CIS5200: Machine Learning Spring 2023

Lecture 20: Semi-supervised/Active Learning

Date: March 30, 2023 *Author: Surbhi Goel*

Acknowledgements. These notes are heavily inspired by notes by Nina Balcan (CMU).

Disclaimer. These notes have not been subjected to the usual scrutiny reserved for formal publications. If you notice any typos or errors, please reach out to the author.

1 Beyond Fully Supervised Learning

So far we have looked at several methods for supervised learning where the machine learning algorithm has access to labelled examples. This includes tasks like regression and classification using algorithms such as decision trees, logistic regression, linear regression, SVMs, etc.

This paradigm makes an assumption that we have access to human annotators or domain experts that can provide good quality labels for our input examples. However, as the amount of data available has increased drastically over time (think about all the websites, photos, text, videos available on the internet), this assumption is hard to satisfy. Due to the high resource cost of labeling data, low supply of domain experts, and the potential of errors and inconsistencies, the demand for ML methods to make the most efficient use of labelled data and tap into the large amount of unlabelled data has become increasingly important.

On the opposite extreme of supervised learning lies unsupervised learning which has no access to labels. This paradigm is useful for extracting structure or patterns from data, but without the guiding signal from the labels, it can lead to much lower performance. Today we will discuss two paradigms of learning: semi-supervised learning and active learning, that lie in between unsupervised and supervised learning.

2 Semi-supervised Learning

In semi-supervised learning, we are given access to a large set of unlabelled examples and a small set of labelled training examples. More formally,

Data: Labelled training set $S_l = \{(x_1, y_1), \ldots, (x_{m)l}, y_{m_l})\}$ and unlabelled training set $S_u = \{x'_1, \ldots, x'_{m_u}\}$ with $m_u >> m_l$.

Goal: Find a predictor that has good performance on the underlying distribution the data is drawn from.

A natural question to ask is: *why is the unlabelled data helpful?* For the unlabelled data to be helpful, there must be some relationship between the labels and the underlying data distribution. Some examples of assumptions include: smoothness (points close to each other have similar labels),

Figure 1: An example to show how unlabelled data can be helpful to find a better classifier. The green circles and purple triangles indicate labelled examples and the gray boxes are unlabelled examples. On the left is the SVM solution with only the labelled examples, and on the right is the solution using the unlabelled examples.

clustering (points in the same cluster have the same label), margin (the classifier has large margin with respect to the distribution).

Here we will discuss a few methods for semi-supervised learning.

2.1 Self-training

One of the oldest algorithms for semi-supervised learning is self-training, dating back to 1960s. The idea is pretty simple:

1. Use your labelled training data to learn a classifier
2. Use this classifier to label all the unlabelled training data points
3. Of these, choose a subset of the most *confident* predictions, assign them pseudo-labels according to the classifier
4. Add these to the labelled dataset and repeat.

This method can work as wrapper on any supervised machine learning method however it has some major issues. If the pseudo-labels are incorrect then this could bias the model into trusting those and lead to a cascade of errors in future rounds. To address some of these concerns, there are other approaches such as co-training (by Tom Mitchell and Avrim Blum) that uses two classifiers and using each classifier's predictions to improve the other classifier.

2.2 Semi-supervised/Transductive SVM

Suppose we make the assumption that the labelling function has large margin with respect to the data distribution. Then we can use the approach by Joachims from 1999, that modifies the standard SVM algorithm to include the unlabelled data. See Figure 1 to see why this would be helpful.

Transductive SVM modifies the original SVM objective by adding two additional constraints:

$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2}\|w\|_2^2 \\ \text{such that} \quad & y_i(w^\top x_i + b) \geq 1, \forall i \in [m_l] \\ & \textit{Constraints from unlabelled data:} \\ & y_i'(w^\top x_i' + b) \geq 1, \forall i \in [m_u] \\ & y_i' \in \{-1, 1\}, \forall i \in [m_u] \end{aligned}$$

The idea here is to maximize margin under all possible labelling of the unlabelled data.

The main challenge is that adding the last constraint makes the problem non-convex and NP hard to solve. If we could guess the labels y_i', then the problem would be convex again. There is a heuristic method to find the local optimum solution:

1. First maximize margin with only labelled samples
2. Use the w, b found to label the unlabelled samples
3. Try flipping labels of unlabelled points and see if margin increases
4. Keep going till no more improvements

Transductive SVMs don't always work. The setting in Figure 1 is ideal for the algorithm, however if the underlying problem has no margin or there are multiple large margin solutions then this approach may fail. In the case of multiple margin solutions, without enough labelled points, it will be hard to tell which large margin classifier is actually the correct one.

