Practical Session 2

Semi-Supervised Learning

Graphs in Machine Learning
MVA Master Program – ENS Paris-Saclay

omar.darwiche-domingues@inria.fr
pierre.perrault@inria.fr

About this Session

All the code related to the TD must be submitted along with a written report.

1 Harmonic Function Solution (HFS)

Semi-supervised learning (SSL) studies learning from both labeled and unlabeled examples. This paradigm is useful for real-world problems where data is abundant but labeling resources are limited.

1.1 Notation

- $G = (V, E)$ is a weighted graph where $V = \{x_1, \ldots, x_n\}$ is the vertex set and E is the edge set
- Each edge $e_{ij} \in E$ has weight w_{ij}. If no edge exists, $w_{ij} = 0$.
- Each node has label $y_i \in \mathbb{R}$.
- Only subset $S \subset V$, $|S| = l$ of node labels are revealed; the remaining $u = n - l$ nodes are in subset $T = V \backslash S$.

We wish to predict values of vertices in T by exploiting graph structure. Since we believe close nodes (similar) should share similar labels, each node should be surrounded by nodes with the same label.

For recovered labels encoded in vector $f \in \mathbb{R}^n$:

$$f_i = \frac{\sum_{i \sim j} f_j w_{ij}}{\sum_{i \sim j} w_{ij}} \tag{1}$$

where $f_i = f(x_i)$.

1.2 Random Walk Interpretation

If weight w_{ij} expresses moving tendency from x_i to x_j, transition probabilities are:

$$P(j|i) = \frac{w_{ij}}{\sum_k w_{ik}} \tag{2}$$

A stationary distribution gives a valid solution.

1.3 Smoothness Preference

This can be expressed as a penalty on non-smooth solutions using Laplacian L:

$$\Omega(f) = \sum_{i \sim j} w_{ij}(f_i - f_j)^2 = f^T L f \tag{3}$$

Initially, we assume labeled data is always correct and enforce labels exactly. This promotes smoothness on unlabeled points while guaranteeing correct labels:

$$\min_{f \in \mathbb{R}^n} \sum_{i,j=1}^{n} w_{ij} \left(f(x_i) - f(x_j)\right)^2$$
$$s.t. \quad y_i = f(x_i) \quad \forall i = 1, \dots, l$$

Questions

1. Complete `hard_hfs` and `two_moons_hfs`. Select uniformly 4 labels (S), compute labels for unlabeled nodes (T) using hard-HFS. Plot resulting labeling and accuracy.
2. At home, run `two_moons_hfs` using `data_2moons_large.mat` (1000 samples). Sample only 4 labels. What can go wrong?

1.4 Soft-HFS

When labels are noisy or some samples are mislabeled, relabeling might be beneficial. Soft-HFS balances smoothness and satisfying training labels.

Define C and y as:

$$C_{ii} = \begin{cases} c_l & \text{for labeled examples} \\ c_u & \text{otherwise} \end{cases} \qquad y_i = \begin{cases} \text{true label} & \text{for labeled examples} \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

Soft-HFS objective:

$$\min_{f \in \mathbb{R}^n} (f - y)^\top C (f - y) + f^\top L f \tag{5}$$

Questions

1. Complete `soft_hfs` and test with `two_moons_hfs`. Complete `hard_vs_soft_hfs`. Compare soft-HFS and hard-HFS results.

2 Face Recognition with HFS

We apply HFS to face recognition—classifying faces as different persons. Since faces share common features, we leverage large quantities of unlabeled data to improve accuracy.

Complete `offline_face_recognition` to classify faces and plot results.

Questions

1. How did you manage to label more than two classes?
2. Which preprocessing steps (e.g., `cv.GaussianBlur`, `cv.equalizeHist`) did you apply before constructing the similarity graph? Which gave best performance?
3. Does HFS reach good performance on this task?

2.1 Dataset Augmentation

Try augmenting the dataset with more unlabeled data. In `extended_dataset.tar.gz` you'll find additional pictures to expand the dataset.

