Advanced Machine Learning : from Theory to Practice
Lecture 7
Graphs and Machine Learning II - Semi-supervised learning

F. d'Alché-Buc

Fall 2017

Introduction to Semi-supervised learning
Learning from labeled and unlabeled data

Why Semi-supervised learning ?

- Labels are expensive
- Benefit from the availability of huge sets of unlabeled data
- Unlabeled data inform us about the probability distribution of the data $p(x)$
- Can we use it ? does it improve the performance of the resulting regressors/classifiers ?

Applications

- image search, (Fergus et al., 2009)
- genomics (Shi and Zhang,2011)
- natural language parsing (Liang, 2005)
- speech analysis (Liu and Kirchhoff, 2013)

Goal

- Labeled data : $\mathcal{S}_\ell = \{(x_1, y_1), \dots, (x_\ell, y\ell)\}$, it Assumption : i.i.d. data drawn from the joint probability distribution P(X,Y)
- Unlabeled data : $\mathcal{X}_u = \{x_{\ell+1}, \dots, x_{\ell+u}\}$, $n = \ell + u$: available during training ! *Assumption :* i.i.d. data drawn from the marginal P(X)
- Usually $\ell << u$
- Test data : $\mathcal{X}_{test} = \{x_{n+1}, \dots, x_{n+m}\}$: not available during training, again with *Assumption :* i.i.d. data drawn from the marginal P(X)
- **Learn a function f : $\mathcal{X} \to \mathcal{Y}$ (regression/classification) that it generalizes well on test data**

Introduction to Semi-supervised learning

Definition of transduction (example for binary classification)

Goal

- Labeled data : $\mathcal{S}_\ell = \{(x_1, y_1), \ldots, (x_\ell, y\ell)\}$, **Assumption :** i.i.d. data drawn from the joint probability distribution P(X,Y)
- Unlabeled data : $\mathcal{X}_u = \{x_{\ell+1}, \ldots, x_{\ell+u}\}$, $n = \ell + u$: available during training ! **Assumption :** i.i.d. data drawn from the marginal P(X)
- Usually $\ell << u$
- No Test data
- **Learn $Y_U \in \{-1, 1\}$ the prediction vector of size u for the sole unlabeled data available during training !**

Introduction to Semi-supervised learning
Example : a relevant case for transduction

Example : link prediction in a protein-protein network (a binary classification task on pairs of nodes)

- For some species, you know that the set of all proteins is perfectly known.
- You already know the "test" data and you assume that you won't get anymore data to test on

Of course, a semi-supervised learning algorithm (inductive approach) can do the job.

Learn f from $\mathcal{X}$ to $\mathcal{Y}$ using $\mathcal{S}_\ell = \{(x_1, y_1), \ldots, (x_\ell, y_\ell)\}$ and $\mathcal{X}_u = \{x_{\ell+1}, \ldots, x_{\ell+u}\}$

Smoothness assumption of semi-supervised learning

SSL cannot work in all cases. Certain assumptions have been proposed.

- if two input points x_1 and x_2 in a high density region are close, then so should be the corresponding y_1 and y_2.

Cluster assumption

- if two input points x_1 and x_2 are close, they belong to the same class
- in other words, it is a *Low density separation assumption* : the frontier between two classes lies on a low-density region

Manifold assumption

- the (input) data roughly lie in a low-dimensional manifold. We view the manifold as an approximation of high density regions, the smoothness assumption of SSL reduces to the smoothness assumption of supervised-learning applied on the manifold.

Finally, let us emphasize that SSL won't work for uniform P(X).

- Learn f from $\mathcal{X}$ to $\mathcal{Y}$ using $\mathcal{S}_\ell = \{(x_1, y_1), \ldots, (x_\ell, y_\ell)\}$ and $\mathcal{X}_u = \{x_{\ell+1}, \ldots, x_{\ell+u}\}$
- Methods
  - Self-training
  - Margin for unlabeled data
  - Smoothness penalty (graph-based semi-supervised learning)
  - Deep learning (a short view)

