

CSE 6740 Lecture 10

How Can I Reduce/Relate the Features? (Dimension Reduction)

Alexander Gray

agray@cc.gatech.edu

Georgia Institute of Technology

Today

1. Dimensionality Reduction: Linear
2. Dimensionality Reduction: Nonlinear

Dimensionality Reduction: Linear

One of the major unsupervised learning tasks.

Dimensionality Reduction

“Make the data kind of the same, except with fewer columns.” Why? Sometimes:

- Computational reasons (reduce D)
- Statistical reasons (e.g. remove noise)
- Mainly: Visualization/understanding reasons (see data in 2-D or 3-D, or identify fundamental underlying variables)

Principal Components Analysis

As usual, let's start with the simplest case, which is linear. We want to map vectors $x \in \mathbb{R}^D$ to vectors $z \in \mathbb{R}^{D'}$ where $D' < D$.

We can always represent x as a linear combination of a set of D orthonormal vectors u_d ,

$$x = \sum_{d=1}^D z_d u_d \quad (1)$$

where the vectors u_d satisfy the orthonormality relation

$$u_d^T u_{\tilde{d}} = \delta_{d\tilde{d}} \quad (2)$$

where the Kronecker delta $\delta_{d\tilde{d}} = 1$ if $d = \tilde{d}$ and 0 otherwise.

Principal Components Analysis

The coefficients z_d have the form

$$z_d = u_d^T x \quad (3)$$

which can be regarded as a simple rotation of the coordinate system from the original x 's to a new set of coordinates given by the z 's.

Now suppose we retain only $D' < D$ of the basis vectors u_d , so that each vector x is approximated by

$$\hat{x} = \sum_{d=1}^{D'} z_d u_d + \sum_{d=D'+1}^D b_d u_d. \quad (4)$$

Principal Components Analysis

The error in the approximation is

$$x - \hat{x} = \sum_{d=D'+1}^D (z_d - b_d)u_d. \quad (5)$$

We can minimize the sum of squared errors over the whole dataset:

$$E_{D'} = \frac{1}{2} \sum_{i=1}^N \|x_i - \hat{x}_i\|^2 \quad (6)$$

$$= \frac{1}{2} \sum_{i=1}^N \sum_{d=D'+1}^D (z_d - b_d)^2. \quad (7)$$

Principal Components Analysis

Setting the derivative of $E_{D'}$ with respect to b_d to 0 we get

$$b_d = \frac{1}{N} \sum_{i=1}^N z_{id} = u_d^T \bar{x} \quad (8)$$

where $\bar{x} = \frac{1}{N} \sum_{i=1}^N x_i$. For optimal coefficients b_d then,

$$E_{D'} = \frac{1}{2} \sum_{d=D'+1}^D \sum_{i=1}^N \left[u_d^T (x_i - \bar{x}) \right]^2 \quad (9)$$

$$= \frac{1}{2} \sum_{d=D'+1}^D u_d^T \Sigma u_d \quad (10)$$

where $\Sigma = \sum_{i=1}^N (x_i - \bar{x})(x_i - \bar{x})^T$ is the covariance matrix.

Principal Components Analysis

We now need to minimize $E_{D'}$ with respect to the choice of basis vectors u_d . Some linear algebra shows that the minimum occurs when the basis vectors satisfy

$$\Sigma u_d = \lambda_d u_d \quad (11)$$

so that they are the eigenvectors of the covariance matrix. Some algebra allows us to rewrite the sum-of-squares error as

$$E_{D'} = \frac{1}{2} \sum_{d=D'+1}^D \lambda_d. \quad (12)$$

Thus, the minimum error is obtained by discarding the $D - D'$ smallest eigenvalues and their corresponding eigenvectors.

