CS 4650/7650, Lecture 16: Semi-supervised learning and domain adaptation

Jacob Eisenstein

October 17, 2013

You can't always get what you want. -Mick Jagger

So far we have focused on learning a classifier — typically represented by a set of weights $\boldsymbol{w}$ — from a set of labeled examples $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{\ell}$. But what if you don't have those labeled examples for the domain or task that you want to solve?

- You can use some other labeled data and hope it works.
  This rarely works well.
- You can label data yourself.
  This is a lot of work.

This kind of thing happens all the time, especially in class projects. And labeled data is really expensive:

- **The Switchboard corpus** contains phoneme annotations of telephone conversations, e.g.

  $$\textit{film} \rightarrow \text{F IH\_N UH\_GL\_N M}$$
  $$\textit{be all} \rightarrow \text{BCL B IY IY\_TR AO\_TR AO L\_DL}$$

  How long do you think this took? 400 hours of annotation time per hour of speech.

- **The Penn Chinese Treebank** is a set of CFG annotations for Chinese. It took 2 years to get 4000 sentences annotated.

1 Learning with less annotation effort

We can think of our annotated data as a *sample* from some underlying distribution.

$$\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{\ell} \sim \mathcal{D} \tag{1}$$

This allows us to formulate various learning scenarios:

1.1 Semisupervised learning

- $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{\ell} \sim \mathcal{D}$: labeled examples
- $\{(\boldsymbol{x}_i)\}_{i=\ell+1}^{\ell+u}$: unlabeled examples
- often $u \gg \ell$

We've already seen an example of semi-supervised learning in document classification, when we applied expectation maximization to impute labels of unlabeled documents. Today we will see some approaches that tend to work better than EM.

Active learning is closely related to semi-supervised learning, but you can now query the labels for a few examples while learning. Your goal is to select these examples carefully, so that you get the best accuracy from a fixed number of examples, or so that you get to a certain level of accuracy with as few examples as possible.

1.2 Domain adaptation

- $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{\ell_S} \sim \mathcal{D}_S$: labeled examples in *source* domain
- Some *target domain* $\mathcal{D}_T$. You may have a small amount of labeled data in the target domain ("supervised domain adaptation") or not ("unsupervised domain adaptation").

  The prototypical example for domain adaptation is product reviews: you have annotated reviews of coffee machines, but you want to train a classifier for reviews of bicycles. Another example is that you have a POS tagger for news genre text, but you want one for social media.
- Suppose that $P_T(X) = P_S(X)$, i.e. the distribution over observed data is the same, but $P_T(Y|X) \neq P_S(Y|X)$. This setting is sometimes called **transfer learning** or **multitask learning**.

  For example, you have labeled data for part-of-speech tagging, but you want to train a system for named entity recognition. Or, you have a classifier for detecting mentions of people and you want one for detecting mentions of places.

2 Why would unlabeled data help?

Suppose you want to do sentiment analysis in French. I give you two labeled examples:

- ☺ émouvant avec grâce et **style**
- ☹ fastidieusement inauthentique et **banale**

You have a bunch of unlabeled examples too:

1. pleine de **style** et d'**intrigue**
2. la **banalité** n'est dépassée que par sa **prétention**
3. **prétentieux**, de la première minute au rideau final
4. imprégné d'un air d'**intrigue**

What can we do? If we just learn from the labeled data, we might decide that *style* is positive and that **banale** is negative. This isn't much. However, we can propagate this information to the unlabeled data, and potentially learn more.

- If we are confident about *style* being positive, then we can guess that example 1 is also positive.
- That suggests that *intrigue* is also positive.
- We can then propagate this information to example 4, and learn more.
- Similarly, we can propagate from the labeled data to example 2, which we guess to be negative. This suggests that *pretention* is also negative, which we propagate to example 3.

What happened here?

