CS 189 Introduction to Machine Learning
Spring 2018

Note 8

1 Kernel Ridge Regression

In ridge regression, we given a vector $\mathbf{y} \in \mathbb{R}^n$ and a matrix $\mathbf{X} \in \mathbb{R}^{n \times \ell}$, where n is the number of training points and ℓ is the dimension of the raw data points. In most settings we don't want to work with just the raw feature space, so we augment features to the data points and replace $\mathbf{X}$ with $\mathbf{\Phi} \in \mathbb{R}^{n \times d}$, where $\boldsymbol{\phi}_i^\top = \phi(\mathbf{x}_i) \in \mathbb{R}^d$. Then we solve a well-defined optimization problem that involves $\mathbf{\Phi}$ and $\mathbf{y}$, over the parameters $\mathbf{w} \in \mathbb{R}^d$. Note the problem that arises here. If we have polynomial features of degree at most p in the raw ℓ dimensional space, then there are $d = \binom{\ell+p}{p}$ terms that we need to optimize, which can be very, very large (much larger than the number of training points n). Wouldn't it be useful, if instead of solving an optimization problem over d variables, we could solve an equivalent problem over (potentially much smaller) n variables, and achieve a computational runtime independent of the number of augmented features? As it turns out, the concept of kernels (in addition to a technique called the kernel trick) will allow us to achieve this goal. Recall the solution to ridge regression:

$$\mathbf{w}^* = (\mathbf{\Phi}^\top \mathbf{\Phi} + \lambda \mathbf{I})^{-1} \mathbf{\Phi}^\top \mathbf{y}$$

This operation involves calculating $\mathbf{\Phi}^\top \mathbf{\Phi}$, which is a $d \times d$ matrix and takes $O(d^2 n)$ time to compute. The matrix inversion operation takes an additional $O(d^3)$ time to compute. What we would really like is to have an $n \times n$ matrix that takes $O(n^3)$ to invert. Here's a simple observation: if we flip the order of $\mathbf{\Phi}^\top$ and $\mathbf{\Phi}$, we end up with an $n \times n$ matrix $\mathbf{\Phi}\mathbf{\Phi}^\top$. In fact, the matrix $\mathbf{\Phi}\mathbf{\Phi}^\top$ has a very intuitive meaning: it is the matrix of inner products between all of the augmented datapoints, which in loose terms measures the "similarity" among of the datapoints and captures their relationship. Now let's see if we could somehow express the solution to ridge regression using the matrix $\mathbf{\Phi}\mathbf{\Phi}^\top$.

1.1 Derivation

For simplicity of notation, let's revert back to using $\mathbf{X}$ instead of $\mathbf{\Phi}$ (pretend that we are only working with raw features, our analysis of kernel ridge regression still holds if we use just the raw features). Rearranging the terms of the original ridge regression solution, we have

$$\begin{aligned}
\mathbf{w} &= (\mathbf{X}^\top \mathbf{X} + \lambda I)^{-1} \mathbf{X}^\top \mathbf{y} \\
(\mathbf{X}^\top \mathbf{X} + \lambda I)\mathbf{w} &= \mathbf{X}^\top \mathbf{y} \\
\mathbf{X}^\top \mathbf{X}\mathbf{w} + \lambda \mathbf{w} &= \mathbf{X}^\top \mathbf{y} \\
\lambda \mathbf{w} &= \mathbf{X}^\top \mathbf{y} - \mathbf{X}^\top \mathbf{X}\mathbf{w} \\
\mathbf{w} &= \frac{1}{\lambda}(\mathbf{X}^\top \mathbf{y} - \mathbf{X}^\top \mathbf{X}\mathbf{w})
\end{aligned}$$

$$\mathbf{w} = \frac{\mathbf{X}^\top \mathbf{y} - \mathbf{X}^\top \mathbf{X}\mathbf{w}}{\lambda}$$
$$\mathbf{w} = \mathbf{X}^\top \frac{\mathbf{y} - \mathbf{X}\mathbf{w}}{\lambda}$$

which says that *whatever* $\mathbf{w}$ *is*, it is some linear combination of the training points $\mathbf{x}_i$ (because anything of the form $\mathbf{X}^\top \mathbf{v}$ is a linear combination of the columns of $\mathbf{X}^\top$, which are the training points). To find $\mathbf{w}$ it suffices to find $\mathbf{v}$, where $\mathbf{w} = \mathbf{X}^\top \mathbf{v}$.

