Basic idea

- Traditional classifiers learn only from labeled data.
- Label data can be expensive / difficult to collect.
  - Human annotation is slow, boring!
  - Labels can require experts, or special devices to acquire.
- We prefer to get better performance for free: Unlabeled data!
- Goal of semi-supervised learning is to exploit both labeled and unlabeled examples.
- Most of today will be on semi-supervised classification; brief discussion of semi-supervised regression and semi-supervised clustering.
Example of hard-to-get labels

Task: speech analysis
- Switchboard dataset
- Telephone conversation transcription
- 400 hours annotation time for each hour of speech

film $\Rightarrow$ f ih n uh gl n m
be all $\Rightarrow$ bcl b iy iy_tr ao_tr ao l dl

Example of hard-to-get labels

Task: natural language parsing
- Penn Chinese Treebank
- 2 years for 4000 sentences

```
IP-HLN
  |__________NN__________
    |                  
    NP-SBJ            VY
    |                  
    NP                
    |                
    DT NN NP NN NN
```

全国 田径 冠军赛

“The National Track and Field Championship has finished.”
Example of not-so-hard-to-get labels

For some tasks, it may not be too difficult to label 1000+ instances.

Task: image categorization of “eclipse”

There are ways like the EPS game (www.epsgame.org) to encourage “human computation” for more labels.
Example of not-so-hard-to-get labels

For some tasks, it may not be too difficult to label 1000+ instances. Nonetheless...

Goal: Use both labeled and unlabeled data to build better learners, than using each one alone.

Notation

• Given:
  - Labeled data: \( (X_l, Y_l) = \{x_{1:l}, y_{1:l}\} \) available during training
  - Unlabeled data: \( X_u = \{x_{l+1:n}\} \) available during training
  - Test data: \( X_{test} = \{x_{n+1:N}\} \) NOT available during training

• Usually \( l \ll n \), so much more unlabeled data than labeled data.
Notation

supervised learning (classification, regression) \( \{(x_{1:n}, y_{1:n})\} \)

\[ \downarrow \]

semi-supervised classification/regression \( \{(x_{1:l}, y_{1:l}, x_{l+1:n}, x_{test})\} \)

transductive classification/regression \( \{(x_{1:l}, y_{1:l}, x_{l+1:n})\} \)

\[ \downarrow \]

semi-supervised clustering \( \{x_{1:n}, \text{must-}, \text{cannot-links}\} \)

\[ \downarrow \]

unsupervised learning (clustering) \( \{x_{1:n}\} \)

How can unlabeled data help?

- Assuming each class is a coherent group (e.g. Gaussian)
- With vs without unlabeled data: Decision boundary shifts.
Self-training algorithm

- Assume: One’s own high confidence predictions are correct.
- Basic algorithm:
  - Train \( f \) from \((X_l, Y_l)\).
  - Predict for \( x \in X_u \).
  - Add \((x, f(x))\) to labeled data.
  - Repeat.
- Variations:
  - Add a few most confident \((x, f(x))\) to labeled data.
  - Add all \((x, f(x))\) to labeled data.
  - Add all \((x, f(x))\) to labeled data, weigh each by confidence.

Self-training example: Image categorization

- Train a Naïve Bayes classifier on two initial labeled images:
Self-training example: image categorization

- Each image is divided into small patches.
- 10x10 grid, random size of 10 ~ 20

All patches are normalized.
- Define a dictionary of 200 “visual words” (cluster centroids) with 200-means clustering on all patches.
- Represent a patch by the index of its closest visual word.
Self-training example: image categorization

- Train a Naïve Bayes classifier on two initial labeled images:

- Classify unlabeled data, sort by confidence $\log P(y=\text{astronomy} | x)$.

Advantages of self-training

- The simplest semi-supervised learning method.
- A wrapper method, applies to existing (complex) classifiers.
- Often used in real tasks like natural language processing.
Disadvantages of self-training

• Early mistakes could reinforce themselves.
  – Heuristic solutions, e.g. “un-label” an instance if its confidence falls below a threshold.

• Cannot say too much in terms of convergence.
  – But there are special cases when self-training is equivalent to the Expectation-Maximization (EM) algorithm.
  – There are also special cases (e.g. linear functions) when the closed-form solution is known.

The generative approach

• Given labeled data, assume each class has a Gaussian distribution.
The generative approach

- Given labeled data, assume each class has a Gaussian distribution.
- The most likely model and its decision boundary.

- Add unlabeled data:
The generative approach

- Given labeled data, assume each class has a Gaussian distribution.
- The most likely model and its decision boundary:
- Add unlabeled data.
- The most likely model and decision boundary change.

