Journal of Machine Learning Research 11 (2010) 411-450 Submitted 11/07; Revised 11/09; Published 1/10

Dimensionality Estimation, Manifold Learning and Function Approximation using Tensor Voting

Philippos Mordohai MORDOHAI@CS.STEVENS.EDU
Castle Point on Hudson
Department of Computer Science
Stevens Institute of Technology
Hoboken, NJ 07030, USA

Gérard Medioni MEDIONI@IRIS.USC.EDU
3737 Watt Way
Institute for Robotics and Intelligent Systems
University of Southern California
Los Angeles, CA 90089, USA

Editor: Sanjoy Dasgupta

Abstract

We address instance-based learning from a perceptual organization standpoint and present methods for dimensionality estimation, manifold learning and function approximation. Under our approach, manifolds in high-dimensional spaces are inferred by estimating geometric relationships among the input instances. Unlike conventional manifold learning, we do not perform dimensionality reduction, but instead perform all operations in the original input space. For this purpose we employ a novel formulation of tensor voting, which allows an N-D implementation. Tensor voting is a perceptual organization framework that has mostly been applied to computer vision problems. Analyzing the estimated local structure at the inputs, we are able to obtain reliable dimensionality estimates at each instance, instead of a global estimate for the entire data set. Moreover, these local dimensionality and structure estimates enable us to measure geodesic distances and perform nonlinear interpolation for data sets with varying density, outliers, perturbation and intersections, that cannot be handled by state-of-the-art methods. Quantitative results on the estimation of local manifold structure using ground truth data are presented. In addition, we compare our approach with several leading methods for manifold learning at the task of measuring geodesic distances. Finally, we show competitive function approximation results on real data.

Keywords: dimensionality estimation, manifold learning, geodesic distance, function approximation, high-dimensional processing, tensor voting

1. Introduction

In this paper, we address a subfield of machine learning that operates in continuous domains and learns from observations that are represented as points in a Euclidean space. This type of learning is termed *instance-based* or *memory-based* learning (Mitchell, 1997). The goal of instance-based learning is to learn the underlying relationships between observations, which are points in an N-D continuous space, under the assumption that they lie in a limited part of the space, typically a manifold, the dimensionality of which is an indication of the degrees of freedom of the underlying system.

Instance-based learning has recently received renewed interest from the machine learning community, due to its many applications in the fields of pattern recognition, data mining, kinematics, function approximation and visualization, among others. This interest was sparked by a wave of new algorithms that advanced the state of the art and are capable of learning nonlinear manifolds in spaces of very high dimensionality. These include kernel PCA (Schölkopf et al., 1998), locally linear embedding (LLE) (Roweis and Saul, 2000), Isomap (Tenenbaum et al., 2000) and charting (Brand, 2003), which are reviewed in Section 2. They aim at reducing the dimensionality of the input space in a way that preserves certain geometric or statistical properties of the data. Isomap, for instance, attempts to preserve the geodesic distances between all points as the manifold is "unfolded" and mapped to a space of lower dimension.

Our research focuses on data presented as large sets of observations, possibly containing outliers, in high dimensions. We view the problem of learning an unknown function based on these observations as equivalent to learning a manifold, or manifolds, formed by a set of points. Having a good estimate of the manifold's structure, one is able to predict the positions of other points on it. The first task is to determine the intrinsic dimensionality of the data. This can provide insights on the complexity of the system that generates the data, the type of model needed to describe it, as well as the actual degrees of freedom of the system, which are not equal to the dimensionality of the input space, in general. We also estimate the orientation of a potential manifold that passes through each point. Dimensionality estimation and structure inference are accomplished simultaneously by encoding the observations as symmetric, second order, non-negative definite tensors and analyzing the outputs of tensor voting (Medioni et al., 2000). Since the process that estimates dimensionality and orientation is performed on the inputs, our approach falls under the "eager learning" category, according to Mitchell (1997). Unlike other eager approaches, however, ours is not global. This offers considerable advantages when the data become more complex, or when the number of instances is large.

We take a different path to manifold learning than Roweis and Saul (2000), Tenenbaum et al. (2000) and Brand (2003). Whereas these methods address the problem as one of dimensionality reduction, we propose an approach that does not embed the data in a lower dimensional space. Preliminary versions of this approach were published in Mordohai and Medioni (2005) and Mordohai (2005). A similar methodology was also presented by Dollár et al. (2007a). We compute local dimensionality estimates, but instead of performing dimensionality reduction, we perform all operations in the original input space, taking into account the estimated dimensionality of the data. We also estimate the orientation of the manifold locally and are able to approximate intrinsic or geodesic distances[1] and perform nonlinear interpolation. Since we perform all processing in the input space we are able to process data sets that are not manifolds globally, or ones with varying intrinsic dimensionality. The latter pose no additional difficulties, since we do not use a global estimate for the dimensionality of the data. Moreover, outliers, boundaries, intersections or disconnected components are handled naturally as in 2-D and 3-D (Medioni et al., 2000; Tang and Medioni, 1998). Quantitative results for the robustness to outliers that outnumber the inliers are presented in Sections 5 and 6. Once processing under our approach has been completed, dimensionality reduction

1. The requirements for a distance to be geodesic are stricter than those for being intrinsic. Intrinsic distances are computed on the manifold, while geodesics are based on the gradient of the distance function. For the manifolds we examine in this paper, these two almost always coincide. See Mémoli and Sapiro (2005) and references therein for more details.

can be performed using any of the approaches described in the next section to reduce the storage requirements, if appropriate and desirable.

Manifold learning serves as the basis for the last part of our work, which addresses function approximation. As suggested by Poggio and Girosi (1990), function approximation from samples and hypersurface inference are equivalent. The main assumption is that some form of smoothness exists in the data and unobserved outputs can be predicted from previously observed outputs for similar inputs. The distinction between low and high-dimensional spaces is necessary, since highly specialized methods for low-dimensional cases exist in the literature. Our approach is local, non-parametric and has a weak prior model of smoothness, which is implemented in the form of votes that communicate a point's preferred orientation to its neighbors. This generic prior and the absence of global computations allow us to address a large class of functions as well as data sets comprising very large numbers of observations. As most of the local methods reviewed in the next section, our algorithm is memory-based. This increases flexibility, since we can process data that do not conform to pre-specified models, but also increases storage requirements, since all samples are kept in memory.

All processing in our method is performed using tensor voting, which is a computational framework for perceptual organization based on the Gestalt principles of proximity and good continuation (Medioni et al., 2000). It has mainly been applied to organize generic points into coherent groups and for computer vision problems that were formulated as perceptual organization tasks. For instance, the problem of stereo vision can be formulated as the organization of potential pixel correspondences into salient surfaces, under the assumption that correct correspondences form coherent surfaces and wrong ones do not (Mordohai and Medioni, 2006). Salient structures are inferred based on the support potential correspondences receive from their neighbors in the form of votes, which are also second order tensors that are cast from each point to all other points within its neighborhood. Each vote conveys the orientation the receiver would have if the voter and receiver were in the same structure. In Section 3, we present a new implementation of tensor voting that is not limited to low-dimensional spaces as the original one of Medioni et al. (2000).

The paper is organized as follows: an overview of related work including the algorithms that are compared with ours is given in the next section; a new implementation of tensor voting applicable to N-D data is described in Section 3; results in dimensionality estimation are presented in Section 4, while results in local structure estimation are presented in Section 5; our algorithm for estimating geodesic distances and a quantitative comparison with state of the art methods are shown in Section 6; function approximation is described in Section 7; finally, Section 8 concludes the paper.

2. Related Work

In this section, we present related work in the domains of dimensionality estimation, manifold learning and multivariate function approximation.

2.1 Dimensionality Estimation

Bruske and Sommer (1998) present an approach for dimensionality estimation where an optimally topology preserving map (OTPM) is constructed for a subset of the data, which is produced after vector quantization. Principal Component Analysis (PCA) is then performed for each node of the OTPM under the assumption that the underlying structure of the data is locally linear. The average of the number of significant singular values at the nodes is the estimate of the intrinsic dimensionality.

Kégl (2003) estimates the capacity dimension of a manifold, which is equal to the topological dimension and does not depend on the distribution of the data, using an efficient approximation based on packing numbers. The algorithm takes into account dimensionality variations with scale and is based on a geometric property of the data, rather than successive projections to increasingly higher-dimensional subspaces until a certain percentage of the data is explained. Raginsky and Lazebnik (2006) present a family of dimensionality estimators based on the concept of quantization dimension. The family is parameterized by the distortion exponent and includes the method of Kégl (2003) when the distortion exponent tends to infinity. The authors show that small values of the distortion exponent yield estimators that are more robust to noise.

Costa and Hero (2004) estimate the intrinsic dimension of the manifold and the entropy of the samples using geodesic-minimal-spanning trees. The method, similarly to Isomap (Tenenbaum et al., 2000), considers global properties of the adjacency graph and thus produces a single global estimate.

Levina and Bickel (2005) compute maximum likelihood estimates of dimensionality by examining the number of neighbors included in spheres, the radii of which are selected in such a way that they contain enough points and that the density of the data contained in them can be assumed constant. These requirements cause an underestimation of the dimensionality when it is very high.

The difference between our approach and those of Bruske and Sommer (1998), Kégl (2003), Brand (2003), Weinberger and Saul (2004), Costa and Hero (2004) and Levina and Bickel (2005) is that it produces reliable dimensionality estimates at the point level. While this is not important for data sets with constant dimensionality, the ability to estimate local dimensionality reliably becomes a key factor when dealing with data generated by different unknown processes. Given reliable local estimates, the data set can be segmented in components with constant dimensionality.

2.2 Manifold Learning

Here, we briefly present recent approaches for learning low dimensional embeddings from points in high dimensional spaces. Most of them are inspired by linear techniques, such as Principal Component Analysis (PCA) (Jolliffe, 1986) and Multi-Dimensional Scaling (MDS) (Cox and Cox, 1994), based on the assumption that nonlinear manifolds can be approximated by locally linear parts.

Schölkopf et al. (1998) propose kernel PCA that extends linear PCA by implicitly mapping the inputs to a higher-dimensional space via kernels. Conceptually, applying PCA in the high-dimensional space allows the extraction of principal components that capture more information than their counterparts in the original space. The mapping to the high-dimensional space does not need to carried out explicitly, since dot product computations suffice. The choice of kernel is still an open problem. Weinberger et al. (2004) describe an approach to compute the kernel matrix by maximizing variance in feature space in the context of dimensionality reduction.

Locally Linear Embedding (LLE) was presented by Roweis and Saul (2000) and Saul and Roweis (2003). The underlying assumption is that if data lie on a locally linear, low-dimensional manifold, then each point can be reconstructed from its neighbors with appropriate weights. These weights should be the same in a low-dimensional space, the dimensionality of which is greater or equal to the intrinsic dimensionality of the manifold. The LLE algorithm computes the basis of such a low-dimensional space. The dimensionality of the embedding, however, has to be given as a parameter, since it cannot always be estimated from the data (Saul and Roweis, 2003). Moreover,

the output is an embedding of the given data, but not a mapping from the ambient to the embedding space. Global coordination of the local embeddings, and thus a mapping, can be computed according to Teh and Roweis (2003). LLE is not isometric and often fails by mapping distant points close to each other.

Tenenbaum et al. (2000) propose Isomap, which is an extension of MDS that uses geodesic instead of Euclidean distances and thus can be applied to nonlinear manifolds. The geodesic distances between points are approximated by graph distances. Then, MDS is applied on the geodesic distances to compute an embedding that preserves the property of points to be close or far away from each other. Isomap can handle points not in the original data set, and perform interpolation. C-Isomap, a variant of Isomap applicable to data with intrinsic curvature, but known distribution, and L-Isomap, a faster alternative that only uses a few landmark point for distance computations, have also been proposed by de Silva and Tenenbaum (2003). Isomap and its variants are limited to convex data sets.

The Laplacian Eigenmaps algorithm was developed by Belkin and Niyogi (2003). It computes the normalized graph Laplacian of the adjacency graph of the input data, which is an approximation of the Laplace-Beltrami operator on the manifold. It exploits locality preserving properties that were first observed in the field of clustering. The Laplacian Eigenmaps algorithm can be viewed as a generalization of LLE, since the two become identical when the weights of the graph are chosen according to the criteria of the latter. Much like LLE, the dimensionality of the manifold also has to be provided, the computed embeddings are not isometric and a mapping between the two spaces is not produced. The latter is addressed by He and Niyogi (2004) where a variation of the algorithm is proposed.

Donoho and Grimes (2003) propose Hessian LLE (HLLE), an approach similar to the above, which computes the Hessian instead of the Laplacian of the graph. The authors claim that the Hessian is better suited than the Laplacian in detecting linear patches on the manifold. The major contribution of this approach is that it proposes a global, isometric method, which, unlike Isomap, can be applied to non-convex data sets. The requirement to estimate second derivatives from possibly noisy, discrete data makes the algorithm more sensitive to noise than the others reviewed here.

Semidefinite Embedding (SDE) was proposed by Weinberger and Saul (2004, 2006) who address the problem of manifold learning by enforcing local isometry. The lengths of the sides of triangles formed by neighboring points are preserved during the embedding. These constraints can be expressed in terms of pairwise distances and the optimal embedding can be found by semidefinite programming. The method is among the most computationally demanding reviewed here, but can reliably estimate the underlying dimensionality of the inputs by locating the largest gap between the eigenvalues of the Gram matrix of the outputs. Similarly to our approach, dimensionality estimation does not require a threshold.

