

ME 343: Homework Assignment 1

Total number of points = 100.

Submission instructions: submit your homework using gradescope. Please register with entry code MPW2X3. This is required if you take this class for credit. We expect you to upload your answers as a PDF file along with a zip file containing your code. See Homework 1 and Homework 1 Code on gradescope.

You can create the PDF using any software you want. It is possible to write your homework paper using pen and paper, and take pictures using your phone. Make sure the lighting is sufficient. Create a single PDF with all the pages. There are many phone apps that can do this.

Problem 1: Ridge Regression

In class, we talked about Gaussian Process Regression. In this homework, we will look into another regression algorithm, ridge regression. Suppose we have input vectors $x^{(i)} \in \mathbb{R}^d$, and target variable $y^{(i)} \in \mathbb{R}$, $i = 1, 2, \dots, n$. Ridge regression solves for $\theta \in \mathbb{R}^d$ that minimizes the loss function

$$J(\theta) = \sum_{i=1}^n (y^{(i)} - \theta^\top x^{(i)})^2 + \lambda \|\theta\|_2^2$$

where λ is a hyper-parameter that controls the weight decay or regularization. The loss function can be conveniently rewritten in a matrix form. Define matrix $X \in \mathbb{R}^{n \times d}$, where the i th row in X is $x^{(i)}$. Let $y = [y^{(1)}, y^{(2)}, \dots, y^{(n)}]^\top \in \mathbb{R}^n$. Then the loss function becomes:

$$J(\theta) = \|X\theta - y\|_2^2 + \lambda \|\theta\|_2^2$$

1. 10 points. Show that the minimizer of $J(\theta)$ is $\theta = (X^\top X + \lambda I)^{-1} X^\top y$.
2. 10 points. From $X^\top X\theta + \lambda I\theta = X^\top y$, show that θ can be written as $\theta = X^\top \alpha$ and $\alpha = (XX^\top + \lambda I)^{-1} y$. [Hint: Prove $(X^\top X + \lambda I)^{-1} X^\top = X^\top (XX^\top + \lambda I)^{-1}$.]
3. 10 points. Show that the inference $\theta^\top x$ for a new data x can be written as

$$\sum_i \alpha_i k(x, x^{(i)})$$

where $k(x, x^{(i)}) = x^\top x^{(i)}$.

Problem 2: Kernel Methods

In the previous problem, we explored linear regression (interpolation) with ridge regression, trying to fit the data with a line. In most cases, linear interpolation is too simple to fit the data. In order to perform

non-linear interpolation, we can use non-linear feature functions $\phi(x^{(i)}) \in \mathbb{R}^m$, and then perform regression on $\phi(x^{(i)})$. And now the inference for a new data x becomes

$$\sum_i \alpha_i \phi^\top(x) \phi(x^{(i)})$$

Let's take an example in 2D with $x = (x_1, x_2)^\top \in \mathbb{R}^2$. We may wish to find ϕ such that

$$\phi^\top(x) \phi(x^{(i)}) = (x^\top x^{(i)})^2$$

If you expand the dot product you will see that the following function ϕ satisfies the equation above

$$\phi(x) = (x_1 x_1, x_1 x_2, x_2 x_1, x_2 x_2)^\top \in \mathbb{R}^4$$

The kernel trick consists in realizing that instead of using the function ϕ we can define a kernel function such that

$$\phi^\top(x) \phi(x^{(i)}) = k(x, x^{(i)})$$

In fact, what's special is that we can drop entirely $\phi(x)$ and use $k(x, x')$ only. The definition of $k(x, x')$ is the only thing we need in our algorithm. This opens the door to many different definitions of the kernel $k(x, x')$ and kernel ridge regression schemes.

Now let's define the concept of kernel ridge regression. We want to minimize:

$$J(\theta) = \sum_{i=1}^n (y^{(i)} - \theta^\top \phi(x^{(i)}))^2 + \lambda \|\theta\|_2^2$$

where $\phi(x^{(i)})$ maps $x^{(i)}$ to \mathbb{R}^m . The inference on new data x is

$$\sum_{i=1}^n \alpha_i \phi^\top(x) \phi(x^{(i)}) = \sum_{i=1}^n \alpha_i k(x, x^{(i)})$$

where $\alpha = (\lambda I + K)^{-1} y$ and $K \in \mathbb{R}^{n \times n}$ is the kernel matrix of the observation data: $K_{ij} = k(x^{(i)}, x^{(j)})$, for $i, j = 1, \dots, n$. In this problem, we will explore different types of kernel for kernel ridge regression.

