Lecture 17: Perspectives on Kernel methods, feature maps & Kernel PCA.

We spent the last few lectures discussing Kernel methods & in particular kernel ridge regression. Our goal in this lecture is to take a closer look at kernel methods & the features & also connect them back to some of the methods we saw earlier in the course.

17.1 Kernel perspective vs. feature perspective

Recall our formulations of Kernel Ridge regression from Lec 16

(FR) minimize $\|A\beta - Y\|^2 + \lambda\|\beta\|^2$

(KR) minimize $\|\Theta \underline{a} - Y\|^2 + \lambda \underline{a}^T \Theta \underline{a}$

where $A_{ij} = F_j(\underline{x}_i)$ ← matrix of feature values

$$\Theta_{ij} = K(\underline{x}_i, \underline{x}_j)$$

Clearly problems (FR) & (KR) are equivalent, as a result of Mercer's thm & the representer thm. But the question is which one is more useful?

- Observe that (FR) & (KR) have diff. number of unknowns ie, different sizes.
  - If K has $J \geq 0$ features, ie,
    $$K(\underline{x}, \underline{x}') = \sum_{j=0}^{J-1} F_j(\underline{x}) F_j(\underline{x}')$$
    Then $\beta \in \mathbb{R}^J$.
  - If dataset $X \in \mathbb{R}^{d \times N} = \left[\underline{x}_0 \middle| \cdots \middle| \underline{x}_{N-1} \right]$ then $\underline{a} \in \mathbb{R}^N$.

This means if we have a large data set with $N \gg J$ then it makes sense to work with (FR). On the other hand many kernels, such as Gaussian kernel in last lecture have infinitely many features, $J = \infty$, then we are only left with (KR). Unless we can tolerate some errors.

Note: If K is a kernel s.t. $K(\underline{x}, \underline{x}') = \sum_{j=0}^{\infty} F_j(\underline{x}) F_j(\underline{x}')$ then $\hat{K}(\underline{x}, \underline{x}') = \sum_{j=0}^{J-1} F_j(\underline{x}) F_j(\underline{x}')$ is also a kernel.

In Summary: whether you prefer the kernel or feature perspective is up to you & the problem at hand.

17.2 Connecting Kernels to Fourier Series & Wavelets

Recall from our discussion of Fourier series that we modelled a signal $f:[0,2\pi] \to \mathbb{R}$ as

$$f(t) = \sum_{k=-N/2}^{N/2-1} \hat{f}_k \exp(ikt), \quad \hat{f}_k = \frac{1}{2\pi}\int_0^{2\pi} f(x)\exp(-ikt)\,dt.$$

Based on this model we can identify a kernel, called the Dirichlet Kernel.

$$K(t,s) = \sum_{k=-N/2}^{N/2-1} \exp(ik(t-s))$$

* This kernel is complex valued but that's ok. Our def^n. generalizes in a natural manner.

By definition the features of the Dirichlet Kernel are precisely

$$K(t,s) = \sum_{k=-N/2}^{N/2} \exp(ikt)\exp(-iks) = \sum_{k=-N/2}^{N/2} F_k(t)\,\overline{F_k}(s),$$

$$F_k(s) = \exp(iks)$$

which implies, the RKHS of the Dirichlet kernel is,

$$\mathcal{H}_K^N = \left\{ f : [0, 2\pi] \to \mathbb{R} \;\middle|\; f(t) = \sum_{k=N/2}^{N_2 - 1} c_k \exp(ikt), c_k \in \mathbb{C} \right\}.$$

In this light, we can view the DFT as solving the following interpolation problem.

find $\hat{f} \in \mathcal{H}_K^N$ s.t. $\hat{f}(t_j) = f(t_j)$ on a uniform grid $t_0 = 0, t_1 = \delta t, \ldots, t_{N-1} = 2\pi$.

We also have the interesting identity

$$\int_0^{2\pi} K(t, s) f(s) ds = \sum_{k=N/2}^{N_2 - 1} \hat{f}_k \exp(ikt)$$

the same image is true in case of the wavelet transform except that the resulting kernel does not have a closed form.

$\{\psi_{m,n}(t)\}_{m=0,n=0}^{M,N}$, order the wavelets in a long vector $\{\psi_j(t)\}_{j=0}^{J}$.

define the kernel $K(t,s) = \sum_{j=0}^{J} \psi_j(t) \psi_j(s)$.

then $\mathcal{H}_K^J := \{f:[0,2\pi] \to \mathbb{R} \mid f(t) = \sum_{j=0}^{J} c_j \psi_j(t)\}$.

