Semi-Supervised Learning

在所有的 data 中，unlabeled data 的數量遠大於 labeled data 時所使用的學習模式。又可細分為 Transductive learning 與 Inductive learning。前者的 unlabeled data 就是它的 testing set；而後者則完全不使用 testing data。

一、Semi-supervised Learning for Generative Model

首先初始化參數，這裡可以使用 labeled data 估測一個值或者 random 取值。

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

接著根據現有的 θ 估算每一筆 unlabeled data 屬於 class 1 的機率。(E)

$$P_\theta(C_1|x^u)$$

算出這個機率以後，再來就是要 update model。與原先沒有 unlabeled data 的情況相比，此處需考慮 unlabeled data 裡 C_1 出現的次數。(M)

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

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

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

此處獲得 μ^1 之後又可重回第一步更新 $P_\theta(C_1|x^u)$，反覆重複上述步驟之後直到收斂為止。該方法稱 EM algorithm。

Why?

這裡可以想成，我們要找一個 θ 去 maximize Log 的 likelihood，然而該式子並非 convex 所以解它的時候變成要用 EM algorithm 解。

- 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

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

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

二、Semi-supervised Learning for Low-density Separation

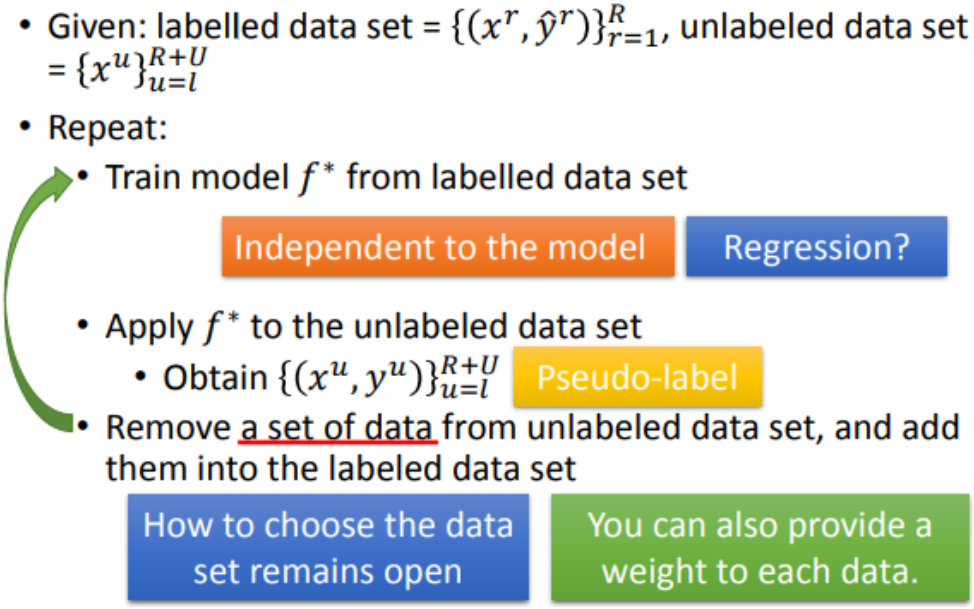
該方法的精神就是非黑即白，意思是說在 class 之間會有一個非常明顯的鴻溝，labeled data 以及 unlabeled data 都可以明確被分在某一個 class 中。而鴻溝的劃分取決於 density 的密度。

1. Self-training

首先從 labeled data 去 train 一個 model，這個 model 叫做 f*。

接著根據這個 f*，將 x^u 丟入去 label unlabeled data 得出 y^u，這些得到的 labeled data 稱 Pseudo-label。

