Theorem[section]

STATS 601 Project Report:

An adventure in Semi-supervised learning

TRONG DAT DO [1,*] and ZIYI SONG [1,**]

[1] *University of Michigan, Department of Statistics. West Hall, 1085 South University, Ann Arbor, MI, U.S.A., 48109. E-mail:* [*] dodat@umich.edu; [**] ziyisong@umich.edu

Contents

1. Introduction

In traditional problems as well as in our course, we only concern about studies where all the data are labeled (supervised learning) or all are unlabeled (unsupervised learning). But in real life, there are many problems where we need to face with amalgam of both labeled and unlabeled data, such as in medical or internet data, where labelling the data is much harder than collecting them. Semi-supervised learning is a learning paradigm invented to deal with this question. In this project, we study some semi-supervised learning algorithms and their connection to those that we learn on class. The rest of the project is structured as follows. In this section, we will propose the research questions and introduce the data set. The semi-supervised learning algorithms are described in section 2. In section 3, we talk about the results and comment on the performance of each method, their strength and weakness. Section 4 is the conclusion and discussion.

1.1. Research question 1: How can we use unlabeled data to improve classification quality (or what is Semi-supervised classification)?

Our first proposed question is: In the scenario that we have both labeled and unlabeled data, can we use the unlabeled to help predict the labels better? To answer this question, we will investigate 3 semi-supervised classifiers below and their corresponding supervised version, to see if semi-supervised learning indeed outperforms supervised learning or not. The last algorithm (Label spreading) is a beautiful algorithm of semi-supervised learning with a natural idea based on graph Bengio, Delalleau and Roux (2006), but we could not find any famous supervised learning algorithm corresponding to it.

Supervised learning method	Semi-Supervised learning method
Discriminant Analysis	Mixture Model
k-Nearest Neighborhood (kNN)	Self-training with kNN
AdaBoost	Co-training by committee
N/A	Label spreading

Table 1. Supervised classifiers and their corresponding semi-supervised ones

We will conduct our experiments as follows. Given a train set (includes both labeled and unlabeled data) and a test set (labeled only), we apply the semi-supervised classifier on the full train set and supervised classifier on the labeled part, then compare the two models using the test set. What we will see is, semi-supervised algorithms are better most of the time, but not all the time Zhu (2005). By using unlabeled data, we need to trade off by making some assumptions about data structure, which are not always true. We will also talk about this phenomena. For the last algorithm, we will compare its result to the others.

1.2. Research question 2: How can the limited labeled data help with clustering, especially when some labels not appeared in the labeled data but hidden in the unlabeled part?

Data encountered in clustering problems always consists of large amounts of unlabeled data and small proportion of labeled data. Because assigning labels is mainly accomplished by manual work and requires professional domain knowledge, labeled data is limited and contains precious information. To utilize the information in the setting of clustering, some methods are proposed. A typical, natural, and naive one is Cluster-then-Label algorithm Zhu and Goldberg (2009). Its illustration is deferred in section 2.5.

Cluster-then-Label algorithm supposes that all the different classes appear in the labeled part of data so that the number of classes is set. The Cluster-then-Label algorithm shows applicability in some cases; however, in many real world settings, the number of potential clusters is unknown and some undiscovered classes hide in unlabeled data. Therefore, Dirichlet process priors are involved to estimate the number of clusters from data itself. Furthermore, to take advantage of the information contained in labeled data, a Semi Dirichlet Process Mixture model is proposed Echraibi et al. (2019). Its philosophy is given in detail in section 2.6, and experimental results and comparisons are present in section 3.2.2.

1.3. The KEEL Datasets

We get the data sets from KEEL data set repository[1]. Because some of the algorithms that we consider can only handle real value data, we only choose the data set where all the features are real. Those can be seen in the table below, where each row is the name of data set, the sample size, number of features and number of classes to predict.

Data set	Sample size	Features	Classes
appendicitis	106	7	2
banana	5300	2	2
cleveland	303	13	5
ecoli	336	7	8
glass	214	9	7
iris	150	4	3
led7digit	500	7	10
pima	786	8	2
titanic	2201	3	2
wine	178	13	3

Table 2. Data sets from KEEL repository

Each data set has 4 versions: 10%, 20%, 30%, 40% of train data being labeled. The test set is the same for 4 versions. Each of the train data set also has been divided into 10 folds

[1] https://sci2s.ugr.es/keel/semisupervised.php

to be convenient for cross validating. All partition is done by KEEL. It is noteworthy that the label proportion in train data are not set equally to keep the nature of semi-supervised learning problems (indeed, in real life, we can not expect having balance labeled data).

