Computational Pattern Analysis and Statistical Learning.

Practical Session 3.

John Shawe-Taylor Tijl De Bie Konstantin Tretyakov

November 17, 2006

Abstract

Yesterday we introduced important concepts such as regularization, duality and the kernel trick. Today, we will use the same approach to derive the kernel versions of some important algorithms, and apply them.
Keywords: normalizing, centering, PCA, K-means.

1 Normalizing and centering

Normalizing a kernel matrix In some cases, it is a good idea to normalize all samples to unit norm. This can be done easily on the sample matrix $\mathbf{X}$ itself:

$$\mathbf{X}_{\text{norm}} = \begin{pmatrix} \mathbf{x}_1^T\mathbf{x}_1 & 0 & \cdots & 0 \\ 0 & \mathbf{x}_2^T\mathbf{x}_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{x}_n^T\mathbf{x}_n \end{pmatrix}^{-1/2} \cdot \mathbf{X}. \tag{1}$$

Note that the diagonal matrix has $\mathbf{x}_i^T\mathbf{x}_i = \mathbf{K}_{ii}$ on its diagonal. Again, it turns out we can also compute the normalized kernel matrix $\mathbf{K}_{\text{norm}}$ corresponding to the kernel $\mathbf{K}$, without ever needing to know the data matrix $\mathbf{X}$ itself:

$$\begin{aligned} \mathbf{K}_{\text{norm}} &= \mathbf{X}_{\text{norm}}\mathbf{X}_{\text{norm}}^T \\ &= \text{diag}(\mathbf{x}_1^T\mathbf{x}_1\, \mathbf{x}_2^T\mathbf{x}_2\, \ldots\, \mathbf{x}_n^T\mathbf{x}_n)^{-1/2} \cdot \mathbf{X}\mathbf{X}^T \cdot \text{diag}(\mathbf{x}_1^T\mathbf{x}_1\, \mathbf{x}_2^T\mathbf{x}_2\, \ldots\, \mathbf{x}_n^T\mathbf{x}_n)^{-1/2} \\ &= \text{diag}(\mathbf{K}_{11}\, \mathbf{K}_{22}\, \ldots\, \mathbf{K}_{nn})^{-1/2} \cdot \mathbf{K} \cdot \text{diag}(\mathbf{K}_{11}\, \mathbf{K}_{22}\, \ldots\, \mathbf{K}_{nn})^{-1/2} \end{aligned} \tag{2}$$

It's easy to check that all diagonal entries of $\mathbf{K}_{\text{norm}}$, which are the norms of the normalized samples $\mathbf{X}_{\text{norm}}$, are equal to 1 indeed.

Centering a kernel matrix Given a sample $\mathbf{X}$, centering can be performed by subtracting the means of the features of $\mathbf{X}$ (corresponding to columns of $\mathbf{X}$):

$$\mathbf{X}_{\text{cent}} = \mathbf{X} - \frac{1}{n}\mathbf{1}\mathbf{1}^T\mathbf{X} \tag{3}$$

Thus, the matrix containing all inner products of the centered vectors $\mathbf{X}_{\text{cent}}$ is

$$\mathbf{X}_{\text{cent}}\mathbf{X}_{\text{cent}}^T = (\mathbf{X} - \frac{1}{n}\mathbf{1}\mathbf{1}^T\mathbf{X})(\mathbf{X} - \frac{1}{n}\mathbf{1}\mathbf{1}^T\mathbf{X})^T$$

which is equal to

$$\mathbf{X}_{\text{cent}}\mathbf{X}_{\text{cent}}^T = \mathbf{X}\mathbf{X}^T - \frac{1}{n}\mathbf{1}\mathbf{1}^T\mathbf{X}\mathbf{X}^T - \frac{1}{n}\mathbf{X}\mathbf{X}^T\mathbf{1}\mathbf{1}^T + \frac{1}{n^2}\mathbf{1}\mathbf{1}^T\mathbf{X}\mathbf{X}^T\mathbf{1}\mathbf{1}^T.$$

Also here we can use the kernel trick and replace $\mathbf{X}\mathbf{X}^T$ by the kernel matrix $\mathbf{K}$. Thus, the centered kernel matrix $\mathbf{K}_{\text{cent}}$ can be computed directly from the uncentered kernel matrix as:

$$\mathbf{K}_{\text{cent}} = \mathbf{K} - \frac{1}{n}\mathbf{1}\mathbf{1}^T\mathbf{K} - \frac{1}{n}\mathbf{K}\mathbf{1}\mathbf{1}^T + \frac{1}{n^2}\mathbf{1}\mathbf{1}^T\mathbf{K}\mathbf{1}\mathbf{1}^T \tag{4}$$

Exercise 1 (Normalizing and centering) *The files* `english.txt`, `french.txt`, `german.txt` *and* `italian.txt` *each contain 195 little texts in the corresponding languages (articles from the Swiss constitution). You can verify that i-th text in one language is a translation of the i-th text in the other languages. In this practical session we shall work with this data set.*