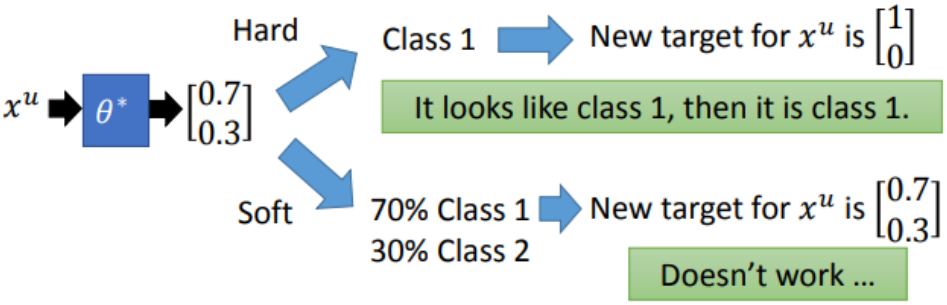
最後再取一部份的 Pseudo-label 加 labeled data set 裡面。
（可以給每一筆 unlabeled data provide weight 比較它們的 confidence）



(1) Self-training VS. Generative Model

兩者的差別在於做 Self-training 的時候用的是 Hard label；在做 Generative model 的時候用的是 Soft label。

舉例來說，Self-training 會強制 assign 一筆 training data，它一定是屬於某一個 class；但是 Generative model 的時候會根據它的 posterior probability，可能有部分屬於 class 1，有部分屬於 class 2。



而做 NN 時，Soft label 表現較差。

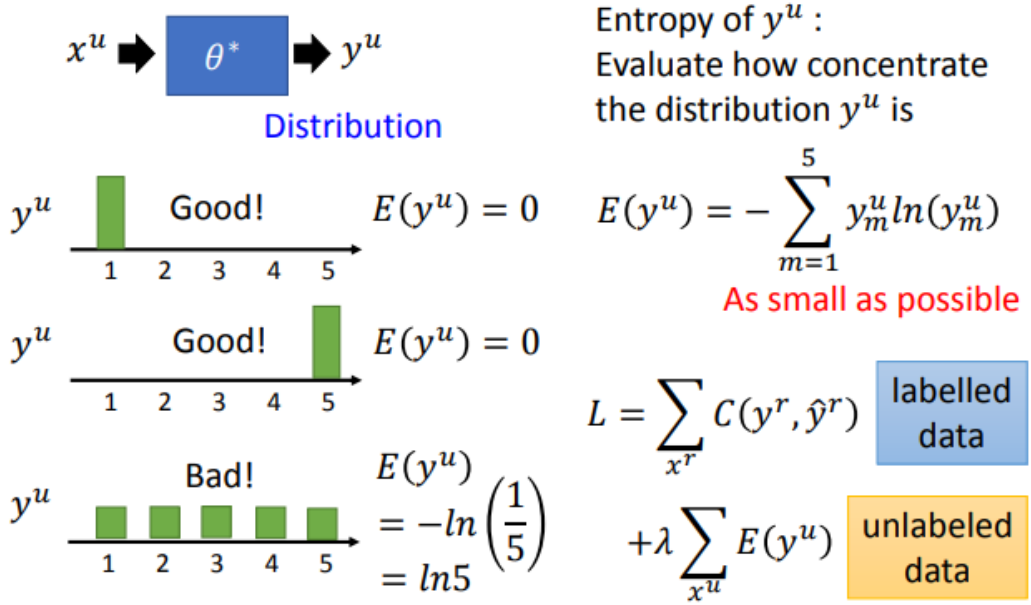
2. Entropy-based Regularization

用 neural network 的時候，output 是一個 distribution，而不是限制說這個 output 一定要是 class 1 或是 class 2。
但是必須假設這個 output 的 distribution 它一定要很集中，假設我們現在做 5 個 class 的分類，如果 output 都是在 class 1 的機率很大，而 4 在其它 class 的機率很小，那這個結果是好的。

至於要怎麼用數值的方法來 evaluate 這個 distribution 到底是好的還是不好，這邊要用的東西叫做 Entropy，所以我們需要做的事情是希望這個 model 的 output 在 labeled data 上，它的分類要正確；但是在 unlabeled data 上，它的 output 的 entropy 要越小越好。