Modify `offline_face_recognition_augmented` for your extended dataset:

Questions

1. Did adding more data improve performance? If so, which kind?
2. If performance doesn't improve, justify why. Which additional data degrades performance instead of improving it?

3 Online SSL

SSL is designed for problems where collecting large supervised training data is difficult, but obtaining unlabeled samples from the same process is inexpensive.

In stream processing, few labeled examples are provided initially to set the system bias while unlabeled examples are gathered online and update the bias continuously.

3.1 Computational Challenges

In online learning, after the game starts we don't observe any more true labels y_t. To adapt to environment changes, we rely on indirect feedback like data structure.

When t becomes large, naive hard-HFS (recomputing from scratch) has prohibitive costs. In streaming settings with near real-time requirements, we need scalable time and memory costs.

Since most HFS operations scale with nodes, subsampling is effective: compute approximate solutions on smaller subsets that generalize well.

Incremental k-centers **?** guarantees on distortion allow us to provide theoretical guarantees.

Algorithm 1 Incremental k-centers

Input: unlabeled x_t, centroids C_{t-1}, multiplicities v_{t-1}, taboo list b
if $(|C_{t-1}| = k)$ **then**
  $c_1, c_2 \leftarrow$ two closest centroids with at least one not in b
  // Decide c_{rep} (represents both) and c_{add} (represents x_t)
  if c_1 in b **then**
    $c_{\text{rep}} \leftarrow c_1$, $c_{\text{add}} \leftarrow c_2$
  else if c_2 in b **then**
    $c_{\text{rep}} \leftarrow c_2$, $c_{\text{add}} \leftarrow c_1$
  else if $v_{t-1}(c_2) \leq v_{t-1}(c_1)$ **then**
    $c_{\text{rep}} \leftarrow c_1$, $c_{\text{add}} \leftarrow c_2$
  else
    $c_{\text{rep}} \leftarrow c_2$, $c_{\text{add}} \leftarrow c_1$
  end if
  $v_t \leftarrow v_{t-1}$
  $v_t(c_{\text{rep}}) \leftarrow v_t(c_{\text{rep}}) + v_t(c_{\text{add}})$
  $c_{\text{add}} \leftarrow x_t$, $v_t(c_{\text{add}}) = 1$
else
  $C_t \leftarrow C_{t-1}.\text{append}(x_t)$
  $v_t \leftarrow v_{t-1}.\text{append}(1)$
end if

Algorithm 2 Online HFS with Graph Quantization

Input: t, centroids C_t, multiplicities v_t, labels y
$V \leftarrow \text{diag}(v_t)$
$[\widetilde{W}_q]_{ij} \leftarrow$ weight between centroids i and j
Compute Laplacian L of graph represented by $W_q = V\widetilde{W}_q V$
// Infer labels using hard-HFS
$\widehat{y}_t \leftarrow \text{hardHFS}(L, y)$
// Remark: x_t is always present in reduced graph and doesn't share centroids

3.2 Practical Considerations

Implementation

- Labeled nodes are fundamentally different. Keep them separate, never merge. Taboo list b tracks nodes that cannot be merged.
- In streaming, it's preferable to pay small cost at every step for smooth execution. Centroids update at every step.
- When a new node arrives with too many centroids, we choose two closest centroids c_{add} and c_{rep}. c_{add} forgets the old centroid and points to new sample; c_{rep} represents all nodes that belonged to c_{add}.

Use `create_user_profile` in `helper_online_ssl.py` to capture training set of your face and someone else's. Faces are preprocessed and saved in `data/faces`, loaded by `online_face_recognition`.

Questions

1. Complete `online_ssl_update_centroids` using Algorithm 1.
2. Complete `online_ssl_compute_solution` following Algorithm 2.
3. Read `preprocess_face` in `helper_online_ssl.py` and run `online_face_recognition`. Include resulting frames (not too similar) showing faces correctly labeled, commenting on implementation choices.
4. What happens if an unknown person's face is captured? Modify code to disregard unrecognizable faces. Include frames showing unknown faces correctly labeled as unknown.