半监督学习

1. 给定有标记样本集合 $\mathbb{D}_l=\{(\vec{\mathbf{x}}_1,y_1),(\vec{\mathbf{x}}_2,y_2),\cdots,(\vec{\mathbf{x}}_l,y_l)\}$ ，和未标记样本集合 $\mathbb{D}_u=\{(\vec{\mathbf{x}}_{l+1},y_{l+1}),(\vec{\mathbf{x}}_{l+2},y_{l+2}),\cdots,(\vec{\mathbf{x}}_{l+u},y_{l+u})\}$ ，其中 $l\ll u$ 。

   学习器自动地利用未标记的 $\mathbb{D}_u$ 来提升学习性能，这就是半监督学习`semi-supervised learning`。

2. 半监督学习的现实需求非常强烈，因为现实中往往能够容易地收集到大量未标记样本，但是对其标记需要耗费大量的人力、物力。如：在医学影像分析上，对影像的疾病标记需要专家人工进行。

   因此可以通过专家人工标注少量的样本，然后采用半监督学习。

3. 虽然未标记样本集 $\mathbb{D}_u$ 没有直接包含标记信息，但是如果假设 $\mathbb{D}_u$ 与带 $\mathbb{D}_l$ 从同样的数据源独立同分布采样而来，则 $\mathbb{D}_u$ 所包含的关于数据分布的信息对建立模型是有好处的。

4. 要利用未标记样本，必然需要对未标记样本的分布与已标记样本的分布的关联做出假设。
   - 最常见的假设是聚类假设`cluster assumption`：假设数据存在簇结构，同一个簇的样本属于同一个类别。
   - 另一种常见假设是流形假设`manifold assumption`：假设数据分布在一个流形结构上，邻近的样本拥有相似的输出值。其中，邻近的程度用相似度来刻画。
   - 流形假设可以看作是聚类假设的推广，但流形假设对于输出值没有限制（可以为类别，也可以为实数），因此比聚类假设的适用程度更广，可用于多类型的学习任务。
   - 无论聚类假设还是流形假设，本质都假设是：`相似的样本有相似的输出`。

5. 半监督学习可以划分为：纯`pure`半监督学习和直推学习`transduction learning`。
   - 纯半监督学习：假定训练数据中的未标记样本集 $\mathbb{D}_u$ 并非待预测的数据。

     纯半监督学习是开放性的，它学得的模型能够适用于额外的未观测数据。
   - 直推学习：假定学习过程中考虑的未标记样本集 $\mathbb{D}_u$ 就是待预测的数据，学习的目标就是在 $\mathbb{D}_u$ 上获取最优泛化性能。

     直推学习是封闭性的，它学得的模型仅仅是针对学习过程中的未标记样本集 $\mathbb{D}_u$ 。

一、生成式半监督学习方法

1. 生成式`generative methods`半监督学习方法：直接基于生成式模型的方法。
2. 生成式半监督学习方法假设所有数据（无论是否有标记），都是由同一个潜在的模型生成的。
   - 该假设使得能够通过潜在模型的参数将未标记样本与学习目标联系起来。
   - 未标记样本的标记可以视作模型的缺失参数，通常可以基于`EM`算法进行极大似然估计求解。
3. 生成式半监督学习方法其实是一个算法框架，内部不同算法的主要区别在于生成式模型的假设：不同的假设将产生不同的方法。

1.1 生成式高斯混合半监督学习

1. 给定样本 $\vec{\mathbf{x}}$ ，其真实类别标记为 $y\in\mathcal{Y}=\{1,2,\cdots,K\}$ 。

   假设样本由高斯混合模型产生，且每个类别对应一个高斯混合成分。即数据样本是基于概率密度：

$$p(\vec{\mathbf{x}})=\sum_{k=1}^{K}\alpha_k p_k(\vec{\mathbf{x}};\vec{\mu}_k,\Sigma_k)$$

来产生的。其中：

- $p_k(\vec{\mathbf{x}};\vec{\mu}_k,\Sigma_k)$ 是样本 $\vec{\mathbf{x}}$ 的第 k 个高斯混合成分的概率。
- $\vec{\mu}_k,\Sigma_k$ 为该高斯混合成分的参数。
- 混合系数 $\alpha_k\geq 0,\sum_{k=1}^K\alpha_k=1$ 。

2. 令 $f(\vec{\mathbf{x}})\in\mathcal{Y}$ 为模型 f 对 $\vec{\mathbf{x}}$ 的预测标记，$\Theta\in\{1,2,\cdots,K\}$ 表示样本 $\vec{\mathbf{x}}$ 隶属的高斯混合成分。

   根据最大化后验概率，有：

$$f(\vec{\mathbf{x}})=\arg\max_{j\in\mathcal{Y}}p(y=j\mid\vec{\mathbf{x}})$$

   - 考虑到 $p(y=j\mid\vec{\mathbf{x}})=\sum_{k=1}^K p(y=j,\Theta=k\mid\vec{\mathbf{x}})$，则有：

$$f(\vec{\mathbf{x}})=\arg\max_{j\in\mathcal{Y}}\sum_{k=1}^K p(y=j,\Theta=k\mid\vec{\mathbf{x}})$$

   - 由于 $p(y=j,\Theta=k\mid\vec{\mathbf{x}})=p(y=j\mid\Theta=k,\vec{\mathbf{x}})\cdot p(\Theta=k\mid\vec{\mathbf{x}})$，则有：

$$f(\vec{\mathbf{x}})=\arg\max_{j\in\mathcal{Y}}\sum_{k=1}^K p(y=j\mid\Theta=k,\vec{\mathbf{x}})\cdot p(\Theta=k\mid\vec{\mathbf{x}})$$

     - $p(\Theta=k\mid\vec{\mathbf{x}})$ 为已知样本 $\vec{\mathbf{x}}$ ，则它由第 k 个高斯混合成分生成的后验概率

$$p(\Theta=k\mid\vec{\mathbf{x}})=\frac{\alpha_k p_k(\vec{\mathbf{x}};\vec{\mu}_k,\Sigma_k)}{\sum_{k=1}^K\alpha_k p_k(\vec{\mathbf{x}};\vec{\mu}_k,\Sigma_k)}$$

     - $p(y=j\mid\Theta=k,\vec{\mathbf{x}})$ 为已知 $\vec{\mathbf{x}}$ 由第 k 个高斯混合成分生成，则其类别为 j 的概率

3. 在 $f(\vec{\mathbf{x}})=\arg\max_{j\in\mathcal{Y}}\sum_{k=1}^K p(y=j\mid\Theta=k,\vec{\mathbf{x}})\cdot p(\Theta=k\mid\vec{\mathbf{x}})$ 中，$p(y=j\mid\Theta=k,\vec{\mathbf{x}})$ 需要知道样本的标记 y ； 而 $p(\Theta=k\mid\vec{\mathbf{x}})$ 并不需要样本的标记。因此有标记和无标记的数据均可利用。

   因此通过引入大量的未标记数据，对 $p(y=j,\Theta=k\mid\vec{\mathbf{x}})$ 的估计可以由于数据量的增长而更为准确，于是上式的整体估计可能会更准确。

4. 给定标记样本集 $\mathbb{D}_l=\{(\vec{\mathbf{x}}_1,y_1),(\vec{\mathbf{x}}_2,y_2),\cdots,(\vec{\mathbf{x}}_l,y_l)\}$ ，和未标记样本集 $\mathbb{D}_u=\{\vec{\mathbf{x}}_{l+1},\vec{\mathbf{x}}_{l+2},\cdots,\vec{\mathbf{x}}_{l+u}\}$ ，其中 $l\ll u,\quad l+u=N$ 。

   假设所有样本独立同分布，且都是由同一个高斯混合模型 $p(\vec{\mathbf{x}})=\sum_{k=1}^K\alpha_k p_k(\vec{\mathbf{x}};\vec{\mu}_k,\Sigma_k)$ 生成的。

   - 高斯混合模型的参数 $\{(\alpha_k,\vec{\mu}_k,\Sigma_k),k=1,2,\cdots,K\}$ 采用极大似然法来估计。
   - $\mathbb{D}_l\bigcup\mathbb{D}_u$ 的对数似然是：

$$\begin{aligned}\mathcal{L}=&\sum_{(\vec{\mathbf{x}}_i,y_i)\in\mathbb{D}_l}\log\left(\sum_{k=1}^K\alpha_k p_k(\vec{\mathbf{x}}_i;\vec{\mu}_k,\Sigma_k)\cdot p(y_i\mid\Theta=k,\vec{\mathbf{x}}_i)\right)\\&+\sum_{\vec{\mathbf{x}}_i\in\mathbb{D}_u}\log\left(\sum_{k=1}^K\alpha_k p_k(\vec{\mathbf{x}}_i;\vec{\mu}_k,\Sigma_k)\right)\end{aligned}$$

     - 第一项对数项中，为联合概率 $p(\vec{\mathbf{x}}_i,y_i)$ ：

$$p(\vec{\mathbf{x}}_i,y_i)=p(y_i\mid\vec{\mathbf{x}}_i)p(\vec{\mathbf{x}}_i)=\sum_{k=1}^K\alpha_k p_k(\vec{\mathbf{x}}_i;\vec{\mu}_k,\Sigma_k)\cdot p(y_i\mid\Theta=k,\vec{\mathbf{x}}_i)$$

     - 第二项对数项中，为概率 $p(\vec{\mathbf{x}}_i)$ ：

$$p(\vec{\mathbf{x}}_i)=\sum_{k=1}^{K}\alpha_k p_k(\vec{\mathbf{x}}_i;\vec{\mu}_k,\Sigma_k)$$

5. 高斯混合模型参数估计可以用`EM`算法求解。迭代更新步骤为：
   - `E`步：根据当前模型参数 $\{(\hat{\alpha}_k,\hat{\vec{\mu}}_k,\hat{\Sigma}_k),k=1,2,\cdots,K\}$ 计算未标记样本 $\vec{\mathbf{x}}_i$ 属于各高斯混合成分的概率：

     $$\gamma_{i,k}=\frac{\hat{\alpha}_k p_k(\vec{\mathbf{x}}_i;\hat{\vec{\mu}}_k,\hat{\Sigma}_k)}{\sum_{k=1}^{K}\hat{\alpha}_k p_k(\vec{\mathbf{x}}_i;\hat{\vec{\mu}}_k,\hat{\Sigma}_k)}$$

   - `M`步：基于 $\gamma_{i,k}$ 更新模型参数。

     令 l_k 为第 k 类的有标记样本数目，则：

     $$\hat{\vec{\mu}}_k=\frac{1}{\sum_{\vec{\mathbf{x}}_i\in\mathbb{D}_u}\gamma_{i,k}+l_k}\left(\sum_{\vec{\mathbf{x}}_i\in\mathbb{D}_u}\gamma_{i,k}\vec{\mathbf{x}}_i+\sum_{(\vec{\mathbf{x}}_i,y_i)\in\mathbb{D}_l\ and\ y_i=k}\gamma_{i,k}\vec{\mathbf{x}}_i\right)$$

     $$\hat{\Sigma}_k=\frac{1}{\sum_{\vec{\mathbf{x}}_i\in\mathbb{D}_u}\gamma_{i,k}+l_k}\left(\sum_{\vec{\mathbf{x}}_i\in\mathbb{D}_u}\gamma_{i,k}(\vec{\mathbf{x}}_i-\hat{\vec{\mu}}_k)(\vec{\mathbf{x}}_i-\hat{\vec{\mu}}_k)^T+\sum_{(\vec{\mathbf{x}}_i,y_i)\in\mathbb{D}_l\ and\ y_i=k}\gamma_{i,k}(\vec{\mathbf{x}}_i-\hat{\vec{\mu}}_k)(\vec{\mathbf{x}}_i-\hat{\vec{\mu}}_k)^T\right)$$

     $$\hat{\alpha}_k=\frac{1}{N}\left(\sum_{\vec{\mathbf{x}}_i\in\mathbb{D}_u}\gamma_{i,k}+l_k\right)$$

   以上过程不断迭代直至收敛，即可获得模型参数。

6. 预测过程：根据式子：

   $$f(\vec{\mathbf{x}})=\arg\max_{j\in\mathcal{Y}}\sum_{k=1}^{K}p(y=j\mid\Theta=k,\vec{\mathbf{x}})\cdot p(\Theta=k\mid\vec{\mathbf{x}})$$

   $$p(\Theta=k\mid\vec{\mathbf{x}})=\frac{\alpha_k p_k(\vec{\mathbf{x}};\vec{\mu}_k,\Sigma_k)}{\sum_{k=1}^{K}\alpha_k p_k(\vec{\mathbf{x}};\vec{\mu}_k,\Sigma_k)}$$

   来对样本 $\vec{\mathbf{x}}$ 进行分类。

1.2 性质

1. 如果将上述过程中的高斯混合模型替换成其他模型，则可以推导出其他的生成式半监督学习方法。
2. 生成式半监督学习方法优点：方法简单，易于实现。在有标记数据极少的情况下，往往比其他方法性能更好。

