# 4033/5033 Assignment: Kernel Ridge Regression

Sasank Sribhashyam

In this assignment, we will implement a new method called Accelerated Kernel Ridge Regression (AKRR) and show its training time complexity is merely $O(mnp + m^2n)$ instead of $O(n^2p + n^3)$, where $m$ is a hyper-parameter. We will empirically show AKRR performs similarly to KRR when $m$ is way smaller than $n$, which justifies the efficacy of this new method. In experiment, we will evaluate the AKRR model on the Crime and Community data set. Split the data set into a training set $S$ and a testing set $T$. Recall the optimal KRR model has the form

$$\beta = \sum_{i=1}^{n} \alpha_i \phi(x_i), \tag{1}$$

where $x_1, \dots, x_n$ are all the training data points. In AKRR, we approximate the above $\beta$ by

$$\tilde{\beta} = \sum_{i=1}^{m} \alpha_i \phi(x_i), \tag{2}$$

where $m$ is a hyper-parameter in $[0, n]$. In other words, AKRR expresses the optimal model using only the first $m$ training instances (instead of all as in KRR). The rest part of AKRR is the same as KRR i.e., we plug (2) back into the objective function

$$J(\tilde{\beta}) = \sum_{i=1}^{n} (\phi(x_i)^T \tilde{\beta} - y_i)^2 + \lambda \tilde{\beta}^T \tilde{\beta}, \tag{3}$$

and get a dual objective function $J(\tilde{\alpha})$, where $\tilde{\alpha} = [\alpha_1, \dots, \alpha_m]^T$. Then, we find an $\tilde{\alpha}$ that minimizes $J(\tilde{\alpha})$.

Task 1. Derive the dual objective function $J(\tilde{\alpha})$ and express it in a matrix form using the following notations – $\tilde{K}_{nm} \in \mathbb{R}^{n \times m}$ is a kernel matrix with $\tilde{K}_{ij} = k(x_i, x_j)$ and $i = 1, \dots, n$ and $j = 1, \dots, m$. – $\tilde{K}_{mm} \in \mathbb{R}^{m \times m}$ is a kernel matrix with $\tilde{K}_{ij} = k(x_i, x_j)$ and $i, j = 1, \dots, m$. – $Y = [y_1, \dots, y_n]^T \in \mathbb{R}^n$ is a label vector.

To derive the dual objective function $J(\tilde{\alpha})$ for Accelerated Kernel Ridge Regression (AKRR) and express it in matrix form, we will follow the steps you've provided. Let's begin:

1. Start with the expression for the objective function $J(\tilde{\beta})$:

$$J(\tilde{\beta}) = \sum_{i=1}^{n} \left( \phi(x_i)^T \tilde{\beta} - y_i \right)^2 + \lambda \tilde{\beta}^T \tilde{\beta}$$

2. Substitute the expression for $\tilde{\beta}$ from equation (2):

$$J(\tilde{\beta}) = \sum_{i=1}^{n} \left( \phi(x_i)^T \left( \sum_{j=1}^{m} \alpha_j \phi(x_j) \right) - y_i \right)^2 + \lambda \left( \sum_{j=1}^{m} \alpha_j \phi(x_j) \right)^T \left( \sum_{j=1}^{m} \alpha_j \phi(x_j) \right)$$

3. Expand the square term in the first sum:

$$J(\tilde{\beta}) = \sum_{i=1}^{n} \Bigg[ \left( \phi(x_i)^T \left( \sum_{j=1}^{m} \alpha_j \phi(x_j) \right) \right)^2$$

$$- 2\phi(x_i)^T \left( \sum_{j=1}^{m} \alpha_j \phi(x_j) \right) y_i + y_i^2 \; ]$$

$$+ \lambda \left( \sum_{j=1}^{m} \alpha_j \phi(x_j) \right)^T \left( \sum_{j=1}^{m} \alpha_j \phi(x_j) \right)$$

4. Notice that $\sum_{j=1}^{m} \alpha_j \phi(x_j) y_i$ is a scalar (dot product), and $\sum_{j=1}^{m} \alpha_j \phi(x_j)$ is a vector (linear combination of the kernel functions). We can express these terms using matrices and vectors.

Let: - $\tilde{K}_{nm}$ be the kernel matrix between the first $n$ training instances and the first $m$ training instances. - $\tilde{K}_{mm}$ be the kernel matrix between the first $m$ training instances.

Using these matrices, we can write the above expression as:

$$J(\tilde{\beta}) = (\Phi\alpha - Y)^T (\Phi\alpha - Y) + \lambda\alpha^T \tilde{K}_{mm}\alpha$$

Where: - $\Phi$ is the $n \times m$ matrix with elements $\Phi_{ij} = \phi(x_i)^T \phi(x_j) = \tilde{K}_{nm}$. - $\alpha$ is the $m \times 1$ vector with elements $\alpha_i$. - $Y$ is the $n \times 1$ vector with elements $y_i$.

5. Rewrite the expression using matrix notation and the given notations:

$$J(\tilde{\alpha}) = (\tilde{K}_{nm}\alpha - Y)^T (\tilde{K}_{nm}\alpha - Y) + \lambda\alpha^T \tilde{K}_{mm}\alpha$$

6. This is the dual objective function $J(\tilde{\alpha})$ for Accelerated Kernel Ridge Regression (AKRR) expressed in matrix form.

