Machine Learning Course - CS-433

Kernel Ridge Regression and the Kernel Trick

Nov 2, 2022

changes by Nicolas Flammarion 2021,2020, changes by Martin Jaggi 2019, changes by Rüdiger Urbanke 2018, changes by Martin Jaggi 2016, 2017 ©Mohammad Emtiyaz Khan 2015

Last updated on: October 30, 2022

EPFL

Motivation

In our last lecture we formulated the optimization problem corresponding to SVMs. We then derived an alternative formulation using duality. We saw that in this alternative formulation the data only enters in the form of a "kernel" $\mathbf{K} = \mathbf{X}\mathbf{X}^\top$.

The aim of today is the following. First, we will discuss a second problem, namely ridge regression, that admits an alternative formulation that is "kernelized," i.e., the alternative formulation depends on the data only via the kernel $\mathbf{K} = \mathbf{X}\mathbf{X}^\top$. Second, we will see that for any kernelized problem we can apply the *kernel trick*. This trick will allow us to use a significantly augmented feature vector without incurring extra costs. We will discuss what kernel functions are *admissible* for this trick and how to construct new kernel functions from old ones.

Even though we present the kernel trick in the context of ridge regression it will hopefully be clear by the end that the same trick can be applied to any problem that can be brought into kernelized form.

Alternative formulation of ridge regression

Recall that ridge regression corresponds to the following optimization problem:

$$\min_{\mathbf{w}} \quad \tfrac{1}{2N}\|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2 + \tfrac{\lambda}{2}\|\mathbf{w}\|^2.$$

It has the solution

$$\mathbf{w}^\star = \tfrac{1}{N}(\tfrac{1}{N}\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I}_D)^{-1}\mathbf{X}^\top\mathbf{y}.$$

We claim that this solution can be written alternatively as

$$\mathbf{w}^\star = \tfrac{1}{N}\mathbf{X}^\top(\tfrac{1}{N}\mathbf{X}\mathbf{X}^\top + \lambda \mathbf{I}_N)^{-1}\mathbf{y}. \tag{1}$$

This second formulation can be proved using the following identity: let $\mathbf{P}$ be an $N \times M$ matrix and $\mathbf{Q}$ be an $M \times N$ matrix. Then, trivially,

$$\mathbf{P}(\mathbf{Q}\mathbf{P} + \mathbf{I}_M) = \mathbf{P}\mathbf{Q}\mathbf{P} + \mathbf{P} = (\mathbf{P}\mathbf{Q} + \mathbf{I}_N)\mathbf{P}.$$

If we now assume that $(\mathbf{Q}\mathbf{P} + \mathbf{I}_M)$ and $(\mathbf{P}\mathbf{Q} + \mathbf{I}_N)$ are invertible we have the identity

$$(\mathbf{P}\mathbf{Q} + \mathbf{I}_N)^{-1}\mathbf{P} = \mathbf{P}(\mathbf{Q}\mathbf{P} + \mathbf{I}_M)^{-1}.$$

To derive from this general statement our alternative representation, let $\mathbf{P} = \mathbf{X}^\top$ and $\mathbf{Q} = \frac{1}{\lambda N}\mathbf{X}$.
Why is this alternative representation useful?

1. Define $\boldsymbol{\alpha}^\star = \frac{1}{N}(\frac{1}{N}\mathbf{X}\mathbf{X}^\top + \lambda \mathbf{I}_N)^{-1}\mathbf{y}$. Then we can write

$$\mathbf{w}^\star = \mathbf{X}^\top \boldsymbol{\alpha}^\star. \tag{2}$$

   From this representation we see that $\mathbf{w}^\star$ lies in the column space of $\mathbf{X}^\top$, i.e., the space spanned by the feature vectors.

2. The original formulation involves computation of order $O(D^3 + ND^2)$, while the second can be computed in time $O(N^3 + DN^2)$. Hence it depends on the size of D and N which of the two is more efficient.

3. We will see that (1) and (2) are the crucial ingredients for the kernel trick to work.

The representer theorem

The representer theorem generalizes this result: for a $\mathbf{w}^\star$ minimizing the following function for any $\mathcal{L}_n$,

$$\min_{\mathbf{w}} \frac{1}{N}\sum_{n=1}^{N} \mathcal{L}_n(\mathbf{x}_n^\top \mathbf{w}, y_n) + \tfrac{\lambda}{2}\|\mathbf{w}\|^2$$

there exists $\boldsymbol{\alpha}^\star$ such that $\mathbf{w}^\star = \mathbf{X}^\top \boldsymbol{\alpha}^\star$.