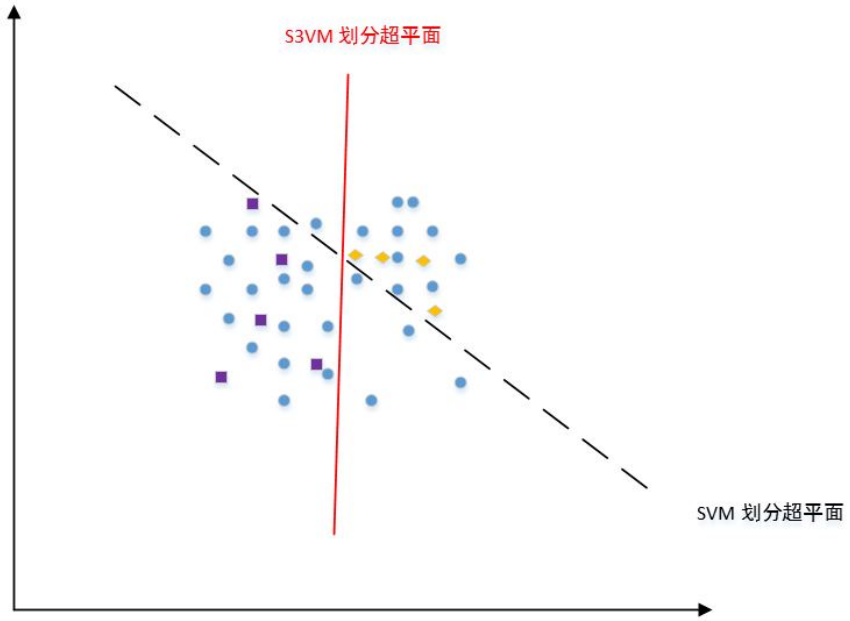
   缺点：模型假设必须准确，即假设的生成式模型必须与真实数据分布吻合，否则利用未标记数据反倒会降低泛化性能。

   在现实任务中往往很难事先做出准确的模型假设，除非拥有充分可靠的领域知识。

二、半监督 SVM

1. 半监督支持向量机`Semi-Supervised Support Vector Machine：S3VM`是支持向量机在半监督学习上的推广。
2. 在不考虑未标记样本时，支持向量机试图找到最大间隔划分超平面；在考虑未标记样本之后，`S3VM`试图找到能将两类有标记样本分开，且穿过数据低密度区域的划分超平面。

   如下图中，蓝色点为未标记样本，紫色点为正类样本，黄色点为负类样本。



3. 半监督`SVM`的基本假设是：低密度分隔`low-density separation`。这是聚类假设在考虑了线性超平面划分后的推广。

2.1 TVSM

1. 半监督支持向量机中最著名的是`TSVM:Transductive Support Vector Machine`。

   与标准`SVM`一样，`TSVM`也是针对二分类问题的学习方法。

2. `TSVM`试图考虑对未标记样本进行各种可能的标记指派`label assignment`：
   - 尝试将每个未标记样本分别作为正例或者反例。
   - 然后在所有这些结果中，寻求一个在所有样本（包括有标记样本和进行了标记指派的未标记样本）上间隔最大化的划分超平面。
   - 一旦划分超平面得以确定，未标记样本的最终标记指派就是其预测结果。
3. 给定标记样本集 $\mathbb{D}_l=\{(\vec{\mathbf{x}}_1,y_1),(\vec{\mathbf{x}}_2,y_2),\cdots,(\vec{\mathbf{x}}_l,y_l)\}$ ，和未标记样本集 $\mathbb{D}_u=\{\vec{\mathbf{x}}_{l+1},\vec{\mathbf{x}}_{l+2},\cdots,\vec{\mathbf{x}}_{l+u}\}$ ，其中 $l\ll u,\ l+u=N,\ y_i\in\{-1,+1\},i=1,2,\cdots,l$ 。

   `TSVM`学习的目标是：为 $\mathbb{D}_u$ 中的样本给出预测标记 $\hat{\mathbf{y}}=(\hat{y}_{l+1},\hat{y}_{l+2},\cdots,\hat{y}_{l+u})^T,\hat{y}_i\in\{-1,+1\},i=l+1,l+2,\cdots,N$ 使得：

$$\begin{aligned}&\min_{\vec{\mathbf{w}},b,\hat{\vec{\mathbf{y}}},\vec{\xi}}\frac{1}{2}||\vec{\mathbf{w}}||_2^2+C_l\sum_{i=1}^{l}\xi_i+C_u\sum_{i=l+1}^{N}\xi_i\\&s.t.\ \ y_i(\vec{\mathbf{w}}^T\vec{\mathbf{x}}_i+b)\geq 1-\xi_i,\quad i=1,2,\cdots,l\\&\hat{y}_i(\vec{\mathbf{w}}^T\vec{\mathbf{x}}_i+b)\geq 1-\xi_i,\quad i=l+1,l+2,\cdots,N\\&\xi_i\geq 0,\quad i=1,2,\cdots,N\end{aligned}$$

其中：

- $(\vec{\mathbf{w}},b)$ 确定了一个划分超平面。
- $\vec{\xi}$ 为松弛向量 ：
  - $\xi_i,i=1,2,\cdots,l$ 对应于有标记样本。
  - $\xi_i,i=l+1,l+2,\cdots,N$ 对应于未标记样本。
- C_l,C_u 是由用户指定的用于平衡模型复杂度、有标记样本、未标记样本重要程度的折中参数。

4. `TSVM`尝试未标记样本的各种标记指派是一个穷举过程，仅当未标记样本很少时才有可能直接求解。因此通常情况下，必须考虑更高效的优化策略。

   `TSVM` 采用局部搜索来迭代地寻求上式的近似解。具体来说：

   - 首先利用有标记样本学得一个`SVM`：即忽略上式中关于 $\mathbb{D}_u$ 与 $\hat{\vec{\mathbf{y}}}$ 的项以及约束。
   - 然后利用这个`SVM`对未标记数据进行标记指派：即将`SVM`预测的结果作为伪标记`pseudo-label`赋予未标记样本。
     - 此时 $\hat{\vec{\mathbf{y}}}$ 得到求解，将其代入上式即可得到一个标准`SVM`问题。于是求解可以解出新的划分超平面和松弛向量。
     - 注意到此时的未标记样本的伪标记很可能不准确，因此 C_u 要设置为比 C_l 小的值，使得有标记样本所起的作用更大。
   - 接下来，`TSVM`找出两个标记指派为异类且很可能发生错误的未标记样本，交换它们的标记，再重新基于上式求解出更新后的划分超平面和松弛向量。
   - 再接下来，`TSVM`再找出两个标记指派为异类且很可能发生错误的未标记样本，交换它们的标记，再重新基于上式求解出更新后的划分超平面和松弛向量。
   - ...
   - 标记指派调整完成后，逐渐增大 C_u 以提高未标记样本对优化目标的影响，进行下一轮标记指派调整，直至 C_u 达到指定阈值为止。

5. `TSVM`算法：

   - 算法输入：
     - 有标记样本集 $\mathbb{D}_l=\{(\vec{\mathbf{x}}_1,y_1),(\vec{\mathbf{x}}_2,y_2),\cdots,(\vec{\mathbf{x}}_l,y_l)\}$ ，其中 $y_i\in\{-1,+1\},i=1,2,\cdots,l$
     - 未标记样本集 $\mathbb{D}_u=\{\vec{\mathbf{x}}_{l+1},\vec{\mathbf{x}}_{l+2},\cdots,\vec{\mathbf{x}}_{l+u}\}$ ，其中 $l\ll u,\quad l+u=N$
     - 折中参数 C_l,C_u
   - 算法输出： 未标记样本的预测结果
     $\hat{\vec{\mathbf{y}}}=(\hat{y}_{l+1},\hat{y}_{l+2},\cdots,\hat{y}_{l+u})^T,\hat{y}_i\in\{-1,+1\},i=l+1,l+2,\cdots,N$
   - 算法步骤：
     - 用 $\mathbb{D}_l$ 训练一个`SVM_1`
     - 用`SVM_1`对 $\mathbb{D}_u$ 中样本进行预测，得到 $\hat{\vec{\mathbf{y}}}=(\hat{y}_{l+1},\hat{y}_{l+2},\cdots,\hat{y}_{l+u})^T$
     - 初始化 $\tilde{C}_u$ ，其中 $\tilde{C}_u\ll C_u,\tilde{C}_u>0$
     - 迭代，迭代终止条件为 $\tilde{C}_u\geq C_u$ 。迭代过程为：
       - 基于 $\mathbb{D}_l,\mathbb{D}_u,\hat{\vec{\mathbf{y}}},C_l,\tilde{C}_u$ ，求解下式，得到 $(\vec{\mathbf{w}},b),\vec{\xi}$：

$$\min_{\vec{\mathbf{w}},b,\hat{\mathbf{y}},\vec{\xi}} \frac{1}{2}||\vec{\mathbf{w}}||_2^2+C_l\sum_{i=1}^{l}\xi_i+\tilde{C}_u\sum_{i=l+1}^{N}\xi_i$$
$$s.t.\ y_i(\vec{\mathbf{w}}^T\vec{\mathbf{x}}_i+b)\geq 1-\xi_i,\quad i=1,2,\cdots,l$$
$$\hat{y}_i(\vec{\mathbf{w}}^T\vec{\mathbf{x}}_i+b)\geq 1-\xi_i,\quad i=l+1,l+2,\cdots,N$$
$$\xi_i\geq 0,\quad i=1,2,\cdots,N$$

- 对于所有的一对标记指派为异类且很可能发生错误的未标记样本（其条件为：$\hat{y}_i\hat{y}_j<0 \text{ and } \xi_i>0 \text{ and } \xi_j>0 \text{ and } \xi_i+\xi_j>2$），执行下列操作：
  - 交换二者的标记：$\hat{y}_i=-\hat{y}_i,\quad \hat{y}_j=-\hat{y}_j$。

    > 该操作等价于交换标记，因为 $\hat{y}_i\hat{y}_j$ 异号，且其取值为 -1 或者 +1。

  - 基于 $\mathbb{D}_l,\mathbb{D}_u,\hat{\mathbf{y}},C_l,\tilde{C}_u$，重新求解得到得到 $(\vec{\mathbf{w}},b),\vec{\xi}$。
- 更新 $\tilde{C}_u=\min(2\tilde{C}_u,C_u,)$。这里采用简单的倍乘，也可以采用其它增长函数。
- 迭代终止时，输出 $\hat{\mathbf{y}}=(\hat{y}_{l+1},\hat{y}_{l+2},\cdots,\hat{y}_{l+u})^T$

6. 在对未标记样本进行指标指派及调整的过程中，有可能出现类别不平衡问题，即某类的样本远多于另一类。这将对`SVM`的训练造成困扰。

   为了减轻类别不平衡性造成的不利影响，可对上述算法稍加改进：将优化目标中的 C_u 项拆分为 C_u^+ 和 C_u^- 两项，分别对应基于伪标记而当作正、反例使用的未标记样本。并在初始化时，令：

   $$C_u^+=\frac{u_-}{u_+}C_u^-$$

   其中 u_- 和u_+ 分别为基于伪标记而当作反、正例而使用的未标记样本数。

2.2 性质

1. `TSVM` 最终得到的`SVM` 不仅可以给未标记样本提供了标记，还能对训练过程中未见的样本进行预测。
2. 在 `TSVM` 算法中，寻找标记指派可能出错的每一对未标记样本进行调整，这是一个涉及巨大计算开销的大规模优化问题。
   - 在论文`《Large Scale Transductive SVMs》` 中，约 2000 个未标记样本，原始`TVSM` 迭代收敛大约需要 1个小时。
   - 半监督`SVM`研究的一个重点是如何设计出高效的优化求解策略。由此发展成很多方法，如基于图核函数梯度下降的`LDS`算法，基于标记均值估计的`meanS3VM`算法等。

三、图半监督学习

3.1 标签传播算法

1. 给定一个数据集，可以将其映射为一个图，数据集中每个样本对应于图中的一个结点。若两个样本之间的相似度很高（或者相关性很强），则对应的结点之间存在一条边，边的强度正比于样本之间的相似度（或相关性）。

   将有标记样本所对应的结点视作为已经染色，而未标记样本所对应的结点尚未染色。于是半监督学习就对应于“颜色”在图上扩散或者传播的过程。这就是标记传播算法`label propagation`。

2. 给定标记样本集 $\mathbb{D}_l=\{(\vec{\mathbf{x}}_1,y_1),(\vec{\mathbf{x}}_2,y_2),\cdots,(\vec{\mathbf{x}}_l,y_l)\},y_i\in\{-1,+1\}$ ，和未标记样本集 $\mathbb{D}_u=\{\vec{\mathbf{x}}_{l+1},\vec{\mathbf{x}}_{l+2},\cdots,\vec{\mathbf{x}}_{l+u}\}$，其中 $l\ll u,\quad l+u=N$ 。

   基于 $\mathbb{D}_l\bigcup\mathbb{D}_u$ 构建一个图 $\mathcal{G}=(\mathbb{V},\mathbb{E})$ 。其中

   - 结点集 $\mathbb{V}=\{\vec{\mathbf{x}}_1,\vec{\mathbf{x}}_2,\cdots,\vec{\mathbf{x}}_l,\vec{\mathbf{x}}_{l+1},\vec{\mathbf{x}}_{l+2},\cdots,\vec{\mathbf{x}}_{l+u}\}$
   - 边集 $\mathbb{E}$ 的权重可以表示为一个亲和矩阵 `affinity matirx` $\mathbf{W}=(w_{i,j})_{N\times N}$，一般基于高斯函数，其定义为：

     $$w_{i,j}=\begin{cases}\exp\left(-\frac{||\vec{\mathbf{x}}_i-\vec{\mathbf{x}}_j||_2^2}{2\sigma^2}\right), & i\neq j\\ 0, & i=j\end{cases},\quad i,j\in\{1,2,\cdots,N\}$$