所以根據這個假設可以設計 loss function，希望找一組參數在 labeled data 上的 model 的 output，跟正確的 model 的 output 距離越近越好，這裡就是用 cross entropy 來 evaluate 它們之間的距離。
在 unlabeled data 的部分則希望這些 unlabeled data 的 entropy 越小越好。

那在這兩項中間可以乘一個 weight 來考慮要偏向 unlabeled data 多一點還是少一點。

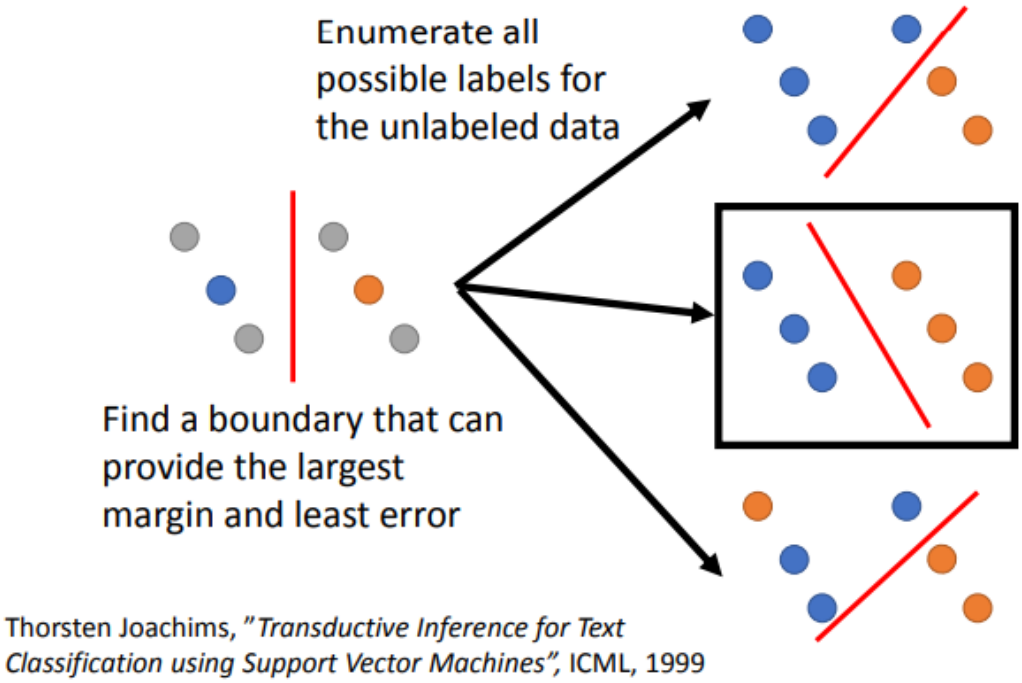


三、Outlook:Semi-supervised SVM

SVM 在處理該問題時，會窮舉所有可能的 label 裡面，哪一個可能性可以讓你的 margin 最大，同時又 minimize error。

舉例來說，這邊有 4 筆 unlabeled data，每一筆它都可以是屬於 class 1，也可以是屬於 class 2，窮舉它所有可能的 label 之後為下圖所示。
然後對每一個可能的結果都做一個 SVM，再去看哪一個 unlabeled data 的可能性，可以讓 margin 最大，同時又 minimize error。

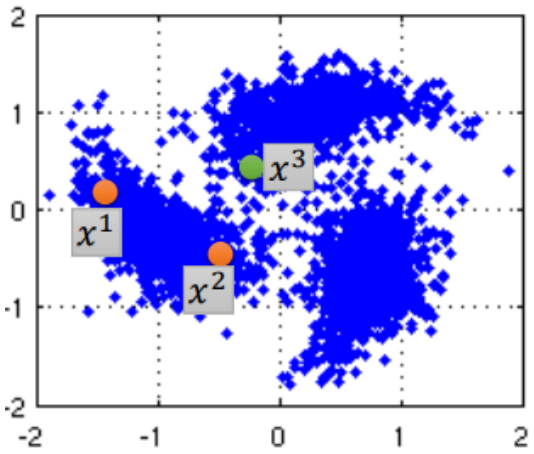
像是第一個分類正確但 margin 較大，而第三個假設不僅分類錯誤且 margin 也較大，因此選擇第二個假設。



