1）可用于高度稀疏数据场景；2）具有线性的计算复杂度

对于 categorical(类别)类型特征，需要经过 One-Hot Encoding 转换成数值型特征。CTR/CVR 预测时，用户的性别、职业、教育水平、品类偏好，商品的品类等，经过 One-Hot 编码转换后都会导致样本数据的稀疏性。特别是商品品类这种类型的特征，如商品的末级品类约有 550 个，采用 One-Hot 编码生成 550 个数值特征，但每个样本的这 550 个特征，有且仅有一个是有效的（非零）。由此可见，经过 One-Hot 编码之后，大部分样本数据特征是比较稀疏的(即特定样本的特征向量很多维度为 0)，同时导致特征空间大~~。(对于每一个特征，如果它有 m 个可能值，那么经过独热编码后，就变成了 m 个二元特征(取值 0 或 1)。并且，这些特征互斥，每次只有一个激活。因此，数据会变成稀疏的。)~~ sklearn 中 preprocessing.OneHotEncoder 实现该编码方法。

通过观察大量的样本数据可以发现，某些特征经过关联之后，与 label 之间的相关性就会提高。例如，"USA"与"Thanksgiving"、"China"与"Chinese New Year"这样的关联特征，对用户的点击有着正向的影响。换句话说，来自"China"的用户很可能会在"Chinese New Year"有大量的浏览、购买行为，而在"Thanksgiving"却不会有特别的消费行为。这种关联特征与 label 的正向相关性在实际问题中是普遍存在的，如"化妆品"类商品与"女"性，"球类运动配件"的商品与"男"性，"电影票"的商品与"电影"品类偏好等。因此，引入两个特征的组合是非常有意义的。(我的理解：个性化特征)

一般的线性模型为：

$$y = \omega_0 + \sum_{i-1}^{n} \omega_i x_i$$

从上面的式子很容易看出，一般的线性模型压根没有考虑特征间的关联(组合)。为了表述特征间的相关性，我们采用多项式模型。在多项式模型中，特征 x_i 与 x_j 的组合用 $x_i x_j$ 表示。为了简单起见，我们讨论二阶多项式模型。具体的模型表达式如下：

$$y = \omega_0 + \sum_{i-1}^{n} \omega_i x_i + \sum_{i-1}^{n-1} \sum_{j-i+1}^{n} \omega_{ij} x_i x_j$$

上式中，n 表示样本的特征数量，x_i 表示第 i 个特征。

与线性模型相比，FM(Factorization Machine)的模型就多了后面特征组合的部分。

从公式(1)可以看出，组合特征的参数一共有 $n(n-1)/2$ 个，任意两个参数都是独立的。然而，在数据稀疏性普遍存在的实际应用场景中，二次项参数的训练是很困难的。其原因是，每个参数 w_{ij} 的训练需要大量 x_i 和 x_j 都非零的样本；由于样本数据本来就比较稀疏，满足"x_i 和 x_j 都非零"的样本将会非常少。训练样本的不足，很容易导致参数 w_{ij} 不准确，最终将严重影响模型的性能。

如何解决二次项参数的训练问题呢？**矩阵分解**提供了一种解决思路。在 model-based 的协同过滤中，一个 rating 矩阵可以分解为 user 矩阵和 item 矩阵，每个 user 和 item 都可以采用一个隐向量表示。我们把每个 user 表示成一个二维向量，同时把每个 item 表示成一个二维向量，两个向量的点积就是矩阵中 user 对 item 的打分。

类似地，所有二次项参数 $W_{i,j}$ 可以组成一个对称阵 W,那么这个矩阵就可以分解为 $W=VV^T$,V 的第 i 行便是第 i 维特征的隐向量。换句话说，每个参数 $W_{i,j} = <V_i,V_j>$.
V_i 表示 X_i 的隐向量，V_j 表示 X_j 的隐向量
为了求出 $W_{i,j}$，我们对每一个特征分量 X_i 引入辅助向量

$$V_i = (v_{i1}, v_{i2}, \ldots, v_{ik})$$

然后，利用 $V_i V_j^T$ 对 W_{ij} 进行求解。对辅助向量的维度 k 值的限定，反映了 FM 模型的表达能力。

$$\mathbf{V} = \begin{pmatrix} v_{11} & v_{12} & \cdots & v_{1k} \\ v_{21} & v_{22} & \cdots & v_{2k} \\ \vdots & \vdots & & \vdots \\ v_{n1} & v_{n2} & \cdots & v_{nk} \end{pmatrix}_{n \times k} = \begin{pmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \vdots \\ \mathbf{v}_n \end{pmatrix}$$

那么 ωij 组成的矩阵可以表示为:

$$\hat{\mathbf{W}} = \mathbf{V}\mathbf{V}^T = \begin{pmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \vdots \\ \mathbf{v}_n \end{pmatrix} \begin{pmatrix} \mathbf{v}_1^T & \mathbf{v}_2^T & \cdots & \mathbf{v}_n^T \end{pmatrix}$$

则 FM 的模型方程为：

则二次项的参数数量减少为 kn 个，远少于多项式模型的参数数量.我觉得上式应该是 $w_{i,j} = <v_i,v^T_j>$,但是上面的写法才是对的，因为是点乘，两向量得是相同维度。还有 i 的取值为 1 到 n-1,j 的取值是 i+1 到 n，因为特征不可能自己和自己组合

FM 算法的求解过程：

Lemma 3.1: The model equation of a factorization machine (eq. (1)) can be computed in linear time $O(kn)$.