- Instances 1 and 2 were "similar" to our labeled examples for positivity and negativity, respectively. We used them to expand those concepts, which allowed us to correctly label instances 3 and 4, which didn't share any important features with our original labeled data.
- We made a key assumption: that similar instances will have similar labels. Is this a strong assumption? Keep this question in mind.
- In this case, we defined similarity in terms of sharing some key words (non-stopwords).
- To see how this can help conceptually, think about similarity just in terms of 1D space. If you have only the two labeled instances, your decision boundary should be right in between. Do you remember what criterion justifies this choice? But if you have a bunch of unlabeled instances, you might want to draw this boundary in a different place.
- Let's see how we can operationalize this idea in an algorithm.

2.1 Semi-supervised learning with EM

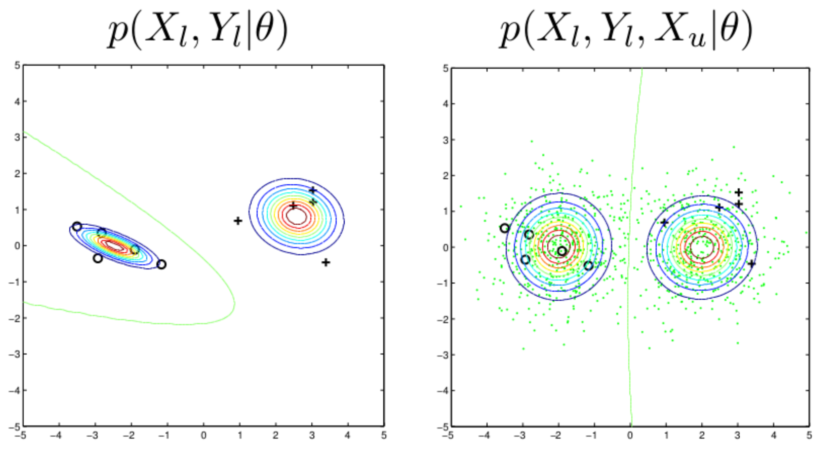
We've already seen one way to do this: use expectation-maximization to marginalize over the labels of the unseen data. So we are maximizing

$$P(X^\ell, Y^\ell, X^U) = P(X^\ell, Y^\ell) \sum_{Y^U} P(X^U, Y^U) \tag{2}$$

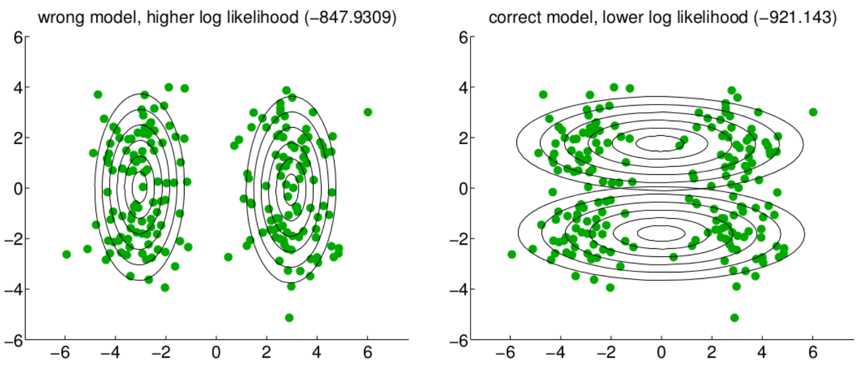
We did this by

- **E**. fitting a distribution $Q(y_i)$ for all unlabeled i,
- **M**. maximizing the expected likelihood under this distribution.

You can see why this can work in an example:



We get a much more reasonable decision boundary. However, things can also go wrong:



The correct model has a lower log-likelihood than the incorrect model. The basic problem is that our model is wrong. The label is related to the observations, but not in the simplistic, Gaussian way that we had assumed. We discussed a heuristic to try to deal with this problem: downweighting the contribution of the unseen data to the likelihood function.

2.2 Bootstrapping

EM is sort of like self-training or **bootstrapping**: we use our current model to estimate $Q(y_i)$, and then update the model as if $Q(y_i)$ is correct.

- The probabilistic nature of this is nice, but it limits us to relatively weak classifiers.
- If we are willing to give up on probability, we can use **any** classifier.

