected_features,
                                                   lle_mapping_3d)
```

3.20.4.1 2D Mappings

```
In [29]: SNE_trustworthiness_2d = [mia.coranking.trustworthiness(SNE_mapping_2d_cm, k)
                                   for k in range(1, max_k)]
         iso_trustworthiness_2d = [mia.coranking.trustworthiness(iso_mapping_2d_cm, k)
                                   for k in range(1, max_k)]
         lle_trustworthiness_2d = [mia.coranking.trustworthiness(lle_mapping_2d_cm, k)
                                   for k in range(1, max_k)]
```

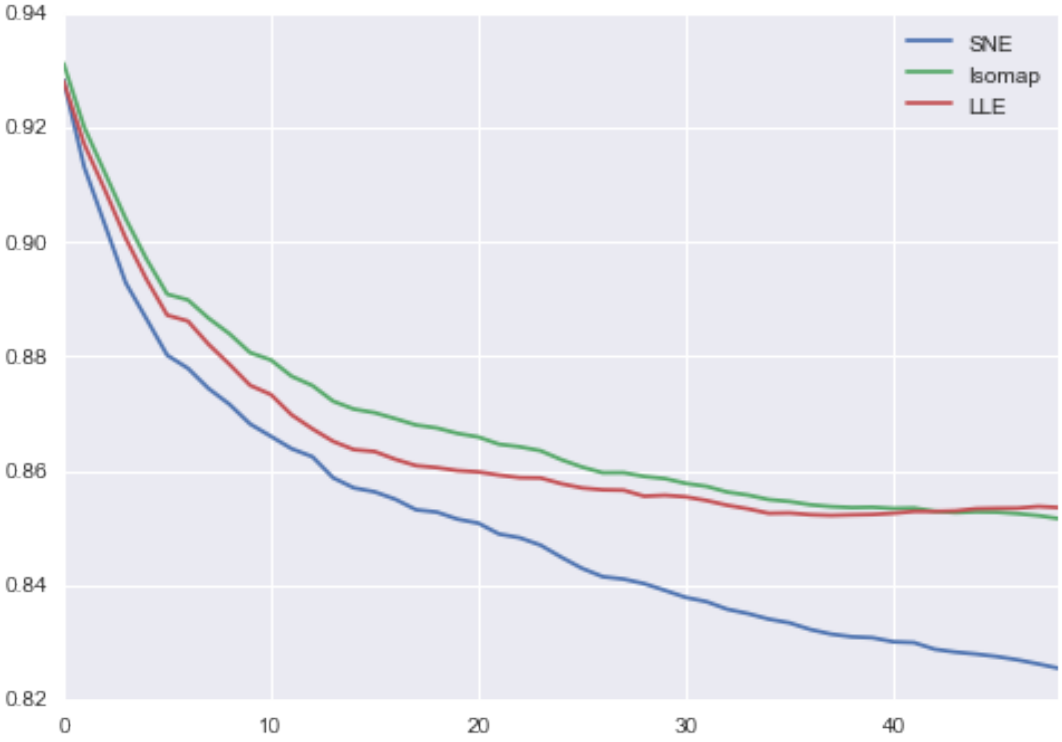
```
In [30]: trustworthiness_df = pd.DataFrame([SNE_trustworthiness_2d,
                                            iso_trustworthiness_2d,
                                            lle_trustworthiness_2d],
                                            index=['SNE', 'Isomap', 'LLE']).T
         trustworthiness_df.plot()
         plt.savefig('figures/quality_measures/line_intensity_trustworthiness_2d.png', dpi=300)
```




```
In [31]: SNE_continuity_2d = [mia.coranking.continuity(SNE_mapping_2d_cm, k)
                              for k in range(1, max_k)]
         iso_continuity_2d = [mia.coranking.continuity(iso_mapping_2d_cm, k)
                              for k in range(1, max_k)]
         lle_continuity_2d = [mia.coranking.continuity(lle_mapping_2d_cm, k)
                              for k in range(1, max_k)]
```

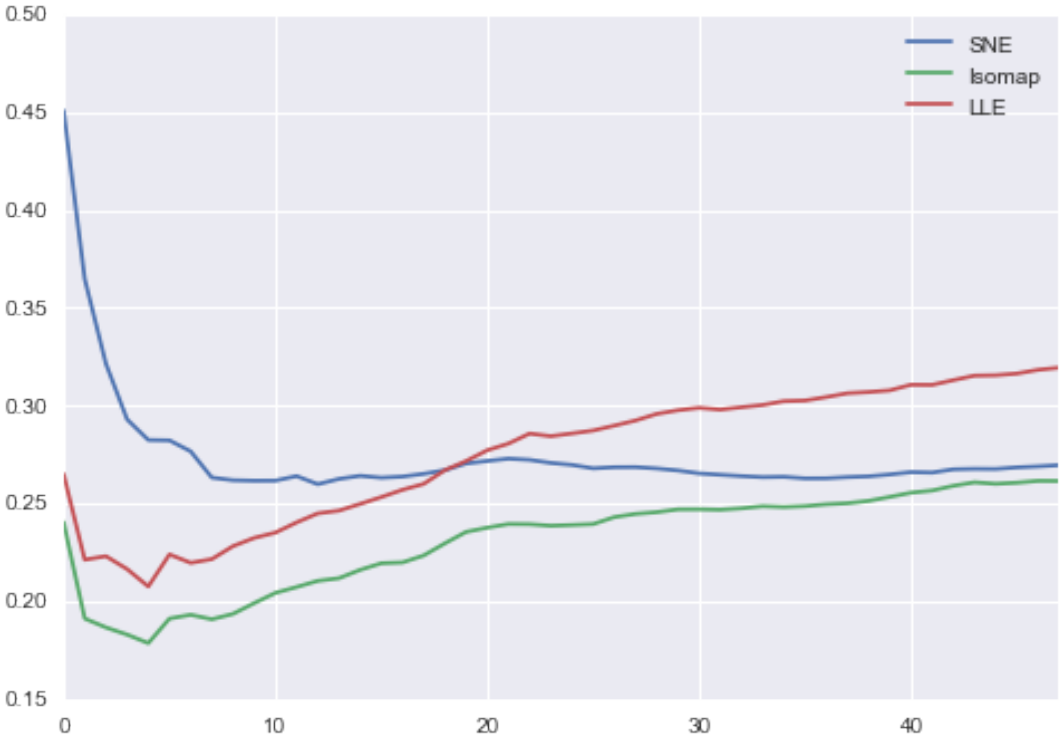
```
In [32]: continuity_df = pd.DataFrame([SNE_continuity_2d,
                                       iso_continuity_2d,
```

```
                                    lle_continuity_2d],
                                    index=['SNE', 'Isomap', 'LLE']).T
        continuity_df.plot()
        plt.savefig('figures/quality_measures/line_intensity_continuity_2d.png', dpi=300)
```




