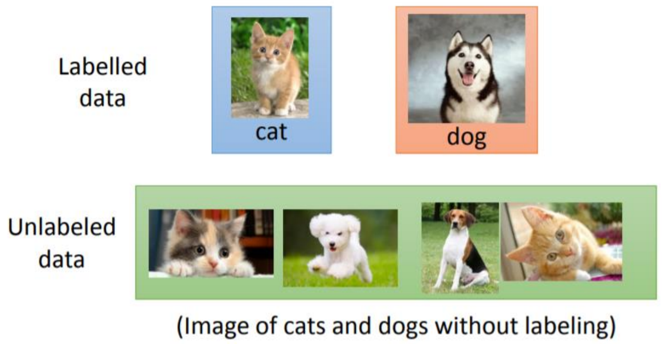
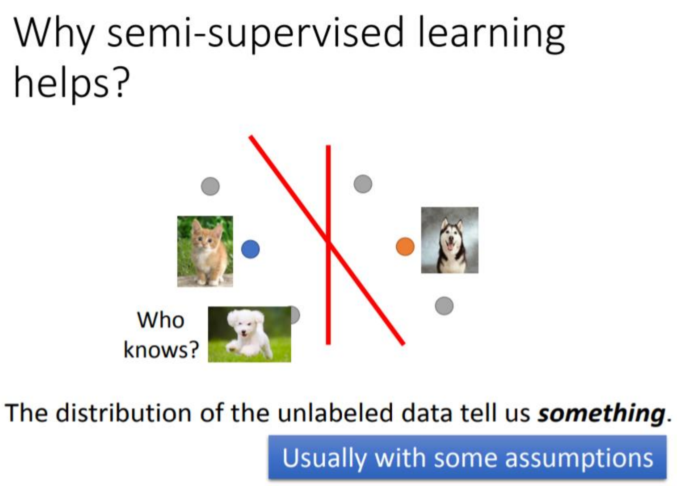
Semi-supervised

Introduction

- Supervised learning:$\{(x^r,\hat{y}^r)\}_{r=1}^{R}$
  例如：x^r: image, $\hat{y}^r$: class labels
- Semi-supervised learning: $\{(x^r,\hat{y}^r)\}_{r=1}^{R}, \{(x^u)\}_{u=R}^{R+U}$
  A set of unlabeled data, usually U>>R
  Transductive learning: unlabeled data is the testing data
  Inductive learning: unlabeled data is not the testing data
- Why semi-supervised learning?
  Collecting data is easy, but collecting “labelled” data is expensive
  We do semi-supervised learning in our lives

对于猫狗分类问题，如果只有一部分 data 有 label，还有其他很大一部分 data 是 unlabeled，那么我们可以认为 unlabeled data 对我们网络的训练是无用的吗?





A: 如图所示，图中灰色圆点表示 unlabeled data，其他圆点表示 labeled data， 如果没有 unlabeled data， 此时可以用一条竖直的线将猫狗进行分类，boundary 为竖直的那条线；但 unlabeled data 的分布也可以告诉我们一些信息，对我们的训练也是有帮助的，有了 unlabeled data，此时的 boundary 为斜直线

Semi-supervised Learning for Generative Model

Intuitive

不考虑 unlabeled data，只要 labeled data

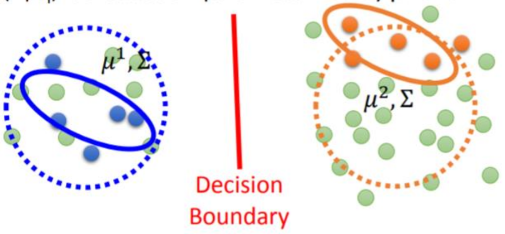
- Given labelled training examples $x^r \in C_1, C_2$
  - looking for most likely prior probability $P(C_i)$ and class-dependent probability $P(x|C_i)$
  - $P(x|C_i)$ is a Gaussian parameterized by μ^i and Σ