我的理解：第一步是一个矩阵(矩阵中所有元素求和)减去对角线部分，然后除以 2。多项式部分的计算复杂度是 **O(kn).即 FM 可以在线性时间对新样本作出预测**

于是, FM 的最优化问题就变成了

$$\Theta^* = \arg\min_{\Theta} \sum_{i=1}^{N} loss(\hat{y}(\mathbf{x}^{(i)}), y^{(i)}),$$

当然, 我们通常还考虑 L2 正则, 因此, 最优化问题就变成

$$\Theta^* = \arg\min_{\Theta} \sum_{i=1}^{N} \left(loss(\hat{y}(\mathbf{x}^{(i)}), y^{(i)}) + \sum_{\theta \in \Theta} \lambda_\theta \theta^2 \right),$$

其中, λ_θ 表示参数 θ 的正则化系数.

ALGORITHM 1: Stochastic Gradient Descent (SGD)

Input: Training data S, regularization parameters λ, learning rate η, initialization σ
Output: Model parameters $\Theta = (w_0, \mathbf{w}, \mathbf{V})$
$w_0 \leftarrow 0$; $\mathbf{w} \leftarrow (0, \ldots, 0)$; $\mathbf{V} \sim \mathcal{N}(0, \sigma)$;
repeat
  for $(\mathbf{x}, y) \in S$ **do**
    $w_0 \leftarrow w_0 - \eta \left(\frac{\partial}{\partial w_0} l(\hat{y}(\mathbf{x}|\Theta), y) + 2\,\lambda^0\, w_0 \right)$;
    for $i \in \{1, \ldots, p\} \wedge x_i \neq 0$ **do**
      $w_i \leftarrow w_i - \eta \left(\frac{\partial}{\partial w_i} l(\hat{y}(\mathbf{x}|\Theta), y) + 2\,\lambda^w_{\pi(i)}\, w_i \right)$;
      for $f \in \{1, \ldots, k\}$ **do**
        $v_{i,f} \leftarrow v_{i,f} - \eta \left(\frac{\partial}{\partial v_{i,f}} l(\hat{y}(\mathbf{x}|\Theta), y) + 2\,\lambda^v_{f,\pi(i)}\, v_{i,f} \right)$;
      end
    end
  end
until *stopping criterion is met*;

回归问题：最小均方误差(the least square error) ~~均方(一组数的平方的平均值)~~

$$loss^R(\hat{y}, y) = \frac{1}{2} \sum_{i=1}^{m} \left(\hat{y}^{(i)} - y^{(i)} \right)^2$$

二分类问题：对数损失函数，其中σ表示的是阶跃函数 Sigmoid $\sigma(x) = \frac{1}{1+e^{-x}}$

对数损失是用于最大似然估计的，一组参数在一堆数据下的似然值，等于每一条数据的概率之积，而损失函数一般是每条数据的损失之和，为了把积变为和(我的理解：方便计算)，就取了对数。再加个负号是为了让最大似然值和最小损失对应起来(本来求和最大时对应的参数，加上负号后，求和最小时对应的参数，则等价于求最小损失)。

$$loss^C(\hat{y}, y) = \sum_{i=1}^{m} -ln\sigma\left(\hat{y}^{(i)} y^{(i)}\right)$$ 这个就是标准形式的对数损失函数，将 sigmoid 函数带入，符号抵消，即为 log(1+exp(-yf(x)))

对于回归问题：~~可以理解为 SGD,单样本训练~~

对于二分类问题：

<=这个结果很重要，要能够推导(可以将求和符号化为+,然后容易理解)

在使用 SGD 训练模型时，在每次迭代中，只需计算一次所有 f 的 $\sum_{j=1}^{n} v_{j,f} x_j$，就能够方便得到所有 $V_{i,f}$ 的梯度，(上述偏导结果求和公式中没有 i,即与 i 无关，只与 f 有关)显然计算所有 f 的 $\sum_{j=1}^{n} v_{j,f} x_j$ 的复杂度是 O(kn),模型参数一共有 nk + n + 1 个。因此，FM 参数训练的复杂度也是 O(kn).综上可知，**FM 可以在线性时间训练和预测**，是一种非常高效的模型。

