Programming, Data Analysis and Deep Learning in Python

Niklas Markert
bt709885
1611460

Lukas Thiersch
bt708626
1607110

June 24, 2021

Exercise 29

a)

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning

b)

- Task: Predict continues output variables from input
- Experience: A dataset full of examples (x, y) where y should be the output variable from the input x
- Perforance Measure: The Cost function which sums up all the loss of the predictions to the original values

c)

The difference between test and training data is their use. The trainig data is used to train the model and the test data is used to evaluate how good the model is.

d)

You should differentiate between training and test data, because evaluating a model with the same data as you trained it causes Overfitting.
The test data in the lecture was chosen randomly, in order to avoid having the training data from one value range and the training data from a completely other value range.

e)

For fitting a line to the data (= linear Regression) you need two parameters w and b (for a linear function f(x) = w * x + b). There w represents the slope and b the y-intercept („y-Achsen-Abschnitt“).

f)

It was used a quadratic loss function ($L(y^{(i)}, h(x^{(i)})) = \frac{1}{2}(y^{(i)} - h(x^{(i)})^2$) because:

- The Loss should always be positiv and otherwise, when the loss function wouldn't be quadratic, the loss could be negativ, when a predicted value is higher than the exact value.
- It is easier to bild the later needed derivation from a quadratic function than from an absolute term (|...|), which also would solve the problem described in point one.

g)

The minimization problem of the linear regresseion from the lecture was: $\min_{w,b} J(w,b)$. That means the task was to find the pair of parameters w and b, where J(w, b) gets the smallest.

$J(w,b) = \sum_{i=1}^{n} L(y^{(i)}, h(x^{(i)})$ is the **cost function** for all examples (x, y).

$L(y^{(i)}, h(x^{(i)})) = \frac{1}{2}(y^{(i)} - h(x^{(i)})^2$ is the **loss function** for a single example $(x^{(i)}, y^{(i)})$.

$h(x^{(i)}) = w * x^{(i)} + b$ is the **learned approximation** used for the predictions.

w is the **slope** and b the **y-intercept** of the linear function $h(x^{(i)})$.

h)

The learning rate α is a parameter, for how much the gradients (dw and db) should be applied to the parameters (w and b).
α should be chosen carefully, because if you choose it too small, you need too many seps to come in the near of the searched minimum. But if you choose it to high, it could be that you unintentionally skip the searched minimum.

i)

No you can not expect the cost function J to always go to zero. The only case in which J can go to zero (or close to zero) whould be if all values lies exactly one line, which is then approximated by the linear regression. But in nearly all cases you have you don't this, maybe you have values which lies around a line, which you can approximate, but even then the cost function can't be zero, because for the most of the point the loss function, maybe is small, but still is not zero.

j)

You can't always say on which data the error is lower. In the most cases probably the error is lower on the training data, because the linear regression was done on this data and was bild that there the error is as low as possible. But there could also be cases where the test set has simply the better data for the approximated regression.

k)

$$\frac{\delta}{\delta w_0} J(w) = ... = \sum_{i=1}^{m} (h_w(x^{(i)}) - y^{(i)}) * \frac{\delta}{\delta w_0} (h_w(x^{(i)}) - y^{(i)}) = \sum_{i=1}^{m} (h_w(x^{(i)}) - y^{(i)}) * 1 \tag{1}$$

$$\frac{\delta}{\delta w_1} J(w) = ... = \sum_{i=1}^{m} (h_w(x^{(i)}) - y^{(i)}) * \frac{\delta}{\delta w_1} (h_w(x^{(i)}) - y^{(i)}) = \sum_{i=1}^{m} (h_w(x^{(i)}) - y^{(i)}) * x^{(i)} \tag{2}$$

$$\frac{\delta}{\delta w_2} J(w) = ... = \sum_{i=1}^{m} (h_w(x^{(i)}) - y^{(i)}) * \frac{\delta}{\delta w_2} (h_w(x^{(i)}) - y^{(i)}) = \sum_{i=1}^{m} (h_w(x^{(i)}) - y^{(i)}) * x^{(i)^2} \tag{3}$$