DSC 140A - Homework 05

Due: Wednesday, May 8

Instructions: Write your solutions to the following problems either by typing them or handwriting them on another piece of paper. Show your work or provide justification unless otherwise noted. If you write code to solve a problem, include the code by copy/pasting or as a screenshot. You may use `numpy`, `matplotlib` (or another plotting library), and any standard library module, but no other third-party libraries unless specified. Submit homeworks via Gradescope by 11:59 PM.

Problem 1.

In this problem, you will derive the solution to the ridge regression optimization problem.

Recall that the ridge regression regularized risk function is

$$\tilde{R}(\vec{w}) = \frac{1}{n}\sum_{i=1}^{n}(\vec{\phi}(\vec{x}^{(i)}) \cdot \vec{w} - y_i)^2 + \lambda\|\vec{w}\|^2.$$

Here, $\vec{\phi}(\vec{x})$ is a feature map. Also recall that this risk function can be equivalently written in matrix-vector form as

$$\tilde{R}(\vec{w}) = \frac{1}{n}\|\Phi\vec{w} - \vec{y}\|^2 + \lambda\|\vec{w}\|^2,$$

where Φ is the design matrix; its ith row is $\vec{\phi}(\vec{x}^{(i)})$.

a) Show that $\tilde{R}(\vec{w}) = \frac{1}{n}\left(\vec{w}^T\Phi^T\Phi\vec{w} - 2\vec{w}^T\Phi^T\vec{y} + \vec{y}^T\vec{y}\right) + \lambda\vec{w}^T\vec{w}$.

Solution: Using the fact that, for any vector $\vec{u}$, $\|vecu\|^2 = \vec{u}^T\vec{u}$, we have:

$$\begin{aligned}
\tilde{R}(\vec{w}) &= \frac{1}{n}\|\Phi\vec{w} - \vec{y}\|^2 + \lambda\|\vec{w}\| \\
&= \frac{1}{n}(\Phi\vec{w} - \vec{y})^T(\Phi\vec{w} - \vec{y}) + \lambda\vec{w}^T\vec{w} \\
&= \frac{1}{n}(\vec{w}^T\Phi^T - \vec{y}^T)(\Phi\vec{w} - \vec{y}) + \lambda\vec{w}^T\vec{w} \\
&= \frac{1}{n}\left(\vec{w}^T\Phi^T\Phi\vec{w} - \vec{w}^T\Phi^T\vec{y} - \vec{y}^T\Phi\vec{w} + \vec{y}^T\vec{y}\right) + \lambda\vec{w}^T\vec{w}
\end{aligned}$$

$\vec{y}^T\Phi\vec{w}$ is a scalar, and any scalar transposed is just itself. So $(y^T\Phi\vec{w})^T = \vec{w}^T\Phi^T\vec{y}$, and we have:

$$= \frac{1}{n}\left(\vec{w}^T\Phi^T\Phi\vec{w} - 2\vec{w}^T\Phi^T\vec{y} - \vec{y}^T\vec{y}\right) + \lambda\vec{w}^T\vec{w}$$

b) So far this quarter, we have seen a few vector calculus identities. For example, we know that $\frac{d}{d\vec{w}}(\vec{w}^T\vec{w}) = 2\vec{w}$.

Using these identities, show that

$$\frac{d}{d\vec{w}}\tilde{R}(\vec{w}) = \frac{1}{n}\left(2\Phi^T\Phi\vec{w} - 2\Phi^T\vec{y}\right) + 2\lambda\vec{w}.$$

Solution:

$$\begin{aligned}
&\frac{d}{d\vec{w}}\left[\frac{1}{n}\left(\vec{w}^T\Phi^T\Phi\vec{w} - 2\vec{w}^T\Phi^T\vec{y} - \vec{y}^T\vec{y}\right) + \lambda\vec{w}^T\vec{w}\right] \\
&\qquad = \left[\frac{1}{n}\left(\frac{d}{d\vec{w}}\vec{w}^T\Phi^T\Phi\vec{w} - \frac{d}{d\vec{w}}2\vec{w}^T\Phi^T\vec{y} - \frac{d}{d\vec{w}}\vec{y}^T\vec{y}\right) + \frac{d}{d\vec{w}}\lambda\vec{w}^T\vec{w}\right] \\
&\qquad = \frac{1}{n}\left(2\Phi^T\Phi\vec{w} - \Phi^T\vec{y} - \vec{0}\right) + 2\lambda\vec{w}
\end{aligned}$$

c) Show that the minimizer of $\tilde{R}(\vec{w})$ is $\vec{w}^* = (\Phi^T\Phi + n\lambda I)^{-1}\Phi^T\vec{y}$.

Solution: Setting the gradient to sero and solving for $\vec{w}$, we have:

