3 Regression

To obtain extra points for the written exam, prepare the solution for **Programming Task 3.3** using the Jupyter notebook provided and upload your work to StudOn. Students may discuss with each other while preparing the solution, but each student must submit their work individually. Refer the Supplements page in StudOn for details regarding the submission deadline.

3.1 Ridge regression models

The ordinary least squares regression cost function with an additional quadratic penalty term is given by:

$$J(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{n}\left(\sum_{j=1}^{m}(w_j x_{ij}) - y_i\right)^2 + \frac{\lambda}{2}\sum_{j=1}^{m} w_j^2,$$

where x_{ij} is the j^{th} feature of an m-dimensional input feature vector for i^{th} data sample, y_i is the target output of i^{th} data sample, n is the number of samples and w_j represents the j^{th} weight element in the weight vector $\mathbf{w}$ of the linear model.

i. Write down the above cost function in matrix notation.

ii. Using the cost function in matrix notation, derive a closed-form solution for $\mathbf{w}$ that minimizes the cost function.

3.2 Programming Task: Gradient Descent

Consider the function

$$J(\mathbf{w}) = J([w_1, w_2]^\top) = -e^{-\frac{1}{100}(w_1^2+w_2^2-w_1w_2-2w_1+4w_2+5)} - 2e^{-(w_1^2+w_2^2-4w_1-9w_2+25)}.$$

i. Plot $J(\mathbf{w})$ as a function of $\mathbf{w}$ using the contour plot using the Matplotlib module.

ii. Obtain the gradient of the above function by hand.
$Hint: \nabla J(\mathbf{w}) = \begin{pmatrix} \frac{\partial J(\mathbf{w})}{\partial w_1} \\ \frac{\partial J(\mathbf{w})}{\partial w_2} \end{pmatrix}$

iii. Implement gradient descent algorithm described in the lecture to find the minimum of this function using the NumPy module. Plot the location of the new $\mathbf{w}$ after each iteration on the contour plot.

3.3 Programming Task: Housing Price Regression Problem

The file `house_prices.txt` contains a data set of house prices: the first column is the house size in square feet, the second column is the number of bedrooms, the third column is the price in USD.

i. Plot house prices vs. house sizes as a scatter plot.

ii. Next, fit the linear regression to these data points. You should not use any other python module besides NumPy to find the weights of the model.

- Consider the linear model $h_{\mathbf{w}}(\mathbf{x}) = w_0 x_0 + w_1 x_1 = \mathbf{w}^\top \mathbf{x}$, where x_1 is the house size in the first column of `house_prices.txt`, $x_0 = 1$ by convention, $\mathbf{x} = [x_0, x_1]^\top$, and $\mathbf{w} = [w_0, w_1]^\top$. Define the cost function on the dataset:

$$J(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^{n} \left(h_{\mathbf{w}}(\mathbf{x}^{(i)}) - y^{(i)} \right)^2 .$$

  Above, n is the number of rows in `house_prices.txt`, $y^{(i)}$ is the house price given in the third column of the file, $\mathbf{x}^{(i)} = [1\ x_1^{(i)}]$ and $x_1^{(i)}$ is the house size from the first column and ith row of the file. Plot $J(\mathbf{w})$ as a function of $\mathbf{w}$ using the contour plot. You may rescale the data if required.

- Start with some initial value $\mathbf{w}$ and run the steps of the gradient descent algorithm (you may reuse revelant parts of the solution from task 3.2). Plot the location of each new $\mathbf{w}$ on the contour plot similar to the previous task.

- Experiment by making changes to the learning rate of the gradient descent algorithm. Observe how the path of the algorithm changes. Make sure that the algorithm converges to the true minimum of the function $J(\mathbf{w})$.

iii. Determine the weights of the model using the closed form solution for $\mathbf{w}$:

$$\hat{\mathbf{w}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}.$$

Above, $\mathbf{y} = [y^{(1)}, \cdots y^{(n)}]^\top$ and $\mathbf{X}$ is the data matrix whose rows are $\mathbf{x}^{(i)}$. Is the the same point that you have found above using gradient descent?