Semi-supervised Learning

Piyush Rai

CS5350/6350: Machine Learning

November 8, 2011

Semi-supervised Learning

- Supervised Learning models require labeled data
- Learning a reliable model usually requires plenty of labeled data
- Labeled Data: Expensive and Scarce
- Unlabeled Data: Abundant and Free/Cheap
  - E.g., webpage classification: easy to get unlabeled webpages

Semi-supervised Learning

- Supervised Learning models require labeled data
- Learning a reliable model usually requires plenty of labeled data
- Labeled Data: Expensive and Scarce
- Unlabeled Data: Abundant and Free/Cheap
  - E.g., webpage classification: easy to get unlabeled webpages

- **Semi-supervised Learning:** Devising ways of utilizing unlabeled data with labeled data to learn better models

Semi-supervised Learning: Formally

- General Idea: Learning from both labeled and unlabeled data

Semi-supervised Learning: Formally

- General Idea: Learning from both labeled and unlabeled data
- Semi-supervised Classification/Regression
  - Given: Labeled training data $\mathcal{L} = \{\mathbf{x}_i, y_i\}_{i=1}^{L}$, unlabeled data $\mathcal{U} = \{\mathbf{x}_j\}_{j=L+1}^{L+U}$ (usually $U \gg L$)

Semi-supervised Learning: Formally

- General Idea: Learning from both labeled and unlabeled data
- Semi-supervised Classification/Regression
  - Given: Labeled training data $\mathcal{L} = \{\mathbf{x}_i, y_i\}_{i=1}^{L}$, unlabeled data $\mathcal{U} = \{\mathbf{x}_j\}_{j=L+1}^{L+U}$ (usually $U \gg L$)
  - Goal: Learning a classifier f better than using labeled data alone

Semi-supervised Learning: Formally

- General Idea: Learning from both labeled and unlabeled data
- Semi-supervised Classification/Regression
  - Given: Labeled training data $\mathcal{L} = \{\mathbf{x}_i, y_i\}_{i=1}^{L}$, unlabeled data $\mathcal{U} = \{\mathbf{x}_j\}_{j=L+1}^{L+U}$ (usually $U \gg L$)
  - Goal: Learning a classifier f better than using labeled data alone
- Semi-Unsupervised Learning
  - Given: Unlabeled data $\{\mathbf{x}_i\}_{1=1}^{N}$ and the goal could be to do clustering or dimensionality reduction. Additionally given: Some constraints on the data.

Semi-supervised Learning: Formally

- General Idea: Learning from both labeled and unlabeled data
- Semi-supervised Classification/Regression
  - Given: Labeled training data $\mathcal{L} = \{\mathbf{x}_i, y_i\}_{i=1}^{L}$, unlabeled data $\mathcal{U} = \{\mathbf{x}_j\}_{j=L+1}^{L+U}$ (usually $U \gg L$)
  - Goal: Learning a classifier f better than using labeled data alone
- Semi-Unsupervised Learning
  - Given: Unlabeled data $\{\mathbf{x}_i\}_{1=1}^{N}$ and the goal could be to do clustering or dimensionality reduction. Additionally given: Some constraints on the data.
    - E.g., for clustering: two points must be in the same cluster, or two points must not be in the same cluster; for dimensionality reduction: two points must be close after the projection

Semi-supervised Learning: Formally

- General Idea: Learning from both labeled and unlabeled data
- Semi-supervised Classification/Regression
  - Given: Labeled training data $\mathcal{L} = \{\mathbf{x}_i, y_i\}_{i=1}^{L}$, unlabeled data $\mathcal{U} = \{\mathbf{x}_j\}_{j=L+1}^{L+U}$ (usually $U \gg L$)
  - Goal: Learning a classifier f better than using labeled data alone
- Semi-Unsupervised Learning
  - Given: Unlabeled data $\{\mathbf{x}_i\}_{1=1}^{N}$ and the goal could be to do clustering or dimensionality reduction. Additionally given: Some constraints on the data.
    - E.g., for clustering: two points must be in the same cluster, or two points must not be in the same cluster; for dimensionality reduction: two points must be close after the projection
- **This class:** Semi-supervised Learning (SSL) will refer to Semi-supervised Classification/Regression

Semi-supervised Learning (SSL) vs Transductive Learning

- SSL: Given labeled training data $\mathcal{L} = \{\mathbf{x}_i, y_i\}_{i=1}^{L}$, unlabeled data $\mathcal{U} = \{\mathbf{x}_j\}_{j=L+1}^{L+U}$, learn a function f
- In SSL, f is used to predict labels for the future test data

Semi-supervised Learning (SSL) vs Transductive Learning

- SSL: Given labeled training data $\mathcal{L} = \{\mathbf{x}_i, y_i\}_{i=1}^{L}$, unlabeled data $\mathcal{U} = \{\mathbf{x}_j\}_{j=L+1}^{L+U}$, learn a function f
- In SSL, f is used to predict labels for the future test data
- This is called Inductive Learning (learning a function to be applied on test data). Semi-supervised learning is therefore inductive.