2. Methodology

Notation: let $\{(x_1, y_1), \ldots, (x_l, y_l)\}$ be labeled train data and $\{x_{l+1}, \ldots, x_{l+u}\}$ unlabeled train data, where x_j is a p-dimensional features vector for all $j = 1, \ldots, l+u$. There are K classes in total.

2.1. Semi-supervised mixture model

It can be said that the semi-supervised mixture model is the combination of supervised mixture model (solved using Maximum Likelihood Estimation) and unsupervised mixture model (using Expectation-Maximization). We assume that our data are generate from a mixture of Gaussians

$$p(x|\pi, \mu, \Sigma) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k),$$

where $\mathcal{N}$ is the $p-$dimensional Gaussian distribution, $\pi = (\pi_k)_k$ sums up to 1, $\mu = (\mu_k)_k$ are means and $\Sigma = (\Sigma_k)_k$ are valid covariance matrices. Our goal is to estimate the parameters $(\pi_k, \mu_k, \Sigma_k)_k$. To make it convenient, we consider the equivalent model with latent variables Z,

$$X|(\pi, \mu, \Sigma, Z = k) \sim \mathcal{N}(X|\mu_k, \Sigma_k), \quad Z \sim \text{Categorical}((\pi_1, \ldots, \pi_k)).$$

The equivalence can be seen by law of total probability. With this model and the parameters $(\pi_k, \mu_k, \Sigma_k)_k$, we can compute (by Bayes' rule)

$$z_{jk} := P(Z = k|X_j, \pi, \mu, \Sigma) = \frac{\pi_k \mathcal{N}(x_j|\mu_k, \Sigma_k)}{\sum_{h=1}^{K} \pi_h \mathcal{N}(x_j|\mu_h, \Sigma_h)},$$

and can choose the k maximizing z_{jk} to be the class of x_j. To estimate $(\pi_k, \mu_k, \Sigma_k)_k$, we can use Maximum Likelihood Estimation (MLE) method. As we have seen on class, for the labeled data, because we have their classes z_{jk} (equals 1 if x_j belongs to class k and 0 otherwise), it is possible to use MLE to estimate the parameters. But for unlabeled data, we need to use E-M algorithm to alternatively estimate z_{jk} and the parameters. For semi-supervised learning, we can combine both of them. We will treat $(z_{jk})_{j=1}^{l}$ as deterministic numbers, and do not update them, meanwhile $(z_{jk})_{j=l+1}^{l+u}$ will be treated the same as in unsupervised Gaussian Mixture Model. This yields the following algorithm.

Algorithm 2.1 (EM algorithm for Gaussian Mixture Model)**.** *Given the labeled data* $\{(x_1, y_1), \ldots, (x_l, y_l)\}$*, and unlabeled data* $\{x_{l+1}, \ldots, x_{l+u}\}$*, our goal is to maximize the likelihood function of Gaussian Mixture Model with respect to parameters* (π, μ, Σ)*.*

1. *Initialize the parameters* $(\pi_k, \mu_k, \Sigma_k)_k$ *as the MLE based on the labeled data. (Similar to Discriminant Analysis). For all* $j = 1, \ldots, l$*, set*

$$z_{jk} = \begin{cases} 1 & \text{if } x_j \text{ belongs to class } k, \\ 0 & \text{otherwise.} \end{cases} \tag{2.1}$$

2. *E-step: Evaluate the conditional class probability for unlabeled data*

$$z_{jk} = \frac{\pi_k \mathcal{N}(x_j|\mu_k, \Sigma_k)}{\sum_{h=1}^{K} \pi_h \mathcal{N}(x_j|\mu_h, \Sigma_h)}, \tag{2.2}$$

   for all $j = l+1, \ldots, l+u, k = 1, \ldots, K$*.*
3. *M-step: Update the parameters based on the conditional probability: For all* $k = 1, \ldots, K$

$$z_k \leftarrow \sum_{j=1}^{l+u} z_{jk} \tag{2.3}$$

$$\pi_k \leftarrow \frac{z_k}{l+u} \tag{2.4}$$

$$\mu_k \leftarrow \frac{1}{z_k} \sum_{j=1}^{l+u} z_{jk} x_j \tag{2.5}$$

$$\Sigma_k \leftarrow \frac{1}{z_k} \sum_{j=1}^{l+u} z_{jk} (x_j - \mu_k)(x_j - \mu_k)^T \tag{2.6}$$

4. *Repeat 2. and 3. until the log likehood function converges to its local maximum*

$$l = \sum_{j=1}^{l+u} \log(\sum_{k=1}^{K} \pi_k \mathcal{N}(x_j|\mu_k, \Sigma_k)). \tag{2.7}$$

Return: The parameters (π, μ, Σ) *and class probability* $(z_{jk})_{j,k}$*.*

It is notable that to make this algorithm works well, we need to assume that our data come from a Mixture of Gaussian distributions. If the structure of data is different from that, the algorithm can not surpass the ordinary supervised learning algorithm, or even hurts the predictions from it.