Principal Components Analysis

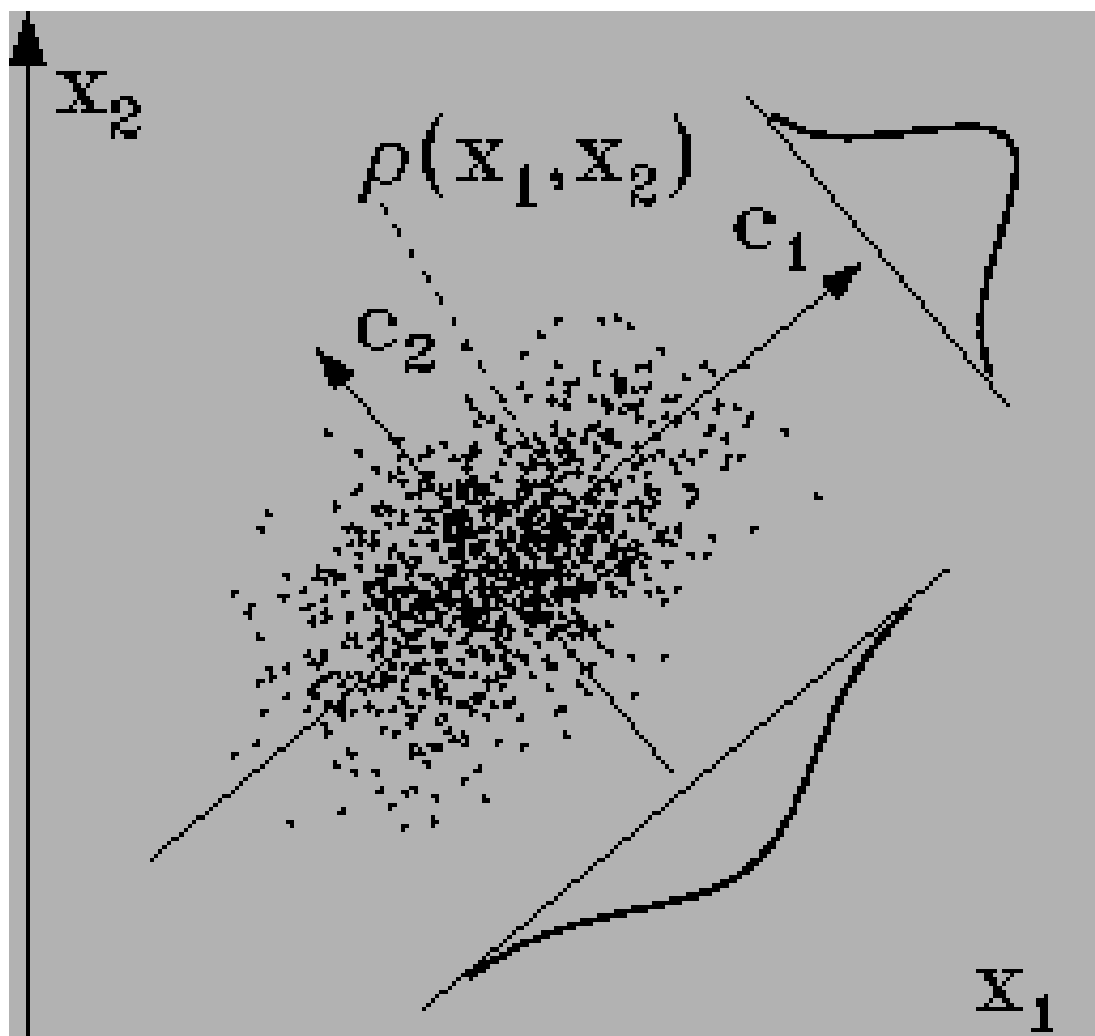
The eigenvectors u_d are called the *principal components* and the method is called *principal components analysis* (PCA) or the *Karhunen-Loeve transform*.

The first principal component captures the largest amount of variance. The second captures the second largest amount of variance, and so on.

It can be shown that the solution is the best possible D' -rank approximation to the data in several senses, including the Frobenius norm.

Principal Components Analysis

One way to perform PCA: First de-mean the data (subtract the mean \bar{x} from each of the data x_i), then compute the covariance matrix, then find its eigenvectors.



PCA and SVD

A different way to perform PCA is to construct the *singular value decomposition* (SVD) of the de-meanned data \mathbf{X} :

$$\mathbf{X} = \mathbf{U}\mathbf{\Lambda}\mathbf{V}^T \quad (13)$$

where \mathbf{U} is an $N \times D$ orthogonal matrix ($\mathbf{U}^T\mathbf{U} = \mathbf{I}$) containing the eigenvectors, \mathbf{V} is a $D \times D$ orthogonal matrix ($\mathbf{V}^T\mathbf{V} = \mathbf{I}$), and $\mathbf{\Lambda}$ is a $D \times D$ diagonal matrix containing the eigenvalues in order from largest to smallest. The columns of $\mathbf{U}\mathbf{\Lambda}$ contain the principal components of \mathbf{X} .