```
In [33]: SNE_lcmc_2d = [mia.coranking.LCMC(SNE_mapping_2d_cm, k)
                        for k in range(2, max_k)]
         iso_lcmc_2d = [mia.coranking.LCMC(iso_mapping_2d_cm, k)
                        for k in range(2, max_k)]
         lle_lcmc_2d = [mia.coranking.LCMC(lle_mapping_2d_cm, k)
                        for k in range(2, max_k)]
```

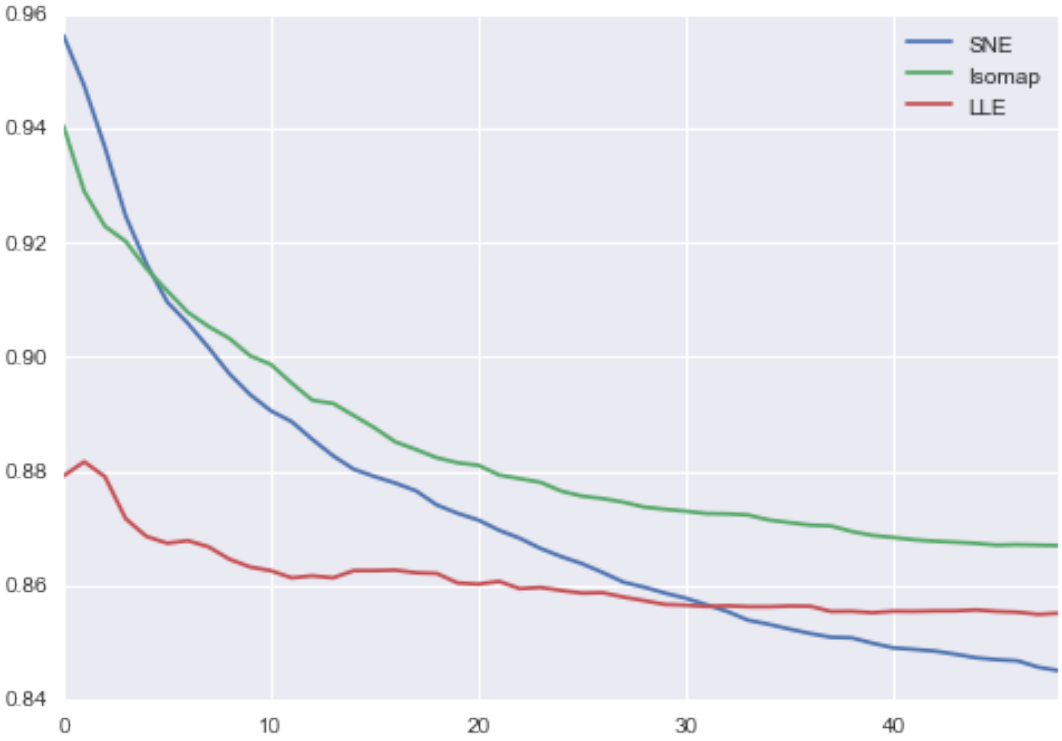
```
In [34]: lcmc_df = pd.DataFrame([SNE_lcmc_2d,
                                 iso_lcmc_2d,
                                 lle_lcmc_2d],
                                 index=['SNE', 'Isomap', 'LLE']).T
         lcmc_df.plot()
         plt.savefig('figures/quality_measures/line_intensity_lcmc_2d.png', dpi=300)
```



3.20.4.2 3D Mappings

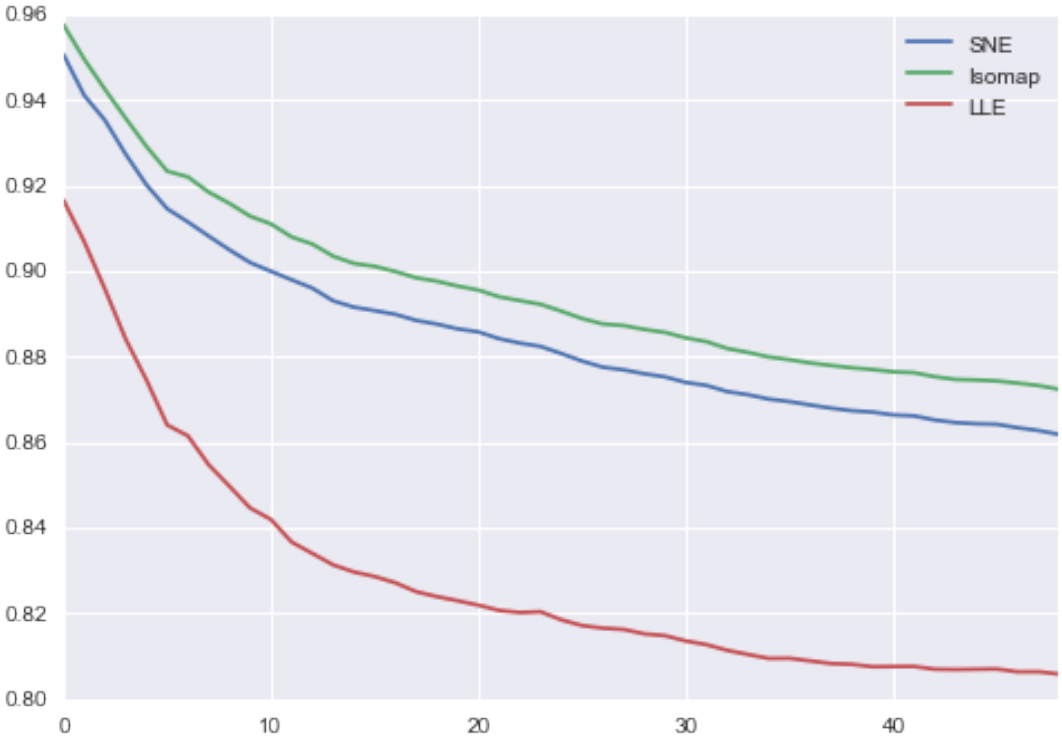
```
In [35]: SNE_trustworthiness_3d = [mia.coranking.trustworthiness(SNE_mapping_3d_cm, k)
                                   for k in range(1, max_k)]
         iso_trustworthiness_3d = [mia.coranking.trustworthiness(iso_mapping_3d_cm, k)
                                   for k in range(1, max_k)]
         lle_trustworthiness_3d = [mia.coranking.trustworthiness(lle_mapping_3d_cm, k)
                                   for k in range(1, max_k)]


In [36]: trustworthiness3d_df = pd.DataFrame([SNE_trustworthiness_3d,
                                              iso_trustworthiness_3d,
                                              lle_trustworthiness_3d],
                                              index=['SNE', 'Isomap', 'LLE']).T
         trustworthiness3d_df.plot()
         plt.savefig('figures/quality_measures/line_intensity_trustworthiness_3d.png', dpi=300)
```



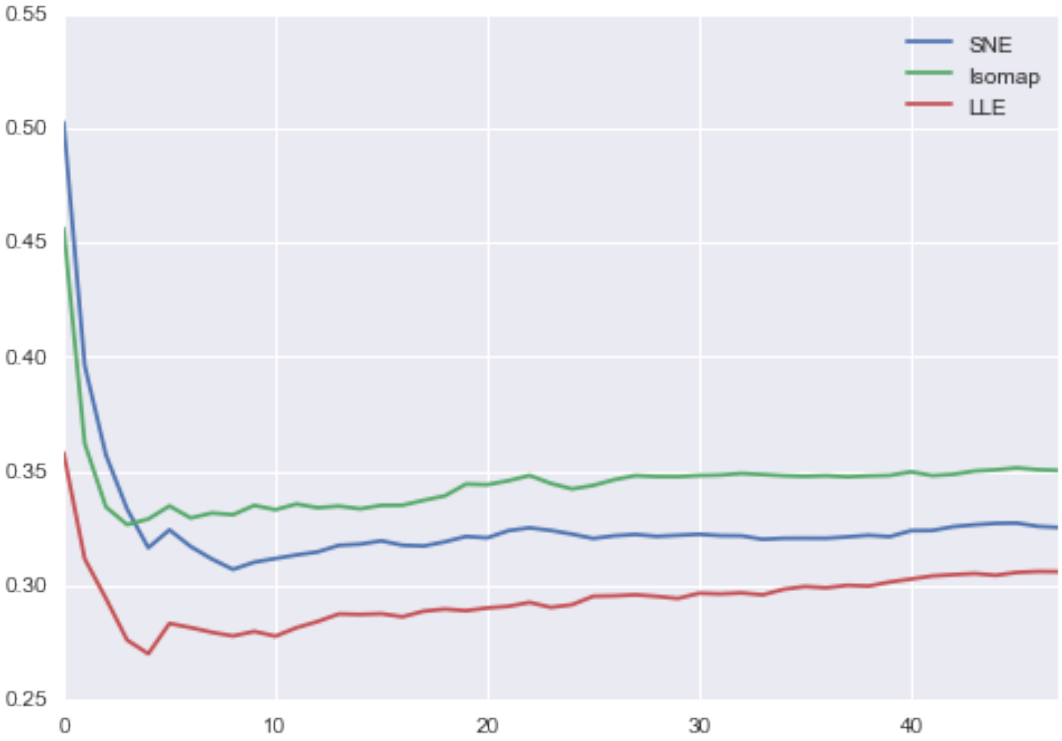

```
In [37]: SNE_continuity_3d = [mia.coranking.continuity(SNE_mapping_3d_cm, k)
                              for k in range(1, max_k)]
         iso_continuity_3d = [mia.coranking.continuity(iso_mapping_3d_cm, k)
                              for k in range(1, max_k)]
         lle_continuity_3d = [mia.coranking.continuity(lle_mapping_3d_cm, k)
                              for k in range(1, max_k)]
```

```
In [38]: continuity3d_df = pd.DataFrame([SNE_continuity_3d,
                                         iso_continuity_3d,
                                         lle_continuity_3d],
                                         index=['SNE', 'Isomap', 'LLE']).T
         continuity3d_df.plot()
         plt.savefig('figures/quality_measures/line_intensity_continuity_3d.png', dpi=300)
```




```
In [39]: SNE_lcmc_3d = [mia.coranking.LCMC(SNE_mapping_3d_cm, k)
                        for k in range(2, max_k)]
         iso_lcmc_3d = [mia.coranking.LCMC(iso_mapping_3d_cm, k)
                        for k in range(2, max_k)]
         lle_lcmc_3d = [mia.coranking.LCMC(lle_mapping_3d_cm, k)
                        for k in range(2, max_k)]
```

```
In [40]: lcmc3d_df = pd.DataFrame([SNE_lcmc_3d,
                                  iso_lcmc_3d,
                                  lle_lcmc_3d],
                                  index=['SNE', 'Isomap', 'LLE']).T
         lcmc3d_df.plot()
         plt.savefig('figures/quality_measures/line_intensity_lcmc_3d.png', dpi=300)
```



Line Texture Analysis

```
In [41]: %matplotlib qt
         import pandas as pd
         import numpy as np
         import scipy.stats as stats
         import matplotlib.pyplot as plt
         import mia
```

```
Warning: Cannot change to a different GUI toolkit: qt. Using osx instead.
```

3.21 Loading and Preprocessing

Loading the hologic and synthetic datasets.

```
In [3]: hologic = pd.DataFrame.from_csv("real_texture.csv")
        hologic.drop(hologic.columns[:2], axis=1, inplace=True)
        hologic.drop('breast_area', axis=1, inplace=True)

        phantom = pd.DataFrame.from_csv("synthetic_texture.csv")
        phantom.drop(phantom.columns[:2], axis=1, inplace=True)
        phantom.drop('breast_area', axis=1, inplace=True)
```

Loading the meta data for the real and synthetic datasets.