Recall that the relationship we have to satisfy is $\mathbf{X}^\top \mathbf{X}\mathbf{w} - \lambda \mathbf{w} = \mathbf{X}^\top \mathbf{y}$. Let's assume that we had $\mathbf{v}$, and just substitute $\mathbf{X}^\top \mathbf{v}$ in for all the $\mathbf{w}$'s.

$$\begin{aligned}
\mathbf{X}^\top \mathbf{X}(\mathbf{X}^\top \mathbf{v}) + \lambda(\mathbf{X}^\top \mathbf{v}) &= \mathbf{X}^\top \mathbf{y} \\
\mathbf{X}^\top \mathbf{X}\mathbf{X}^\top \mathbf{v} + \mathbf{X}^\top(\lambda \mathbf{v}) &= \mathbf{X}^\top \mathbf{y} \\
\mathbf{X}^\top(\mathbf{X}\mathbf{X}^\top \mathbf{v} + \lambda \mathbf{v}) &= \mathbf{X}^\top(\mathbf{y})
\end{aligned}$$

We can't yet isolate $\mathbf{v}$ and have a closed-form solution for it, but we *can* make the observation that if we found an $\mathbf{v}$ such that we had

$$\mathbf{X}\mathbf{X}^\top \mathbf{v} + \lambda \mathbf{v} = \mathbf{y}$$

that would *imply* that this $\mathbf{v}$ also satisfies the above equation. Note that we did not "cancel the $\mathbf{X}^\top$'s on both sides of the equation." We saw that having $\mathbf{v}$ satisfy one equation implied that it satisfied the other as well. So, indeed, we can isolate $\mathbf{v}$ in this new equation:

$$(\mathbf{X}\mathbf{X}^\top + \lambda \mathbf{I})\mathbf{v} = \mathbf{y} \implies \mathbf{v}^* = (\mathbf{X}\mathbf{X}^\top + \lambda \mathbf{I})^{-1}\mathbf{y}$$

and have that the $\mathbf{v}$ which satisfies this equation will be such that $\mathbf{X}^\top \mathbf{v}$ equals $\mathbf{w}$. We conclude that the optimal $\mathbf{w}$ is

$$\mathbf{w}^* = \mathbf{X}^\top \mathbf{v}^* = \mathbf{X}^\top (\mathbf{X}\mathbf{X}^\top + \lambda \mathbf{I})^{-1}\mathbf{y}$$

Recall that previously, we derived ridge regression and ended up with

$$\mathbf{w}^* = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1}\mathbf{X}^\top \mathbf{y}$$

In fact, these two are equivalent expressions! The question that now arises is which expression should you pick? Which is *more efficient* to calculate? We will answer this question after we introduce kernels.

1.2 Linear Algebra Derivation

The previous derivation involved using some intuitive manipulations to achieve the desired answer. Let's formalize our derivation using more principled arguments from linear algebra and optimization Before we do so, we must first introduce the **Fundamental Theorem of Linear Algebra (FTLA)**: Suppose that there is a matrix (linear map) $\mathbf{X}$ that maps $\mathbb{R}^\ell$ to $\mathbb{R}^n$. Denote $\mathcal{N}(\mathbf{X})$ as the nullspace of $\mathbf{X}$, and $\mathcal{R}(\mathbf{X})$ as the range of $\mathbf{X}$. Then the following properties hold:

1. $\mathcal{N}(\mathbf{X}) \overset{\perp}{\oplus} \mathcal{R}(\mathbf{X}^\top) = \mathbb{R}^\ell$ and $\mathcal{N}(\mathbf{X}^\top) \overset{\perp}{\oplus} \mathcal{R}(\mathbf{X}) = \mathbb{R}^n$ by symmetry

   The symbol $\oplus$ indicates that we taking a **direct sum** of $\mathcal{N}(\mathbf{X})$ and $\mathcal{R}(\mathbf{X}^\top)$, which means that $\forall u \in \mathbb{R}^\ell$ there exist unique elements $u_1 \in \mathcal{N}(\mathbf{X})$ and $u_2 \in \mathcal{R}(\mathbf{X}^\top)$ such that $u = u_1 + u_2$. Furthermore, the symbol $\perp$ indicates that $\mathcal{N}(\mathbf{X})$ and $\mathcal{R}(\mathbf{X}^\top)$ are orthogonal subspaces.

