I

Where to draw decision boundary?

(a) + -
(b) + -

Ans: (a)

II

How to label?

(a) + -
(b) - +

Ans: ?

III

Now where would you draw decision?

(a) + -
(b) + -

Ans: (b)

Compare condition I and III. Which one is better?

Goal of semi-supervised learning:
Use both labeled & unlabeled data to build a classifier that's better than what can be achieved using either data alone.
**Motivations for SSL**

- Labeled data is expensive to obtain, so there is never enough.
- Unlabeled data is cheap and plentiful.

![Graph](image)

**Note:**

1. Unlabeled data may not always help. Sometimes you may do worse.
2. The usefulness of unlabeled data may depend on how much labeled data is available.
3. Different SSL algorithms make different assumptions about how unlabeled data can be useful. This assumption must match the problem in order for SSL to work.

**Preliminaries**

Labeled data: \( \{(X_i, y_i)\}_{i=1}^{L} \) \( \text{Sample } X_i \in \mathbb{R}^d \)

Unlabeled data: \( \{(X_i)\}_{i=L+1}^{L+U} \)

We learn a function \( f(x) \) by minimizing some loss function on the data: \( f = \arg\min_{f} L(\hat{f}, \{X_i, y_i\}_{i=1}^{L}, \{X_i\}_{i=L+1}^{L+U}) \)

Today we'll talk about two major classes of SSL algo:

- Low Density Separation
- Graph-based methods

There are others (e.g., EM, Co-training)
<table>
<thead>
<tr>
<th>Inductive</th>
<th>Transductive</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Goal</strong></td>
<td></td>
</tr>
<tr>
<td>Classifier $f(x)$ needs to predict on new test samples. i.e. $f$ is defined over all input space $\mathbb{R}^d$</td>
<td>Classifier $f(x)$ needs to predict on unlabeled samples given, i.e. $f$ is defined only on select points</td>
</tr>
<tr>
<td><strong>Usage</strong></td>
<td></td>
</tr>
<tr>
<td>Train once to get $f$. Can use $f$ anytime</td>
<td>Train once to get $\hat{f}$ on unlabeled samples. If there's new samples, need to retrain</td>
</tr>
<tr>
<td><strong>Analogy</strong></td>
<td>In-class exam</td>
</tr>
</tbody>
</table>

**Terminology:**

$\text{Inductive SSL} \overset{\text{SSL}}{\leftarrow} \text{Transductive Learning/Inference}$

Learning with partially labeled Data.

Transductive algo will give prediction only on these points. Inductive algo covers the whole space.
Low Density Separation

Transductive SVM / Semi-supervised SVM as primary example

Assumption: Unlabeled data from different classes are separated by large margin.

\[
\begin{align*}
+ & \quad - \\
+ & \quad - \\
\text{SVM} & \quad \text{TSVM}
\end{align*}
\]

Intuitive idea:
1. Enumerate all \( 2^n \) possible labeling of unlabeled data
2. Train SVM on \( U+L \) datapoints.
3. Pick the SVM w/ largest margin

Soft Margin SVM objective:

\[
\min_{\mathbf{w}} \|\mathbf{w}\| + \frac{1}{C} \sum_{i=1}^{n} \xi_i
\]

\[c_i y_i (\mathbf{w} \cdot \mathbf{x}_i) \geq 1 - \xi_i \quad \forall \ i \in \{1,...,L\}, \quad \xi_i \geq 0\]

\[\begin{cases} 
\text{if } y_i (\mathbf{w} \cdot \mathbf{x}_i) \geq 1 \text{ is satisfied, } \xi_i = 0 \\
\text{else, } \xi_i = 1 - y_i (\mathbf{w} \cdot \mathbf{x}_i) \end{cases}\]

\[
\Rightarrow \text{Hinge Loss}
\]

(4)
TSUM Objective

\[
\begin{align*}
\min_{w} & \quad \frac{1}{l} \sum_{i=1}^{l} \left( 1 - \frac{y_i (w x_i)}{\epsilon_i} \right) + C \sum_{i=1}^{l} \epsilon_i \\
\text{st.} & \quad y_i (w x_i) \geq 1 - \epsilon_i, \quad i = 1 \ldots l \\
& \quad \|w\|_1 \leq \frac{C}{\epsilon}, \quad \epsilon_i \geq 0.
\end{align*}
\]

\[\Rightarrow \text{HAT LOSS}\]

Unlabeled data will incur loss if it is too close to the decision boundary.

