Lecture 6 Sept. 25-2006

Data Visualization

STAT 442 / 890, CM 462

Lecture: Ali Ghodsi

1 Dual PCA

It turns out that the singular value decomposition also allows us to formulate the principle components algorithm entirely in terms of dot products between data points and limit the direct dependence on the original dimensionality n. This fact will become important below.

Assume that the dimensionality n of the $n \times t$ matrix of data X is large (i.e., $n >> t$). In this case, Algorithm 1 (Table **??**) is impractical. We would prefer a run time that depends only on the number of training examples t, or that at least has a reduced dependence on n.

Note that in the SVD factorization $X = U\Sigma V^T$, the eigenvectors in U corresponding to nonzero singular values in Σ (square roots of eigenvalues) are in a one-to-one correspondence with the eigenvectors in V.

Now assume that we perform dimensionality reduction on U and keep only the first d eigenvectors, corresponding to the top d nonzero singular values in Σ. These eigenvectors will still be in a one-to-one correspondence with the first d eigenvectors in V:

$$X\,V \;=\; U\,\Sigma$$

where the dimensions of these matrices are:

$$\begin{array}{cccc} X & U & \Sigma & V \\ n \times t & n \times d & d \times d & t \times d \\ & & \text{diagonal} & \end{array}$$

Crucially, Σ is now square and invertible, because its diagonal has nonzero entries. Thus, the following conversion between the top d eigenvectors can be derived:

$$U \;=\; X\, V\, \Sigma^{-1} \tag{1}$$

Replacing all uses of U in Algorithm 1 with $XV\Sigma^{-1}$ gives us the dual form of PCA, Algorithm 2 (see Table 1). Note that in Algorithm 2 (Table 1), the steps of *"Reconstruct training data"* and *"Reconstruction test example"* still depend on n, and therefore still will be impractical in the case that the original dimensionality n is very large. However all other steps can be done conveniently in the run time that depends only on the number of training examples t.

Algorithm 2

Recover basis: Calculate $X^\top X$ and let V = eigenvectors of $X^\top X$ corresponding to the top d eigenvalues. Let Σ = diagonal matrix of *square roots* of the top d eigenvalues.

Encode training data: $Y = U^\top X = \Sigma V^\top$ where Y is a $d \times t$ matrix of encodings of the original data.

Reconstruct training data: $\hat{X} = UY = U\Sigma V^\top = XV\Sigma^{-1}\Sigma V^\top = XVV^\top$.

Encode test example: $y = U^\top x = \Sigma^{-1}V^\top X^\top x = \Sigma^{-1}V^\top X^\top x$ where y is a d dimensional encoding of x.

Reconstruct test example: $\hat{x} = Uy = UU^\top x = XV\Sigma^{-2}V^\top X^\top x = XV\Sigma^{-2}V^\top X^\top x$.

Table 1: *Dual PCA Algorithm*

2 Kernel PCA

Consider a feature space $\mathcal{H}$ such that:

$$\Phi : x \to \mathcal{H}$$

$$x \mapsto \Phi(x)$$

In kernel PCA we map from the original space X into a feature space $\mathcal{H}$ and then into the lower dimensional or embeded space Y.

$$X \to \mathcal{H} \to Y$$

The goal is to reduce the dimensionality from the space X to the dimensionality of space Y by passing through $\mathcal{H}$ without having to know $\Phi(x)$ exactly.

Suppose $\sum_i^t \Phi(x_i) = 0$ (we will return to this point below, and show how this condition can be satisfied in a Hilbert space). This allows us to formulate the kernel PCA objective as follows:

$$\min \sum_i^t ||\Phi(x_i) - U_q U_q^T \Phi(x_i)||$$

By the same argument used for PCA, the solution can be found by SVD:

$$\Phi(X) = U\Sigma V^T$$

where U contains the eigenvectors of $\Phi(X)\Phi(X)^T$. Note that if $\Phi(X)$ is $\mathbf{n} \times t$ and the dimensionality of the feature space $\mathbf{n}$ is large, then U is $\mathbf{n} \times \mathbf{n}$ which will make PCA impractical.

To reduce the dependence on $\mathbf{n}$, first assume that we have a kernel $K(\cdot, \cdot)$ that allows us to compute $K(x, y) = \Phi(x)^\top \Phi(y)$. Given such a function, we can then compute the matrix $\Phi(X)^\top \Phi(X) = K$ efficiently, without computing $\Phi(X)$ explicitly. Crucially, K is $t \times t$ here and does not depend on $\mathbf{n}$. Therefore it can be computed in a run time that depends only on t. Also, note that PCA can be formulated entirely in terms of dot products between data points (Algorithm 2 represented in Table 1). Replacing dot products in Algorithm 2 (1) by kernel function K, which is in fact equivalent to the inner product of a Hilbert space yields to the Kernel PCA algorithm.

The algorithm for Kernel PCA is similar to the one for Dual PCA except that in the case of Kernel PCA neither the training data nor the test data can be reconstructed.

Algorithm 3

Recover basis: Calculate $K = \Phi(X)^\top \Phi(X)$ using the kernel K and let $V =$ eigenvectors of $\Phi(X)^\top \Phi(X)$ corresponding to the top d eigenvalues. Let $\Sigma =$ diagonal matrix of *square roots* of the top d eigenvalues.

Encode training data: $Y = U^\top \Phi(X) = \Sigma V^\top$ where Y is a $d \times t$ matrix of encodings of the original data. This can be done since the calculation of Y does not depend on $\Phi(X)$

Reconstruct training data: $\hat{X} = UY = U\Sigma V^\top = \Phi(X)V\Sigma^{-1}\Sigma V^\top = \Phi(X)VV^\top$. $\Phi(X)$ is unknown so reconstruction **can not be done**.

Encode test example: $y = U^\top \Phi(x) = \Sigma^{-1}V^\top \Phi(X)^\top \Phi(x) = \Sigma^{-1}V^\top \Phi(X)^\top \Phi(x)$. This result can be calcuated because the quantity $\Phi(X)^\top \Phi(x) = K(X, x)$.

Reconstruct test example: $\hat{x} = Uy = UU^\top \Phi(x) = \Phi(X)V\Sigma^{-2}V^\top \Phi(X)^\top \Phi(x) = \Phi(X)V\Sigma^{-2}V^\top \Phi(X)^\top \Phi(x)$. $\Phi(X)$ is unknown so reconstruction **can not be done**.

Table 2: *Kernel PCA Algorithm*