     其中 $\sigma>0$ 是用户指定的高斯函数带宽参数。

     可以看到：

     - $w_{i,j}=w_{j,i}$ ，因此 $\mathbf{W}$ 为对称矩阵。
     - 图 $\mathcal{G}$ 是全连接的，任意两点之间都存在边。
     - 两个点的距离越近，说明两个样本越相似，则边的权重越大；距离越远，说明两个样本越不相似，则边的权重越小。
     - 权重越大说明样本越相似，则标签越容易传播。

3.1.1 能量函数

1. 假定从图 $\mathcal{G}=(\mathbb{V},\mathbb{E})$ 学得一个实值函数 $f:\mathbb{V}\to\mathbb{R}$， 其对应的分类规则为：$y_i=\mathrm{sign}(f(\vec{\mathbf{x}}_i)),y_i\in\{-1,+1\}$。

   直观上看，相似的样本应该具有相似的标记，于是可以定义关于 f 的能量函数 `energy function`：

   $$E(f)=\frac{1}{2}\sum_{i=1}^N\sum_{j=1}^N w_{i,j}(f(\vec{\mathbf{x}}_i)-f(\vec{\mathbf{x}}_j))^2$$

   $$=\frac{1}{2}\sum_{i=1}^N\sum_{j=1}^N\left[w_{i,j}f(\vec{\mathbf{x}}_i)^2+w_{i,j}f(\vec{\mathbf{x}}_j)^2-2w_{i,j}f(\vec{\mathbf{x}}_i)f(\vec{\mathbf{x}}_j)\right]$$

   $$=\frac{1}{2}\left[\sum_{i=1}^N\left(f(\vec{\mathbf{x}}_i)^2\sum_{j=1}^N w_{i,j}\right)+\sum_{j=1}^N\left(f(\vec{\mathbf{x}}_j)^2\sum_{i=1}^N w_{i,j}\right)-2\sum_{i=1}^N\sum_{j=1}^N w_{i,j}f(\vec{\mathbf{x}}_i)f(\vec{\mathbf{x}}_j)\right]$$

   - 两个点距离越远，$(f(\vec{\mathbf{x}}_i)-f(\vec{\mathbf{x}}_j))^2$ 平方级的增大，而 $w_{i,j}$ 指数级下降，因此 $w_{i,j}(f(\vec{\mathbf{x}}_i)-f(\vec{\mathbf{x}}_j))^2$ 下降。

     因此能量函数 $E(f)$ 由距离较近的样本对决定。

   - 标签传播算法假定系统能量最小，即 $E(f)$ 最小。

     考虑到$E(f)$ 由距离较近的样本对决定，而 $w_{w,j}$ 是已知的量，因此算法倾向于使得：**距离较近的样本具有相近的输出** 。

2. 定义对角矩阵 $\mathbf{D}=\mathrm{diag}(d_1,d_2,\cdots,d_N)$ ，其中 $d_i=\sum_{j=1}^N w_{i,j}$ 为矩阵 $\mathbf{W}$ 的第 i 行元素之和。

   d_i 的物理意义为：第 i 个顶点的度（所有与它相连的边的权重之和）。因此 $\mathbf{D}$ 也称作度矩阵。

$$\mathbf{D}=\begin{bmatrix}d_1 & 0 & 0 & \cdots & 0\\ 0 & d_2 & 0 & \cdots & 0\\ 0 & 0 & d_3 & \cdots & 0\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & 0 & \cdots & d_N\end{bmatrix}$$

定义 $\vec{\mathbf{f}}=(f(\vec{\mathbf{x}}_1),\cdots,f(\vec{\mathbf{x}}_l),f(\vec{\mathbf{x}}_{l+1}),\cdots,f(\vec{\mathbf{x}}_N))^T$ 为函数 f 在所有样本上的预测结果。 其中：

- $\vec{\mathbf{f}}_l=(f(\vec{\mathbf{x}}_1),f(\vec{\mathbf{x}}_2),\cdots,f(\vec{\mathbf{x}}_l))^T$ 为函数 f 在有标记样本上的预测结果。
- $\vec{\mathbf{f}}_u=(f(\vec{\mathbf{x}}_{l+1}),f(\vec{\mathbf{x}}_{l+2}),\cdots,f(\vec{\mathbf{x}}_{l+u}))^T$ 为函数 f 在未标记样本上的预测结果。

结合 $\mathbf{D}$ 的定义以及 $\mathbf{W}$ 的对称性，有：

$$\begin{aligned}E(f)&=\frac{1}{2}\left[\sum_{i=1}^N f(\vec{\mathbf{x}}_i)^2d_i+\sum_{j=1}^N f(\vec{\mathbf{x}}_j)^2d_j-2\sum_{i=1}^N\sum_{j=1}^N w_{i,j}f(\vec{\mathbf{x}}_i)f(\vec{\mathbf{x}}_j)\right]\\&=\sum_{i=1}^N f(\vec{\mathbf{x}}_i)^2d_i-\sum_{i=1}^N\sum_{j=1}^N w_{i,j}f(\vec{\mathbf{x}}_i)f(\vec{\mathbf{x}}_j)=\vec{\mathbf{f}}^T(\mathbf{D}-\mathbf{W})\vec{\mathbf{f}}\end{aligned}$$

3. 标签传播算法将样本 $\vec{\mathbf{x}}$ 的标记 $f(\vec{\mathbf{x}})$ 视作能量。
   - 有标记样本的能量是已知的，未标记样本的能量是未知的。
   - 能量在样本之间流动。对于样本 $\vec{\mathbf{x}}_i$ ，它流向样本 $\vec{\mathbf{x}}_j$ 的能量为 $w_{i,j}f(\vec{\mathbf{x}}_j)$ 。
     - $w_{i,j}$ 表示边的权重，它就是能量流出的比例（类比于管道的大小）。
     - 流出的能量可正可负，因为 $f(\vec{\mathbf{x}})\in\{1,-1\}$ 。
     - 注意：能量不能在有标记样本与有标记样本之间流动，也不能从未标记样本流向有标记样本。
   - 流经每个未标记样本的能量是守恒的。对未标记样本$\vec{\mathbf{x}}_i,i=l+1,\cdots,N$ ：
     - 其能量流向其它的所有**未标记结点**，能量流出为：$\sum_{j=l+1}^N w_{i,j}f(\vec{\mathbf{x}}_j)$ 。
     - 其它所有结点（**包括有标记样本**）都向其汇入能量，能量流入为：$\sum_{j=1}^N w_{j,i}f(\vec{\mathbf{x}}_i)$ 。

     考虑到 $w_{i,j}=w_{j,i}$ ，以及 $d_i=\sum_{j=1}^N w_{i,j}$ ，则有：$d_if(\vec{\mathbf{x}}_i)=\sum_{j=l+1}^N w_{i,j}f(\vec{\mathbf{x}}_j)$ 。 考虑所有的未标记样本，则有：

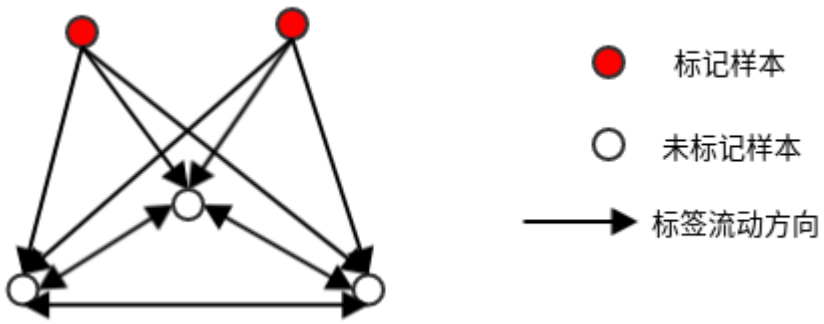
$$\begin{bmatrix}d_{l+1} & 0 & \cdots & 0\\ 0 & d_{l+2} & \cdots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \cdots & d_N\end{bmatrix}\times\begin{bmatrix}f(\vec{\mathbf{x}}_{l+1})\\ f(\vec{\mathbf{x}}_{l+2})\\ \vdots\\ f(\vec{\mathbf{x}}_N)\end{bmatrix}=\begin{bmatrix}w_{l+1,l+1} & w_{l+1,l+2} & \cdots & w_{l+1,N}\\ w_{l+2,l+1} & w_{l+2,l+2} & \cdots & w_{l+2,N}\\ \vdots & \vdots & \ddots & \vdots\\ w_{N,l+1} & w_{N,l+2} & \cdots & w_{N,N}\end{bmatrix}\times\begin{bmatrix}f(\vec{\mathbf{x}}_{l+1})\\ f(\vec{\mathbf{x}}_{l+2})\\ \vdots\\ f(\vec{\mathbf{x}}_N)\end{bmatrix}$$

   - 从每个有标记样本流出的能量也是守恒的。对于有标记样本$\vec{\mathbf{x}}_i,i=1,\cdots,l$ ，它**仅仅流出到未标记样本**，因此流出能量为：$\sum_{j=1}^l w_{i,j}f(\vec{\mathbf{x}}_j)$ 。

     由于有标记样本只有能量流出，没有能量流入，因此有：$\sum_{j=1}^l w_{i,j}f(\vec{\mathbf{x}}_j)=0$ 。
   - 综合两种能量守恒的情况，有：

$$\mathbf{D}\times\begin{bmatrix}0\\ \vdots\\ 0\\ f(\vec{\mathbf{x}}_{l+1})\\ \vdots\\ f(\vec{\mathbf{x}}_N)\end{bmatrix}=\mathbf{W}\times\begin{bmatrix}0\\ \vdots\\ 0\\ f(\vec{\mathbf{x}}_{l+1})\\ \vdots\\ f(\vec{\mathbf{x}}_N)\end{bmatrix}$$

即有：$(\mathbf{D}-\mathbf{W})(0,\cdots,0,f(\vec{\mathbf{x}}_{l+1}),\cdots,f(\vec{\mathbf{x}}_N))^T=\vec{\mathbf{0}}$ 。



4. 标签传播算法假定在满足约束条件的条件下，能量函数 $E(f)$ 最低。其中约束条件为：
   - 标记约束：函数 f 在标记样本上满足 $f(\vec{\mathbf{x}}_i)=y_i,i=1,2,\cdots,l$ 。
   - 能量守恒：定义拉普拉斯矩阵 $\mathbf{L}=\mathbf{D}-\mathbf{W}$ ，则有：$\mathbf{L}(0,\cdots,0,f(\vec{\mathbf{x}}_{l+1}),\cdots,f(\vec{\mathbf{x}}_N))^T=\vec{\mathbf{0}}$

   因此标签传播算法就是求解约束最优化问题：

$$\min_f E(f)$$
$$s.t.\quad f(\vec{\mathbf{x}}_i)=y_i,i=1,2,\cdots,l$$
$$(\mathbf{D}-\mathbf{W})(0,\cdots,0,f(\vec{\mathbf{x}}_{l+1}),\cdots,f(\vec{\mathbf{x}}_N))^T=\vec{\mathbf{0}}$$

.