In Summary: most linear methods that write a signal/function/data etc. in the form

$$f = \sum_j c_j F_j,$$

have a kernel interpretation & can be written as an optimization problem over an RKHS.

17.3 Feature maps, linear models & kernel PCA

There is yet another perspective on kernel methods that is very useful in data analysis & leads to surprisingly simple but powerful algorithms.

Recall: (FR)

minimize $\|A\beta - Y\|^2 + \lambda\|\beta\|^2$

which assumes the model

$$y(x) = \sum_{j=0}^{J-1} \beta_j F_j(x)$$

Simply put, the output y is a linear combination of $F_j(\underline{x})$.

Define the mapping $F: \mathbb{R}^d \to \mathbb{R}^J$

$$F(\underline{x}) = (F_0(\underline{x}), \ldots, F_{J-1}(\underline{x}))^T \in \mathbb{R}^J$$

Then the above model is equivalent to saying that $y = \beta^T z$ where $z \in \mathbb{R}^J$ is a new variable obtained by transforming $\underline{x}$ by $z = F(\underline{x})$.

The powerful idea here is that, if we have a good non-linear feature map (ie good kernel) then the output y is almost a linear function of $z = F(\underline{x})$, the transformed inputs!

We can now ask the following:

Can we design new algorithms by applying classic, linear methods to the z variables (aka in the feature space)? Answer is Yes!

We will briefly consider a simple example, kernel PCA.

Given data set $X = \left[\underline{x}_0 \middle| \underline{x}_1 \middle| \cdots \middle| \underline{x}_{N-1} \right] \in \mathbb{R}^{d \times N}$

standard PCA simply writes $X = U \Sigma V^T$ & takes columns $\underline{u}_j$ of U as the principal components of X.

In kernel PCA we first transform the data with the feature map

$$\underline{z}_n = F(\underline{x}_n) = \left(F_0(\underline{x}_n), \ldots, F_{J-1}(\underline{x}_n) \right)^T \in \mathbb{R}^J$$

then $Z := F(X) = \left[F(\underline{x}_0) \middle| \cdots \middle| F(\underline{x}_{N-1}) \right] \in \mathbb{R}^{J \times N}$.

We can now consider the SVD of Z,

$$Z = \tilde{U} \tilde{\Sigma} \tilde{V}^T$$

If J is not too large, specifically if $J \ll d$ then this calculation is efficient. But in many interesting cases $J \gg d$ so we prefer to skip the SVD of Z. Instead we compute the projection of Z on the k-th principal component directly

More precisely, we want to find the eigenvectors of the matrix $$\tilde{C} = \frac{1}{N} \sum_{n=0}^{N-1} \underline{z}_n \underline{z}_n^T$$

Then the principal components of $\tilde{C}$, denoted as $\tilde{\underline{u}}_j$ solve

$$\tilde{C}\, \tilde{\underline{u}}_j = \lambda_j \tilde{\underline{u}}_j, \qquad \lambda_j \in \mathbb{R}, \text{ eigenval.}$$

Now we can write

$$\underline{z}_n^T \tilde{C}\, \tilde{\underline{u}}_j = \lambda_j \underline{z}_n^T \tilde{\underline{u}}_j \quad , \text{ for } n \in \{0, \dots, N-1\},$$

we further write $\tilde{\underline{u}}_j = \sum_{n=0}^{N-1} a_n^{(j)} \underline{z}_n$, and we have

$$\frac{1}{N} \sum_{k=0}^{N-1} \underline{z}_n^T \underline{z}_k \underline{z}_k^T \left(\sum_{l=0}^{N-1} a_l^{(j)} \underline{z}_l \right) = \lambda_j \sum_{k=0}^{N-1} a_k^{(j)} \underline{z}_n^T \underline{z}_k$$

But observe $\underline{z}_n^T \underline{z}_k = \sum_{j=0}^{J-1} F_j(\underline{a}_n) F_j(\underline{a}_k) = K(\underline{a}_n, \underline{a}_k)$ so, the above equation form one row of

$$\frac{1}{N} \Theta^2 \underline{a}^{(j)} = \lambda_j \Theta\, \underline{a}^{(j)}, \text{ assuming } \Theta \text{ is invertible,}$$