With $P(C_1), P(C_2), \mu^1, \mu^2, \Sigma$

$$P(C_1|x) = \frac{P(x|C_1)P(C_1)}{P(x|C_1)P(C_1) + P(x|C_2)P(C_2)}$$

如果把 unlabeled data 也考虑进来，此时的 boundary 也发生了变化

- Given labelled training examples $x^r \in C_1, C_2$
  - looking for most likely prior probability $P(C_i)$ and class-dependent probability $P(x|C_i)$
  - $P(x|C_i)$ is a Gaussian parameterized by μ^i and Σ



The unlabeled data x^u help re-estimate $P(C_1), P(C_2), \mu^1, \mu^2, \Sigma$

公式：

- Initialization: $\theta = \{P(C_1), P(C_2), \mu^1, \mu^2, \Sigma\}$

E
- Step 1: compute the posterior probability of unlabeled data

  $P_\theta(C_1|x^u)$ Depending on model θ

Back to step 1

M
- Step 2: update model

$$P(C_1) = \frac{N_1 + \sum_{x^u} P(C_1|x^u)}{N}$$

N: total number of examples
N_1: number of examples belonging to C_1

$$\mu^1 = \frac{1}{N_1}\sum_{x^r \in C_1} x^r + \frac{1}{\sum_{x^u} P(C_1|x^u)}\sum_{x^u} P(C_1|x^u)x^u \quad \ldots\ldots$$

Why?

$$\theta = \{P(C_1), P(C_2), \mu^1, \mu^2, \Sigma\}$$

- Maximum likelihood with labelled data Closed-form solution

$$logL(\theta) = \sum_{x^r} logP_\theta(x^r, \hat{y}^r)$$

$$\boxed{P_\theta(x^r, \hat{y}^r) = P_\theta(x^r|\hat{y}^r)P(\hat{y}^r)}$$

- Maximum likelihood with labelled + unlabeled data

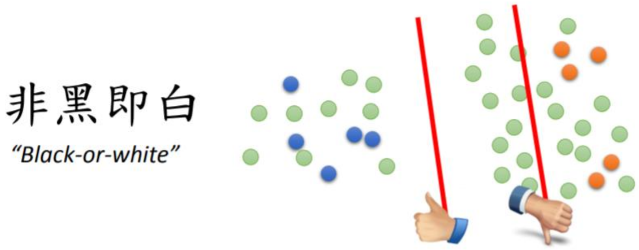
$$logL(\theta) = \sum_{x^r} logP_\theta(x^r, \hat{y}^r) + \sum_{x^u} logP_\theta(x^u)$$

Solved iteratively

$$\boxed{P_\theta(x^u) = P_\theta(x^u|C_1)P(C_1) + P_\theta(x^u|C_2)P(C_2)}$$

(x^u can come from either C_1 and C_2)

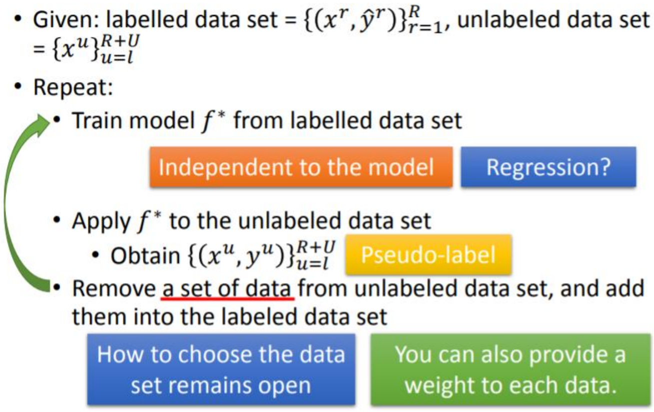
Low-density Separation



Self-training

有 labeled data 和 unlabeled data，重复以下过程:

- 从 labeled data 中 train 了模型f^*;
- 将f^*应用到 unlabeled data，得到带 label 的数据，称为 Pseudo-label
- 从 unlabeled data 中移出这部分 data，并加入 labeled data；要移出哪部分 data，要根据具体的限制条件而定
- 有了更多的 label data，就可以继续训练我们的模型，返回第一步

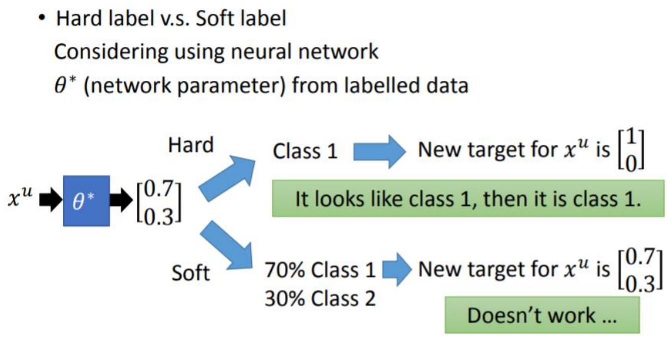


Q：这种训练方式对 regression 有用吗?