Self-training approaches
Self-training algorithm for supervised classification

- Any classifier : f

Principle

```
1. k=0 ; $\mathcal{S}_0 = \mathcal{S}$ ; $\mathcal{D}_0 = \emptyset$
2. Learn $f_0$ by training on $\mathcal{S}_0$
3. Delta = 1000
4. WHILE (Delta $\geq \varepsilon$ and $k \leq$ Max) DO
    • Use $f_k$ to label $\mathcal{X}_u - \mathcal{D}_k$ and get $\mathcal{D}_{k+1}$, subset of $\mathcal{X}_u - \mathcal{D}_k$ with the most confident labels predicted by $f_k$
    • build $\mathcal{S}_{k+1} = \mathcal{S}_k \cup \mathcal{D}_{k+1}$
    • Learn $f_{k+1}$ by training on $\mathcal{S}_{k+1}$
    • $Delta = Distance(f_{k+1}, f_k)$ ; k := k+1
```

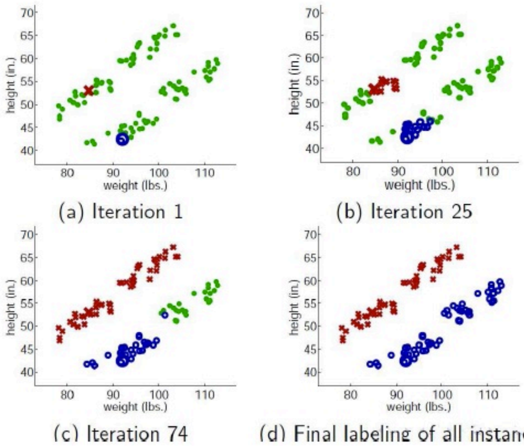
- How to define the most confident labels ? and how many ?

Self-training approaches

Self-training : example with k-NN (1)

- Two nice clusters without outliers [example Piyush Ray]

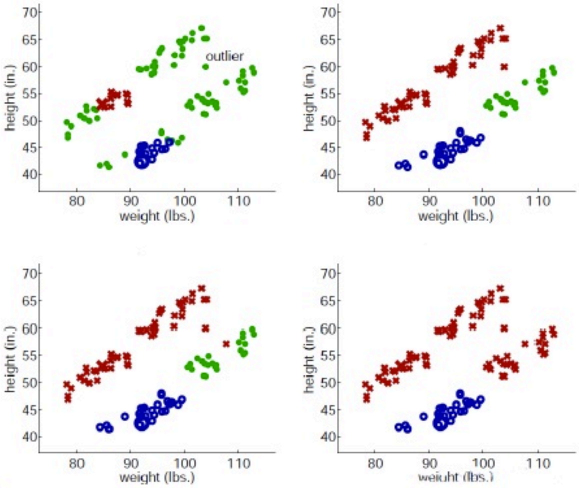
Base learner: KNN classifier



Self-training approaches

Self-training : example with k-NN (2)

- Two clusters with outliers



Self-training approaches
Self-training algorithm with generative models

Log-likelihood for a mixture model :

$$\mathcal{L}(\theta, X, Y) = \log P(\theta)) + \sum_{x_i \in \mathcal{X}_u} \log \sum_{j=1}^{M} P(c_j|\theta) P(x_i|c_j, \theta) + \dots$$
$$\sum_{x_i \in \mathcal{S}_\ell} \log(P(y_i = c_j|\theta) P(x_i|y_i = c_j|\theta)$$

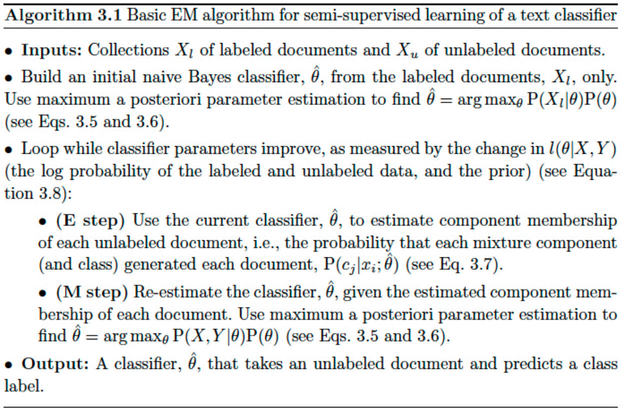
NB : the constants are dropped.

