Kernels II

Cornell CS 4/5780 — Spring 2022

Well-defined kernels

Kernels can be built by recursively combining one or more of the following rules. For any well-defined kernels k_1 and k_2 the following are also well-defined kernels.

1. $\mathsf{k}(\mathbf{x}, \mathbf{z}) = \mathbf{x}^\top \mathbf{z}$ or more generally $\mathsf{k}(\mathbf{x}, \mathbf{z}) = \mathbf{x}^\top \mathbf{A} \mathbf{z}$ for A a positive semi-definite matrix
2. $\mathsf{k}(\mathbf{x}, \mathbf{z}) = c\mathsf{k}_1(\mathbf{x}, \mathbf{z})$ for $c \geq 0$
3. $\mathsf{k}(\mathbf{x}, \mathbf{z}) = \mathsf{k}_1(\mathbf{x}, \mathbf{z}) + \mathsf{k}_2(\mathbf{x}, \mathbf{z})$
4. $\mathsf{k}(\mathbf{x}, \mathbf{z}) = \mathsf{k}_1(\mathbf{x}, \mathbf{z})\mathsf{k}_2(\mathbf{x}, \mathbf{z})$
5. $\mathsf{k}(\mathbf{x}, \mathbf{z}) = f(\mathbf{x})\mathsf{k}_1(\mathbf{x}, \mathbf{z})f(\mathbf{z})$ for any function $f : \mathcal{X} \to \mathbb{R}$
6. $\mathsf{k}(\mathbf{x}, \mathbf{z}) = e^{\mathsf{k}_1(\mathbf{x}, \mathbf{z})}$ or more generally $\mathsf{k}(\mathbf{x}, \mathbf{z}) = g(\mathsf{k}(\mathbf{x}, \mathbf{z}))$ for g a polynomial or polynomial series with positive coefficients

These rules let us build up and reason about kernels without referring to their underlying feature transformations! This is a lot more intuitive, so we usually do this instead of constructing feature maps directly.

A kernel being well-defined is equivalent to the corresponding kernel matrix, $\mathbf{K} \in \mathbb{R}^{n \times n}$ given by $\mathbf{K}_{i,j} = k(x_i, x_j)$, being *positive semidefinite*, which is equivalent to any of the following statements:

1. All eigenvalues of the kernel matrix $\mathbf{K}$ are non-negative.
2. There exists a real matrix P such that $\mathbf{K} = P^\top P$.
3. For any vector $u \in \mathbb{R}^n$, $0 \leq u^T \mathbb{K} u = \sum_{i=1}^n \sum_{j=1}^n u_i u_j k(x_i, x_j)$.

It is also true that for any well-defined kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, there exists a real *Hilbert space* V and feature transformation $\phi : \mathcal{X} \to V$ such that $k(x, z) = \langle \phi(x), \phi(z) \rangle$. A Hilbert space is a possibly-infinite dimensional vector space on which we can take inner products. The fact that kernels can support these infinite-dimensional feature transformations is what makes them more expressive/powerful than finite-dimensional feature maps.

It is trivial to prove using these rules that linear kernel and polynomial kernel with integer d are both well-defined kernels.

Quiz: Prove that the RBF kernel $k(x, z) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{z}\|^2}{\sigma^2}\right)$ is a well-defined kernel.

You can even define kernels of sets, or strings or molecules. The domain of the kernel need not be a vector space!

The following kernel is defined for S_1, S_2 being any two finite sets: $\mathsf{k}(S_1, S_2) = \exp\left(|S_1 \cap S_2|\right)$. Prove that this is a well-defined kernel.

Kernel Machines

(In practice) an algorithm can be kernelized in three steps:

1. Prove that the solution lies in the span of the training points (i.e. $\mathbf{w} = \sum_{i=1}^n \alpha_i \mathbf{x}_i$ for some α_i)
2. Rewrite the algorithm and the classifier so that all training or testing inputs $\mathbf{x}_i$ are only accessed in inner-products with other inputs, e.g. $\mathbf{x}_i^\top \mathbf{x}_j$.
3. Define a kernel function and substitute $\mathsf{k}(\mathbf{x}_i, \mathbf{x}_j)$ for $\mathbf{x}_i^\top \mathbf{x}_j$.

Kernelized Linear Regression

We begin by expressing the solution $\mathbf{w}$ as a linear combination of the training inputs $\mathbf{w} = \sum_{i=1}^n \alpha_i \mathbf{x}_i = \mathbf{X}\vec{\alpha}$. We derived in the previous lecture that such a vector $\vec{\alpha}$ must always exist by observing the gradient updates that occur if (5) is minimized with gradient descent and the initial vector is set to $\mathbf{w}_0 = \vec{0}$ (because the squared loss is convex the solution is independent of its initialization.) We can now immediately kernelize the algorithm by substituting $k(\mathbf{x}, \mathbf{z})$ for any inner-product $\mathbf{x}^\top \mathbf{z}$. It remains to show that we can also solve for the values of α in closed form. As it turns out, this is straight-forward.