Now, there exist kernels specially designed for strings and texts that compute a certain similarity measure between two texts. Two of these kernels are the bag-of-words kernel *and the* 2-mer kernel*. Since computing them on these* $780 = 4 \cdot 195$ *texts would take a while, you don't have to do that here. The two kernel matrices are given precomputed to you in the files* `K_bow.tab` *and* `K_2mer.tab`[1]*. These kernel matrices were evaluated on all documents of all languages, and the ordering in which the documents appear is: english, french, german and italian, each language in the same ordering (so for an english document at position* $1 \leq i \leq 195$*, the german translation occurs at position* $2 \times 195 + i$*).*

a. *Have a look at both kernels by printing part of them on the screen. Do you notice that small texts generally give rise to smaller kernel values? This is undesirable in many applications: very often, the similarity between two texts should not be based on their length, but rather on how much they overlap relative to their length. To get rid of this artefact, you can normalize the kernels. Do this, and save the resulting normalized kernels as K_bow_n and K_2mer_n.*

b. *As a second preprocessing step, center the kernels and save the results as K_bow_nc and K_2mer_nc.*

[1] You can load them into Scilab with the `fscanfMat` command.

2 Principal Component Analysis (PCA)

Principal component analysis is a tool to find directions along which a sample represented by $\mathbf{X}$ or by a kernel matrix $\mathbf{K}$ has a large variance, i.e. is spread out a lot. We will first derive the standard, primal version of PCA. Subsequently we will derive and use the kernel formulation kPCA.

2.1 PCA, primal version

The variance along a direction $\mathbf{w}$ with $\|\mathbf{w}\|^2 = 1$ is given by $\mathbf{w}^T\mathbf{X}^T\mathbf{X}\mathbf{w}$. Since we are interested in finding the direction $\mathbf{w}^*$ with largest such variance, we want to solve the following optimization problem:

$$\mathbf{w}^* = \quad \text{argmax}_{\mathbf{w}} \quad \mathbf{w}^T\mathbf{X}^T\mathbf{X}\mathbf{w} \tag{5}$$

$$\text{s.t.} \qquad \|\mathbf{w}\|^2 = 1 \tag{6}$$

Exercise 2 (PCA, derivation) *Prove that the solution of this constrained optimization problem can be found by solving for the largest eigenvalue and corresponding eigenvector of the eigenvalue problem* $\mathbf{X}^T\mathbf{X}\mathbf{w} = \lambda\mathbf{w}$.

Whereas the optimization problem we started from only searches for the direction of largest variance, it is often interesting to keep more than one direction. Other directions of large variance are found as the second, third, etc... principal components, belonging to the second, third, etc... largest eigenvalues λ.

Exercise 3 (PCA, artificial example) *Randomly generate a centered two-dimensional cloud of points, that is more stretched out along some direction*[2]*. Perform PCA*[3]*. Plot the direction of largest variance on the scatter plot of the cloud of points.*

2.2 kPCA, the kernel formulation

The derivation of kPCA can be carried out along the same lines as the kRR derivation in yesterday's exercise session. It is left as an exercise (ask for help if you don't see how to do it):

Exercise 4 (kPCA, derivation) *Show that the kernel version of PCA (kPCA) is given by the eigenvalue problem:*

$$\mathbf{K}\boldsymbol{\alpha} = \lambda\boldsymbol{\alpha},$$

where $\mathbf{w} = \mathbf{X}^T\boldsymbol{\alpha}$.

[2]Hint:

```
rand('normal');
X = rand(100, 2);
X = X * [5 0; 0 1] * [cos(1) sin(1); -sin(1) cos(1)];
plot(X(:, 1), X(:, 2), '.');
```

[3]Hint: `[V, D] = spec(X'*X);`

Since (k)PCA is usually carried out on a centered data set, you can now use the kernel-centering algorithm you just wrote.

Exercise 5 (kPCA, example) *Perform kPCA on the kernel matrices you just normalized and centered in exercise 1. Compute the vectors $\boldsymbol{\alpha}_1$ and $\boldsymbol{\alpha}_2$ corresponding to the two largest eigenvalues. Project all samples on the directions corresponding to these dual vectors (remember that this projection can be carried out by making use of kernel evaluations only: $\mathbf{x}^T\mathbf{w} = \mathbf{x}^T\mathbf{X}\boldsymbol{\alpha} = \sum_i k(\mathbf{x}, \mathbf{x}_i)\boldsymbol{\alpha}_i$). Plot the dots corresponding to texts of different languages in different colors. If you want, you can plot the projections on some less important principal components for comparison.*

3 K-means Clustering

In the previous exercise you have seen that the texts in different languages naturally form distinct groups in the feature space. It is often of interest to detect such groups in the data in an *unsupervised* manner, which is known as *clustering*. K-means is probably the simplest algorithm for this task.