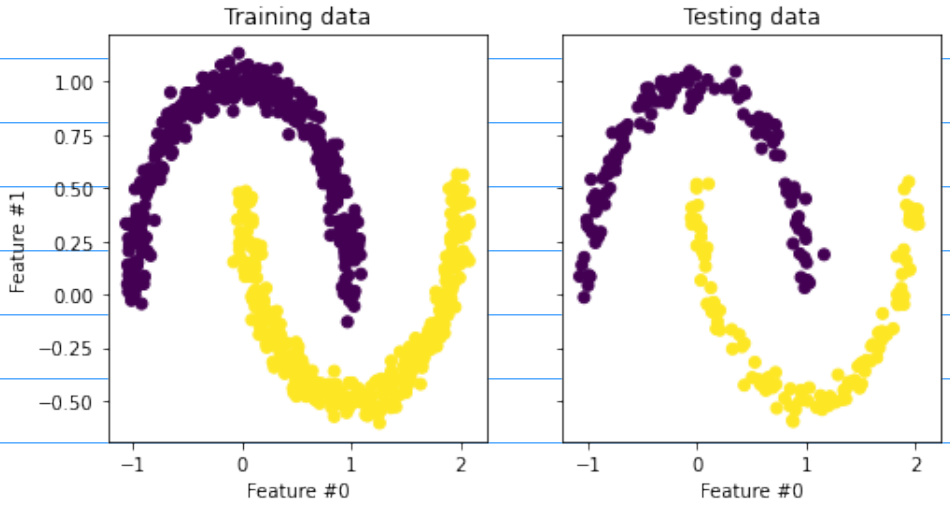
$\Rightarrow \Theta\, \underline{a}^{(j)} = N \lambda_j\, \underline{a}^{(j)}$, ie, the $a^{(j)}$ are the eigen. vectors of the kernel matrix Θ.

Then if a new point $\underline{x}_N$ comes along we can simply compute

$$\hat{\underline{u}}_j^T F(\underline{x}_N) = \sum_{n=0}^{N-1} \underline{a}_n^{(j)} F(\underline{x}_n)^T F(\underline{x})$$

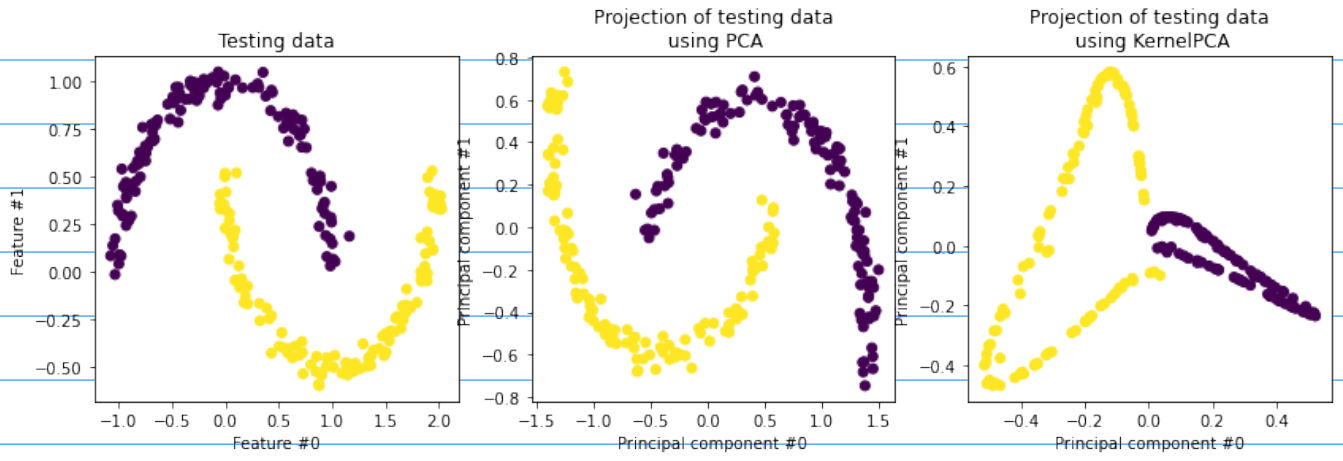
$$= \sum_{n=0}^{N-1} \underline{a}_n^{(j)} K(\underline{x}_n, \underline{x}).$$

Quick demo: the two moons data set



demo projection on PCA & kPCA modes

linearly separable!



↑ Not linearly separable!