2.2. Self-training with k-NN

The self-training algorithm is exactly what its name says. We will use a learning algorithm trained on the labeled data to label the unlabeled data, so that we can augment the labeled set. From that, we can artificially increase the training set's size and expect for a better classification rule. The algorithm can be stated as follows.

Algorithm 2.2. *Given the labeled data* $L = \{(x_1, y_1), \ldots, (x_l, y_l)\}$*, and unlabeled data* $U = \{x_{l+1}, \ldots, x_{l+u}\}$*. We choose a supervised learning method* f*. Repeat two steps until the unlabeled set becomes empty.*

1. *Train* f *on* L*.*
2. *Use* f *to predict the labels of data from* U*, choose a subset* S *that having highest predicted probability (often called "confidence"), add* S *to* L*, and subtract* S *from* U*.*

Return: Model f *has been trained on all data.*

Self-training algorithms have advantages that we can freely choose the supervised learning method f and the idea is simple. In steps, we need to choose the size of set S to add to L. Small $|S|$ will slow down the learning algorithm much because we need to retrain the model many times, meanwhile large $|S|$ will add many "low confident" predictions to labeled set, which can hurt the performance of original supervised learning algorithm. To make the self-training algorithm works well, we need to assume that its own prediction with high confidence must be correct, which is likely the case that the data are well-separated clusters.

In our experiments, we choose k Nearest Neighborhood as the based estimator. Because it has locally geometrical meaning, it fits well with the assumption that we made.

2.3. Co-training by committee: AdaBoost

The traditional co-training method is used when the data can be viewed in two ways Zhu and Goldberg (2009). But here our data are not in that type, so we want to implement another kind of co-training that is presented in Hady and Schwenker (2008). The idea is similar to Self-training, but instead of using a single classifier, we will use ensemble learners. AdaBoost is especially fit for this task, because when we train on a larger set, we will introduce more diversity to the ensemble learners, and this helps the ensemble more accurate than its members Brown et al. (2005). In our experiments, we choose AdaBoost as the ensemble learner and Decision tree as the based learner, as they give the best result Hady and Schwenker (2008). The co-training by AdaBoost algorithm is shown below.

Algorithm 2.3. *Given the labeled data* $L = \{(x_1, y_1), \ldots, (x_l, y_l)\}$*, unlabeled data* $U = \{x_{l+1}, \ldots, x_{l+u}\}$*, number of committee members* N*, a subset* U' *sampled randomly without replacement from* U*, and a number of maximum iterations* T*. Repeat for* t *from* 0 *to* T*, and break if* U *becomes empty*

1. $H^{(t)} = AdaBoost(L, N)$*.*
2. *Apply* $H^{(t)}$ *on* U' *for prediction, add the most confident predictions from* U' *to* L *and discard them from* U'*. Then we refill* U' *by sampling without replacement from* U *again.*

Return: committee $H^{(T)}$*.*

2.4. Label spreading

There are many graph based method that arose recently in the literature of semi-supervised learning Bengio, Delalleau and Roux (2006). The idea is to build a graph with vertices are data points, and equip it a distance matrix W, which is typically radial basis function $W_{ij} = \exp\left(\frac{-\|x_i - x_j\|^2}{2\sigma^2}\right)$. The algorithm first uses W to spread the label to all train data, and use them to predict labels for the test set.

The first graph based method is called Label propagation Zhu and Ghahramani

Algorithm 2.4 (Label propagation)**.** *Given the train set (both labeled and unlabeled data), we compute the dissimilarity matrix W, and its diagonal degree matrix D by $D_{ii} = \sum_j W_{ij}$. Denote Y_l the classes of labeled data. Initialize $\hat{Y}^{(0)} \in \mathbb{R}^{(l+u)\times K}$ (one-hot presentation for classes) such that*

$$Y_{ij}^{(0)} = \begin{cases} 1, & \text{if } x_i \text{ belongs to class } j, \\ 0, & \text{otherwise,} \end{cases} \tag{2.8}$$

and we write Y_l the first l row of matrix Y. Repeat

1. $\hat{Y}^{(t+1)} \leftarrow D^{-1}W\hat{Y}^{(t)}$
2. $\hat{Y}_l^{(t+1)} \leftarrow Y_l$

until converging to $\hat{Y}^{(\infty)}$. Then the label of X_i is estimated by $\arg\max_j Y_{ij}$ (softmax).

In words, the algorithm take the average of labels with respect to the distances in W and assign to each data point for every step, then recover the information about the labels of labeled data. But in this report, we will use a slightly different graph based method called Label spreading Zhou et al. (2004), which allows us to recover the labels information in a soft way by introducing one more parameter α.