Semi-supervised Learning (SSL) vs Transductive Learning

- SSL: Given labeled training data $\mathcal{L} = \{\mathbf{x}_i, y_i\}_{i=1}^{L}$, unlabeled data $\mathcal{U} = \{\mathbf{x}_j\}_{j=L+1}^{L+U}$, learn a function f
- In SSL, f is used to predict labels for the future test data
- This is called Inductive Learning (learning a function to be applied on test data). Semi-supervised learning is therefore inductive.
- Transductive Learning: Given labeled training data $\mathcal{L} = \{\mathbf{x}_i, y_i\}_{i=1}^{L}$, unlabeled data $\mathcal{U} = \{\mathbf{x}_j\}_{j=L+1}^{L+U}$
- Transductive Learning: No explicit function is learned. We don't get some "future" test data. All we care about is the predictions for $\mathcal{U}$

Semi-supervised Learning (SSL) vs Transductive Learning

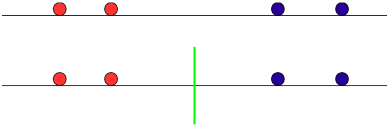
- SSL: Given labeled training data $\mathcal{L} = \{\mathbf{x}_i, y_i\}_{i=1}^{L}$, unlabeled data $\mathcal{U} = \{\mathbf{x}_j\}_{j=L+1}^{L+U}$, learn a function f
- In SSL, f is used to predict labels for the future test data
- This is called Inductive Learning (learning a function to be applied on test data). Semi-supervised learning is therefore inductive.
- Transductive Learning: Given labeled training data $\mathcal{L} = \{\mathbf{x}_i, y_i\}_{i=1}^{L}$, unlabeled data $\mathcal{U} = \{\mathbf{x}_j\}_{j=L+1}^{L+U}$
- Transductive Learning: No explicit function is learned. We don't get some "future" test data. All we care about is the predictions for $\mathcal{U}$
- Transductive Learning: The set $\mathcal{U}$ is the test data and is available at the training time

Why/How Might Unlabeled Data Help?

- Red: $+1$, Dark Blue: -1

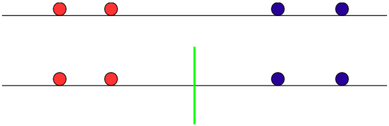
Why/How Might Unlabeled Data Help?

- Red: $+1$, Dark Blue: -1



Why/How Might Unlabeled Data Help?

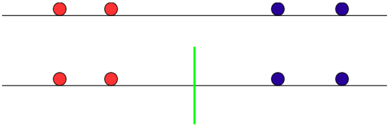
- Red: + 1, Dark Blue: -1



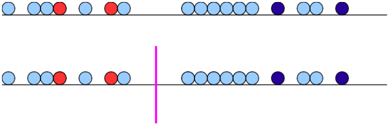
- Let's include some additional unlabeled data (Light Blue points)

Why/How Might Unlabeled Data Help?

- Red: $+$ 1, Dark Blue: -1

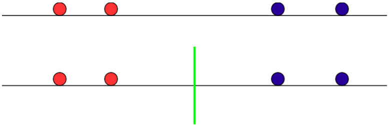


- Let's include some additional unlabeled data (Light Blue points)

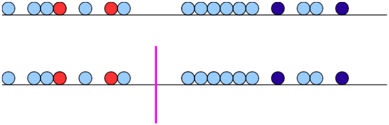


Why/How Might Unlabeled Data Help?

- Red: $+1$, Dark Blue: -1



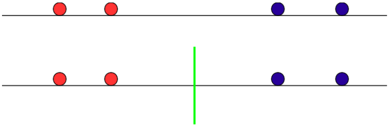
- Let's include some additional unlabeled data (Light Blue points)



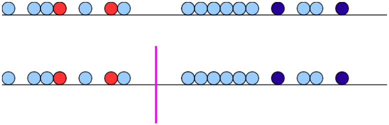
- Assumption: Examples from the same class follow a coherent distribution

Why/How Might Unlabeled Data Help?

- Red: + 1, Dark Blue: -1



- Let's include some additional unlabeled data (Light Blue points)



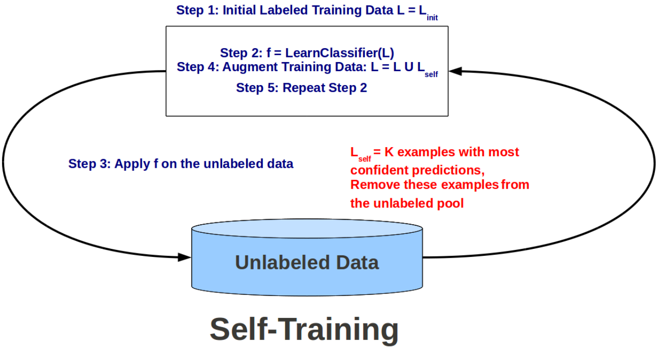
- Assumption: Examples from the same class follow a coherent distribution
- Unlabeled data can give a better sense of the class separation boundary

A Simple Algorithm: Self-Training

- **Given:** Small amount of initial labeled training data

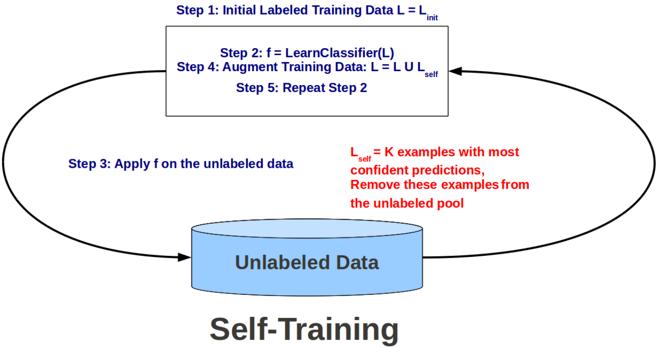
A Simple Algorithm: Self-Training

- **Given:** Small amount of initial labeled training data
- **Idea:** Train, predict, re-train using your own (best) predictions, repeat