```
In [4]: hologic_meta = mia.analysis.create_hologic_meta_data(hologic, "meta_data/real_meta.csv")
        phantom_meta = mia.analysis.create_synthetic_meta_data(phantom,
                                                               "meta_data/synthetic_meta.csv")
        phantom_meta.index.name = 'img_name'
```

Prepare the BI-RADS/VBD labels for both datasets.

```
In [5]: hologic_labels = hologic_meta.drop_duplicates().BIRADS
        phantom_labels = phantom_meta['VBD.1']

        class_labels = pd.concat([hologic_labels, phantom_labels])
        class_labels.index.name = "img_name"
        labels = mia.analysis.remove_duplicate_index(class_labels)[0]
```

3.22 Creating Features

Create blob features from distribution of blobs

```
In [6]: hologic_texture_features = mia.analysis.group_by_scale_space(hologic)
        phantom_texture_features = mia.analysis.group_by_scale_space(phantom)
```

Take a random subset of the phantom mammograms. This is important so that each case is not over represented.

```
In [7]: syn_feature_meta = mia.analysis.remove_duplicate_index(phantom_meta)
        phantom_texture_features['phantom_name'] = syn_feature_meta.phantom_name.tolist()
        phantom_texture_features_subset = mia.analysis.create_random_subset(phantom_texture_features,
                                                                            'phantom_name')
```

Combine the features from both datasets.

```
In [8]: features = pd.concat([hologic_texture_features, phantom_texture_features_subset])
        assert features.shape[0] == 366
        features.head()
```

```
Out[8]:                          contrast  dissimilarity  homogeneity    energy  \
        p214-010-60001-cl.png  180.015284      10.623496     0.092774  0.068239
        p214-010-60001-cr.png  241.273806      10.449699     0.101719  0.073124
        p214-010-60001-ml.png  217.490041      11.182093     0.089156  0.068559
        p214-010-60001-mr.png  288.353799      11.644971     0.093407  0.073823
        p214-010-60005-cl.png  160.227140      10.034384     0.099470  0.069002

                               contrast_1  dissimilarity_1  homogeneity_1  energy_1  \
        p214-010-60001-cl.png  281.402566        10.773226       0.103179  0.056349
        p214-010-60001-cr.png  257.052029        10.725951       0.098164  0.051589
        p214-010-60001-ml.png  255.275936        11.383161       0.091460  0.052581
        p214-010-60001-mr.png  237.048490        11.298087       0.089295  0.050686
        p214-010-60005-cl.png  157.603883         9.991076       0.096144  0.051515

                               contrast_2  dissimilarity_2    ...     homogeneity_7  \
        p214-010-60001-cl.png  166.662740        10.250118    ...          0.099589
        p214-010-60001-cr.png  236.447806        10.048496    ...          0.104395
        p214-010-60001-ml.png  223.895510        10.859934    ...          0.094498
        p214-010-60001-mr.png  173.248493        10.351740    ...          0.099926
        p214-010-60005-cl.png  148.042716         9.647045    ...          0.106255

                               energy_7  contrast_8  dissimilarity_8  homogeneity_8  \
        p214-010-60001-cl.png  0.016556  163.937053        10.084599       0.099519
        p214-010-60001-cr.png  0.015297  166.288393         9.918993       0.107002
        p214-010-60001-ml.png  0.014957  171.293827        10.346745       0.094648
        p214-010-60001-mr.png  0.015496  166.981656        10.147627       0.096699
        p214-010-60005-cl.png  0.017021  166.288393         9.918993       0.107002

                               energy_8  contrast_9  dissimilarity_9  homogeneity_9  \
        p214-010-60001-cl.png  0.013505  157.454233         9.781551       0.105151
        p214-010-60001-cr.png  0.022819  157.454233         9.781551       0.105151
        p214-010-60001-ml.png  0.014245  157.454233         9.781551       0.105151
        p214-010-60001-mr.png  0.013841  157.454233         9.781551       0.105151
```

```
p214-010-60005-cl.png  0.022819  127.449202         8.913421       0.110486

                       energy_9
p214-010-60001-cl.png  0.019975
p214-010-60001-cr.png  0.019975
p214-010-60001-ml.png  0.019975
p214-010-60001-mr.png  0.019975
p214-010-60005-cl.png  0.016995

[5 rows x 40 columns]
```

Filter some features, such as the min, to remove noise.

```
In [9]: selected_features = features.copy()
```

3.23 Compare Real and Synthetic Features

Compare the distributions of features detected from the real mammograms and the phantoms using the Kolmogorov-Smirnov two sample test.

```
In [80]: ks_stats = [list(stats.ks_2samp(hologic_texture_features[col],
                                        phantom_texture_features[col]))
                                        for col in hologic_texture_features.columns]

         ks_test = pd.DataFrame(ks_stats, columns=['KS', 'p-value'],
                                index=hologic_texture_features.columns)
         ks_test.to_latex("tables/texture_features_ks.tex")
         ks_test
```

Out[80]:

	KS	p-value
contrast	0.381944	5.383186e-09
dissimilarity	1.000000	3.587622e-59
homogeneity	1.000000	3.587622e-59
energy	1.000000	3.587622e-59
contrast_1	0.586111	1.318746e-20
dissimilarity_1	1.000000	3.587622e-59
homogeneity_1	1.000000	3.587622e-59
energy_1	1.000000	3.587622e-59
contrast_2	0.863889	2.874007e-44
dissimilarity_2	1.000000	3.587622e-59
homogeneity_2	1.000000	3.587622e-59
energy_2	1.000000	3.587622e-59
contrast_3	0.923611	1.538596e-50
dissimilarity_3	1.000000	3.587622e-59
homogeneity_3	1.000000	3.587622e-59
energy_3	1.000000	3.587622e-59
contrast_4	0.845833	1.870608e-42
dissimilarity_4	1.000000	3.587622e-59
homogeneity_4	1.000000	3.587622e-59
energy_4	1.000000	3.587622e-59
contrast_5	0.979167	9.485680e-57
dissimilarity_5	1.000000	3.587622e-59
homogeneity_5	1.000000	3.587622e-59
energy_5	1.000000	3.587622e-59
contrast_6	1.000000	3.587622e-59
dissimilarity_6	1.000000	3.587622e-59
homogeneity_6	1.000000	3.587622e-59
energy_6	0.994444	1.605940e-58
contrast_7	0.969444	1.230285e-55
dissimilarity_7	1.000000	3.587622e-59
homogeneity_7	1.000000	3.587622e-59
energy_7	1.000000	3.587622e-59
contrast_8	0.986111	1.497318e-57
dissimilarity_8	1.000000	3.587622e-59
homogeneity_8	1.000000	3.587622e-59
energy_8	0.997222	7.598385e-59

```
contrast_9       1.000000  3.587622e-59
dissimilarity_9  1.000000  3.587622e-59
homogeneity_9    1.000000  3.587622e-59
energy_9         1.000000  3.587622e-59
```

3.24 Dimensionality Reduction

3.24.1 t-SNE

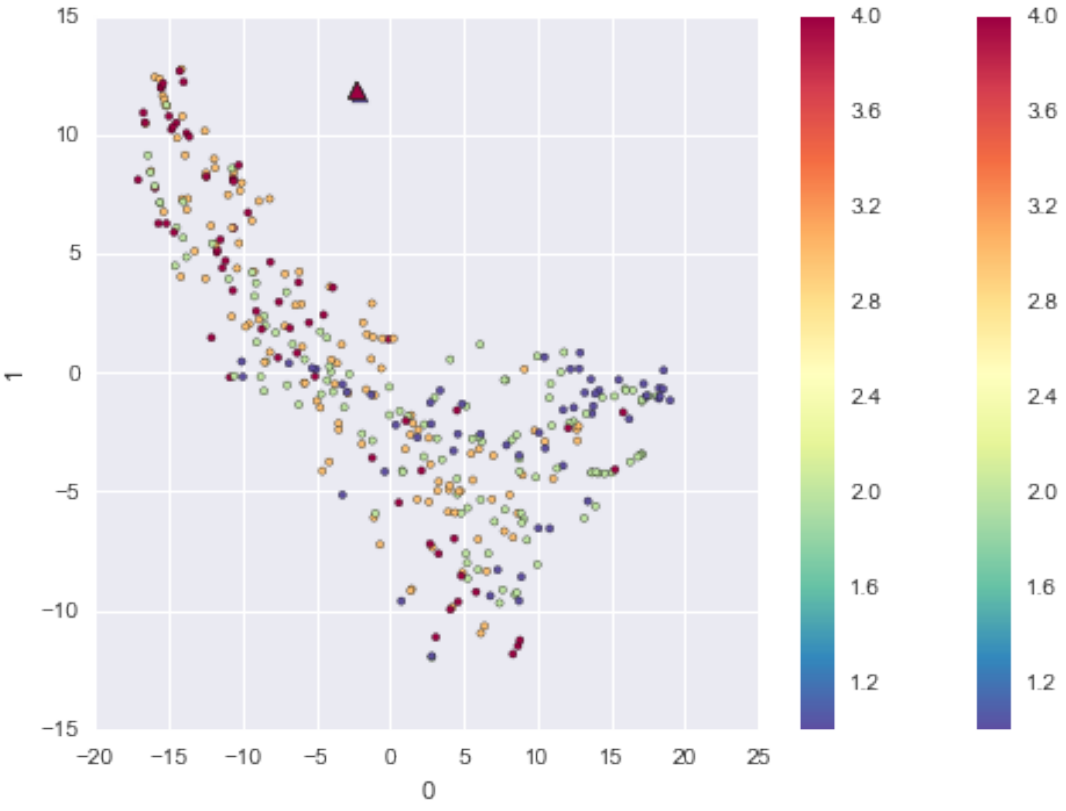
Running t-SNE to obtain a two dimensional representation.

