1 Neuroscale

1.1 Slide 8

Radial basis function networks can be linked with a number of other modelling techniques: function approximation, noisy interpolation, and kernel regression (see Bishop, Chapter 6). A common theme of all these viewpoints is that the sum of the basis functions $\sum_{j=0}^{M} \phi_j$ should form a representation of the unconditional probability density of the input data. We shall see later how this can be used to develop efficient training algorithms for RBF networks, but first we use it to motivate the choice of basis function. Let us suppose that we are solving a classification problem with c classes. We use a set of M density functions, labelled by an index j, to represent the class-conditional probability densities:

$$p(\mathbf{x}|\mathcal{C}_k) = \sum_{j=1}^{M} p(\mathbf{x}|j)P(j|\mathcal{C}_k), \qquad k = 1, \ldots, c. \tag{1.1}$$

To find the unconditional data density, we sum (1.1) over all classes

$$p(\mathbf{x}) = \sum_{k=1}^{c} p(\mathbf{x}|\mathcal{C}_k)P(\mathcal{C}_k) \tag{1.2}$$

$$= \sum_{j=1}^{M} p(\mathbf{x}|j)P(j), \tag{1.3}$$

where

$$P(j) = \sum_{k=1}^{c} P(j|\mathcal{C}_k)P(\mathcal{C}_k). \tag{1.4}$$

Then we can compute the posterior probabilities of class membership by substituting (1.1) and (1.3) into Bayes' theorem to obtain

$$P(\mathcal{C}_k|\mathbf{x}) = \sum_{j=1}^{M} w_{kj}\phi_j(\mathbf{x}), \tag{1.5}$$

where the basis functions ϕ_j are given by

$$\phi_j(\mathbf{x}) = \frac{p(\mathbf{x}|j)P(j)}{\sum_{j'=1}^{M} p(\mathbf{x}|j')P(j')} \tag{1.6}$$

$$= P(j|\mathbf{x}) \tag{1.7}$$

and the second layer weights are given by

$$w_{kj} = \frac{P(j|\mathcal{C}_k)P(\mathcal{C}_k)}{P(j)} \tag{1.8}$$

$$= P(\mathcal{C}_k|j). \tag{1.9}$$

To ensure that the network as a whole approximates probabilities, we can use positive definite normalised basis functions which are themselves density functions: in this case, we have a mixture model formalisation. One obvious choice for the basis functions is therefore a Gaussian. However, there is a drawback to this choice: the resulting density estimator is necessarily biased for a finite set of samples. If we allow non-negative basis functions, then the asymptotic convergence of the bias (as the number of samples n tends to infinity) is faster than if we insist on positive definite basis functions [Rosenblatt, 1956, Shapiro, 1969, Yamato, 1972].

1.2 Slide 10

Equating the derivative of the sum-squared error to zero yields the *normal* equations

$$(\mathbf{\Phi}^T\mathbf{\Phi})\mathbf{w} = \mathbf{\Phi}^T\mathbf{Y} \tag{1.10}$$

which can be solved efficiently by computing the *pseudo-inverse* $\mathbf{\Phi}^\dagger$ of $\mathbf{\Phi}$ and setting $\mathbf{w} = \mathbf{\Phi}^\dagger\mathbf{Y}$. This is numerically more stable than computing the inverse of the square matrix $\mathbf{\Phi}^T\mathbf{\Phi}$ explicitly.

1.3 Slide 12

The key references are [Lowe and Tipping, 1996] and [Tipping and Lowe, 1997].

1.4 Slide 14

Recall that the distance between points $\mathbf{y}_i$ and $\mathbf{y}_j$ is denoted by d_{ij}, and the quality of the projection is measured by the *Sammon stress metric*:

$$E_{\text{sam}} = \sum_{i=1}^{N}\sum_{j>i}^{N}(d_{ij} - d_{ij}^*)^2. \tag{1.11}$$

Using the chain rule, we write

$$\frac{\partial E}{\partial w_{kr}} = \sum_{i}^{N}\frac{\partial E}{\partial \mathbf{y}_i}\frac{\partial \mathbf{y}_i}{\partial w_{kr}}. \tag{1.12}$$

By differentiating (1.11), it is easy to see that

$$\frac{\partial E}{\partial \mathbf{y}_i} = -2\sum_{j\neq i}\left(\frac{d_{ij}^* - d_{ij}}{d_{ij}}\right)(\mathbf{y}_i - \mathbf{y}_j), \tag{1.13}$$

while, for an RBF network, the gradient of $\mathbf{y}_i$ with respect to the output network weights is given by the same equation as for the sum-of-squares error

$$\frac{\partial \mathbf{y}_i}{\partial w_{kr}} = \delta_{ir}z_k, \tag{1.14}$$

where z_k is the output of the kth hidden unit, and δ_{ir} is the Kronecker delta. Assuming that the hidden unit centres and widths are chosen to model the input data distribution as in the usual two-stage training process for supervised problems, these derivatives are all that is needed to train an RBF network to perform a topographic projection.