We map from x to its rank D' reconstruction by $\mathbf{V}'\mathbf{V}'^T x$ where \mathbf{V}' is the first D' columns of \mathbf{V} .

PCA as a Latent-Variable Model

The singular value decomposition $\mathbf{X} = \mathbf{U}\mathbf{\Lambda}\mathbf{V}^T$ has a latent-variable interpretation. Writing $\mathbf{S} = \sqrt{N}\mathbf{U}$ and $\mathbf{A}^T = \mathbf{\Lambda}\mathbf{V}^T / \sqrt{N}$, we have

$$\mathbf{X} = \mathbf{S}\mathbf{A}^T \quad (14)$$

i.e. each of the columns of \mathbf{X} is a linear combination of the columns of \mathbf{S} :

$$X_d = a_{d1}S_1 + a_{d2}S_2 + \cdots + a_{dD}S_D \quad (15)$$

or simply

$$\mathbf{X} = \mathbf{A}\mathbf{S}. \quad (16)$$

PCA as a Latent-Variable Model

Since U is orthogonal, and the columns of X , and thus U each have mean zero, this implies that the columns of S have zero mean, are uncorrelated, and have unit variance. In other words each variable X_d is represented as an expansion in the uncorrelated, unit variance variables S_d .

Unfortunately given any orthogonal $D \times D$ matrix R , we can write

$$X = AS \quad (17)$$

$$= AR^T RS \quad (18)$$

$$= \tilde{A}\tilde{S}, \quad (19)$$

so there is no unique set of latent variables which can be determined, *i.e.* the model is unidentifiable.

PCA as a Latent-Variable Model

We can make the model generative by adding a zero-mean noise term:

$$X = \mathbf{A}S + \epsilon, \quad (20)$$

modeling S and ϵ as Gaussian random variables. Then this *probabilistic PCA* corresponds to maximum likelihood estimation. (In fact the EM algorithm can be applied, though it is not the most numerically stable solution.)

If S is a vector of length $D' < D$, the model is called *factor analysis*.

Principal Components Analysis

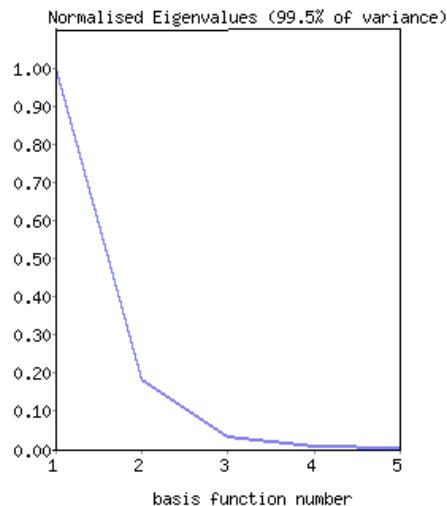
Task: Dimension reduction (can be cast as density estimation or multiple-output regression).

Model class: Linear combinations of features (parametric).

Loss: Squared error (in $(x - \hat{x})$), or likelihood.

Optimizer: Eigenvalue methods, EM algorithm (numerically unstable), online methods.

Generalization mechanism: Eyeball (usual) or cross-validation for number of components.



Independent Components Analysis

In *independent component analysis* (ICA) the model has the same form as in PCA:

$$X = AS, \quad (21)$$

except that now the S entries are assumed to be *statistically independent* rather than uncorrelated. Independence is a stronger condition. Lack of correlation determines the second-order cross-moments (covariances) of a multivariate distribution, while in general independence determines all of the cross-moments.

Independent Components Analysis

These extra moment conditions allow us to identify the elements of A uniquely. Since the Gaussian is determined by its second moments alone, it is the exception; Gaussian independent components are unidentifiable. We must therefore seek non-Gaussian independent components.