We can try this first using one nearest-neighbor (see slides). Like EM, it can work, but it can also fail. The failure modes are different though; can you characterize the difference?

2.3 Co-training

"Folk wisdom" about when bootstrapping works:

- Better for generative models (e.g., Naive Bayes) than for discriminative models (e.g., perceptron)
- Better when the Naive Bayes assumption is stronger.
  - Suppose we want to classify NEs as PERSON or LOCATION
  - Features: string and context
    - *located on Peachtree Street*
    - *Dr. Walker said ...*

$$P(x_1 = \textit{street}, x_2 = \textit{on} | \textsc{loc}) \\ \approx P(x_1 = \textit{street} | \textsc{loc}) P(x_2 = \textit{on} | \textsc{loc})$$

Cotraining makes the bootstrapping assumptions explicit.

- Assume two, **conditionally independent**, views of a problem.
- Assume each view is sufficient to do good classification.

Sketch of learning algorithm:

- On labeled data, minimize error.
- On unlabeled data, **constrain** the models from different views to agree with each other.

Co-training example See the slides for an animated version of this. Assume we want to do named entity classification: determine whether an NE is a Location or Person. We have two views: the name itself, and its context.

	$\boldsymbol{x}^{(1)}$	$\boldsymbol{x}^{(2)}$	y
1.	Peachtree Street	located on	LOC
2.	Dr. Walker	said	PER
3.	Zanzibar	located in	? $\rightarrow$ LOC
4.	Zanzibar	flew to	? $\rightarrow$ LOC
5.	Dr. Robert	recommended	? $\rightarrow$ PER
6.	Oprah	recommended	? $\rightarrow$ PER

Algorithm

- Use classifier 1 to label example 5.
- Use classifier 2 to label example 3.
- Retrain both classifiers, using newly labeled data.
- Use classifier 1 to label example 4.
- Use classifier 2 to label example 6.

Multiview Learning is another approach in this style.

- Co-training treats the output of each view's classifier as a labeled instance for the other view.
- In multiview learning, we add a **co-regularizer** that penalizes disagreement between the views on the unlabeled instances.
- This allows us to define a single objective function. In the case of two-view linear regression, the function is

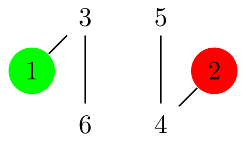
$$\begin{aligned}\min_{w,v} \sum_{i}^{L} (y_i - \boldsymbol{w}^\top \boldsymbol{x}_i^{(1)})^2 + (y_i - \boldsymbol{v}^\top \boldsymbol{x}_i^{(2)})^2 + \lambda_1 ||\boldsymbol{w}||^2 + \lambda_1 ||\boldsymbol{v}||^2 \\ + \lambda_2 \sum_{i=L+1}^{L+U} (\boldsymbol{w}^\top \boldsymbol{x}_i^{(1)} - \boldsymbol{v}^\top \boldsymbol{x}_i^{(2)})^2\end{aligned}$$

  The only difference from standard regression is the co-regularizer, which penalizes disagreement on the unlabeled data.

- An early version of this idea is **co-boosting** [CS99], where each view is a boosting classifier, and features are added incrementally to each view.

2.4 Graph-based approaches

Let's go back to sentiment analysis in French.



We can view this data as a **graph**, with edges between similar instances. Unlabeled instances propagate information through the graph.

Where does the graph come from?

- Sometimes there is a natural similarity metric (time, position in the document).

- Otherwise, we can compute similarity from features. If the features are Gaussian, we could say:

$$\text{sim}(i,j) = \exp\left(-\frac{||x_i - x_j||^2}{2\sigma^2}\right)$$

  If the features are discrete, we might use KL-divergence.

