Factorization Machines

DSTA

1 Factorization Machines

1.1 Genesis

Invented by Steffen Rendle, now Google Research:

- 2010 IEEE International Conference on Data Mining
- ACM Transactions on Intelligent Systems and Technology (TIST) 3 (3), 57

1.2 Problem statement

Instance:

- a collection (dataset) $\mathbf{D}$ of m numerical datapoints (points in $\mathbb{R}^n$)
- a classification system $C = \{c_1, c_2, \dots c_k\}$

. . .

Solution: classification function $\gamma : \mathbf{X} \to C$

Measure: misclassification

. . .

[PF] "classification predicts whether something will happen, whereas regr. predicts how much something will happen."

1.3 Supervised version

Fig. 1. Example (from Rendle [2010]) for representing a recommender problem with real valued feature vectors $\mathbf{x}$. Every row represents a feature vector $\mathbf{x}_i$ with its corresponding target y_i. For easier interpretation, the features are grouped into indicators for the active user (blue), active item (red), other movies rated by the same user (orange), the time in months (green), and the last movie rated (brown).

. . .

Estimate the rating for the *new* user/film combination $\mathbf{x_8}$: most cells are 0, y_8 is unknown.

. . .

We face *sparsity.*

$\mathbf{D} = \{(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \dots\}$

Find rating estimate function $Y : \mathbb{R}^n \to T$ s.t.

. . .

- $T = \mathbb{R}$ for regression,
- $T = \{+, -\}$ for classification.

$\hat{\mathbf{D}} = \{(\mathbf{x}^{(m+1)}, Y(y^{(m+1)}), (\mathbf{x}^{(m+2)}, Y(y^{(m+2)}), \dots\}$

Note: Rendle uses different letters; here n=*dimensions(*$\mathbf{D}$*)*)

1.4 For reference: the constraints scenario

$\mathbf{D} = \{\mathbf{x}^{(a)}, \mathbf{x}^{(b)} \dots\}$

re-arrange the rows so that $\mathbf{x}^{(a)}$ *maps* higher than $\mathbf{x}^{(b)}$ and so on.

Ideal for Top-k searchs and recommendations

2 The Model

2.1 Intuition

extend linear regression to capture *synergetic* effects between variables:

introduce a minimal quadratic effect $x_i x_j$

fill the table by looking at values on the same row or column of the target cell

2.2 General estimation

$$\hat{y}(\mathbf{x}) := w_0 + \sum_{i=1}^{n} w_i x_i$$

an initial (fixed) bias + linear regression.

To look at quadratic interactions, fix $d = 2$:

$$\hat{y}(\mathbf{x}) := w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} w_{ij} x_i x_j$$

. . .

- lots of training to find out all n^2 coefficients w_{ij}
- the w_{ij}'s may not even be significant (too close to 0)
- computing even a single prediction costs $\Theta(n^2)$

3 A simpler model

3.1 In practice

1. fix d=2 and a small integer k (e.g., # of *genres*)
2. build a model of how the n dimensions relate to the k *genres:* a $V_{n \times k}$ matrix

. . .

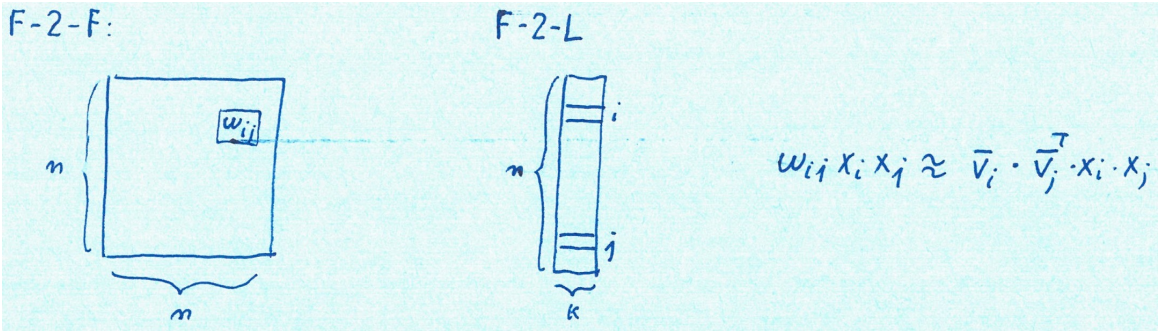
$$W = V \cdot V^T \ \Rightarrow w_{ij} = \mathbf{v}_i^T \cdot \mathbf{v}_j = < \mathbf{v}_i, \mathbf{v}_j >$$

Key point: W contains $\frac{n^2}{2} - \frac{n}{2}$ estimates while the equivaleny V only has $n \cdot k$ (latent) estimates.

$$\hat{y}(\mathbf{x}) := w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} < \mathbf{v}_i, \mathbf{v}_j > x_i x_j$$

Where the inner/dot product is

$$< \mathbf{v}_i, \mathbf{v}_j >= \mathbf{v}_i^T \cdot \mathbf{v}_j = \sum_{f=1}^{k} v_{if} v_{jf}$$



3.2 [Rendle, 2010]

$\hat{w}_{i,j} :=< \mathbf{v}_i, \mathbf{v}_j >$ *models the interaction between the i-th and j-th variable.*

Instead of using an own model parameter $w_{i,j} \in \mathbb{R}$ *for each interaction, the FM models the interaction by factorizing it.*

We will see later on, that this is the key point which allows high quality parameter estimates of higher-order interactions ($d \geq 2$) *under sparsity.*

4 Computational costs

4.1 Th: cost is linear in *n*

$$\hat{y}(\mathbf{x}) := w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} < \mathbf{v}_i, \mathbf{v}_j > x_i x_j$$

where

$$< \mathbf{v}_i, \mathbf{v}_j >= \mathbf{v}_i^T \cdot \mathbf{v}_j = \sum_{f=1}^{k} v_{if} v_{jf}$$

How can this be computed in $\Theta(kn) = \Theta(n)$ iteration?

$$\sum_{i=1}^{n} \sum_{j=i+1}^{n} < \mathbf{v}_i, \mathbf{v}_j > x_i x_j$$

. . .