We wish to find the mixing matrix A in $X = AS$. First we whiten X so that $\text{cov}(X) = I$, typically by the SVD as in PCA. This implies A is orthogonal, since $\text{cov}(S) = I$.

Solving the ICA problem amounts to finding an orthogonal A such that the components of the vector random variable $S = A^T X$ are independent and non-Gaussian.

Independent Components Analysis

How to measure independence? In other words if we have some mixing matrix A how can we score the transformation $Z = A^T X$?

We can use the Kullback-Liebler divergence between the joint density $f(z)$ of Z and its independence version $\prod_d f_d(z_d)$ where $f_d(z_d)$ is the marginal density of Z_d .

Independent Components Analysis

We can express this in terms of quantities with names from information theory. The *differential entropy* H of a random variable Z with density $f(z)$ is given by

$$H(Z) = - \int f(z) \log f(z) dz. \quad (22)$$

Among all random variables with equal variance, Gaussian variables have the maximum entropy. The *mutual information* between the components of the random vector Z is another way to write the K-L divergence we have in mind:

$$I(Z) = \sum_{d=1}^D H(Z_d) - H(Z). \quad (23)$$

Independent Components Analysis

Now if X has covariance \mathbf{I} and $Z = \mathbf{A}^T X$ with \mathbf{A} orthogonal, then

$$I(Z) = \sum_{d=1}^D H(Z_d) - H(X) - \log |\det \mathbf{A}| \quad (24)$$

$$= \sum_{d=1}^D H(Z_d) - H(X). \quad (25)$$

We seek the \mathbf{A} that minimizes $I(Z) = I(\mathbf{A}^T X)$, *i.e.* that leads to the most independence between its components. This is equivalent to minimizing the sum of the entropies of the separate components of Z , which it turn is equivalent to maximizing their departures from Gaussianity.

Independent Components Analysis

ICA can do things PCA cannot. For example, *blind source separation*, or un-mixing a recording of D simultaneous conversations.

Task: Dimension reduction (can be cast as density estimation or multiple-output regression).

Model class: Linear combinations of original dimensions (parametric).

Loss: Likelihood.

Optimizer: Unconstrained optimization methods.

Generalization mechanism: Eyeball (usual) or cross-validation for number of components.

Dimensionality Reduction: Nonlinear

Let's consider how to possibly go beyond a linear parametric model.

Multidimensional Scaling

Both PCA and ICA map points $\{x_i\} \in \mathbb{R}^D$ to $\{z_i\} \in \mathbb{R}^{D'}$. Here is a different approach which does this, in a more direct way, called *multidimensional scaling* (MDS).

Given the matrix of pairwise distances (or dissimilarities) Δ_{ij} between points x_i and x_j in the original space, we wish to find points $\{z_i\}$ having distances δ_{ij} in the lower-dimensional space which are as close as possible to the original distances.

Multidimensional Scaling

In other words we want to find

$$\min_{\{z_i\}} \sum_{i \neq j} (\Delta_{ij} - \delta_{ij})^2 \quad (26)$$

or

$$\min_{\{z_i\}} \sum_{i \neq j} (\|x_i - x_j\|_D - \|z_i - z_j\|_{D'})^2. \quad (27)$$

This is called *least squares* or *Kruskal-Shepard* scaling. There are many variations on the objective function, leading to methods with different names.

Multidimensional Scaling

In *classical* scaling, we use similarities instead of dissimilarities, such as the centered inner product

$$\Delta_{ij} = \langle x_i - \bar{x}, x_j - \bar{x} \rangle \quad (28)$$

which we also use in the lower-dimensional space. The solution to this is an eigenvalue problem, in fact equivalent to PCA.

Classical scaling is not equivalent to least-squares scaling, since inner products rely on a choice of origin while pairwise distances do not. A set of inner products determines a set of pairwise distances but not vice versa.

Multidimensional Scaling

Notice that the input is a dissimilarity matrix, not point coordinates. Non-metric dissimilarities are possible. Unlike PCA and ICA, there is no explicit mapping from x 's to z 's.

Task: Dimension reduction.

Model class: Possibly nonparametric (general case).

Loss: Squared error (in $(\|x_i - x_j\| - \|\hat{x}_i - \hat{x}_j\|)$).

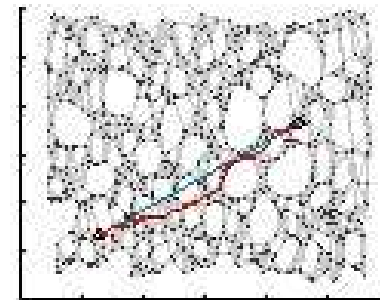
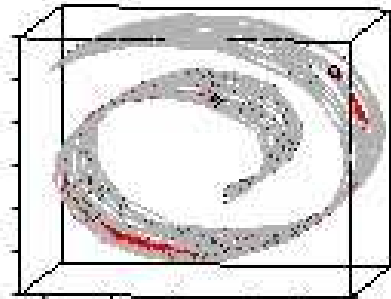
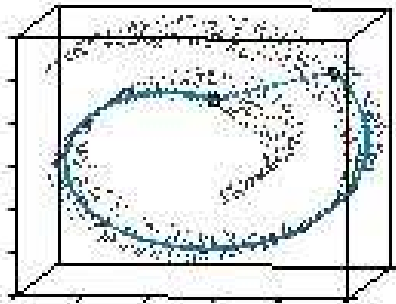
Optimizer: Unconstrained optimization methods.

Generalization mechanism: Eyeball (usual) or cross-validation for number of components.

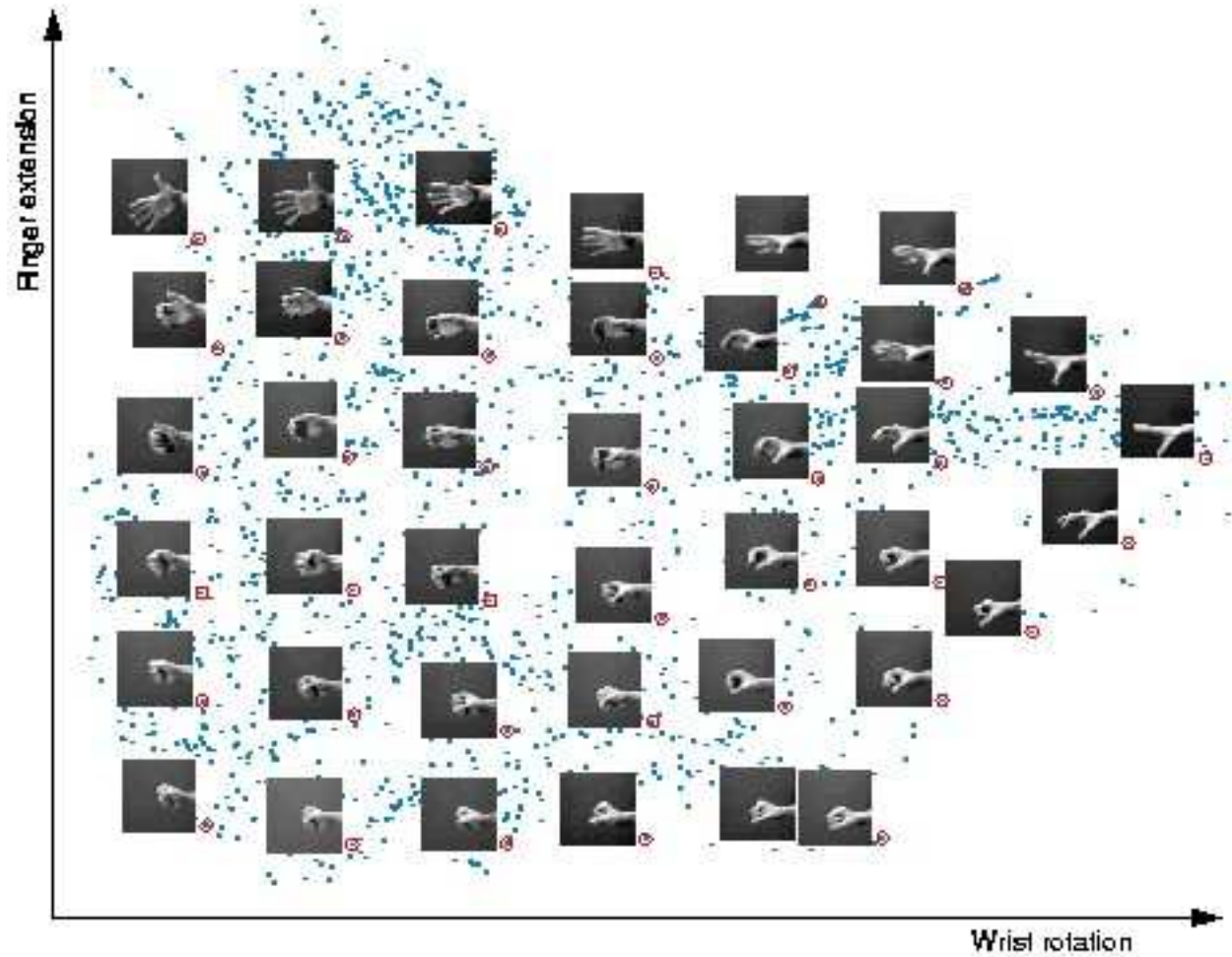
Evaluation algorithm: Generalized N -body methods.