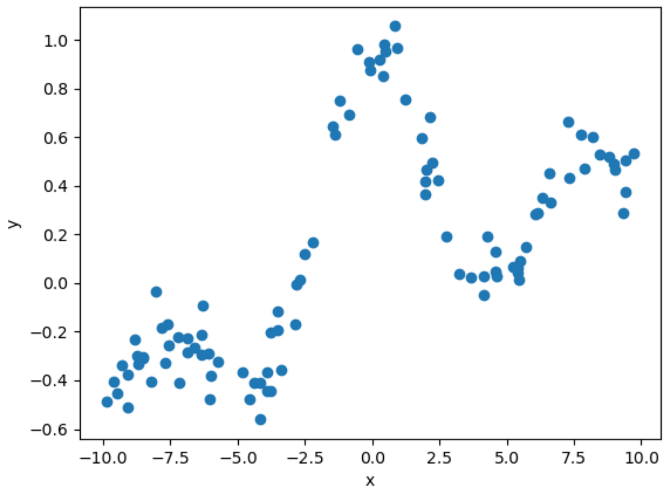
$$\begin{aligned}
&\frac{1}{n}\left(2\Phi^T\Phi\vec{w} - 2\Phi^T\vec{y}\right) + 2\lambda\vec{w} = \vec{0} && \text{(multiply both sides by } n) \\
&\Rightarrow 2\Phi^T\Phi\vec{w} - 2\Phi^T\vec{y} + 2\lambda n\vec{w} = \vec{0} && \text{(grouping terms with } \vec{w}) \\
&\Rightarrow 2\Phi^T\Phi\vec{w} + 2\lambda n\vec{w} = 2\Phi^T\vec{y} && \text{(dividing both sides by 2)} \\
&\Rightarrow \Phi^T\Phi\vec{w} + \lambda n\vec{w} = \Phi^T\vec{y} && \text{(factoring out } \vec{w}) \\
&\Rightarrow (\Phi^T\Phi + n\lambda I)\vec{w} = \Phi^T\vec{y} && \text{(solving for } \vec{w}) \\
&\Rightarrow \vec{w} = (\Phi^T\Phi + n\lambda I)^{-1}\Phi^T\vec{y}
\end{aligned}$$

Problem 2.

The data set linked below contains data for performing non-linear regression. The first column is x (the independent variable), and the second column is y (the dependent variable).

`https://f000.backblazeb2.com/file/jeldridge-data/010-nonlinear_regression/data.csv`

Plotting the data shows that there is a non-linear relationship between x and y:



a) Fit a function of the form $H(\vec{x}) = w_1\phi_1(\vec{x}) + w_2\phi_2(\vec{x}) + \ldots + w_{50}\phi_{50}(\vec{x})$, where each $\phi_i(\vec{x})$ is a Gaussian basis function. Your 50 Gaussian basis functions should be equally-spaced, with the first at $\mu_1 = -10$ and the last at $\mu_{50} = 10$. The width of each Gaussian should be $\sigma = 1$. You should *not* augment $\vec{x}$.

For your answer, report only w_1 (the first component of $\vec{w}$), and show your code.

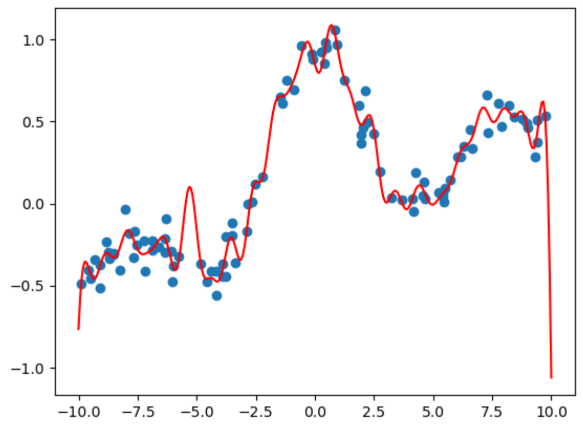
Solution: You should find $w_0 \approx -51$.

The code to find this is shown below:

```
data = np.loadtxt('./data.csv', delimiter=',')
data_x, data_y = data.T

def make_phi(mu, sigma=1):
    def phi(x):
        return np.exp(-(x - mu)**2 / sigma**2)
    return phi

k = 50

phis = [make_phi(mu) for mu in np.linspace(-10, 10, k)]

def phi_transformation(x):
    return np.array([phi(x) for phi in phis])

Phi = np.vstack([phi_transformation(x) for x in data_x])

w_unreg = np.linalg.solve(Phi.T @ Phi, Phi.T @ data_y)
```

b) Plot the prediction function $H(\vec{x})$ that you trained in the previous part, on top of the data. Provide your plot and the code used to generate it.

Solution:

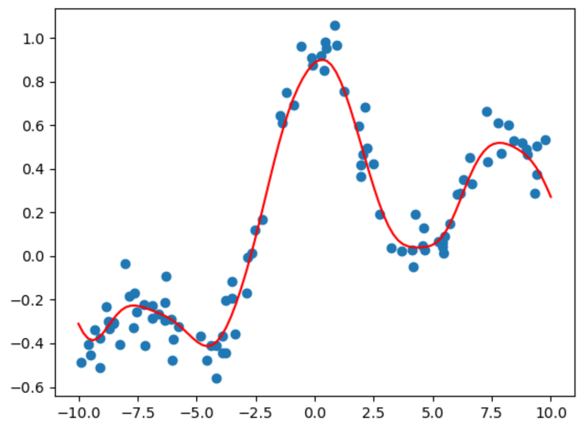


This overfits the data slightly.

c) You should see that the prediction function $H(\vec{x})$ slighly overfits the data. Now perform ridge regression on the same data, using the same Gaussian basis functions. Choose the regularization parameter λ to reduce overfitting (you may do so by trial and error – no need to perform cross-validation). For your answer, state λ and plot your new prediction function on top of the data. Also provide your code.

Solution: By eye, a regularization parameter of $\lambda = 0.02$ seems to work well. Anything around this value is acceptable.

When we plot the new prediction function on top of the data, we see that it generalizes better (overfits less):



The code for training this regularized model is shown below:

```
# add a regularization term to the least squares problem
w_reg_1 = np.linalg.solve(Phi.T @ Phi + n * .02 * np.eye(k), Phi.T @ data_y)

def h_unreg(x):
    return w_unreg @ phi_transformation(x)

def h_reg_1(x):
    return w_reg_1 @ phi_transformation(x)
```