**Using Kernel Principal component analysis for nonlinear mappings**

Using kernel PCA , we will learn how to transform data that is not linearly separable onto a new , lower-dimensional subspace that is suitable for linear classifiers.

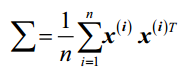
**Kernel functions and the kernel trick**

Via kernel PCA we perform a nonlinear mapping that transforms the data onto a higher-dimensional space and use standard PCA in this higher-dimensional space to project the data back onto a lower-dimensional space where the samples can be separated by a linear classifier.

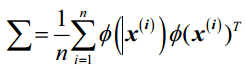
Downside: it is computationally very expensive.

Using kernel trick, we can compute the similarity between two high-dimension feature vectors in the original feature space.

The general equation to calculate standard PCA covariance matrix  is :



Kernel Principal Component Analysis replace the dot products between samples in the original feature space by the nonlinear feature combinations via :

 (2)

Its eigenvalues and eigenvectors are given by

 (3)

Where k = 1,2,…,M. From Eq.(2) and Eq.(3) we have

 (4)

Which can be rewritten as

 (5)

Now by substituting vk in Eq.(4) with Eq.(20), we have

 (6)

If we define the function

 (7)

And multiply both sides of Eq.(6) by , we have

 (8)

We can use the matrix notation

 (9)

Where

(similarity) (10)

And is the N-dimensional column vector of aki:

 (11)

 can be solved by

 (12)

And the resulting kernel principal components can be calculated using

 (13)

If the projected dataset  does not have zero mean, we can use the Gram matrix  to substitute the kernel matrix K. The Gram matrix is given by

 (14)

Where 1N is the NxN matrix with all elements equal to 1/N.

The most commonly used kernels are as follows:

1. The polynomial kernel:

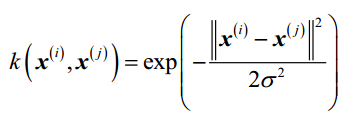


Here, it the threshold and P is the power that has to be specified by the user.

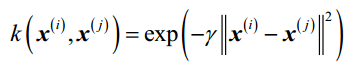
1. The hyperbolic tangent (sigmoid) kernel:



1. The Radial Basis Function (RBF) or Gaussian kernel :



It is also written as follows:



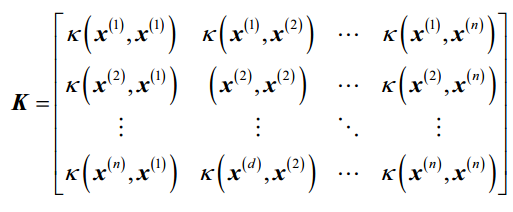
There is no universal value for the tuning parameter lamda that works well for different datasets. To find a lamda value that is appropriate for a given problem requires experimentation.

The power of kernel methods is that we do not have to compute  explicitly.

The standard steps of kernel PCA dimensionality reduction can be summarized as:

1. Construct the kernel (similarity) matrix ***K*** from the training data set {xi} using Eq. (10);

We do this for each pair of samples:



For example , if our dataset contains 100 training samples, the symmetric kernel matrix of the pair-wise similarities would be 100x100 dimensional.

1. Compute the Gram matrix  using Eq.(29)
2. Using Eq (12) to solve for the vectors ai (substitute ***K*** with )
3. Compute the kernel principal components  using Eq. (13).

It is worth nothing that kernel PCA, in contrast to standard PCA, is a memory-based method , which means that we have to reuse the original training set each time to project new samples. We have to calculate the pairwise kernel(similarity) between each ith sample in the training dataset and the new sample .After calculating the similarity between the new samples and the samples in the training set, we have to normalize the eigenvector ***a*** by its eigenvalue.

Source :machine-learning ,github

Reference:

1. Python machine learning
2. Kernel Principal Component Analysis and its Applications in Face Recognition and Active Shape Models, Quan Wang.