

Machine Learning

Kernelization & Structured Input

Marc Toussaint
University of Stuttgart
Summer 2016

Kernel Ridge Regression—the “Kernel Trick”

- Reconsider solution of Ridge regression (using the *Woodbury* identity):

$$\hat{\beta}^{\text{ridge}} = (X^T X + \lambda \mathbf{I}_k)^{-1} X^T y = X^T (X X^T + \lambda \mathbf{I}_n)^{-1} y$$

Kernel Ridge Regression—the “Kernel Trick”

- Reconsider solution of Ridge regression (using the *Woodbury* identity):

$$\hat{\beta}^{\text{ridge}} = (X^T X + \lambda \mathbf{I}_k)^{-1} X^T y = X^T (X X^T + \lambda \mathbf{I}_n)^{-1} y$$

- Recall $X^T = (\phi(x_1), \dots, \phi(x_n)) \in \mathbb{R}^{k \times n}$, then:

$$f^{\text{ridge}}(x) = \phi(x)^T \beta^{\text{ridge}} = \underbrace{\phi(x)^T X^T}_{\kappa(x)^T} \underbrace{(X X^T + \lambda I)}_K^{-1} y$$

K is called *kernel matrix* and has elements

$$K_{ij} = k(x_i, x_j) := \phi(x_i)^T \phi(x_j)$$

κ is the vector: $\kappa(x)^T = \phi(x)^T X^T = k(x, x_{1:n})$

The kernel function $k(x, x')$ calculates the scalar product in feature space.

The Kernel Trick

- We can rewrite kernel ridge regression as:

$$f^{\text{ridge}}(x) = \kappa(x)^\top (K + \lambda I)^{-1} y$$

$$\text{with } K_{ij} = k(x_i, x_j)$$

$$\kappa_i(x) = k(x, x_i)$$

- at no place we actually need to compute the parameters $\hat{\beta}$
 - at no place we actually need to compute the features $\phi(x_i)$
 - we only need to be able to compute $k(x, x')$ for any x, x'
- This rewriting is called *kernel trick*.
 - It has great implications:
 - Instead of inventing funny non-linear features, we may directly invent funny kernels
 - Inventing a kernel is intuitive: $k(x, x')$ expresses how correlated y and y' should be: it is a measure of similarity, it compares x and x' . Specifying how 'comparable' x and x' are is often more intuitive than defining "features that might work".

- Every choice of features implies a kernel.
- But, does every choice of kernel correspond to a specific choice of features?

Reproducing Kernel Hilbert Space

- Let's define a vector space \mathcal{H}_k , spanned by infinitely many basis elements

$$\{\phi_x = k(\cdot, x) : x \in \mathbb{R}^d\}$$

Vectors in this space are linear combinations of such basis elements, e.g.,

$$f = \sum_i \alpha_i \phi_{x_i}, \quad f(x) = \sum_i \alpha_i k(x, x_i)$$

- Let's define a scalar product in this space, by first defining the scalar product for every basis element,

$$\langle \phi_x, \phi_y \rangle := k(x, y)$$

This is positive definite. Note, it follows

$$\langle \phi_x, f \rangle = \sum_i \alpha_i \langle \phi_x, \phi_{x_i} \rangle = \sum_i \alpha_i k(x, x_i) = f(x)$$

- The $\phi_x = k(\cdot, x)$ is the 'feature' we associate with x . Note that this is a function and infinite dimensional. Choosing $\alpha = (K + \lambda I)^{-1}y$ represents $f^{\text{ridge}}(x) = \sum_{i=1}^n \alpha_i k(x, x_i) = \kappa(x)^\top \alpha$, and shows that ridge regression has a finite-dimensional solution in the basis elements $\{\phi_{x_i}\}$. A more general version of this insight is called **representer theorem**.

