Weakly Supervised Learning Brief Review

Lizi Chen

The way to create something beautiful is often to make subtle tweaks to something that already exists, or to combine existing ideas in a slightly new way. – "Hackers & Painters"

Motivation: Teammates ain't got enough time for the variations of algorithms in this field.

1 Semi-supervised Learning for Generative Model [1]

1.1 Supervised Generative Model vs. Semi-supervised Generative model

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 distribution (example in this case), parameterized by μ^i and Σ.

For **supervised generative model**: with $P(C_1), P(C_2), \mu^1, \mu^2, \Sigma$, we can have

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

For **semi-supervised generative model**: the unlabeled data x^u can help re-estimate $P(C_1), P(C_2), \mu^1, \mu^2, \Sigma$.

Steps:
Initialize model params: $\theta = P(C_1), P(C_2), \mu^1, \mu^2, \Sigma$

1. E-step, compute the posterior probability of unlabeled data:

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

2. M-step, update model

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

N is total number of examples.
N_1 is number of examples belonging to C_1

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

Maximum likelihood with labeled data:

$$\log L(\theta) = \sum_{x^r} \log P_\theta(x^r, \hat{y}^r), P_\theta(x^r, \hat{y}^r) = P_\theta(X^r|\hat{y}^r)P(\hat{y}^r)$$

Maximum likelihood with labeled and unlabeled data:

$$\log L(\theta) = \sum_{x^r} \log P_\theta(x^r, \hat{y}^r) + \sum_{x^u} \log P_\theta(x^u)$$

, where

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

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

1.2 Low-density separation

Given labelled data set $(x_r, \hat{y}_r)_{r=1}^{R}$ and unlabeled data set $x_u{}_{u=1}^{R+U}$, train model f_θ from labeled data set, apply f_θ to the unlabeled data set to obtain psudo-label for the unlabeled data.
Similar to active learning.

1.3 Entropy-based Regularization

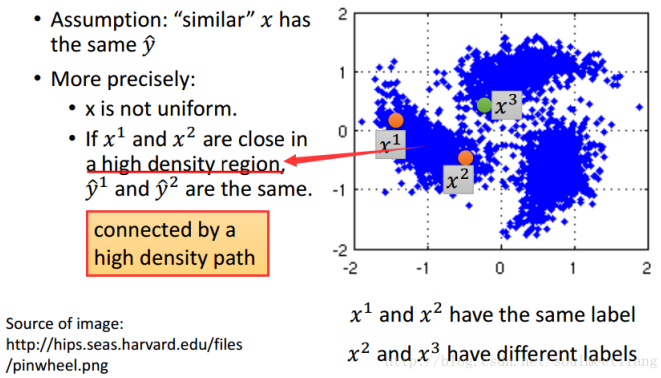
Evaluate how concentrate the distribution y_u is.

$$Entropy(y_u) = -\sum_{m=1}^{S} y_m log(y_m)$$

1.4 Semi-supervised SVM

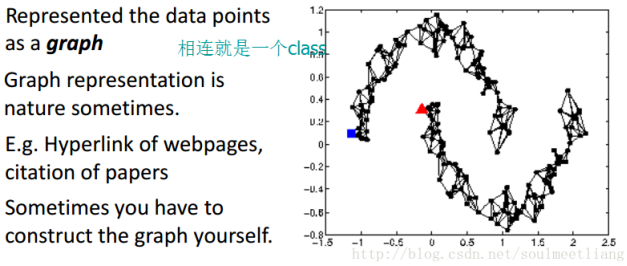
- Enumerate all possible labels for the unlabeled data.
- Find the boundary that can provide the largest margin and least error.
Source: Thorsten Joachims, "Transductive Inference for Text Classification using Support Vector Machines", ICML, 1999.

1.5 Smoothness Assumption



1.5.1 Clustering then Labeling

Such approach requires obvious class separation.



1.5.2 Graph-based Approach

Example: Label-Propagation (Details in the later section)
Define similarity $s(x_i, x_j)$.
Edge weight between x_i and x_j is proportional to $s(x_i, x_j)$.
For example: Gaussian Radial Basis Function (rbf):

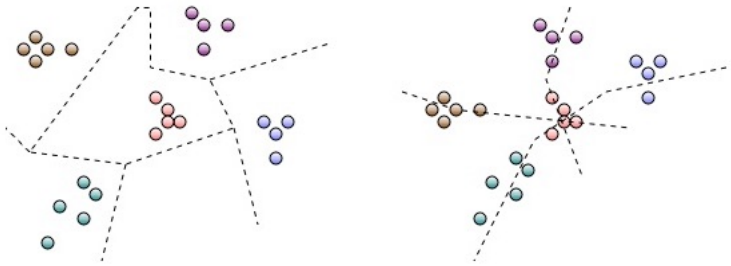
$$s_{rbf}(x_i, x_j) = e^{-\gamma||x_i - x_j||^2}$$

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

2 Appendix: Brief Review of Weakly Supervised Learning [2]

Problem of naive unsupervised clustering: i.e., K-means; can go wrong, when boundary go through clusters.