Such a general statement was originally proved by *Schölkopf, Herbrich and Smola (2001)*.

Kernelized ridge regression

Let us go back to ridge regression. The representer theorem allows us to write an equivalent optimization problem in terms of $\boldsymbol{\alpha}$:

$$\mathbf{w}^\star = \arg\min_{\mathbf{w}} \quad \tfrac{1}{2N}\|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2 + \tfrac{\lambda}{2}\|\mathbf{w}\|^2$$

$$\boldsymbol{\alpha}^\star = \arg\min_{\boldsymbol{\alpha}} \frac{1}{2}\boldsymbol{\alpha}^\top(\tfrac{1}{N}\mathbf{X}\mathbf{X}^\top + \lambda \mathbf{I}_N)\boldsymbol{\alpha} - \tfrac{1}{N}\boldsymbol{\alpha}^\top \mathbf{y}.$$

To see that these two problems have equivalent solutions, note that if we take the gradient of the second expression we get $(\mathbf{X}\mathbf{X}^\top + \lambda \mathbf{I}_N)\boldsymbol{\alpha} - \mathbf{y}$. Setting this to 0 and solving for $\boldsymbol{\alpha}$ results in

$$\boldsymbol{\alpha}^\star = \tfrac{1}{N}(\tfrac{1}{N}\mathbf{X}\mathbf{X}^\top + \lambda \mathbf{I}_N)^{-1}\mathbf{y}.$$

If we combine this with the representer theorem $\mathbf{w}^\star = \mathbf{X}^\top \boldsymbol{\alpha}^\star$ we find back the solution (1).

As we discussed previously, depending on the D, the dimension of the feature space, and N, the number of samples, one or the other of the two formulations might be more efficient. But there is an arguably even more important reason why the second expression is of interest. In this second expression the data only enters in terms of the kernel matrix $\mathbf{K} = \mathbf{X}\mathbf{X}^\top$.

Kernel functions

Recall that the kernel is defined as

$$\mathbf{K} = \mathbf{X}\mathbf{X}^\top = \begin{bmatrix} \mathbf{x}_1^\top\mathbf{x}_1 & \mathbf{x}_1^\top\mathbf{x}_2 & \dots & \mathbf{x}_1^\top\mathbf{x}_N \\ \mathbf{x}_2^\top\mathbf{x}_1 & \mathbf{x}_2^\top\mathbf{x}_2 & \dots & \mathbf{x}_2^\top\mathbf{x}_N \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{x}_N^\top\mathbf{x}_1 & \mathbf{x}_N^\top\mathbf{x}_2 & \dots & \mathbf{x}_N^\top\mathbf{x}_N \end{bmatrix}.$$

For reasons that will become clear shortly, we call this the *linear* kernel.

Assume that we had first augmented the feature space to $\boldsymbol{\phi}(\mathbf{x})$. The associated kernel with basis functions $\boldsymbol{\phi}(\mathbf{x})$ would then be $\mathbf{K} = \boldsymbol{\Phi}\boldsymbol{\Phi}^\top$. Explicitly,

$$\begin{bmatrix} \boldsymbol{\phi}(\mathbf{x}_1)^\top\boldsymbol{\phi}(\mathbf{x}_1) & \boldsymbol{\phi}(\mathbf{x}_1)^\top\boldsymbol{\phi}(\mathbf{x}_2) & \dots & \boldsymbol{\phi}(\mathbf{x}_1)^\top\boldsymbol{\phi}(\mathbf{x}_N) \\ \boldsymbol{\phi}(\mathbf{x}_2)^\top\boldsymbol{\phi}(\mathbf{x}_1) & \boldsymbol{\phi}(\mathbf{x}_2)^\top\boldsymbol{\phi}(\mathbf{x}_2) & \dots & \boldsymbol{\phi}(\mathbf{x}_2)^\top\boldsymbol{\phi}(\mathbf{x}_N) \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{\phi}(\mathbf{x}_N)^\top\boldsymbol{\phi}(\mathbf{x}_1) & \boldsymbol{\phi}(\mathbf{x}_N)^\top\boldsymbol{\phi}(\mathbf{x}_2) & \dots & \boldsymbol{\phi}(\mathbf{x}_N)^\top\boldsymbol{\phi}(\mathbf{x}_N) \end{bmatrix}.$$