- a basic model one cluster per class
- More expressive model is possible

Self-training approaches

Self-training algorithm with generative models

Basic algorithm



Algorithm 3.1 Basic EM algorithm for semi-supervised learning of a text classifier

- **Inputs:** Collections X_l of labeled documents and X_u of unlabeled documents.
- Build an initial naive Bayes classifier, $\hat{\theta}$, from the labeled documents, X_l, only. Use maximum a posteriori parameter estimation to find $\hat{\theta} = \arg\max_\theta \mathrm{P}(X_l|\theta)\mathrm{P}(\theta)$ (see Eqs. 3.5 and 3.6).
- Loop while classifier parameters improve, as measured by the change in $l(\theta|X, Y)$ (the log probability of the labeled and unlabeled data, and the prior) (see Equation 3.8):
    - **(E step)** Use the current classifier, $\hat{\theta}$, to estimate component membership of each unlabeled document, i.e., the probability that each mixture component (and class) generated each document, $\mathrm{P}(c_j|x_i;\hat{\theta})$ (see Eq. 3.7).
    - **(M step)** Re-estimate the classifier, $\hat{\theta}$, given the estimated component membership of each document. Use maximum a posteriori parameter estimation to find $\hat{\theta} = \arg\max_\theta \mathrm{P}(X, Y|\theta)\mathrm{P}(\theta)$ (see Eqs. 3.5 and 3.6).
- **Output:** A classifier, $\hat{\theta}$, that takes an unlabeled document and predicts a class label.

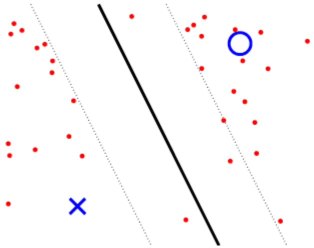
Ref :(Nigam et al. 2006)

- Pros
  - Can be used with any classifier
  - Simple
- Cons
  - Heuristic and *ad hoc* approach
  - Not well founded
  - Even in case of generative models : Avrum and Cohen (2006) showed a possible performance degradation.

Margin-based approaches

Learn the unknown labels of the training set

Using Transductive SVM : here is the data



Idea : during the learning phase, learn both the parameters of the SVM and the unknown labels of $\mathcal{X}_u$

Margin-based approaches
Transductive Support Vector Machine (Joachims, 1999)

Joachims proposed a Transductive SVM with a soft margin. Let us call $\mathbf{y}^* = [y_1^*, \dots, y_u^*]$ the prediction vector for the unknown labels of the unlabeled part of the training set.

TSVM

$$\underset{\mathbf{w}, y^*, b, \xi, \xi^*}{\text{minimize}} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{\ell} \xi_i + C^* \sum_{j=1}^{u} \xi_i^*$$

under the constraints

$$y_i(\mathbf{w}^T \mathbf{x}_i + \mathrm{b}) \geq 1 - \xi_i,\ i = 1, \dots, n$$

$$y_j^*(\mathbf{w}^T \mathbf{x}_{\ell+j} + \mathrm{b}) \geq 1 - \xi_j^*,\ i = 1, \dots, n$$

$$y_j^* \in \{-1, +1\},\ j = 1, \dots, u$$

$$\xi_i \geq 0$$

$$\xi_j^* \geq 0$$

Ref : Joachims, 1999.

Margin-based approaches

Transductive Support Vector Machine (Joachims, 1999)

A few remarks about the nature of the method

- It is called transductive because the algorithm seems to focus on learning the unknown labels of the training set : therefore it solves a transductive problem
- However it is an inductive method in the sense that after learning you can use the resulting prediction function on new un labeled data : a single name for different variants of the method is now used Semi-Supervised SVM or S^3VM

About the optimization problem :

- It is a combinatorial problem
- It is NP-hard to find the integer y_i^*'s !

Margin-based approaches
Solution to the optimization problem in S^3VM

Keeping the exact/combinatorial problem

- Mixed integer programming method : S^3VM by Bennet and Demiriz 1999, 2001 :
- Branch and bound algorithm, Chapelle, Sindwani and Keerthi, 2006

Margin-based approaches
Solution to the optimization problem in S^3VM

Relaxing the exact/combinatorial problem

- Relaxation by Semi-definite programming : De Bie and Cristianini, 2004,2006
- Heuristic Joachims, 2003

Margin-based approaches

Semi-definite programming for S^3VM (1)

Idea :

solve the problem in the dual space

$$\min_{Y_u} \max_{\alpha} 2\alpha^T 1 - \alpha^T (K \odot YY^T)\alpha$$

under the constraints

$$Y = \begin{pmatrix} Y \\ Y_u \end{pmatrix}$$

$$Y_u = \{-1, 1\}^u$$

Margin-based approaches
Semi-definite programming for S^3VM (2)