W: 不能，regression 输出的是一个真实的值

Hard label vs soft label

Self-training 用的是 hard label；generative mode 用的是 soft label



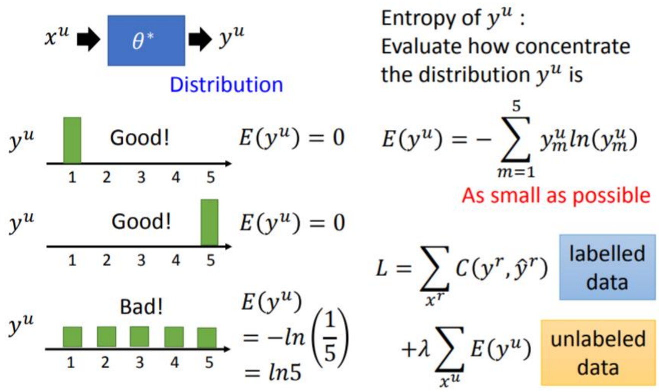
Entropy-based Regularization

如果输出的每个类别的概率是相近的，那么这个模型就比较 bad，输出的类别差距很大，比如某个类别的概率为 1，其他都是 0；我么可以用$E(y^u)$来衡量

$$E(y^u) = -\sum_{m=1}^{5} y_m^u \, ln(y_m^u)$$

对于第一个和第二个 distribution，那么$E(y^u) = 0$；

对于第三个 distribution，那么$E(y^u) = -ln\left(\frac{1}{5}\right) = ln5$

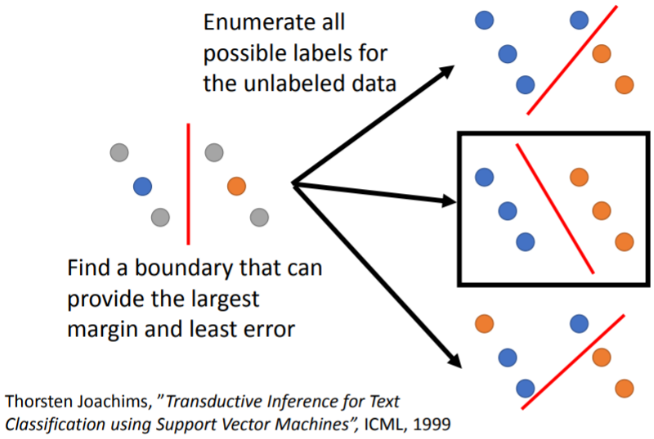


那么我们现在就可以重新设计 loss function，用 cross entropy 来估计$y^r, \hat{y}^r$之间的差距，即$C(y^r, \hat{y}^r)$，使用 labeled data，还加上了一个 regularization term

$$L = \sum_{x^r} C(y^r, \hat{y}^r) + \lambda \sum_{x^u} E(y^u)$$

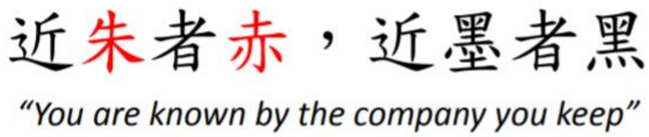
Outlook: Semi-supervised SVM

对于 unlabeled data，如果是 SVM 二分类问题，可以把所有的 unlabeled data 都穷举为 Class1 或 Class2，列举出所有可能的方案，再找出对应的 boundary，计算 loss，可以发现下图中黑色框图具有最小的 loss