Insight: i and j never appear *together:* their iteration can be separated.

. . .

Idea: iterate over k outside, push i and j iterations inside.

5 Implementations

5.1 The LibFM source

libfm.org is the repository for the 'official' C++ implementation of FMs, which ended in 2014.

5.2 FMs in Python

PyFM

provides a new environment for running FMs within Python.

```
pip install git+https://github.com/coreylynch/pyFM
```

. . .

```
# Build and train a Factorization Machine
myfm = pylibfm.FM(num_factors=10,
                  num_iter=100,
                  task="regression",
                  ...)

myfm.fit(X_train,y_train)
...
```

5.3 1-Hot econding

```
from pyfm import pylibfm
from sklearn.feature_extraction import DictVectorizer
import numpy as np

train = [
    {"user": "1", "item": "5", "age": 19},
    {"user": "2", "item": "43", "age": 33},
    ...
]
```

four users, four items: 8 columns

. . .

```
v = DictVectorizer()
X = v.fit_transform(train)
print(X.toarray())
[[ 19.   0.   0.   0.   1.   1.   0.   0.   0.]
 [ 33.   0.   0.   1.   0.   0.   1.   0.   0.]
 ...
]
```

What is the estimated appreciation of user 1, aged 24 now, for item 10 once he or she buys it?

```
y = np.repeat(1.0, X.shape[0])

fm = pylibfm.FM()

fm.fit(X, y)

fm.predict(v.transform({"user": "1", "item": "10", "age": 24}))
```

6 Coda: The time-efficency of MF

6.1 Computational aspects

Rendle proved that an intrisically quadratic activity: compute all possible second-order, $x_i \cdot x_j$, effects, can be done in a time linear and not quadratic in n.

$\hat{y}(\mathbf{x}) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} w_{ij} x_i x_j$

To do so, Rendle models feature interactions by learning k latent factors:

$\langle \mathbf{v}_i, \mathbf{v}_j \rangle = \sum_{f=1}^{k} v_{i,f} v_{j,f}$

6.2 Optimisation

While computing the mathematical formula for polynomial regression takes $\Theta(n^2)$ ops., Rendle does it in $\Theta(kn)$.

Notice how summing over different pairs is equivalent to summing over all pairs minus the self-interactions (divided by 2):

a correction factor $\frac{1}{2}$ is introduced from the beginning of the derivation.

$\sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle \mathbf{v}_i, \mathbf{v}_j \rangle x_i x_j$

. . .

$= \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \langle \mathbf{v}_i, \mathbf{v}_j \rangle x_i x_j - \frac{1}{2} \sum_{i=1}^{n} \langle \mathbf{v}_i, \mathbf{v}_i \rangle x_i x_i$

6.3 Steps

$\sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle \mathbf{v}_i, \mathbf{v}_j \rangle x_i x_j$

$= \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \langle \mathbf{v}_i, \mathbf{v}_j \rangle x_i x_j - \frac{1}{2} \sum_{i=1}^{n} \langle \mathbf{v}_i, \mathbf{v}_i \rangle x_i x_i$

. . .

$= \frac{1}{2} \left(\sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{f=1}^{k} v_{i,f} v_{j,f} x_i x_j \right) - \frac{1}{2} \left(\sum_{i=1}^{n} \sum_{f=1}^{k} v_{i,f} v_{i,f} x_i x_i \right)$

. . .

$= \frac{1}{2} \left(\sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{f=1}^{k} v_{i,f} v_{j,f} x_i x_j - \sum_{i=1}^{n} \sum_{f=1}^{k} v_{i,f} v_{i,f} x_i x_i \right)$

$= \frac{1}{2} \left(\sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{f=1}^{k} v_{i,f} v_{j,f} x_i x_j - \sum_{i=1}^{n} \sum_{f=1}^{k} v_{i,f} v_{i,f} x_i x_i \right)$

. . .

$= \frac{1}{2} \sum_{f=1}^{k} \left((\sum_{i=1}^{n} v_{i,f} x_i) \left(\sum_{j=1}^{n} v_{j,f} x_j \right) - \sum_{i=1}^{n} v_{i,f}^2 x_i^2 \right)$

. . .

$= \frac{1}{2} \sum_{f=1}^{k} \left((\sum_{i}^{n} v_{i,f} x_i)^2 - \sum_{i=1}^{n} v_{i,f}^2 x_i^2 \right)$

Now the summations in i are inside the k summation but separated from each other.

. . .

Substituting back into the factorization machine formula:

$\hat{y}(\mathbf{x}) = w_0 + \sum_{i=1}^{n} w_i x_i + \frac{1}{2} \sum_{f=1}^{k} \left((\sum_{i}^{n} v_{i,f} x_i)^2 - \sum_{i=1}^{n} v_{i,f}^2 x_i^2 \right)$