Reformulate with matrix $\Gamma = YY^T$

$$\min_{\Gamma} \max_{\alpha} 2\alpha^T 1 - \alpha^T (K \odot \Gamma)\alpha$$

under the constraints

$$\Gamma = \begin{pmatrix} Y_\ell Y_\ell^T & Y_\ell Y_u^T \\ Y_u Y_\ell^T & Y_u Y_u^T \end{pmatrix}$$

$$Y_U = \{-1, 1\}^u$$

Re-parametrize the problem in terms of Γ a sdp matrix, with rank 1 constraint. ref : De Bie, Cristianini, 2006, `http://www.tijldebie.net/publications/SSLusingSDP`

Margin-based approaches
Semi-supervised learning with a new margin definition

Let us go further, we do not need to explicitly find Y_u. Let us define a margin for the unlabeled data as : $\rho(x,h) = |h(x)|$. Then,

$$\underset{\mathbf{w},b,\xi}{\text{minimize}} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{\ell}\xi_i + C^*\sum_{i=\ell+1}^{\ell+u}\xi_i$$

under the constraints

$$y_i(\mathbf{w}^T\mathbf{x}_i + \mathrm{b}) \geq 1 - \xi_i,\ i = 1,\dots,\ell$$

$$|\mathbf{w}^T\mathbf{x}_i + \mathrm{b}| \geq 1 - \xi_i,\ i = \ell+1,\dots,\ell+u$$

$$\xi_i \geq 0, i = 1,\dots,n = \ell+u$$

ref :Collobert et al., JMLR, 2006.

Use differentiable functions of the margin to solve the problem + concave - convex methods

Margin-based approaches

Semi-supervised learning with a new margin definition

- Margin : $\rho(x, y, h) = y.h(x)$
- Which margin for unlabeled data ?
- Reinforce the confidence of the classifier
  - $\rho_2(x, h) = h(x)^2$
  - $\rho_1(x, h) = |h(x)|$
  - **Implicit assumption** : cluster assumption : data in the same cluster share the same label
- Worked for SVM, MarginBoost, . . .

Margin-based approaches

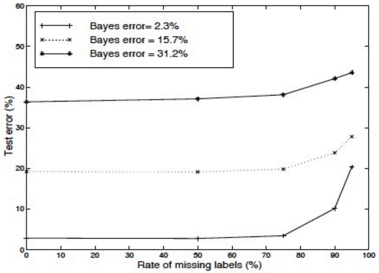
Semi-supervised MarginBoost

- $h_t \in \mathcal{H}$: base classifier
- Boosting model : $H_T(x) = \sum_t \alpha_t h_t(x)$
- Loss function : $J(H_t) = \sum_{i=1}^{\ell} exp(-\rho(x_i, y_i, H_t)) + \lambda \sum_{j=\ell+1}^{n} exp(-\rho_u(x_j, H_t))$

ref :Semi-Supervised MarginBoost, NIPS, 2001.

Margin-based approaches
Semi-supervised MarginBoost

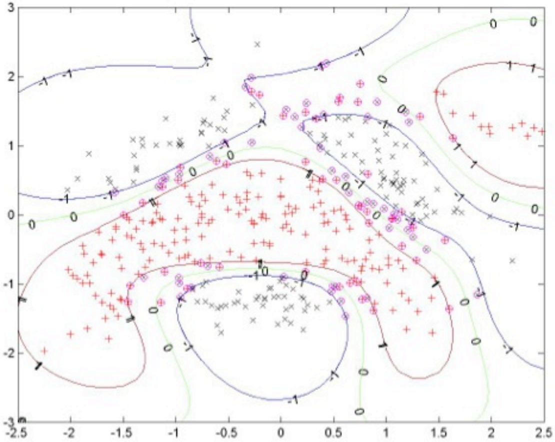
- Toys problems with different level of difficulty (we control Bayes error by mixing more or less the generative models)



[figure : NIPS 2001]

Margin-based approaches

Data used in the previous sample



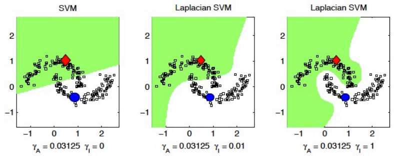
Graph-based approaches

How to use the geometry of the marginal distribution P_x ?