Smoothness Assumption

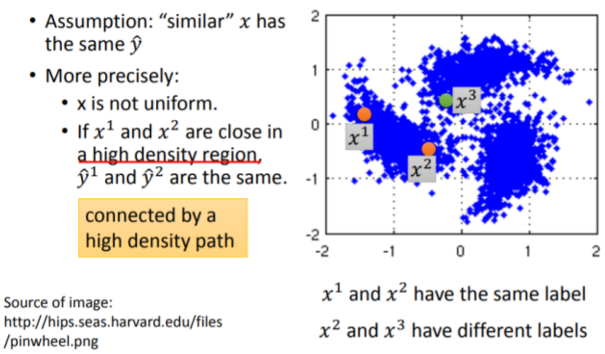
Introduction



假设：如果x是 similar，那么他们的 y 也是一样的

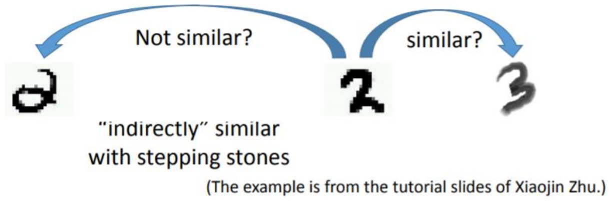
这样的假设是非常不精确的，下面我们做出一个更加精彩的假设：

- x是分布不均匀的，有的地方很密集，有的地方很稀疏
- x^1, x^2中间有个 high density region，那么 label y^1, y^2就可能是一样的；但x^2, x^3中间没有 high density region，其 label 相同的概率就非常小

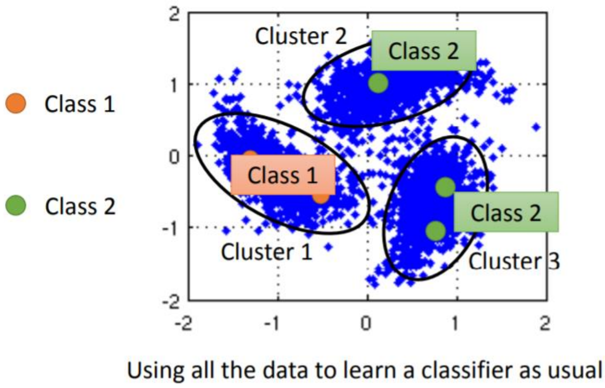


对于下图中的数字，2 之间是有过渡形态的，所以这两个图片是 similar 的；而 2 与 3 之间没有过渡形态，因此是

similar 的



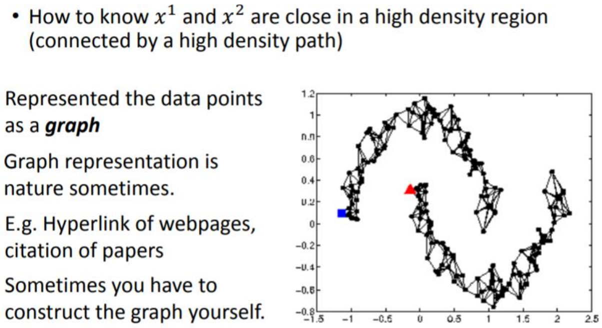
比较直观的做法是先进行 cluster，再进行 label



Graph-based Approach

那么我们到底要怎么才能知道x^1,x^2到底在 high density region 是不是 close 呢?

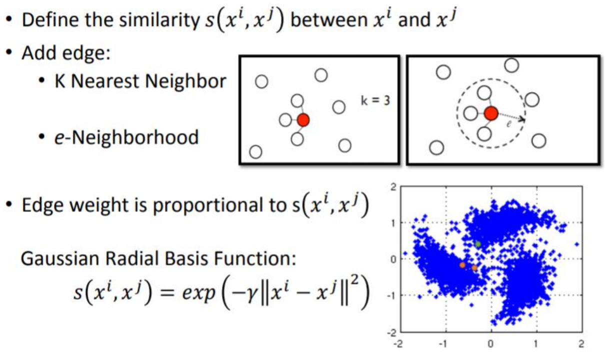
我们可以把 data point 用图来表示，图的表示有时是比较 nature，有时需要我们自己找出来 point 之间的联系