We write the output of the network in matrix form as

$$\mathbf{Y} = \mathbf{\Phi}\mathbf{W}, \tag{1.15}$$

where $\mathbf{\Phi}$ is the matrix of hidden unit activations including bias terms. (We assume that the hidden unit parameters have been already determined.)

$$\mathbf{\Phi} = \begin{bmatrix} \phi_0(\mathbf{x}_1) & \phi_1(\mathbf{x}_1) & \phi_2(\mathbf{x}_1) & \cdots & \phi_M(\mathbf{x}_1) \\ \phi_0(\mathbf{x}_2) & \phi_1(\mathbf{x}_2) & \phi_2(\mathbf{x}_2) & \cdots & \phi_M(\mathbf{x}_2) \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \phi_0(\mathbf{x}_N) & \phi_1(\mathbf{x}_N) & \phi_2(\mathbf{x}_N) & \cdots & \phi_M(\mathbf{x}_N) \end{bmatrix}. \tag{1.16}$$

Then we can express (1.12) (for output dimension r) in the form

$$\boldsymbol{\nabla} E = \boldsymbol{\Phi}^T \mathbf{e}_r, \tag{1.17}$$

where

$$\boldsymbol{\nabla} E = \left(\frac{\partial E}{\partial w_{1r}}, \frac{\partial E}{\partial w_{2r}}, \ldots, \frac{\partial E}{\partial w_{Mr}} \right)^T, \tag{1.18}$$

and

$$\mathbf{e}_r = \left(\frac{\partial E}{\partial \mathbf{y}_{1r}}, \frac{\partial E}{\partial \mathbf{y}_{2r}}, \ldots, \frac{\partial E}{\partial \mathbf{y}_{Mr}} \right)^T. \tag{1.19}$$

Equation (1.17) is simply an expression of the chain rule for a model whose output is linear in its weights. In a least-squares problem, with error

$$E = \frac{1}{2} \sum_{i=1}^{N} \|\mathbf{y}_i - \mathbf{t}_i\|, \tag{1.20}$$

where $\mathbf{t}_i$ represents an explicit target value, the same equation is valid but with

$$\frac{\partial E}{\partial \mathbf{y}_i} = (\mathbf{y}_i - \mathbf{t}_i). \tag{1.21}$$

The key idea of the shadow targets algorithm is to use (1.21) to estimate *hypothetical* targets $\hat{\mathbf{t}}_i$.

$$\begin{aligned} \hat{\mathbf{t}}_i &= \mathbf{y}_i - \frac{\partial E}{\partial \mathbf{y}_i} \\ &= \mathbf{y}_i + 2 \sum_{j \neq i} \left(\frac{d_{ij} - d^*_{ij}}{d_{ij}} \right) (\mathbf{y}_i - \mathbf{y}_j), \end{aligned} \tag{1.22}$$

the last step following from (1.13). The vectors $\hat{\mathbf{t}}_i$ represent (or shadow) the exact targets for the network that would lead to an identical expression for the weight derivatives in the RBF in the least squares regression problem. For a fixed set of targets, the least squares problem can be solved directly:

$$\mathbf{W} = \boldsymbol{\Phi}^\dagger \hat{\mathbf{T}}. \tag{1.23}$$

Of course, for our problem the estimated targets $\hat{\mathbf{t}}_i$ are not fixed, since $\partial E / \partial \mathbf{y}_i$ depends on the network weights. A logical approach is to iterate this procedure, re-estimating the shadow targets at each step. However, in the early stages of training, the targets estimated by (1.22) may be poor, and hence the weights given by (1.23) may increase the error. It is therefore more practical to trust the approximation given by (1.22) to a limited extent and to increase our trust only when E_{sam} decreases. This is achieved by introducing an additional parameter η and estimating the targets by

$$\hat{\mathbf{t}}_i = \mathbf{y}_i - \eta \frac{\partial E}{\partial \mathbf{y}_i}. \tag{1.24}$$

It is clear that η should be restricted to the range $(0, 1)$.