```
In [11]: kwargs = {
             'learning_rate': 300,
             'perplexity': 30,
             'verbose': 1
         }
```

```
In [12]: SNE_mapping_2d = mia.analysis.tSNE(selected_features, n_components=2, **kwargs)
```

```
[t-SNE] Computing pairwise distances...
[t-SNE] Computed conditional probabilities for sample 366 / 366
[t-SNE] Mean sigma: 0.834920
[t-SNE] Error after 83 iterations with early exaggeration: 12.847253
[t-SNE] Error after 296 iterations: 0.467811
```

```
In [13]: mia.plotting.plot_mapping_2d(SNE_mapping_2d, hologic_texture_features.index,
                                     phantom_texture_features_subset.index, labels)
         plt.savefig('figures/mappings/texture_SNE_mapping_2d.png', dpi=300)
```



Running t-SNE to obtain a 3 dimensional mapping

```
In [14]: SNE_mapping_3d = mia.analysis.tSNE(selected_features, n_components=3, **kwargs)
```

```
[t-SNE] Computing pairwise distances...
[t-SNE] Computed conditional probabilities for sample 366 / 366
[t-SNE] Mean sigma: 0.834920
[t-SNE] Error after 100 iterations with early exaggeration: 14.777595
[t-SNE] Error after 271 iterations: 0.948866
```

```
In [42]: mia.plotting.plot_mapping_3d(SNE_mapping_3d, hologic_texture_features.index,
                                     phantom_texture_features_subset.index, labels)
```

```
Out[42]: <matplotlib.axes._subplots.Axes3DSubplot at 0x110926f90>
```

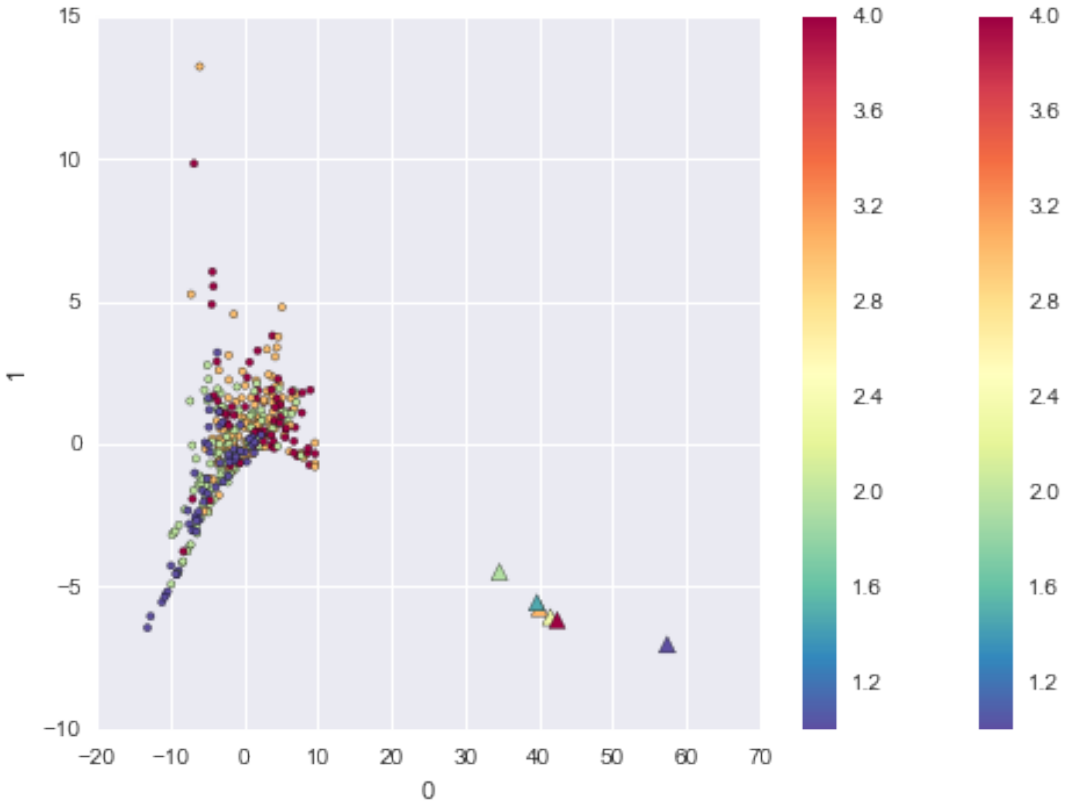
3.24.2 Isomap

Running Isomap to obtain a 2 dimensional mapping

```
In [16]: iso_kwargs = {
             'n_neighbors': 10,
         }
```

```
In [17]: iso_mapping_2d = mia.analysis.isomap(selected_features, n_components=2, **iso_kwargs)
```

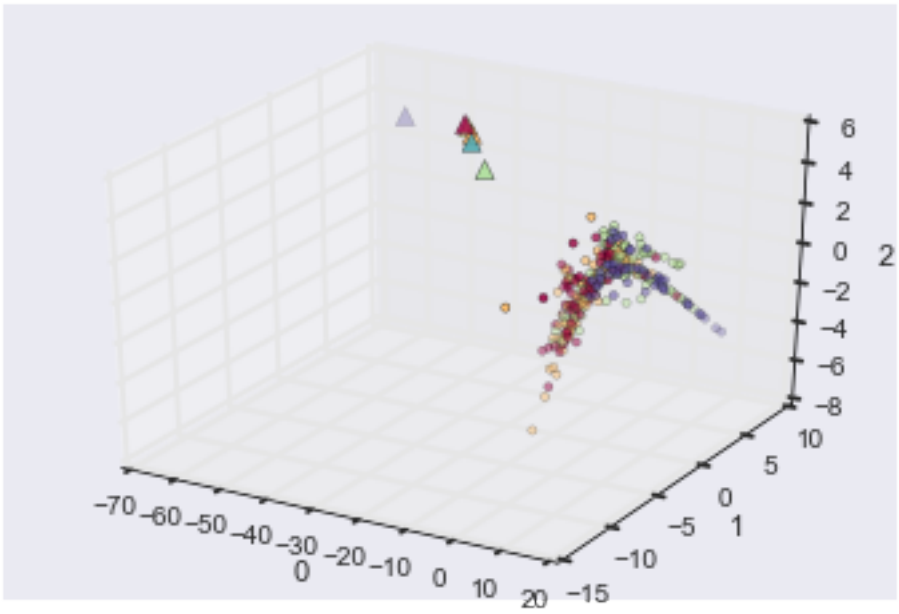
```
In [18]: mia.plotting.plot_mapping_2d(iso_mapping_2d, hologic_texture_features.index,
                                     phantom_texture_features_subset.index, labels)
         plt.savefig('figures/mappings/texture_iso_mapping_2d.png', dpi=300)
```



```
In [19]: iso_mapping_3d = mia.analysis.isomap(selected_features, n_components=3, **iso_kwargs)
```

```
In [81]: mia.plotting.plot_mapping_3d(iso_mapping_3d, hologic_texture_features.index,
                                      phantom_texture_features_subset.index, labels)
```

```
Out[81]: <matplotlib.axes._subplots.Axes3DSubplot at 0x118f53690>
```




```
<matplotlib.figure.Figure at 0x118f3ec10>
```

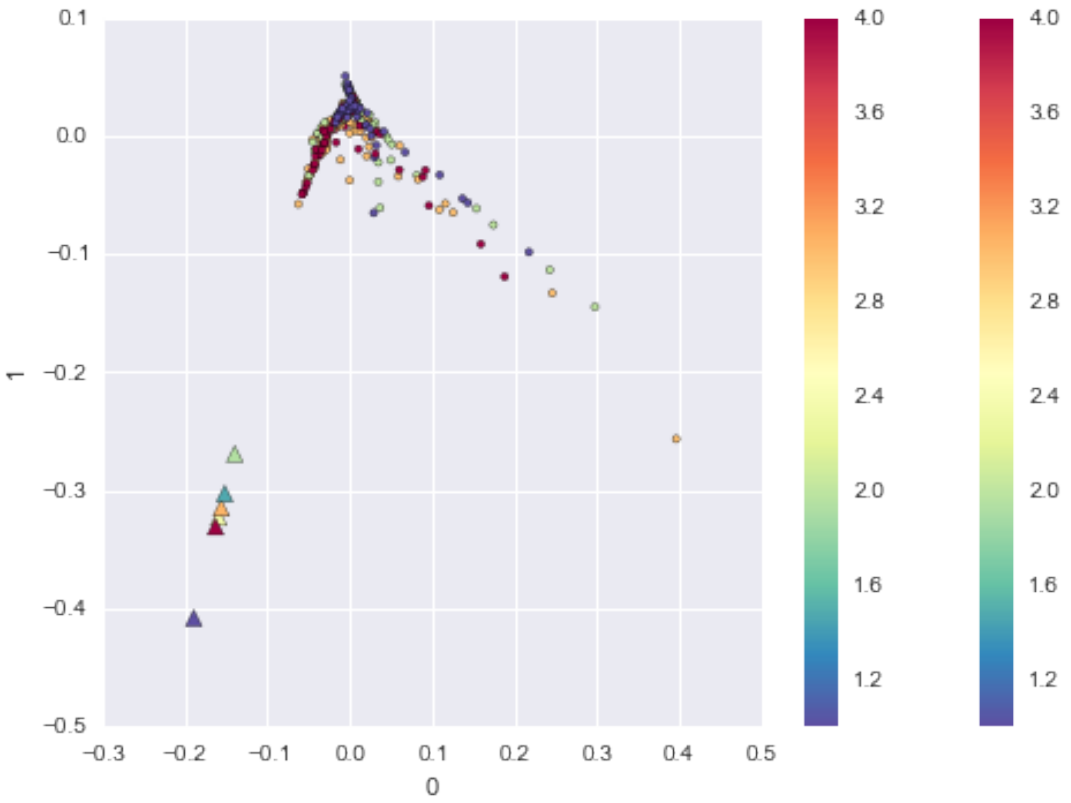
3.24.3 Locally Linear Embedding

Running locally linear embedding to obtain 2d mapping