We assume that you will write Python code for the homework using a Jupyter Notebook. That's the easiest way to write Python code and visualize results immediately. For more information see for example: <https://jupyter.org/install>. You can do your homework with other languages such as Matlab or Julia. However, the solution set will be written in Python. Later on this quarter, we will learn about TensorFlow and Keras, which is an API for TensorFlow written in Python.

1. 15 points. Implement the Gaussian or "RBF" kernel

$$k_{\text{RBF}(\sigma)}(x, x') = \exp\left(-\frac{\|x - x'\|^2}{2\sigma^2}\right)$$

and the polynomial kernel

$$k_{\text{poly}(a, d)}(x, x') = (a + x^\top x')^d$$

in a Jupyter Notebook. The kernel functions in Python should take $X \in \mathbb{R}^{n_1 \times d}$, $X' \in \mathbb{R}^{n_2 \times d}$ and relevant parameters as input, and return a matrix $M \in \mathbb{R}^{n_1 \times n_2}$ where

$$M_{ij} = k(X'_{i,:}, X_{j,:})$$

The linear kernel

$$k_{\text{linear}}(x, x') = x^\top x'$$

has been implemented for you as an example. Attach your code for these two functions in the submission. [You may find the `scipy` function `cdist(X1, X2, 'sqeuclidean')` in the package `scipy.spatial.distance` useful. It computes the squared Euclidean distance $\|u - v\|_2^2$ between the vectors u and v .]

2. Suppose we have the data set (observations) $\mathcal{D} = \{(-4, 2), (-1, 0), (0, 3), (2, 5)\}$. As we have seen previously, the solution will be of the form:

$$\sum_{i=1}^4 \alpha_i k(x, x^{(i)})$$

So the solution is a linear combination of $k(x, x^{(i)})$ with the points $x^{(i)}$ being $\{x^{(i)}\} = \{-4, -1, 0, 2\}$.

- (a) 5 points. Plot the four functions $x \mapsto k_{\text{poly}(1,3)}(x, x^{(i)})$ with $x^{(i)} \in \{-4, -1, 0, 2\}$ and for $x \in [-6, 6]$. The definition of $k_{\text{poly}(1,3)}$ is given above.
- (b) 5 points. Similarly, plot the four functions $x \mapsto k_{\text{RBF}(1)}(x, x^{(i)})$ with the same $x^{(i)}$ and interval for x . `RBF(1)` means the RBF kernel above with $\sigma = 1$.

The plot for linear kernel has been generated for you as an example.

Now consider a one-dimensional regression problem, where $x^{(i)} \in \mathbb{R}$. We'll fit this data using kernel ridge regression, and we'll compare the results using several different kernel functions. Because the input space is one-dimensional, we can easily visualize the results. In the zip file for this assignment, you'll find a training and test set, `krr-train.txt` and `krr-test.txt`. The data has been loaded for you. In machine learning, a training set is a dataset used to train a model, i.e., the observations. And after the learning is done, the trained model is evaluated on the test set. In our case the evaluation metrics is the average square loss and it has been implemented for you. See the instance function `score` in the class `KernelRidgeRegression`.

3. 15 points. Implement the function `predict` in `Kernel_Learning` class, and the function `train_kernel_ridge_regression`. In order to make it flexible to use different kernels, we use the `sklearn` wrapper for our kernel ridge regression. Attach your code of the instance function `predict` and the function `train_kernel_ridge_regression` in the submission. [`sklearn` is a powerful machine learning library for Python, built on NumPy, SciPy, and matplotlib.]
4. 10 points. Use the code provided to plot your fits to the training data for the RBF kernel with a fixed regularization parameter of 0.0001 for 3 different values of σ : 0.01, 0.1, and 1.0. What values of σ do you think would be more likely to overfit, and which less?
5. 10 points. Use the code provided to plot your fits to the training data for the RBF kernel with a fixed σ of 0.02 and 4 different values of the regularization parameter λ : 0.0001, 0.01, 0.1, and 2.0. What happens to the prediction function as $\lambda \rightarrow \infty$?
6. 10 points. Find the hyperparameter settings (including kernel parameters and the regularization parameter λ) for the Gaussian kernel and the polynomial kernel that give a low average square loss on the test set. Report the average square error on the test set and the ideal hyperparameter combination for each kernel. Plot the prediction on the test set. For the RBF kernel, the average square error given by `score` on the test set should be lower than 0.015, and for the polynomial kernel, it should be lower than 0.035.