Figure 1: Left: Ground truth. Right: Possible bad result.



TODO: add proof example of failure.

2.1 Incomplete Supervision

2.1.1 Active Learning

Active Learning vs. Random Sampling
Active Learning can be more efficient than passive learning.

Two Types of Active Learning: Stream-based and Pool-based
Both types are used interchangeably; however, stream-based active learning is more realistic and pool-based is more widely used practically.
Stream-based active learning have new data comes in asking whether to label it or not. The trained machine labels it if it can classify; otherwise, an oracle(i.e., human, domain expert) has to present to decide the labeling. In this case, each data is inspected, either by an machine or human.
The **pool-based** active learning; as a post-hoc analysis, does not necessarily look at each data points. A pool of unlabeled dataset is given and the machine ranks all data by; for example, its informativeness. A human annotator then decide the labels of the chosen data points.

2.2 Selection Strategies:

2.2.1 Uncertainty Sampling

To select examples which the current model θ is the most uncertain about.
Measure of uncertainty using label probability: $P - \theta(y|\hat{x})$

1. Least Confident (LC):
$$x^*_{LC} = \arg\max_{x \in R^d}(1 - P_\theta(\hat{y}|x))$$
where $\hat{y}$ is the most probable label for x under the current model θ

2. Smallest Margin (SM):
$$x^*_{SM} = \arg\min_{x \in R^d}\big(P_\theta(y_i|x) - P_\theta(y_j|x)\big),\ i \neq j$$
y_i and y_j are the **two most probable labels** for x under the current model.

3. Label Entropy - to choose the data points whose label entropy is maximum:

$$x^*_{LE} = \arg\max_{x \in R^d} \sum P_\theta(y_i|x) \log P_\theta(y_i|x)$$

Figure. 2 shows that by using uncertainty sampling, the 30 labeled instances are only the 'hard' cases, and this results in much better accuracy. 'Hard' cases, means that the data instances in the space that are the most difficult to distinguish.

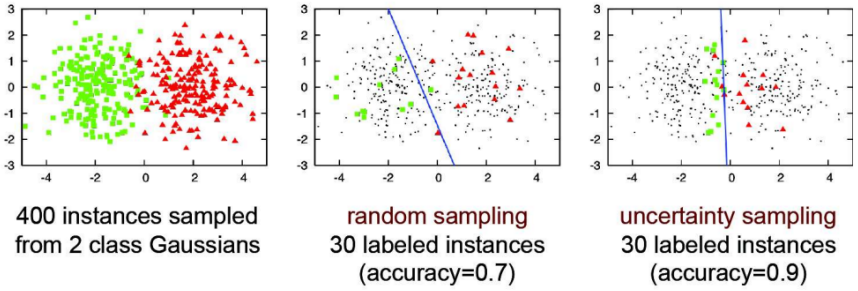


Figure 2: Random Sampling vs. Uncertainty Sampling

2.2.2 Query-By-Committee (QBC)

Many different classifiers, called **Committee**/

- C: Number of classes.
- $\mathcal{L} = \theta^1, \ldots, \theta^C$: A list(committee) of models.
- $\mathcal{D}_L$: Labeled data.
- $\mathcal{D}_U$: unlabeled data.

All models $\mathcal{L}$ vote for their predictions on $\mathcal{D}_U$. The examples that have maximum disagreement across all the models are chosen for labeling. One example of measuring the disagreement is called **Vote Entropy**:

$$x^*_{VE} = \arg\max_{x \in \mathcal{D}_U} \sum \underbrace{-\frac{V(y_i)}{C} \log \frac{V(y_i)}{C}}_{Entropy}$$

,where $V(y_i)$ is the number of votes received to label y_i.

2.2.3 Note: Effect of Outliner Examples:

An outlier can be very 'informative' according to the probabilistic model; however, such outlier examples will be useless/misleading.
Instead of using the confidence of model on an example, see how a labeled example affects the model itself. The data that affects the model the most is probably the most informative one.

2.2.4 Expected Model Change

Select the unlabeled data points whose inclusion brings the maximum change in the model. For example, the gradient of the loss function; after adding the unlabeled data (after its been labeled), has the greatest change.