We have already discussed that sometimes it is useful to augment the feature space. This will lead to a more powerful model. Here is a link to a video explaining this point in more detail: https://www.youtube.com/watch?v=3liCbRZPrZA

The kernel trick

The big advantage of using kernels is that rather than first augmenting the feature space and then computing the kernel, we can do both steps together, and we can do it more efficiently. Let us discuss how this works.
Let us define a "kernel function" $\kappa(\mathbf{x}, \mathbf{x}')$ and let us compute the (i, j)-th entry of $\mathbf{K}$ as $K_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$. For the right choice of kernel κ it turns out to be equivalent to first augmenting the features to some suitable $\boldsymbol{\phi}(\mathbf{x})$ and then computing the inner product

$$\boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\phi}(\mathbf{x}')$$

in the augmented space. In other words, for the right choices we have

$$\kappa(\mathbf{x}, \mathbf{x}') = \boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\phi}(\mathbf{x}') \ .$$

This is probably best seen by looking at examples:

1. To start trivially, if we pick the linear kernel $\kappa(\mathbf{x}, \mathbf{x}') = \mathbf{x}^\top \mathbf{x}'$, then the corresponding feature map is of course $\boldsymbol{\phi}(\mathbf{x}') = \mathbf{x}'$.

2. Assume that $\mathbf{x} \in \mathbb{R}$, i.e., $\mathbf{x}$ is a scalar. The kernel $\kappa(x, x') = (xx')^2$ corresponds to $\phi(x) = x^2$.

3. Assume that $\mathbf{x} \in \mathbb{R}^3$, i.e., $\mathbf{x}$ is a vector of dimension 3. The kernel $\kappa(\mathbf{x}, \mathbf{x}') = (x_1 x_1' + x_2 x_2' + x_3 x_3')^2$ corresponds to

   $$\boldsymbol{\phi}(\mathbf{x})^\top = \left[\ x_1^2, \ x_2^2, \ x_3^2, \ \sqrt{2} x_1 x_2, \ \sqrt{2} x_1 x_3, \ \sqrt{2} x_2 x_3 \ \right] .$$

   This is an example of what is called a *polynomial kernel*.

4. The kernel

$$\kappa(\mathbf{x}, \mathbf{x}') = \exp\left[-(\mathbf{x} - \mathbf{x}')^\top(\mathbf{x} - \mathbf{x}')\right]$$

corresponds to an infinite feature map! It is called the the *radial basis function* (RBF) kernel. In order to look at this more in detail, consider the simple case where the $\mathbf{x}$ and $\mathbf{x}'$ are scalars. In this case we have the expansion

$$\mathbf{K}(x, x') = e^{-(x)^2} e^{-(x')^2} \sum_{k=0}^{\infty} \frac{2^k (x)^k (x')^k}{k!}.$$

We see that we can think of this as the inner product of infinite-dimensional vectors whose k-th component, $k = 0, 1, \cdots$ is equal to

$$e^{-(x)^2} \sqrt{\frac{2^k}{k!}} (x)^k \text{ and } e^{-(x')^2} \sqrt{\frac{2^k}{k!}} (x')^k,$$

respectively. And although this is not obvious, let us state that this kernel cannot be represented as an inner product in a finite-dimensional space.

5. You can find many further examples in Section 14.2 of Murphy's book.

6. Building new kernels from old kernels.

   a) $\kappa(\mathbf{x}, \mathbf{x}') = a\kappa_1(\mathbf{x}, \mathbf{x}') + b\kappa_2(\mathbf{x}, \mathbf{x}')$ for all $a, b \geq 0$.