2. $\mathcal{N}(\mathbf{X}^\top\mathbf{X}) = \mathcal{N}(\mathbf{X})$ and $\mathcal{N}(\mathbf{X}\mathbf{X}^\top) = \mathcal{N}(\mathbf{X}^\top)$ by symmetry

3. $\mathcal{R}(\mathbf{X}^\top\mathbf{X}) = \mathcal{R}(\mathbf{X}^\top)$ and $\mathcal{R}(\mathbf{X}\mathbf{X}^\top) = \mathcal{R}(\mathbf{X})$ by symmetry.

Here's where FTLA comes, in the context of kernel ridge regression. We know that we can express any $\mathbf{w} \in \mathbb{R}^\ell$ as a unique combination $\mathbf{w} = \mathbf{w}_1 + \mathbf{w}_2$, where $\mathbf{w}_1 \in \mathcal{R}(\mathbf{X}^\top)$ and $\mathbf{w}_2 \in \mathcal{N}(\mathbf{X})$. Equivalently we can express this as $\mathbf{w} = \mathbf{X}^\top\mathbf{v} + \mathbf{r}$, where $\mathbf{v} \in \mathbb{R}^n$ and $\mathbf{r} \in \mathcal{N}(\mathbf{X})$. Now, instead of optimizing over $\mathbf{w} \in \mathbb{R}^\ell$, we can optimize over $\mathbf{v} \in \mathbb{R}^n$ and $\mathbf{r} \in \mathbb{R}^\ell$, which equates to optimizing over $n + \ell$ variables. However, as we shall see, the optimization over $\mathbf{r}$ will be trivial so we just have to optimize an n dimensional problem.

We know that $\mathbf{w} = \mathbf{X}^\top\mathbf{v} + \mathbf{r}$, where $\mathbf{v} \in \mathbb{R}^n$ and $\mathbf{r} \in \mathcal{N}(\mathbf{X})$. Let's now solve ridge regression by optimizing over the variables $\mathbf{v}$ and $\mathbf{r}$ instead of $\mathbf{w}$:

$$\begin{aligned}
\mathbf{v}^*, \mathbf{r}^* &= \underset{\mathbf{v}\in\mathbb{R}^n, \mathbf{r}\in\mathcal{N}(\mathbf{X})}{\arg\min} \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2 + \lambda\|\mathbf{w}\|_2^2 \\
&= \underset{\mathbf{v}\in\mathbb{R}^n, \mathbf{r}\in\mathcal{N}(\mathbf{X})}{\arg\min} \|\mathbf{X}(\mathbf{X}^\top\mathbf{v} + \mathbf{r}) - \mathbf{y}\|_2^2 + \lambda\|\mathbf{X}^\top\mathbf{v} + \mathbf{r}\|_2^2 \\
&= \underset{\mathbf{v}\in\mathbb{R}^n, \mathbf{r}\in\mathcal{N}(\mathbf{X})}{\arg\min} \|\mathbf{X}\mathbf{X}^\top\mathbf{v} + \cancel{\mathbf{X}\mathbf{r}} - \mathbf{y}\|_2^2 + \lambda\|\mathbf{X}^\top\mathbf{v} + \mathbf{r}\|_2^2 \\
&= \underset{\mathbf{v}\in\mathbb{R}^n, \mathbf{r}\in\mathcal{N}(\mathbf{X})}{\arg\min} \left(\mathbf{v}^\top\mathbf{X}\mathbf{X}^\top\mathbf{X}\mathbf{X}^\top\mathbf{v} - 2\mathbf{v}^\top\mathbf{X}\mathbf{X}^\top\mathbf{y} + \mathbf{y}^\top\mathbf{y}\right) + \lambda\left(\mathbf{v}^\top\mathbf{X}\mathbf{X}^\top\mathbf{v} + \cancel{2\mathbf{v}^\top\mathbf{X}\mathbf{r}} + \mathbf{r}^\top\mathbf{r}\right) \\
&= \underset{\mathbf{v}\in\mathbb{R}^n, \mathbf{r}\in\mathcal{N}(\mathbf{X})}{\arg\min} \left(\mathbf{v}^\top\mathbf{X}\mathbf{X}^\top\mathbf{X}\mathbf{X}^\top\mathbf{v} - 2\mathbf{v}^\top\mathbf{X}\mathbf{X}^\top\mathbf{y}\right) + \lambda\left(\mathbf{v}^\top\mathbf{X}\mathbf{X}^\top\mathbf{v} + \mathbf{r}^\top\mathbf{r}\right)
\end{aligned}$$