3.1 K-means, primal version

We shall group the data points into k groups, such that the deviation of the elements in each group l from its mean $\boldsymbol{\mu}_l$ is as small as possible. Let $\mathbf{x}_i$ be, as usual, the data points, and let $g(i)$ indicate a group to which a point $\mathbf{x}_i$ belongs. The mean $\boldsymbol{\mu}_l$ of group l is then

$$\boldsymbol{\mu}_l = \frac{1}{|\{i \,:\, g(i) = l\}|} \sum_{g(i)=l} \mathbf{x}_i.$$

We shall search for such a grouping $\mathbf{g}$, for which

$$\frac{1}{n} \sum_{i=1}^{n} \|\mathbf{x}_i - \boldsymbol{\mu}_{g(i)}\|^2 \tag{7}$$

is minimal.

Exercise 6 (K-means, alternative formulation) *In the lecture you saw an alternative formulation for K-means: find a set of vectors $\{\boldsymbol{\mu}_l\}$, that minimizes*

$$\pi_{\{\boldsymbol{\mu}_l\}}(\mathbf{X}) = \frac{1}{n} \sum_{i=1}^{n} \min_l \|\mathbf{x}_i - \boldsymbol{\mu}_l\|^2.$$

Show that it is equivalent to (7).

In contrast to the previous cases, minimization of (7) is a non-convex discrete optimization problem, the exact solution to which can't be found efficiently. However, it is easy to derive an approximate iterative algorithm. We shall start with some arbitrary grouping $\mathbf{g}$, and update it in a greedy manner similar to *hill-climbing*.

Algorithm 1 (K-means clustering)

1. *Start with arbitrary* $\mathbf{g}$*, calculate corresponding means* $\{\boldsymbol{\mu}_l\}$*.*
2. *Assume the means are fixed. Update* $\mathbf{g}$*:*

$$g'(i) = \operatorname*{argmin}_l \|\mathbf{x}_i - \boldsymbol{\mu}_l\|^2.$$

3. *Recalculate the means* $\{\boldsymbol{\mu}_l\}$ *for the new grouping.*

$$\boldsymbol{\mu}'_l = \frac{1}{|\{i \,:\, g(i) = l\}|} \sum_{g(i)=l} \mathbf{x}_i.$$

4. *Repeat until convergence.*

The algorithm always converges (and often does it rather quickly), but it is not guaranteed to find the global minimum.

3.2 K-means, the kernel formulation

In order to derive the kernel-version of K-means we adopt a dual representation for the means $\boldsymbol{\mu}_l$ and reformulate all the operations in terms of kernel evaluations.

The means $\boldsymbol{\mu}_l$ in the K-means algorithm can be naturally represented as linear combinations of the traning samples:

$$\boldsymbol{\mu}_l = \sum_{i=1}^{n} \mathbf{x}_i \alpha_l(i) = \mathbf{X}^T \boldsymbol{\alpha}_l. \tag{8}$$

The operations required for the K-means algorithm, computing distances and taking means, can be kernelized easily. The distance computation goes as follows:

$$\begin{aligned}
\|\mathbf{x}_i - \boldsymbol{\mu}_l\|^2 &= \|\mathbf{x}_i - \mathbf{X}^T\boldsymbol{\alpha}_l\|^2 = \mathbf{x}_i^T\mathbf{x}_i - 2\boldsymbol{\alpha}_l^T\mathbf{X}\mathbf{x}_i + \boldsymbol{\alpha}_l^T\mathbf{X}\mathbf{X}^T\boldsymbol{\alpha}_l \\
&= k(\mathbf{x}_i, \mathbf{x}_i) - 2\boldsymbol{\alpha}_l^T\mathbf{X}\mathbf{X}^T\mathbf{1}_i + \boldsymbol{\alpha}^T\mathbf{X}\mathbf{X}^T\boldsymbol{\alpha} \\
&= \mathbf{K}_{ii} - 2\boldsymbol{\alpha}_l^T\mathbf{K}\mathbf{1}_i + \boldsymbol{\alpha}^T\mathbf{K}\boldsymbol{\alpha},
\end{aligned}$$

where $\mathbf{1}_i$ is a vector containing 1 at position i and 0 otherwise.

Derivation of the dual representation of the mean of a set of samples is left as an exercise (examine equation (8) carefully).

Exercise 7 (Kernel K-means, example) *Perform kernel K-means on the Swiss constitution dataset. As clustering with the* bag-of-words *kernel is probably "too simple", so it is more interesting to use the* `K_2mer` *kernel. Do the clustering with* $k = 4$*. Does the algorithm correctly identify groups of texts corresponding to the 4 languages? What happens for other values of* k*? You can plot the PCA projections of the texts classified to different groups in different colors.*