A Simple Algorithm: Self-Training

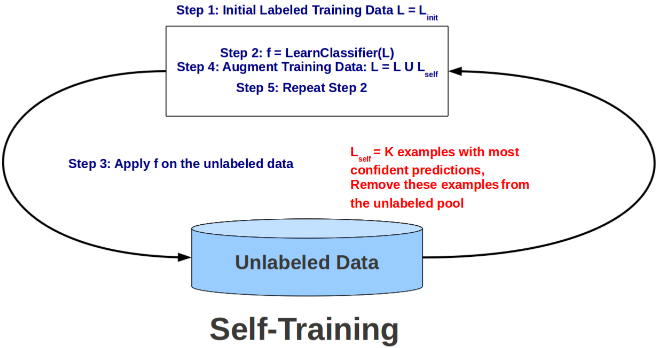
- **Given:** Small amount of initial labeled training data
- **Idea:** Train, predict, re-train using your own (best) predictions, repeat



- Can be used with any supervised learner. Often works well in practice

A Simple Algorithm: Self-Training

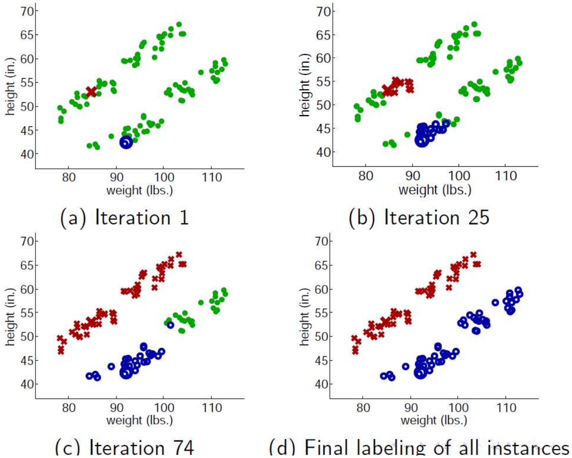
- **Given:** Small amount of initial labeled training data
- **Idea:** Train, predict, re-train using your own (best) predictions, repeat



- Can be used with any supervised learner. Often works well in practice
- **Caution:** Prediction mistake can reinforce itself

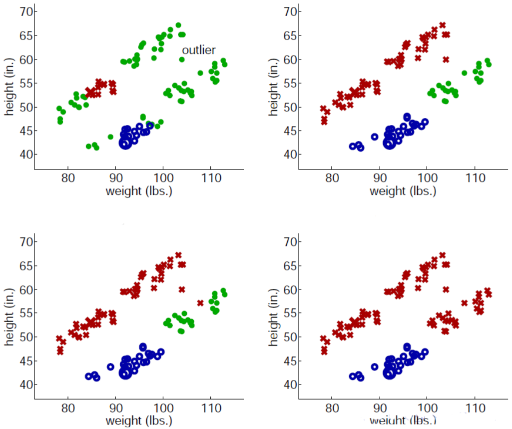
Self-Training: A Good Case

- Base learner: KNN classifier



Self-Training: A Bad Case

- Base learner: KNN classifier
- Things can go wrong if there are **outliers**. Mistakes get reinforced



Co-Training

- Given: Labeled data $\{\mathbf{x}_i, y_i\}_{i=1}^{L}$, unlabeled data $\{\mathbf{x}_j\}_{j=L+1}^{L+U}$

Co-Training

- Given: Labeled data $\{\mathbf{x}_i, y_i\}_{i=1}^{L}$, unlabeled data $\{\mathbf{x}_j\}_{j=L+1}^{L+U}$
- Each example has 2 views: $\mathbf{x} = [\mathbf{x}^{(1)} \quad \mathbf{x}^{(2)}]$

Co-Training

- Given: Labeled data $\{\mathbf{x}_i, y_i\}_{i=1}^{L}$, unlabeled data $\{\mathbf{x}_j\}_{j=L+1}^{L+U}$
- Each example has 2 views: $\mathbf{x} = [\mathbf{x}^{(1)} \quad \mathbf{x}^{(2)}]$
- How do we get different views?

Co-Training

- Given: Labeled data $\{\mathbf{x}_i, y_i\}_{i=1}^{L}$, unlabeled data $\{\mathbf{x}_j\}_{j=L+1}^{L+U}$
- Each example has 2 views: $\mathbf{x} = [\mathbf{x}^{(1)} \quad \mathbf{x}^{(2)}]$
- How do we get different views?
- Naturally available (different types of features for the same object)
  - Webpages: view 1 from page text; view 2 from social tags
  - Images: view 1 from pixel features; view 2 from fourier coefficients

Co-Training

- Given: Labeled data $\{\mathbf{x}_i, y_i\}_{i=1}^{L}$, unlabeled data $\{\mathbf{x}_j\}_{j=L+1}^{L+U}$
- Each example has 2 views: $\mathbf{x} = [\mathbf{x}^{(1)} \quad \mathbf{x}^{(2)}]$
- How do we get different views?
- Naturally available (different types of features for the same object)
  - Webpages: view 1 from page text; view 2 from social tags
  - Images: view 1 from pixel features; view 2 from fourier coefficients
- .. or by splitting the original features into two groups

