Machine Learning
(IT3190E)

Quang Nhat NGUYEN

quang.nguyennhat@hust.edu.vn

Hanoi University of Science and Technology

School of Information and Communication Technology

Academic year 2020-2021

The course's content:

- Introduction
- Performance evaluation of ML system
- **Supervised learning**
  - **Regression problem**
  - **Linear regression**
- Unsupervised learning
- Ensemble learning
- Reinforcement learning

Supervised vs. Unsupervised learning

- **Supervised** learning
  - The training set is a set of examples, each associated **with a class/output value**
  - The goal is to learn (approximate) a hypothesis (e.g., a classification function, or a regression function) that fits the given **labelled** training set
  - The learned hypothesis will then be used to classify/predict future (unseen) examples
- **Unsupervised** learning
  - The training set is a set of instances **with no class/output value**
  - The goal is to find some intrinsic groups/structures/relations

Regression problem

- Regression problem belongs to supervised learning
- The goal of the regression problem is to predict a vector of continuous (i.e., real) values

$$f: X \rightarrow Y$$

where Y is a vector of real values

Regression problem: Performance evaluation

- The system's output is a numeric value
- Error (i.e., loss) function
  - MAE (mean absolute error):
    $$MAE-Error(x)=\frac{\sum_{i=1}^{n}|d(x)-o(x)|}{n}$$
  - RMSE (root mean squared error):
    $$RMSE-Error(x)=\sqrt{\frac{\sum_{i=1}^{n}\left(d(x)-o(x)\right)^2}{n}}$$
  - The overall error on the entire test set :
    $$Error=\frac{1}{|D\_test|}\sum_{x\in D\_test}Error(x);$$
    - n: The number of outputs
    - $o(x)$: Vector of the outputs predicted by the system for example x
    - $d(x)$: Vector of the true (i.e., desired) outputs for example x
- Accuracy can be defined as an inverse function of the error function

Linear regression – Introduction

- Given an input example, to predict a real-valued output
- A simple-but-effective machine learning method <u>appropriate when the target function (to be learned) is a linear function</u>

$$f(x) = w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_n x_n = w_0 + \sum_{i=1}^{n} w_i x_i \qquad (w_i, x_i \in \mathbb{R})$$

- To learn (approximately) a target function `f`

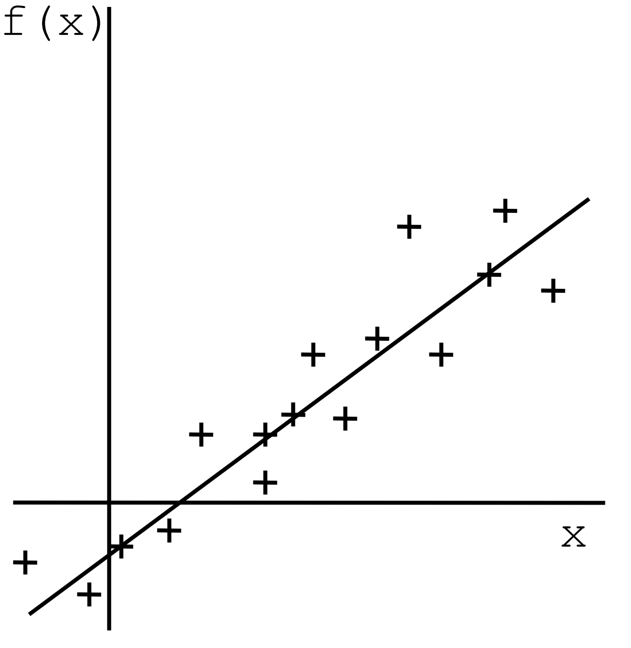
$$f: \mathbf{X} \rightarrow Y$$

  - X: Input space (n-dimensional vector space – $\mathbb{R}^n$)
  - Y: Output space (real-valued space – $\mathbb{R}$)
  - f: The function to be learned (a linear mapping function)
- In fact, to learn a vector of weights: $w = (w_0, w_1, w_2, \ldots, w_n)$

Linear regression – Example

Which linear function `f(x)` is appropriate?

x	f(x)
0.13	-0.91
1.02	-0.17
3.17	1.61
-2.76	-3.31
1.44	0.18
5.28	3.36
-1.74	-2.46
7.93	5.56
...	...