Thorsten Joachims, "*Transductive Inference for Text Classification using Support Vector Machines*", ICML, 1999

四、Semi-supervised Learning for Smoothness Assumption

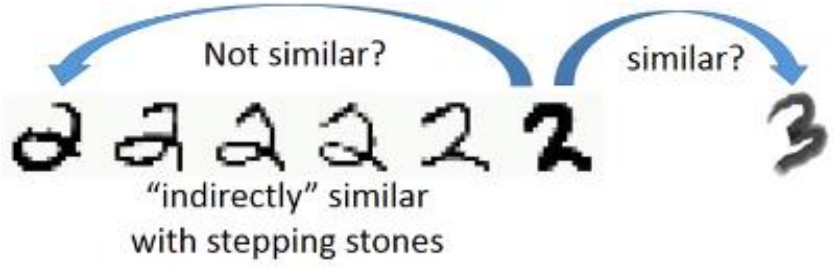
該方法的精神就是近朱者赤，意即 x 的分布是不平均的，而它在某些地方是很集中，某些地方又很分散。如果 x^1 和 x^2 在一個 high density 的 region 裡很接近的話，則 x^1 的 label y^1\head 跟 x^2 的 label y^2\head，它們才會很像。



x^1 and x^2 have the same label

x^2 and x^3 have different labels

進一步說明，high density 的 region 表示兩個 input data 中間有許多相似的 data 不斷「演化」過去；反之若兩者之間相似的 data 很少，代表它們之間為 low density region，即兩者並不相似。

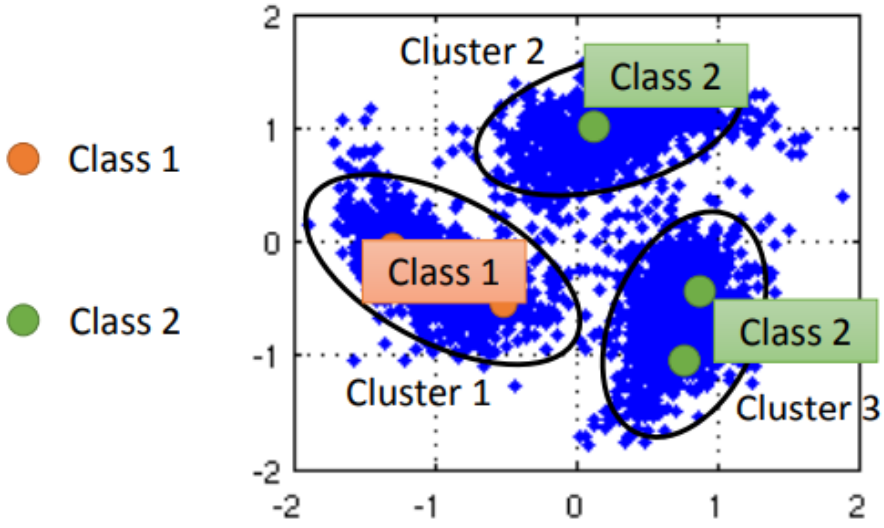


(The example is from the tutorial slides of Xiaojin Zhu.)

Source of image: http://www.moehui.com/5833.html/5/

1. Cluster and then Label

將所有 labeled data cluster 之後，在將 unlabeled data 透過 classifier 加上 label。（如果單純使用 pixel 來做 clustering 表現較差，需要有很好的方法，來描述 image，例如用 Deep Autoencoder call feature 然後再 call clustering）



Using all the data to learn a classifier as usual

2. Graph-based Approach

計算某一點與其 neighbor 之間的 similarity，之後將 similarity 高的點 connect 在一起，建成 graph 之後即可得知所有 data 的 high density path。
建 graph 的方法有很多種，像是 K Nearest Neighbor。該方法先設定一個 k 值，接著每一個 point 都跟它最近的、相似度最像的 k 個點做相連。