Co-Training

- Given: Labeled data $\{\mathbf{x}_i, y_i\}_{i=1}^{L}$, unlabeled data $\{\mathbf{x}_j\}_{j=L+1}^{L+U}$
- Each example has 2 views: $\mathbf{x} = [\mathbf{x}^{(1)} \;\; \mathbf{x}^{(2)}]$
- How do we get different views?
- Naturally available (different types of features for the same object)
  - Webpages: view 1 from page text; view 2 from social tags
  - Images: view 1 from pixel features; view 2 from fourier coefficients
- .. or by splitting the original features into two groups
- Assumption: Given sufficient data, each view is good enough to learn from

Co-Training

- Given: Labeled data $\{\mathbf{x}_i, y_i\}_{i=1}^{L}$, unlabeled data $\{\mathbf{x}_j\}_{j=L+1}^{L+U}$
- Each example has 2 views: $\mathbf{x} = [\mathbf{x}^{(1)} \quad \mathbf{x}^{(2)}]$
- How do we get different views?
- Naturally available (different types of features for the same object)
  - Webpages: view 1 from page text; view 2 from social tags
  - Images: view 1 from pixel features; view 2 from fourier coefficients
- .. or by splitting the original features into two groups
- Assumption: Given sufficient data, each view is good enough to learn from
- Co-training: Utilize both views to learn better with fewer labeled examples

Co-Training

- Given: Labeled data $\{\mathbf{x}_i, y_i\}_{i=1}^{L}$, unlabeled data $\{\mathbf{x}_j\}_{j=L+1}^{L+U}$
- Each example has 2 views: $\mathbf{x} = [\mathbf{x}^{(1)} \quad \mathbf{x}^{(2)}]$
- How do we get different views?
- Naturally available (different types of features for the same object)
  - Webpages: view 1 from page text; view 2 from social tags
  - Images: view 1 from pixel features; view 2 from fourier coefficients
- .. or by splitting the original features into two groups
- Assumption: Given sufficient data, each view is good enough to learn from
- Co-training: Utilize both views to learn better with fewer labeled examples
- **Idea:** Each view teaching (training) the other view

Co-Training

- Given: Labeled data $\{\mathbf{x}_i, y_i\}_{i=1}^{L}$, unlabeled data $\{\mathbf{x}_j\}_{j=L+1}^{L+U}$
- Each example has 2 views: $\mathbf{x} = [\mathbf{x}^{(1)} \ \ \mathbf{x}^{(2)}]$
- How do we get different views?
- Naturally available (different types of features for the same object)
  - Webpages: view 1 from page text; view 2 from social tags
  - Images: view 1 from pixel features; view 2 from fourier coefficients
- .. or by splitting the original features into two groups
- Assumption: Given sufficient data, each view is good enough to learn from
- Co-training: Utilize both views to learn better with fewer labeled examples
- **Idea:** Each view teaching (training) the other view
- Technical Condition: Views should be conditionally independent
  - Intuitively, we don't want redundancy between the views

Co-Training

- Given labeled data L and unlabeled data U
- Create two labeled datasets L_1 and L_2 from L using views 1 and 2

Co-Training

- Given labeled data L and unlabeled data U
- Create two labeled datasets L_1 and L_2 from L using views 1 and 2
- Learn classifier $f^{(1)}$ using L_1 and classifier $f^{(2)}$ using L_2

Co-Training

- Given labeled data L and unlabeled data U
- Create two labeled datasets L_1 and L_2 from L using views 1 and 2
- Learn classifier $f^{(1)}$ using L_1 and classifier $f^{(2)}$ using L_2
- Apply $f^{(1)}$ and $f^{(2)}$ on unlabeled data pool U to predict labels
  - Predictions are made only using their own set (view) of features

Co-Training

- Given labeled data L and unlabeled data U
- Create two labeled datasets L_1 and L_2 from L using views 1 and 2
- Learn classifier $f^{(1)}$ using L_1 and classifier $f^{(2)}$ using L_2
- Apply $f^{(1)}$ and $f^{(2)}$ on unlabeled data pool U to predict labels
  - Predictions are made only using their own set (view) of features
- Add K most confident predictions $((\mathbf{x}, f^{(1)}(\mathbf{x}))$ of f_1 to L_2

Co-Training

- Given labeled data L and unlabeled data U
- Create two labeled datasets L_1 and L_2 from L using views 1 and 2
- Learn classifier $f^{(1)}$ using L_1 and classifier $f^{(2)}$ using L_2
- Apply $f^{(1)}$ and $f^{(2)}$ on unlabeled data pool U to predict labels
    - Predictions are made only using their own set (view) of features
- Add K most confident predictions $((\mathbf{x}, f^{(1)}(\mathbf{x}))$ of f_1 to L_2
- Add K most confident predictions $((\mathbf{x}, f^{(2)}(\mathbf{x}))$ of f_2 to L_1

Co-Training

- Given labeled data L and unlabeled data U
- Create two labeled datasets L_1 and L_2 from L using views 1 and 2
- Learn classifier $f^{(1)}$ using L_1 and classifier $f^{(2)}$ using L_2
- Apply $f^{(1)}$ and $f^{(2)}$ on unlabeled data pool U to predict labels
  - Predictions are made only using their own set (view) of features
- Add K most confident predictions $((\mathbf{x}, f^{(1)}(\mathbf{x}))$ of f_1 to L_2
- Add K most confident predictions $((\mathbf{x}, f^{(2)}(\mathbf{x}))$ of f_2 to L_1
- Note: Absolute margin could be used to measure confidence