Kernelized ordinary least squares has the solution $\vec{\alpha} = \mathbf{K}^{-1}\mathbf{y}$.

$$\begin{aligned}
\mathbf{X}\vec{\alpha} &= \mathbf{w} = (\mathbf{X}\mathbf{X}^\top)^{-1}\mathbf{X}\mathbf{y} \quad \text{| multiply from left by } \mathbf{X}^\top\mathbf{X}\mathbf{X}^\top \\
(\mathbf{X}^\top\mathbf{X})(\mathbf{X}^\top\mathbf{X})\vec{\alpha} &= \mathbf{X}^\top(\mathbf{X}\mathbf{X}^\top(\mathbf{X}\mathbf{X}^\top)^{-1})\mathbf{X}\mathbf{y} \quad \text{|substitute } \mathbf{K} = \mathbf{X}^\top\mathbf{X} \\
\mathbf{K}^2\vec{\alpha} &= \mathbf{K}\mathbf{y} \quad \text{|multiply from left by } (\mathbf{K}^{-1})^2 \\
\vec{\alpha} &= \mathbf{K}^{-1}\mathbf{y}
\end{aligned}$$

Kernel regression can be extended to the kernelized version of *ridge regression*. The solution then becomes $\vec{\alpha} = (\mathbf{K} + \tau^2\mathbf{I})^{-1}\mathbf{y}$. In practice a small value of $\tau^2 > 0$ increases stability, especially if $\mathbf{K}$ is not invertible. If $\tau = 0$ kernel ridge regression, becomes kernelized ordinary least squares. Typically *kernel ridge regression* is also referred to as *kernel regression*.

Testing. Remember that we defined $\mathbf{w} = \mathbf{X}\vec{\alpha}$. The prediction of a test point $\mathbf{z}$ then becomes

$$h(\mathbf{z}) = \mathbf{z}^\top\mathbf{w} = \mathbf{z}^\top\underbrace{\mathbf{X}\vec{\alpha}}_{\mathbf{w}} = \underbrace{\mathbf{k}_*}_{\mathbf{z}^\top\mathbf{X}}\underbrace{(\mathbf{K} + \tau^2\mathbf{I})^{-1}\mathbf{y}}_{\vec{\alpha}} = \mathbf{k}_*\vec{\alpha},$$

or, if everything is in closed form: $h(\mathbf{z}) = \mathbf{k}_*(\mathbf{K} + \tau^2\mathbf{I})^{-1}\mathbf{y}$, where $\mathbf{k}_*$ is the kernel (vector) of the test point with the training points, i.e. the i^{th} dimension corresponds to $[\mathbf{k}_*]_i = \phi(\mathbf{z})^\top\phi(\mathbf{x}_i)$, the inner-product between the test point $\mathbf{z}$ with the training point $\mathbf{x}_i$ after the mapping into feature space through ϕ.

Kernel SVM

When we kernelize the SVM, we usually kernelize its dual formulation, rather than going directly for the original optimization problem we looked at when we first derived the SVM. The original, **primal** soft-margin SVM is a quadratic programming problem:

$$\begin{aligned}
&\min_{\mathbf{w},b} \mathbf{w}^\top\mathbf{w} + C\sum_{i=1}^n \xi_i \\
\text{s.t. } \forall i, \quad &y_i(\mathbf{w}^\top\mathbf{x}_i + b) \geq 1 - \xi_i \\
&\xi_i \geq 0
\end{aligned}$$

has the **dual** form

$$\begin{aligned}
&\min_{\alpha_1,\cdots,\alpha_n} \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j K_{ij} - \sum_{i=1}^n \alpha_i \\
\text{s.t.} \quad &0 \leq \alpha_i \leq C \\
&\sum_{i=1}^n \alpha_i y_i = 0
\end{aligned}$$

where $\mathbf{w} = \sum_{i=1}^n \alpha_i y_i \phi(\mathbf{x}_i)$ (although this is never computed) and

$$h(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^n \alpha_i y_i k(\mathbf{x}_i, \mathbf{x}) + b\right) \quad \text{where} \quad b = y_i - \sum_j y_j\alpha_j k(\mathbf{x}_j, \mathbf{x}_i)$$

Support Vectors

There is a very nice interpretation of the dual problem in terms of support vectors. For the primal formulation we know (from a previous lecture) that only support vectors satisfy the constraint with equality:

$$y_i(\mathbf{w}^\top\phi(x_i) + b) = 1.$$

In the dual, these same training inputs can be identified as their corresponding dual values satisfy $\alpha_i > 0$ (all other training inputs have $\alpha_i = 0$). For test-time you only need to compute the sum in $h(\mathbf{x})$ over the support vectors and all inputs $\mathbf{x}_i$ with $\alpha_i = 0$ can be discarded after training.