1. **学习率** η: SGD 的收敛依赖于 η 的取值, η 太大, 则可能不收敛; η 太小, 则收敛会很慢. 因此这个参数要取得合适.
2. **正则化系数** λ: FM 模型的泛化能力 (相应的预测质量) 很大程度上依赖于这些正则化系数的选取. 它们通常是在单独的 holdout set 上通过 grid search 方法来获取, 由于它们个数多, 取值区间大, 因此, 通过 grid search 方法来选取会非常费时. 一种提高效率的做法是减少它们的个数. 例如, 可以考虑让矩阵 V 中行号经 π 映射后为同一个值的那些行使用同一个正则化系数. 此时, 正则化系数集 λ 变为

$$\lambda^0, \quad \lambda^w_{\pi(i)}, \quad \lambda^v_{\pi(i)}, \quad i \in \{1, 2, \cdots, n\}.$$

   文 [3] 提出了一种自适应选取这些正则化系数的方法.
3. **正态分布方差参数** σ: 矩阵 V 的初始化采用符合正态分布 $\mathcal{N}(0, \sigma)$ 的随机初始化, 方差参数 σ 通常取得很小.

我的理解：**正则化系数用于衡量正则项与损失项的比重**

总结：FM 是一种比较灵活的模型，通过合适的特征变换方式，FM 可以模拟二阶多项式核的 SVM 模型、MF 模型、SVD++模型等。相比 SVM 的二阶多项式核而言，FM 在样本稀疏的情况下是有优势的；而且，FM 的训练/预测复杂度是线性的，而二项多项式核 SVM 需

要计算核矩阵，核矩阵复杂度就是 N 平方。SVD++与 MF 类似，在特征的扩展性上都不如 FM，在此不再赘述。

转自：

http://blog.csdn.net/itplus/article/details/40534923

http://blog.csdn.net/itplus/article/details/40536025

---

logistic 回归两种形式：

第一种形式：label 取值为 0 或 1

$$P(y=1|\beta,x)=\frac{1}{1+exp(-\beta^T x)}=\frac{exp(\beta^T x)}{1+exp(\beta^T x)}$$

$$P(y=0|\beta,x)=1-\frac{1}{1+exp(-\beta^T x)}=\frac{1}{1+exp(\beta^T x)}$$

第二种形式：将 label 和预测函数放在一起，label 取值为 1 或-1

$$P(g=\pm 1|\beta,x)=\frac{1}{1+exp(-g\beta^T x)}$$

显然，$P(g=1|\beta,x)=1-P(g=-1|\beta,x)$，上述两种形式等价。

第一种形式的分类法则：

$$\frac{\frac{\exp(\boldsymbol{\beta}^T\mathbf{x})}{1+\exp(\boldsymbol{\beta}^T\mathbf{x})}}{\frac{1}{1+\exp(\boldsymbol{\beta}^T\mathbf{x})}} > 1 \rightarrow y=1$$

$$\exp(\boldsymbol{\beta}^T\mathbf{x}) > 1$$

$$\boldsymbol{\beta}^T\mathbf{x} > 0$$

第二种形式的分类法则：

$$\frac{\frac{1}{1+\exp(-\boldsymbol{\beta}^T\mathbf{x})}}{\frac{1}{1+\exp(\boldsymbol{\beta}^T\mathbf{x})}} > 1 \rightarrow g = 1$$

$$\frac{1+\exp(\boldsymbol{\beta}^T\mathbf{x})}{1+\exp(-\boldsymbol{\beta}^T\mathbf{x})} > 1$$

$$\exp(\boldsymbol{\beta}^T\mathbf{x}) > 1$$

$$\boldsymbol{\beta}^T\mathbf{x} > 0$$

第一种形式的损失函数可由极大似然估计推出，对于第二种形式的损失函数（标准的对数损失函数形式，参考 https://en.wikipedia.org/wiki/Loss_functions_for_classification 中的 logistic loss），

$$L(y, f(x)) = log(1 + exp(-yf(x))) = log(\frac{1}{P(y|\beta, x)})$$

其中，$f(x) = \beta^T x$

则 loss 最小化可表示为：

$$\underset{\beta}{\operatorname{argmin}} \sum_i L(y_i, f(x_i)) = \underset{\beta}{\operatorname{argmin}} \sum_i log(\frac{1}{P(y_i|\beta, x_i)})$$
$$= \underset{\beta}{\operatorname{argmax}} \sum_i log(P(y_i|\beta, x_i)) = \underset{\beta}{\operatorname{argmax}} \prod_i P(y_i|\beta, x_i)$$

上式最后即为极大似然估计的表示形式，则 logistic 回归模型使用的 loss 函数为对数损失函数，使用极大似然估计的目的是为了使 loss 函数最小。