Co-Training

- Given labeled data L and unlabeled data U
- Create two labeled datasets L_1 and L_2 from L using views 1 and 2
- Learn classifier $f^{(1)}$ using L_1 and classifier $f^{(2)}$ using L_2
- Apply $f^{(1)}$ and $f^{(2)}$ on unlabeled data pool U to predict labels
  - Predictions are made only using their own set (view) of features
- Add K most confident predictions $((\mathbf{x}, f^{(1)}(\mathbf{x}))$ of f_1 to L_2
- Add K most confident predictions $((\mathbf{x}, f^{(2)}(\mathbf{x}))$ of f_2 to L_1
- Note: Absolute margin could be used to measure confidence
- Remove these examples from the unlabeled pool

Co-Training

- Given labeled data L and unlabeled data U
- Create two labeled datasets L_1 and L_2 from L using views 1 and 2
- Learn classifier $f^{(1)}$ using L_1 and classifier $f^{(2)}$ using L_2
- Apply $f^{(1)}$ and $f^{(2)}$ on unlabeled data pool U to predict labels
  - Predictions are made only using their own set (view) of features
- Add K most confident predictions $((\mathbf{x}, f^{(1)}(\mathbf{x}))$ of f_1 to L_2
- Add K most confident predictions $((\mathbf{x}, f^{(2)}(\mathbf{x}))$ of f_2 to L_1
- Note: Absolute margin could be used to measure confidence
- Remove these examples from the unlabeled pool
- Re-train $f^{(1)}$ using L_1, $f^{(1)}$ using L_2

Co-Training

- Given labeled data L and unlabeled data U
- Create two labeled datasets L_1 and L_2 from L using views 1 and 2
- Learn classifier $f^{(1)}$ using L_1 and classifier $f^{(2)}$ using L_2
- Apply $f^{(1)}$ and $f^{(2)}$ on unlabeled data pool U to predict labels
  - Predictions are made only using their own set (view) of features
- Add K most confident predictions $((\mathbf{x}, f^{(1)}(\mathbf{x}))$ of f_1 to L_2
- Add K most confident predictions $((\mathbf{x}, f^{(2)}(\mathbf{x}))$ of f_2 to L_1
- Note: Absolute margin could be used to measure confidence
- Remove these examples from the unlabeled pool
- Re-train $f^{(1)}$ using L_1, $f^{(1)}$ using L_2
- Like self-training but two classifiers teaching each other

Co-Training

- Given labeled data L and unlabeled data U
- Create two labeled datasets L_1 and L_2 from L using views 1 and 2
- Learn classifier $f^{(1)}$ using L_1 and classifier $f^{(2)}$ using L_2
- Apply $f^{(1)}$ and $f^{(2)}$ on unlabeled data pool U to predict labels
  - Predictions are made only using their own set (view) of features
- Add K most confident predictions $((\mathbf{x}, f^{(1)}(\mathbf{x}))$ of f_1 to L_2
- Add K most confident predictions $((\mathbf{x}, f^{(2)}(\mathbf{x}))$ of f_2 to L_1
- Note: Absolute margin could be used to measure confidence
- Remove these examples from the unlabeled pool
- Re-train $f^{(1)}$ using L_1, $f^{(1)}$ using L_2
- Like self-training but two classifiers teaching each other
- Finally, use a voting or averaging to make predictions on the test data

Multi-view Learning

- A general class of algorithms for semi-supervised learning
- Based on using multiple views (feature representations) of the data
- Co-training is a special type of multi-view learning algorithm

Multi-view Learning

- A general class of algorithms for semi-supervised learning
- Based on using multiple views (feature representations) of the data
- Co-training is a special type of multi-view learning algorithm
- General Idea: Train multiple classifiers, each using a different view

Multi-view Learning

- A general class of algorithms for semi-supervised learning
- Based on using multiple views (feature representations) of the data
- Co-training is a special type of multi-view learning algorithm
- General Idea: Train multiple classifiers, each using a different view
- Modus Operandi: Multiple classifiers must agree on the unlabeled data

Multi-view Learning

- A general class of algorithms for semi-supervised learning
- Based on using multiple views (feature representations) of the data
- Co-training is a special type of multi-view learning algorithm
- General Idea: Train multiple classifiers, each using a different view
- Modus Operandi: Multiple classifiers must agree on the unlabeled data
- How might it help learn better?

Multi-view Learning

- A general class of algorithms for semi-supervised learning
- Based on using multiple views (feature representations) of the data
- Co-training is a special type of multi-view learning algorithm
- General Idea: Train multiple classifiers, each using a different view
- Modus Operandi: Multiple classifiers must agree on the unlabeled data
- How might it help learn better?
  - Learning is essentially searching for the best classifier

Multi-view Learning

- A general class of algorithms for semi-supervised learning
- Based on using multiple views (feature representations) of the data
- Co-training is a special type of multi-view learning algorithm
- General Idea: Train multiple classifiers, each using a different view
- Modus Operandi: Multiple classifiers must agree on the unlabeled data
- How might it help learn better?
  - Learning is essentially searching for the best classifier
  - By enforcing agreement among classifiers, we are reducing the search space
    $\Rightarrow$ hope is that the best classifier can be found easily with little labeled data

Multi-view Learning

- A general class of algorithms for semi-supervised learning
- Based on using multiple views (feature representations) of the data
- Co-training is a special type of multi-view learning algorithm
- General Idea: Train multiple classifiers, each using a different view
- Modus Operandi: Multiple classifiers must agree on the unlabeled data
- How might it help learn better?
  - Learning is essentially searching for the best classifier
  - By enforcing agreement among classifiers, we are reducing the search space
    $\Rightarrow$ hope is that the best classifier can be found easily with little labeled data
- For test data, these multiple classifiers can be combined
  - E.g., voting, consensus, etc.

