EE 510: Mathematical Foundations of Machine Learning **Winter 2021**

Demo: Kernel Ridge Regression

Instructor Name: John Lipor

Introduction to Kernels

In this demo, you will implement a version of Ridge Regression for nonlinear functions known as Kernel Ridge Regression (KRR). Recall in Homework 4, Problem 3 you used the *linear* least-squares formulation to fit a nonlinear function. To do this, you mapped the one-dimensional "feature" vector x to a $(p+1)$-dimensional vector corresponding to a polynomial of degree p. Mathematically, this map looked like

$$x \in \mathbb{R} \mapsto \Phi(x) = \begin{bmatrix} 1 \\ x \\ x^2 \\ \vdots \\ x^p \end{bmatrix} \in \mathbb{R}^{p+1}.$$

You then performed least-squares regression to the transformed data $\Phi(x)$ to obtain function that was linear in $\mathbb{R}^{p+1}$ but nonlinear in $\mathbb{R}$. Revisiting our least-squares cost function, you solved the following problem

$$\hat{w} = \arg\min_{w \in \mathbb{R}^{p+1}} \left\| \begin{bmatrix} \Phi(x_1)^T \\ \Phi(x_2)^T \\ \vdots \\ \Phi(x_N)^T \end{bmatrix} w - \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \right\|_2^2 = \arg\min_{w \in \mathbb{R}^{p+1}} \sum_{i=1}^{N} \left(\Phi(x_i)^T w - y_i \right)^2.$$

The above gives us a way to easily fit arbitrary functions to data, which seems great! We saw that this mapping can overfit the data if we let p be large, but of course this can be fixed by regularization (i.e., ridge regression). The bigger problem with the above formulation is that as the dimension of $\Phi(x)$ grows, the computational complexity of this problem increases to the point where it becomes infeasible. To avoid this problem, we use something known in machine learning as the *kernel trick*, which hinges on the following two properties

- Many machine learning algorithms depend on $\Phi(x)$ only via inner products $\langle \Phi(x), \Phi(y) \rangle$.
- For certain Φ, the function
$$k(x, y) = \langle \Phi(x), \Phi(y) \rangle$$
can be computed effeciently, even if p is huge or infinite.

When the function $k(x, y)$ above satisfies some important properties described here, it is known as an *inner product kernel*. For our purposes, it is sufficient to know that we can take advantage of such a mapping as long as

- The function $k(x, y)$ is symmetric, i.e., $k(x, y) = k(y, x)$.
- The matrix K whose (i, j)th entry is $k(x_i, x_j)$ is symmetric and positive semi-definite.

The above two properties mean that k is a *positive definite kernel*, which implies that k is an inner product kernel. Another way to view kernels is that we are making use of the fact that the data is linear in some high-dimensional space. We map to that space via the function Φ, which is computationally inefficient, but we can do inner products in that space using the function k, which is computationally efficient.

Task 1: Kernelizing Ridge Regression

To make use of kernels in ridge regression, we need to write the solution down in terms of inner products between the vectors $\Phi(x)$ (or simply x if we consider the original space). Recall the solution to ridge regression

$$\hat{w} = \left(X^T X + \lambda I\right)^{-1} X^T y.$$

To this point, we've been most interested in finding w explicitly. However, the basic usage of w is to apply it to a new "test" vector x_t to predict the resulting label y_t. The resulting prediction is simply the inner product

$$y_t = \hat{w}^T x_t = x_t^T \hat{w} = x_t^T \left(X^T X + \lambda I\right)^{-1} X^T y.$$

Since the vectors x_i are the *rows* of X, the matrix $X^T X$ above is not a Gram matrix and does not depend on inner products. We'll rewrite this using the matrix inversion lemma

$$(P + QRS)^{-1} = P^{-1} - P^{-1}Q(R^{-1} + SP^{-1}Q)^{-1}SP^{-1}$$

with

$$P = \lambda I \qquad Q = X^T \qquad R = I \qquad S = X.$$

The resulting simplified prediction for a single test point x_t is

$$y_t = y^T (XX^T + \lambda I)^{-1} X x_t = \hat{w}^T x_t. \tag{1}$$

- Find all the inner products $\langle x_i, x_j \rangle$ in (1) and replace them with kernels $k(x_i, x_j)$. Make use of the matrix K defined above, and call $k(x_t)$ the vector whose ith element is $k(x_i, x_t)$.
- Write a concise expression for determining the vector $\hat{y} \in \mathbb{R}^N$ of estimated labels for each vector (*row*) in X. This gives you the estimated label/value at each point in your training set.
- Raise your hand when you've finished this part, and I'll come check your answer.
- If at the **end** of this demo you still have time to spare, try to derive (1) on your own.

Task 2: Train KRR

Your second task is to complete the first portion of the `KRR.m` function. The function takes in a set of feature vectors and labels, a ridge regression tuning parameter λ, the tuning parameter σ for the radial basis function (RBF) kernel given below. The output is the corresponding vector of labels for each test point. The RBF and polynomial kernels are given below. Implementing the polynomial kernel is optional.

1. $k(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|_2^2}{2\sigma^2}\right)$, where $\sigma > 0$ (radial basis function/RBF kernel)
2. $k(x_i, x_j) = \left(x_i^T x_j + c\right)^p$, where $c \in \mathbb{R}$ and $p \in \mathbb{N}$ (polynomial kernel of degree p)

- Implement the training portion of `KRR.m` as instructed above.
- Test your function on the first portion of `syntheticTest.m`. Try a few different values for the various tuning parameters. What effect does the parameter σ have?
- Now vary the two frequency parameters `f1, f2`. How does this change the best choice of σ?
- Raise your hand once you've finished this part, and I'll come by to look at your work.

Task 3: Test KRR

Your final task is to complete the testing portion of the `KRR.m` function. This function makes use of `Xtest`, a set of feature vectors with unknown labels (called test data). The output is the set of predicted labels for each point in the test data set.

- Implement the testing portion of `KRR.m` using your kernelized verison of (1).
- Test your predictions on the test data given in the second portion of `syntheticTest.m`.
- Calculate the resulting mean-squared error (MSE, i.e., the least-squares cost) for the test data.
- If time allows, compute the MSE for a range of tuning parameters and display the result.