Algorithm 2.5 (Label propagation)**.** *With the same setup in 2.4, set $W_{ii} \leftarrow 0$, and compute $L \leftarrow D^{-1/2}WD^{-1/2}$. Repeat*

$$\hat{Y}^{(t+1)} \leftarrow \alpha L\hat{Y}^{(t)} + (1-\alpha)\hat{Y}^{(0)}$$

until converging to $\hat{Y}^{(\infty)}$. Then the label of X_i is estimated by $\arg\max_j Y_{ij}$ (softmax).

The convergence of this algorithm can be proven by "contraction mapping" idea, and it is exponentially fast. It can be shown that this procedure is equivalent to finding minimizer for a quadratic criterion Zhou et al. (2004). When we predict the labels for the test set, we also use minimize that criterion but with respect to the label of test set.

2.5. clustering via Cluster-then-Label

Algorithm 2.6. *Zhu and Goldberg (2009) Given labeled data* $\{(x_1, y_1), \ldots, (x_l, y_l)\}$, *unlabeled data* $\{x_{l+1}, \ldots, x_{l+u}\}$, *a clustering method* $\mathcal{A}$, *a supervised method* $\mathcal{L}$

1. *Cluster* $\{x_1, ..., x_{l+u}\}$ *using* $\mathcal{A}$
2. *For each resulting cluster, let* $\mathcal{S}$ *be the set of labeled instances in this cluster*
   (a) ***If*** $\mathcal{S}$ *is non-empty* ***then*** *learn a supervised model from* $\mathcal{S}$, $f_{\mathcal{S}} = \mathcal{L}(\mathcal{S})$, *apply* $f_{\mathcal{S}}$ *to all unlabeled data in this cluster*
   (b) ***If*** $\mathcal{S}$ *is empty* ***then*** *use predictor* f *trained from all labeled data*

Return: labels on unlabeled data $\{y_{l+1}, ..., y_{l+u}\}$

We can use any clustering algorithm $\mathcal{A}$ and any supervised trainer $\mathcal{L}$. The intuition behind this idea is that: the labeled data helps identify clusters, assume that all classes appear in the labeled data and the number of labels is known. The assumption is inappropriate if undiscovered labels are hidden in the unlabeled part of data. But sometimes if such hidden labels are minor or less important, Cluster-then-Label can be used to make problem much easier with little sacrifice on accuracy. Experiments of this algorithm are very easy, unnecessary to show in the report.

2.6. Clustering via Bayesian Non-parametric

To handle the situation when the number of clusters K is unknown, Bayesian Non-parametric approaches are under our consideration because they do not make any presumption on K; in other words, K is assume to be infinite. One prominent approach is Dirichlet Process Gaussian Mixture Model (DPGMM).

Different from classical Gaussian Mixture Model, it assumes that data follows the DPGMM which has infinite components of Gaussian Mixtures, and the weights of components follow Dirichlet process priors. It does not require us to specify the number of components K at first. One of our goals is then to use the posterior distributions of K to find an optimal choice for clustering.

We assume that our data are from a mixture of infinite Gaussians

$$\sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k),$$

where K is infinite, π_k is weight of each component. Denote latent variables z_i as indicator showing which cluster point x_i belongs to. Moreover, DPGMM is inferred in Bayesian perspective. With regards to priors, we usually set μ_k follow Gaussian and Σ_k follow Inverse-Wishart distribution. The model structure is shown below:

$$\boldsymbol{\pi} \sim GEM(\eta)$$

$$\mu_k \sim \mathcal{N}(\mu_0, \Sigma_0)$$
$$\Sigma_k \sim Inv - Wishart(\Psi_0, \nu_0)$$
$$\boldsymbol{z}_i|\boldsymbol{\pi} \sim Categorical(\boldsymbol{\pi})$$
$$\boldsymbol{x}_i|\boldsymbol{z}_i = k, \mu, \Sigma \sim p_x(\cdot|\boldsymbol{z}_i = k, \mu, \Sigma)$$
$$k = 1, \ldots, K$$
$$i = 1, \ldots, N$$

where N is the number of whole data points, $\boldsymbol{\pi} \sim GEM(\eta)$ is equivalent to

$$\beta_k \sim Beta(1, \eta) \forall k$$

$$\pi_k = \beta_k \prod_{i=1}^{k-1} (1 - \beta_i)$$

which is the *stick-breaking construction* definition Sethuraman (1994) of Dirichlet Process. To draw inference on this probabilistic model, we compute or estimate the posterior distribution

$$p(\boldsymbol{z}_{1:N}, \mu, \Sigma, \boldsymbol{\pi}|\boldsymbol{x}_{1:N})$$

using Markov Chain Monte Carlo methods, e.g., Gibbs sampling, etc, or variational inference to approximate. However, it in fact wastes the information of labeled data carried if the DPGMM is directly applied on the whole semi-labeled data. Because of densities of latent variables $\boldsymbol{z}_i$, $\boldsymbol{z}_i|\boldsymbol{\pi} \sim Categorical(\boldsymbol{\pi})$, it basically indicates that all data points are treated in the same way, whether they are labeled or unlabeled. Thus the valued labels are not incorporated.