3.1.2 最优化求解

1. 以第 l 行第 l 列为界，采用分块矩阵表示方式：

$$\mathbf{W}=\begin{bmatrix}\mathbf{W}_{l,l} & \mathbf{W}_{l,u}\\ \mathbf{W}_{u,l} & \mathbf{W}_{u,u}\end{bmatrix},\quad \mathbf{D}=\begin{bmatrix}\mathbf{D}_{l,l} & \mathbf{0}_{l,u}\\ \mathbf{0}_{u,l} & \mathbf{D}_{u,u}\end{bmatrix}$$

   则：

$$E(f)=\vec{\mathbf{f}}^T(\mathbf{D}-\mathbf{W})\vec{\mathbf{f}}=(\vec{\mathbf{f}}_l^T\quad\vec{\mathbf{f}}_u^T)\left(\begin{bmatrix}\mathbf{D}_{l,l} & \mathbf{0}_{l,u}\\ \mathbf{0}_{u,l} & \mathbf{D}_{u,u}\end{bmatrix}-\begin{bmatrix}\mathbf{W}_{l,l} & \mathbf{W}_{l,u}\\ \mathbf{W}_{u,l} & \mathbf{W}_{u,u}\end{bmatrix}\right)\begin{bmatrix}\vec{\mathbf{f}}_l\\ \vec{\mathbf{f}}_u\end{bmatrix}$$
$$=\vec{\mathbf{f}}_l^T(\mathbf{D}_{l,l}-\mathbf{W}_{l,l})\vec{\mathbf{f}}_l-2\vec{\mathbf{f}}_u^T\mathbf{W}_{u,l}\vec{\mathbf{f}}_l^T+\vec{\mathbf{f}}_u^T(\mathbf{D}_{u,u}-\mathbf{W}_{u,u})\vec{\mathbf{f}}_u$$

   考虑到 $\vec{\mathbf{f}}_l$ 是已知的，因此 $E(f)$ 完全由 $\vec{\mathbf{f}}_u$ 决定。为求得 $E(f)$ 的最小值，则根据 $\frac{\partial E(f)}{\partial\vec{\mathbf{f}}_u}=\vec{\mathbf{0}}$ 有：

$$\vec{\mathbf{f}}_u=(\mathbf{D}_{u,u}-\mathbf{W}_{u,u})^{-1}\mathbf{W}_{u,l}\vec{\mathbf{f}}_l$$

2. 令:

$$\mathbf{P}=\mathbf{D}^{-1}\mathbf{W}=\begin{bmatrix}\mathbf{D}_{l,l}^{-1} & \mathbf{0}_{l,u}\\ \mathbf{0}_{u,l} & \mathbf{D}_{u,u}^{-1}\end{bmatrix}\begin{bmatrix}\mathbf{W}_{l,l} & \mathbf{W}_{l,u}\\ \mathbf{W}_{u,l} & \mathbf{W}_{u,u}\end{bmatrix}=\begin{bmatrix}\mathbf{D}_{l,l}^{-1}\mathbf{W}_{l,l} & \mathbf{D}_{l,l}^{-1}\mathbf{W}_{l,u}\\ \mathbf{D}_{u,u}^{-1}\mathbf{W}_{u,l} & \mathbf{D}_{u,u}^{-1}\mathbf{W}_{u,u}\end{bmatrix}$$

   令：$\mathbf{P}_{u,u}=\mathbf{D}_{u,u}^{-1}\mathbf{W}_{u,u},\quad \mathbf{P}_{u,l}=\mathbf{D}_{u,u}^{-1}\mathbf{W}_{u,l}$ ，则有：

$$\begin{aligned}\vec{\mathbf{f}}_u &= (\mathbf{D}_{u,u}-\mathbf{W}_{u,u})^{-1}\mathbf{W}_{u,l}\vec{\mathbf{f}}_l\\ &= (\mathbf{D}_{u,u}(\mathbf{I}-\mathbf{D}_{u,u}^{-1}\mathbf{W}_{u,u}))^{-1}\mathbf{W}_{u,l}\vec{\mathbf{f}}_l\\ &= (\mathbf{I}-\mathbf{D}_{u,u}^{-1}\mathbf{W}_{u,u})^{-1}\mathbf{D}_{u,u}^{-1}\mathbf{W}_{u,l}\vec{\mathbf{f}}_l\\ &= (\mathbf{I}-\mathbf{P}_{u,u})^{-1}\mathbf{P}_{u,l}\vec{\mathbf{f}}_l\end{aligned}$$

于是，将 $\mathbb{D}_l$ 上的标记信息作为 $\vec{\mathbf{f}}_l=(y_1,y_2,\cdots,y_l)^T$ 代入上式，即可求得未标记样本的预测值 $\vec{\mathbf{f}}_u$ 。

3. $\mathbf{P}=(p_{i,j})_{N\times N}$ 矩阵其实为概率转移矩阵：

$$p_{i,j}=\frac{w_{i,j}}{d_i}=\frac{w_{i,j}}{\sum_{j=1}^N w_{i,j}}$$

它表示从节点 i 转移到节点 j 的概率。

注意到 $\frac{w_{i,j}}{d_i}\neq\frac{w_{j,i}}{d_j}$ ，因此 $\mathbf{P}$ 不是对称的。即：`节点 i 转移到节点 j 的概率` 不等于 `节点i 转移到节点 j 的概率` 。因此在`Label Spreading` 算法中，提出新的标记传播矩阵 $\mathbf{S}=\mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$ ：

$$\mathbf{S}=\begin{bmatrix}\frac{w_{1,1}}{d_1} & \frac{w_{1,2}}{\sqrt{d_1\times d_2}} & \cdots & \frac{w_{1,N}}{\sqrt{d_1\times d_N}}\\ \frac{w_{2,1}}{\sqrt{d_2\times d_1}} & \frac{w_{2,2}}{d_2} & \cdots & \frac{w_{2,N}}{\sqrt{d_2\times d_N}}\\ \vdots & \vdots & \ddots & \vdots\\ \frac{w_{N,1}}{\sqrt{d_N\times d_1}} & \frac{w_{N,2}}{\sqrt{d_N\times d_2}} & \cdots & \frac{w_{N,N}}{d_N}\end{bmatrix}$$

因此有：$\mathbf{S}=\mathbf{S}^T$ ，`节点 i 转移到节点 j 的概率` 等于 `节点i 转移到节点 j 的概率` 。此时的转移概率是非归一化的概率。

4. 矩阵求逆 $(\mathbf{I}-\mathbf{P}_{u,u})^{-1}$ 的算法复杂度是 $O(u^3)$ 。如果未标记样本的数量 u 很大，则求逆的过程非常耗时。这时一般选择迭代算法来实现。
   - 首先执行初始化：$\vec{\mathbf{f}}^{<0>}=(y_1,y_2,\cdots,y_l,0,\cdots,0)^T$ 。
   - 迭代过程：
     - 执行标签传播：$\vec{\mathbf{f}}^{<t+1>}=\mathbf{P}\vec{\mathbf{f}}^{<t>}$ 。
     - 重置 $\vec{\mathbf{f}}$ 中的标签样本标记：$\vec{\mathbf{f}}_l^{<t+1>}=(y_1,y_2,\cdots,y_l)^T$ 。

3.2 多类标签传播算法

1. 给定标记样本集 $\mathbb{D}_l=\{(\vec{\mathbf{x}}_1,y_1),(\vec{\mathbf{x}}_2,y_2),\cdots,(\vec{\mathbf{x}}_l,y_l)\},y_i\in\{1,2,\cdots,K\}$ ，和未标记样本集 $\mathbb{D}_u=\{\vec{\mathbf{x}}_{l+1},\vec{\mathbf{x}}_{l+2},\cdots,\vec{\mathbf{x}}_{l+u}\}$ ，其中 $l\ll u,\quad l+u=N$ 。

   与二类标签传播算法一样，首先基于 $\mathbb{D}_l\bigcup\mathbb{D}_u$ 构建一个图 $\mathcal{G}=(\mathbb{V},\mathbb{E})$ ，然后定义边的权重矩阵 $\mathbf{W}$ 和度矩阵 $\mathbf{D}$ 。

2. 令 $\vec{\mathbf{x}}_i$ 的标记向量为 $\vec{\mathbf{F}}_i=(F_{i,1},F_{i,2},\cdots,F_{i,K})^T$ ， 其中 $F_{i,k}$ 表示 $\vec{\mathbf{x}}_i$ 属于类别 k 的概率。根据概率的定义有：

$$\sum_{k=1}^K F_{i,k}=1,\quad F_{i,k}\geq 0$$

   - 对于标记样本 $\vec{\mathbf{x}}_i$ ，$(F_{i,1},F_{i,2},\cdots,F_{i,K})$ 中只有一个值为1 ，其他值为 0。设 $\vec{\mathbf{x}}_i$ 的标记为 $\tilde{K}$ ，即有：

$$F_{i,k}=\begin{cases}1, & k=\tilde{K}\\ 0, & k!=\tilde{K}\end{cases}$$

- 对于未标记样本 $\vec{\mathbf{x}}_i$ ，$(F_{i,1},F_{i,2},\cdots,F_{i,K})$ 表示一个概率分布，依次是该样本属于各个类别的概率。

3. 当给定 $\vec{\mathbf{x}}_i$ 的标记向量 $\vec{\mathbf{F}}_i$ 时，样本的分类规则为：

$$\hat{y}_i=\arg\max_{1\leq j\leq K}F_{i,j}$$

其中 $\hat{y}_i$ 表示模型对样本 $\vec{\mathbf{x}}_i$ 的预测输出。

4. 定义非负的标记矩阵为： $\mathbf{F}=(\vec{\mathbf{F}}_1,\vec{\mathbf{F}}_2,\cdots,\vec{\mathbf{F}}_N)^T\in\mathbb{R}^{N\times K}$

$$\mathbf{F}=\begin{bmatrix}F_{1,1}&F_{1,2}&\cdots&F_{1,K}\\F_{2,1}&F_{2,2}&\cdots&F_{2,K}\\\vdots&\vdots&\ddots&\vdots\\F_{N,1}&F_{N,2}&\cdots&F_{N,K}\end{bmatrix}$$

即：$\mathbf{F}$ 的每一行代表每个样本属于各类别的概率。因此 $\mathbf{F}$ 也称为软标记`soft label`矩阵。

定义非负的常量矩阵 $\mathbf{Y}=(Y_{i,j})_{N\times K}$ 为：

$$Y_{i,j}=\begin{cases}1,&\text{if}\quad 1\leq i\leq l\quad\text{and}\quad y_i=j\\0,&\text{otherwise}\end{cases}$$

即：$\mathbf{Y}$ 的前 l 行就是 l 个有标记样本的标记向量。后 u 行全为 0 。

3.2.1 Label Propagation

1. `Label Propagation` 算法通过节点之间的边来传播标记，边的权重越大则表示两个节点越相似，则标记越容易传播。
   - 定义概率转移矩阵 $\mathbf{P}=(p_{i,j})_{N\times N}$ ，其中：

$$p_{i,j}=\frac{w_{i,j}}{d_i}=\frac{w_{i,j}}{\sum_{j=1}^N w_{i,j}}$$

     它表示标签从结点 i 转移到结点 j 的概率。
   - 定义标记矩阵 $\mathbf{Y}_l=(Y^l_{i,j})_{l\times K}$，其中：

$$Y_{i,j}=\begin{cases}1,&\text{if}\quad y_i=j\\0,&\text{otherwise}\end{cases}$$

     即：若 $y_i=k$，则第 i 行的第 k 个元素为1，该行其他元素都是 0。
   - 定义未标记矩阵 $\mathbf{Y}_u=(Y^u_{i,j})_{u\times K}$ ，矩阵的第 i 行为样本 $\vec{\mathbf{x}}_{l+u}$ 属于各标签的概率。
   - 合并 $\mathbf{Y}_l$ 和 $\mathbf{Y}_u$ 即可得到 $\mathbf{F}$ 。
2. `Label Propagation` 是个迭代算法。算法流程为：
   - 首先执行初始化：$\mathbf{F}^{<0>}=\mathbf{Y}$ 。
   - 迭代过程：
     - 执行标签传播：$\mathbf{F}^{<t+1>}=\mathbf{P}\mathbf{F}^{<t>}$ 。
     - 重置 $\mathbf{F}$ 中的标签样本标记：$\mathbf{F}_l^{<t+1>}=\mathbf{Y}_l$ ，其中 $\mathbf{F}_l$ 表示 $\mathbf{F}$ 的前 l 行。
3. `Label Propatation` 算法：
   - 输入：

- 有标记样本集 $\mathbb{D}_l=\{(\vec{\mathbf{x}}_1,y_1),(\vec{\mathbf{x}}_2,y_2),\cdots,(\vec{\mathbf{x}}_l,y_l)\},y_i\in\{1,2,\cdots,K\}$
- 未标记样本集 $\mathbb{D}_u=\{\vec{\mathbf{x}}_{l+1},\vec{\mathbf{x}}_{l+2},\cdots,\vec{\mathbf{x}}_{l+u}\},l+u=N$
- 构图参数 σ

- 输出：未标记样本的预测结果
  $\hat{\mathbf{y}}=(\hat{y}_{l+1},\hat{y}_{l+2},\cdots,\hat{y}_{l+u})^T,\hat{y}_i\in\{1,2,\cdots,K\},i=l+1,l+2,\cdots,l+u$
- 步骤：
  - 根据下式，计算 $\mathbf{W}$：

$$w_{i,j}=\begin{cases}\exp\left(-\frac{||\vec{\mathbf{x}}_i-\vec{\mathbf{x}}_j||_2^2}{2\sigma^2}\right), & i\neq j\\ 0, & i=j\end{cases},\quad i,j\in\{1,2,\cdots,N\}$$

  - 基于 $\mathbf{W}$ 构造概率转移矩阵 $\mathbf{P}=\mathbf{D}^{-1}\mathbf{W}$。其中 $\mathbf{D}^{-1}=\mathrm{diag}(\frac{1}{d_1},\frac{1}{d_2},\cdots,\frac{1}{d_N})$，$d_i=\sum_{j=1}^N w_{i,j}$ 为矩阵 $\mathbf{W}$ 的第 i 行元素之和。
  - 构造非负的常量矩阵 $\mathbf{Y}=(Y_{i,j})_{N\times K}$：

$$Y_{i,j}=\begin{cases}1, & \text{if}\quad 1\leq i\leq l\quad\text{and}\quad y_i=j\\ 0, & \text{otherwise}\end{cases}$$