Other research related to ours includes the charting algorithm of Brand (2003). It computes a pseudo-invertible mapping of the data, as well as the intrinsic dimensionality of the manifold, which is estimated by examining the rate of growth of the number of points contained in hyper-spheres as a function of the radius. Linear patches, areas of curvature and noise can be distinguished using the proposed measure. At a subsequent stage a global coordinate system for the embedding is defined. This produces a mapping between the input space and the embedding space.

Wang et al. (2005) propose an adaptive version of the local tangent space alignment (LTSA) of Zhang and Zha (2004), a local dimensionality reduction method that is a variant of LLE. Wang et al. address a limitation of most of the approaches presented in this section, which is the use of

a fixed number of neighbors (k) for all points in the data. Inappropriate selection of k can cause problems at points near boundaries, or if the density of the data is not approximately constant. The authors propose a method to adapt the neighborhood size according to local criteria and demonstrate its effectiveness on data sets of varying distribution. Using an appropriate value for k at each point is important for graph-based methods, since the contributions of each neighbor are typically not weighted, making the algorithms very sensitive to the selection of k.

In a more recent paper, Sha and Saul (2005) propose Conformal Eigenmaps, an algorithm that operates on the output of LEE or Laplacian Eigenmaps to produce a conformal embedding, which preserves angles between edges in the original input space, without incurring a large increase in computational cost. A similar approach that "stiffens" the inferred manifolds employing a multi-resolution strategy was proposed by Brand (2005). Both these papers address the limitation of some of the early algorithms which preserve graph connectivity, but not local structure, during the embedding.

The most similar method to ours is that of Dollár et al. (2007a) and Dollár et al. (2007b) in which the data are not embedded in a lower dimensional space. Instead, the local structure of a manifold at a point is learned from neighboring observations and represented by a set of radial basis functions (RBFs) centered on K points discovered by K-means clustering. The manifold can then be traversed by "walking" on its tangent space between and beyond the observations. Representation by RBFs without dimensionality reduction allows the algorithm to be robust to outliers and be applicable to non-isometric manifolds. An evaluation of manifold learning using geodesic distance preservation as a metric, similar to the one of Section 6.1, is presented in Dollár et al. (2007b).

A different approach for intrinsic distance estimation that bypasses learning the structure of the manifold has been proposed by Mémoli and Sapiro (2005). It approximates intrinsic distances and geodesics by computing extrinsic Euclidean distances in a thin band that surrounds the points. The algorithm can handle manifolds in any dimension and of any co-dimension and is more robust to noise than graph-based methods, such as Isomap, since in the latter the outliers are included in the graph and perturb the approximation of geodesics.

Souvenir and Pless (2005) present an approach capable of learning multiple, potentially intersecting, manifolds of different dimensionality using an expectation maximization (EM) algorithm with a variant of MDS as the M step. Unlike our approach, however, the number and dimensionality of the manifolds have to be provided externally.

2.3 Function Approximation

Here we review previous research on function approximation focusing on methods that are applicable to large data sets in high-dimensional spaces. Neural networks are often employed as global methods for function approximation. Poggio and Girosi (1990) addressed function approximation in a regularization framework implemented as a three-layer neural network. They view the problem as hypersurface reconstruction, where the main assumption is that of smoothness. The emphasis is on the selection of the appropriate approximating functions and optimization algorithm. Other research based on neural networks includes the work of Sanger (1991) who proposed a tree-structured neural network, which does not suffer from the exponential growth with dimensionality of the number of models and parameters that plagued previous approaches. It does, however, require the selection of an appropriate set of basis functions to approximate a given function. Neural network based approaches with pre-specified types of models are also proposed by: Barron (1993) who derived

the bounds for approximation using a superposition of sigmoidal functions; Breiman (1993) who proposed a simpler and faster model based on hinging hyperplanes; and Saha et al. (1993) who used RBFs.

Xu et al. (1995) modified the training scheme for the mixture of experts architecture so that a single-loop EM algorithm is sufficient for optimization. Mitaim and Kosko (2001) approached the problem within the fuzzy systems framework. They investigated the selection of the shape of fuzzy sets for an adaptive fuzzy system and concluded that no shape emerges as the best choice. These approaches, as well as the ones based on neural networks, are global and model-based. They can achieve good performance, but they require all the inputs to be available at the same time for training and the selection of an appropriate model that matches the unknown function. If the latter is complex, the resulting model may have an impractically large number of parameters.

Support Vector Machines (SVMs), besides classification, have also been extensively applied for regression based on the work of Vapnik (1995). Collobert and Bengio (2001) address a limitation of the SVM algorithm for regression, which is its increased computational complexity as the number of samples grows, with a decomposition algorithm. It operates on a working set of the variables, while keeping fixed variables that are less likely to change.

All the above methods are deterministic and make hard decisions. On the other hand, Bayesian learning brings the advantages of probabilistic predictions and a significant decrease in the number of basis functions. Tresp (2000) introduced the Bayesian Committee Machine that is able to handle large data sets by splitting them in subsets and training an estimator for each. These estimators are combined with appropriate weights to generate the prediction. What is noteworthy about this approach is the fact that the positions of query points are taken into account in the design of the estimator and that performance improves when multiple query points are processed simultaneously. Tipping (2001) proposed a sparse Bayesian learning approach, which produces probabilistic predictions and automatically detects nuisance parameters, and the Relevance Vector Machine that can be viewed as stochastic formulation of an SVM. A Bayesian treatment of SVM-based regression can also be found in the work of Chu et al. (2004). Its advantages include reduced computational complexity over Gaussian Process Regression (GPR), reviewed below, and robustness against outliers. Inspired by Factor Analysis Regression, Ting et al. (2006) propose a Bayesian regression algorithm that is robust to ill-conditioned data, detects relevant features and identifies input and output noise.

An approach that has attracted a lot of attention is the use of Gaussian Processes (GPs) for regression. Williams and Rasmussen (1996) observed that Bayesian analysis of neural networks is difficult due to complex prior distributions over functions resulting even from simple priors over weights. Instead, if one uses Gaussian processes as priors over the functions, then Bayesian analysis can be carried out exactly. Despite the speed up due to GPs, faster implementations were still needed for practical applications. A sparse greedy GP regression algorithm was presented by Smola and Bartlett (2001) who approximate the MAP estimate by expanding in terms of a small set of kernels. Csató and Opper (2002) described an alternative sparse representation for GP regression models. It operates in an online fashion and maintains a sparse basis which is dynamically updated as more data become available.

Lawrence et al. (1996) compared a global approach using a multi-layer perceptron with a linear local approximation model. They found that the local model performed better when the density of the input data deviated a lot from being uniform. Furthermore, the local model allowed for incremental learning and cross-validation. On the other hand, it showed poorer generalization, slower performance after training and required more memory, since all input data had to be stored. The

global model performed better in higher dimensions, where data sparsity becomes a serious problem for the local alternative. Wedge et al. (2006) bring together the advantages of global and local approaches using a hybrid network architecture that combines RBFs and sigmoid neural networks. It first identifies global features of the system before adding local details via the RBFs.

Schaal and Atkeson (1998) proposed a nonparametric, local, incremental learning approach based on receptive field weighted regression. The approach is truly local since the parameters for each model and the size and shape of each receptive field are learned independently. The provided mechanisms for the addition and pruning of local models enable incremental learning as new data points become available.

Atkeson et al. (1997) survey local weighted learning methods and identify the issues that must be taken into account. These include the selection of the distance metric, the weighting function, prediction assessment and robustness to noise. The authors argue that in certain cases no values of the parameters of a global model can provide a good approximation of the true function. In these cases, a local approximation using a simpler, even linear model, is a better approach than increasing the complexity of the global model. Along these lines, Vijayakumar and Schaal (2000) proposed locally weighted projection regression, an algorithm based on successive univariate regressions along projections of the data in directions given by the gradient of the underlying function.

We opt for a local approach and address the problem as an extension of manifold learning. Note, however, that we are not limited to functions that are strictly manifolds. Using tensor voting, we are able to reliably estimate the normal and tangent space at each sample, as described in the following section. These estimates allow us to perform nonlinear interpolation and generate outputs for unobserved inputs, even under severe noise corruption.

3. Tensor Voting in High-Dimensional Spaces

The tensor voting framework, in its preliminary version (Guy and Medioni, 1996), is an implementation of two Gestalt principles, namely proximity and good continuation, for grouping generic tokens in 2-D. The 2-D domain has always been the main focus of research in perceptual organization, beginning with the research of Köhler (1920), Wertheimer (1923) and Koffka (1935). The generalization of perceptual organization to 3-D is relatively straightforward, since salient groupings can be detected by the human visual system in 3-D based on the same principles. Guy and Medioni (1997) extended tensor voting to three dimensions. The term saliency here refers to *structural saliency*, which, according to Shashua and Ullman (1988) is the property of structures to stand out due to the configuration of local elements. That is, the local elements of the structure are not salient in isolation, but instead the arrangement of the elements is what makes the structure salient. The saliency of each element is estimated by accumulating *votes* cast from its neighbors. Tensor voting is a pairwise operation in which elements cast and collect votes in local neighborhoods. Each vote is a tensor and encodes the orientation the receiver would have according to the voter, if the voter and receiver belonged in the same structure. According to the Gestalt theory, simple figures are preferable to complex alternatives, if no evidence favors the latter. The evidence in tensor voting are the position and preferred orientation, if available, of the voter and the position of the receiver. Given this information, we show examples of votes cast by an oriented voter in 2-D in Figure 1. The voter V is a point on a horizontal curve and its normal is represented by the orange arrow. (Details on the representation can be found in Section 3.1). The other points, R_1-R_4, act as receivers. The simplest curve, the one with minimum total curvature, that passes through an oriented voter and

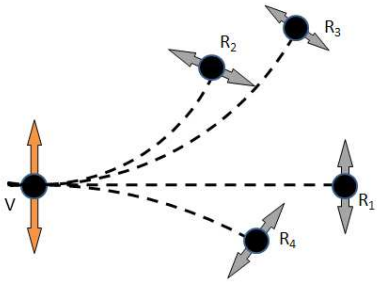
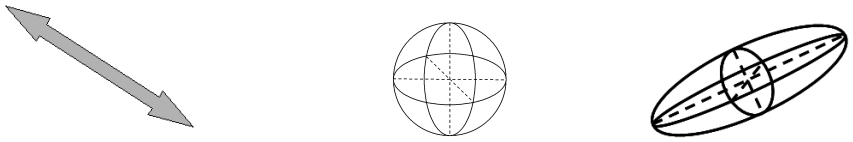


Figure 1: Illustration of tensor voting in 2-D. The voter is an oriented curve element V on a horizontal curve whose normal is represented by the orange arrow. The four receivers R_1-R_4 collect votes from V. (In practice, they would also cast votes to V and among themselves, but this is omitted here.) Each receiver is connected to V by a circular arc which is the simplest structure that can be inferred from two points, one of which is oriented. The gray votes at the receivers indicate the curve normal the receivers should have according to the voter.

a receiver is a circular arc, for which curvature is constant. Therefore, we connect the voter and receiver by a circular arc which is tangent at the voter and passes through the receiver and define the vote cast as the normal to this arc at the location of the receiver. The votes shown as gray arrows in Figure 1 represent the orientations the receivers would have according to the voter V. The magnitude of the votes decays with distance and curvature. It will be defined formally in Section 3.2. Voting from all possible types of voters, such as surface or curve elements in 3-D, can be derived from the fundamental case of a curve element voter in 2-D (Medioni et al., 2000). Tensor voting is based on strictly local computations in the neighborhoods of the inputs. The size of these neighborhoods is controlled by the only critical parameter in the framework: the scale of voting σ. The scale parameter is introduced in Eq. 3. By determining the size of the neighborhoods, scale regulates the amount of smoothness and provides a knob to the user for balancing fidelity to the data and noise reduction.

Regardless of the computational feasibility of an implementation, the same grouping principles apply to spaces with even higher dimensions. For instance, Tang et al. (2001) observed that pixel correspondences can be viewed as points in the 8-D space of free parameters of the fundamental matrix. Correct correspondences align to form a hyperplane in that space, while wrong correspondences are randomly distributed. By applying tensor voting in 8-D, Tang et al. were able to infer the dominant hyperplane and the desired parameters of the fundamental matrix. Storage and computation requirements, however, soon become prohibitively high as the dimensionality of the space increases. Even though the applicability of tensor voting as a manifold learning technique seems to have merit, the generalization of the implementation of Medioni et al. (2000) is not practical, mostly due to computational complexity and storage requirements in N dimensions. The bottleneck is the generation and storage of voting fields, the number of which is equal to the dimensionality of the space.

In this section, we describe the tensor voting framework beginning with data representation and proceeding to the voting mechanism and vote analysis. The representation and vote analysis schemes are N-D extensions of the original implementation (Medioni et al., 2000). The novelty of



(a) Oriented or *stick* tensor (b) Unoriented or *ball* tensor (c) Generic tensor

Figure 2: Examples of tensors in 3-D. The tensor on the left has only one non-zero eigenvalue and encodes a preference for an orientation parallel to the eigenvector corresponding to that eigenvalue. The eigenvalues of the tensor in the middle are all equal, and thus the tensor does not encode a preference for a particular orientation. The tensor on the right is a generic 3-D tensor.

our work is a new formulation of the voting process that is practical for spaces of dimensionality up to a few hundreds. Efficiency is considerably higher than the preliminary version of this formulation presented in Mordohai and Medioni (2005), where we focused on dimensionality estimation.