We crossed out $\mathbf{X}\mathbf{r}$ and $2\mathbf{v}^\top\mathbf{X}\mathbf{r}$ because $\mathbf{r} \in \mathcal{N}(\mathbf{X})$ and therefore $\mathbf{X}\mathbf{r} = \mathbf{0}$. Now we are optimizing over $L(\mathbf{v}, \mathbf{r})$, which is **jointly convex** in $\mathbf{v}$ and $\mathbf{r}$, because its Hessian is PSD. Let's show that this is indeed the case:

$$\nabla_\mathbf{r}^2 L(\mathbf{v}, \mathbf{r}) = 2\mathbf{I} \succeq \mathbf{0}$$

$$\nabla_\mathbf{r}\nabla_\mathbf{v} L(\mathbf{v}, \mathbf{r}) = \nabla_\mathbf{v}\nabla_\mathbf{r} L(\mathbf{v}, \mathbf{r}) = \mathbf{0}$$

$$\nabla_\mathbf{v}^2 L(\mathbf{v}, \mathbf{r}) = 2\mathbf{X}\mathbf{X}^\top\mathbf{X}\mathbf{X}^\top + 2\lambda\mathbf{X}\mathbf{X}^\top \succeq \mathbf{0}$$

Since the cross terms of the Hessian are $\mathbf{0}$, it suffices that $\nabla_\mathbf{r}^2 L(\mathbf{v}, \mathbf{r})$ and $\nabla_\mathbf{v}^2 L(\mathbf{v}, \mathbf{r})$ are PSD to establish joint convexity. With joint convexity established, we can set the gradient to $\mathbf{0}$ w.r.t $\mathbf{r}$ and $\mathbf{v}$ and obtain the global minimum:

$$\nabla_\mathbf{r} L(\mathbf{v}, \mathbf{r}^*) = 2\mathbf{r}^* = \mathbf{0} \implies \mathbf{r}^* = \mathbf{0}$$

Note that $\mathbf{r}^* = \mathbf{0}$ just so happens in to be in $\mathcal{N}(\mathbf{X})$, so it is a feasible point.

$$\begin{aligned}
\nabla_\mathbf{v} L(\mathbf{v}^*, \mathbf{r}^*) = 2\mathbf{X}\mathbf{X}^\top\mathbf{X}\mathbf{X}^\top\mathbf{v}^* - 2\mathbf{X}\mathbf{X}^\top\mathbf{y} + 2\lambda\mathbf{X}\mathbf{X}^\top\mathbf{v}^* &= \mathbf{0} \\
\implies \mathbf{X}\mathbf{X}^\top(\mathbf{X}\mathbf{X}^\top + \lambda\mathbf{I})\mathbf{v}^* &= \mathbf{X}\mathbf{X}^\top(\mathbf{y}) \\
\implies \mathbf{v}^* &= (\mathbf{X}\mathbf{X}^\top + \lambda\mathbf{I})^{-1}\mathbf{y}
\end{aligned}$$

Note that $\mathbf{X}\mathbf{X}^\top + \lambda\mathbf{I}$ is positive definite and therefore invertible, so we can compute $(\mathbf{X}\mathbf{X}^\top + \lambda\mathbf{I})^{-1}\mathbf{y}$. Even though $(\mathbf{X}\mathbf{X}^\top + \lambda\mathbf{I})^{-1}\mathbf{y}$ is a critical point for which the gradient is $\mathbf{0}$, it must achieve the global minimum because the objective is jointly convex. We conclude that

$$\mathbf{w}^* = \mathbf{X}^\top(\mathbf{X}\mathbf{X}^\top + \lambda\mathbf{I})^{-1}\mathbf{y}$$

and arrive at the same solution as in the previous derivation.

