EECS 189 Introduction to Machine Learning
Fall 2020

DIS4

This discussion was released **Friday, September 25**.

This discussion material will cover parts of Problem 3 (Kernel Ridge Regression: Theory) and Problem 4 (Learning $1d$ function with kernel regression) in HW4. It aims to give you a better understanding of kernel ridge regression, and it should get you started on the homework problems.

- We will first start with the problem on (**using Jamboard**) deriving the prediction of kernel ridge regression for a test point $\mathbf{x}$ via the *augmented design matrix perspective*.
  ($\sim$ 10 minutes.)
- Then we will bring everyone back to the main room.
- Next we will move to Jupyter notebook part and play around with $1d$ function with kernel ridge regression. We will try to do some visualizations on kernel ridge regression on $1d$, study the effect of two hyperparameters in kernel ridge regression, as well as using k-fold cross-validation to pick hyperparameter.

As a reminder, if you have questions, we will answer them via the queue at oh.eecs189.org. Once you complete the first part, you could go to the Jupyter notebook.

1 Kernel Ridge Regression: Theory

In traditional ridge regression, we are 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 practical settings we don't want to work with just the raw feature space, so we lift/distill the data points using features and replace $\mathbf{X}$ with $\mathbf{\Phi} \in \mathbb{R}^{n \times d}$, where $\phi_i^\top = \phi(\mathbf{x}_i) \in \mathbb{R}^d$. Then we solve a well-defined optimization problem that involves the matrix $\mathbf{\Phi}$ and $\mathbf{y}$ to find 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 (often 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 n variables (where n is potentially much smaller than d), 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.

When people talk about "the kernel trick," they usually mean the following: imagine that we have some machine learning algorithm, which only uses data vectors to compute scalar products with other data vectors. Then we can substitute the operation of taking inner products $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$ with some other function $K(\mathbf{x}_i, \mathbf{x}_j)$ and obtain a new ML algorithm! There is however, another interesting consideration: if our algorithm only operates with inner products, and we do featurization, maybe

we could use features so that those scalar products $\langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle =: K(\mathbf{x}_i, \mathbf{x}_j)$ were easy to compute (e.g. without explicitly writing out long vectors $\phi(\mathbf{x}_i)$ and $\phi(\mathbf{x}_j)$). As it turns out, those ideas are closely connected: kernels and features provide two different views of the same thing, and it may sometimes be more convenient to think about one and sometimes about the other.

In this problem we will derive this connection between features and kernels, observe which features correspond to some particular kernels and vice versa, and see how kernel perspective can help with computation.

(a) As we already know, the following procedure gives us the solution to ridge regression in feature space:

$$\arg\min_{\mathbf{w}} \|\mathbf{\Phi}\mathbf{w} - \mathbf{y}\|_2^2 + \lambda \|\mathbf{w}\|_2^2 \tag{1}$$

To obtain the prediction for a test point $\mathbf{x}$ one needs to compute $\phi(\mathbf{x})\hat{\mathbf{w}}$, where $\hat{\mathbf{w}}$ is the solution to (1). In this part we will show how $\phi(\mathbf{x})\hat{\mathbf{w}}$ can be computed using only the kernel $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)\phi(\mathbf{x}_j)^\top$. Denote the following two objects:

$$\mathbf{k}(\mathbf{x}) := [K(\mathbf{x}, \mathbf{x}_1), K(\mathbf{x}, \mathbf{x}_2), \ldots, K(\mathbf{x}, \mathbf{x}_n)]^\top, \quad \mathbf{K} = \{K(\mathbf{x}_i, \mathbf{x}_j)\}_{i,j=1}^n$$

Where the matrix $\mathbf{K}$ is the matrix filled with pairwise kernel similarity computations among the training points.

Show that

$$\phi(\mathbf{x})\hat{\mathbf{w}} = \mathbf{k}(\mathbf{x})^\top (\mathbf{K} + \lambda \mathbf{I})^{-1} \mathbf{y}.$$

In other words, if we define $\hat{\alpha} := (\mathbf{K} + \lambda \mathbf{I})^{-1}\mathbf{y}$, then

$$\phi(\mathbf{x})\hat{\mathbf{w}} = \sum_{i=1}^{n} \alpha[i] K(\mathbf{x}, \mathbf{x}_i)$$

— our prediction is a linear combination of kernel functions at different data points.

HINT: use problem 6b of HW1 to write $\hat{\mathbf{w}}$ using the Moore–Penrose inverse (pseudoinverse), and then use the explicit expression for pseudoinverse for matrices with linearly independent rows. After that you will just need to understand how $\mathbf{K}$ and $\mathbf{k}(\mathbf{x})$ can be expressed through $\mathbf{\Phi}$ and $\phi(\mathbf{x})$.

2 Learning 1d function with kernel regression

(a) **Do the tasks from part a of the associated jupyter notebook and report the results.**

(b) **Do the tasks from part b of the associated jupyter notebook and report the results.**

(c) **Do the tasks from part c of the associated jupyter notebook and report the results.**

(d) **Do the tasks from part d of the associated jupyter notebook and report the results.**

(e) **Do the tasks from part e of the associated jupyter notebook and report the results.**

(f) **Do the tasks from part f of the associated jupyter notebook and report the results.**

(g) **Do the tasks from part g of the associated jupyter notebook and report the results.**

(h) **Do the tasks from part h of the associated jupyter notebook and report the results.**

(i) **Do the tasks from part i of the associated jupyter notebook and report the results.**

Contributors:

- Alexander Tsigler
- Anant Sahai
- Josh Sanz
- Philipp Moritz
- Rahul Arya
- Yaodong Yu