  - 初始化 $\mathbf{F}^{<0>}$：$\mathbf{F}^{<0>}=\mathbf{Y}$
  - 初始化 $t=0$
  - 迭代，迭代终止条件是 $\mathbf{F}$ 收敛至 $\mathbf{F}^*$。迭代过程为：
    - $\mathbf{F}^{<t+1>}=\mathbf{P}\mathbf{F}^{<t>}$
    - $\mathbf{F}_l^{<t+1>}=\mathbf{Y}_l$，其中 $\mathbf{F}_l$ 表示 $\mathbf{F}$ 的前 l 行。
    - $t=t+1$
  - 构造未标记样本的预测结果：

$$\hat{y}_i=\arg\max_{j\in\{1,2,\cdots,K\}}F^*_{i,j},i=l+1,l+2,\cdots,N$$

  - 输出结果 $\hat{\mathbf{y}}=(\hat{y}_{l+1},\hat{y}_{l+2},\cdots,\hat{y}_{l+u})^T$

3.2.2 Label Spreading

1. `Label Spreading` 算法也是个迭代算法：
   - 首先初始化 $\mathbf{F}$ 为：$\mathbf{F}^{<0>}=\mathbf{Y}$，其中 $<0>$ 表示第 0 次迭代。
   - 然后迭代：$\mathbf{F}^{<t+1>}=\alpha\mathbf{S}\mathbf{F}^{<t>}+(1-\alpha)\mathbf{Y}$。其中：
     - $\mathbf{S}$ 为标记传播矩阵 $\mathbf{S}=\mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$，其中 $\mathbf{D}^{-1/2}=\mathrm{diag}(\frac{1}{\sqrt{d_1}},\frac{1}{\sqrt{d_2}},\cdots,\frac{1}{\sqrt{d_N}})$。
     - $\alpha\in(0,1)$ 为用户指定的参数，用于对标记传播项 $\mathbf{S}\mathbf{F}^{<t>}$ 和初始化项 $\mathbf{Y}$ 的重要性进行折中。
       - $\alpha\to 0$，则每次迭代时尽可能保留初始化项 $\mathbf{Y}$。最终数据集的分布与初始化项 $\mathbf{Y}$ 非常相近。
       - $\alpha\to 1$，则每次迭代时尽可能不考虑 $\mathbf{Y}$。最终数据集的分布与 $\mathbf{Y}$ 差距较大。

   迭代直至收敛，可以得到：$\mathbf{F}^*=\lim_{t\to\infty}\mathbf{F}^{<t>}=(1-\alpha)(\mathbf{I}-\alpha\mathbf{S})^{-1}\mathbf{Y}$。于是可以从 $\mathbf{F}^*$ 中可以获取 $\mathbb{D}_u$ 中样本的标记。

2. 由于引入 α ，因此 `Label Spreading` 最终收敛的结果中，标记样本的标签可能会发生改变。这在一定程度上能对抗噪音的影响。

   如：$\alpha=0.2$ 意味着有 80% 的初始标记样本的标签会保留下来，有 20% 的初始标记样本的标签发生改变。

3. `Label Spreading` 算法：

   - 输入：
     - 有标记样本集 $\mathbb{D}_l=\{(\vec{\mathbf{x}}_1,y_1),(\vec{\mathbf{x}}_2,y_2),\cdots,(\vec{\mathbf{x}}_l,y_l)\},y_i\in\{1,2,\cdots,K\}$
     - 未标记样本集 $\mathbb{D}_u=\{\vec{\mathbf{x}}_{l+1},\vec{\mathbf{x}}_{l+2},\cdots,\vec{\mathbf{x}}_{l+u}\},l+u=N$
     - 构图参数 σ
     - 折中参数 α
   - 输出：未标记样本的预测结果
     $\hat{\vec{\mathbf{y}}}=(\hat{y}_{l+1},\hat{y}_{l+2},\cdots,\hat{y}_{l+u})^T,\hat{y}_i\in\{1,2,\cdots,K\},i=l+1,l+2,\cdots,l+u$
   - 步骤：
     - 根据下式，计算 $\mathbf{W}$：

     $$w_{i,j}=\begin{cases}\exp\left(-\frac{||\vec{\mathbf{x}}_i-\vec{\mathbf{x}}_j||_2^2}{2\sigma^2}\right), & i\neq j\\ 0, & i=j\end{cases},\quad i,j\in\{1,2,\cdots,N\}$$

     - 基于 $\mathbf{W}$ 构造标记传播矩阵 $\mathbf{S}=\mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$。其中 $\mathbf{D}^{-1/2}=\text{diag}(\frac{1}{\sqrt{d_1}},\frac{1}{\sqrt{d_2}},\cdots,\frac{1}{\sqrt{d_N}})$ ，$d_i=\sum_{j=1}^N w_{i,j}$ 为矩阵 $\mathbf{W}$ 的第 i 行元素之和。
     - 构造非负的常量矩阵 $\mathbf{Y}=(Y_{i,j})_{N\times K}$ ：

     $$Y_{i,j}=\begin{cases}1, & \text{if}\quad 1\leq i\leq l\quad\text{and}\quad y_i=j\\ 0, & \text{otherwise}\end{cases}$$

     - 初始化 $\mathbf{F}^{<0>}$ ：$\mathbf{F}^{<0>}=\mathbf{Y}$
     - 初始化 $t=0$
     - 迭代，迭代终止条件是 $\mathbf{F}$ 收敛至 $\mathbf{F}^*$。迭代过程为：
       - $\mathbf{F}^{<t+1>}=\alpha\mathbf{S}\mathbf{F}^{<t>}+(1-\alpha)\mathbf{Y}$
       - $t=t+1$
     - 构造未标记样本的预测结果：

     $$\hat{y}_i=\arg\max_{j\in\{1,2,\cdots,K\}}F^*_{i,j},i=l+1,l+2,\cdots,N$$

     - 输出结果 $\hat{\vec{\mathbf{y}}}=(\hat{y}_{l+1},\hat{y}_{l+2},\cdots,\hat{y}_{l+u})^T$

3.3 性质

1. 其实上述算法都对应于正则化框架：

$$\min_{\mathbf{F}}\frac{1}{2}\left(\sum_{i=1}^N\sum_{j=1}^N w_{i,j}||\frac{1}{\sqrt{d_i}}\vec{\mathbf{F}}_i-\frac{1}{\sqrt{d_j}}\vec{\mathbf{F}}_j||_2^2\right)+\mu\sum_{i=1}^l||\vec{\mathbf{F}}_i-\vec{\mathbf{Y}}_i||_2^2$$

   其中：

   - $\mu>0$ 为正则化参数。当 $\mu=\frac{1-\alpha}{\alpha}$ 时，上式的最优解恰好为 $\mathbf{F}^*=(1-\alpha)(\mathbf{I}-\alpha\mathbf{S})^{-1}\mathbf{Y}$

- 上式第二项：迫使学得结果在有标记样本上的预测与真实标记尽可能相同。
- 上式第一项：迫使相近样本具有相似的标记。

这里的标记既可以是离散的类别，也可以是连续的值。

2. 虽然二类标签传播算法和多类标签传播算法理论上都收敛，而且都知道解析解。但是：
   - 对二类标签传播算法，矩阵求逆 $(\mathbf{I}-\mathbf{P}_{u,u})^{-1}$ 的算法复杂度是 $O(u^3)$ 。如果未标记样本的数量 u 很大，则求逆的过程非常耗时。
   - 对多类标签传播算法，矩阵求逆 $(\mathbf{I}-\alpha\mathbf{S})^{-1}$ 的算法复杂度是 $O(N^3)$ 。如果样本总数量 N 很大，则求逆的过程非常耗时。

   因此标签传播算法一般选择迭代算法来实现。
3. 图半监督学习方法在概念上相当清晰，且易于通过对所涉及矩阵运算的分析来探索算法性质。

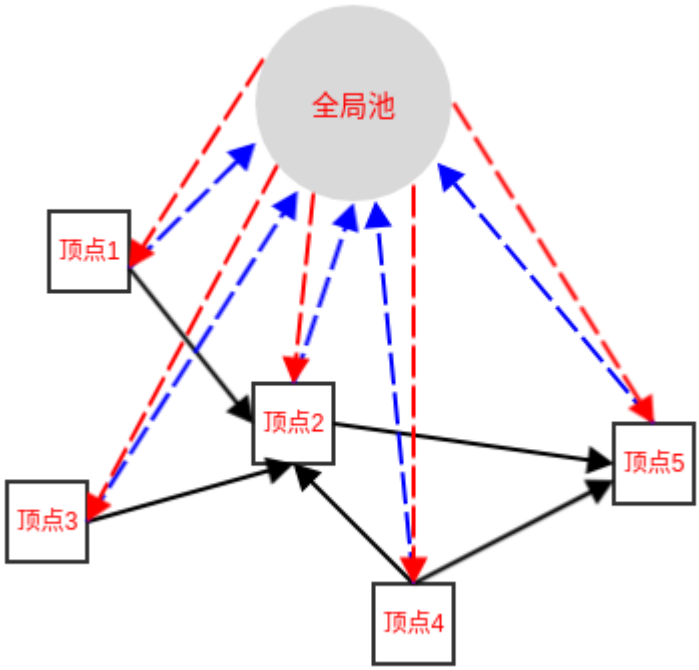
   缺点：
   - 在存储开销上较大，使得此类算法很难直接处理大规模数据。
   - 由于构图过程仅能考虑训练样本集，难以判断新的样本在图中的位置。因此在接收到新样本时，要么将其加入原数据集对图进行重构并重新进行标记传播，要么引入额外的预测机制。

3.4 标签传播与 PageRank

1. `PageRank` 算法用于对网页进行排名。它也是利用能量在有向图 $\mathcal{G}=(\mathbb{V},\mathbb{E})$ 中的流动来获得网页的重要性得分。
   - 每个网页代表图 $\mathcal{G}$ 中的一个顶点，所有顶点的集合为 $\mathbb{V}$ 。
   - 如果存在超链接，该超链接从顶点 i 对应的网页指向顶点 j 对应的网页，则存在一条有向边从顶点 i 指向顶点 j 。

     所有的边的集合为 $\mathbb{E}$ 。
   - 每个顶点都存储有能量，能量对应着网页的重要性得分。
   - 对每个网页，设其能量为 y ：
     - 用户以概率 $1-e$ 选择一个超链接，进入下一个网页。这意味着有 $(1-e)\times y$ 的能量从当前顶点流失，流向下一个网页。
     - 用户以概率 e 随机选择一个网页。这意味着有 $e\times y$ 的能量从当前顶点流失，流向全局能量池。同时有 $\frac{e\times\mathrm{Total}}{N}$ 的能量流入当前顶点，其中 Total 是系统中所有顶点的总能量，N 为顶点数量。

       这是因为每个顶点都有 e 比例的能量流入全局能量池，则全局能量池的流入能量为 $e\times\mathrm{Total}$ 。而全局能量池的能量又平均流入到每个顶点中，则每个顶点从全局能量池得到的能量为 $\frac{e\times\mathrm{Total}}{N}$ 。
   - 当系统取得平衡时，满足以下条件：
     - 全局能量池的流入、流出能量守恒。
     - 每个顶点的流入、流出能量守恒。
     - 系统所有顶点的总能量为常数。



2. 假设 $\text{Total}=1$ ，即系统所有顶点的总能量恒定为 1 。对给定的顶点 i ，假设其能量为 y_i 。
   - 流出能量为： $(1-e)\times y_i+e\times y_i$ 。
   - 流入能量为： $(1-e)\times\sum_{j\neq i}y_jp_{i,j}+\frac{e}{N}$ 。其中 $p_{i,j}$ 为能量从顶点 j 流向顶点 i 的概率。

   则顶点 i 的净入能量为： $(1-e)\times\sum_{j\neq i}y_jp_{j,i}+\frac{e}{N}-y_i$ 。

   考虑所有顶点，令 $\vec{\mathbf{y}}=(y_1,\cdots,y_N)^T$ ， $\mathbf{P}=(p_{i,j})_{N\times N}$ ， $\vec{\mathbf{1}}=(1,1,\cdots,1)^T$ ，则系统每个顶点净流入能量组成的向量为：

$$(1-e)\mathbf{P}\vec{\mathbf{y}}+\frac{e}{N}\vec{\mathbf{1}}-\vec{\mathbf{y}}$$

   当系统稳定时，每个顶点的净流入能量为 0 。因此有： $(1-e)\mathbf{P}\vec{\mathbf{y}}+\frac{e}{N}\vec{\mathbf{1}}-\vec{\mathbf{y}}=\vec{\mathbf{0}}$ 。

3. 考虑到所有顶点的总能量恒定为 1 ，则有 $\sum_i y_i=1$ 。

   定义矩阵 $\mathbf{T}$ 为：

$$\mathbf{T}=\begin{bmatrix}1&1&\cdots&1\\1&1&\cdots&1\\\vdots&\vdots&\ddots&\vdots\\1&1&\cdots&1\end{bmatrix}$$

   则有： $\mathbf{T}\vec{\mathbf{y}}=\vec{\mathbf{1}}$ 。因此有： $\left[(1-e)\mathbf{P}+\frac{e}{N}\mathbf{T}\right]\vec{\mathbf{y}}=\vec{\mathbf{y}}$ 。

   令 $\mathbf{U}=\left[(1-e)\mathbf{P}+\frac{e}{N}\mathbf{T}\right]$ ，则有 $\mathbf{U}\vec{\mathbf{y}}=\vec{\mathbf{y}}$ 。此时的 $\vec{\mathbf{y}}$ 就是对应于 $\mathbf{U}$ 的特征值为 1 的特征向量（经过归一化使得满足的总能量恒定为 1 ）。

3.5 标签传播与社区发现

1. 设图 $\mathcal{G}=(\mathbb{V},\mathbb{E})$ ，社区发现就是在图 $\mathcal{G}$ 中确定 K 个社区 $\mathcal{C}=\{\mathbb{C}_1,\cdots,\mathbb{C}_K\}$ ，其中满足 $\mathbb{C}_1\bigcup\cdots\bigcup\mathbb{C}_K=\mathbb{V}$ 。