1.3 Non-i.i.d. Case

So far we have assumed the special i.i.d. case of ridge regression, where

$$\mathbf{Y}|\mathbf{W} \sim \mathcal{N}(\mathbf{X}\mathbf{W}, \sigma^2\mathbf{I}), \quad \mathbf{W} \sim \mathcal{N}(\mathbf{0}, \sigma_h^2\mathbf{I})$$

In the non-i.i.d case we consider arbitrary covariance matrices:

$$\mathbf{Y}|\mathbf{W} \sim \mathcal{N}(\mathbf{X}\mathbf{W}, \Sigma_\mathbf{Z}), \quad \mathbf{W} \sim \mathcal{N}(\mathbf{0}, \Sigma_\mathbf{W})$$

As we've seen already, the solution in this case can be expressed in two forms, either the familiar case

$$\mathbf{w}^* = (\mathbf{X}^\top\boldsymbol{\Sigma}_\mathbf{Z}^{-1}\mathbf{X} + \boldsymbol{\Sigma}_\mathbf{W}^{-1})^{-1}\mathbf{X}^\top\boldsymbol{\Sigma}_\mathbf{Z}^{-1}\mathbf{y}$$

or the case that we desire in kernel ridge regression

$$\mathbf{w}^* = \boldsymbol{\Sigma}_\mathbf{W}\mathbf{X}^\top(\mathbf{X}\boldsymbol{\Sigma}_\mathbf{W}\mathbf{X}^\top + \boldsymbol{\Sigma}_\mathbf{Z})^{-1}\mathbf{y}$$

The principal difference in the non-i.i.d case is that we are computing $\mathbf{X}\boldsymbol{\Sigma}_\mathbf{W}\mathbf{X}^\top$ as opposed to $\mathbf{X}\mathbf{X}^\top$.

1.4 Kernels

Having derived the kernel ridge regression formulation for the raw data matrix $\mathbf{X}$, we can apply the exact same logic to the augmented data matrix $\boldsymbol{\Phi}$ and replace the optimal expression with

$$\mathbf{w}^* = \boldsymbol{\Phi}^\top(\boldsymbol{\Phi}\boldsymbol{\Phi}^\top + \lambda\mathbf{I})^{-1}\mathbf{y}$$

Let's explore the $\boldsymbol{\Phi}\boldsymbol{\Phi}^\top$ term in kernel ridge regression in more detail:

$$\boldsymbol{\Phi}\boldsymbol{\Phi}^\top = \begin{pmatrix} — & \boldsymbol{\phi}_1^\top & — \\ — & \boldsymbol{\phi}_2^\top & — \\ & \vdots & \\ — & \boldsymbol{\phi}_n^\top & — \end{pmatrix} \begin{pmatrix} | & | & & | \\ \boldsymbol{\phi}_1 & \boldsymbol{\phi}_2 & \cdots & \boldsymbol{\phi}_n \\ | & | & & | \end{pmatrix} = \begin{pmatrix} \boldsymbol{\phi}_1^\top\boldsymbol{\phi}_1 & \boldsymbol{\phi}_1^\top\boldsymbol{\phi}_2 & \cdots \\ \boldsymbol{\phi}_2^\top\boldsymbol{\phi}_1 & \ddots & \\ \vdots & & \boldsymbol{\phi}_n^\top\boldsymbol{\phi}_n \end{pmatrix}$$

Each entry $\boldsymbol{\Phi}\boldsymbol{\Phi}_{ij}^\top$ is a dot product between $\phi(\mathbf{x}_i)$ and $\phi(\mathbf{x}_j)$ and can be interpreted as a similarity measure:

$$\boldsymbol{\Phi}\boldsymbol{\Phi}_{ij}^\top = \langle\boldsymbol{\phi}_i, \boldsymbol{\phi}_j\rangle = \langle\phi(\mathbf{x}_i), \phi(\mathbf{x}_j)\rangle = k(\mathbf{x}_i, \mathbf{x}_j)$$

where $k(.,.)$ is the **kernel** function. The kernel function takes raw-feature inputs and outputs their inner product in the augmented feature space. We denote the matrix of $k(\mathbf{x}_i, \mathbf{x}_j)$ terms as the **Gram matrix** and denote it as $\mathbf{K}$:

$$\mathbf{K} = \mathbf{\Phi}\mathbf{\Phi}^\top = \begin{pmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & k(\mathbf{x}_1, \mathbf{x}_2) & \dots \\ k(\mathbf{x}_2, \mathbf{x}_1) & \ddots & \\ \vdots & & k(\mathbf{x}_n, \mathbf{x}_n) \end{pmatrix}$$

Formally, $k(\mathbf{x}_i, \mathbf{x}_j)$ is defined to be a valid kernel function if either of the following definitions are met:

- There exists a feature map $\phi(.)$ such that $\forall \mathbf{x}_i, \mathbf{x}_j, \quad k(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$
- For all sets $\mathcal{D} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$, the Gram matrix $\mathbf{K}(\mathcal{D})$ is PSD

We will now state some basic properties of kernels.