The key ideas :

- We assume that a better knowledge of the marginal distribution $P_x(x)$ will give us bette knowledge of $P(Y|x)$.
- If two points x_1 and x_2 are close in the intrinsic geometry of P_x then the conditional distribution $P(y|x_1)$ and $P(y|x_2)$ will be close.

Graph-based approaches
Manifold regularization



- If $\mathcal{M}$, the support of P_x is a submanifold $\subset \mathbb{R}^p$, then we can try to minimize the penalty :

$$||f||_I^2 = \int_{\mathcal{M}} ||\nabla_{\mathcal{M}} f||^2 dP_x$$

  tat reflects the intrinsic structure of the distribution P_x.

The gradient of f is taken with respect to the Riemanian manifold $\mathcal{M}$, meaning that we take into account the geometry underlying the support of P_x.

Graph-based approaches

Where the Laplace-Beltrami operator comes into scene

Expression of $||f||_I^2$

If the manifold is infinite or if the support of P_x vanishes at the boundary of $\mathcal{M}$, then the following holds :

$$||f||_I^2 = \int_{\mathcal{M}} ||\nabla_{\mathcal{M}} f||^2 dP_x = \int_{\mathcal{M}} f\mathcal{L}f dP_x = \langle f, \mathcal{L}f \rangle_{L^2(P_x)} \qquad (1)$$

where $\mathcal{L}$ is the Laplace-Beltrami operator on functions :

$$\mathcal{L}f = \text{div } \nabla f$$

Graph-based approaches

Analogue of Laplace-Beltrami operator in graph theory

Let $\mathcal{G}$ be a graph with an adjacency matrix W. $L^2(\mathcal{G})$ is a space of functions $\mathcal{G} \to \mathbb{R}$. The graph Laplacian :

$$L = D - W$$

where D is the diagonal matrix of degrees (sum of weights of each node),
is a positive definite operator on $L^2(\mathcal{G})$. Eigenfunctions of the Laplacian for an orthonormal basis for $L^2(\mathcal{G})$.

Graph-based approaches
Approximation of $||f||_I^2$

We have :

$$\sum_{ij} w_{ij}(f(x_i) - f(x_j))^2 = 2f^T Lf$$

Manifold regularization : smoothness on the data graph

$$||f||_I^2 \approx \sum_{ij} w_{ij}(f(x_i) - f(x_j))^2,$$

Ref :Belkin, Nyogi and Sindwani (2006)

Graph-based approaches

Semi-supervised learning with a smoothness constraint

Let k be a positive definite kernel and $\mathcal{H}_k$ the unique RKHS induced by k.

Smoothness constraint / Manifold regularization 1/2

- Training data : $\mathcal{S}_\ell = \{(x_i, y_i, i =, \ldots \ell\}$ and $\mathcal{S}_u = \{x_{\ell+1}, \ldots, x_{\ell+u}\}$
- For $f \in \mathcal{H}_k$ and W a similarity matrix between data
- Impose an additional penalty that ensures smoothness of function f : for two close inputs, f takes close values
-

Ref :Belkin, Nyogi and Sindwani (2006)

Graph-based approaches

Semi-supervised learning with a smoothness constraint 2/2

Let k be a positive definite kernel and $\mathcal{H}_k$ the unique RKHS induced by k.

Smoothness constraint / Manifold regularization

Minimize $J(f)$ in $\mathcal{H}_k$:

$$J(f) = \frac{1}{\ell}\sum_{i=1}^{\ell} V(x_i, y_i, f) + \lambda ||f||_k^2 + \lambda_u \sum_{ij} w_{ij}(f(x_i) - f(x_j))^2$$

Graph-based approaches

Semi-supervised learning with a smoothness constraint 2/2

Let k be a positive definite kernel and $\mathcal{H}_k$ the unique RKHS induced by k.