The full shadow targets algorithm consists of the following steps:

1. Initialise the weights $\mathbf{W}$ to small random values.
2. Initialise η to some small positive value.
3. Calculate $\boldsymbol{\Phi}^\dagger$.

4. Use (1.24) to compute estimated targets $\hat{\mathbf{t}}_i$.

5. Solve for the weights $\mathbf{W} = \mathbf{\Phi}^\dagger \hat{\mathbf{T}}$.

6. Calculate E_{sam} and compare with previous value.

   (a) If E_{sam} has increased, set $\eta = \eta \times k_{\text{down}}$. Restore previous values of $\mathbf{W}$.

   (b) If E_{sam} has decreased, set $\eta = \eta \times k_{\text{up}}$.

7. If convergence has not been achieved, return to Step 4.

Note that one of the most computationally expensive parts of this algorithm, the calculation of the pseudo-inverse $\mathbf{\Phi}^\dagger$, need only be done once. The algorithm is robust to changes in k_{down} and k_{up}: in practice the values 0.1 and 1.2 have worked well. It has been shown that the stationary points of this algorithm are the same as gradient-based optimisation algorithms, but that it tends to produce models with better generalisation [Tipping, 1996].

2 Stochastic Neighbourhood Embedding

2.1 Slide 5

The similarity of datapoint $\mathbf{x}_j$ to datapoint $\mathbf{x}_i$ is the conditional probability, $p_{j|i}$, that $\mathbf{x}_i$ would pick $\mathbf{x}_j$ as its neighbour if neighbours were picked in proportion to their probability density under a Gaussian centered at $\mathbf{x}_i$. For nearby datapoints, $p_{j|i}$ is relatively high, whereas for widely separated datapoints, $p_{j|i}$ will be almost infinitesimal (for reasonable values of the variance of the Gaussian, σ_i).

2.2 Slide 9

The MNIST data set contains 60,000 grayscale images of handwritten digits[1]. For the experiments, we randomly selected 6,000 of the images for computational reasons. The Olivetti faces data set[2] consists of images of 40 individuals with small variations in viewpoint, large variations in expression, and occasional addition of glasses. The data set consists of 400 images (10 per individual), and is labeled according to identity. The COIL-20 data set[3] contains images of 20 different objects viewed from 72 equally spaced orientations, yielding a total of 1,440 images.

We start by using PCA to reduce the dimensionality of the data to 30. This speeds up the computation of pairwise distances between the datapoints and suppresses some noise without severely distorting the interpoint distances. We then use each of the dimensionality reduction techniques to convert the 30-dimensional representation to a two-dimensional map and we show the resulting map as a scatterplot. For all of the data sets, there is information about the class of each datapoint, but the class information is only used to select a color and/or symbol for the map points. The class information is not used to determine the spatial coordinates of the map points. The coloring thus provides a way of evaluating how well the map preserves the similarities within each class.

[1] The MNIST data set is publicly available from `https://deepai.org/dataset/mnist`

[2] `https://scikit-learn.org/0.19/datasets/olivetti_faces.html`

[3] `https://www.cs.columbia.edu/CAVE/software/softlib/coil-20.php`

2.3 Slides 13–14

Isomap is the simplest non-linear dimensionality reduction method that uses *graph distance* as an approximation of *geodesic distance* (the latter is the distance between points measured on the embedding manifold). This measures the distance between two points by constraining the path between them to be made up of short jumps between neighbourhoods (i.e. one forms a graph where nodes are connected only if they are close together or by choosing the K nearest neighbours). Isomap is a linear projection that uses graph distance in the projected space. There is an implementation in sklearn.

Locally-linear embedding also begins by finding a set of the nearest neighbors of each point. It then computes a set of weights for each point that best describes the point as a linear combination of its neighbors. Finally, it uses an eigenvector-based optimization technique to find the low-dimensional embedding of points, such that each point is still described with the same linear combination of its neighbors. LLE tends to handle non-uniform sample densities poorly because there is no fixed unit to prevent the weights from drifting as various regions differ in sample densities.

2.4 Slide 15

Isomap and LLE produce solutions that provide little insight into the class structure of the data. The map constructed by Sammon mapping is significantly better, since it models many of the members of each class fairly close together, but none of the classes are clearly separated in the Sammon map. In contrast, t-SNE does a much better job of revealing the natural classes in the data. Some individuals have their ten images split into two clusters, usually because a subset of the images have the head facing in a significantly different direction, or because they have a very different expression or glasses. For these individuals, it is not clear that their ten images form a natural class when using Euclidean distance in pixel space.

2.5 Slide 16

For many of the 20 objects, t-SNE accurately represents the one-dimensional manifold of viewpoints as a closed loop. For objects which look similar from the front and the back, t-SNE distorts the loop so that the images of front and back are mapped to nearby points. For the four types of toy car in the COIL-20 data set (the four aligned 'sausages'in the bottom-left of the t- SNE map), the four rotation manifolds are aligned by the orientation of the cars to capture the high similarity between different cars at the same orientation. This prevents t-SNE from keeping the four manifolds clearly separate. The figure also reveals that the other three techniques are not nearly as good at cleanly separating the manifolds that correspond to very different objects. In addition, Isomap and LLE only visualize a small number of classes from the COIL-20 data set, because the data set comprises a large number of widely separated submanifolds that give rise to small connected components in the neighborhood graph.

2.6 Slide 20

The gradient descent is initialized by sampling map points randomly from an isotropic Gaussian with small variance that is centered around the origin.

In addition, in the early stages of the optimization, Gaussian noise is added to the map points after each iteration. Gradually reducing the variance of this noise performs a type of simulated annealing that helps the optimization to escape from poor local minima in the cost function. If the variance of the noise changes very slowly at the critical point at which the global structure of the map starts to form, SNE tends to find maps with a better global organization. Unfortunately, this requires sensible choices of the initial amount of Gaussian noise and the rate at which it decays. Moreover, these choices interact with the amount of momentum and the step size that are employed in the gradient descent. It is therefore common to run the optimization several times on a data set to find appropriate values for the parameters.

2.7 Slide 22

We use a Student t-distribution with a single degree of freedom, because it has the particularly nice property that $1+\|\mathbf{y}_i-\mathbf{y}_j\|^2)^{-1}$ approaches an inverse square law for large pairwise distances $\|\mathbf{y}_i-\mathbf{y}_j\|$ in the low-dimensional map. This makes the map's representation of joint probabilities (almost) invariant to changes in the scale of the map for map points that are far apart. It also means that large clusters of points that are far apart interact in just the same way as individual points, so the optimization operates in the same way at all but the finest scales.

3 UMAP

3.1 Slide 5

Thus n represents some degree of trade-off between fine-grained and large-scale manifold features: smaller values will ensure detailed manifold structure is accurately captured (at a loss of the 'big picture' view of the manifold), while larger values will capture large scale manifold structures, but at a loss of fine detail structure which will get averaged out in the local approximations. With smaller n values the manifold tends to be broken into many small connected components (care needs to be taken with the spectral embedding for initialization in such cases).

3.2 Slide 6

Low values of n spuriously interpret structure from the random sampling noise in the toy dataset.

Reviewing Figure 3.1, more of the loops in the COIL20 dataset are kept intact, including the intertwined loops. Similarly the global relationships among different digits in the MNIST digits dataset are more clearly captured with 1 (red) and 0 (dark red) at far corners of the embedding space, and 4,7,9 (yellow, sea-green, and violet) and 3,5,8 (orange, chartreuse, and blue) separated as distinct clumps of similar digits. In the Fashion MNIST dataset the distinction between clothing (dark red, yellow, orange, vermillion) and footwear (chartreuse, sea-green, and violet) is made more clear. Finally, while both t-SNE and UMAP capture groups of similar word vectors, the UMAP embedding arguably evidences a clearer global structure among the various word clusters.

References

[Lowe and Tipping, 1996] Lowe, D. and M. E. Tipping 1996. Feed-forward neural networks and topographic mappings for exploratory data analysis. *Neural Computing and Applications* **4**, 83–95.

[Rosenblatt, 1956] Rosenblatt, M. 1956. Remarks on some non-parametric estimates of a density function. *Ann. Math. Statist.* **27**, 832–837.

[Shapiro, 1969] Shapiro, J. S. 1969. *Smoothing and Approximation of Functions*. New York: Van Nostrand Reinhold.

[Tipping, 1996] Tipping, M. E. 1996. Topographic Mappings and Feed-Forward Neural Networks. Ph.D. thesis, Aston University, Birmingham, UK.

[Tipping and Lowe, 1997] Tipping, M. E. and D. Lowe 1997. Shadow targets: A novel algorithm for topographic projections by radial basis functions. In *Proceedings of the International Conference on Artificial Neural Networks*, Volume 440, pp. 7–12. IEE.