3.1 Data Representation

The representation of a token (a generic data point) is a second order, symmetric, non-negative definite tensor, which is equivalent to an $N \times N$ matrix and an ellipsoid in an N-D space. All tensors in this paper are second order, symmetric and non-negative definite, so any reference to a tensor automatically implies these properties. Three examples of tensors, in 3-D, can be seen in Figure 2. A tensor represents the structure of a manifold going through the point by encoding the normals to the manifold as eigenvectors of the tensor that correspond to non-zero eigenvalues, and the tangents as eigenvectors that correspond to zero eigenvalues. (Note that eigenvectors and vectors in general in this paper are column vectors.) For example, a point in an N-D hyperplane has one normal and $N-1$ tangents, and thus is represented by a tensor with one non-zero eigenvalue associated with an eigenvector parallel to the hyperplane's normal. The remaining $N-1$ eigenvalues are zero. A point belonging to a 2-D manifold in N-D is represented by two tangents and $N-2$ normals, and thus is represented by a tensor with two zero eigenvalues associated with the eigenvectors that span the tangent space of the manifold. The tensor also has $N-2$ non-zero, equal eigenvalues whose corresponding eigenvectors span the manifold's normal space. Two special cases of tensors are: the *stick* tensor that has only one non-zero eigenvalue and represents perfect certainty for a hyperplane normal to the eigenvector that corresponds to the non-zero eigenvalue; and the *ball* tensor that has all eigenvalues equal and non-zero which represents perfect uncertainty in orientation, or, in other words, just the presence of an unoriented point.

The tensors can be formed by the summation of the direct products ($\vec{n}\vec{n}^T$) of the eigenvectors that span the normal space of the manifold. The tensor at a point on a manifold of dimensionality d, with $\vec{n}_i$ being the unit vectors that span the normal space, can be computed as follows:

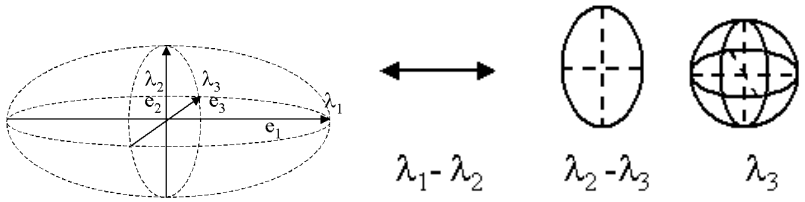
$$T = \sum_{i=1}^{d} \vec{n}_i \vec{n}_i^T . \tag{1}$$

An unoriented point can be represented by a *ball* tensor which contains all possible normals and is encoded as the $N \times N$ identity matrix. Any point on a manifold of known dimensionality and orientation can be encoded in this representation by appropriately constructed tensors, according to Eq. 1.

On the other hand, given an N-D second order, symmetric, non-negative definite tensor, the type of structure encoded in it can be inferred by examining its eigensystem. Any such tensor can be decomposed as in the following equation:

$$\begin{aligned}\mathbf{T} &= \sum_{d=1}^{N} \lambda_d \hat{e}_d \hat{e}_d^T = \\ &= (\lambda_1 - \lambda_2)\hat{e}_1\hat{e}_1^T + (\lambda_2 - \lambda_3)(\hat{e}_1\hat{e}_1^T + \hat{e}_2\hat{e}_2^T) + ... + \lambda_N(\hat{e}_1\hat{e}_1^T + \hat{e}_2\hat{e}_2^T + ... + \hat{e}_N\hat{e}_N^T) \\ &= \sum_{d=1}^{N-1} [(\lambda_d - \lambda_{d+1}) \sum_{k=1}^{d} \hat{e}_d\hat{e}_d^T] + \lambda_N(\hat{e}_1\hat{e}_1^T + ... + \hat{e}_N\hat{e}_N^T)\end{aligned} \quad (2)$$

where λ_d are the eigenvalues in descending order of magnitude and $\hat{e}_d$ are the corresponding eigenvectors. The tensor simultaneously encodes *all* possible types of structure. The confidence, or *saliency* in perceptual organization terms, of the type that has d normals is encoded in the difference $\lambda_d - \lambda_{d+1}$, or λ_N for the ball tensor. If only one of these eigenvalue differences is not zero, then the tensor encodes a single type of structure. Otherwise, more than one type can be present at the location of the tensor, each having a saliency value given by the appropriate difference between consecutive eigenvalues of λ_N. An illustration of tensor decomposition in 3-D can be seen in Figure 3.



(a) Generic 3-D tensor (b) Elementary 3-D tensors

Figure 3: Tensor decomposition in 3-D. A generic tensor can be decomposed into the stick, plate and ball components that have a normal subspace of rank one, two and three respectively.

3.2 The Voting Process

After the inputs have been encoded with tensors, the information they contain is propagated to their neighbors via a voting operation. Given a tensor at A and a tensor at B, the vote the token at A (the *voter*) casts to B (the *receiver*) has the orientation the receiver would have, if both the voter and receiver belong to the same structure. The magnitude of the vote is a function of the confidence we have that the voter and receiver indeed belong to the same structure.

3.2.1 STICK VOTING

We first examine the case of a voter associated with a *stick* tensor, that is the normal space is a single vector in N-D. We claim that, in the absence of other information, the arc of the *osculating circle* (the circle that shares the same normal as a curve at the given point) at A that goes through B is the most likely smooth path between A and B, since it minimizes total curvature. The center of the circle is denoted by C in Figure 4(a). (For visualization purposes, the illustrations are for the 2-D and 3-D cases.) In case of straight continuation from A to B, the osculating circle degenerates to a straight line. Similar use of circular arcs can also be found in Parent and Zucker (1989), Saund (1992), Sarkar and Boyer (1994) and Yen and Finkel (1998). The vote is also a stick tensor and is generated as described in Section 3 according to the following equation:

$$\begin{aligned}
\mathbf{S}_{vote}(s,\theta,\kappa) &= e^{-(\frac{s^2+c\kappa^2}{\sigma^2})} \begin{bmatrix} -\sin(2\theta) \\ \cos(2\theta) \end{bmatrix} [-\sin(2\theta) \;\; \cos(2\theta)], \qquad (3)\\
\theta &= \arcsin(\frac{\vec{v}^T \hat{e}_1}{\|\vec{v}\|}),\\
s &= \frac{\theta\|\vec{v}\|}{\sin(\theta)},\\
\kappa &= \frac{2\sin(\theta)}{\|\vec{v}\|}.
\end{aligned}$$

In the above equation, s is the length of the arc between the voter and receiver, and κ is its curvature (which can be computed from the radius AC of the osculating circle in Figure 4(a)), σ is the scale of voting, and c is a constant, which controls the degree of decay with curvature. The constant c is a function of the scale and is optimized to make the extension of two orthogonal line segments to from a right angle equally likely to the completion of the contour with a rounded corner (Guy and Medioni, 1996). Its value is given by: $c = \frac{-16log(0.1)\times(\sigma-1)}{\pi^2}$. The scale σ essentially controls the range within which tokens can influence other tokens. It can also be viewed as a measure of smoothness that regulates the inevitable trade-off between over-smoothing and over-fitting. Small values preserve details better, but are more vulnerable to noise and over-fitting. Large values produce smoother approximations that are more robust to noise. As shown in Section 7, the results are very stable with respect to the scale. Note that σ is the only free parameter in the framework.

The vote as defined above is on the plane defined by A, B and the normal at A. Regardless of the dimensionality of the space, stick vote generation always takes place in a 2-D subspace defined by the position of the voter and the receiver and the orientation of the voter. (This explains why Eq. 3 is defined in a 2-D space.) Stick vote computation is identical in any space between 2 and N dimensions. After the vote has been computed, it has to be transformed to the N-D space and aligned to the voter by a rotation and translation. For simplicity, we also use the notation:

$$\mathbf{S}_{vote}(A,B,\vec{n}) = \mathbf{S}(s(A,B,\vec{n}),\theta(A,B,\vec{n}),\kappa(A,B,\vec{n})) \qquad (4)$$

to denote the stick vote from A to B with $\vec{n}$ being the normal at A. $s(A,B,\vec{n})$, $\theta(A,B,\vec{n})$ and $\kappa(A,B,\vec{n})$ are the resulting values of the parameters of Eq. 3 given A,B and $\vec{n}$.

According to the Gestalt principles we wish to enforce, the magnitude of the vote should be a function of proximity and smooth continuation. Thus the influence from a point to another attenuates with distance, to minimize interference from unrelated points; and curvature, to favor straight

continuation over curved alternatives. Moreover, no votes are cast if the receiver is at an angle larger than 45° with respect to the tangent of the osculating circle at the voter. Similar restrictions on regions of influence also appear in Heitger and von der Heydt (1993), Yen and Finkel (1998) and Li (1998) to prevent high-curvature connections without support from the data. Votes corresponding to such connections would have been very weak regardless of the restriction since their magnitude is attenuated due to curvature. The saliency decay function (Gaussian) of Eq. 3 has infinite support, but for practical purposes the field is truncated so that negligible votes do not have to be computed. For all experiments shown here, we limited voting neighborhoods to the extent in which the magnitude of a vote is more than 3% of the magnitude of the voter. Both truncation beyond 45° and truncation beyond a certain distance are not critical choices, but are made to eliminate the computation of insignificant votes.

3.2.2 *N*-D FORMULATION OF TENSOR VOTING

We have shown that stick vote computation is identical up to a simple transformation from 2-D to *N*-D. Now we turn our attention to votes generated by voters that are not stick tensors. In the original formulation (Medioni et al., 2000) these votes can be computed by integrating the votes cast by a rotating stick tensor that spans the normal space of the voting tensor. Since the resulting integral has no closed form solution, the integration is approximated numerically by taking sample positions of the rotating stick tensor and adding the votes it generates at each point within the voting neighborhood. As a result, votes that cover the voting neighborhood are pre-computed and stored in voting fields. The advantage of this scheme is that all votes are generated based on the stick voting field. Its computational complexity, however, makes its application in high-dimensional spaces prohibitive. Voting fields are used as look-up tables to retrieve votes via interpolation between the pre-computed samples. For instance, a voting field in 10-D with k samples per axis, requires storage for k^{10} 10×10 tensors, which need to be computed via numerical integration over 10 variables. Thus, the use of pre-computed voting fields becomes impractical as dimensionality grows. At the same time, the probability of using a pre-computed vote decreases.

Here, we present a simplified vote generation scheme that allows the direct computation of votes from arbitrary tensors in arbitrary dimensions. Storage requirements are limited to storing the tensors at each sample, since explicit voting fields are not used any more. The advantage of the novel vote generation scheme is that it does not require integration. As in the original formulation, the eigenstructure of the vote represents the normal and tangent spaces that the receiver would have, if the voter and receiver belong in the same smooth structure.

3.2.3 BALL VOTING

For the generation of ball votes, we propose the following direct computation. It is based on the observation that the vote generated by a ball voter propagates the voter's preference for a straight line that connects it to the receiver (Figure 4(b)). The straight line is the simplest and smoothest continuation from a point to another point in the absence of other information. Thus, the vote generated by a ball voter is a tensor that spans the $(N-1)$-D normal space of the line and has one zero eigenvalue associated with the eigenvector that is parallel to the line. Its magnitude is a function of only the distance between the two points, since curvature is zero. Taking these observations into account, the ball vote can be constructed by subtracting the direct product of the unit vector in the direction from the voter to the receiver from a full rank tensor with equal eigenvalues (i.e., the

identity matrix). The resulting tensor is attenuated by the same Gaussian weight according to the distance between the voter and the receiver.

$$\mathbf{B}_{vote}(s) = e^{-\left(\frac{s^2}{\sigma^2}\right)} \left(I - \frac{\vec{v}\vec{v}^T}{\|\vec{v}^T\vec{v}\|} \right) \tag{5}$$

where $\vec{v}$ is a unit vector parallel to the line connecting the voter and the receiver and I is the N-D identity matrix. In this case, $s = |\vec{v}|$ and we omit θ and κ since they do not affect the computation.

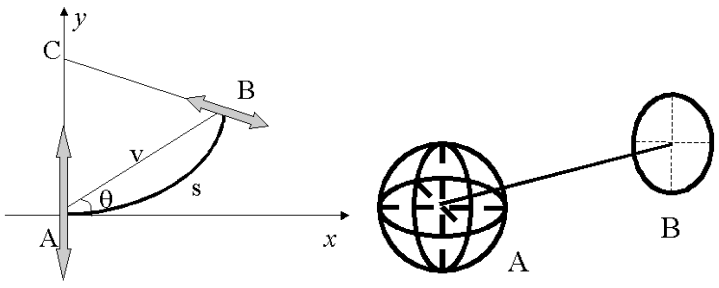
Along the lines of Equation 4, we define a simpler notation:

$$\mathbf{B}_{vote}(A,B) = \mathbf{B}_{vote}(s(A,B)) \tag{6}$$

where $s(A,B) = |\vec{v}|$.