To utilize the information provided by the limited labeled data, we need to consider the statistical dependence between hidden random variables z_n with the same label: $\boldsymbol{z}_n$ is neighbor of $\boldsymbol{z}_m \Longleftrightarrow \ell_n = \ell_m$, the notion introduced by Hidden Markov Random Field (HMRF) Basu, Banerjee and Mooney (2004). To create statistical dependencies between latent variables $\boldsymbol{z}_{1:N}$, each connected pair (z_n, z_m) is associated with a factor $e^{-\lambda\nu(\boldsymbol{z}_n, \boldsymbol{z}_m)}$

$$e^{-\lambda\nu(\boldsymbol{z}_n, \boldsymbol{z}_m)} \begin{cases} = 1 & \text{if } \boldsymbol{z}_n = \boldsymbol{z}_m, \\ < 1 & \text{if } \boldsymbol{z}_n \neq \boldsymbol{z}_m. \end{cases} \tag{2.9}$$

Such statistical dependency is used to construct the joint prior distribution of $\boldsymbol{z}_{1:N}$, different from the $\boldsymbol{z}_i|\boldsymbol{\pi}$ ~Categorical($\boldsymbol{\pi}$) in DPGMM. It is such different joint distribution of $\boldsymbol{z}_{1:N}$ incorporates the information of labeled data, and characterizes Semi-DPGMM model Echraibi et al. (2019) proposed to handle clustering with partially labeled data. The core of Semi-DPGMM is shown as following:

$$\boldsymbol{\pi} \sim GEM(\eta) \tag{2.10}$$

$$\mu_k \sim \mathcal{N}(\mu_0, \Sigma_0) \tag{2.11}$$

$$\Sigma_k \sim Inv - Wishart(\Psi_0, \nu_0) \tag{2.12}$$

$$\boldsymbol{z}_{1:N}|\boldsymbol{\pi} \sim p_{\boldsymbol{z}_{1:N}}(\cdot|\boldsymbol{\pi}) \tag{2.13}$$

$$\begin{gathered}\boldsymbol{x}_i|\boldsymbol{z}_i = k, \mu, \Sigma \sim p_x(\cdot|\boldsymbol{z}_i = k, \mu, \Sigma) \\ k = 1, \ldots, K \\ i = 1, \ldots, N\end{gathered} \tag{2.14}$$

where the joint density of latent variables $\boldsymbol{z}_{1:N}$ is

$$p_{\boldsymbol{z}_{1:N}}(\cdot|\boldsymbol{\pi}) = \frac{1}{\Gamma}\prod_{n=1}^{N}\boldsymbol{\pi}_{\boldsymbol{z}_n}^{1[\mathcal{N}_n=\emptyset]}\prod_{n\sim m}e^{-\lambda\nu(\boldsymbol{z}_n,\boldsymbol{z}_m)}$$

Γ is the normalization constant, $\mathcal{N}_n$ is neighborhood of $\boldsymbol{z}_n$, $n \sim m$ means $\boldsymbol{z}_n$ is neighbor of $\boldsymbol{z}_m$.

The intuition behind this joint density $p_{\boldsymbol{z}_{1:N}}(\cdot|\boldsymbol{\pi})$ is that: take $\nu(\boldsymbol{z}_n, \boldsymbol{z}_m)$ as KL divergence of densities of two clusters, or say, as a "distance". If $\boldsymbol{z}_n = \boldsymbol{z}_m$, then $\nu(\boldsymbol{z}_n, \boldsymbol{z}_m) = 0$, then $e^{-\lambda\nu(\boldsymbol{z}_n,\boldsymbol{z}_m)} = 1$; if $\boldsymbol{z}_n \neq \boldsymbol{z}_m$, then $\nu(\boldsymbol{z}_n, \boldsymbol{z}_m) > 0$, then $e^{-\lambda\nu(\boldsymbol{z}_n,\boldsymbol{z}_m)} < 1$, thus data log-likelihood gets smaller, and $\nu(\boldsymbol{z}_n, \boldsymbol{z}_m)$ plays a role of penalty. So neighboring data points should have the same label, otherwise the log-likelihood of the data is penalized by a factor $\nu(\boldsymbol{z}_n, \boldsymbol{z}_m)$. To make inference on this Bayesian model, posterior $p(\boldsymbol{z}_{1:N}, \mu, \Sigma, \boldsymbol{\pi}|\boldsymbol{x}_{1:N})$ is estimated by variational inference. In this way, the Semi-DPGMM uses the information contained in the partial labeled data to help identify the number of clusters and guide the clustering. But Semi-DPGMM is not versatile under any circumstances, it has constraint as well. To apply it, data needs to have a structure of heterogeneity. And if some minor clusters are less important to detect, Semi-DPGMM make things complicated while good results are not guaranteed either. The experiments on KEEL data sets are given in section 3.2.2.