Multi-view Learning

- A general class of algorithms for semi-supervised learning
- Based on using multiple views (feature representations) of the data
- Co-training is a special type of multi-view learning algorithm
- General Idea: Train multiple classifiers, each using a different view
- Modus Operandi: Multiple classifiers must agree on the unlabeled data
- How might it help learn better?
  - Learning is essentially searching for the best classifier
  - By enforcing agreement among classifiers, we are reducing the search space
    $\Rightarrow$ hope is that the best classifier can be found easily with little labeled data
- For test data, these multiple classifiers can be combined
  - E.g., voting, consensus, etc.
- Backed by a number of theoretical results

Cluster-and-Label Approach

Input: $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l)$, $\mathbf{x}_{l+1}, \ldots, \mathbf{x}_{l+u}$,
a clustering algorithm $\mathcal{A}$, a supervised learning algorithm $\mathcal{L}$

1. Cluster $\mathbf{x}_1, \ldots, \mathbf{x}_{l+u}$ using $\mathcal{A}$.
2. For each cluster, let S be the labeled instances in it:
3. Learn a supervised predictor from S: $f_S = \mathcal{L}(S)$.
4. Apply f_S to all unlabeled instances in this cluster.

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

- Finally train a supervised learner on the entire labeled data

Cluster-and-Label Approach

Input: $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l), \mathbf{x}_{l+1}, \ldots, \mathbf{x}_{l+u}$,
a clustering algorithm $\mathcal{A}$, a supervised learning algorithm $\mathcal{L}$

1. Cluster $\mathbf{x}_1, \ldots, \mathbf{x}_{l+u}$ using $\mathcal{A}$.
2. For each cluster, let S be the labeled instances in it:
3. Learn a supervised predictor from S: $f_S = \mathcal{L}(S)$.
4. Apply f_S to all unlabeled instances in this cluster.

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

- Finally train a supervised learner on the entire labeled data
- Assumption: Clusters coincide with decision boundaries
  - Poor results if this assumption is wrong

Expectation Maximization Approach

- **Given:** Labeled data $\mathcal{L} = \{\mathbf{x}_i, y_i\}_{i=1}^{L}$, unlabeled data $\mathcal{U} = \{\mathbf{x}_j\}_{j=L+1}^{L+U}$

Expectation Maximization Approach

- **Given:** Labeled data $\mathcal{L} = \{\mathbf{x}_i, y_i\}_{i=1}^{L}$, unlabeled data $\mathcal{U} = \{\mathbf{x}_j\}_{j=L+1}^{L+U}$
- Expectation-Maximization based Semi-supervised Learning:
  - Train an initial model using just $\mathcal{L}$ (e.g., using MLE or MAP)

$$\hat{\theta} = \text{TrainModel}(mathcalL)$$

Expectation Maximization Approach

- **Given:** Labeled data $\mathcal{L} = \{\mathbf{x}_i, y_i\}_{i=1}^{L}$, unlabeled data $\mathcal{U} = \{\mathbf{x}_j\}_{j=L+1}^{L+U}$
- Expectation-Maximization based Semi-supervised Learning:
  - Train an initial model using just $\mathcal{L}$ (e.g., using MLE or MAP)

$$\hat{\theta} = \text{TrainModel}(mathcalL)$$

  - Use this model to "guess" the label of each $\mathbf{x}_j \in \mathcal{U}$ (compute expected label). Assuming binary labels (+1/-1), we can compute:

$$\mathbb{E}[y_j] = +1 \times P(y_j = +1|\hat{\theta}, \mathbf{x}_j) + (-1) \times P(y_j = -1|\hat{\theta}, \mathbf{x}_j)$$

Expectation Maximization Approach

- **Given:** Labeled data $\mathcal{L} = \{\mathbf{x}_i, y_i\}_{i=1}^{L}$, unlabeled data $\mathcal{U} = \{\mathbf{x}_j\}_{j=L+1}^{L+U}$
- Expectation-Maximization based Semi-supervised Learning:
  - Train an initial model using just $\mathcal{L}$ (e.g., using MLE or MAP)

$$\hat{\theta} = \text{TrainModel}(mathcalL)$$

  - Use this model to "guess" the label of each $\mathbf{x}_j \in \mathcal{U}$ (compute expected label). Assuming binary labels (+1/-1), we can compute:

$$\mathbb{E}[y_j] = +1 \times P(y_j = +1|\hat{\theta}, \mathbf{x}_j) + (-1) \times P(y_j = -1|\hat{\theta}, \mathbf{x}_j)$$