3.2.4 VOTING BY ELEMENTARY TENSORS

To complete the description of vote generation, we need to describe the case of a tensor that has d equal eigenvalues, where d is not equal to 1 or N. (An example of such a tensor would be a curve element in 3-D, which has a rank-two normal subspace and a rank-one tangent subspace.) The description in this section also applies to ball and stick tensors, but we use the above direct computations, which are faster. Let $\vec{v}$ be the vector connecting the voting and receiving points. It can be decomposed into $\vec{v}_t$ and $\vec{v}_n$ in the tangent and normal spaces of the voter respectively. The new vote generation process is based on the observation that curvature in Eq. 3 is not a factor when θ is zero, or, in other words, if the voting stick is orthogonal to $\vec{v}_n$. We can exploit this by defining a new basis for the normal space of the voter that includes $\vec{v}_n$. The new basis is computed using the Gramm-Schmidt procedure. The vote is then constructed as the tensor addition of the votes cast by stick tensors parallel to the new basis vectors. Among those votes, only the one generated by the stick tensor parallel to $\vec{v}_n$ is not parallel to the normal space of the voter and curvature has to be considered. All other votes are a function of the length of $\vec{v}_t$ only. See Figure 5 for an illustration in 3-D. Analytically, the vote is computed as the summation of d stick votes cast by the new basis of the normal space. Let N_S denote the normal space of the voter and let $\vec{b}_i, i \in [1,d]$ be a basis for it with $\vec{b}_1$ being parallel to $\vec{v}_n$. If $S_{vote}(A,B,\vec{b})$ is the function that generates the stick vote from a



(a) Stick voting (b) Ball voting

Figure 4: Vote generation for a stick and a ball voter. The votes are functions of the position of the voter A and receiver B and the tensor of the voter.

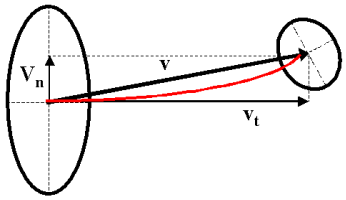


Figure 5: Vote generation for generic tensors. The voter here is a tensor with a 2-D normal subspace in 3-D. The vector connecting the voter and receiver is decomposed into $\vec{v}_n$ and $\vec{v}_t$ that lie in the normal and tangent space of the voter. A new basis that includes $\vec{v}_n$ is defined for the normal space and each basis component casts a stick vote. Only the vote generated by the orientation parallel to $\vec{v}_n$ is not parallel to the normal space. Tensor addition of the stick votes produces the combined vote.

unit stick tensor at A parallel to $\vec{b}$ to the receiver B, then the vote from a generic tensor with normal space N is given by:

$$\mathbf{V}_{vote}(A,B,\mathbf{T}_{e,d}) = S_{vote}(A,B,\vec{b}_1) + \sum_{i\in[2,d]} S_{vote}(A,B,\vec{b}_i). \tag{7}$$

In the above equation, $\mathbf{T}_{e,d}$ denotes the elementary voting tensor with d equal non-zero eigenvalues. On the right-hand side, all the terms are pure stick tensors parallel to the voters, except the first one which is affected by the curvature of the path connecting the voter and receiver. Therefore, the computation of the last $d-1$ terms is equivalent to applying the Gaussian weight to the voting sticks and adding them at the position of the receiver. Only one vote requires a full computation of orientation and magnitude. This makes the proposed scheme computationally inexpensive.

3.2.5 THE VOTING PROCESS

During the voting process (Algorithm 1), each point casts a vote to all its neighbors within the voting neighborhood. If the voters are not pure elementary tensors, that is if more than one saliency value is non-zero, they are decomposed before voting according to Eq. 2. Then, each component votes separately and the vote is weighted by $\lambda_d - \lambda_{d+1}$, except the ball component whose vote is weighted by λ_N. Besides the voting tensor $\mathbf{T}$, points also have a receiving tensor $\mathbf{R}$ that acts as vote accumulator. Votes are accumulated at each point by tensor addition, which is equivalent to matrix addition.

3.3 Vote Analysis

Vote analysis takes place after voting to determine the most likely type of structure and the orientation at each point. There are $N+1$ types of structure in an N-D space ranging from 0-D points to N-D hypervolumes.

The eigensystem of the receiving tensor $\mathbf{R}$ is computed once at the end of the voting process and the tensor is decomposed as in Eq. 2. The estimate of local intrinsic dimensionality is given by the maximum gap in the eigenvalues. Quantitative results on dimensionality estimation are presented in Section 4. In general, if the maximum eigenvalue gap is $\lambda_d - \lambda_{d+1}$, the estimated local intrinsic

Algorithm 1 The Voting Process

1. Initialization
Read M input points P_i and initial conditions, if available.
for all $i \in [1,M]$ **do**
  Initialize $\mathbf{T}_i$ according to initial conditions or set equal to the identity I.
  Compute $\mathbf{T}_i$'s eigensystem $(\lambda_d^{(i)}, \hat{e}_d^{(i)})$.
  Set vote accumulator $\mathbf{R}_i \leftarrow 0$
end for
Initialize Approximate Nearest Neighbor (ANN) k-d tree (Arya et al., 1998) for fast neighbor retrieval.

2. Voting
for all $i \in [1,M]$ **do**
  for all P_j in P_i's neighborhood **do**
    if $\lambda_1 - \lambda_2 > 0$ **then**
      Compute stick vote $\mathbf{S}_{vote}(P_i, P_j, \hat{e}_1^{(i)})$ from P_i to P_j according to Eq. 4.
    end if
    if $\lambda_N > 0$ **then**
      Compute ball vote $\mathbf{B}_{vote}(P_i, P_j)$ from P_i to P_j according to Eq. 6.
    end if
    for $d = 2$ to $N-1$ **do**
      if $\lambda_d - \lambda_{d+1} > 0$ **then**
        Compute vote $\mathbf{V}_{vote}(P_i, P_j, \mathbf{T}_{e,d}^{(i)})$ according to Eq. 7.
      end if
    end for
    Add votes to P_j's accumulator

$$\mathbf{R}_j \leftarrow \mathbf{R}_j + (\lambda_1 - \lambda_2)\mathbf{S}_{vote}(P_i, P_j, \hat{e}_1) + (\lambda_N)\mathbf{B}_{vote}(P_i, P_j) + \sum_{d \in [2, N-1]} (\lambda_d - \lambda_{d+1})\mathbf{V}_{vote}(P_i, P_j, \mathbf{T}_{e,d}^{(i)})$$

  end for
end for

3. Vote Analysis
for all $i \in [1,M]$ **do**
  Compute eigensystem of $\mathbf{R}_i$ (Eq. 2) to determine dimensionality and orientation.
  $\mathbf{T}_i \leftarrow \mathbf{R}_i$
end for

dimensionality is $N-d$, and the manifold has d normals and $N-d$ tangents. Moreover, the first d eigenvectors that correspond to the largest eigenvalues are the normals to the manifold, and the remaining eigenvectors are the tangents. Outliers can be detected since all their eigenvalues are

small and no preferred structure type emerges. This happens because they are more isolated than inliers, thus they do not receive votes that consistently support any salient structure. Our past and current research has demonstrated that tensor voting is very robust against outliers.

This vote accumulation and analysis method does not optimize any explicit objective function, especially not a global one. Dimensionality emerges from the accumulation of votes, but it is not a equal to the average, nor the median, nor the majority of the dimensionalities of the voters. For instance, the accumulation of votes from elements of two or more intersecting curves in 2-D results in a rank-two normal space at the junction. If one restricts the analysis to the estimates of orientation, tensor voting can be viewed as a method for maximizing an objective at each point. The weighted (tensor) sum of all votes received is up to a constant equivalent to the weighted mean in the space of symmetric, non-negative definite, second-order tensors. This can be thought of as the tensor that maximizes the consensus among the incoming votes. In that sense, assuming dimensionality is provided by some other process, the estimated orientation at each point is the maximum likelihood estimate given the incoming votes. It should be pointed out here, that the sum is used for all subsequent computations, since the magnitude of the eigenvalues and the of the gaps between them are measures of saliency.

In all subsequent sections, the eigensystem of the accumulator tensor is used as the voter during subsequent processing steps described in the following sections.

Figure 6: 20,000 points sampled from the "Swiss Roll" data set in 3-D.

4. Dimensionality Estimation

In this section, we present experimental results in dimensionality estimation. According to Section 3.3, the intrinsic dimensionality at each point can be found as the maximum gap in the eigenvalues of the tensor after votes from its neighboring points have been collected. All inputs consist of unoriented points since no orientation information is provided and are encoded as ball tensors.

4.1 Swiss Roll

The first experiment is on the "Swiss Roll" data set, which is frequently used in the literature and is available at `http://isomap.stanford.edu/`. It contains $20,000$ points on a 2-D manifold in 3-D (Figure 6). We sample two random variables from independent uniform distributions to generate Swiss rolls with as many as $20,000$ points, as in Tenenbaum et al. (2000), and as few as $1,250$ points. We present quantitative results for 10 different instances of $1,250$, $5,000$ and $20,000$ points in Table 1 as a function of σ. The first column for each set of results shows the percentage of points with correct dimensionality estimates, that is for which $\lambda_1 - \lambda_2$ is the maximal eigenvalue gap. The

Points	1250			5000			20000		
σ	DE	NN	Time	DE	NN	Time	DE	NN	Time
1	0.5%	1.6	0.2	4.3%	3.6	0.4	22.3%	11.2	3.8
2	4.2%	3.7	0.2	22.3%	11.1	0.8	67.9%	40.8	5.4
4	21.5%	11.0	0.2	70.0%	40.7	0.8	98.0%	156.3	12.3
8	64.5%	38.3	0.2	96.9%	148.9	2.6	99.9%	587.2	36.9
12	87.4%	83.4	0.4	99.3%	324.7	5.0	99.9%	1285.4	77.4
16	94.3%	182.4	0.7	99.5%	715.7	10.1	99.8%	2838.0	164.8
20	94.9%	302.6	1.0	99.2%	1179.4	16.5	99.5%	4700.9	268.7
30	91.0%	703.6	2.3	97.0%	2750.8	37.4	97.6%	10966.5	615.2
40	83.6%	1071.7	3.6	86.7%	4271.5	57.7	87.7%	16998.2	947.3

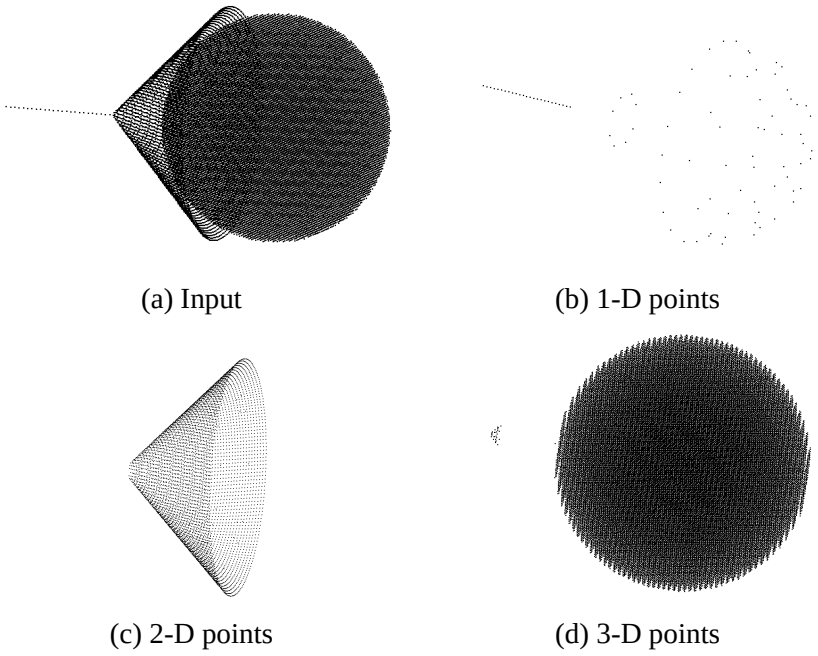
Table 1: Rate of correct dimensionality estimation (DE), average number of neighbors per point and execution times (in seconds) as functions of σ and the number of samples for the "Swiss Roll" data set. All experiments have been repeated 10 times on random samplings of the Swiss Roll function. Note that the range of scales includes extreme values as evidenced by the very high and very low numbers of neighbors in several cases.

second column shows the average number of nearest neighbors included in the voting neighborhood of each point. The third column shows processing times of a single-threaded C++ implementation running on an Intell Pentium 4 processor at $2.50GHz$. We have also repeated the experiment on 10 instances of 500 points from the Swiss Roll using the same values of the scale. Meaningful results are obtained for $\sigma > 8$. The peak of correct dimensionality estimation is at 80.3% for $\sigma = 20$.

A conclusion that can be drawn from Table 1 is that the accuracy is high and stable for a large range of values of σ, as long as a few neighbors are included in each neighborhood. The majority of neighborhoods being empty is an indication of inappropriate scale selection. Performance degrades as scale increases and the neighborhoods become too large to capture the curvature of the manifold. This robustness to large variations in parameter selection are due to the weighting of the votes according to Eqs. 3, 5 and 7 and alleviates the need for extensive parameter tuning.

4.2 Structures with Varying Dimensionality

The second data set is in 4-D and contains points sampled from three structures: a line, a 2-D cone and a 3-D hyper-sphere. The hyper-sphere is a structure with three degrees of freedom. It cannot be unfolded unless we remove a small part from it. Figure 7(a) shows the first three dimensions of the data. The data set contains a total $135,864$ points, which are encoded as ball tensors. Tensor voting is performed with $\sigma = 200$. Figures 7(b-d) show the points classified according to their dimensionality. Methods based on dimensionality reduction or methods that estimate a single dimensionality estimate for the data set would fail for this data set because of the presence of structures with different dimensionalities and because the hyper-sphere cannot be unfolded.