3. Experiments result

3.1. Semi-supervised learning vs. supervised learning

We applied the four algorithms to the KEEL data set, with different version of labeled proportions. We write first three algorithms using basic python package *Numpy*, the Label spreading algorithm is already implemented in *sklearn* Pedregosa et al. (2011).

Semi-supervised mixture model vs. Discriminant Analysis The results of Discriminant Analysis and Semi-supervised Gaussian Mixture Model are shown in the table below.

Table 3. Test error of Gaussian Mixture Model for 10%, 20%, 30%, 40% labeled data set

Data set	10% Super	10% Semi-super	20% Super	20% Semi-super	30% Super	30% Semi-super	40% Super	40% Semi-super
appendicitis	21.7%	20.75%	24.53%	22.64%	24.53%	18.87%	22.64%	17.92%
banana	44.68%	41.51%	44.42%	39.0%	44.43%	37.58%	44.21%	37.25%
cleveland	46.13%	37.37%	46.13%	45.45%	46.13%	44.78%	46.13%	43.43%
ecoli	56.85%	53.27%	57.44%	55.06%	57.14%	50.0%	57.14%	49.7%
glass	64.49%	70.09%	63.08%	65.89%	63.55%	44.86%	63.08%	55.61%
iris	66.67%	3.33%	59.33%	2.67%	50.0%	3.33%	32.67%	2.67%
led7digit	89.6%	61.8%	85.0%	38.4%	66.6%	42.0%	64.0%	26.2%
pima	30.6%	34.24%	33.2%	32.55%	33.33%	29.56%	33.46%	28.52%
titanic	32.08%	32.08%	32.08%	32.08%	32.08%	22.67%	32.08%	22.67%
wine	59.55%	6.18%	33.71%	3.37%	48.88%	0.56%	12.92%	0.56%

As we can see, the semi-supervised algorithm outperform the supervised algorithm in most cases. We can plot the test errors of 4 data sets as illustration.

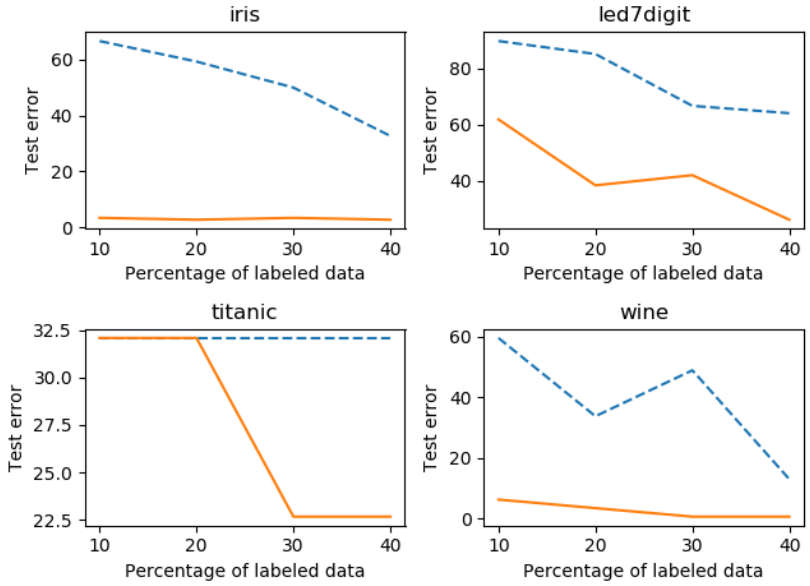

Figure 1: Test error (in percentage) of supervised (blue line) and semi-supervised method (orange line)

As we comment in the methodology part, the semi-supervised Mixture Model is good

if the data appears to have the Gaussian Mixture structure, which is true for *Iris* and *Wine* data set.

Self-training with k-NN Because the self-training and co-training are more efficient when the number of labeled data is relatively small, we choose and present the case where there are 10% labeled data only. It is noticeable that the classification error is not strictly decreasing as we keep adding predicted unlabeled data to labeled data, so for semi-supervised learning, we choose the model yielding the lowest classification error and then report the value of test error. This procedure is used in many semi-supervised learning papers Hady and Schwenker (2008), Triguero, García and Herrera (2015). As in the papers, we also compute the improvement of semi-supervised with respect to supervised method by

$$\text{Improvement} = \frac{\text{test error} - \text{semi-supervised test error}}{\text{test error}}$$

The result for self-training is shown below.