[Yamato, 1972] Yamato, H. 1972. Some statistical properties of estimators of density and distribution functions. *Bull. Math. Stat.* **19**, 113–131.

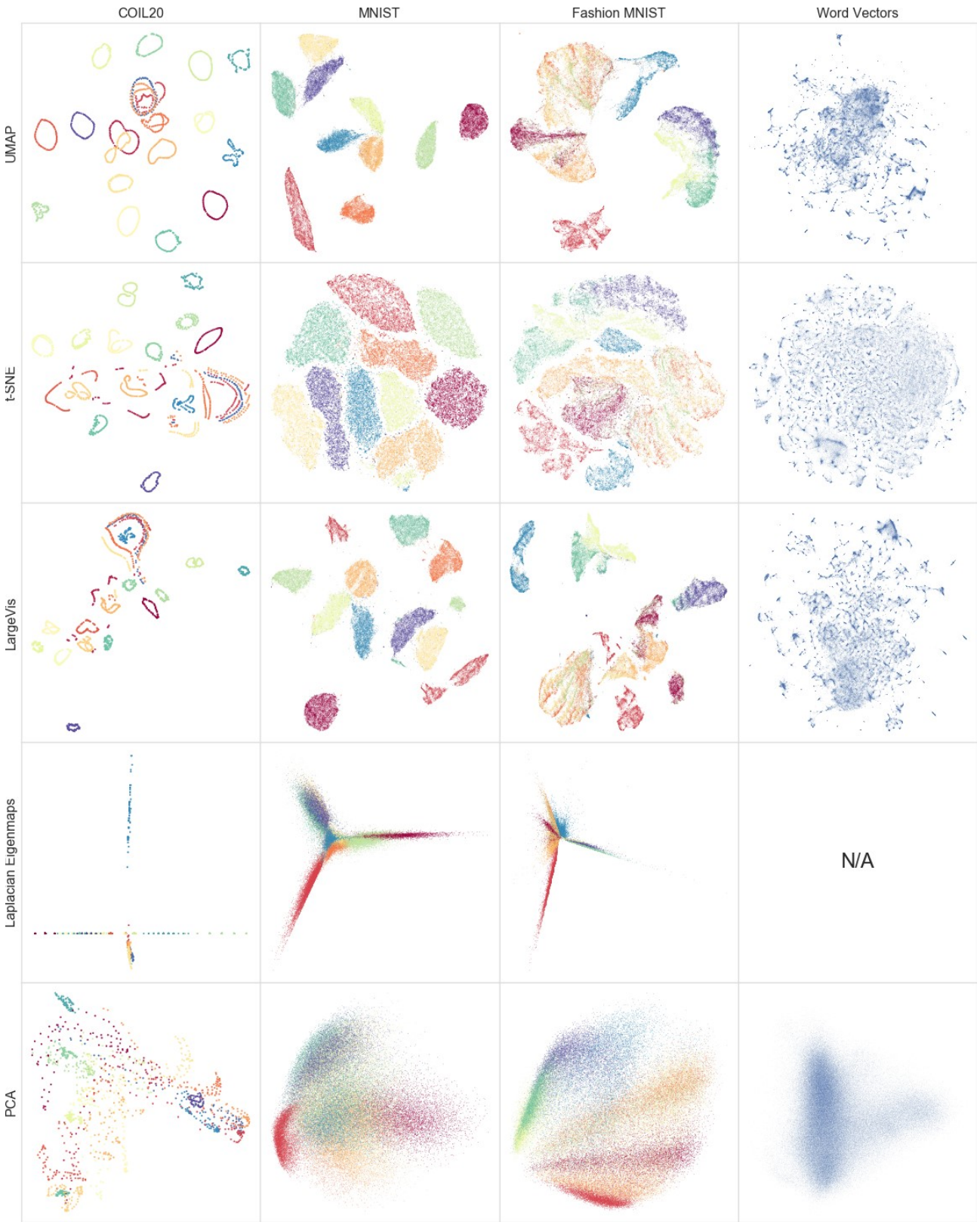

Figure 3.1: Comparison of several dimension reduction algorithms. We note that UMAP successfully reflects much of the large scale global structure that is well represented by Laplacian Eigenmaps and PCA (particularly for MNIST and Fashion-MNIST), while also preserving the local fine structure similar to t-SNE and LargeVis.