- Then we add an edge between i and j when $\text{sim}(i,j) > \tau$

Given a graph with edge weights s_{ij}, we can formulate semi-supervised learning as an optimization problem:

$$y_i \in \{0,1\}$$
$$\text{Fix } Y_l = \{y_1, y_2, \dots y_\ell\}$$
$$\text{Solve for } Y_u = \{y_{\ell+1}, \dots, y_{\ell+m}\}$$
$$\min_{Y_u} \sum_{i,j} s_{ij}(y_i - y_j)^2$$

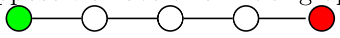
- This looks like a combinatorial problem. Specifically, it looks like (binary) integer linear programming, which is NP-complete.
- But assuming $s_{ij} \geq 0$, this specific problem can be reformulated as maximum-flow, with polynomial time solutions.

Rao and Ramichandran [RR09] apply this idea to expanding polarity lexicons.

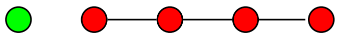
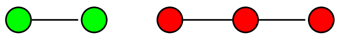
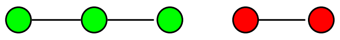
- Nodes are words
- Edges are wordnet relations
- They label a few nodes for sentiment polarity, and propagate those labels to other parts of the graph.
- However, they use a slightly modified version of the mincut idea: randomized min-cut [BLRR04].

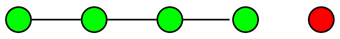
2.5 Randomized min-cut

Suppose we have this initial graph:



What is the solution? Actually, the following solutions are all equivalent:

- 
- 
- 

- 

Another problem with mincuts is that it doesn't distinguish high-confidence and low-confidence predictions. Both of these problems can be dealt with by randomization:

- Add random noise to adjacency matrix.
- Rerun mincuts multiple times.
- Deduce the final classification by voting.

2.6 Label propagation

A related approach is **label propagation** [ZG02], which Rao and Ravichandran also consider. The basic idea is that we relax y_i from an integer $\{0, 1\}$ to a real number $\mathbb{R}$. Then we solve the optimization problem,

$$\min_Y \sum_{i,j} s_{ij}(y_i - y_j)^2$$
$$s.t. Y_L \text{ is clamped to initial values}$$

The advantages are:

- a unique global optimum
- a natural notion of confidence: distance of y_i from 0.5

Let's look at the objective:

$$\begin{aligned} J =& \frac{1}{2}\sum_{i,j} s_{ij}(y_i - y_j)^2 \\ =& \frac{1}{2}\sum_{i,j} s_{ij}(y_i^2 + y_j^2 - 2y_i y_j) \\ =& \sum_i y_i^2 \sum_j s_{i,j} - \sum_{i,j} s_{ij} y_i y_j \\ =& \boldsymbol{y}^\mathsf{T}\mathbf{D}\boldsymbol{y} - \boldsymbol{y}^\mathsf{T}\mathbf{S}\boldsymbol{y} \\ =& \boldsymbol{y}^\mathsf{T}\mathbf{L}\boldsymbol{y} \end{aligned}$$

We have introduced three matrices

- Let $\mathbf{S}$ be the $n \times n$ similarity matrix.
- Let $\mathbf{D}$ be the **degree matrix**, $d_{ii} = \sum_j s_{ij}$. $\mathbf{D}$ is diagonal.
- Let $\mathbf{L}$ be the unnormalized **graph Laplacian** $\mathbf{L} = \mathbf{D} - \mathbf{S}$

- So we want to minimize $\boldsymbol{y}^\top \mathbf{L} \boldsymbol{y}$ with respect to $\boldsymbol{y}_u$, the labels of the unannotated instances.

In principle, this is easily solveable:

- Partition the Laplacian $\mathbf{L} = \begin{bmatrix} \mathbf{L}_{\ell\ell} & \mathbf{L}_{\ell u} \\ \mathbf{L}_{u\ell} & \mathbf{L}_{uu} \end{bmatrix}$
- Then the closed form solution is $\boldsymbol{y}_u = -\mathbf{L}_{uu}^{-1} \mathbf{L}_{u\ell} \boldsymbol{y}_\ell$
- This is great ... if we can invert $\mathbf{L}_{uu}$.

In practice, $\mathbf{L}_{u,u}$ is huge, so we can't invert it unless it has special structure. Zhu and Ghahramani [ZG02] propose an iterative solution called **label propagation**.