- Given two valid kernels k_a and k_b, their linear combination

  $$k(\mathbf{x}_i, \mathbf{x}_j) = \alpha k_a(\mathbf{x}_i, \mathbf{x}_j) + \beta k_b(\mathbf{x}_i, \mathbf{x}_j)$$

  where $\alpha, \beta \geq 0$ is also a valid kernel. We can show this from the second property:

  $$\forall \mathbf{v} \in \mathbb{R}^n, \mathbf{v}^\top(\alpha \mathbf{K}_a + \beta \mathbf{K}_b)\mathbf{v} = \alpha \mathbf{v}^\top \mathbf{K}_a \mathbf{v} + \beta \mathbf{v}^\top \mathbf{K}_b \mathbf{v} \geq 0$$

- Given a positive semidefinite matrix $\mathbf{\Sigma}$,

  $$k(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^\top \mathbf{\Sigma} \phi(\mathbf{x}_j)$$

  is a valid kernel. We can show this from the first property: $\widetilde{\phi}(\mathbf{x}_i) = \mathbf{\Sigma}^{\frac{1}{2}} \phi(\mathbf{x}_i)$

- Given a valid kernel k_a,

  $$k(\mathbf{x}_i, \mathbf{x}_j) = f(\mathbf{x}_i) f(\mathbf{x}_j) k_a(\mathbf{x}_i, \mathbf{x}_j)$$

  is a valid kernel. We can show this from the first property: $\widetilde{\phi}(\mathbf{x}_i) = f(\mathbf{x}_i)\phi(\mathbf{x}_i)$

Computing the each Gram matrix entry $k(\mathbf{x}_i, \mathbf{x}_j)$ can be done in a straightforward fashion if we apply the feature map to $\mathbf{x}_i$ and $\mathbf{x}_j$ and then take their dot product in the augmented feature space — this takes $O(d)$ time, where d is the dimensionality of the problem in the augmented feature space. However, if we use the **kernel trick**, we can perform this operation in $O(\ell + \log p)$ time, where ℓ is the dimensionality of the problem in the raw feature space and p is the degree of the polynomials in the augmented feature space.

2 Kernel Trick

Suppose that we are computing $k(\mathbf{x}, \mathbf{z})$, using a p-degree polynomial feature map that maps ℓ dimensional inputs to $d = O(\ell^p)$ dimensional outputs. Let's take $p = 2$ and $\ell = 2$ as an example. Define the polynomial feature map as

$$\phi(\mathbf{x}) = \begin{bmatrix} x_1^2 & x_2^2 & \sqrt{2}x_1x_2 & \sqrt{2}x_1 & \sqrt{2}x_2 & 1 \end{bmatrix}^\top$$

the kernel function can be expressed as

$$\begin{aligned} k(\mathbf{x}, \mathbf{z}) &= \phi(\mathbf{x})^\top \phi(\mathbf{z}) \\ &= \begin{bmatrix} x_1^2 & x_2^2 & \sqrt{2}x_1x_2 & \sqrt{2}x_1 & \sqrt{2}x_2 & 1 \end{bmatrix}^\top \begin{bmatrix} z_1^2 & z_2^2 & \sqrt{2}z_1z_2 & \sqrt{2}z_1 & \sqrt{2}z_2 & 1 \end{bmatrix} \\ &= x_1^2z_1^2 + x_2^2z_2^2 + 2x_1z_1x_2z_2 + 2x_1z_1 + 2x_2z_2 + 1 \\ &= (x_1^2z_1^2 + 2x_1z_1x_2z_2 + x_2^2z_2^2) + 2x_1z_1 + 2x_2z_2 + 1 \\ &= (x_1z_1 + x_2z_2)^2 + 2(x_1z_1 + x_2z_2) + 1 \\ &= (\mathbf{x}^\top\mathbf{z})^2 + 2\mathbf{x}^\top\mathbf{z} + 1 \\ &= (\mathbf{x}^\top\mathbf{z} + 1)^2 \end{aligned}$$

We can compute $k(\mathbf{x}, \mathbf{z})$ either by

1. Raising the inputs to the augmented feature space and take their inner product
2. Computing $(\mathbf{x}^\top\mathbf{z} + 1)^2$, which involves an inner product of the raw-feature inputs

Clearly, the latter option is much cheaper to calculate, taking $O(\ell + \log p)$ time, instead of $O(\ell^p)$ time. In fact, this concept generalizes for any arbitrary ℓ and p, and for p-degree polynomial features, we have that

$$k(\mathbf{x}, \mathbf{z}) = (\mathbf{x}^\top\mathbf{z} + 1)^p$$

The kernel trick makes computations significantly cheaper to perform, making kernelization much more appealing! The takeaway here is that no matter what the degree p is, the computational complexity is the same — it is only dependent on the dimensionality of the raw feature space!

