Technische Hochschule Ingolstadt, M-APE, Prof. A. Schiendorfer Lukas Lodes M.Sc.

Intro to AI and Neural Networks (Summer 2023)

Assignment 05

Exercise 1 ***(Linear Regression – Comprehension questions)***

- Can Gradient Descent get stuck in a local minimum when training a Logistic Regression model?
- Do all Gradient Descent algorithms lead to the same model, provided you let them run long enough?
- Suppose you use Batch Gradient Descent and you plot the validation error at every iteration. If you notice that the validation error consistently goes up, what is likely going on? How can you fix this?
- Is it a good idea to stop Mini-batch Gradient Descent immediately when the validation error goes up?
- Suppose you are using Polynomial Regression. You plot the learning curves and you notice that there is a large gap between the training error and the validation error. What is happening? What are three ways to solve this?

Exercise 2 ***(Linear Regression – Calculations)***

Consider the following toy data set:

x_0	x_1	y
1	0.7	5
1	2.5	2.75
1	5.8	-0.8

You will work through a single iteration of gradient descent. You can use all code resources discussed in class or you find helpful online, or you can work out the solutions on paper using a calculator. Assume you want to learn a linear function $f(\mathbf{x}) = w_0 + w_1 \cdot x_1$ that minimizes the mean squared error.

a) Start with an arbitrary set of weights, e.g., $\mathbf{w} = [w_0, w_1] = [0.1, 0.2]$. What's the MSE loss of these parameters?

b) Calculate the gradient of the MSE loss with respect to the parameters $\mathbf{w}$. Compare each individual data point (which could be used for *stochastic gradient descent* with the total gradient over all data points (*batch gradient descent*).

c) Update the parameters according to the gradient descent rule: $\mathbf{w}_{\text{new}} = \mathbf{w} - \alpha \cdot \nabla_{\mathbf{w}} L$ for different learning rate values (e.g., $\alpha \in \{0.1, 0.01, 0.001\}$).

d) Compare the MSE loss values on the updated parameters for different learning rates. Do you see an increase or decrease?

e) (Coding): Now print the first 10 MSE values you would get when repeating the gradient update with the best learning rate from the previous step. Continue for 1000 iterations, what parameter values do you find? Plot the data points and line returned by linear regression.

f) (Math): Suppose you were not only doing pure linear regression but added a penalty term for large weights (so-called *ridge regression*) to counteract overfitting.

$$L(\mathbf{w}; X, y) = \|w\|_2 + \sum_i (\underbrace{(w_0 + w_1 \cdot x^{(i)})}_{\hat{y}^{(i)}} - y^{(i)})^2 = \sqrt{w_0^2 + w_1^2} + \sum_i (\hat{y}^{(i)} - y^{(i)})^2$$

How do the gradient expressions $\frac{\partial L}{\partial w_0}$ and $\frac{\partial L}{\partial w_1}$ change?

Exercise 3 ***(Model Selection for Basis Function Expansions (Coding))***

The file `linear_regression_basis_functions.ipynb`[1] contains a Jupyter notebook that prepares the selection of the most appropriate complexity level for a nonlinear problem. Specifically, you extend a one-dimensional data series x with polynomial base functions $[x^2, x^3, ..., x^k]$. Your task is to determine a suitable polynomial degree k.

a) In the existing code, the entire data set is divided into a training set and a preliminary test set. To compare the performance of different models, you also need a validation set. To do this, divide the provisional test quantity equally into the validation quantity and the final test quantity.

b) Use basic polynomial functions to extend the scalar input x depending on the polynomial degree k so that the resulting new features can be entered into the linear regression model. Then implement a mean squared error (MSE) evaluation of the polynomial models on the training and validation set. Which polynomial degree fits best the given data? What happens if you set the maximum complexity even higher?

c) Create a plot for the so-called *learning curve*. It shows the course of training and validation errors for the different polynomial degrees. Determine the generalization error of the best model (from subtask b)) on the test set and compare it with the validation error. What do you find?

[1] Also available on Colab: `https://colab.research.google.com/github/Alexander-Schiendorfer/Alexander-Schiendorfer.github.io/blob/master/notebooks/thi/linear_regression_basis_functions.ipynb`