若任意两个社区的顶点集合的交集均为空，则称 $\mathcal{C}$ 为非重叠社区，此时等价于聚类。否则称作重叠社区，此时一个顶点可能从属于多个社区。

2. 社区划分的好坏是通过考察当前图的社区划分，与随机图的社区划分的差异来衡量的。
   - 当前图的社区划分：计算当前图的社区结构中，内部顶点的边占所有边的比例 ：$\frac{1}{2M}\sum_{i\in\mathbb{V}}\sum_{j\in\mathbb{V}}w_{i,j}\delta(c_i,c_j)$ 。

     其中：
     - $w_{i,j}$ 表示顶点 i 与顶点 j 之间的边的权重， M 表示图 $\mathcal{G}$ 中所有边的权重之和 $M=\frac{1}{2}\sum_{i\in\mathbb{V}}\sum_{j\in\mathbb{V}}w_{i,j}$。
     - c_i 表示顶点 i 所属的社区， c_j 表示顶点 j 所属的社区。$\delta(\cdot,\cdot)$ 函数定义为：
     $$\delta(x,y)=\begin{cases}0, & x\neq y\\ 1, & x=y\end{cases}$$
     - 因为 $w_{i,j}=w_{j,i}$ ，因此每条边被计算了两次，所以系数为 $2M$ 。

     它可以简化为：$\sum_{k=1}^{K}\frac{m_k}{M}$ ，其中 m_k 表示社区 $\mathbb{C}_k$ 中所有内部边的总权重。
   - 随机图的社区划分：计算随机图的社区结构中，内部顶点的边占所有边的比例的期望。

     随机图是这样生成的：每个顶点的度保持不变，边重新连接。

     记顶点 i 和 j 之间的边的期望权重为 $p_{i,j}$ ，则它满足下列条件：
     - 因为每个顶点的度不变，则最终总度数不变。即：$\sum_{i\in\mathbb{V}}\sum_{j\in\mathbb{V}}p_{i,j}=2M$ 。
     - 对每个顶点，它的度保持不变。即：$\sum_{j\in\mathbb{V}}p_{i,j}=d_i$ 。其中 d_i 表示顶点 i 的度。
     - 随机连边时，一个边的两个顶点的选择都是独立、随机的。因此对于 $p_{i,j}$ ，选到 i 的概率与 d_i 有关，选到 j 的概率与 d_j 有关。根据独立性有：$p_{i,j}=f(d_i)f(d_j)$ ，其中 $f(\cdot)$ 为某个映射函数。

     根据 $\sum_{j\in\mathbb{V}}p_{i,j}=d_i$ ，以及 $\sum_{j\in\mathbb{V}}p_{i,j}=f(d_i)\sum_{j\in\mathbb{V}}f(d_j)$ 有：$d_i=f(d_i)\sum_{j\in\mathbb{V}}f(d_j)$ 。

     由于 $\sum_{j\in\mathbb{V}}f(d_j)$ 不包含 d_i ，因此 $f(d_i)$ 与 d_i 是倍乘关系。不妨假设 $f(d_i)=Td_i$，其中 T 为常量。则有：
     $$\sum_{i\in\mathbb{V}}\sum_{j\in\mathbb{V}}p_{i,j}=\sum_{i\in\mathbb{V}}\sum_{j\in\mathbb{V}}T^2d_i\times d_j=2M$$

     考虑到 $\sum_{i\in\mathbb{V}}d_i=2M$，则有： $(2M)^2=\left(\sum_{i\in\mathbb{V}}d_i\right)^2=\sum_{i\in\mathbb{V}}\sum_{j\in\mathbb{V}}d_i\times d_j$ 。因此有：
     $$T^2\times(2M)^2=2M$$

     因此有：$p_{i,j}=T^2\times d_i\times d_j=\frac{d_i\times d_j}{2M}$ 。

     因此随机图的社区结构中，内部顶点的边占所有边的比例的期望为：
     $$\frac{1}{2M}\sum_{i\in\mathbb{V}}\sum_{j\in\mathbb{V}}p_{i,j}\delta(c_i,c_j)=\frac{1}{2M}\sum_{i\in\mathbb{V}}\sum_{j\in\mathbb{V}}\frac{d_i\times d_j}{2M}\delta(c_i,c_j)$$

3. 定义 `modularity` 指标为 Q ：
   $$Q=\frac{1}{2M}\sum_{i\in\mathbb{V}}\sum_{j\in\mathbb{V}}\left[w_{i,j}-\frac{d_i\times d_j}{2M}\right]\delta(c_i,c_j)$$

   它就是：当前网络中连接社区结构内部顶点的边所占的比例，与另外一个随机网络中连接社区结构内部顶点的便所占比例的期望相减，得到的差值。用于刻画社区划分的好坏。

- 第一项：

$$\frac{1}{2M}\sum_{i\in\mathbb{V}}\sum_{j\in\mathbb{V}}w_{i,j}\delta(c_i,c_j)=\frac{1}{2M}\sum_{k=1}^{K}\sum_{i\in\mathbb{C}_k}\sum_{j\in\mathbb{C}_k}w_{i,j}$$
$$=\sum_{k=1}^{K}\frac{2m_k}{2M}=\sum_{k=1}^{K}\frac{m_k}{M}$$

- 第二项：

$$\frac{1}{2M}\sum_{i\in\mathbb{V}}\sum_{j\in\mathbb{V}}\frac{d_i\times d_j}{2M}\delta(c_i,c_j)=\sum_{k=1}^{K}\sum_{i\in\mathbb{C}_k}\sum_{j\in\mathbb{C}_k}\frac{d_i}{2M}\times\frac{d_j}{2M}$$
$$=\sum_{k=1}^{K}\left(\frac{\sum_{i\in\mathbb{C}_k}d_i}{2M}\right)^2=\sum_{k=1}^{K}\left(\frac{D_k}{2M}\right)^2$$

因此，经过化简之后为：

$$Q=\sum_{k=1}^{K}\left(\frac{m_k}{M}-\left(\frac{D_k}{2M}\right)^2\right)$$

其中：

- m_k 表示社区 $\mathbb{C}_k$ 中所有内部边的总权重 $m_k=\frac{1}{2}\sum_{i\in\mathbb{C}_k}\sum_{j\in\mathbb{C}_k}w_{i,j}$ 。
- D_k 表示社区 $\mathbb{C}_k$ 中所有顶点的度之和 $D_i=\sum_{i\in\mathbb{C}_k}d_i$。
  - $\frac{D_k}{2M}$ 表示社区 $\mathbb{C}_k$ 的内部顶点的度数之和，占总权重的比例。
  - D_k 也可以表示为：$D_k=2m_k+O_k$，其中 O_k 表示社区 $\mathbb{C}_k$ 与其它所有社区之间的边的数量。
- Q 的值在 `(-1,1)` 之间。
  - 当社区之间不存在边的连接时， Q 最大。
  - 当每个点都是一个社区时， Q 最小。

3.5.1 Fast Unfolding

1. `Fast Unfolding` 算法是基于 `modularity` 的社区划分算法。

   它是一种迭代算法，每一步3迭代的目标是：使得划分后整个网络的 `modularity` 不断增大。

2. `Fast Unfolding` 算法主要包括两个阶段：

   - 第一阶段称作 `modularity optimization` ：将每个顶点划分到与其邻接的顶点所在的社区中，以使得 `modularity` 不断增大。

     考虑顶点 i ，设其邻接顶点 j 位于社区 $\mathbb{C}_k$ 中。

     - 未划分之前，顶点 i 是一个独立的社区，它对 Q 的贡献为：$-\left(\frac{d_i}{2M}\right)^2$ 。因为该社区只有一个顶点，因此所有顶点的度之和就是 d_i 。
     - 未划分之前，社区 $\mathbb{C}_k$ 对于 Q 的贡献为：$\frac{m_k}{M}-\left(\frac{D_k}{2M}\right)^2$ 。
     - 划分之后，除了社区 $\mathbb{C}_k$ 与顶点 i 之外，所有社区、顶点在划分前后保持不变，因此 Q 的变化为：

$$\Delta Q=\left[\frac{m_k'}{M}-\left(\frac{D_k'}{2M}\right)^2\right]-\left[\frac{m_k}{M}-\left(\frac{D_k}{2M}\right)^2-\left(\frac{d_i}{2M}\right)^2\right]$$

其中 m'_k, D'_k 分别表示划分之后社区 $\mathbb{C}_k$ 的所有内部边的总权重、所有顶点的度之和。

设顶点 i 与社区 $\mathbb{C}_k$ 相连的边的度数之和为 $d_i^{(k)}$ （可能 i 有多条边与 k 相连），则有：$m'_k = m_k + \frac{1}{2}d_i^{(k)}$ 。

由于顶点 i 现在成为社区 $\mathbb{C}_k$ 的内部顶点，因此 $D'_k = D_k + d_i$。

因此有：

$$\Delta Q = \frac{d_i^{(k)}}{2M} - \frac{2D_k \times d_i}{(2M)^2}$$

- 如果 $\Delta Q > 0$ ，则将顶点 i 划入到社区 $\mathbb{C}_k$ 中。
- 如果 $\Delta Q \leq 0$ ，则顶点 i 保持不变。

- 第二阶段称作 `modularity Aggregation` ：将第一阶段划分出来的社区聚合成一个点，这相当于重新构造网络。

重复上述过程，直到网络中的结构不再改变为止。

3. `Fast Unfolding` 算法：
   - 输入：
     - 数据集 $\mathbb{D} = \{\vec{\mathbf{x}}_1, \cdots, \vec{\mathbf{x}}_N\}$
   - 输出：社区划分 $\mathcal{C} = \{\mathbb{C}_1, \cdots, \mathbb{C}_K\}$ 。
   - 算法步骤：
     - 构建图：根据 $\mathbb{D}$ 构建图 $\mathcal{G} = (\mathbb{V}, \mathbb{E})$ 。
     - 初始化社区：构建 N 个社区，将每个顶点划分到不同的社区中。

       此时每个社区有且仅有一个顶点。
     - 迭代，迭代停止条件为：图保持不变 。迭代步骤为：
       - 遍历每个顶点：

         随机选择与其相连的顶点的社区，考察 ΔQ 。如果 $\Delta Q > 0$ ，则将该顶点划入到该社区中；否则保持不变。
       - 重复上述过程，直到不能再增大 `modularity` 为止。
       - 构造新的图：将旧图中每个社区合并为新图中的每个顶点，旧社区之间的边的总权重合并为新图中的边的权重。

         然后重复执行上述两步。
4. 对于顶点 i ，在考察 ΔQ 时，可以遍历与其相连的所有顶点所属的社区，然后选择最大的那个 ΔQ 。
5. 上述算法是串行执行：逐个选择顶点，重新计算其社区。在这个过程中其它顶点是不能变化的。

   事实上可以将其改造为并行算法，在每一步迭代中同时更新多个顶点的信息。
6. `Fast Unfolding` 算法的结果比较理想，对比 `LPA` 算法会更稳定。另外 `Fast Unfolding` 算法不断合并顶点并构造新图，这会大大减少计算量。

3.5.2 LPA

1. `Usha` 等人于2007年首次将标签传播算法用于社区发现。

   不同于半监督学习，在社区发现中并没有任何样本的标注信息，这使得算法不稳定，可能得到多个不同的社区划分结果。

2. 标签传播算法在迭代过程中，对于顶点 i ，会根据它的邻居顶点更新 i 的社区。更新规则为：i 的邻居顶点中，哪个社区出现最多，则顶点 i 就属于哪个社区。

   如果多个社区出现的次数都是最多的，则随机选择一个。

3. 标签传播算法：

   - 输入：
     - 数据集 $\mathbb{D}=\{\vec{\mathbf{x}}_1,\cdots,\vec{\mathbf{x}}_N\}$
   - 输出：社区划分 $\mathcal{C}=\{\mathbb{C}_1,\cdots,\mathbb{C}_K\}$ 。
   - 算法步骤：
     - 构建图：根据 $\mathbb{D}$ 构建图 $\mathcal{G}=(\mathbb{V},\mathbb{E})$ 。
     - 初始化：为每个顶点指定一个唯一的标签。
     - 迭代，迭代停止条件为社区划分收敛。迭代步骤为：
       - 随机打乱顶点的顺序。
       - 遍历每个顶点 i ，设置顶点 i 的标签为：

         $$c(i)=\arg\max_k\sum_{j\in\mathbb{N}(i)}I(c(j)=k)$$

         其中 $\mathbb{N}(i)$ 表示顶点 i 的邻居顶点集合， $c(i)$ 表示顶点 i 的标签， $I(\cdot)$ 表示示性函数。
       - 判断是否收敛。收敛条件为：遍历每个顶点 i ，满足：

         $$\sum_{j\in\mathbb{N}(i)}I(c(j)=c(i))\geq\sum_{j\in\mathbb{N}(i)}I(c(j)=k),k=1,2,\cdots$$