```
In [21]: lle_kwargs = {
             'n_neighbors': 10,
         }
```

```
In [22]: lle_mapping_2d = mia.analysis.lle(selected_features, n_components=2, **lle_kwargs)
```

```
In [23]: mia.plotting.plot_mapping_2d(lle_mapping_2d, hologic_texture_features.index,
                                      phantom_texture_features_subset.index, labels)
         plt.savefig('figures/mappings/texture_lle_mapping_2d.png', dpi=300)
```




```
In [24]: lle_mapping_3d = mia.analysis.lle(selected_features, n_components=3, **lle_kwargs)
```

```
In [44]: mia.plotting.plot_mapping_3d(lle_mapping_3d, hologic_texture_features.index,
                                     phantom_texture_features_subset.index, labels)
```

```
Out[44]: <matplotlib.axes._subplots.Axes3DSubplot at 0x1105098d0>
```

3.24.4 Quality Assessment of Dimensionality Reduction

Assess the quality of the DR against measurements from the co-ranking matrices. First create co-ranking matrices for each of the dimensionality reduction mappings

```
In [26]: max_k = 10
```

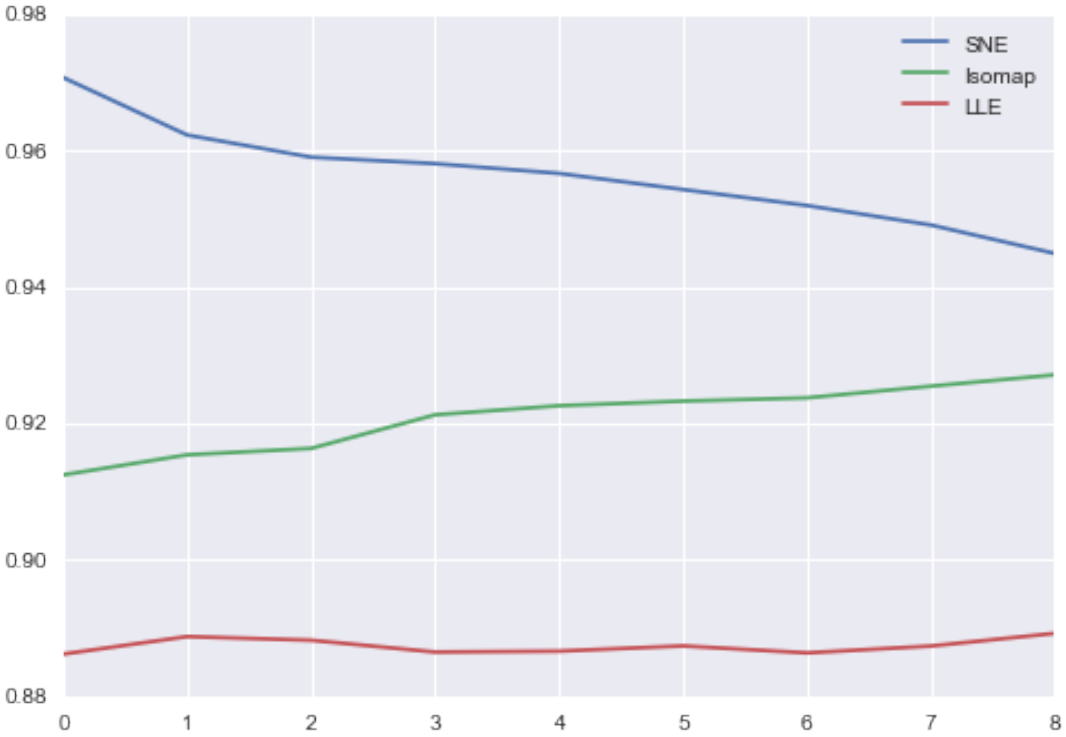
```
In [27]: SNE_mapping_2d_cm = mia.coranking.coranking_matrix(selected_features,
                                                           SNE_mapping_2d)
         iso_mapping_2d_cm = mia.coranking.coranking_matrix(selected_features,
                                                           iso_mapping_2d)
         lle_mapping_2d_cm = mia.coranking.coranking_matrix(selected_features,
                                                           lle_mapping_2d)

         SNE_mapping_3d_cm = mia.coranking.coranking_matrix(selected_features,
                                                           SNE_mapping_3d)
         iso_mapping_3d_cm = mia.coranking.coranking_matrix(selected_features,
                                                           iso_mapping_3d)
         lle_mapping_3d_cm = mia.coranking.coranking_matrix(selected_features,
                                                           lle_mapping_3d)
```

3.24.4.1 2D Mappings

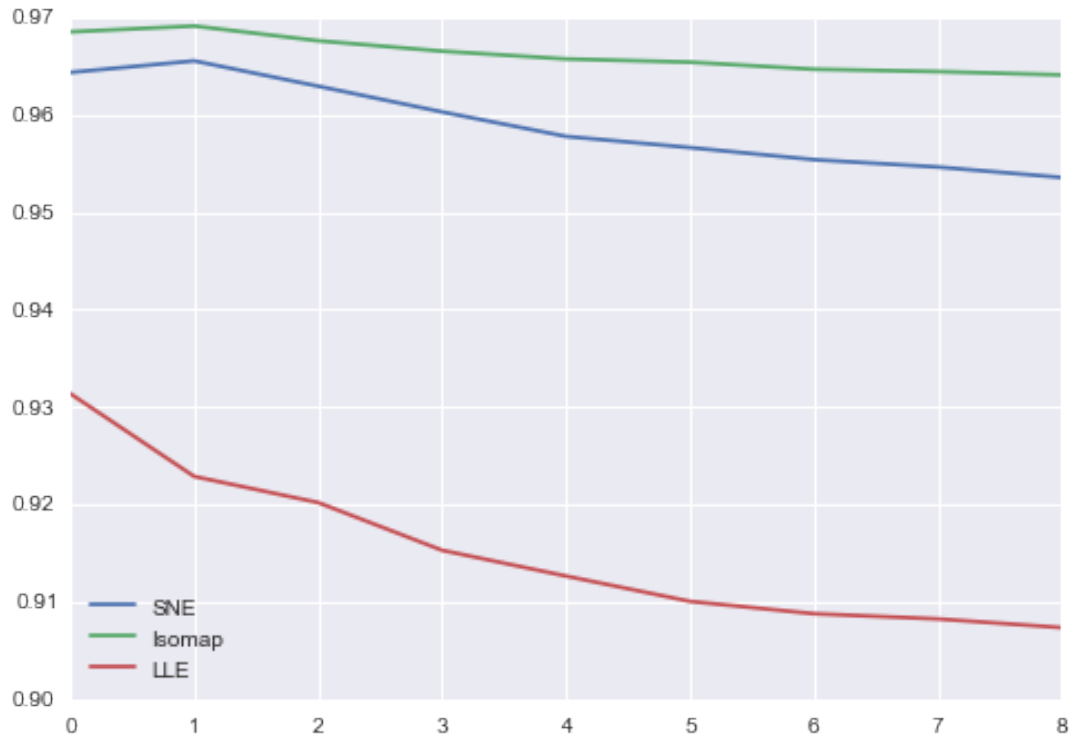
```
In [28]: SNE_trustworthiness_2d = [mia.coranking.trustworthiness(SNE_mapping_2d_cm, k)
                                   for k in range(1, max_k)]
         iso_trustworthiness_2d = [mia.coranking.trustworthiness(iso_mapping_2d_cm, k)
                                   for k in range(1, max_k)]
         lle_trustworthiness_2d = [mia.coranking.trustworthiness(lle_mapping_2d_cm, k)
                                   for k in range(1, max_k)]
```