<u>Task 2</u>. Derive the optimal $\tilde{\alpha}$ and express it in a matrix form using the above notations.

To derive the optimal $\tilde{\alpha}$ for Accelerated Kernel Ridge Regression (AKRR) and express it in matrix form, we find the value of $\tilde{\alpha}$ that minimizes the dual objective function $J(\tilde{\alpha})$ derived earlier.

The objective function is:

$$J(\tilde{\alpha}) = (\tilde{K}_{nm}\alpha - Y)^T (\tilde{K}_{nm}\alpha - Y) + \lambda\alpha^T \tilde{K}_{mm}\alpha$$

To find the optimal $\tilde{\alpha}$, we can take the derivative of $J(\tilde{\alpha})$ with respect to $\alpha$ and set it equal to zero:

$$\frac{dJ(\tilde{\alpha})}{d\alpha} = 2\tilde{K}_{nm}^T (\tilde{K}_{nm}\alpha - Y) + 2\lambda\tilde{K}_{mm}\alpha = 0$$

Now, let's solve for $\tilde{\alpha}$:

$$\tilde{K}_{nm}^T (\tilde{K}_{nm}\alpha - Y) + \lambda\tilde{K}_{mm}\alpha = 0$$

$$\tilde{K}_{nm}^T \tilde{K}_{nm}\alpha - \tilde{K}_{nm}^T Y + \lambda\tilde{K}_{mm}\alpha = 0$$

$$\left(\tilde{K}_{nm}^T \tilde{K}_{nm} + \lambda \tilde{K}_{mm}\right) \alpha = \tilde{K}_{nm}^T Y$$

Now, we can express the optimal $\tilde{\alpha}$ in matrix form:

$$\alpha^* = \left(\tilde{K}_{nm}^T \tilde{K}_{nm} + \lambda \tilde{K}_{mm}\right)^{-1} \tilde{K}_{nm}^T Y$$

This is the optimal $\tilde{\alpha}$ for Accelerated Kernel Ridge Regression (AKRR) expressed in matrix form using the provided notations. It is obtained by solving the above linear system of equations. Once you have $\alpha^*$, you can use it to compute $\tilde{\beta}$ using Equation (2).

Task 3. Justify the time complexity for computing your optimal solution is $O(mnp + m^2n)$. You should first explain the time complexity for computing each part of the solution and then combine them.

To justify the time complexity for computing the optimal solution $\tilde{\alpha}$ in Accelerated Kernel Ridge Regression (AKRR), we'll break down the time complexity for each part of the solution and then combine them.

1. Computing $\tilde{K}_{nm}$: - $n$ is the number of training data points, and $m$ is a hyper-parameter. - Computing the kernel matrix $\tilde{K}_{nm}$ requires evaluating the kernel function for all pairs of data points between the first $n$ training instances and the first $m$ training instances. - The time complexity for computing $\tilde{K}_{nm}$ is $O(n \cdot m \cdot p)$, where $p$ is the time complexity of evaluating the kernel function for a pair of data points.

2. Computing $\tilde{K}_{mm}$: - Computing the kernel matrix $\tilde{K}_{mm}$ between the first $m$ training instances requires evaluating the kernel function for all pairs of these $m$ instances. - The time complexity for computing $\tilde{K}_{mm}$ is $O(m^2 \cdot p)$.

3. Solving the linear system: - After obtaining $\tilde{K}_{nm}$, $\tilde{K}_{mm}$, and $Y$, we need to solve the linear system of equations:

$$\left(\tilde{K}_{nm}^T \tilde{K}_{nm} + \lambda \tilde{K}_{mm}\right) \alpha = \tilde{K}_{nm}^T Y$$

- Solving a linear system typically involves matrix inversion or using techniques like Cholesky decomposition. - The time complexity for solving the linear system is usually dominated by the matrix inversion part, which is approximately $O(m^3)$ for a $m \times m$ matrix.

Now, let's combine these complexities:

- Computing $\tilde{K}_{nm}$: $O(n \cdot m \cdot p)$ - Computing $\tilde{K}_{mm}$: $O(m^2 \cdot p)$ - Solving the linear system: $O(m^3)$

To find the overall time complexity, we can take the dominant term, which is $O(m^3)$ for solving the linear system. Therefore, the overall time complexity for computing the optimal solution $\tilde{\alpha}$ in AKRR is $O(m^3)$.

In cases where $m$ is much smaller than $n$, this time complexity $O(m^3)$ is significantly more efficient than the naive Kernel Ridge Regression (KRR) time complexity of $O(n^2 \cdot p + n^3)$. This demonstrates the computational advantage of AKRR when dealing with large datasets.

Task 4. Implement your AKRR model with RBF kernel from scratch.

Task 5. Train the AKRR model on $S$ and evaluate it on $T$. Pick a proper hyper-parameter for the RBF kernel yourself, and report testing error versus $m$ in Figure 1. In this figure, pick 10 values of $m$ yourself but the last one must be $m = n$. (This is when AKRR becomes KRR.)

**Fig. 1.** ARKK Testing Error versus $m$.