**Introduction of Factorization Machine (FM)**

FM was first proposed by [Steffen Rendle](https://scholar.google.co.jp/citations?user=yR-ugIoAAAAJ&hl=ja) in [this paper](https://www.ismll.uni-hildesheim.de/pub/pdfs/Rendle2010FM.pdf).

FM is good because it can

* capture interactions between features efficiently, especially in sparse data.
* compute the interactions in linear time complexity

The core of FM is the formula:

* + - * : bias
      * , i = 1..n: weight of feature
      * k-dimensional vector of feature . denotes dot product between . may be referred a latent vector.

FM is formed by a linear model part (), and the interaction part between features (). The essence of FM is in the interaction part, where the weights of each interaction is estimated using the inner product of the two latent vectors.

We will examine FM by the following example.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | site\_id (si) | site\_domain (sd) | site\_cat (sc) | Click |
|  | 1 | 4 | 7 | 1 |
|  | 1 | 4 | 8 | 1 |
|  | 2 | 5 | 9 | 0 |
|  | 2 | 6 | 7 | 0 |

Given the above data. The target is to predict whether or not a “click” event happens based on information of “site\_id”, “site\_domain”, “site\_cat”.

From the data, we notice that some interactions between features may lead to Click event, e.g Si = 1, and Sd = 4, then the rate of Click may be high. How FM captures such interactions?

1. TRAINING PROCESS
2. First, all features are one-hot encoded

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Si\_1 | Si\_2 | Sd\_4 | Sd\_5 | Sd\_6 | Sc\_7 | Sc\_8 | Sc\_9 | click |
|  | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |
|  | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 |
|  | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
|  | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |

1. Each feature now is considered as a vector with k dimensions. k is a parameter that users need to define in advance. Suppose k = 3

|  |  |  |
| --- | --- | --- |
| Si\_1 | 🡪 |  |
| Si\_2 | 🡪 |  |
| Sd\_4 | 🡪 |  |
| Sd\_5 | 🡪 |  |
| Sd\_6 | 🡪 |  |
| Sc\_7 | 🡪 |  |
| Sc\_8 | 🡪 |  |
| Sc\_9 | 🡪 |  |

1. The training process starts with the above formula:

|  |  |  |
| --- | --- | --- |
| =(1, 0, 1, 0, 0, 1, 0, 0) | 🡪 |  |
| =(1, 0, 1, 0, 0, 0, 1, 0) | 🡪 |  |
| =(0, 1, 0, 1, 0, 0, 1, 0) | 🡪 |  |
| =(0, 1, 0, 0, 1, 1, 0, 0) | 🡪 |  |

* **, ,**  are hyper-parameters that we need to learn
* These parameters are randomly initiated, then to be learnt through [SGD](https://ja.wikipedia.org/wiki/%E7%A2%BA%E7%8E%87%E7%9A%84%E5%8B%BE%E9%85%8D%E9%99%8D%E4%B8%8B%E6%B3%95)

1. VALIDATION PROCESS

Suppose we need to make prediction for the following new events

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Si\_1** | **Si\_2** | **Sd\_4** | **Sd\_5** | **Sd\_6** | **Sc\_7** | **Sc\_8** | **Sc\_9** | **click** |
|  | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | ? |
|  | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | ? |

Based on the parameter we already learned during the training process, we can predict new events as following:

|  |  |  |
| --- | --- | --- |
| =(1, 0, 1, 0, 0, 0, 1, 0) | 🡪 |  |
| =(1, 0, 0, 0, 1, 0, 1, 0) | 🡪 |  |