Smoothness constraint / Manifold regularization

Minimize $J(f)$ in $\mathcal{H}_k$:

$$\begin{aligned} J(f) &= \frac{1}{\ell}\sum_{i=1}^{\ell} V(x_i, y_i, f) + \lambda||f||_k^2 + \lambda_u \sum_{ij} w_{ij}(f(x_i) - f(x_j))^2 \\ &= \frac{1}{\ell}\sum_{i=1}^{\ell} V(x_i, y_i, f) + \lambda||f||_k^2 + \lambda_u f^T L f \end{aligned}$$

Graph-based approaches

Representer theorem

$$\begin{aligned} J(f) &= \frac{1}{\ell}\sum_{i=1}^{\ell} V(x_i, y_i, f) + \lambda ||f||_k^2 + \lambda_u \sum_{ij=1}^{\ell+u} w_{ij}(f(x_i) - f(x_j))^2 \\ &= \frac{1}{\ell}\sum_{i=1}^{\ell} V(x_i, y_i, f) + \lambda ||f||_k^2 + \lambda_u f^T L f \end{aligned}$$

Any minimizer of $J(f)$ admits a representation
$\hat{f}(\cdot) = \sum_{i=1}^{\ell+u} \alpha_i k(x_i, \cdot)$

Graph-based approaches

Laplacian Regularized Least Square regression

- Closed-from solution : extension of ridge regression

$$V(x_i, y_i, f) = (y_i - f(x_i))^2$$

$$\lambda_L = \frac{\lambda_u}{u + \ell}$$

$$\hat{\alpha} = (JK + \lambda \ell Id + \frac{\lambda_u \ell}{(u+\ell)^2} LK)^{-1} Y$$

K : Gram matrix for all data
J : $(\ell + u) \times (\ell + u)$ diagonal matrix with the first ℓ values equal to 1 and the remaining ones to 0.

Graph-based approaches

Laplacian SVM

We choose the hinge loss functions :

$$\min_{f \in \mathcal{H}_k} \frac{1}{\ell} \sum_{i=1}^{\ell} (1 - y_i f(x_i))_+ + \lambda ||f||_k^2 + \frac{\lambda_u}{u + \ell} f^T L f$$

We benefit from the representer theorem.

Graph-based approaches
Laplacian SVM

In practise, we solve :

$$\min_{\alpha \in \mathbb{R}^{l+u}, \xi \in \mathbb{R}^{l}} \frac{1}{l} \sum_{i=1}^{l} \xi_i + \gamma_A \alpha^T K \alpha + \frac{\gamma_I}{(u+l)^2} \alpha^T K L K \alpha$$

$$\text{subject to: } y_i \left(\sum_{j=1}^{l+u} \alpha_j K(x_i, x_j) + b \right) \geq 1 - \xi_i, \quad i = 1, \ldots, l$$

$$\xi_i \geq 0 \quad i = 1, \ldots, l.$$

Graph-based approaches
Laplacian SVM

In practise, we solve :

$$\min_{\alpha \in \mathbb{R}^{l+u}, \xi \in \mathbb{R}^{l}} \frac{1}{l} \sum_{i=1}^{l} \xi_i + \gamma_A \alpha^T K \alpha + \frac{\gamma_I}{(u+l)^2} \alpha^T K L K \alpha$$

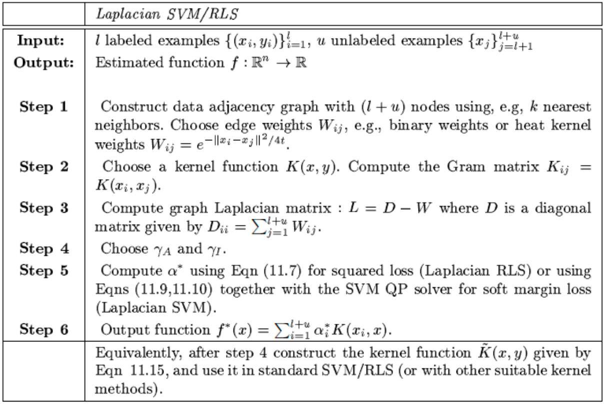
$$\text{subject to: } y_i \Big(\sum_{j=1}^{l+u} \alpha_j K(x_i, x_j) + b \Big) \geq 1 - \xi_i, \quad i = 1, \ldots, l$$

$$\xi_i \geq 0 \quad i = 1, \ldots, l.$$

Graph-based approaches

Laplacian SVM/RLS algorithm

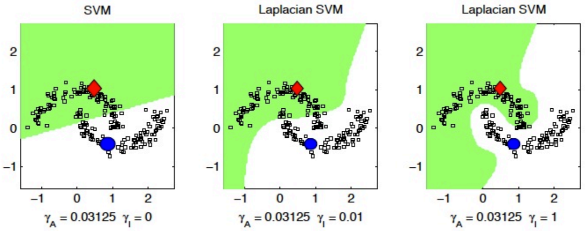
Results : Belkin et al. 2006, in Book : Semi-supervised Learning .