TSUM optimization is hard!

- Had loss is non-convex; Enumerating \(2^d\) labels is combinatorial.
- Many heuristics proposed (active area of research)
GRAPH-BASED METHODS

2.2
Not similar. OCR may fail

2.2.2.2
Similarity can be indirectly established by a chain

ASSUMPTION: SAMPLES THAT ARE LOCALLY CLOSE HAVE THE SAME LABEL. SAMPLES THAT ARE CONNECTED BY A CHAIN OF CLOSE SAMPLES ALSO HAVE THE SAME LABEL.

This similarity info is given in the form of a graph.

Each data point $X_i, i=1...n$ is a point. Edge represents strength of similarity: e.g. $E_{j,k} = \frac{1}{||X_j - X_k||}$
An algorithm: Label Propagation

\[ P_{ij} = P(i \rightarrow j) = \frac{e^{-\frac{i-j}{\sigma^2}}}{\sum_k e^{-\frac{i-k}{\sigma^2}}} \quad \text{Probability of transition from node } i \text{ to } j \]

\[ P \]

\[ (L+U) \times (L+U) \text{ transition matrix.} \]

\[ f \]

\[ (L+U) \times c \text{ prediction matrix.} \]

1. Propagate \( f \rightarrow Pf \)

2. Clamp labeled data \( f(X_{\text{labelled}}) = [0 \ 0 \ 1] \)

\[ \text{where } 1 \text{ occurs at class label} \]

3. Repeat until convergence

- There is also a closed-form solution \( f_0 = (I - Pu)^{T}Pu + Y \)

- We can view this as a random walk, starting at unlabeled node \( i \) and stopping until we hit a labeled node. \( f(i) \) represents the probability of each label for \( x_i \).

- We can also view this as applying kNN to each unlabeled point, where \( f(X_i) \) is a weighted avg of its neighbors, \( \text{ie. } \frac{1}{D_{ij} \sum_{j=1}^{n} w_{ij}} \).
Graph construction is essential to success

- In a sense, there's no learning in graph-based methods. It's just inference. So all knowledge is in the graph.

\[ \begin{align*}
\text{All neighbors graph (ensures full connectivity)} \\
\text{kNN graph (ensures k neighbors per node)} \\
\text{ε-neighborhood graph (some node has more neighbors)}
\end{align*} \]

\[ \begin{align*}
\text{Initial similarity measure: } & \frac{1}{||x_i - x_j||} \\
\text{Graph edge similarity score: } & \in [0, 1.5] \\
& \frac{1}{||x_i - x_j||} \\
& e\left(\frac{-||x_i - x_j||^2}{\sigma^2}\right)
\end{align*} \]

This is open research area! Not as investigated as it should be!
How to classify new test sample?

1. Rebuild graph. Rerun label propagation
2. Use nearest neighbor(s) to test sample.
   The neighbor may have initially been unlabeled. Ideally you get good cover of the whole space w/ unlabeled data.
3. Graph regularization approach.

Graph Regularization (Manifold Reg.)

Define $D$, diagonal matrix of $D_{ii} = \sum_{j} e_{ij}$

Graph Laplacian $L = D - W$ (This is the "un-normalized version. There's other definitions of $L$)

This has a nice property:

$$f^T L f = \frac{1}{2} \sum_{i<j} e_{ij} (f_i - f_j)^2$$

Proof: $f^T L f = f^T D f - f^T W f = \sum_{i} D_{ii} f_i^2 - \sum_{i<j} e_{ij} f_i f_j$

$$= \frac{1}{2} \sum_{i} (\sum_{j} D_{ij} f_j^2 - 2 e_{ij} f_i f_j + \sum_{j} e_{ij} f_j^2) = \frac{1}{2} \sum_{i,j} e_{ij} (f_i - f_j)^2$$

Objective: $\min_{f} f^T L f$

$s.t.$ $f$ is correct on labeled data.
The above objective ensures that \( f \) is smooth over the graph (i.e., for close points, \( f \) doesn’t vary much).

- We can even view \( L \) as a sort of precision matrix: \( q(x) = e^{-f^T f} \)

- New samples can be handled by assuming a functional form for \( f \):

\[
\min_w w^T w + c \sum_{i,j} e_{ij} (wx_i - wx_j)^2
\]

s.t. \( y_i f(wx) \geq 1 \)