         即：顶点 i 的邻居顶点集合中，标签为 $c(i)$ 的点的个数最多。

         之所以取 `=` 是因为：可能出现个数一样多的情况，这时也是满足条件的。

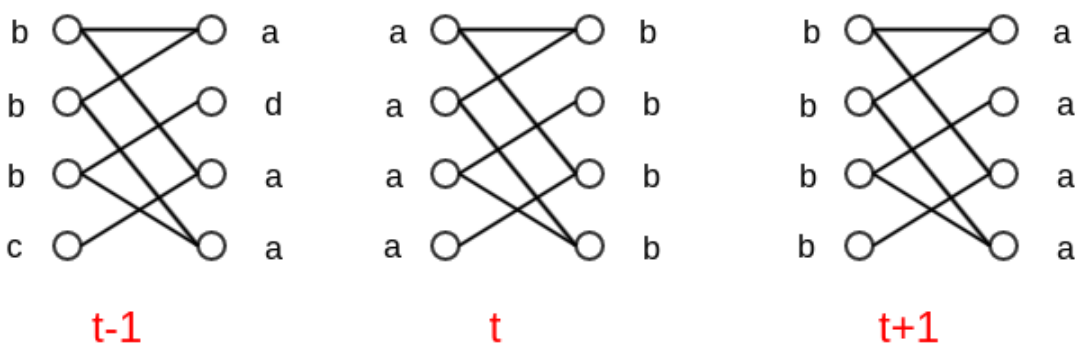
4. 标签传播算法的优点是：简单、高效、快速。缺点是每次迭代结果不稳定，准确率不高。

5. 标签传播算法中，顶点的标签有同步更新和异步更新两种方式。

   - 同步更新（假设更新次数为 t ）：

     $$c(i)^{<t>}=\arg\max_k\sum_{j\in\mathbb{N}(i)}I(c(j)^{<t-1>}=k)$$

     同步更新方法在二分图中可能出现震荡的情况，如下图所示。



   - 异步更新：

$$c(i)^{<t>}=\arg\max_k\sum_{j\in\mathbb{N}(i)}I(c(j)^{<newst>}=k)$$

$c(j)^{<newst>}$ 为顶点 j 的最新的标签值。如果它最近的更新是在第 $t-1$ 轮，则 $c(j)^{<newst>}=c(j)^{<t-1>}$ ；如果它最近的更新是在第 t 轮，则 $c(j)^{<newst>}=c(j)^{<t>}$ 。

异步更新可以保证收敛，因此标签传播算法采用异步的策略更新顶点的标签，并在每次迭代之前对顶点重新进行随机排序从而保证顶点是随机遍历的。

6. 标签传播算法的时间复杂度为 $O(N)$，空间复杂度为 $O(N^2)$ 。因此标签传播算法具有线性的时间复杂度并且可以很好地适应大规模社区的检测。

   它既不需优化预定义的目标函数，也不需要关于社区的数量和规模等先验信息。

3.5.3 SLPA

1. `SLPA` 由 `Jierui Xie` 等人于 2011 年提出，它是一种重叠型社区发现算法。通过设定参数，也可以退化为非重叠型社区发现算法。
2. `SLPA` 与 `LPA` 的重要区别：
   - `SLPA` 引入 `Listener` 和 `Speaker` 的概念。其中 `Listener` 就是待更新标签的顶点， `Speaker` 就是该顶点的邻居顶点集合。

     在 `LPA` 中， `Listener` 的标签由 `Speaker` 中出现最多的标签决定。而 `SLPA` 中引入了更多的规则。
   - `SLPA` 会记录每个顶点的历史标签序列。假设更新了 T 次，则每个顶点会保存一个长度为 T 的序列。

     当迭代停止之后，对每个顶点的历史标签序列中各标签出现的频率作出统计，按照某一给定的阈值 r 过滤掉那些频率小的标签，剩下的就是该顶点的标签，通常会有多个标签。
3. `SLPA` 算法：
   - 输入：
     - 数据集 $\mathbb{D}=\{\vec{\mathbf{x}}_1,\cdots,\vec{\mathbf{x}}_N\}$
     - 迭代步数 T
     - 标签频率阈值 r
     - `Speaker rule`
     - `Listener rule`
   - 输出：社区划分 $\mathcal{C}=\{\mathbb{C}_1,\cdots,\mathbb{C}_K\}$ 。
   - 算法步骤：
     - 构建图：根据 $\mathbb{D}$ 构建图 $\mathcal{G}=(\mathbb{V},\mathbb{E})$ 。
     - 初始化：为每个顶点分配一个标签队列；为每个顶点指定一个唯一的标签，并将标签加入到该顶点的标签队列中。
     - 迭代 T 步。迭代步骤为：
       - 随机打乱顶点的顺序。
       - 遍历每个顶点 i ，将顶点 i 设置为 `Listener` ， 将顶点 i 的邻居顶点都设置为 `Speaker` 。
         - `Speaker` 根据 `Speaker rule` 向 `Listener` 发送消息。

           `Speaker rule` 为发送规则，如： `Speaker` 从它的标签队列中发送最新的标签、或者 `Speaker` 从它的标签队列中随机选择一个标签发送（标签选择的概率等于标签队列中各标签出现的频率）。

- `Listener` 接收消息，并根据 `Listener rule` 更新标签，并将最新的标签加入到它的标签队列中。

  `Listener rule` 为接收规则。如：`Listener` 选择接受的标签中出现最多的那个作为最新的标签。

- 遍历每个顶点 i ，将顶点 i 的标签队列中，标签出现频率低于 r 的标签丢弃。标签队列中剩下的标签就是顶点 i 的标签（可能有多个）。

四、基于分歧的方法

1. 基于分歧的方法 `disagreement-based methods` 使用多个学习器，学习器之间的分歧 `disagreement` 对未标记数据的利用至关重要。

   协同训练 `co-traning` 是此类方法的重要代表。它最初是针对多视图 `multi-view` 数据设计的，因此也被视作多视图学习 `multi-view learning` 的代表。

4.1 数据视图

1. 在不少现实应用中，一个数据对象往往同时拥有多个属性集 `attribute-set` 。每个属性集就构成了一个视图。
2. 假设数据集为 $\mathbb{D}=\{(\vec{\mathbf{x}}_1,y_1),(\vec{\mathbf{x}}_2,y_2),\cdots,(\vec{\mathbf{x}}_N,y_N)\}$ ，其中 $\vec{\mathbf{x}}_i=(x_{i,1},x_{i,2},\cdots,x_{i,d})^T$ 。属性集合 $\mathbf{x}=\{x_1,x_2,\cdots,x_d\}$。假设属性集合划分为 $\mathbf{x}=\mathbf{x}^{(1)}\bigcup\mathbf{x}^{(2)}$ ，其中：

   $$\mathbf{x}^{(1)}=\{x_1,x_2,\cdots,x_{d_1}\},\quad \mathbf{x}^{(2)}=\{x_{d_1+1},x_{d_1+2},\cdots,x_d\}$$

   即将属性划分为两个属性集，前 d_1 个属性为第一个属性集，后 $d-d_1$ 个属性属于第二个属性集。

   - 原始样本 $\vec{\mathbf{x}}_i$ 被划分为两个属性向量 $<\vec{\mathbf{x}}_i^{(1)},\vec{\mathbf{x}}_i^{(2)}>$ ，它们分别属性这两个属性集。
   - $(<\vec{\mathbf{x}}_i^{(1)},\vec{\mathbf{x}}_i^{(2)}>,y_i)$ 这样的数据就是多视图数据，每个属性集都是一个视图。

   如：`<身高、体重、年龄、学历、爱好、工作>` 可以划分为两个属性集：`<身高、体重、年龄>` 以及 `<学历、爱好、工作>` 。

3. 假设不同视图具有相容性 `compatibility` ：即其所包含的关于输出空间 $\mathcal{Y}$ 的信息是一致的。

   令 $\mathcal{Y}^1$ 为从第一个属性集判别的标记空间， $\mathcal{Y}^2$ 为从第二个属性集判别的标记空间，则有 $\mathcal{Y}=\mathcal{Y}^1=\mathcal{Y}^2$ 。

   注意：这里仅要求标记空间相同，并没有要求每个标记相同。

4.2 协同训练算法

1. 协同训练充分利用了多视图的相容互补性。
2. 假设数据拥有两个充分且条件独立视图。
   - 充分：指每个视图都包含足以产生最优学习器的信息。
   - 条件独立：指在给定类别标记条件下，两个视图独立。

   此时，可以用一个简单的办法来利用未标记数据：
   - 首先在每个视图上，基于有标记样本，分别训练出一个分类器。
   - 然后让每个分类器分别去挑选自己最有把握的未标记样本赋予伪标记，并将伪标记样本提供给另一个分类器作为新增的有标记样本用于训练更新。
   - 该“ 互相学习、共同进步”的过程不断迭代进行，直到两个分类器都不再发生变化，或者到达指定的迭代轮数为止。

注意：

- 如果在每轮学习中都考察分类器在所有未标记样本上的分类置信度，则会有很大的计算开销。因此在算法中使用了未标记样本缓冲池。
- 分类置信度的估计因基学习算法而异。

3. 协同训练算法：
   - 输入：
     - 有标记样本集 $\mathbb{D}_l=\{(<\vec{\mathbf{x}}_1^{(1)},\vec{\mathbf{x}}_1^{(2)}>,y_1),(<\vec{\mathbf{x}}_2^{(1)},\vec{\mathbf{x}}_2^{(2)}>,y_2),\cdots,(<\vec{\mathbf{x}}_l^{(1)},\vec{\mathbf{x}}_l^{(2)}>,y_l)\}$
     - 未标记样本集 $\mathbb{D}_u=\{<\vec{\mathbf{x}}_{l+1}^{(1)},\vec{\mathbf{x}}_{l+1}^{(2)}>,<\vec{\mathbf{x}}_{l+2}^{(1)},\vec{\mathbf{x}}_{l+2}^{(2)}>,\cdots,<\vec{\mathbf{x}}_{l+u}^{(1)},\vec{\mathbf{x}}_{l+u}^{(2)}>\},l+u=N$
     - 缓冲池大小 s
     - 每轮挑选的正例数量 p
     - 每轮挑选的反例数量 n
     - 基学习算法 f
     - 学习轮数 T
   - 输出：未标记样本的预测结果
     $\hat{\tilde{\mathbf{y}}}=(\hat{y}_{l+1},\hat{y}_{l+2},\cdots,\hat{y}_{l+u})^T,\hat{y}_i\in\{1,2,\cdots,K\},i=l+1,l+2,\cdots,l+u$
   - 步骤：
     - 从 $\mathbb{D}_u$ 中随机抽取 s 个样本构建缓冲池 $\mathbb{D}_s$
     - $\mathbb{D}_u=\mathbb{D}_u-\mathbb{D}_s$
     - 从 $\mathbb{D}_l$ 中分别构建：$\mathbb{D}^{(1)},\mathbb{D}^{(2)}$ 分别表示第一视图有标记数据集、第二视图有标记数据集：
       $$\mathbb{D}_l^{(1)}=\{(\vec{\mathbf{x}}_1^{(1)},y_1),(\vec{\mathbf{x}}_2^{(1)},y_2),\cdots,(\vec{\mathbf{x}}_l^{(1)},y_l)\}$$
       $$\mathbb{D}_l^{(2)}=\{(\vec{\mathbf{x}}_1^{(2)},y_1),(\vec{\mathbf{x}}_2^{(2)},y_2),\cdots,(\vec{\mathbf{x}}_l^{(2)},y_l)\}$$
     - 开始迭代，迭代终止条件是：迭代收敛或者迭代次数达到 T 。迭代过程为：
       - 从 $\mathbb{D}_s$ 中分别构建（其中 m 为其元素个数）：$\mathbb{D}_s^{(1)},\mathbb{D}_s^{(2)}$ 分别表示第一视图缓冲数据集、第二视图缓冲数据集：
         $$\mathbb{D}_s^{(1)}=\{\vec{\mathbf{x}}_{s1}^{(1)},\vec{\mathbf{x}}_{s2}^{(1)},\cdots,\vec{\mathbf{x}}_{sm}^{(1)}\}$$
         $$\mathbb{D}_s^{(2)}=\{\vec{\mathbf{x}}_{s1}^{(2)},\vec{\mathbf{x}}_{s2}^{(2)},\cdots,\vec{\mathbf{x}}_{sm}^{(2)}\}$$
       - 考察视图一：
         - 利用 $\mathbb{D}_l^{(1)}$ 训练 f 得到分类器 f_1 。
         - 然后考察 f_1 在 $\mathbb{D}_s^{(1)}$ 上的分类置信度。挑选 p 个正例置信度最高的样本 $\mathbb{D}_p^{(1)}\subset\mathbb{D}_s$、挑选 n 个反例置信度最高的样本 $\mathbb{D}_n^{(1)}\subset\mathbb{D}_s$ 。
         - 然后将 $\mathbb{D}_p^{(1)}$ 的视图二标记为正例，将 $\mathbb{D}_n^{(1)}$ 的视图二标记为反例：
           $$\tilde{\mathbb{D}}_p^{(2)}=\{(\vec{\mathbf{x}}_i^{(2)},1)\mid\vec{\mathbf{x}}_i^{(1)}\in\mathbb{D}_p^{(1)}\}$$
           $$\tilde{\mathbb{D}}_n^{(2)}=\{(\vec{\mathbf{x}}_i^{(2)},-1)\mid\vec{\mathbf{x}}_i^{(1)}\in\mathbb{D}_n^{(1)}\}$$