- Let $\mathbf{T}_{ij} = \frac{s_{ij}}{\sum_k s_{kj}}$, row-normalizing $\mathbf{S}$.
- Let $\mathbf{Y}$ be an $n \times C$ matrix of labels, where C is the number of classes.
- Until tired,
  - Set $\mathbf{Y} = \mathbf{T}\mathbf{Y}$
  - Row-normalize $\mathbf{Y}$
  - Clamp the seed examples in $\mathbf{Y}$ to their original values
- There's a flavor of EM here, with $\mathbf{Y}$ representing our belief $q_i(y_i)$. But there's no M-step in which we update model parameters. That's because we're in a graph-based framework, and we're assuming the graph is correct.

Both mincut and label propagation are **transductive** learning algorithms: they learn jointly over the training and test data. This is fine in some settings, but not if you want to train a system and then apply it to new test data later — you'd have to retrain it all over again.

Manifold regularization [BNS06] addresses this issue, by learning functions that are smooth on the "graph manifold." In practice, this means that they give similar labels to nearby datapoints in the unlabeled data. Suppose we are interested in learning a classification function f. Then we can optimize:

$$\arg\min_f \frac{1}{\ell} \sum_i \ell(f(\boldsymbol{x}_i), y_i) + \lambda_1 ||f||^2 + \lambda_2 \sum_{i,j} s_{ij} (f(\boldsymbol{x}_i) - f(\boldsymbol{x}_j))^2$$

- The first term corresponds to the loss on the labeled training data; we can use any convex loss functions, such as logistic or hinge loss.
- The second term corresponds to the smoothness, akin to regularizing the weights in a linear classifier.
- The third term penalizes making different predictions for similar instances in the unlabeled data

The representer theorem guarantees that we can solve for f as long as ℓ is convex. We can then apply f to any new unlabeled test data.

3 Domain adaptation

In domain adaptation, we have a lot of labeled data, but it's in the wrong domain. Some features will be shared across domains. For example, if we are classifying movies or toasters, *good* is a good word, and *sucks* is a bad word. But as we've seen, real review text is usually more subtle. What about a word like *unpredictable*? This is a good word for a movie, but not such a good word for a kitchen appliance.

3.1 Supervised domain adaptation

In supervised domain adaptation (transfer learning), we have:

- Lots of labeled data in a "source" domain, $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{\ell_S} \sim \mathcal{D}_S$ (e.g., reviews of restaurants)
- A little labeled data in a "target" domain, $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{\ell_T} \sim \mathcal{D}_T$ (e.g., reviews of chess stores)

Here are some (surprisingly-competitive) baselines (see slides)

- Source-only: train on the source data, apply it to the target data.
- Target-only: forget the source data, just train on the limited target data.
- Big blob: merge the source and target data into a single training set. Optionally downweight the source data.
- Prediction: train a classifier on the source data, use its prediction as a feature in the target data.
- Interpolation: train two classifiers, combine their outputs

Here are two less-obvious approaches:

Priors :
Train a (logistic-regression) classifier on the source data. Treat its weights as the priors on the target data, and regularize towards these weights rather than towards zero (Chelba and Acero 2004).

Feature augmentation Create copies of each feature, for each domain and for the cross-domain setting.

- The copies fire in the appropriate domains, and the learning algorithm decides whether a feature is general or domain-specific.
- For example, suppose we have domains for Appliances and Movies, and features *outstanding* and *sturdy*. We replicate the features, obtaining

$$\langle \textit{outstanding}, \textsc{App.}\rangle, \langle \textit{outstanding}, \textsc{Mov.}\rangle, \langle \textit{outstanding}, \textsc{All}\rangle$$
$$\langle \textit{sturdy}, \textsc{App.}\rangle, \langle \textit{sturdy}, \textsc{Mov.}\rangle, \langle \textit{sturdy}, \textsc{All}\rangle$$

- Ideally, we will learn a positive weight for ⟨*outstanding*, ALL⟩, because the feature works in both domains, and a small weight for the domain-specific copies of the *outstanding* feature.
- We will also learn a positive weight for ⟨*sturdy*, APP⟩, because the feature works only in the Appliance domain.