```
In [29]: trustworthiness_df = pd.DataFrame([SNE_trustworthiness_2d,
                                            iso_trustworthiness_2d,
                                            lle_trustworthiness_2d],
                                            index=['SNE', 'Isomap', 'LLE']).T
         trustworthiness_df.plot()
         plt.savefig('figures/quality_measures/texture_trustworthiness_2d.png', dpi=300)
```



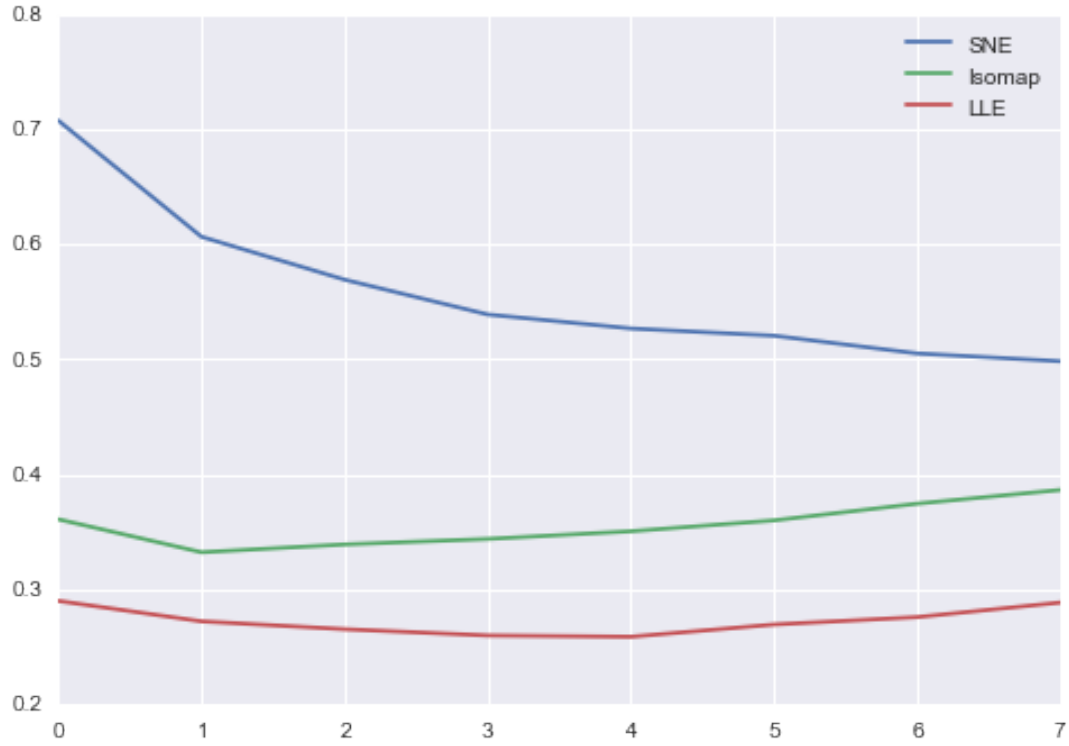

```
In [30]: SNE_continuity_2d = [mia.coranking.continuity(SNE_mapping_2d_cm, k)
                              for k in range(1, max_k)]
         iso_continuity_2d = [mia.coranking.continuity(iso_mapping_2d_cm, k)
                              for k in range(1, max_k)]
         lle_continuity_2d = [mia.coranking.continuity(lle_mapping_2d_cm, k)
                              for k in range(1, max_k)]
```

```
In [31]: continuity_df = pd.DataFrame([SNE_continuity_2d,
                                        iso_continuity_2d,
                                        lle_continuity_2d],
                                        index=['SNE', 'Isomap', 'LLE']).T
         continuity_df.plot()
         plt.savefig('figures/quality_measures/texture_continuity_2d.png', dpi=300)
```




```
In [32]: SNE_lcmc_2d = [mia.coranking.LCMC(SNE_mapping_2d_cm, k)
                        for k in range(2, max_k)]
         iso_lcmc_2d = [mia.coranking.LCMC(iso_mapping_2d_cm, k)
                        for k in range(2, max_k)]
         lle_lcmc_2d = [mia.coranking.LCMC(lle_mapping_2d_cm, k)
                        for k in range(2, max_k)]
```

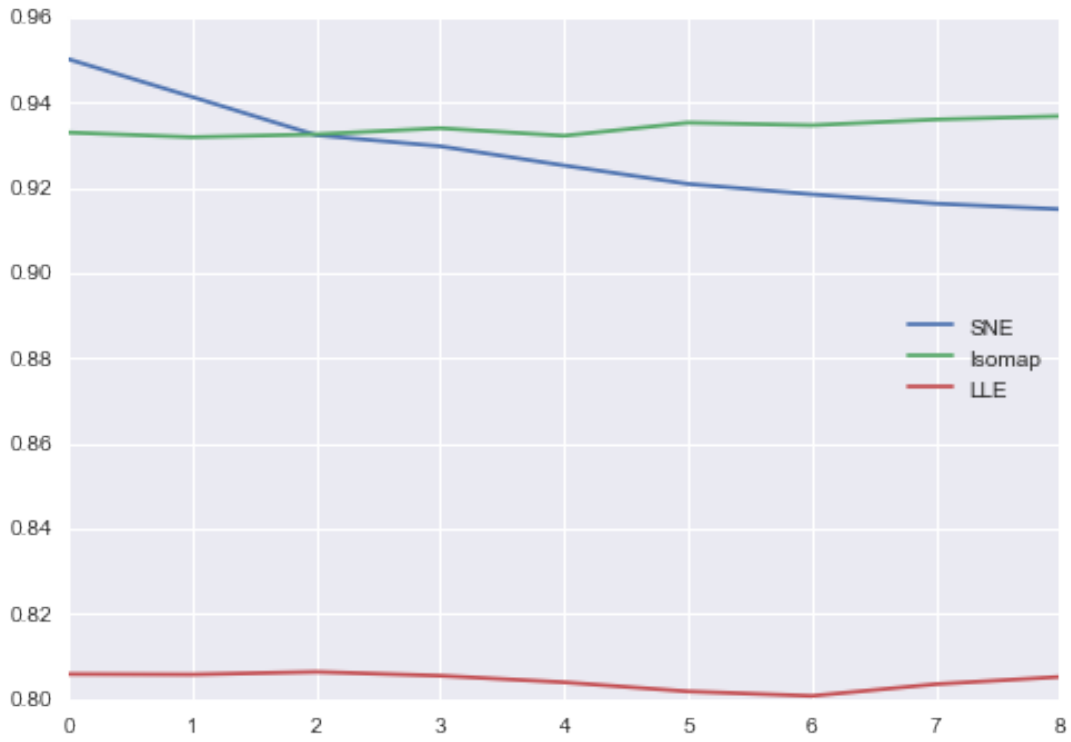
```
In [33]: lcmc_df = pd.DataFrame([SNE_lcmc_2d,
                                 iso_lcmc_2d,
                                 lle_lcmc_2d],
                                 index=['SNE', 'Isomap', 'LLE']).T
         lcmc_df.plot()
         plt.savefig('figures/quality_measures/texture_lcmc_2d.png', dpi=300)
```



3.24.4.2 3D Mappings

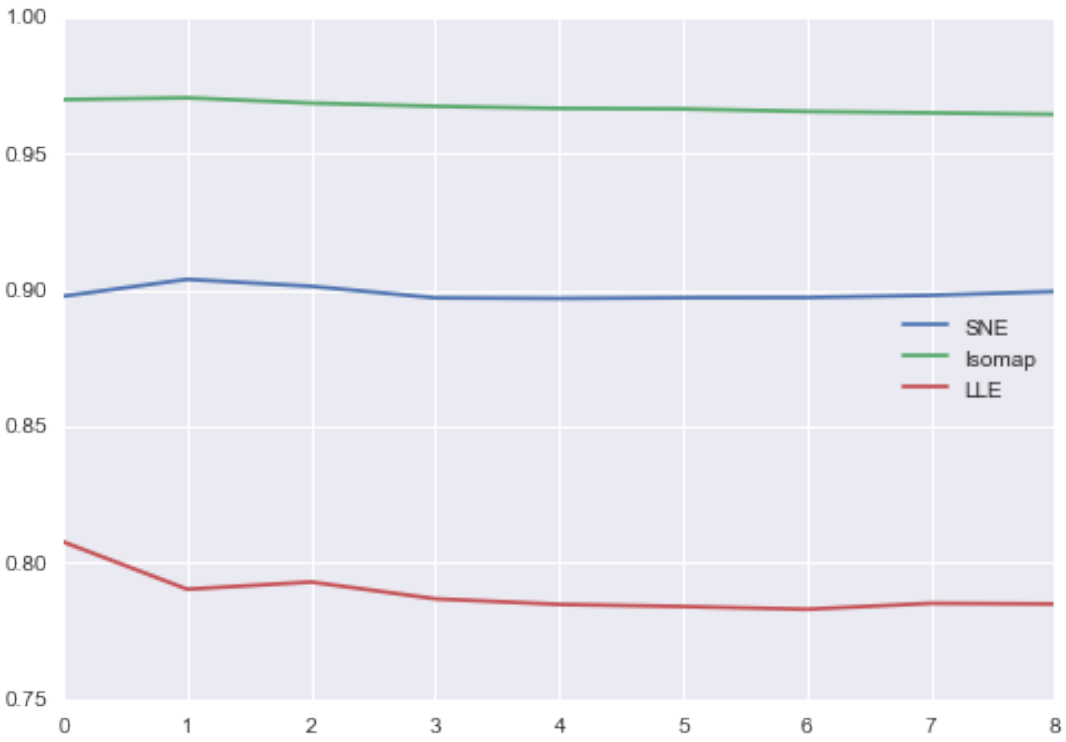
```
In [34]: SNE_trustworthiness_3d = [mia.coranking.trustworthiness(SNE_mapping_3d_cm, k)
                                   for k in range(1, max_k)]
         iso_trustworthiness_3d = [mia.coranking.trustworthiness(iso_mapping_3d_cm, k)
                                   for k in range(1, max_k)]
         lle_trustworthiness_3d = [mia.coranking.trustworthiness(lle_mapping_3d_cm, k)
                                   for k in range(1, max_k)]
```

