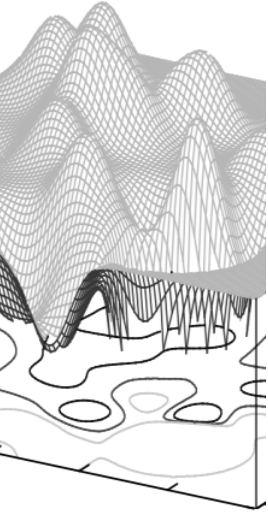


Machine Learning

Kernelization

Marc Toussaint
University of Stuttgart
Summer 2019

Kernel Ridge Regression—the "Kernel Trick"

- Reconsider solution of Ridge regression (using the *Woodbury* identity):
$$\hat\beta^{\text{ridge}} = (X^\top X + \lambda \mathbf{I}_k)^{\text{-1}} X^\top y = X^\top (X X^\top + \lambda \mathbf{I}_n)^{\text{-1}} y$$

Kernel Ridge Regression—the "Kernel Trick"

- Reconsider solution of Ridge regression (using the *Woodbury* identity):
$$\hat\beta^{\text{ridge}} = (X^\top X + \lambda \mathbf{I}_k)^{\text{-}1} X^\top y = X^\top (X X^\top + \lambda \mathbf{I}_n)^{\text{-}1} y$$

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

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

K is called *kernel matrix* and has elements

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

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

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

The Kernel Trick

- We can rewrite kernel ridge regression as:

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

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

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

$\rightarrow$ at no place we actually need to compute the parameters $\hat{\beta}$
$\rightarrow$ at no place we actually need to compute the features $\phi(x_i)$
$\rightarrow$ we only need to be able to compute $k(x, x')$ for any x, x'

The Kernel Trick

- We can rewrite kernel ridge regression as:

$$f^{\text{rigde}}(x) = \kappa(x)^\top (K + \lambda I)^{\text{-}1} y$$
$$\text{with } K_{ij} = k(x_i, x_j)$$
$$\kappa_i(x) = k(x, x_i)$$

$\to$ at no place we actually need to compute the parameters $\hat\beta$
$\to$ at no place we actually need to compute the features $\phi(x_i)$
$\to$ 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 meassure 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. Assuming $k(\cdot, \cdot)$ is positive definite, we first define the scalar product for every basis element,

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

  Then 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

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

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

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

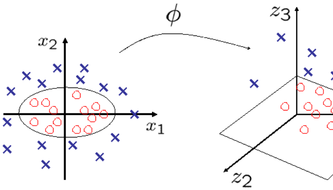
  $L(f(x_1), .., f(x_n)) = L(f_s(x_1), .., f_s(x_n))$

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

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: $k(x, x') = (x^\top x' + c)^d$
    Let's verify for $d = 2$, $\phi(x) = \left(1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, \sqrt{2}x_1x_2, x_2^2\right)^\top$:

$$
\begin{aligned}
k(x, x') &= \left((x_1, x_2)\begin{pmatrix} x_1' \\ x_2' \end{pmatrix} + 1\right)^2 \\
&= (x_1x_1' + x_2x_2' + 1)^2 \\
&= x_1^2 {x_1'}^2 + 2x_1x_2x_1'x_2' + x_2^2 {x_2'}^2 + 2x_1x_1' + 2x_2x_2' + 1 \\
&= \phi(x)^\top \phi(x')
\end{aligned}
$$



  - 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

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

- Gaussian Process regression will explain that $k(x, x')$ has the semantics of an (apriori) *correlatedness* of the yet unknown underlying function values $f(x)$ and $f(x')$
  - $k(x, x')$ should be high if you believe that $f(x)$ and $f(x')$ might be similar
  - $k(x, x')$ should be zero if $f(x)$ and $f(x')$ might be fully unrelated

Kernel Logistic Regression*

For logistic regression we compute β using the Newton iterates

$$\beta \leftarrow \beta - (X^\top W X + 2\lambda I)^{-1}\ [X^\top (p-y) + 2\lambda\beta] \tag{1}$$
$$= -(X^\top W X + 2\lambda I)^{-1}\ X^\top [(p-y) - WX\beta] \tag{2}$$

Using the Woodbury identity we can rewrite this as

$$(X^\top W X + A)^{-1} X^\top W = A^{-1} X^\top (X A^{-1} X^\top + W^{-1})^{-1} \tag{3}$$
$$\beta \leftarrow -\frac{1}{2\lambda} X^\top (X \frac{1}{2\lambda} X^\top + W^{-1})^{-1}\ W^{-1} [(p-y) - WX\beta] \tag{4}$$
$$= X^\top (X X^\top + 2\lambda W^{-1})^{-1}\ \Big[X\beta - W^{-1}(p-y)\Big]. \tag{5}$$

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:

$$f_X \leftarrow X X^\top (X X^\top + 2\lambda W^{-1})^{-1}\ \Big[X\beta - W^{-1}(p-y)\Big] \tag{6}$$
$$= K(K + 2\lambda W^{-1})^{-1}\ \Big[f_X - W^{-1}(p_X - y)\Big] \tag{7}$$

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

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