Note that we can equivalently express the degree-2 polynomial features problem using the more natural mapping

$$\widetilde{\phi(\mathbf{x})} = \begin{bmatrix} x_1^2 & x_2^2 & x_1x_2 & x_1 & x_2 & 1 \end{bmatrix}^\top$$

in which case the kernel function would be expressed as

$$k(\mathbf{x}, \mathbf{z}) = \widetilde{\phi(\mathbf{x})}^\top \mathbf{\Sigma} \widetilde{\phi(\mathbf{z})} = (\mathbf{x}^\top\mathbf{z} + 1)^2, \quad \mathbf{\Sigma} = \text{Diag}\begin{pmatrix} 1 & 1 & 2 & 2 & 2 & 1 \end{pmatrix}$$

Thus we can view kernel ridge regression with the kernel trick in two ways:

1. i.i.d. prior $\mathbf{W} \sim \mathcal{N}\left(\mathbf{0}, \text{Diag}\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}\right)$, using the feature mapping $\phi(\mathbf{x})$

2. non-i.i.d prior $\mathbf{W} \sim \mathcal{N}\left(\mathbf{0}, \text{Diag}\begin{pmatrix}1 & 1 & 2 & 2 & 2 & 1\end{pmatrix}\right)$, using the feature mapping $\widetilde{\phi(\mathbf{x})}$ (note that the kernel trick is only applicable for this specific setting of $\mathbf{\Sigma}$ — it does not necessarily apply to arbitrary $\mathbf{\Sigma}$.)

2.1 Computational Analysis

Back to the original question: in ridge regression, should we compute

$$\mathbf{w}^* = \mathbf{\Phi}^\top(\mathbf{\Phi}\mathbf{\Phi}^\top + \lambda\mathbf{I})^{-1}\mathbf{y}$$

or

$$\mathbf{w}^* = (\mathbf{\Phi}^\top\mathbf{\Phi} + \lambda\mathbf{I})^{-1}\mathbf{\Phi}^\top\mathbf{y}$$

Let's compare their computational complexities. Suppose you are given an arbitrary test point $\mathbf{z} \in \mathbb{R}^\ell$, and you would like to compute its predicted value $\hat{\mathbf{y}}$. Let's see how these values are calculated in each case:

1. Kernelized

$$\hat{\mathbf{y}} = \langle\phi(\mathbf{z}), \mathbf{w}^*\rangle = \phi(\mathbf{z})^\top\mathbf{\Phi}^\top(\mathbf{\Phi}\mathbf{\Phi}^\top + \lambda\mathbf{I})^{-1}\mathbf{y} = \begin{bmatrix}k(\mathbf{x}_1, \mathbf{z}) & \dots & k(\mathbf{x}_n, \mathbf{z})\end{bmatrix}(\mathbf{K} + \lambda\mathbf{I})^{-1}\mathbf{y}$$

   Computing the $\mathbf{K}$ term takes $O(n^2(\ell + \log p))$, and inverting the matrix takes $O(n^3)$. These two computations dominate, for a total computation time of $O(n^3 + n^2(\ell + \log p))$.

2. Non-kernelized

$$\hat{\mathbf{y}} = \langle\phi(\mathbf{z}), \mathbf{w}^*\rangle = \phi(\mathbf{z})^\top(\mathbf{\Phi}^\top\mathbf{\Phi} + \lambda\mathbf{I})^{-1}\mathbf{\Phi}^\top\mathbf{y}$$

   Computing the $\mathbf{\Phi}^\top\mathbf{\Phi}$ term takes $O(d^2n)$, and inverting the matrix takes $O(d^3)$. These two computations dominate, for a total computation time of $O(d^3 + d^2n)$.

Here is the takeaway: if $d \ll n$, the non-kernelized method is preferable. Otherwise if $n \ll d$, the kernelized method is preferable.