  - Re-train the model using $\mathcal{L} + \mathcal{U}$ with its guessed labels

Expectation Maximization Approach

- **Given:** Labeled data $\mathcal{L} = \{\mathbf{x}_i, y_i\}_{i=1}^{L}$, unlabeled data $\mathcal{U} = \{\mathbf{x}_j\}_{j=L+1}^{L+U}$
- Expectation-Maximization based Semi-supervised Learning:
  - Train an initial model using just $\mathcal{L}$ (e.g., using MLE or MAP)

$$\hat{\theta} = \text{TrainModel}(mathcalL)$$

  - Use this model to "guess" the label of each $\mathbf{x}_j \in \mathcal{U}$ (compute expected label). Assuming binary labels (+1/-1), we can compute:

$$\mathbb{E}[y_j] = +1 \times P(y_j = +1|\hat{\theta}, \mathbf{x}_j) + (-1) \times P(y_j = -1|\hat{\theta}, \mathbf{x}_j)$$

  - Re-train the model using $\mathcal{L} + \mathcal{U}$ with its guessed labels
  - Use the new model $\hat{\theta}$ to refine the guesses of the labels $\mathbb{E}[y_j]$ of $\mathcal{U}$

Expectation Maximization Approach

- **Given:** Labeled data $\mathcal{L} = \{\mathbf{x}_i, y_i\}_{i=1}^{L}$, unlabeled data $\mathcal{U} = \{\mathbf{x}_j\}_{j=L+1}^{L+U}$
- Expectation-Maximization based Semi-supervised Learning:
  - Train an initial model using just $\mathcal{L}$ (e.g., using MLE or MAP)

$$\hat{\theta} = \text{TrainModel}(mathcalL)$$

  - Use this model to "guess" the label of each $\mathbf{x}_j \in \mathcal{U}$ (compute expected label). Assuming binary labels (+1/-1), we can compute:

$$\mathbb{E}[y_j] = +1 \times P(y_j = +1|\hat{\theta}, \mathbf{x}_j) + (-1) \times P(y_j = -1|\hat{\theta}, \mathbf{x}_j)$$

  - Re-train the model using $\mathcal{L} + \mathcal{U}$ with its guessed labels
  - Use the new model $\hat{\theta}$ to refine the guesses of the labels $\mathbb{E}[y_j]$ of $\mathcal{U}$
  - Repeat until converged

Expectation Maximization Approach

- **Given:** Labeled data $\mathcal{L} = \{\mathbf{x}_i, y_i\}_{i=1}^{L}$, unlabeled data $\mathcal{U} = \{\mathbf{x}_j\}_{j=L+1}^{L+U}$
- Expectation-Maximization based Semi-supervised Learning:
  - Train an initial model using just $\mathcal{L}$ (e.g., using MLE or MAP)

$$\hat{\theta} = \text{TrainModel}(mathcalL)$$

  - Use this model to "guess" the label of each $\mathbf{x}_j \in \mathcal{U}$ (compute expected label). Assuming binary labels (+1/-1), we can compute:

$$\mathbb{E}[y_j] = +1 \times P(y_j = +1|\hat{\theta}, \mathbf{x}_j) + (-1) \times P(y_j = -1|\hat{\theta}, \mathbf{x}_j)$$

  - Re-train the model using $\mathcal{L} + \mathcal{U}$ with its guessed labels
  - Use the new model $\hat{\theta}$ to refine the guesses of the labels $\mathbb{E}[y_j]$ of $\mathcal{U}$
  - Repeat until converged
- A general scheme; can be used with any probabilistic learning model
  - E.g., naïve Bayes, logistic regression, linear regression etc.
  - $P(y_j|\hat{\theta}, \mathbf{x}_j)$ would have to be defined accordingly

Graph Based Semi-supervised Learning

- Graph based approaches exploit the property of label smoothness

Graph Based Semi-supervised Learning

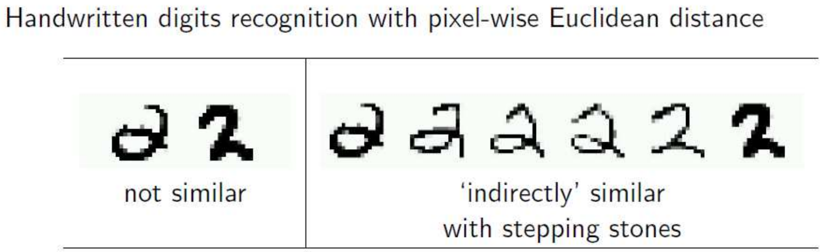
- Graph based approaches exploit the property of label smoothness
- Idea: Represent each example (labeled/unlabeled) as vertices of some graph
- Idea: The labels should vary smoothly along the graph
  $\Rightarrow$ Nearby vertices should have similar labels

Graph Based Semi-supervised Learning

- Graph based approaches exploit the property of label smoothness
- Idea: Represent each example (labeled/unlabeled) as vertices of some graph
- Idea: The labels should vary smoothly along the graph
  $\Rightarrow$ Nearby vertices should have similar labels
- This idea is called Graph-based Regularization

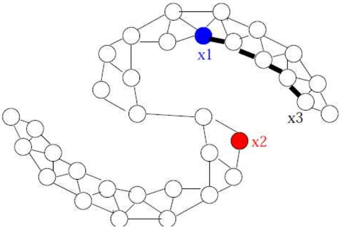
Graph Based Semi-supervised Learning

- Graph based approaches exploit the property of label smoothness
- Idea: Represent each example (labeled/unlabeled) as vertices of some graph
- Idea: The labels should vary smoothly along the graph
  $\Rightarrow$ Nearby vertices should have similar labels
- This idea is called Graph-based Regularization



Graph Regularization

- Nodes: $X_l \cup X_u$
- Edges: similarity weights computed from features, e.g.,
  - k-nearest-neighbor graph, unweighted (0, 1 weights)
  - fully connected graph, weight decays with distance $w = \exp\left(-\|x_i - x_j\|^2/\sigma^2\right)$
  - ϵ-radius graph
- Assumption Instances connected by heavy edge tend to have the same label.