The generative approach

- Decision boundaries are different because they maximize different quantities.

\[ p(X_1, Y_1 | \theta) \quad \text{and} \quad p(X_1, Y_1, X_u | \theta) \]
Revisiting the EM algorithm

- **Setup**: 
  - Observed data: \( D = (X_p, Y_p, X_u) \)
  - Hidden data: \( H = Y_u \)
  - \( P(D|\theta) = \sum_H p(D, H | \theta) \)

- **Goal**: Find \( \theta \) to maximize \( p(D|\theta) \)

- **Algorithm**: 
  - Start with some arbitrary \( \theta_0 \).
  - **E-step**: Estimate \( p(H|D, \theta) \)
  - **M-step**: Find \( \arg\max_{\theta} \sum_H p(D, H | \theta) \)

- **Comments**: EM iteratively improves \( p(D|\theta) \). Converges to a local minima of \( \theta \). K-means is a special case of this.

Comments on the generative approach

- This offers a clear, well-studied, probabilistic framework.
- Can be very effective if the model is close to correct.
- Often difficult to verify the correctness of the model. Unlabeled data can hurt the solution if the generative model is wrong.
- EM converges to a local optima.
- There are other ways than EM to find parameters, e.g. variational approximation.
Alternate method: Cluster-and-label

Instead of running EM with the probabilistic generative model using the labeled data:

- Run the clustering algorithm assuming all data is unlabeled.
- Label all points within a cluster by the majority of labeled points in that cluster.

- **Pro**: Another simple wrapper method.
- **Con**: Can be difficult to analyze; labels within a cluster may disagree.

Alternate method: Autoencoder + supervised layer

**From Lecture 17**: Train an autoencoder, then add a supervised layer and train the full network with backpropagation using error on the predicted output, 

\[ Err(W) = \sum_{i=1}^{n} L \left[ y_i, o(x) \right] \]

http://www.dmi.usherbrooke.ca/~larocheh/projects_deep_learning.html
Many more methods!

- **Co-training.**
- Semi-supervised SVMs.
- Graph-based algorithms.
- Etc.

---

Co-training: Feature split

- Key idea: Use dual views of a datapoint to classify.
- Each instance represented by two sets of features: \( x = [x^{(1)}, x^{(2)}] \).
  - \( x^{(1)} \) = image features
  - \( x^{(2)} \) = web page text
  - This is a natural feature split (or multiple views).
Co-training

• E.g. Two views of an item: image and HTML text

Co-training: Feature split

• Key idea: Use dual views of a datapoint to classify.

• Each instance represented by two sets of features: $x = [x^{(1)}, x^{(2)}]$.
  – $x^{(1)}$ = image features
  – $x^{(2)}$ = web page text

• Co-training idea:
  – Train an image classifier and a text classifier.
  – The two classifiers teach each other.

• Assumptions:
  – $x^{(1)}$ or $x^{(2)}$ is sufficient to train a good classifier.
  – $x^{(1)}$ and $x^{(2)}$ are conditionally independent given the class.
Co-training algorithm

- Basic algorithm:
  - Train two classifiers: \( f(1) \) from \( (X_l^{(1)}, Y_l) \), \( f(2) \) from \( (X_l^{(2)}, Y_l) \).
  - Classify \( X_u \) with \( f(1) \) and \( f(2) \) separately.
  - Add \( f(1) \)'s \( k \)-most confident examples to \( f(1) \)'s labeled data.
  - Add \( f(2) \)'s \( k \)-most confident examples to \( f(2) \)'s labeled data.
  - Repeat.

Pros and cons of co-training

- Pros:
  - Simple wrapper method. Applies to almost all existing classifiers.
  - Less sensitive to mistakes than self-training.

- Cons:
  - Natural feature splits may not exist.
  - Models using BOTH features should do better.

- Many variants on co-training.
Many more methods!

- Co-training.
- **Semi-supervised SVMs.**
- Graph-based algorithms.
- Etc.

Semi-supervised SVMs

- Semi-supervised SVMs = Transductive SVMs.
- Idea: Maximize "unlabeled data margin". (More in literature.)
What method should you use?

- Do the classes produce well clustered data?
  - If yes, try EM with generative mixture model!
- Do the features naturally split into two sets?
  - If yes, try cotraining.
- Already using SVMs?
  - Try transductive SVMs.
- Is the existing supervised classifier complicated & hard to modify?
  - Try self-training, as a wrapper method.
- Do two points with similar features tend to be in the same class?
  - If yes, try graph-based methods.

Does unlabeled data always help?

- There’s no free lunch! Semi-supervised learning typically makes strong model assumptions (to compensate for lack of labels).
- Performance can degrade by addition of unlabeled data when the modeling assumptions are not appropriate. This has been empirically observed by many researchers.
- So far, we have discussed missing labels.
- In many problems, we are missing some of the features.
- More on this on Friday!
Final note

• Significant material (pictures, text, equations) for these slides was taken from: