LELEC2870: Practical Sessions

September 28, 2021

Notations and vocabulary

- a scalar: A number represented with an upper-case letter when defining bounds on a series, or a lower-case one otherwise e.g. x in $1..X$
- a vector: A 1-dimensional array that will always be written with a **bold lower-case letter**
- a matrix: A 2-dimensional array that will always be written with a **bold upper-case letter**
- a tensor: A N-dimensional array that will always be written with a **bold upper-case letter**. While it uses the same notation as the matrix, it should always be clear which one is meant.
- an iteration: A time-measure for *iterative* algorithms. It denotes a pass over a part of the training dataset.
- an epoch: A time-measure for *iterative* algorithms. It denotes *one* pass over the complete training dataset. An epoch is composed of at least 1 iteration, but most of the time of multiple iterations especially when large datasets are involved.

1 Session 1 - Linear Regression

Objectives

In this first exercise session, you'll implement linear regression between the target values and one of the features of the dataset. You'll then move on to the multivariate linear regression (linear regression between the target and several input features). Next, you will learn how to build a linear model on a complete dataset i.e. using all features, even thought there is no clear linear relation between the features and the target. Finally, you will implement an iterative method named Stochastic Gradient Descent (SGD) useful for optimizing objective function. We provide you a dataset for the topic covered in this session. This dataset can be found on the Moodle page of this course.

Linear Regression Notations

The output prediction of a linear regressor for an element p can then be written as follows:

$$t_p = w_p^0 + w_1.x_p^1 + w_2.x_p^2 + ... + w_D.x_p^D \tag{1}$$

with D the number of dimensions of input vector $\mathbf{x}$, $w_1...w_D$ the weights of each feature/dimension, and w_0 the independent term (=bias).

A Linear regressor can be equivalently described as a simple neural network with only one layer and a linear activation. The analytical expression of its output is

$$t_p = \sigma(\mathbf{x}_p\mathbf{w} + w_0). \tag{2}$$

where $\mathbf{w}$ is the weights column vector, w_0 is the bias, σ is the activation function and t_p is the output of the network. In order to simplify notations, the bias is generally included in the inputs and weights vectors, i.e. $\mathbf{x}_p$ becomes

$$\begin{pmatrix} 1 & \mathbf{x}_p \end{pmatrix}$$

and $\mathbf{w}$ becomes

$$\begin{pmatrix} w_0 \\ \mathbf{w} \end{pmatrix}.$$

This convention is used in the rest of this exercise session. Since the activation is linear, the output of one-layer neural networks becomes simply

$$t_p = \mathbf{x}_p\mathbf{w}. \tag{3}$$

Notice that now, $\mathbf{x}_p$ and $\mathbf{w}$ are both vectors of length $(D + 1)$.
We can extend this to vectorial outputs (of length T) in the following way

$$\mathbf{t} = \mathbf{X}\mathbf{w}, \tag{4}$$

with the shape of $\mathbf{X}$ being $T \times (D + 1)$

1.1 Feature Selection

Since 10 features are available in the provided dataset, we ask you to select one (or more) of them to perform regression on in the following exercises.

For each feature, **visualize** the relationship between the feature and the target by using `scatterplot` (`matplotlib`). How can you at a glance tell which features will be most useful?

1.2 Univariate Linear Regression (30min)

Pseudo-Inverse method

We can present our dataset of P datapoints with $\mathbf{t}$ the vector of the targets of length N and $\mathbf{X}$ of shape $P \times (D+1)$ (the $+1$ is there to accommodate for the bias). We want to find the vector of weights $\mathbf{w}$ that minimizes the Mean Squared Error (MSE) criterion. Leading to the following equation

$$E = \frac{1}{P}\|\mathbf{t} - \mathbf{X}\mathbf{w}\|^2 \tag{5}$$

$$\Leftrightarrow \left(\frac{\partial E}{\partial \mathbf{w}}\right) = \frac{2}{P}\left(\mathbf{t} - \mathbf{X}\mathbf{w}\right)\mathbf{X} = \mathbf{0} \tag{6}$$

$$\Leftrightarrow \quad \mathbf{0} = \mathbf{X}^T\mathbf{t} - \mathbf{X}^T\mathbf{X}\mathbf{w} \tag{7}$$

$$\Leftrightarrow \mathbf{X}^T\mathbf{X}\mathbf{w} = \mathbf{X}^T\mathbf{t} \tag{8}$$

$$\Leftrightarrow \quad \mathbf{w} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{t}. \tag{9}$$

This is different from the course's definition due to the *transposed* way the matrices are defined.

For this first regression you'll have to **choose** one feature. Once you have selected it (based on your results from previous section), you can perform linear regression. **Implement** linear regression using the pseudo-inverse method `inv` (`numpy.linalg`). **Visualize** the result of your regression. In particular, compare your prediction with the actual target values. Are the results satisfactory ? How could you improve it?

1.3 Bivariate Linear Regression (15 min)

Choose the 2 best features (still using only visual clues) and **train** another linear regression. **Visualize** the results and compare the predicted target values with the actual target values.

RMSE is a formal criterion that can be used to show that linear regression is able to approximate the target values more accurately using two features than only one feature (with respect to training data).

The expression of RMSE is the following:

$$RMSE = \sqrt{\frac{1}{P}\|\mathbf{t} - \mathbf{X}\mathbf{w}\|^2} \tag{10}$$

where $\mathbf{t}$ is the value of the target (ground truth) and $\mathbf{X}\mathbf{w}$ is the prediction (output) of a model.

Implement the computation of RMSE to observe that bivariate regression is better than univariate regression.

1.4 Multivariate Linear Regression (10 min)

Perform linear regressions with a growing number of features and **plot** the accuracy of the models trained over the growing number of features. What can you conclude?

1.5 Train-test Generalization (10 min)

Split the dataset in two equal parts at random ({X1, t1} and {X2,t2}). Build a multivariate regression model on the first sub dataset {X1, t1} and test it on the same {X1,t1} data. Draw a target vs prediction scatterplot and compute the rmse. Is it a good model? Let's see how it generalizes. Test your model on {X2,t2}. Is it as good as the first one? How do you explain the difference between the two? What are the key concepts behind that?

1.6 Stochastic Gradient Descent (20 min)

While the pseudo-inverse method gives the exact solution, it isn't always possible to inverse matrices. It remains however possible to update the parameters or weights in an iterative fashion by using Stochastic Gradient Descent (SGD). Through the computation of the gradient of the weights with respect to the objective function (= Mean Squared Error in our case) for an input $\mathbf{x}$, it is possible to approach the optimal solution. We also refer to this objective function as the Loss ($\mathcal{L}$). During the SGD process the point is to *minimize* this Loss.

Back-propagation for bi-variate Linear Regression

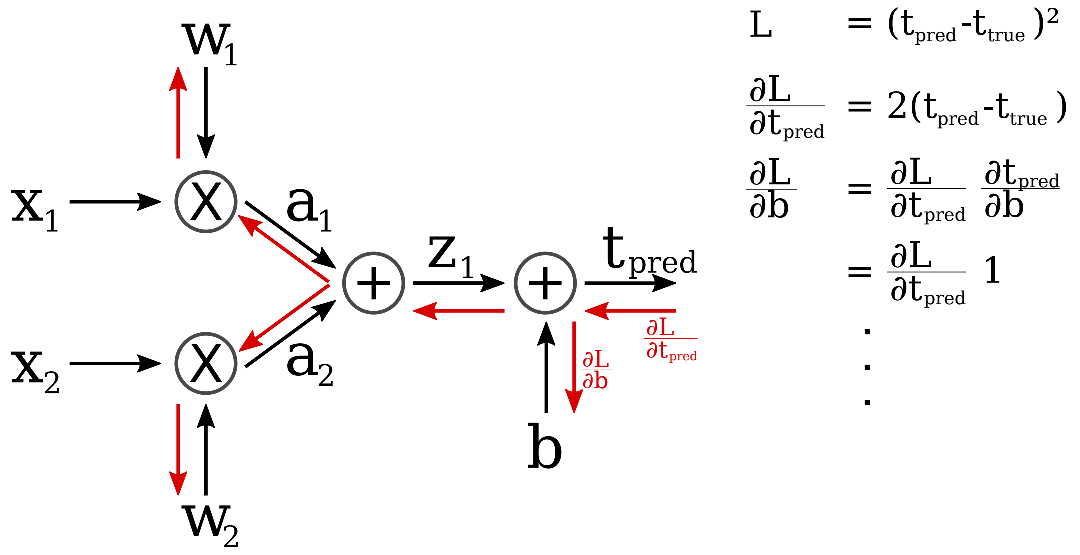

Figure 1: Computational graph of a bi-variate linear regression. The red arrows represent the gradients flowing from the loss to the weights. **Compute the missing gradients**

Transcribing this to equations we obtain:

$$t_{pred} = \mathbf{wx} \tag{11}$$

$$\mathcal{L}(t_{pred}, t_{true}) = (t_{pred} - t_{true})^2 \tag{12}$$

$$\mathbf{w} = \mathbf{w} - \alpha \frac{\partial \mathcal{L}}{\partial \mathbf{w}} \tag{13}$$

$$\iff w_i = w_i - \alpha \frac{\partial \mathcal{L}}{\partial w_i} \tag{14}$$

with t_{true} the true output associated to $\mathbf{x}$. It is now your turn to implement SGD for bi-variate linear regression *with* bias. Refer to Figure 1 to implement the gradient computation and fill in the missing gradients. Does your solution converge to the optimal solution? Try **tuning** the meta-parameters (`n_epochs, lr, lr_annealing`). Does this make the solution better? Which meta-parameter is the most important one? And what happens when you skew the *initial* value of the parameters towards their optimal values?