   Proof. By assumption κ_1 and κ_2 are valid kernels. Hence there exist feature maps ϕ_1 and ϕ_2 so that

$$\kappa_1(\mathbf{x}, \mathbf{x}') = \phi_1(\mathbf{x})^\top \phi_1(\mathbf{x}'),$$
$$\kappa_2(\mathbf{x}, \mathbf{x}') = \phi_2(\mathbf{x})^\top \phi_2(\mathbf{x}').$$

Hence,

$$\kappa(\mathbf{x}, \mathbf{x}') = a\phi_1(\mathbf{x})^\top \phi_1(\mathbf{x}') + b\phi_2(\mathbf{x})^\top \phi_2(\mathbf{x}').$$

This can be represented as an inner product via the feature map

$$(\sqrt{a}\phi_1(\cdot), \sqrt{b}\phi_2(\cdot)).$$

□

b) $\kappa(\mathbf{x}, \mathbf{x}') = \kappa_1(\mathbf{x}, \mathbf{x}')\kappa_2(\mathbf{x}, \mathbf{x}')$.

Proof. Let the two feature maps be ϕ_1 and ϕ_2. Assume that they are of dimensions d_1 and d_2. Then ϕ is a feature map of dimension $d_1 d_2$ of the form

$$\begin{aligned}\phi(\mathbf{x})^\top =&((\phi_1(\mathbf{x}))_1(\phi_2(\mathbf{x}))_1, \cdots, (\phi_1(\mathbf{x}))_1(\phi_2(\mathbf{x}))_{d_2}, \cdots, \\ &(\phi_1(\mathbf{x}))_{d_1}(\phi_2(\mathbf{x}))_1, \cdots, (\phi_1(\mathbf{x}))_{d_1}(\phi_2(\mathbf{x}))_{d_2}).\end{aligned}$$

□

c) $\kappa(\mathbf{x}, \mathbf{x}') = \kappa_1(f(\mathbf{x}), f(\mathbf{x}'))$ for any f from the domain to itself.

Proof. Let $\phi_1(\cdot)$ be the feature map corresponding to $\kappa_1(\cdot, \cdot)$. Then by direct inspection we see that $\phi(\cdot) = \phi_1(f(\cdot))$ is the feature map corresponding to $\kappa(f(\cdot), f(\cdot))$. Indeed,

$$\phi_1(f(\mathbf{x}))^\top \phi_1(f(\mathbf{x}')) = \kappa_1(f(\mathbf{x}), f(\mathbf{x}') = \kappa(\mathbf{x}, \mathbf{x}').$$

□

d) $\kappa(\mathbf{x}, \mathbf{x}') = f(\mathbf{x})f(\mathbf{x}')$ for any real-valued f. Clearly, $\phi(\mathbf{x}) = f(\mathbf{x})$ will be the corresponding feature map.

Classifying with the kernel K

We have seen so far how we can compute the optimal parameter vector $\boldsymbol{\alpha}$ using only the kernel (and not having to go to the extended feature space). We have also discussed that in some cases the feature space is in fact infinite. All this would not be useful if there was not also a way to do the prediction using only the kernel. Indeed this is possible.
Recall that the classifier predicts $y = \phi(\mathbf{x})^\top \mathbf{w}^*$ which, using (2), can be expressed as

$$y = \phi(\mathbf{x})^\top \phi(\mathbf{X})^\top \boldsymbol{\alpha} = \sum_{n=1}^{N} \kappa(\mathbf{x}, \mathbf{x}_n)\boldsymbol{\alpha}_n.$$

I.e., we can express the prediction in terms of the kernel function applied to the new feature vector and the data vector in the original space and do not need to go into the augmented space.
From this expression we can also clearly see that although the classifier in the extended space $\phi(\mathbf{x})$ is linear, if we look at the decision regions in the original space $\mathbf{x}$ it is non-linear.

Properties of kernels: Mercer's Condition

A natural question is the following: how can we ensure that there exists a $\boldsymbol{\phi}$ corresponding to a given kernel function κ?

I.e., how do we ensure that the kernel function is an inner-product in some feature space?
Mercer's condition states that this is true if and only if the following two conditions are fulfilled. In the following, given the kernel function κ and some arbitrary input set $\{\mathbf{x}_n\}_{n=1}^N$, let K be the associated kernel matrix, $\mathbf{K}_{i,j} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$.

1. The kernel function κ must be symmetric, i.e. $\kappa(\mathbf{x}, \mathbf{x}') = \kappa(\mathbf{x}', \mathbf{x})$; equivalently, the kernel matrix $\mathbf{K}$ must be symmetric for all possible input sets.

2. The kernel matrix $\mathbf{K}$ must be positive semi-definite for all possible input sets.