Artificial Neural Networks

Prof. Dr. Sen Cheng

Oct 21, 2019

Problem Set 3: Regression

Tutors: Sandhiya Vijayabaskaran (sandhiya.vijayabaskaran@rub.de), Mohammad M. Nejad (m.mohagheghi@ini.rub.de)

1. **Analytical solution of linear regression**

   Although in real-world problems you do not know the function that has generated the dataset, here we assume that we know the generating function known as "ground-truth". You will use the functions here to generate data and later fit a linear regression to the data, so you can see how close the regression is to the ground-truth.

   Let us start with the function

   $$y = 4x + 5 + \varepsilon \tag{1}$$

   as the ground-truth to generate a dataset. ε is a uniform random variable, which generates random numbers between -0.5 and 0.5.

   (a) Generate a vector x containing 100 linearly spaced numbers from -3 to 3 (use `numpy.linspace`) and a vector of 100 numbers for ε (use `numpy.random.uniform`). Now compute vector y according to Eq. 1 and make a scatter plot of y against x.

   (b) Use the solution for linear regression to estimate the parameters m and b from the data. Implement the equations yourself using only elementary operations.

   (c) Use the analytical solution for multiple linear regression to estimate the parameters. Make sure you have designed your X matrix correctly. Implement the equations yourself using only elementary matrix operations.

   (d) Compare the parameters you estimated to the ground-truth. Check how good the estimate is by plotting the the regression line along with the data points. Also compute the explained variance, and generate the residual plot.

   (e) Use the scikit-learn python package to perform linear regression. From `sklearn` import `linear_model` and use the method `LinearRegression`. Using the data you generated in the first step, fit a linear regressor. First look at the instructions provided by `help(linear_model.LinearRegression)`. Calculate the R2-score using `sklearn.metrics.r2_score`.

2. **Polynomial regression**

   Take a stochastic polynomial function

   $$y = -x^5 + 1.5x^3 - 2x^2 + 4x + 3 + \varepsilon, \tag{2}$$

   where $-2 < x < 2$ and ε is a random number uniformly distributed between -1 and 1.

   Follow the steps below:

   (a) Generate 100 data points using the polynomial function

   (b) Fit a 5th-order polynomial with intercept to the dataset. Use the equation for multiple linear regression and a properly designed matrix X to find the coefficients of the polynomial.

(c) Compare the parameters you estimated to the ground-truth. Check how the estimated parameters vary with the number of data points by plotting the error in the estimates versus the number of data points N. Use the l2-norm of the difference between true and estimated parameters as an error measure, and vary N in steps of 5 up to a range of 100.

3. **Incremental version of linear regression**

Often in practice, datasets can be extremely large, which makes it infeasible to use the analytical solution. Use stochastic gradient descent (SGD) to compute the regression parameters as follows :

(a) Program stochastic gradient descent update for linear regression using the l_2-loss function. Again, only use elementary operations. (Hint: don't forget to shuffle the data before iterating through it!)

(b) Test this incremental form on the function you used in exercise 1.

(c) Compare the parameters you estimated to the ground-truth. Plot both the data and predicted values for a visual check. Check how the number of data points affect the estimates by generating a plot as in Problem 2c. Use a constant learning rate of 0.01 for this and use 3 runs of SGD. Next, keep the number of data points constant at 100 and use 3 runs of SGD, and vary the learning rate in steps of 0.001 up to a range of 0.4. Generate a similar plot of error vs. learning rate.