Manifolds and Embedding

A *manifold* is a formal object in geometry. In machine learning the term is used loosely to refer to the vicinity of the data cloud, which typically is well-described in some lower-dimensional subspace. We say it is *embedded* within the D -dimensional space.



Manifolds and Embedding



IsoMap

The Swiss roll example motivates the need for a more careful notion of distance.

Define the neighborhood of x to be the K nearest points to it, or all the points within radius h of it. Define a graph which links all neighboring points, having weights on the arcs equal to the Euclidean distances. The *geodesic* distance between two points is approximated by the sum of the arc lengths along the shortest path linking both points.

The *IsoMap* method is MDS using this distance, *i.e.* find the eigenvalues of the matrix of geodesic distances and use the first D' dimensions.

IsoMap

This method can unroll the Swiss roll.

Task: Dimension reduction.

Model class: Possibly nonparametric.

Loss: Squared error (in $(\|x_i - x_j\| - \|\hat{x}_i - \hat{x}_j\|)$).

Optimizer: Dijkstra's algorithm for shortest paths.
Eigenvalue methods.

Generalization mechanism: Eyeball (usual) or cross-validation for number of components.

Evaluation algorithm: Generalized N -body methods.

Locally Linear Embedding

Here's another way to effectively have a nonlinear model. The idea of *locally linear embedding* (LLE) is to model each point as a linear combination of its local neighbors:

$$\hat{x} = \sum_k w_k x_k \quad (29)$$

where the x_k are the K nearest neighbors of x and the weights w_k sum to 1. The weights, because they are invariant to rotations, rescalings, and translations of the data, capture local geometry in a way that doesn't depend on a particular frame of reference.

Locally Linear Embedding

These weights are first found as the weights which minimize

$$\sum_i (x_i - \hat{x}_i)^2 = \sum_i \left(x_i - \sum_k w_{ik} x_k \right)^2. \quad (30)$$