(a) Input (b) 1-D points

(c) 2-D points (d) 3-D points

Figure 7: Data of varying dimensionality in 4-D. (The first three dimensions of the input and the classified points are shown.) Note that the hyper-sphere is empty in 4-D, but appears as a full sphere when visualized in 3-D.

4.3 Data in High Dimensions

The data sets for this experiment were generated by sampling a few thousand points from a low-dimensional space (3- or 4-D) and mapping them to a medium dimensional space (14- to 16-D) using linear and quadratic functions. The generated points were then rotated and embedded in a 50- to 150-D space. Outliers drawn from a uniform distribution inside the bounding box of the data were added to the data set. The percentage of correct point-wise dimensionality estimates after tensor voting can be seen in Table 2.

Intrinsic Dimensionality	Linear Mappings	Quadratic Mappings	Space Dimensions	Dimensionality Estimation (%)
4	10	6	50	93.6
3	8	6	100	97.4
4	10	6	100	93.9
3	8	6	150	97.3

Table 2: Rate of correct dimensionality estimation for high dimensional data.

5. Manifold Learning

In this section we show quantitative results on estimating manifold orientation for various data sets.

Points	1250	5000	20000
σ	Orientation Error	Orientation Error	Orientation Error
1	47.2	28.1	3.1
2	28.7	3.5	**0.4**
4	4.9	**0.9**	0.5
8	**2.0**	1.3	1.1
12	2.5	2.0	1.9
16	3.5	3.1	3.0
20	5.4	4.8	4.7
30	16.9	15.0	14.9
40	28.3	26.2	25.9

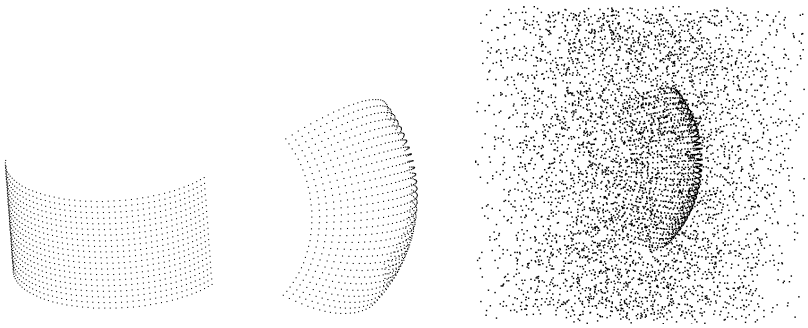
Table 3: Error (in degrees) in surface normal orientation estimation as a function of σ and the number of samples for the "Swiss Roll" data set. The error reported is the (unsigned) angle between the eigenvector corresponding to the largest eigenvalue of the estimated tensor at each point and the ground truth surface normal. See also Table 1 for processing times and the average number of points in each neighborhood.

5.1 Swiss Roll

We begin this section by completing the presentation of our experiments on the Swiss Roll data sets described in the previous section. Here we show the accuracy of normal estimation, regardless of whether the dimensionality was estimated correctly, for the experiments of Table 1. Table 3 is a complement to Table 1, which contains information on the average number of points in the voting neighborhoods and processing time that is not repeated here. The error reported is the (unsigned) angle between the eigenvector corresponding to the largest eigenvalue of the estimated tensor at each point and the ground truth surface normal. These results are over 10 different samplings for each number of points reported in the table.

For comparison, we also estimated the orientation at each point of the Swiss Roll using local PCA computed on the point's k nearest neighbors. We performed an exhaustive search over k, but only report the best results here. As for tensor voting, orientation accuracy was measured on 10 instances of each data set. The lowest errors for $1,250$, $5,000$ and $20,000$ points are 2.51°, 1.16° and 0.56° for values of k equal to 9, 10 and 13 respectively. These errors are approximately 20% larger than the lowest errors achieved by tensor voting, which are shown in bold in Table 3. It should be noted, that, unlike tensor voting, local PCA cannot be used to refine these estimates or take advantage of existing orientation estimates that may be available at the inputs.

We observe that accuracy using tensor voting is very high for a large range of scales and improves, as expected, with higher data density. Random values (around 45°) result when the neighborhood does not contain enough points for normal estimation. See Mitra et al. (2004) and Lalonde et al. (2005) for an analysis of the 3-D case based on the *Gershgorin Circle Theorem* that provides



(a) Cylinder (b) Sphere (c) Noisy sphere

Figure 8: Data sets used in Sections 5 and 6.

bounds for the eigenvalues of square matrices. The authors show how noise and curvature affect the estimation of curve and surface normals under some assumptions about the distribution of the points. The conclusions are that for large neighborhood sizes, errors caused by curvature dominate, while for small neighborhood sizes, errors due to noise dominate. (There is no noise in the data for this experiment.)

5.2 Spherical and Cylindrical Sections

Here, we present quantitative results on simple data sets in 3-D for which ground truth can be analytically computed. In Section 6, we process the same data with state of the art manifold learning algorithms and compare their results against ours. The two data sets are a section of a cylinder and a section of a sphere shown in Figure 8. The cylindrical section spans 150° and consists of 1000 points. The spherical section spans $90^\circ \times 90^\circ$ and consists of 900 points. Both are approximately uniformly sampled. The points are represented by ball tensors, assuming no information about their orientation. In the first part of the experiment, we compute local dimensionality and normal orientation as a function of scale. The results are presented in Tables 4 and 5. The results show that if the scale is not too small, dimensionality estimation is very reliable. For all scales the orientation errors are below 0.4^o.

The same experiments were performed for the spherical section in the presence of outliers. Quantitative results are shown in the following tables for a number of outliers that ranges from 900 (equal to the inliers) to 5000. The latter data set is shown in Figure 8(c). The outliers are drawn from a uniform distribution inside an extended bounding box of the data. Note that performance was evaluated only on the points that belong to the sphere and not the outliers. Larger values of the scale prove to be more robust to noise, as expected. The smallest values of the scale result in voting neighborhoods that include less than 10 points, which are insufficient. Taking this into account, performance is still good even with wrong parameter selection. Also note that one could reject the outliers by thresholding, since they have smaller eigenvalues than the inliers, and perform tensor voting again to obtain even better estimates of structure and dimensionality. Even a single pass of tensor voting, however, turns out to be very effective, especially considering that no other method can handle such a large number of outliers. Foregoing the low-dimensional embedding is a main reason that allows our method to perform well in the presence of noise, since embedding random outliers in a low-dimensional space would make their influence more detrimental. This is

σ	Average Neighbors	Orientation Error (°)	Dimensionality Estimation (%)
10	5	0.06	4
20	9	0.07	90
30	9	0.08	90
40	12	0.09	90
50	20	0.10	100
60	20	0.11	100
70	23	0.12	100
80	25	0.12	100
90	30	0.13	100
100	34	0.14	100

Table 4: Results on the cylinder data set. Shown in the first column is σ, in the second is the average number of neighbors that cast votes to each point, in the third the average error in degrees of the estimated normals, and in the fourth the accuracy of dimensionality estimation.

σ	Average Neighbors	Orientation Error (°)	Dimensionality Estimation (%)
10	5	0.20	44
20	9	0.23	65
30	11	0.24	93
40	20	0.26	94
50	21	0.27	94
60	23	0.29	94
70	26	0.31	94
80	32	0.34	94
90	36	0.36	94
100	39	0.38	97

Table 5: Results on the sphere data set. The columns are the same as in Table 4.

due to the structure imposed to them by the mapping, which makes the outliers less random, and due to the increase in their density in the low-dimensional space compared to that in the original high-dimensional space.

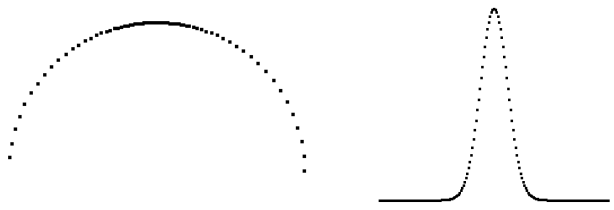
5.3 Data with Non-uniform Density

We also conducted two experiments on functions proposed in Wang et al. (2005). The key difficulty with these functions is the non-uniform density of the data. In the first example we attempt to estimate the tangent at the samples of:

$$x_i = [\cos(t_i) \;\; \sin(t_i)]^T \qquad t_i \in [0,\pi],\; t_{i+1} - t_i = 0.1(0.001 + |\cos(t_i)|) \tag{8}$$

Outliers	900		3000		5000	
σ	OE	DE	OE	DE	OE	DE
10	1.15	44	3.68	41	6.04	39
20	0.93	65	2.95	52	4.73	59
30	0.88	92	2.63	88	4.15	85
40	0.88	93	2.49	90	3.85	88
50	0.90	93	2.41	92	3.63	91
60	0.93	94	2.38	93	3.50	93
70	0.97	94	2.38	93	3.43	93
80	1.00	94	2.38	94	3.38	94
90	1.04	95	2.38	95	3.34	94
100	1.07	97	2.39	95	3.31	95

Table 6: Results on the sphere data set contaminated by noise. OE: error in normal estimation in degrees, DE: percentage of correct dimensionality estimation.



(a) Samples from Eq. 8 (b) Samples from Eq. 9

Figure 9: Input data for the two experiments proposed by Wang et al. (2005).

where the distance between consecutive samples is far from uniform. See Figure 9(a) for the inputs and the second column of Table 7 for quantitative results on tangent estimation for 152 points as a function of scale.

In the second example, which is also taken from Wang et al. (2005), points are uniformly sampled on the t-axis from the [-6, 6] interval. The output is produced by the following function:

$$x_i = [t_i \;\; 10e^{-t_i^2}]. \tag{9}$$

The points, as can be seen in Figure 9(b), are not uniformly spaced. The quantitative results on tangent estimation accuracy for 180 and 360 samples from the same interval are reported in the last two columns of Table 7. Naturally, as the sampling becomes denser, the quality of the approximation improves. What should be emphasized here is the stability of the results as a function of σ. Even with as few as 5 or 6 neighbors included in the voting neighborhood, the tangent at each point is estimated quite accurately.

σ	Eq. 8 152 points	Eq. 9 180 points	Eq. 9 360 points
10	0.60	4.52	2.45
20	0.32	3.37	1.89
30	0.36	2.92	1.61
40	0.40	2.68	1.43
50	0.44	2.48	1.22
60	0.48	2.48	1.08
70	0.51	2.18	0.95
80	0.54	2.18	0.83
90	0.58	2.02	0.68
100	0.61	2.03	0.57

Table 7: Error in degrees for tangent estimation for the functions of Eq. 8 and Eq. 9.

6. Geodesic Distances and Nonlinear Interpolation

In this section, we present an algorithm that can interpolate, and thus produce new points, on the manifold and is also able to evaluate geodesic distances between points. Both of these capabilities are useful tools for many applications. The key concept is that the intrinsic distance between any two points on a manifold can be approximated by taking small steps on the manifold, collecting votes, estimating the local tangent space and advancing on it until the destination is reached. Such processes have been reported in Mordohai (2005), Dollár et al. (2007a) and Dollár et al. (2007b).

Processing begins by learning the manifold structure, as in the previous section, usually starting from unoriented points that are represented by ball tensors. Then, we select a starting point that has to be on the manifold and a target point or a desired direction from the starting point. At each step, we can project the desired direction on the tangent space of the current point and create a new point at a small distance. The tangent space of the new point is computed by collecting votes from the neighboring points, as in regular tensor voting. Note that the tensors used here are no longer balls, but the ones resulting from the previous pass of tensor voting, according to Algorithm 1, step 3. The desired direction is then projected on the tangent space of the new point and so forth until the destination is reached. The process is illustrated in Figure 10, where we start from point A and wish to reach B. We project $\vec{t}$, the vector from A to B, on the estimated tangent space of A and obtain its projection $\vec{p}$. Then, we take a small step along $\vec{p}$ to point A_1, on which we collect votes to obtain an estimate of its tangent space. The desired direction is then projected on the tangent space of each new point until the destination is reached within ε. The geodesic distance between A and B is approximated by measuring the length of the path. In the process, we have also generated a number of new points on the manifold, which may be a desirable by-product for some applications.

There are certain configurations that cannot be handled by the algorithm described above without additional precautions. One such configuration is when the source or destination point or both are in deep concavities which attract the desired direction if the step from A to A_1 is not large enough to move the path outside the concave region. A multi-scale implementation of the above scheme can overcome this problem. A few intermediate points can be marked using a large value for the step and then used as intermediate destinations with a finer step size. Under this scheme, the path

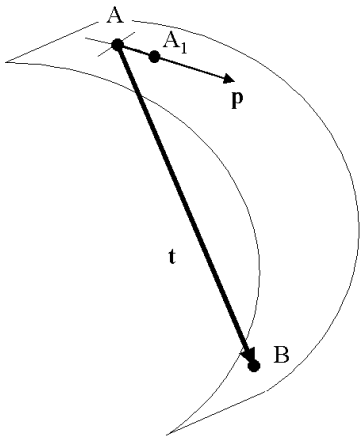


Figure 10: Nonlinear interpolation on the tangent space of a manifold.

converges to the destination and the geodesic distance is approximated accurately using a small step size. A second failure mode of the simple algorithm is for cases where the desired direction may vanish. This may occur in a manifold such as the "Swiss Roll" (Figure 6) if the destination lies on the normal space of the current point. Adding memory or inertia to the system when the desired direction vanishes, effectively addresses this situation. It should be noted that our algorithm does not handle holes and boundaries properly at its current stage of development.