3 Active Learning

In active learning, we start with a large set of unlabelled data and we can adaptively decide which unlabelled examples we want the labels for. The goal is to minimize the number of label requests while ensuring that we find a good predictor.

More formally,

Data: Unlabelled training set $S = \{x_1, \ldots, x_m\}$.

Query Access: Query a subset of unlabelled examples of size m_l. m_l denotes the *label complexity* of the algorithm.

Goal: Find a predictor that has good performance on the underlying distribution the data is drawn from with $m_l << m$.

This is known as *pool-based* active learning. There is an online version of this known as *selective sampling* active learning where the learner gets one sample at a time and has to decide whether to ask for the label.

A natural question to ask is: *why is adaptive querying more helpful than access to random labels?* In many settings where all data points are not equally important, querying the most *informative* points

or the ones we are most *uncertain* about can drastically reduce the need for labels for several other points. For example, in the setting of linear classification, points closer to the decision boundary are more informative than points away from the decision boundary.

3.1 Example: Thresholds

Let us consider a simple example to show the benefit of active learning. Recall the class of linear thresholds defined by a parameter a, $f_a(x) = \begin{cases} 1 & \text{if } x \geq a \\ -1 & \text{otherwise.} \end{cases}$. Recall that the VC dimension of this class is 1 so $O(1/\epsilon)$ samples are enough to learn this class assuming realizable setting. Thus we only need to find the ERM for these many samples.

In the passive setting of supervised learning, we would need the labels of all these points to actually find the ERM. However, if we were allowed to adaptively choose the points to label, we could do this with only $O(\log(1/\epsilon))$ labels. This is an exponential saving. Consider the following binary search algorithm:

1. Let S' be the points we do not know the labels for. Set $S' = S$.
2. Query the median point in S'.
3. If it is positive then update S' to keen only the points that are less than the median. If it is negative then update S' to keen only the points that are greater than the median.
4. Repeat till S' is empty.

Note that at each step, we throw away at least half the points, therefore, the algorithm, will terminate in $\log m = O(\log(1/\epsilon))$ steps and we would have queried that many points. It is not too hard to see why this works.

3.2 Active SVM

The idea behind Active SVM (Tong and Koller'00) is to query points closest to the decision boundary and keep updating the classifier. Here the idea is that points closest to the decision boundary are the ones we are most uncertain about. More formally,

1. Query a few random examples to start
2. Repeat for T iterations:
   (a) Find the max-margin classifier for all the labelled examples so far
   (b) Identify the closest unlabelled example to the decision boundary and query its label

This simple algorithm is pretty useful in practice however it does not always work. Since we are making local greedy decisions, we can end up introducing *sampling bias* where the points we are choosing to label are no longer representative of the underlying distribution. For example, see the example in Figure 2.

There are alternate ways to decide which points to query, such as:

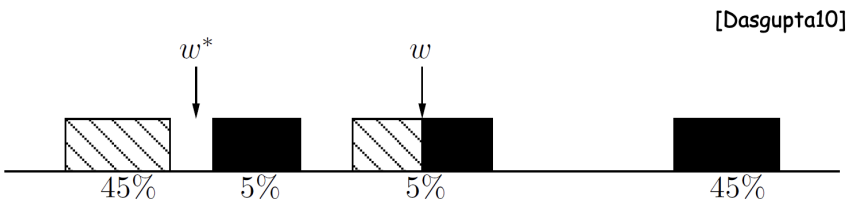

Figure 2: An example to show how active learning strategies can induce sampling bias. Here the stripped regions indicate negative points and the filled black regions indicate positive points. The distribution mass in each of these regions is indicated by the percentage below it. The points are uniformly distributed in each region. Now if we sample a few points initially, with very high probability, we will only get points in the leftmost and rightmost regions. Therefore, we will learn w as the max-margin solution. Subsequently, we will end up querying only the points in the neighborhood, and as long as we query a negative point near before the positive, we won't change the classifier. However w has an error of 5% and w^* has an error of only 2.5%. Therefore, we would have found a sub-optimal solution.

1. *Density-based*: Choose the center of the largest unlabelled cluster.
2. *Maximal Diversity*: Choose the points furthest away from labelled points.
3. *Ensemble*: Combine various pairs, for example, uncertain and maximal, far off points from labelled examples that are also close to the decision boundary.