	Laplacian SVM/RLS
Input:	l labeled examples $\{(x_i, y_i)\}_{i=1}^{l}$, u unlabeled examples $\{x_j\}_{j=l+1}^{l+u}$
Output:	Estimated function $f : \mathbb{R}^n \rightarrow \mathbb{R}$
Step 1	Construct data adjacency graph with $(l+u)$ nodes using, e.g, k nearest neighbors. Choose edge weights W_{ij}, e.g., binary weights or heat kernel weights $W_{ij} = e^{-\|x_i - x_j\|^2/4t}$.
Step 2	Choose a kernel function $K(x, y)$. Compute the Gram matrix $K_{ij} = K(x_i, x_j)$.
Step 3	Compute graph Laplacian matrix : $L = D - W$ where D is a diagonal matrix given by $D_{ii} = \sum_{j=1}^{l+u} W_{ij}$.
Step 4	Choose γ_A and γ_I.
Step 5	Compute α^* using Eqn (11.7) for squared loss (Laplacian RLS) or using Eqns (11.9,11.10) together with the SVM QP solver for soft margin loss (Laplacian SVM).
Step 6	Output function $f^*(x) = \sum_{i=1}^{l+u} \alpha_i^* K(x_i, x)$.
	Equivalently, after step 4 construct the kernel function $\tilde{K}(x, y)$ given by Eqn 11.15, and use it in standard SVM/RLS (or with other suitable kernel methods).

Graph-based approaches
Laplacian SVM :results

Results : Belkin et al. 2006.

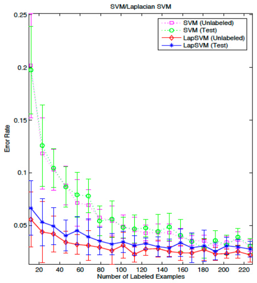


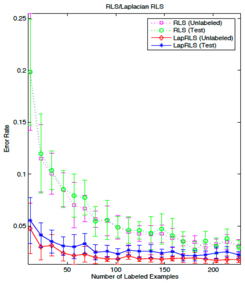
Graph-based approaches

Laplacian SVM :results on image classification (2 classes)

Results : Belkin et al. 2006.

Figure 11.5 Image Classification: Laplacian SVM/RLS performance with respect to number of labeled examples on unlabeled and test data.





Graph-based approaches
Back to unsupervised learning

Again, working in RKHS $\mathcal{H}_k$, we would like to solve the following (relaxed) clustering problem :

$$\min_{f \in \mathcal{H}_k, \sum_i f(x_i)=0, \sum_i f(x_i)^2=1} \gamma \|f\|_k^2 + \sum_{ij} w_{ij}(f(x_i) - f(x_j))^2 \qquad (2)$$

This approach can be seen as a *regularized spectral clustering*. It also benefits from a representer theorem : f^* the minimizer of Eq. 2 satisfies :

$$f^*(\cdot) = \sum_{i=1}^{n} \alpha_i \; k(\cdot, x_i)$$

Therefore the problem boils down to solve :

$$\min_{\alpha \in \mathbb{R}^n, 1^T K\alpha=0, \alpha^T K^2 \alpha=1} \gamma \alpha^T K \alpha + \alpha^T KLK\alpha$$

Graph-based approaches

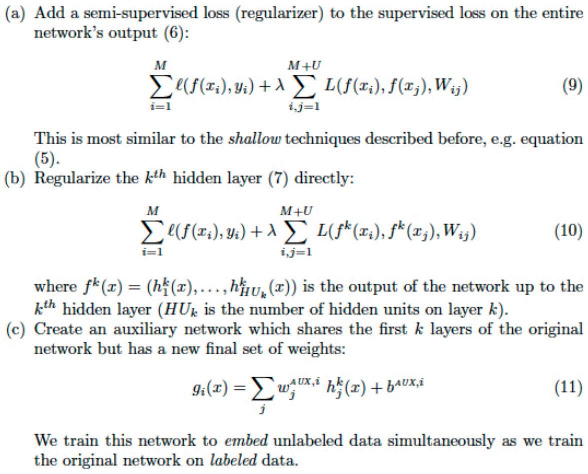
Regularized version of spectral clustering

$$\min_{\alpha \in \mathbb{R}^n, 1^T K\alpha = 0, \alpha^T K^2 \alpha = 1} \gamma \alpha^T K \alpha + \alpha^T KLK\alpha$$