Graph Construction

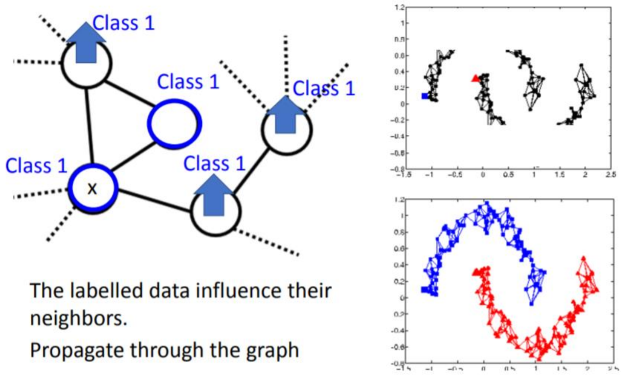
首先定义不同 point 之间的相似度$s(x^i,x^j)$,可以通过以下两个算法来添加 edge:

- KNN,对于图中红色的圆点，与其最相近的三个(k=3)neighbor 相连接
- e-Neighborhood，对于周围的 neighbor，只有和他相似度大于 1 的才会连接起来



edge 并不是只有相连和不相连两种选择而已，也可以给 edge 一些 weight，让这个 weight 和这两个 point 之间的相似度成正比

labeled data 会影响他的邻居，如果这个 point 是 class1，那么他周围的某些 point 也可能是 class1

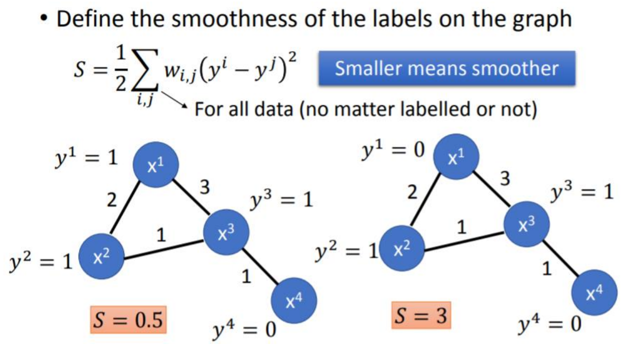


Definition

对于下图中的两幅图，如果从直观上看，我们可以认为左边的图更 smooth 现在我们用数字来定量描述，S 的的定义如下

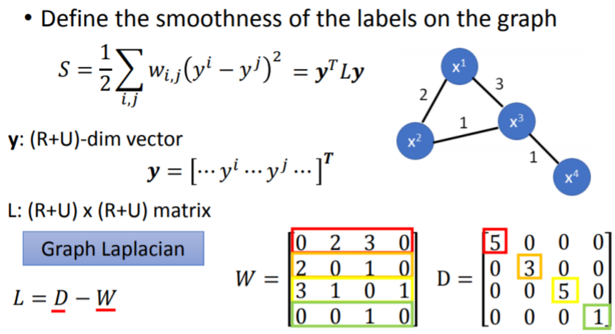
$$S=\frac{1}{2}\sum_{i,j} w_{i,j}\left(y^i-y^j\right)^2$$

根据公式我们可以算出左图的 S=0.5，右图的 S=3，值越小越 smooth，越小越好



对原来的 S 进行改造一下，$S=y^TLy$

其中$L=D-W$，w 为权重矩阵，D 表示将 weight 每行的和放到对角线的位置



loss function 其中一项就包括 cross entropy 计算的 loss；smoothness 的量 S，前面再乘上一个可以调整的参数λ，λS就表示一个 regularization term

网络的整体目标是使 loss function 取得最小值，即 cross entropy 项和 smoothness 都必须要达到最小值，和其他的网络一样，计算相应的 gradient，做 gradient descent 即可

如果要计算 smoothness 不一定非要在 output 的地方，也可以是其他位置，比如 hidden layer 拿出来进行一些 transform，或者直接拿 hidden layer，都可以计算 smoothness

- Define the smoothness of the labels on the graph

$$S = \frac{1}{2}\sum_{i,j} w_{i,j}\left(y^i - y^j\right)^2 = \boldsymbol{y}^T L \boldsymbol{y}$$

Depending on network parameters

$$L = \sum_{x^r} C(y^r, \hat{y}^r) \boxed{+\lambda S}$$

As a regularization term

J. Weston, F. Ratle, and R. Collobert, "Deep learning via semi-supervised embedding," ICML, 2008