For example: `f(x) = -1.02 + 0.83x`

Training/test examples

- For each training example $x=(x_1,x_2,...,x_n)$, where $x_i \in R$
  - Expected output value: c_x ($\in R$)
  - Real output value (i.e., produced by the system): $y_x = w_0 + \sum_{i=1}^{n} w_i x_i$

  → w_i is the system's current estimation for the weight of the *i*-th attribute

  → The real output value y_x is expected to (approximately) be c_x

- For each test example $z=(z_1,z_2,...,z_n)$
  - To predict (i.e., compute) its output value
  - By applying the learned target function f

Error function

- The linear regression learning algorithm needs to define an error (i.e., loss) function

  → To evaluate the degree of error made by the system in the training phase

- Definition of the error function `E`
  - The system's error for each training example `x`:

$$E(x) = \frac{1}{2}(c_x - y_x)^2 = \frac{1}{2}\left(c_x - w_0 - \sum_{i=1}^{n} w_i x_i\right)^2$$

  - The system's error for the entire training set `D`:

$$E = \sum_{x \in D} E(x) = \frac{1}{2}\sum_{x \in D}(c_x - y_x)^2 = \frac{1}{2}\sum_{x \in D}\left(c_x - w_0 - \sum_{i=1}^{n} w_i x_i\right)^2$$

Linear regression – Algorithm

- The learning of the target function `f` is equivalent to the learning of the weights vector **`w`** to minimize the training error `E`

  → This method is called "*Least-Square Linear Regression*"

- The training phase
  - Initialize the weights vector **`w`**
  - Compute the training error `E`
  - Update the weights vector **`w`** according to **the delta rule**
  - Repeat, until converging to a (locally) small error value `E`

- The prediction phase

  For a new example `z`, its output value is computed by:

$$f(z) = w^*_0 + \sum_{i=1}^{n} w^*_i \, z_i$$

where $w^* = (w^*_0, w^*_1, \ldots, w^*_n)$ is the learned weights vector

Delta rule

- To update the weights vector **w** towards the direction of decreasing the training error E
  - η is the learning rate (a positive constant)
    - → Define the degree of changing the weight values at a learning step
  - Instance-to-instance/incremental update: $w_i \leftarrow w_i + \eta(c_x - y_x)x_i$
  - Batch update: $w_i \leftarrow w_i + \eta \sum_{x \in D} (c_x - y_x) x_i$
- Alternative names of the delta rule:
  - LMS (least mean square) rule
  - Adaline rule
  - Widrow-Hoff rule

Batch vs. incremental update

- Batch update
  - At each learning step, the weights are updated *after **all** the training examples of the batch are introduced to (learned by) the system*
    - The error is computed accumulatively for all the training examples
    - The weights are updated according to the entirely accumulated error
- Instance-to-instance/ incremental update
  - At each learning step, the weights are updated *immediately after **each** training example is introduced to (learned by) the system*
    - An individual error is computed for the introduced training example
    - The weights are updated immediately according to this error

LSLR_batch(D, η)

```
for each attribute f_i
    w_i ← a (small) randomly initialized value
while not CONVERGENCE
    for each attribute f_i
        delta_w_i ← 0
    for each training example x∈D
        Compute the real output value y_x
        for each attribute f_i
            delta_w_i ← delta_w_i + η(c_x-y_x)x_i
    for each attribute f_i
        w_i ← w_i + delta_w_i
end while
return w
```