- show that $\alpha^* = Pv$, v being the eigenvector with the smallest eigenvalue of a generalized eigenvector problem.
- Remark : This clustering provides a natural out-of-sample extension (classification of new datapoints)

Semi-supervised Deep learning
Semi-supervised embedding

3 ways proposed by Weston et al. 2008.



(a) Add a semi-supervised loss (regularizer) to the supervised loss on the entire network's output (6):

$$\sum_{i=1}^{M} \ell(f(x_i), y_i) + \lambda \sum_{i,j=1}^{M+U} L(f(x_i), f(x_j), W_{ij}) \qquad (9)$$

This is most similar to the *shallow* techniques described before, e.g. equation (5).

(b) Regularize the k^{th} hidden layer (7) directly:

$$\sum_{i=1}^{M} \ell(f(x_i), y_i) + \lambda \sum_{i,j=1}^{M+U} L(f^k(x_i), f^k(x_j), W_{ij}) \qquad (10)$$

where $f^k(x) = (h_1^k(x), \ldots, h_{HU_k}^k(x))$ is the output of the network up to the k^{th} hidden layer (HU_k is the number of hidden units on layer k).

(c) Create an auxiliary network which shares the first k layers of the original network but has a new final set of weights:

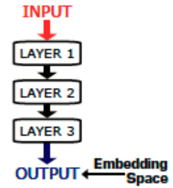
$$g_i(x) = \sum_j w_j^{AUX,i}\, h_j^k(x) + b^{AUX,i} \qquad (11)$$

We train this network to *embed* unlabeled data simultaneously as we train the original network on *labeled* data.

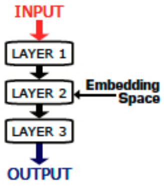
Semi-supervised Deep learning

Semi-supervised embedding

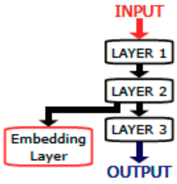
Architectures



(a) Output

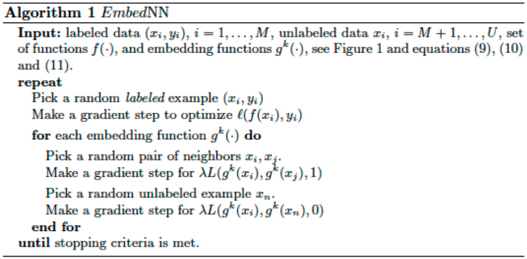


(b) Internal



(c) Auxiliary

Semi-supervised Deep learning
Semi-supervised embedding



```
Algorithm 1 EmbedNN
  Input: labeled data $(x_i, y_i)$, $i = 1, \ldots, M$, unlabeled data $x_i$, $i = M+1, \ldots, U$, set
  of functions $f(\cdot)$, and embedding functions $g^k(\cdot)$, see Figure 1 and equations (9), (10)
  and (11).
  repeat
    Pick a random labeled example $(x_i, y_i)$
    Make a gradient step to optimize $\ell(f(x_i), y_i)$
    for each embedding function $g^k(\cdot)$ do
      Pick a random pair of neighbors $x_i, x_j$.
      Make a gradient step for $\lambda L(g^k(x_i), g^k(x_j), 1)$
      Pick a random unlabeled example $x_n$.
      Make a gradient step for $\lambda L(g^k(x_i), g^k(x_n), 0)$
    end for
  until stopping criteria is met.
```

Ref :Weston et al., ICML 2008.

Outline

- Self-training method no more used
- Generative methods within deep learning are still used howver (group of Welling)
- Transductive SVM and S3VM avriants lack of scalability
- Manifold regularization are the most founded ones but again lack of scalability if not applying kernel approximation
- Theory is still lacking but semi-supervised learning can still bring improvement if the hyperparameter governing its weights is well chosen

- Code the Laplacian SVM or the Laplacian Kernel Ridge regressor
- Study the effect of graph construction
- Book : Semi-supervised learning, Chapelle, Scholpkoft, Zien,MIT