Table 4. Test error of Self-training for 10% labeled data set

Data set	Test super	Test semi-super	improvement
appendicitis	18.87%	16.98%	11.11%
banana	5.81%	5.72%	1.65%
cleveland	21.55%	21.55%	0.0%
ecoli	13.99%	13.39%	4.44%
glass	19.63%	19.16%	2.44%
iris	5.33%	4.0%	33.33%
led7digit	26.4%	27.4%	-3.65%
pima	20.96%	20.05%	4.55%
titanic	20.95%	21.67%	-3.35%
wine	5.06%	4.49%	12.5%

It can be seen that we get the improvement for most of the data sets except *led7digit* and *titanic* data set.

Co-trainning by committee: AdaBoost and Bagging With the similar spirit to Self-training, we apply the co-training algorithm on the data set where 10% of data are labeled and show the result.

Table 5. Test error of co-training with parameters set up as in Hady and Schwenker (2008)

Data set	Test Super	test Semi-super	Improvement
appendicitis	7.55%	8.4%	-12.82%
banana	15.19%	12.13%	20.14%
cleveland	23.3%	20.37%	12.47%
ecoli	12.47%	12.14%	2.52%
glass	16.03%	16.12%	-0.91%
iris	4.67%	4.67%	0.0%
led7digit	24.84%	24.4%	1.76%
pima	24.74%	21.74%	12.11%
titanic	21.08%	21.08%	0.0%
wine	8.31%	7.19%	12.68%

Label Spreading We also perform cross validation to choose the parameters for Label Spreading model and print the test error corresponding to best models.

Table 6. Test error of Label spreading method

Data set	10% labeled	20% labeled	30% labeled	40% labeled
appendicitis	19.81%	5.66%	3.77%	1.89%
banana	8.92%	8.57%	8.62%	8.47%
cleveland	17.95%	10.44%	3.7%	0.34%
ecoli	13.7%	8.63%	5.06%	2.68%
glass	23.36%	15.89%	11.21%	1.87%
iris	4.0%	1.33%	0.0%	0.0%
led7digit	23.6%	23.4%	21.8%	21.2%
pima	20.07%	13.8%	7.95%	4.82%
titanic	21.08%	20.95%	20.95%	20.95%
wine	20.11%	8.43%	3.93%	2.25%

It can be see that the test error is decreasing as the labeled proportion increasing. The result produced by four methods are similar. Although each method yields the best test errors for some data sets but the difference is not significant. In most cases, the semi-supervised learning performs better than supervised learning.

We can make a comment for the *led7digit* and *titanic* data set. The self-training and co-training performs poorly on those data set because most of the features there are binary, and noisy (*led7digit* is a "noisy signal" data set, and *titanic* tries to predict a person was alive from the titanic disaster from very basic information). Therefore the own predictions should not be good, which makes the algorithm add bad examples to the labeled set (contradictory to the assumption of those methods) and hurt the performance of the supervised learning algorithm.

3.2. clustering on partially labeled data with Semi-DPGMM model

3.2.1. evaluating clustering accuracy given true labels

Here we digress a little bit on how to evaluate accuracy of clustering results given the true cluster labels, which is very different from classification accuracy. Knowing the ground truth of cluster labels, to calculate the accuracy of our clustering methods, we have to find the best match between the cluster labels and true labels. The accuracy is defined as:

$$accuracy(y, \hat{y}) = \max_{permutation \in P} \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}(permutation(\hat{y}_i = y_i)$$

where P is the set of all permutations of $[1, \ldots, K]$, K is the number of clusters. For example, if true labels are [1,1,2,2,3,3,3], and our cluster labels are [2,2,1,1,3,4,4], then the accuracy is 1 - $1/7 = 6/7$. Additionally, there are $O(K!)$ permutations, but Hungarian Algorithm computes it in $O(K^3)$.

3.2.2. experiment Semi-DPGMM on partially labeled data

We make experiments on KEEL data sets, each with 4 labeled versions. Note here we use data sets in a different way from previous experiments. Since it is clustering here, we do not partition data into train and test parts. We deal with the whole data set, and 10%, 20%, 30%, 40% whole data are labeled respectively.