2.2.5 Expected Error Reduction

Select the unlabeled data points that reduces the **expected generalization error** the most:

$$\mathbf{R}(x) = \sum_{\mathcal{D}_U} \mathbb{E}_y \Big[\mathbb{H}_{\theta + \langle x,y \rangle} [Y | \mathcal{D}_U] \Big]$$

2.2.6 Variance Reduction

Select the unlabeled data points that reduces the model variance by the most.
Narrower confidence interval for the model's parameters.

2.2.7 Density Weighting

Weight the informativeness of unlabeled data by its average similarity to the entire unlabeled data pool. This way, the outlier will not get much weight.

3 Semi-Supervised Learning

Generative Methods

Graph-based Methods
Example method: Label Prorogation:

1. Step 1. Build the Affinity Matrix $\mathbf{P}$.

$$\mathbf{P} = \begin{bmatrix} p_{11} & p_{12} & \dots & \dots & p_{1n} \\ p_{21} & p_{22} & \dots & \dots & p_{2n} \\ \vdots & \vdots & \ddots & & \vdots \\ \vdots & \vdots & & \ddots & \vdots \\ p_{n1} & p_{n2} & \dots & \dots & p_{nn} \end{bmatrix}$$

   , where

$$p_{ij} = \frac{D_{ij}}{D_{i1} + D_{i2} + \ldots + D_{ik}} = \frac{D_{ij}}{\sum_{k=1}^{n} D_{ik}}$$

2. Step 2. Build the Label Function $\mathbf{F}$ (Label Index Matrix).
   We have $\mathcal{D}_L$ for the labeled data, and $\mathcal{D}_U$ for the unlabeled. We do vertical stack:

```
// Stack two numpy.ndarray
MatX = np.vstack((Matrix_Labeled, Matrix_Unlabeled))
```

$$\begin{array}{c|cc} & Class_A & Class_B \\ \hline d^l_{A1} & 1 & 0 \\ d^l_{A2} & 1 & 0 \\ d^l_{A3} & 1 & 0 \\ & \dots & \dots \\ d^l_{B1} & 0 & 1 \\ d^l_{B2} & 0 & 1 \\ d^l_{B3} & 0 & 1 \\ & \dots & \dots \\ d^u_i & -1 & -1 \\ d^u_j & -1 & -1 \\ d^u_k & -1 & -1 \end{array}$$

, where
d_A^l is a labeled data point for $Class_A$
d_B^l is a labeled data point for $Class_B$
d^u is a unlabeled data point.

3. Step 3. Propagation and Clamping: Eventually, we look at the Label Function $\mathbf{F}$ to see which class has a higher score. Thus, to do the propagation, we have:

$$\mathbf{F}' = \mathbf{P} \cdot \mathbf{F}$$

However, we want to persist $\mathcal{D}_L$, therefore we do the "clamping" process:

```
// Reset the data entries that are pre-labeled.
F[0: Matrix_Labeled.shape[0]] = Matrix_Labeled
```

Proof of Convergence:

Low-Density Separation

Disagreement-based Methods

3.1 Inexact Supervision

3.2 Inaccurate Supervision

4 Examples:

4.1 A Perceptron Based Active Learner:[4]

Based on **Selective Sampling** (looking at one example at a time)

Input: Parameter $b > 0$ (dictates how aggressively we want to query labels)
Initialization: $w = [0, 0, \cdots, 0]$
for $n = 1 : N$ **do**
  Get x_n, compute $p_n = w^T x_n$
  Predict $\hat{y}_n = sign(p_n)$
  Draw Bernoulli r.v. $Z \in 0, 1$ with probability $= \frac{b}{b+|p_n|}$
  if $Z == 1$ **then**
    query the true label y_n
    if $y_n \neq \hat{y}_n$ **then**
      update w
    else
      do not update w
    end
  else
    when $Z == 0$, ignore the example x_n and do not update w
  end
end

Algorithm 1: Perceptron-Based Active Learner

References

[1] "CSDN Blog, Semi-Supervised Learning note," available at https://blog.csdn.net/soulmeetliang/article/details/73251790.

[2] "Digging into data: Active learning," available at https://www.youtube.com/watch?v=8Jwp4_WbRio&list=PLZl9WvLDeMPVnCnAlpjkDV5uC-4FDpJ5p&index=44&t=0s.

[3] O. Chapelle, B. Schlkopf, and A. Zien, *Semi-Supervised Learning*, 1st ed. The MIT Press, 2010.

[4] P. Rai, "Active learning, cs5350/6350: Machine learning," available at https://www.cs.utah.edu/~piyush/teaching/10-11-print.pdf.

[5] X. Zhu, J. Lafferty, and R. Rosenfeld, "Semi-supervised learning with graphs," Ph.D. dissertation, Carnegie Mellon University, Language Technologies Institute, School of Computer Science, 2005.