Representer Theorem

- For

$$f^* = \operatorname{argmin}_{f \in \mathcal{H}_k} L(f(x_1), \dots, f(x_n)) + \Omega(\|f\|_{\mathcal{H}_k}^2)$$

where L is an arbitrary loss function, and Ω a monotone regularization, it holds

$$f^* = \sum_{i=1}^n \alpha_i k(\cdot, x_i)$$

- Proof:

decompose $f = f_s + f_\perp$, $f_s \in \operatorname{span}\{\phi_{x_i} : x_i \in D\}$

$$f(x_i) = \langle f, \phi_{x_i} \rangle = \langle f_s + f_\perp, \phi_{x_i} \rangle = \langle f_s, \phi_{x_i} \rangle = f_s(x_i)$$

$$L(f(x_1), \dots, f(x_n)) = L(f_s(x_1), \dots, f_s(x_n))$$

$$\Omega(\|f\|_{\mathcal{H}_k}^2) \geq \Omega(\|f_s\|_{\mathcal{H}_k}^2)$$

Example Kernels

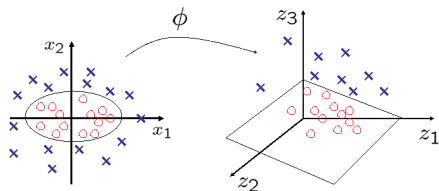
- Kernel functions need to be positive definite: $\forall_{z:|z|>0} : k(z, z') > 0$
 $\rightarrow K$ is a positive definite matrix

- Examples:

- Polynomial: $k(x, x') = (x^\top x')^d$

Let's verify for $d = 2$, $\phi(x) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)^\top$:

$$\begin{aligned}k(x, x') &= ((x_1, x_2) \begin{pmatrix} x_1' \\ x_2' \end{pmatrix})^2 \\&= (x_1x_1' + x_2x_2')^2 \\&= x_1^2x_1'^2 + 2x_1x_2x_1'x_2' + x_2^2x_2'^2 \\&= (x_1^2, \sqrt{2}x_1x_2, x_2^2)(x_1'^2, \sqrt{2}x_1'x_2', x_2'^2)^\top \\&= \phi(x)^\top \phi(x')\end{aligned}$$



- Squared exponential (radial basis function): $k(x, x') = \exp(-\gamma |x - x'|^2)$

Example Kernels

- Bag-of-words kernels: let $\phi_w(x)$ be the count of word w in document x ; define $k(x, y) = \langle \phi(x), \phi(y) \rangle$
- Graph kernels (Vishwanathan et al: Graph kernels, JMLR 2010)
 - Random walk graph kernels

Kernel Logistic Regression*

For logistic regression we compute β using the Newton iterates

$$\beta \leftarrow \beta - (X^T W X + 2\lambda I)^{-1} [X^T(p - y) + 2\lambda\beta] \quad (1)$$

$$= -(X^T W X + 2\lambda I)^{-1} X^T [(p - y) - W X \beta] \quad (2)$$

Using the Woodbury identity we can rewrite this as

$$(X^T W X + A)^{-1} X^T W = A^{-1} X^T (X A^{-1} X^T + W^{-1})^{-1} \quad (3)$$

$$\beta \leftarrow -\frac{1}{2\lambda} X^T (X \frac{1}{2\lambda} X^T + W^{-1})^{-1} W^{-1} [(p - y) - W X \beta] \quad (4)$$

$$= X^T (X X^T + 2\lambda W^{-1})^{-1} [X \beta - W^{-1}(p - y)]. \quad (5)$$

We can now compute the discriminative function values $f_X = X\beta \in \mathbb{R}^n$ at the training points by iterating over those instead of β :

$$f_X \leftarrow X X^T (X X^T + 2\lambda W^{-1})^{-1} [X \beta - W^{-1}(p - y)] \quad (6)$$

$$= K(K + 2\lambda W^{-1})^{-1} [f_X - W^{-1}(p_X - y)] \quad (7)$$

Note, that p_X on the RHS also depends on f_X . Given f_X we can compute the discriminative function values $f_Z = Z\beta \in \mathbb{R}^m$ for a set of m query points Z using

$$f_Z \leftarrow \kappa^T (K + 2\lambda W^{-1})^{-1} [f_X - W^{-1}(p_X - y)], \quad \kappa^T = Z X^T \quad (8)$$