Table 7. Clustering error of Semi-DPGMM method

Data set	10% labeled	20% labeled	30% labeled	40% labeled
appendicitis	19.81%	19.81%	19.81%	19.81%
banana	44.88%	43.12%	44.5%	42.56%
cleveland	76.57%	56.6%	53.67%	47.91%
ecoli	46.73%	41.9%	45.74%	38.15%
glass	53.27%	52.62%	49.77%	43.47%
iris	28%	2%	2.3%	1.8%
led7digit	53.3%	44.8%	33.14%	21.4%
pima	60.86%	54.71%	53.52%	45.57%
titanic	32.08%	22.67%	23.2%	22.8%
wine	68.20%	57.13.%	59.38%	16.35%

This Semi-DPGMM approach is very computationally expensive, especially with large scale data sets or the number of classes is large. As shown in Table 7, clustering performance does not get improved with increasing labeled proportion on some data sets, e.g.: appendicitis, banana,etc. After observation, we find that these data sets usually have only two or three classes which already appeared in labeled part. Thus it is unnecessary to use the complicated Semi-DPGMM while simple Cluster-then-Label suffices enough. But it again leads to the core problem that the number of clusters K cannot be set

in advance, so it is a trade-off. Moreover, we think that these data sets themselves do not have good heterogeneous structures, so Bayesian non-parametric methods may be inappropriate for them. For Cleveland data, it is almost like random guess with 10% labeled, while error rate decreases with 40% labeled; the labeled part does contribute a lot, but the 47.91% clustering error rate still makes it an annoying result. It is difficult for us to explain why Cleveland data is inappropriate for Semi-DPGMM. Some data sets have steadily improved clustering results with increasing labeled data, e.g., ecoli, pima, and glass. It justifies that limited labels help the clustering, although the improvements are not that remarkable. Regarding to iris and wine data sets, Semi-DPGMM shows outstanding performance and clustering errors decrease significantly with more labeled data. It indicates that the information carried by partial labeled data is well leveraged. We think that heterogeneity of such data sets is one of the main factors resulting in the magnificent performance of Semi-DPGMM.

4. Conclusion and Discussion

Semi-supervised learning is a good learning paradigm to use in the case when we have both labeled and unlabeled data. Throughout the project, we have learnt several semi-supervised learning algorithms and witnessed that they performed better than supervised and unsupervised learning in most cases. We are also able to explain in the cases it performs worse. Although we could not reproduce the S3VM (semi-supervised support vector machine) as proposed (we tried to program it but the it did not work well and we could not explain), it turned out to be the the chance for us to learn more about graph based method in semi-supervised learning (Label Spreading). For the semi-supervised clustering part, the Semi-DPGMM shows pleasant performance on some data sets, but unsatisfying on some other data, where simple methods can even do better. No single method is dominant in semi-supervised clustering, and various types of data have different methods matching themselves well. In the report, we cover almost all the basics about Semi-supervised classification and clustering, and connect them to the algorithms we have learned on class. It will be a good resource for introducing to semi-supervised learning and help us to study more sophisticated semi-supervised learning algorithms in the future.

Contribution:

1. Trong Dat Do: Research question 1, section 2.1-4, section 3.1 (Semi-supervised classification)
2. Ziyi Song: Research question 2, section 2.5-6, section 3.2 (Semi-supervised clustering)

References

Basu, S., Banerjee, A. and Mooney, R. J. (2004). Active semi-supervision for pairwise constrained clustering. In *Proceedings of the 2004 SIAM international conference on data mining* 333–344.

Bengio, Y., Delalleau, O. and Roux, N. (2006). Label Propagation and Quadratic Criterion. *Semi-Supervised Learning*.

Brown, G., Wyatt, J., Harris, R. and Yao, X. (2005). Diversity Creation Methods: A Survey And Categorisation. *Information Fusion* **6** 5-20.

Echraibi, A., Flocon-Cholet, J., Gosselin, S. and Vaton, S. (2019). Bayesian Mixture Models For Semi-Supervised Clustering Technical Report, Orange Labs IMT Atlantique, France.

Hady, M. F. A. and Schwenker, F. (2008). Co-training by committee: a new semi-supervised learning framework. In *2008 IEEE International Conference on Data Mining Workshops* 563–572. IEEE.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. and Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* **12** 2825–2830.

Sethuraman, J. (1994). A constructive definition of dirichlet priors. *Statistica sinica* 639-650.

Triguero, I., García, S. and Herrera, F. (2015). Self-labeled techniques for semi-supervised learning: taxonomy, software and empirical study. *Knowledge and Information systems* **42** 245–284.

Zhou, D., Bousquet, O., Lal, T. N., Weston, J. and Schölkopf, B. (2004). Learning with local and global consistency. In *Advances in neural information processing systems* 321–328.

Zhu, X. J. (2005). Semi-supervised learning literature survey Technical Report, University of Wisconsin-Madison Department of Computer Sciences.

Zhu, X. and Goldberg, A. B. (2009). Introduction to semi-supervised learning. *Synthesis lectures on artificial intelligence and machine learning* **3** 1–130.

Zhu, X. and Ghahramani, Z. Learning from labeled and unlabeled data with label propagation.