LSLR_incremental(D, η)

for each attribute f_i

    $w_i \leftarrow$ a (small) randomly initialized value

while not CONVERGENCE

    for each training example $x \in D$

        Compute the real output value y_x

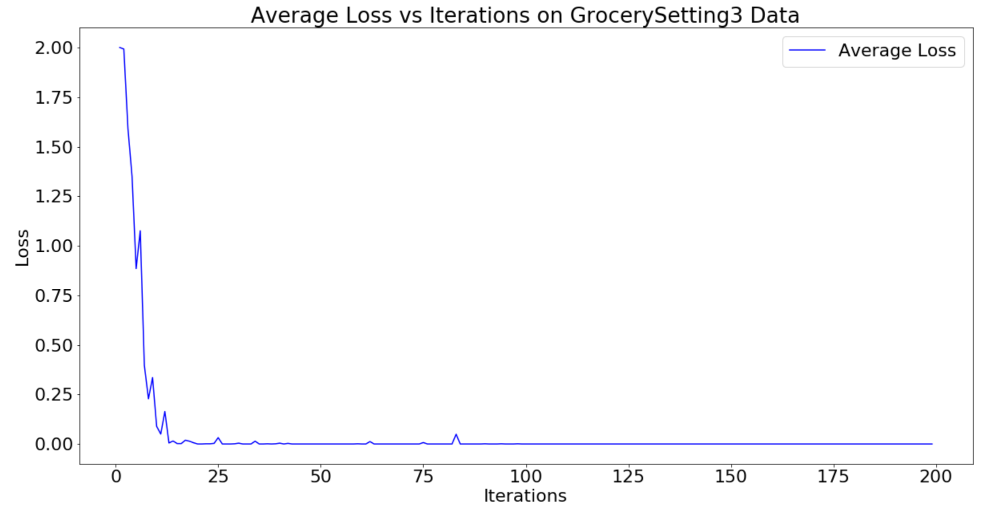
        for each attribute f_i

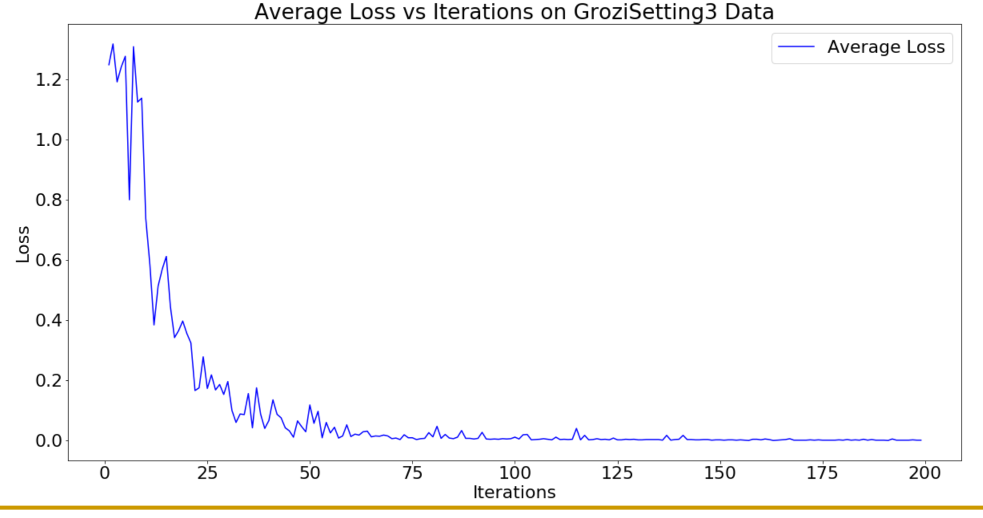
            $w_i \leftarrow w_i + \eta(c_x - y_x)x_i$

end while

return w

Stop conditions for the learning process





Stop conditions for the learning process

- In the learning algorithms `LSLR_batch` and `LSLR_incremental`, the learning process stops when the conditions defined by `CONVERGENCE` are satisified
- The learning stop conditions are often defined based on some system performance evaluation criteria
  - Stop, if the error is less than a threshold
  - Stop, if the error at a learning step (iteration) is higher than the error at the previous learning step
  - Stop, if the difference of the errors between two consecutive learning steps is less than a threshold
  - ...