```
In [35]: trustworthiness3d_df = pd.DataFrame([SNE_trustworthiness_3d,
                                              iso_trustworthiness_3d,
                                              lle_trustworthiness_3d],
                                              index=['SNE', 'Isomap', 'LLE']).T
         trustworthiness3d_df.plot()
         plt.savefig('figures/quality_measures/texture_trustworthiness_3d.png', dpi=300)
```



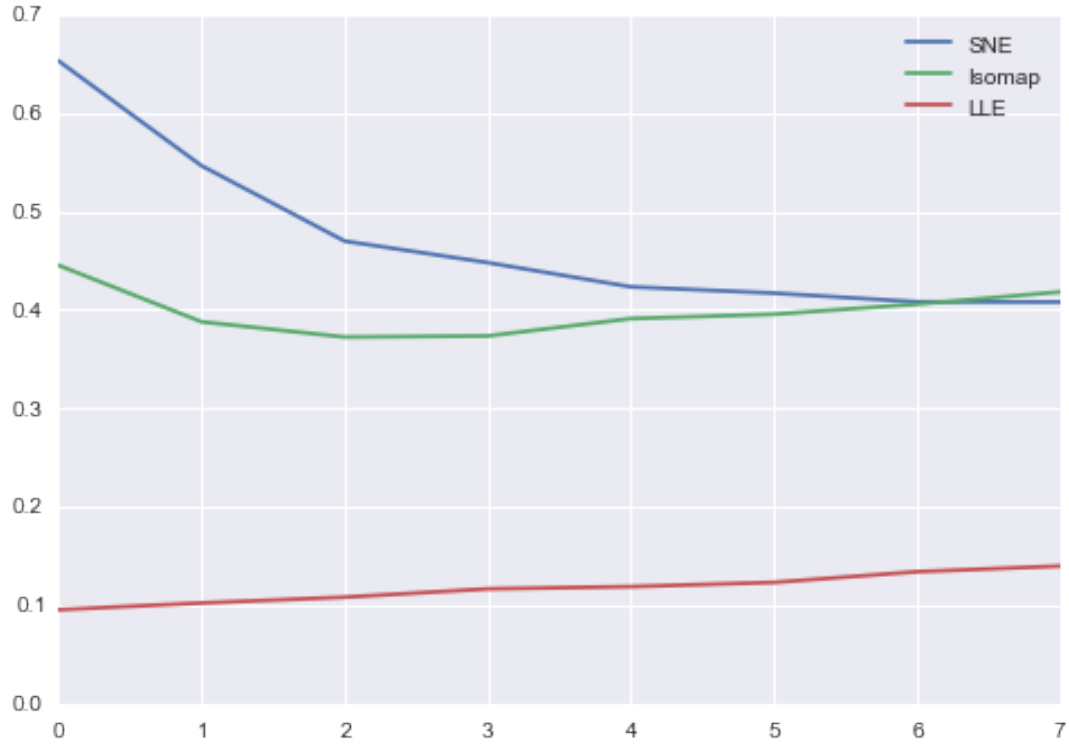

```
In [36]: SNE_continuity_3d = [mia.coranking.continuity(SNE_mapping_3d_cm, k)
                             for k in range(1, max_k)]
         iso_continuity_3d = [mia.coranking.continuity(iso_mapping_3d_cm, k)
                             for k in range(1, max_k)]
         lle_continuity_3d = [mia.coranking.continuity(lle_mapping_3d_cm, k)
                             for k in range(1, max_k)]
```

```
In [37]: continuity3d_df = pd.DataFrame([SNE_continuity_3d,
                                         iso_continuity_3d,
                                         lle_continuity_3d],
                                         index=['SNE', 'Isomap', 'LLE']).T
         continuity3d_df.plot()
         plt.savefig('figures/quality_measures/texture_continuity_3d.png', dpi=300)
```




```
In [38]: SNE_lcmc_3d = [mia.coranking.LCMC(SNE_mapping_3d_cm, k)
                        for k in range(2, max_k)]
         iso_lcmc_3d = [mia.coranking.LCMC(iso_mapping_3d_cm, k)
                        for k in range(2, max_k)]
         lle_lcmc_3d = [mia.coranking.LCMC(lle_mapping_3d_cm, k)
                        for k in range(2, max_k)]
```