而 e-Neighborhood 則是設定某一個 threshold，每一個點只有跟它相似度超過該 threshold 的點才會被相連。

另外所謂的 edge 也不是只有相連和不相連，這樣 binary 的選擇而已。可以給 edge 一些 weight，讓 edge 跟要被連接起來的兩個 data point 之間的相似度是成正比的。

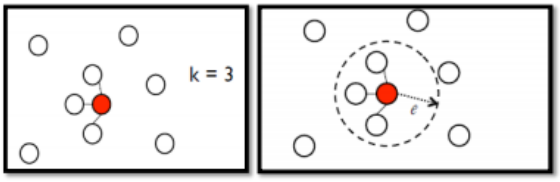
至於如何定義該相似度可以使用 RBM function，先算 x^i 跟 x^j 如果你都把它們用 vector 來表示的話，算它們的 Euclidean distance 前面乘一個參數，再乘一個負號，最終取 exponential。

至於取 exponential 這件事情在經驗上，可以給你比較好的 performance。
因為這個 function，它下降的速度是非常快的，所以只有當 x^i 跟 x^j 非常靠近的時候，它的 singularity 才會大；距離稍微遠一點，singularity 就會下降很快，變得很小。避免連到這種跨海溝的 link（橘色點與綠色點）。

Graph-based Approach - Graph Construction

The image is from the tutorial slides of Amarnag Subramanya and Partha Pratim Talukdar

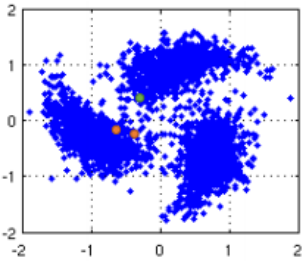
- Define the similarity $s(x^i, x^j)$ between x^i and x^j
- Add edge:
  - K Nearest Neighbor
  - *e*-Neighborhood



- Edge weight is proportional to $s(x^i, x^j)$

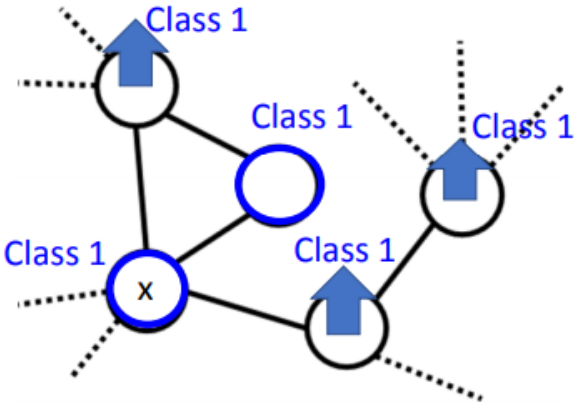
Gaussian Radial Basis Function:

$$s(x^i, x^j) = exp\left(-\gamma \|x^i - x^j\|^2\right)$$



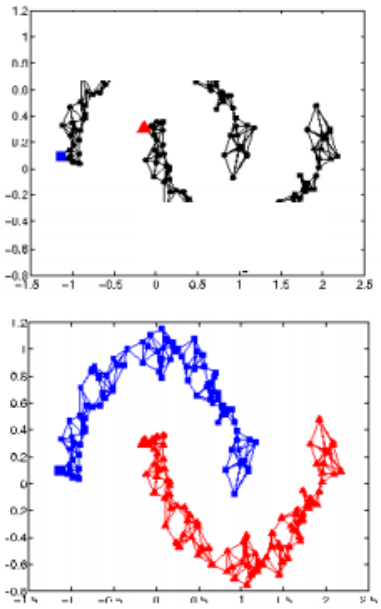
Graph-based 的 approach 還有一個特點就是它會像傳染病一樣傳遞 class 過去。以下圖為例，假設該圖已知 x 為 class 1，那其周圍的點就會被歸在 class 1，而它們相鄰的點被歸在 class 1 的機率亦會跟著上升。

然而要讓 graph-based 這種 Semi-supervised 的方法有用，data 的數量要夠多否則 information 就傳不過去了。



The labelled data influence their neighbors.