6.1 Comparison with State-of-the-Art algorithms

The first experiment on manifold distance estimation is a quantitative evaluation against some of the most widely used algorithms of the literature. For the results reported in Table 8, we learn the local structure of the cylinder and sphere manifolds of the previous section using tensor voting. We also compute embeddings using LLE (Roweis and Saul, 2000), Isomap (Tenenbaum et al., 2000), Laplacian eigenmaps (Belkin and Niyogi, 2003), HLLE (Donoho and Grimes, 2003) and SDE (Weinberger and Saul, 2004). Matlab implementations for these methods can be downloaded from the following internet locations.

- LLE from `http://www.cs.toronto.edu/~roweis/lle/code.html`
- Isomap from `http://isomap.stanford.edu/`
- Laplacian Eigenmaps from `http://people.cs.uchicago.edu/~misha/ManifoldLearning/index.html`
- HLLE from `http://basis.stanford.edu/HLLE` and
- SDE from `http://www.seas.upenn.edu/~kilianw/sde/download.htm`.

We are grateful to the authors for making the code for their methods available to the community. We have also made our software publicly available at:
`http://iris.usc.edu/~medioni/download/download.htm`.

The experiment is performed as follows. We randomly select 5000 pairs of points on each manifold and attempt to measure the geodesic distance between the points of each pair in the input

space using tensor voting and in the embedding space using the other five methods. The estimated distances are compared to the ground truth: $r\Delta\theta$ for the sphere and $\sqrt{(r\Delta\theta)^2+(\Delta z)^2}$ for the cylinder. Among the above approaches, only Isomap and SDE produce isometric embeddings, and only Isomap preserves the absolute distances between the input and the embedding space. To make the evaluation fair, we compute a uniform scale that minimizes the error between the computed distances and the ground truth for all methods, except Isomap for which it is not necessary. Thus, perfect distance ratios would be awarded a perfect rating in the evaluation, even if the absolute magnitudes of the distances are meaningless in the embedding space. For all the algorithms, we tried a wide range for the number of neighbors, K. In some cases, we were not able to produce good embeddings of the data for any value of K. This occurred more frequently for the cylinder, probably due to its data density not being perfectly uniform. Errors above 20% indicate very poor performance, which is also confirmed by visual inspection of the embeddings.

Even though among the other approaches only Isomap and SDE produce isometric embeddings, while the rest produce embeddings that only preserve local structure, we think that the evaluation of the quality of manifold learning based on the computation of pairwise distances is a fair measure for the performance of all algorithms, since high quality manifold learning should minimize distortions. The distances on which we evaluate the different algorithms are both large and small, with the latter measuring the presence of local distortions. Quantitative results, in the form of the average absolute difference between the estimated and the ground truth distances as a percentage of the latter, are presented in Tables 8-10, along with the parameter that achieves the best performance for each method. In the case of tensor voting, the same scale was used for both learning the manifold and computing distances.

We also apply our method in the presence of 900, 3000 and 5000 outliers, while the inliers for the sphere and the cylinder data sets are 900 and 1000 respectively. The outliers are generated according to a uniform distribution. The error rates using tensor voting for the sphere are 0.39%, 0.47% and 0.53% respectively. The rates for the cylinder are 0.77%, 1.17% and 1.22%. Compared with the noise free case, these results demonstrate that our approach degrades slowly in the presence of outliers. The best performance achieved by any other method is 3.54% on the sphere data set with 900 outliers by Isomap. Complete results are shown in Table 9. In many cases, we were unable to achieve useful embeddings for data sets with outliers. We were not able to perform this experiment

Data Set	Sphere		Cylinder	
	K	Err(%)	K	Err(%)
LLE	18	5.08	6	26.52
Isomap	6	1.98	30	0.35
Laplacian Eigenmaps	16	11.03	10	29.36
HLLE	12	3.89	40	26.81
SDE	2	5.14	6	25.57
TV (σ)	60	0.34	50	0.62

Table 8: Error rates in distance measurement between pairs of points on the manifolds. The best result of each method is reported along with the number of neighbors used for the embedding (K), or the scale σ in the case of tensor voting (TV).

Data Set	Sphere 900	outliers	Cylinder 900	outliers
	K	Err(%)	K	Err(%)
LLE	40	60.74	6	15.40
Isomap	18	3.54	14	11.41
Laplacian Eigenmaps	6	13.97	14	27.98
HLLE	30	8.73	30	23.67
SDE		N/A		N/A
TV (σ)	70	0.39	100	0.77

Table 9: Error rates in distance measurement between pairs of points on the manifolds under outlier corruption. The best result of each method is reported along with the number of neighbors used for the embedding (K), or the scale σ in the case of tensor voting (TV). Note that HLLE fails to compute an embedding for small values of K, while SDE fails at both examples for all choices of K.

Data Set	σ	Error rate
Sphere (3000 outliers)	80	0.47
Sphere (5000 outliers)	100	0.53
Cylinder (3000 outliers)	100	1.17
Cylinder (5000 outliers)	100	1.22

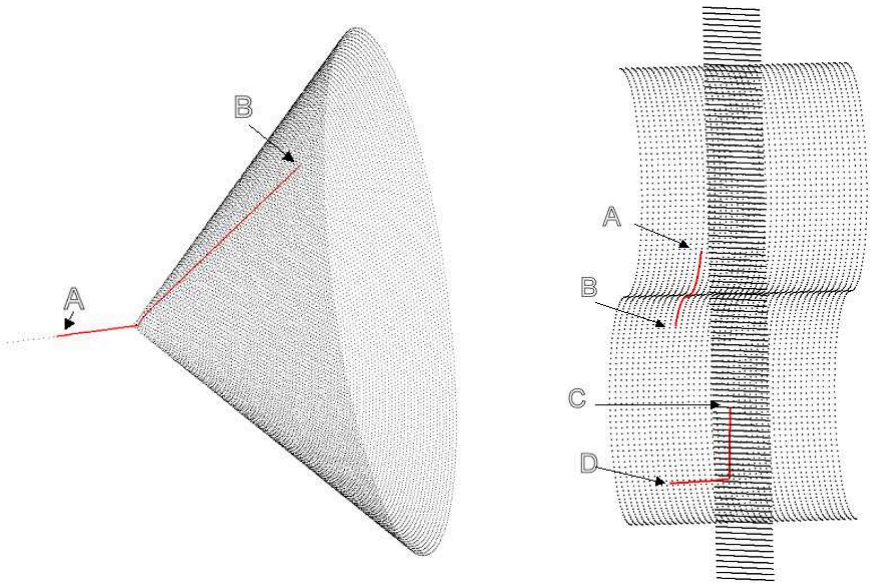
Table 10: Error rates for our approach for the experiment of Section 6.1 in the presence of 3000 and 5000 outliers.

in the presence of more than 3000 outliers with any graph-based method, probably because the graph structure is severely corrupted by the outliers.

6.2 Data Sets with Varying Dimensionality and Intersections

For the final experiment of this section, we create synthetic data in 3-D that were embedded in higher dimensions. The first data set consists of a line and a cone. The points are embedded in 50-D by three orthonormal 50-D vectors and initialized as ball tensors. Tensor voting is performed in the 50-D space and a path from point A on the line to point B on the cone is interpolated as in the previous experiment, making sure that it belongs to the local tangent space, which changes dimensionality from one to two. The data is re-projected back to 3-D for visualization in Figure 11(a).

In the second part of the experiment, we generate an intersecting S-shaped surface and a plane (a total of 11,000 points) and 30,000 outliers from a uniform distribution, and embed them in a 30-D space. Without explicitly removing the noise, we interpolate between two points on the S (A and B) and a point on the S and a point on the plane (C and D) and create the paths shown in Figure 11(b) re-projected in 3-D. The first path is curved, while the second jumps from manifold to manifold without deviating from the optimal path. (The outliers are not shown for clarity.) Processing time for 41,000



(a) Line and cone (b) S and plane

Figure 11: Nonlinear interpolation in 50-D with varying dimensionality (a) and 30-D with intersecting manifolds under noise corruption (b).

points in 30-D is 2 min. and 40 sec. on a Pentium 4 at 2.8GHz using voting neighborhoods that included an average of 44 points.

7. Generation of Unobserved Samples and Nonparametric Function Approximation

In this section, we build upon the results of the previous section to address function approximation. A common practice is to treat functions with multiple outputs as multiple single-output functions. We adopt this scheme here, even though nothing prohibits us from directly approximating multiple-input multiple-output functions. We assume that observations in N-D that include values for the input and output variables are available for training. The difference with the examples of the previous sections is that the queries are given as input vectors with unknown output values, and thus are of lower dimensionality than the voting space. The required module to convert this problem to that of Section 6 is one that can find a point on the manifold that corresponds to an input similar to the query. Then, in order to predict the output y of the function for an unknown input $\vec{x}$, under the assumption of local smoothness, we move on the manifold formed by the training samples until we reach the point corresponding to the given input coordinates. To ensure that we always remain on the manifold, we need to start from a point on it and proceed as in the previous section.

One way to find a suitable starting point is to find the nearest neighbor of $\vec{x}$ in the input space, which has fewer dimensions than the joint input-output (voting) space. Then, we can compute the desired direction in the low dimensional space and project it to the input-output space. If many outputs are possible for a given input (if the data have not been generated by a function in the strict sense), we have to either find neighbors at each branch of the function and produce multiple

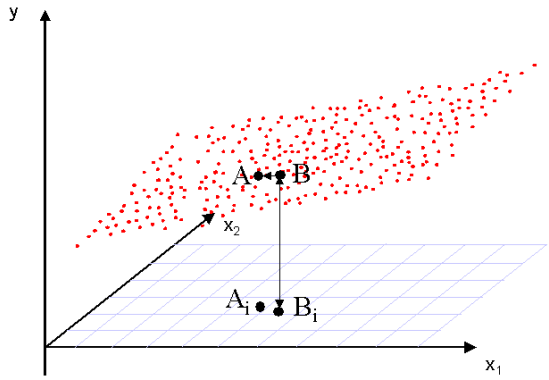


Figure 12: Interpolation to obtain output value for unknown input point A_i. B_i is the nearest neighbor in the input space and corresponds to B in the joint input-output space. We can march from B on the manifold to arrive at the desired solution A that projects on A_i in the input space.

outputs, or use other information, such as the previous state of the system, to pursue only one of the alternatives. One could find multiple nearest neighbors, run the proposed algorithm starting from each of them and produce a multi-valued answer with a probability associated with each potential output value.

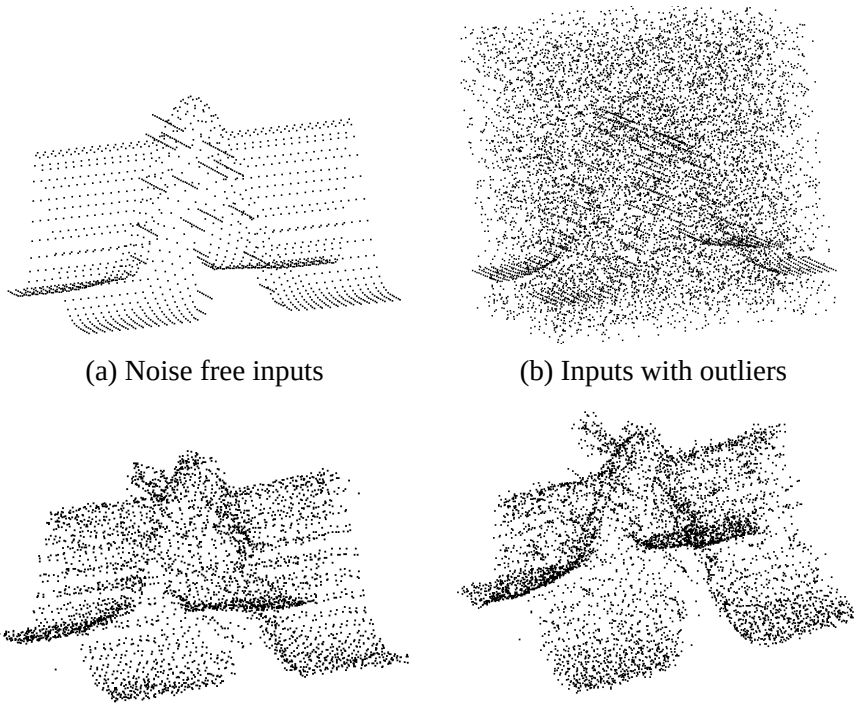
Figure 12 provides a simple illustration. We begin with a point A_i in the input space. We proceed by finding its nearest neighbor among the projections of the training data on the input space B_i. (Even if B_i is not the nearest neighbor the scheme still works but possibly requires more steps.) The sample B in the input-output space that corresponds to B_i is the starting point on the manifold. The desired direction is the projection of the A_iB_i vector on the tangent space of B. Now, we are in the case described in Section 6, where the starting point and the desired direction are known. Processing stops when the input coordinates of the point on the path from B are within ε of A_i. The corresponding point A in the input-output space is the desired interpolated sample.

As in all the experiments presented in this paper, the input points are encoded as ball tensors, since we assume that we have no knowledge of their orientation. We first attempt to approximate the following function, proposed by Schaal and Atkeson (1998):

$$y = max\{e^{-10x_1^2} \;\; e^{-50x_2^2} \;\; 1.25e^{-5(x_1^2+x_2^2)}\}. \tag{10}$$

1681 samples of y are generated by uniformly sampling the $[-1,1]\times[-1,1]$ square. We perform four experiments with increasing degree of difficulty. In all cases, after voting on the given inputs, we generate new samples by interpolating between the input points. The four configurations and noise conditions were:

- In the first experiment, we performed all operations with noise free data in 3-D.
- For the second experiment, we added 8405 outliers (five times more than the inliers) drawn from a uniform distribution in a $2\times2\times2$ cube containing the data.
- For the third experiment, we added Gaussian noise with variance 0.01 to the coordinates of all points while adding the same number of outliers as above.



(a) Noise free inputs (b) Inputs with outliers

(c) Interpolated points with outliers and perturbation (d) Interpolated points with outliers and perturbation in 60-D

Figure 13: Inputs and interpolated points for Eq. 10. The top row shows the noise-free inputs and the noisy input set where only 20% of the points are inliers. The bottom row shows the points generated in 3-D and 60-D respectively. In both cases the inputs were contaminated with outliers and Gaussian noise.

- Finally, we embedded the perturbed data (and the outliers) in a 60-D space, before voting and nonlinear interpolation.

The noise-free and noisy input, as well as the generated points can be seen in Figure 13. We computed the mean square error between the outputs generated by our method and Eq. 10 normalized by the variance of the noise-free data. The NMSE for all cases is reported in Table 11. Robustness against outliers is due to the fact that the inliers form a consistent surface and thus receive votes that support the correct local structure from other inliers. Outliers, on the other hand, are random and do not form any structure. They cast and receive inconsistent votes and therefore neither develop a preference for a certain manifold nor significantly disrupt the structure estimates at the inliers. They can be removed by simple thresholding since all their eigenvalues are small and almost equal, but this is not done here. Note that performance in 60-D is actually better since the interference by outliers is reduced as the dimensionality of the space increases. Tensor voting is also robust against perturbation of the coordinates as long as it is not biased to favor a certain direction. If the perturbation is zero-mean, its effects on individual votes are essentially canceled out, because they only contribute to the ball component of the accumulated tensor at each point, causing small errors in orientation estimation.

Experiment	NMSE
Noise-free	0.0041
Outliers	0.0170
Outliers & N(0, 0.01)	0.0349
Outliers & N(0, 0.01) in 60-D	0.0241

Table 11: Normalized MSE for the interpolated points of Eq. 10 under different noise conditions.

7.1 Results on Real Data

The final experiments are on real data taken from the University of California at Irvine Machine Learning Repository (Newman et al., 1998) available online at `http://www.ics.uci.edu/~mlearn/MLRepository.html` and the University of Toronto DELVE archive (`http://www.cs.toronto.edu/~delve`)(Rasmussen et al., 1996). We used the "abalone", "Boston housing" and "Computer activity" data sets. These data sets were selected because they contain data from a single class in spaces of 9, 14 and 22 dimensions respectively. In each case one variable is treated as the output and all others as inputs. Most of the input variables are continuous, but vary widely in magnitude. Since our method is based on distance relationships between the samples, we re-scaled the data so that the ratio of maximum to minimum standard deviation of the variables was approximately $10:1$, instead of the original, which for several cases exceeded $1000:1$. We split the data in training and test sets and perform tensor voting on the training data to learn the structure of the manifold. For each test sample, we begin by finding the average position of a few nearest neighbors in the $(N-1)$-D input space and then follow a path on the estimated manifold in N-D until the desired input coordinates are reached. The value of the output variable when the input variables are equal to the query is the estimate returned by our method.

In the case of the "abalone" data, we follow common practice and map the first variable from M for male, F for female and I for infant to (1, 0, 0), (0, 1, 0) and (0, 0, 1) respectively. The resulting space is 12-D, with the last variable being the one we attempt to estimate. Again, we scale the data so that the variances of different variables are comparable. Following most authors, we divide the 4177 samples in training and test sets containing 3000 and 1177 samples respectively. Our results after 10 runs of the experiment with randomly selected training and test sets and a comparison with a number of other published results on the same data can be seen in Table 12. We also applied our algorithm to the "Boston housing" data set, which has been extensively used as a benchmark. It contains 506 samples of house prices as a function of 13 variables. We use training and test sets containing 481 and 25 points respectively. Due to the small size of the test set, we repeated the experiment 20 times always using as queries points that had not been included in the test set before, thus using virtually all points for queries. Error rates can be seen in Table 12. Finally, we used the "computer activity" data set from the DELVE archive (Rasmussen et al., 1996). It contains 8192 observations in a 22-D space. We used 2000 samples for training and the rest for testing.

In summary, our algorithm achieves what we think is satisfactory performance despite the fact that the data sets contain insufficient samples to describe complex manifolds. The sparseness of the data under-uses the capability of our approach to handle complex nonlinear manifolds. We observed that for certain query points there are very few similar samples in the training set. In the absence of enough samples, our algorithm may not improve the initial solution given by the nearest neighbors

of the point. These errors cause the ranking of our results in terms of RMS to be worse than in terms of mean absolute error (MAE) in Table 12.

	Abalone		Housing		Computer	
	MAE	RMS	MAE	RMS	MAE	RMS
BCM (Tresp, 2000)	-	-	-	3.100	-	-
GPR1 (Tresp, 2000)	-	-	-	3.013	-	-
RVM (Tipping, 2001)	-	-	-	8.04	-	-
SVM (Tipping, 2001)	-	-	-	7.46	-	-
Sparse GPR (Smola and Bartlett, 2001)	1.785	-	-	-	-	-
GPR2 (Smola and Bartlett, 2001)	1.782	-	-	-	-	-
Online GPR (Schwaighofer and Tresp, 2003)	-	2.111	-	-	-	-
BCM2 (Schwaighofer and Tresp, 2003)	-	2.111	-	-	-	-
Inductive SRM (Schwaighofer and Tresp, 2003)	-	2.109	-	-	-	-
Transductive SRM (Schwaighofer and Tresp, 2003)	-	2.109	-	-	-	-
SVR (Chu et al., 2004)	1.421	2.141	2.13	3.205	2.28	3.715
BSVR (Chu et al., 2004)	1.464	2.134	2.19	3.513	2.33	4.194
GPR-ARD (Chu et al., 2004)	1.493	2.134	2.01	2.884	1.686	2.362
BSVR-ARD (Chu et al., 2004)	1.454	2.119	1.86	2.645	1.687	2.408
Tensor voting	1.630	2.500	1.272	1.860	1.970	2.815

Table 12: Mean Absolute Error (MAE) and Root Mean squared Error (RMS) for benchmark data sets. Unless otherwise noted, training and testing is performed with 3000 and 1177 samples for "abalone", 481 and 25 for "Boston housing" and 2000 and 6192 for "computer activity", respectively. Results by other methods were not generated by us. BCM is the Bayesian committee machine of Tresp (2000) and GPR1 is Gaussian process regression implemented by Tresp. 400 samples are used for training for "Boston housing" for BCM and GPR1. RVM is the relevance vector machine of Tipping (2001) and SVM is a support vector machine implemented by Tipping. Sparse GPR is the algorithm of Smola and Bartlett (2001) and GPR2 is their implementation of Gaussian process regression. 4000 samples are used for training for the "abalone" data set for Sparse GPR and GPR2. Online GPR is the algorithm of Csató and Opper (2002), BCM2 is the Bayesian committee machine implemented by Schwaighofer and Tresp (2003), while inductive and transductive SRM are algorithms from the same paper. The number of training and testing samples is not provided by the authors for Online GPR, BCM2, inductive and transductive SRM. SVR is the support vector regression of Vapnik (1995), BSVR is Bayesian support vector regression of Chu et al. (2004), GPR-ARD is Gaussian process regression using the ARD Gaussian covariance function and BSVR-ARD is BSVR using the ARD function (Chu et al., 2004).

8. Discussion

We have presented an approach for dimensionality estimation, manifold learning and function approximation that offers certain advantages over the state of the art. Tensor voting may on the surface look similar to other local, instance-based learning algorithms that propagate information from point to point, but the fact that the votes are tensors and not scalars allows them to convey considerably more information. The properties of the tensor representation, which can handle the simultaneous presence of multiple orientations and structure types, allow the reliable inference of the normal and tangent space at each point. In addition, tensor voting is very robust against outliers, as demonstrated for the 2-D and 3-D case in numerous publications including Tang and Medioni (1998) and Medioni et al. (2000). This property holds in higher dimensions, where random noise is even more scattered. See for instance the results presented in Table 11.

It should also be noted that the votes attenuate with distance and curvature. This is a more intuitive formulation than using the k nearest neighbors with equal weights, since some of them may be too far, or belong to a different part of the manifold. For both tensor voting and the methods presented in Section 2, however, the distance metric in the input space has to be meaningful. Our method is less sensitive to a somewhat incorrect selection of the distance metric since all neighbors do not contribute equally. After this choice has been made, the only free parameter in our approach is σ, the scale of voting. Small values tend to preserve details better, while large values are more robust to noise. The scale can be selected automatically by randomly sampling a few points before voting and making sure that enough points are included in their voting neighborhoods. Our results show that sensitivity with respect to scale is small, as shown in Tables 1, 3, 4-6 and 7. The number of points that can be considered sufficient is a function of the dimensionality of the space, the intrinsic dimensionality of the data, as well as noise and curvature. The two latter factors have been analyzed by Mitra et al. (2004) and Lalonde et al. (2005), using the Gershgorin Circle Theorem, for the case of curves in 2-D and surfaces in 3-D under mild restrictions on data distribution. To the best of our knowledge no similar analysis has been done for manifolds with co-dimension other than one or in high-dimensional spaces. A thorough investigation of these issues is among the objectives of our future research.

Our algorithms fail when the available observations do not suffice to represent the manifold, as for instance in the face with varying pose and illumination data set of Tenenbaum et al. (2000), where 698 instances represent a manifold in 4096-D. Global methods may be more successful in such situations. The number of sufficient samples for tensor voting cannot be easily predicted from the dimensionality of the space, since it also depends on the complexity (curvature) of the underlying manifolds. (See also the discussion above.) For instance, 486 samples in 14-D turn out to be sufficient for the "Boston housing" function approximation experiment. Nevertheless, in many practical cases the challenges are the over-abundance of data and the need for efficient processing of large data sets. Tensor voting is a well suited framework for such cases, since it can efficiently process hundreds of thousands of points in spaces of up to a few hundred dimensions.

Another important advantage of tensor voting is the absence of global computations, which makes time complexity $O(NMlogM)$, where N is the dimensionality of the space and M is the number of points. This property enables us to process data sets with very large number of points. Computation time does not become impractical as the number of points grows, assuming that more points are added to the data set in such a way that the density remains constant. In this case, the number of votes cast per point remains constant and time requirements grow linearly. Com-

plexity is adversely affected by the dimensionality of the space N, since eigen-decomposition of $N \times N$ tensors has to be performed resulting in a complexity that is cubic with respect to N due to the eigensystem computations that are $O(N^3)$. For most practical purposes, however, the number of points has to be considerably larger[2] than the dimensionality of the space $(M \gg N)$ to allow structure inference. The complexity for a nearest neighbor query using the ANN k-d tree (Arya et al., 1998) is $O(NlogM)$ and one query is required for each voter. Thus the total complexity is $O(NMlogM + MN^3) \approx O(NMlogM)$. Computational complexity, therefore, is reasonable with respect to the largest parameter, which for our methods to work has to be M. Table 1 shows the effect of data set and neighborhood size on processing time. Notice that time is linear with respect to the average number of points in each neighborhood, as expected. Space requirements for M $N \times N$ tensors are $O(MN^2)$.

In terms of dimensionality estimation, we are able to obtain accurate estimates at the point level. Moreover, since the dimensionality is found as the maximum gap in the eigenvalues of the tensor at each point, no thresholds are needed. Under most other approaches, the dimensionality has to be provided, or, at best, an average intrinsic dimensionality is estimated for the entire data set, as in Bruske and Sommer (1998), Brand (2003), Kégl (2003), Weinberger and Saul (2004) and Costa and Hero (2004).

The novelty of our approach regarding manifold learning is that it is not based on dimensionality reduction. Instead, we perform tasks such as geodesic distance measurement and nonlinear interpolation in the input space. Experimental results show that we can perform these tasks in the presence of outlier noise at high accuracy, even without explicitly removing the outliers from the data. This choice also broadens the range of data sets we can process. While isometric embeddings can be achieved for a certain class of manifolds, we are able to process non-flat manifolds and even non-manifolds. The last experiment of Section 6 demonstrates our ability to work with data sets of varying dimensionality or with intersecting manifolds. To the best of our knowledge, this is impossible with any other method. If dimensionality reduction is desired due to its considerable reduction in storage requirements, a dimensionality reduction method, such as Roweis and Saul (2000), Tenenbaum et al. (2000), Belkin and Niyogi (2003), Brand (2003), Donoho and Grimes (2003) and Weinberger and Saul (2006), can be used after tensor voting. The benefits of this process are in the form of noise robustness and smooth component identification, with respect to both dimensionality and orientation, via tensor voting followed by memory savings via dimensionality reduction.

We have also presented, in Section 7, a local nonparametric approach for function approximation that combines the advantages of local methods with the efficient representation and information propagation of tensor voting. Local function approximation methods are more flexible in the type of functions they can approximate, since the properties of the function are allowed to vary throughout the space. Our approach, in particular, has no parameters that have to be selected, such as the number and type of local models to be used, besides the scale of voting. Its drawback, in line with other local methods, is higher memory requirements. We have shown that we can process challenging examples from the literature under very adverse noise conditions. As shown in the example of Eq. 10, even when more than 80% of the samples are outliers and the inliers are corrupted by noise in the form of perturbation, we are still able to correctly predict unobserved outputs. We have also shown in Table 12 promising results on publicly available data sets, despite the fact that they are not

2. While in principal three points are sufficient to define a surface, 10 points are sufficient to define a 9-D manifold and so forth, one or two orders of magnitude more points are required for practical applications in our experience.

well suited for our approach, since the number of observations they contain is hardly sufficient to define the manifold.