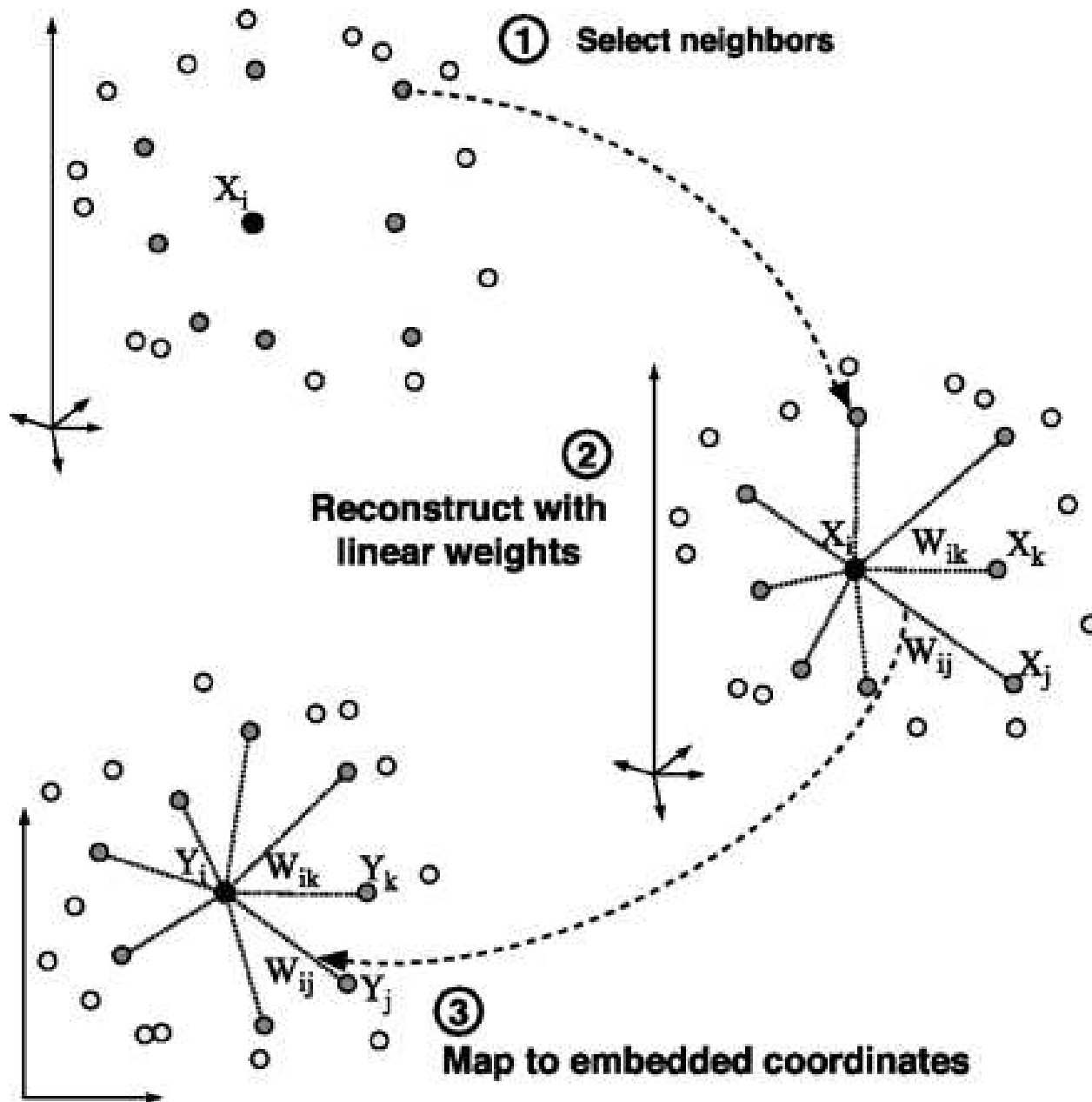
This becomes a separate minimization for each data point.

Now, with these weights fixed, we find points $\{z_i\}$ in the lower-dimensional space which minimize

$$\sum_i (z_i - \hat{z}_i)^2 = \sum_i \left(z_i - \sum_k w_{ik} z_k \right)^2. \quad (31)$$

This is an eigenvalue problem.

Locally Linear Embedding



Locally Linear Embedding

Note that K is a critical parameter of the method.

Task: Dimension reduction.

Model class: Possibly nonparametric.

Loss: Squared error (in $(\|x_i - x_j\| - \|\hat{x}_i - \hat{x}_j\|)$).

Optimizer: Eigenvalue methods, sparse.

Generalization mechanism: Eyeball (usual) or cross-validation for number of components.

Evaluation algorithm: Generalized N -body methods.

Kernel PCA

There are several variations on this theme. In particular, *Hessian eigenmaps* improves upon LLE's behavior - for example it can preserve holes in the manifold.

Kernel PCA is what you get if you kernelize PCA. IsoMap, LLE, and most other manifold methods can be seen as special cases of kernel PCA using different kernels.

Main Things You Should Know

- What PCA and ICA are
- What MDS and IsoMap are
- What LLE is

Sample Final Questions

1. (T/F) Dimension reduction is an unsupervised learning task.
2. (T/F) PCA has a unique solution.
3. (T/F) PCA can be computed using the singular value decomposition (SVD).
4. (T/F) The input to multidimensional scaling is a dissimilarity matrix.