Propagate through the graph



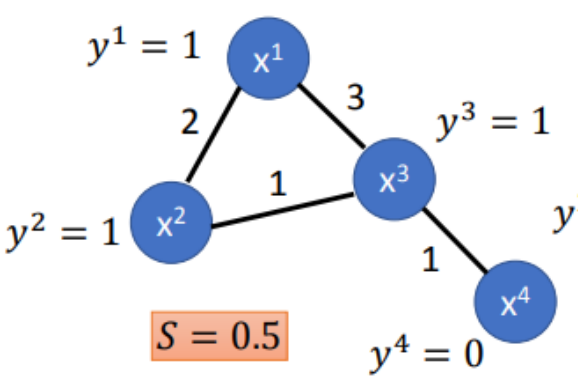
至於要如何判斷在該 graph structure 的 label 標得好不好，就必須定義 label 在 graph 上的 smoothness，這個值越小表示定出來的 label 越好。

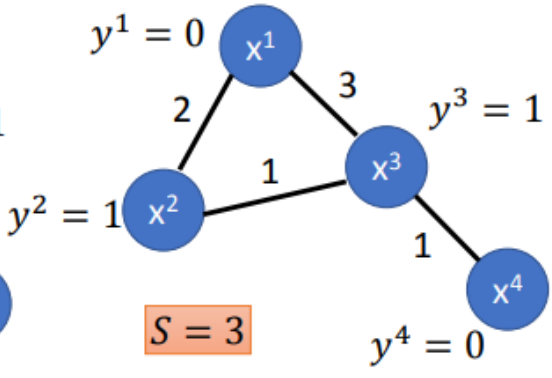
- Define the smoothness of the labels on the graph

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

Smaller means smoother

For all data (no matter labelled or not)



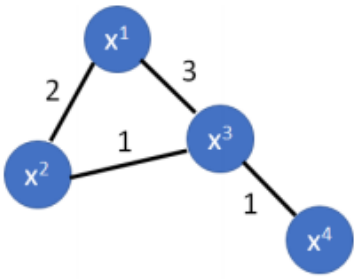


我們可以再整理一下上述 smoothness 的式子。把 y 串成一個 vector，而它包括 labeled data 與 unlabeled data，因此為(R + U)個 dimension，於是 S 就可以寫成 vector y 乘上 matrix L 在乘上 y 的 transpose。

L 是一個(R +U)*(R + U)的 matrix，稱作 Graph Laplacian，可表示為兩個 matrix 的相減，D-W。
W 就是這些 data point 兩兩之間 weight 的 connection 的關係建成的 matrix；
D 則是把 W 的每一個 row 合起來放在 diagonal 的地方，建成的 diagonal matrix。

- Define the smoothness of the labels on the graph

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



y: (R+U)-dim vector

$$\boldsymbol{y} = \left[\cdots y^i \cdots y^j \cdots\right]^T$$

L: (R+U) x (R+U) matrix

Graph Laplacian

$$L = D - W$$

$$W = \begin{bmatrix} 0 & 2 & 3 & 0 \\ 2 & 0 & 1 & 0 \\ 3 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad D = \begin{bmatrix} 5 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

最後 loss function 考慮的除了 labeled data 的距離(cross entropy)外，還需再加上 unlabeled data 的 smoothness 乘上某一個可調的參數 λ。它其實就象徵了一個 Regularization。

至於算 smoothness 時候，不一定要算在 output 的地方。以一個 deep neural network 為例，可以把 smoothness 放 network 的任何地方，除了假設 output 是 smooth 外；
也可以同時假設某一個 hidden layer 接出來，再乘上一些別的 transform，讓它也要 smooth；
也可以使每一個 hidden layer 的 output 都要是 smooth 的。
最後同時把這些 smooth 通通都加到 neural network 上面即可。

- 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

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

As a regularization term

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