           > 这里并没有简单的将 $\mathbb{D}_p^{(1)}$ 标记为正例、 $\mathbb{D}_n^{(1)}$ 标记为反例。而是标记它们对立的那个视图，帮助对方视图增加标记数据。
       - $\mathbb{D}_s=\mathbb{D}_s-(\mathbb{D}_p^{(1)}\bigcup\mathbb{D}_n^{(1)})$ ，更新 $\mathbb{D}_s^{(2)}$ 。此时 $\mathbb{D}_s^{(2)}$ 会缩小，因为有部分样本从视图一中获得了标记信息。

- 考察视图二：
  - 利用 $\mathbb{D}_l^{(2)}$ 训练 f 得到分类器 f_2 。
  - 然后考察 f_2 在 上 $\mathbb{D}_s^{(2)}$ 的分类置信度。挑选 p 个正例置信度最高的样本 $\mathbb{D}_p^{(2)} \subset \mathbb{D}_s$、挑选 n 个反例置信度最高的样本 $\mathbb{D}_n^{(2)} \subset \mathbb{D}_s$ 。
  - 然后将 $\mathbb{D}_p^{(2)}$ 的视图一标记为正例，将 D_n^2 的视图一标记为反例：

$$\tilde{\mathbb{D}}_p^{(1)} = \{(\vec{\mathbf{x}}_i^{(1)}, 1) \mid \vec{\mathbf{x}}_i^{(2)} \in \mathbb{D}_p^{(2)}\}$$
$$\tilde{\mathbb{D}}_n^{(1)} = \{(\vec{\mathbf{x}}_i^{(1)}, -1) \mid \vec{\mathbf{x}}_i^{(2)} \in \mathbb{D}_n^{(2)}\}$$

- $\mathbb{D}_s = \mathbb{D}_s - (\mathbb{D}_p^{(2)} \bigcup \mathbb{D}_n^{(2)})$ 。
- 如果 f_1, f_2 在训练前后，均未发生改变，则迭代收敛。否则继续向下执行。

> 如何判断是否发生改变？通常可以考察它们的预测结果是否一致

- 更新 $\mathbb{D}_l^{(1)}, \mathbb{D}_l^{(2)}$ ：

$$\mathbb{D}_l^{(1)} = \mathbb{D}_l^{(1)} \bigcup (\tilde{\mathbb{D}}_p^{(1)} \bigcup \tilde{\mathbb{D}}_n^{(1)})$$
$$\mathbb{D}_l^{(2)} = \mathbb{D}_l^{(2)} \bigcup (\tilde{\mathbb{D}}_p^{(2)} \bigcup \tilde{\mathbb{D}}_n^{(2)})$$

- 补充 $\mathbb{D}_s$ ：从 $\mathbb{D}_u$ 中随机抽取 $2p+2n$ 个样本加入 $\mathbb{D}_s$ ，同时 $\mathbb{D}_u$ 中移除这 $2p+2n$ 个样本。

  > 这意味着：$\mathbb{D}_s$ 越来越大、$\mathbb{D}_u$ 越来越小。

- 最终得到分类器 f_1, f_2。将这两个分类器用于 $\mathbb{D}_u$ 即可得到未标记样本的预测结果：$\hat{\mathbf{y}} = (\hat{y}_{l+1}, \hat{y}_{l+2}, \cdots, \hat{y}_{l+u})^T$ 。

4.3 性质

1. 协同训练过程虽然简单，但是理论证明：若两个视图充分且条件独立，则可利用未标记样本通过协同训练将弱分类器的泛化性能提升到任意高。
   - 不过视图的条件独立性在现实任务中通常很难满足，因此性能提升幅度没有那么大。
   - 但研究表明，即便在更弱的条件下，协同训练仍可以有效提升弱分类器的性能。
2. 协同训练算法本身是为多试图数据而设计的，但此后出现了一些能在单视图数据上使用的变体算法。
   - 它们或是使用不同的学习算法，或是使用不同的数据采样，甚至使用不同的参数设置来产生不同的学习器，也能有效利用未标记数据来提升性能。
   - 后续理论研究表明，此类算法事实上无需数据拥有多试图，仅需弱学习器之间具有显著的分歧（或者差异），即可通过相互提供伪标记样本的方式来提升泛化性能。

     而不同视图、不同算法、不同数据采样、不同参数设置等，都是产生差异的渠道，而不是必备条件。
3. 基于分歧的方法只需要采用合适的基学习器，就能较少受到模型假设、损失函数非凸性和数据规模问题的影响，学习方法简单有效、理论基础相对坚实、使用范围较为广泛。
   - 为了使用此类方法，需要能生成具有显著分歧、性能尚可的多个学习器。
   - 但当有标记样本较少，尤其是数据不具有多试图时，要做到这一点并不容易，需要巧妙的设计。

五、半监督聚类

1. 聚类是一种典型的无监督学习任务，然而在现实聚类任务中，往往能够获取一些额外的监督信息。

于是可以通过半监督聚类`semi-supervised clustering`来利用监督信息以获取更好的聚类效果。

2. 聚类任务中获得的监督信息大致有两种类型：
   - 第一类是必连`must-link`与勿连`cannot-link`约束：
     - 必连：指样本必属于同一个簇。
     - 勿连：指样本必不属于同一个簇。
   - 第二类是：少量的有标记样本。

5.1 约束 k 均值算法

1. 约束 k 均值算法`Constrained-k-means`是利用第一类监督信息的代表。
2. 给定样本集 $\mathbb{D}=\{\vec{\mathbf{x}}_1,\vec{\mathbf{x}}_2,\cdots,\vec{\mathbf{x}}_N\}$， 以及必连关系集合 $\mathcal{M}$ 和勿连关系集合 $\mathcal{C}$ ，则：
   - $(\vec{\mathbf{x}}_i,\vec{\mathbf{x}}_j)\in\mathcal{M}$ ， 表示 $\vec{\mathbf{x}}_i$ 与 $\vec{\mathbf{x}}_j$ 必属于同簇。
   - $(\vec{\mathbf{x}}_i,\vec{\mathbf{x}}_j)\in\mathcal{C}$ ， 表示 $\vec{\mathbf{x}}_i$ 与 $\vec{\mathbf{x}}_j$ 必不属于同簇。

   约束 k 均值算法是 k 均值算法的扩展，它在聚类过程中要确保 $\mathcal{M}$ 与 $\mathcal{C}$ 中的约束得以满足，否则将返回错误提示。
3. 约束 k 均值算法：
   - 输入：
     - $\mathbb{D}=\{\vec{\mathbf{x}}_1,\vec{\mathbf{x}}_2,\cdots,\vec{\mathbf{x}}_N\}$
     - 必连关系集合 $\mathcal{M}$
     - 勿连关系集合 $\mathcal{C}$
     - 聚类簇数 K
   - 输出：聚类簇划分 $\{\mathbb{C}_1,\mathbb{C}_2,\cdots,\mathbb{C}_K\}$
   - 算法步骤：
     - 从 $\mathbb{D}$ 中随机选取 K 个样本作为初始均值向量 $\{\vec{\mu}_1,\vec{\mu}_2,\cdots,\vec{\mu}_K\}$
     - 迭代：
       - 初始化每个簇 ：$\mathbb{C}_k=\phi,k=1,2,\cdots,K$
       - 对每个样本 $\vec{\mathbf{x}}_i,i=1,2,\cdots,N$， 执行下列操作：
         - 初始化可选的簇的集合为 $\mathcal{K}=\{1,2,\cdots,K\}$
         - 计算 $\vec{\mathbf{x}}_i$ 与各均值向量的距离 $d_{i,k}=||\vec{\mathbf{x}}_i-\vec{\mu}_k||_2^2,k=1,2,\cdots,K$
         - 找出 $k\in\mathcal{K}$ 且 $d_{i,k}$ 最小的 k，设此时 $k=k^*$，即 d_{i,k^*} 最小，则考察将 $\vec{\mathbf{x}}_i$ 划入簇 $\mathbb{C}_{k^*}$ 是否会违背 $\mathcal{M}$ 与 $\mathcal{C}$ 中的约束：
           - 若不违背，则 $\vec{\mathbf{x}}_i$ 划入簇 $\mathbb{C}_{k^*}$ ： $\mathbb{C}_{k^*}=\mathbb{C}_{k^*}\bigcup\{\vec{\mathbf{x}}_i\}$
           - 若违背，则从 $\mathcal{K}$ 中移除 k^*， 继续找出 $k\in\mathcal{K}$ 且 $d_{i,k}$ 最小的 k。
           - 重复上述考察，直到 $\mathcal{K}=\phi$ 或者 $\vec{\mathbf{x}}_i$ 划入簇 $\mathbb{C}_{k^*}$ 不会违背 $\mathcal{M}$ 与 $\mathcal{C}$ 中的约束。

             > 如果 $\mathcal{K}=\phi$ ，则返回错误提示。这意味着 $\vec{\mathbf{x}}_i$ 与所有的簇都发生冲突。
       - 更新均值向量：

$$\vec{\mu}_k=\frac{1}{|\mathbb{C}_k|}\sum_{\vec{\mathbf{x}}_j\in\mathbb{C}_k}\vec{\mathbf{x}}_j,\quad k=1,2,\cdots,K$$

        - 若更新均值向量前后，均值向量变化很小，则迭代结束。
    - 根据最新的均值向量划分 $\mathbb{D}$， 获得聚类簇划分 $\{\mathbb{C}_1,\mathbb{C}_2,\cdots,\mathbb{C}_K\}$ 。

5.2 约束种子 k 均值算法

1. 约束种子 k 均值`Constrained Seed k-means`算法是利用第二类监督的代表。
2. 给定样本集 $\mathbb{D}=\{\vec{\mathbf{x}}_1,\vec{\mathbf{x}}_2,\cdots,\vec{\mathbf{x}}_N\}$ 。 假定少量的有标记样本为 $\mathbb{S}=\bigcup_{k=1}^K\mathbb{S}_k\subset\mathbb{D}$， $\mathbb{S}_k\neq\phi$ 为隶属于第 k 个聚簇的样本。

   直接将 $\mathbb{S}$ 中的样本作为种子，用它们初始化 K 均值算法的 K 个聚类中心，并且在聚类簇迭代更新过程中不改变种子样本的簇隶属关系，这样就得到了约束种子 k 均值算法。
3. 约束种子 k 均值算法：
    - 输入：
        - $\mathbb{D}=\{\vec{\mathbf{x}}_1,\vec{\mathbf{x}}_2,\cdots,\vec{\mathbf{x}}_N\}$
        - 少量有标记样本 $\mathbb{S}=\bigcup_{k=1}^K\mathbb{S}_k\subset\mathbb{D}$
        - 聚类簇数 K
    - 输出：聚类簇划分 $\{\mathbb{C}_1,\mathbb{C}_2,\cdots,\mathbb{C}_K\}$
    - 算法步骤：
        - 利用有标记样本集合 $\mathbb{S}$ 初始化均值向量：

$$\vec{\mu}_k=\frac{1}{|\mathbb{S}_k|}\sum_{\vec{\mathbf{x}}_i\in\mathbb{S}_k}\vec{\mathbf{x}}_i,\quad k=1,2,\cdots,K$$

        - 迭代：
            - 初始化每个簇：$\mathbb{C}_k=\phi,k=1,2,\cdots,K$
            - 将有标记样本集合 $\mathbb{S}$ 中的每个样本 $\vec{\mathbf{x}}_i$ 分别添加到对应的簇中 $\mathbb{C}_k=\mathbb{C}_k\bigcup\mathbb{S}_k$ 。
            - 对未标记样本集合 $\mathbb{D}-\mathbb{S}$ 中的每个样本 $\vec{\mathbf{x}}_i$，将它划分为距离该样本最近的簇中。其中的距离是样本和簇均值向量的距离。
            - 更新簇均值向量：

$$\vec{\mu}_k=\frac{1}{|\mathbb{C}_k|}\sum_{\vec{\mathbf{x}}_j\in\mathbb{C}_k}\vec{\mathbf{x}}_j,\quad k=1,2,\cdots,K$$

            - 若更新均值向量前后，均值向量变化很小，则迭代结束。
        - 根据最新的均值向量划分 $\mathbb{D}$， 获得聚类簇划分 $\{\mathbb{C}_1,\mathbb{C}_2,\cdots,\mathbb{C}_K\}$ 。

六、 总结

1. 各种半监督学习算法的比较：
    - 生成式半监督学习方法需要充分可靠的领域知识才能确保模型不至于太坏。
    - 非监督`SVM`目标函数非凸，因此有不少工作致力于减轻非凸性造成的不利影响。
    - 图半监督学习方法，图的质量极为重要。
    - 基于分歧的方法将集成学习与半监督学习联系起来。
2. 半监督学习在利用未标记样本后并非必然提升泛化性能，在有些情况下甚至会导致性能下降。
    - 对生成式方法，原因通常是模型假设不准确。因此需要依赖充分可靠的领域知识来设计模型。

- 对半监督`SVM`，原因通常是训练数据中存在多个“低密度划分”，而学习算法可能做出不利的选择。

  `S4VM`通过优化最坏情况下性能来综合利用多个低密度划分，提升了此类技术的安全性。
- 更一般的安全半监督学习仍然是未决的难题。

  安全是指：利用未标记样本后，能确保返回性能至少不差于仅利用有标记样本。