See slides for a diagram of how this works.

3.2 Unsupervised domain adaptation

Without labeled data in the target domain, can we learn anything? If the source and target domain are somewhat related, then we can. A very popular approach is structural correspondence learning (SCL) [BDP07].

- Suppose there are a few words that are good predictors in both domains; we'll call these **pivot features**
- Pivot features can be selected by finding words that are
  - Popular in both domains
  - High mutual-information with the label in the source domain
- The label is unknown in the target domain, so we can't learn to predict it. Instead we'll predict the pivots. We train a linear classifier for each pivot, obtaining weights $\boldsymbol{w}_n$ for pivot n.
- For example, we can learn that the domain-specific feature *fast-multicore* is a good predictor of the pivot *excellent*.
- We can horizontally concatenate the pivot predictor weights, forming

$$W = [\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_N] \tag{3}$$

- The matrix W is large, and contains redundant information (since many pivots are closely related to each other). We factor $W \approx USV^T$ using singular value decomposition (SVD).
- We use U to **project** features from both domains into a shared space, $U^\top \boldsymbol{x}$.
- We then learn to predict the label in the source domain, using the augmented instance $\langle \boldsymbol{x}, U^\top \boldsymbol{x} \rangle$. In U contains meaningful correspondences between the domains, then the weights learned on these features will work for the target domain instances too.
- This idea yields substantial improvements in adapting sentiment classifiers across product domains [BDP07].

See the slides for a graphical explanation of these ideas, with slightly different notation.

4 Other learning settings

There are many other settings in which we learn from something other than in-domain labeled data:

- **Active learning**. The model can query the annotator for labels (see above)
- **Feature labeling**. Annotators label *features* rather than instances. For example, you provide a list of five prototype words for each POS tag [HK06].
- **Feature expectations**. Learn from *constraints* on feature-label relationships; for example, the word "the" is a determiner at least 90% of the time. In EMNLP 2013, this idea was applied to multilingual learning (which I'll discuss in the final lecture). The basic idea of this paper is to align words between sentences and insist that aligned words have the same tag most of the time.
- **Multi-instance learning**. The learner gets a "bag" of instances, and a label. If the label is positive, then at least one instance in the bag is positive, but you don't know which one.

  This idea is often related to **distant supervision**. The learner gets a label indicating that there is a relationship, such as BORN-IN(OBAMA, HAWAII), and a set of instances containing sentences that mention the two arguments, *Obama* and *Hawaii*. Many of these sentences do not actually instantiate the desired relation (e.g., *Obama visited Hawaii in 2008...*), but we assume that at least one does, and we must learn from this.

References

[BDP07] John Blitzer, Mark Dredze, and Fernando Pereira. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 440–447, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

[BLRR04] Avrim Blum, John Lafferty, Mugizi Robert Rwebangira, and Rajashekar Reddy. Semi-supervised learning using randomized mincuts. In *Proceedings of the twenty-first international conference on Machine learning*, page 13. ACM, 2004.

[BNS06] Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *The Journal of Machine Learning Research*, 7:2399–2434, 2006.

[CS99] Michael Collins and Yoram Singer. Unsupervised models for named entity classification. In *Proceedings of the joint SIGDAT conference on empirical methods in natural language processing and very large corpora*, pages 189–196, 1999.

[HK06] Aria Haghighi and Dan Klein. Prototype-driven learning for sequence models. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 320–327. Association for Computational Linguistics, 2006.

[RR09] Delip Rao and Deepak Ravichandran. Semi-supervised polarity lexicon induction. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 675–682. Association for Computational Linguistics, 2009.

[ZG02] Xiaojin Zhu and Zoubin Ghahramani. Learning from labeled and unlabeled data with label propagation. Technical report, Technical Report CMU-CALD-02-107, Carnegie Mellon University, 2002.