Graph Regularization

- Assume the predictions on the entire data $\mathcal{L} \cup \mathcal{U}$ to be defined by function f

Graph Regularization

- Assume the predictions on the entire data $\mathcal{L} \cup \mathcal{U}$ to be defined by function f
- Graph regularization assumes that the function f is smooth
  $\Rightarrow$ Similar examples i and j should have similar predictions f_i and f_j

Graph Regularization

- Assume the predictions on the entire data $\mathcal{L} \cup \mathcal{U}$ to be defined by function f
- Graph regularization assumes that the function f is smooth
  $\Rightarrow$ Similar examples i and j should have similar predictions f_i and f_j
- Graph regularization optimizes the following objective:

$$\min_f \sum_{i \in \mathcal{L}} (y_i - f_i)^2 + \lambda \sum_{i,j \in \mathcal{L},\mathcal{U}} w_{ij}(f_i - f_j)^2$$

Graph Regularization

- Assume the predictions on the entire data $\mathcal{L} \cup \mathcal{U}$ to be defined by function f
- Graph regularization assumes that the function f is smooth
  $\Rightarrow$ Similar examples i and j should have similar predictions f_i and f_j
- Graph regularization optimizes the following objective:

$$\min_f \sum_{i \in \mathcal{L}} (y_i - f_i)^2 + \lambda \sum_{i,j \in \mathcal{L},\mathcal{U}} w_{ij}(f_i - f_j)^2$$

- First part is minimizing the loss on labeled data, second part ensures smoothness of labels of labeled and unlabeled data

Graph Regularization

- Assume the predictions on the entire data $\mathcal{L} \cup \mathcal{U}$ to be defined by function f
- Graph regularization assumes that the function f is smooth
  $\Rightarrow$ Similar examples i and j should have similar predictions f_i and f_j
- Graph regularization optimizes the following objective:

$$\min_f \sum_{i \in \mathcal{L}} (y_i - f_i)^2 + \lambda \sum_{i,j \in \mathcal{L},\mathcal{U}} w_{ij}(f_i - f_j)^2$$

- First part is minimizing the loss on labeled data, second part ensures smoothness of labels of labeled and unlabeled data
  $\Rightarrow$ Minimization makes f_i and f_j to be very similar if w_{ij} is large

Graph Regularization

- Assume the predictions on the entire data $\mathcal{L} \cup \mathcal{U}$ to be defined by function f
- Graph regularization assumes that the function f is smooth
  $\Rightarrow$ Similar examples i and j should have similar predictions f_i and f_j
- Graph regularization optimizes the following objective:

$$\min_f \sum_{i \in \mathcal{L}} (y_i - f_i)^2 + \lambda \sum_{i,j \in \mathcal{L},\mathcal{U}} w_{ij}(f_i - f_j)^2$$

- First part is minimizing the loss on labeled data, second part ensures smoothness of labels of labeled and unlabeled data
  $\Rightarrow$ Minimization makes f_i and f_j to be very similar if w_{ij} is large
- λ is a trade-off parameter

Graph Regularization

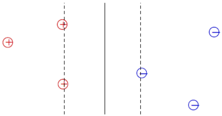
- Assume the predictions on the entire data $\mathcal{L} \cup \mathcal{U}$ to be defined by function f
- Graph regularization assumes that the function f is smooth
  $\Rightarrow$ Similar examples i and j should have similar predictions f_i and f_j
- Graph regularization optimizes the following objective:

$$\min_f \sum_{i \in \mathcal{L}} (y_i - f_i)^2 + \lambda \sum_{i,j \in \mathcal{L},\mathcal{U}} w_{ij}(f_i - f_j)^2$$

- First part is minimizing the loss on labeled data, second part ensures smoothness of labels of labeled and unlabeled data
  $\Rightarrow$ Minimization makes f_i and f_j to be very similar if w_{ij} is large
- λ is a trade-off parameter
- Several variants and ways to solve the above problem (refer to the SSL survey paper's section on graph based methods)

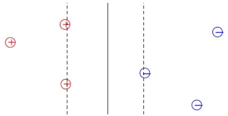
Transductive SVM: Avoiding Dense Regions

SVMs

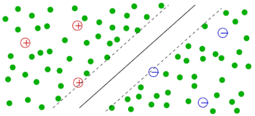


Transductive SVM: Avoiding Dense Regions

SVMs



Semi-supervised SVMs (S3VMs) = Transductive SVMs (TSVMs)



- Unlabeled data from different classes are separated by large margin
- Idea: The decision boundary shouldn't lie in the regions of high density
- For details, refer to the SSL survey paper's section on transductive SVMs

Concluding Thoughts

- Unlabeled data can help if the model assumptions are appropriate
- Incorrect assumptions can hurt (so be careful)
- SSL is also motivated by human learning
- There exists several other ways of learning with labeled data scarcity. E.g.,
  - Active Learning (next class)
  - Crowd-sourcing (free-of-cost labels)

Luis von Ahn: Games with a purpose (ReCaptcha)

Word rejected by OCR
(Optical Character Recogintion)

You provide a free label!