As mentioned above, an issue we do not fully address here is that of the selection of an appropriate distance metric. We assume that the Euclidean distance in the input coordinate system is a meaningful distance metric. This is not the case if the coordinates are not of the same type. On the other hand, a metric such as the Mahalanobis distance is not necessarily appropriate in all cases, since the data typically lie in a limited part of the input space and scaling all dimensions, including the redundant ones, to achieve equal variance would be detrimental. For the experiments shown in Section 7, we apply heuristic scaling of the coordinates when necessary. We intend to develop a systematic way based on cross-validation that automatically scales the coordinates by maximizing prediction performance for the observations that have been left out.

Our future research will focus on addressing the limitations of our current algorithm and extending its capabilities. An interpolation mechanism that takes into account holes and boundaries during geodesic distance approximation should be implemented. Additionally, in the area of function approximation, the issue of approximating functions with multiple branches for the same input value, which often appear in practical applications, has to be handled more rigorously. We also intend to develop an online, incremental version of our approach, possibly including a forgetting and an updating module, that will be able to process data as they are collected, instead of requiring the entire data set to proceed. Potential applications of our work include challenging real problems, such as the study of direct and inverse kinematics where there are typically large numbers of samples in spaces of up to a few hundred dimensions. Function approximation for complex functions, for which global models would become very complicated, is another area where our methods could be effective. Finally, one can view the proposed approach as learning data from a single class, which can serve as the groundwork for an approach for pattern recognition, data mining, supervised and unsupervised classification.

Acknowledgments

This research has been supported by the National Science Foundation grant IIS 03 29247. The authors would like to thank Adit Sahasrabudhe for his help with some of the experiments.

References

S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu. An optimal algorithm for approximate nearest neighbor searching. *Journal of the ACM*, 45:891–923, 1998.

C.G. Atkeson, A.W. Moore, and S. Schaal. Locally weighted learning. *Artificial Intelligence Review*, 11(1-5):11–73, 1997.

A.R. Barron. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information Theory*, 39(3):930–945, 1993.

M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003.

M. Brand. Nonrigid embeddings for dimensionality reduction. In *European Conference on Machine Learning*, pages 47–59, 2005.

M. Brand. Charting a manifold. In *Advances in Neural Information Processing Systems 15*, pages 961–968. MIT Press, Cambridge, MA, 2003.

L. Breiman. Hinging hyperplanes for regression, classification, and function approximation. *IEEE Transactions on Information Theory*, 39(3):999–1013, 1993.

J. Bruske and G. Sommer. Intrinsic dimensionality estimation with optimally topology preserving maps. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(5):572–575, 1998.

W. Chu, S.S. Keerthi, and C.J. Ong. Bayesian support vector regression using a unified loss function. *IEEE Transactions on Neural Networks*, 15(1):29–44, 2004.

R. Collobert and S. Bengio. Svmtorch: Support vector machines for large-scale regression problems. *Journal of Machine Learning Research*, 1(2):143–160, 2001.

J. Costa and A.O. Hero. Geodesic entropic graphs for dimension and entropy estimation in manifold learning. *IEEE Transactions on Signal Process*, 52(8):2210–2221, 2004.

T. Cox and M. Cox. *Multidimensional Scaling*. Chapman & Hall, London, 1994.

L. Csató and M. Opper. Sparse on-line gaussian processes. *Neural Computation*, 14(3):641–668, 2002.

V. de Silva and J.B. Tenenbaum. Global versus local methods in nonlinear dimensionality reduction. In *Advances in Neural Information Processing Systems 15*, pages 705–712. MIT Press, Cambridge, MA, 2003.

P. Dollár, V. Rabaud, and S. Belongie. Learning to traverse image manifolds. In *Advances in Neural Information Processing Systems 19*, pages 361–368. MIT Press, Cambridge, MA, 2007a.

P. Dollár, V. Rabaud, and S. Belongie. Non-isometric manifold learning: Analysis and an algorithm. In *International Conference on Machine Learning*, 2007b.

D. Donoho and C. Grimes. Hessian eigenmaps: new tools for nonlinear dimensionality reduction. In *Proceedings of National Academy of Science*, pages 5591–5596, 2003.

G. Guy and G. Medioni. Inferring global perceptual contours from local features. *International Journal of Computer Vision*, 20(1/2):113–133, 1996.

G. Guy and G. Medioni. Inference of surfaces, 3D curves, and junctions from sparse, noisy, 3D data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(11):1265–1277, 1997.

X. He and P. Niyogi. Locality preserving projections. In *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.

F. Heitger and R. von der Heydt. A computational model of neural contour processing: Figure-ground segregation and illusory contours. In *International Conference on Computer Vision*, pages 32–40, 1993.

I.T. Jolliffe. *Principal Component Analysis*. Springer-Verlag, New York, 1986.

B. Kégl. Intrinsic dimension estimation using packing numbers. In *Advances in Neural Information Processing Systems 15*, pages 681–688. MIT Press, Cambridge, MA, 2003.

K. Koffka. *Principles of Gestalt Psychology*. Harcourt, Brace, New York, 1935.

W. Köhler. Physical gestalten. *W.D. Ellis (ed), A source book of Gestalt psychology (1950)*, pages 17–54, 1920.

J.-F. Lalonde, R. Unnikrishnan, N. Vandapel, and M. Hebert. Scale selection for classification of point-sampled 3-d surfaces. In *International Conference on 3-D Digital Imaging and Modeling*, 2005.

S. Lawrence, A. C. Tsoi, and A. D. Back. Function approximation with neural networks and local methods: Bias, variance and smoothness. In *Australian Conference on Neural Networks*, pages 16–21, 1996.

E. Levina and P. Bickel. Maximum likelihood estimation of intrinsic dimension. In *Advances in Neural Information Processing Systems 17*, pages 777–784. MIT Press, Cambridge, MA, 2005.

Z. Li. A neural model of contour integration in the primary visual cortex. *Neural Computation*, 10: 903–940, 1998.

G. Medioni, M.S. Lee, and C.K. Tang. *A Computational Framework for Segmentation and Grouping*. Elsevier, New York, NY, 2000.

Facundo Mémoli and Guillermo Sapiro. Distance functions and geodesics on submanifolds of R^d and point clouds. *SIAM Journal of Applied Mathematics*, 65(4):1227–1260, 2005.

S. Mitaim and B. Kosko. The shape of fuzzy sets in adaptive function approximation. *IEEE Transactions on Fuzzy Systems*, 9(4):637–656, 2001.

T.M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.

N.J. Mitra, A. Nguyen, and L. Guibas. Estimating surface normals in noisy point cloud data. *International Journal of Computational Geometry and Applications*, 14(4–5):261–276, 2004.

P. Mordohai. *A Perceptual Organization Approach for Figure Completion, Binocular and Multiple-View Stereo and Machine Learning using Tensor Voting*. Ph.D. Thesis, University of Southern California, 2005.

P. Mordohai and G. Medioni. Unsupervised dimensionality estimation and manifold learning in high-dimensional spaces by tensor voting. *International Joint Conference on Artificial Intelligence*, 2005.

P. Mordohai and G. Medioni. Stereo using monocular cues within the tensor voting framework. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(6):968–982, 2006.

D.J. Newman, S. Hettich, C.L. Blake, and C.J. Merz. UCI repository of machine learning databases, 1998. URL `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

P. Parent and S.W. Zucker. Trace inference, curvature consistency, and curve detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(8):823–839, 1989.

T. Poggio and F. Girosi. Networks for approximation and learning. *Proc. of IEEE*, 78(9):1481–1497, 1990.

M. Raginsky and S. Lazebnik. Estimation of intrinsic dimensionality using high-rate vector quantization. In *Advances in Neural Information Processing Systems 18*, pages 1105–1112. MIT Press, Cambridge, MA, 2006.

C.E. Rasmussen, R.M. Neal, G. Hinton, D. van Camp, M. Revow, Z. Ghahramani, R. Kustra, and R. Tibshirani. The DELVE archive, 1996. URL `http://www.cs.toronto.edu/~delve`.

S.T. Roweis and L.K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323–2326, 2000.

A. Saha, C.L. Wu, and D.S. Tang. Approximation, dimension reduction, and nonconvex optimization using linear superpositions of gaussians. *IEEE Transactions on Computers*, 42(10): 1222–1233, 1993.

T. D. Sanger. A tree-structured algorithm for reducing computation in networks with separable basis functions. *Neural Computation*, 3(1):67–78, 1991.

S. Sarkar and K.L. Boyer. A computational structure for preattentive perceptual organization: Graphical enumeration and voting methods. *IEEE Transactions on Systems, Man and Cybernetics*, 24:246–267, 1994.

L. K. Saul and S. T. Roweis. Think globally, fit locally: unsupervised learning of low dimensional manifolds. *Journal of Machine Learning Research*, 4:119–155, 2003.

E. Saund. Labeling of curvilinear structure across scales by token grouping. In *International Conference on Computer Vision and Pattern Recognition*, pages 257–263, 1992.

S. Schaal and C.G. Atkeson. Constructive incremental learning from only local information. *Neural Computation*, 10(8):2047–2084, 1998.

B. Schölkopf, A.J. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5):1299–1319, 1998.

A. Schwaighofer and V. Tresp. Transductive and inductive methods for approximate gaussian process regression. In *Advances in Neural Information Processing Systems 15*, pages 953–960. MIT Press, Cambridge, MA, 2003.

F. Sha and L. Saul. Analysis and extension of spectral methods for nonlinear dimensionality reduction. In *International Conference on Machine Learning*, pages 784–791, 2005.

A. Shashua and S. Ullman. Structural saliency: The detection of globally salient structures using a locally connected network. In *International Conference on Computer Vision*, pages 321–327, 1988.

A.J. Smola and P.L. Bartlett. Sparse greedy gaussian process regression. In *Advances in Neural Information Processing Systems 13*, pages 619–625. MIT Press, Cambridge, MA, 2001.

R. Souvenir and R. Pless. Manifold clustering. In *International Conf on Computer Vision*, pages I: 648–653, 2005.

C.K. Tang and G. Medioni. Inference of integrated surface, curve, and junction descriptions from sparse 3d data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(11):1206–1223, 1998.

C.K. Tang, G. Medioni, and M.S. Lee. N-dimensional tensor voting and application to epipolar geometry estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(8): 829–844, 2001.

Y.W. Teh and S. Roweis. Automatic alignment of local representations. In *Advances in Neural Information Processing Systems 15*, pages 841–848. MIT Press, Cambridge, MA, 2003.

J.B. Tenenbaum, V. de Silva, and J.C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2323, 2000.

J. Ting, A D'Souza, and S. Schaal. bayesian regression with input noise for high dimensional data. In *International Conference on Machine Learning*, 2006.

M.E. Tipping. Sparse bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, 1(3):211–244, 2001.

V. Tresp. A bayesian committee machine. *Neural Computation*, 12(11):2719–2741, 2000.

V. Vapnik. *The Nature of Statistical Learning Theory*. Springer, Berlin Heidelberg New York, 1995.

S. Vijayakumar and S. Schaal. Locally weighted projection regression: An o(n) algorithm for incremental real time learning in high dimensional space. In *International Conference on Machine Learning*, pages I:288–293, 2000.

J. Wang, Z. Zhang, and H. Zha. Adaptive manifold learning. In *Advances in Neural Information Processing Systems 17*. MIT Press, Cambridge, MA, 2005.

D. Wedge, D. Ingram, D. McLean, C. Mingham, and Z. Bandar. On global-local artificial neural networks for function approximation. *IEEE Transactions on Neural Networks*, 17(4):942–952, 2006.

K. Q. Weinberger and L. K. Saul. Unsupervised learning of image manifolds by semidefinite programming. *International Journal of Computer Vision*, 70(1):77–90, 2006.

K. Q. Weinberger, F. Sha, and L. K. Saul. Learning a kernel matrix for nonlinear dimensionality reduction. In *International Conference on Machine Learning*, pages 839–846, 2004.

K.Q. Weinberger and L.K. Saul. Unsupervised learning of image manifolds by semidefinite programming. In *International Conference on Computer Vision and Pattern Recognition*, pages II: 988–995, 2004.

M. Wertheimer. Laws of organization in perceptual forms. *Psycologische Forschung, Translation by W. Ellis, A source book of Gestalt psychology (1938)*, 4:301–350, 1923.

C.K.I. Williams and C.E. Rasmussen. Gaussian processes for regression. In *Advances in Neural Information Processing Systems 8*, pages 514–520. MIT Press, Cambridge, MA, 1996.

L. Xu, M. I. Jordan, and G. E. Hinton. An alternative model for mixtures of experts. In *Advances in Neural Information Processing Systems 7*, pages 633–640. MIT Press, Cambridge, MA, 1995.

S.C. Yen and L.H. Finkel. Extraction of perceptually salient contours by striate cortical networks. *Vision Research*, 38(5):719–741, 1998.

Z. Zhang and H. Zha. Principal manifolds and nonlinear dimension reduction via local tangent space alignment. *SIAM Journal of Scientific Computing*, 26(1):313–338, 2004.