```
In [39]: lcmc3d_df = pd.DataFrame([SNE_lcmc_3d,
                                  iso_lcmc_3d,
                                  lle_lcmc_3d],
                                  index=['SNE', 'Isomap', 'LLE']).T
         lcmc3d_df.plot()
         plt.savefig('figures/quality_measures/texture_lcmc_3d.png', dpi=300)
```


0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0
1
2
3
4
5
6
7
SNE
Isomap
LLE

Annotated Bibliography

[1] P. R. Bakic, M. Albert, D. Brzakovic, and A. D. Maidment, "Mammogram synthesis using a 3d simulation. i. breast tissue model and image acquisition simulation," *Medical physics*, vol. 29, no. 9, pp. 2131–2139, 2002.

[2] ——, "Mammogram synthesis using a 3d simulation. ii. evaluation of synthetic mammogram texture," *Medical physics*, vol. 29, no. 9, pp. 2140–2151, 2002.

[3] ——, "Mammogram synthesis using a three-dimensional simulation. iii. modeling and evaluation of the breast ductal network," *Medical physics*, vol. 30, no. 7, pp. 1914–1925, 2003.

[4] C. Balleyguier, S. Ayadi, K. Van Nguyen, D. Vanel, C. Dromain, and R. Sigal, "Birads^{TM} classification in mammography," *European journal of radiology*, vol. 61, no. 2, pp. 192–194, 2007.

[5] R. Bellman and R. E. Kalaba, *Dynamic programming and modern control theory*. Academic Press New York, 1965.

[6] E. Bertini, A. Tatu, and D. Keim, "Quality metrics in high-dimensional data visualization: an overview and systematization," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 17, no. 12, pp. 2203–2212, 2011.

[7] N. Boyd, J. Byng, R. Jong, E. Fishell, L. Little, A. Miller, G. Lockwood, D. Tritchler, and M. J. Yaffe, "Quantitative classification of mammographic densities and breast cancer risk: results from the canadian national breast screening study," *Journal of the National Cancer Institute*, vol. 87, no. 9, pp. 670–675, 1995.

[8] I. Buciu and A. Gacsadi, "Gabor wavelet based features for medical image analysis and classification," in *Applied Sciences in Biomedical and Communication Technologies, 2009. ISABEL 2009. 2nd International Symposium on*. IEEE, 2009, pp. 1–4.

[9] G. Carlsson, "Topology and data," *Bulletin of the American Mathematical Society*, vol. 46, no. 2, pp. 255–308, 2009.

[10] L. Chen and A. Buja, "Local multidimensional scaling for nonlinear dimension reduction, graph drawing, and proximity analysis," *Journal of the American Statistical Association*, vol. 104, no. 485, pp. 209–219, 2009.

[11] Z. Chen, "Mammographic image analysis: Risk assessment and microcalcification classi cation aspects," 2013.

[12] Z. Chen, L. Wang, E. Denton, and R. Zwiggelaar, "A multiscale blob representation of mammographic parenchymal patterns and mammographic risk assessment," in *Computer Analysis of Images and Patterns*. Springer, 2013, pp. 346–353.

[13] H. Cheng, X. Shi, R. Min, L. Hu, X. Cai, and H. Du, "Approaches for automated detection and classification of masses in mammograms," *Pattern recognition*, vol. 39, no. 4, pp. 646–668, 2006.

[14] I. Christoyianni, E. Dermatas, and G. Kokkinakis, "Fast detection of masses in computer-aided mammography," *Signal Processing Magazine, IEEE*, vol. 17, no. 1, pp. 54–64, 2000.

[15] P. Demartines and J. Hérault, "Curvilinear component analysis: A self-organizing neural network for nonlinear mapping of data sets," *Neural Networks, IEEE Transactions on*, vol. 8, no. 1, pp. 148–154, 1997.

[16] P. Domingos, "A few useful things to know about machine learning," *Communications of the ACM*, vol. 55, no. 10, pp. 78–87, 2012.

[17] J. Donaldson. (2015) T-distributed stochastic neighbor embedding for r (t-sne). [Online]. Available: http://cran.r-project.org/web/packages/tsne/tsne.pdf

[18] C. J. D'orsi, A. C. of Radiology, A. C. of Radiology, B.-R. Committee, *et al.*, *Illustrated Breast Imaging Reporting and Data System:(illustrated BI-RADS)*. American College of Radiology, 1998.

[19] N. El-Faramawy, R. Rangayyan, J. Desautels, and O. Alim, "Shape factors for analysis of breast tumors in mammograms," in *Electrical and Computer Engineering, 1996. Canadian Conference on*, vol. 1. IEEE, 1996, pp. 355–358.

[20] U. Fischer, F. Baum, and S. Luftner-Nagel, *Breast imaging*. Thieme, 2008.

[21] K. Ganesan, U. Acharya, C. K. Chua, L. C. Min, K. Abraham, and K. Ng, "Computer-aided breast cancer detection using mammograms: A review," *Biomedical Engineering, IEEE Reviews in*, vol. 6, pp. 77–98, 2013.

[22] I. T. Gram, E. Funkhouser, and L. Tabár, "The tabar classification of mammographic parenchymal patterns," *European journal of radiology*, vol. 24, no. 2, pp. 131–136, 1997.

[23] S. E. Grigorescu, N. Petkov, and P. Kruizinga, "Comparison of texture features based on gabor filters," *Image Processing, IEEE Transactions on*, vol. 11, no. 10, pp. 1160–1167, 2002.

[24] R. M. Haralick, K. Shanmugam, and I. H. Dinstein, "Textural features for image classification," *Systems, Man and Cybernetics, IEEE Transactions on*, no. 6, pp. 610–621, 1973.

[25] G. E. Hinton and S. T. Roweis, "Stochastic neighbor embedding," in *Advances in neural information processing systems*, 2002, pp. 833–840.

[26] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[27] D. M. A. Holger Diedrich. (2012) Locally linear embedding. [Online]. Available: http://cran.r-project.org/web/packages/lle/lle.pdf

[28] A. Inselberg and B. Dimsdale, "Parallel coordinates," in *Human-Machine Interactive Systems*. Springer, 1991, pp. 199–233.

[29] Itseez. (2015) Opencv. [Online]. Available: http://opencv.org

[30] S. Johansson and J. Johansson, "Interactive dimensionality reduction through user-defined combinations of quality metrics," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 15, no. 6, pp. 993–1000, 2009.

[31] E. Jones, T. Oliphant, and P. Peterson, "{SciPy}: Open source scientific tools for {Python}," 2014.

[32] S. Kaski, J. Nikkilä, M. Oja, J. Venna, P. Törönen, and E. Castrén, "Trustworthiness and metrics in visualizing similarity of gene expression," *BMC bioinformatics*, vol. 4, no. 1, p. 48, 2003.

[33] J. Kilday, F. Palmieri, and M. D. Fox, "Classifying mammographic lesions using computerized image analysis," *Medical Imaging, IEEE Transactions on*, vol. 12, no. 4, pp. 664–669, 1993.

[34] D. B. Kopans and D. Kopans, *Breast imaging*. Lippincott-Raven Philadelphia, 1998.

[35] Lambda Foundry, Inc. and PyData Development Team. (2015) Python data analysis library. [Online]. Available: http://pandas.pydata.org/index.html

[36] J. A. Lee and M. Verleysen, *Nonlinear dimensionality reduction*. Springer Science & Business Media, 2007.

[37] ——, "Quality assessment of dimensionality reduction: Rank-based criteria," *Neurocomputing*, vol. 72, no. 7, pp. 1431–1443, 2009.

[38] J. A. Lee, M. Verleysen, *et al.*, "Rank-based quality assessment of nonlinear dimensionality reduction." in *ESANN*, 2008, pp. 49–54.

[39] S. Marčelja, "Mathematical description of the responses of simple cortical cells*," *JOSA*, vol. 70, no. 11, pp. 1297–1300, 1980.

[40] V. A. McCormack and I. dos Santos Silva, "Breast density and parenchymal patterns as markers of breast cancer risk: a meta-analysis," *Cancer Epidemiology Biomarkers & Prevention*, vol. 15, no. 6, pp. 1159–1169, 2006.

[41] B. Mokbel, W. Lueks, A. Gisbrecht, and B. Hammer, "Visualizing the quality of dimensionality reduction," *Neurocomputing*, vol. 112, pp. 109–123, 2013.

[42] L. Novakova and O. Stepankova, "Radviz and identification of clusters in multidimensional data," in *Information Visualisation, 2009 13th International Conference*. IEEE, 2009, pp. 104–109.

[43] NumPy Developers. (2013) Numpy. [Online]. Available: http://www.numpy.org

[44] A. C. of Radiology. BI-RADS Committee and A. C. of Radiology, *Breast imaging reporting and data system*. American College of Radiology, 1998.

[45] I. U. P. on Breast Cancer Screening *et al.*, "The benefits and harms of breast cancer screening: an independent review," *The Lancet*, vol. 380, no. 9855, pp. 1778–1786, 2012.

[46] N. Otsu, "A threshold selection method from gray-level histograms," *Automatica*, vol. 11, no. 285-296, pp. 23–27, 1975.

[47] J. R. Parker, *Algorithms for image processing and computer vision*. John Wiley & Sons, 2010.

[48] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, *et al.*, "Scikit-learn: Machine learning in python," *The Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[49] T. Perciano. (2014) R image processing and analysis. [Online]. Available: http://cran.r-project.org/web/packages/ripa/ripa.pdf

[50] N. Petrick, H.-P. Chan, B. Sahiner, and M. A. Helvie, "Combined adaptive enhancement and region-growing segmentation of breast masses on digitized mammograms," *Medical Physics*, vol. 26, no. 8, pp. 1642–1654, 1999.

[51] N. Petrick, H.-P. Chan, D. Wei, B. Sahiner, M. A. Helvie, and D. D. Adler, "Automated detection of breast masses on mammograms using adaptive contrast enhancement and texture classification," *Medical Physics*, vol. 23, no. 10, pp. 1685–1696, 1996.

[52] Python Software Foundation. (2015) Python. [Online]. Available: https://www.python.org

[53] R Language. (2015) Principal components analysis. [Online]. Available: https://stat.ethz.ch/R-manual/R-patched/library/stats/html/princomp.html

[54] A. Ronacher. (2015) Click library. [Online]. Available: http://click.pocoo.org/4/

[55] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[56] M. P. Sampat, M. K. Markey, A. C. Bovik, *et al.*, "Computer-aided detection and diagnosis in mammography," *Handbook of image and video processing*, vol. 2, no. 1, pp. 1195–1217, 2005.

[57] SciPy Developers. (2015) Scipy. [Online]. Available: http://www.scipy.org

[58] J. Shlens, "A tutorial on principal component analysis," *CoRR*, vol. abs/1404.1100, 2014. [Online]. Available: http://arxiv.org/abs/1404.1100

[59] R. Siegel, J. Ma, Z. Zou, and A. Jemal, "Cancer statistics, 2014," *CA: a cancer journal for clinicians*, vol. 64, no. 1, pp. 9–29, 2014.

[60] R. A. Smith, D. Manassaram-Baptiste, D. Brooks, V. Cokkinides, M. Doroshenk, D. Saslow, R. C. Wender, and O. W. Brawley, "Cancer screening in the united states, 2014: a review of current american cancer society guidelines and current issues in cancer screening," *CA: a cancer journal for clinicians*, vol. 64, no. 1, pp. 30–51, 2014.

[61] M. Sonka, V. Hlavac, and R. Boyle, *Image processing, analysis, and machine vision*. Cengage Learning, 2014.

[62] H. Strange and R. Zwiggelaar, *Open Problems in Spectral Dimensionality Reduction*. Springer, 2014.

[63] L. Tabár, T. Tot, and P. B. Dean, *Breast cancer: the art and science of early detection with mammography: perception, interpretation, histopathologic correlation*. Thieme, 2005.

[64] A. Tatu, G. Albuquerque, M. Eisemann, J. Schneidewind, H. Theisel, M. Magnor, and D. Keim, "Combining automated analysis and visualization techniques for effective exploration of high-dimensional data," in *Visual Analytics Science and Technology, 2009. VAST 2009. IEEE Symposium on*. IEEE, 2009, pp. 59–66.

[65] J. B. Tenenbaum, V. De Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

[66] The MathWorks, Inc. (2015) Matlab. [Online]. Available: http://uk.mathworks.com/products/matlab/

[67] The R Foundation. (2015) The r project for statistical computing. [Online]. Available: http://www.r-project.org

[68] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. 2579-2605, p. 85, 2008.

[69] S. Van Der Walt, J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart, and T. Yu, "scikit-image: image processing in python," *PeerJ*, vol. 2, p. e453, 2014.

[70] J. S. Weszka, C. R. Dyer, and A. Rosenfeld, "A comparative study of texture measures for terrain classification," *Systems, Man and Cybernetics, IEEE Transactions on*, no. 4, pp. 269–285, 1976.

[71] H. Wickham. (2013) ggplot2. [Online]. Available: http://ggplot2.org

[72] J. N. Wolfe, "Breast patterns as an index of risk for developing breast cancer," *American Journal of Roentgenology*, vol. 126, no. 6, pp. 1130–1137, 1976.

[73] Z. You and A. K. Jain, "Performance evaluation of shape matching via chord length distribution," *Computer vision, graphics, and image processing*, vol. 28, no. 2, pp. 185–198, 1984.

[74] C. Zaiontz. (2015) Real statistics. [Online]. Available: http://www.real-statistics.com/non-parametric-tests/two-sample-kolmogorov-smirnov-test/

[75] R. Zwiggelaar, T. C. Parr, and C. J. Taylor, "Finding orientated line patterns in digital mammographic images." in *BMVC*, 1996, pp. 1–10.