Unsupervised Learning in Computational Linguistics

handout for the introductory lecture

October 23, 2017

On unsupervised learning

Most machine learning applications in computational linguistics are based on supervised methods. To train a supervised machine learning algorithm, we need a training set with labels that we want to learn, or predict.[1] For example, if the task is predicting the part-of-speech (POS) tags, we would need a training set where the POS tag of each word is marked. The unsupervised learning methods, on the other hand, operate only on objects (e.g., without tags for the POS tagging example).

[1] Here, the term *label* refers to either a categorical label (e.g., POS tag), a complex structure (e.g., parse tree), or a numeric value (e.g., age of the author).

The fact that we do not have a ground truth (the labels) makes the unsupervised learning mostly an exploratory methodology. Hence, the primary use case is to discover whether the data has some regularities, or some underlying structure. However, the unsupervised learning can be used more than exploring the data (more on this below).

There are a number of well-known well-studied methods in unsupervised machine learning.

clustering is the task of grouping a set of objects without using any explicit group labels. In clustering, we (typically) assign each object to a single group.

k-means, hierarchical clustering

density estimation is a related method, where we assume that the data is generated by a number of (parametric) probability distributions, and try to estimate the parameters of these distributions.

mixtures of Gaussians, EM algorithm

dimensionality reduction aims to reduce the number of dimensions (or features), while retaining (most of the) information in the data set. In other words, each object, expressed originally as vectors/points in high dimensional space, is mapped to a lower dimensional space. This is particularly useful for visualization, but also helps reducing the computational cost of processing the data.

PCA, autoencoders

The above 'classical' methods have a long-tradition in many branches of science and they are studied thoroughly. It turns out, however, all of these methods (and many more) can be cast as estimating a *probabilistic model*. Probabilistic models use the rules of probability theory to model the phenomenon of interest, and they are also known as (probabilistic) *graphical models* since they can conveniently be expressed as graphs. A popular subclass of these models (directed graphical models) are also known as *Bayesian networks*. The general idea behind using probabilistic models in an unsupervised fashion is to predict unobserved, or latent, variables (for example as underlying groups or clusters). The probabilistic models are used in a number of CL/NLP applications (e.g., latent Dirichlet allocation, LDA), but probably less than their potential in other applications. Furthermore, some non-

probabilistic models (e.g., some neural network models) can be seen as approximations to certain probabilistic models.[2]

In machine learning, 'unsupervised' means that the model (or the method) does not require a target value during training. In this course, we take a more liberal definition of 'unsupervised'. We are interested in methods that do not require any explicit annotation.

[2] Short message from this long paragraph: probabilistic models are good, we want to see them in this seminar.

How/why does unsupervised learning work?

Although we already listed a few methods above for unsupervised learning, it is fair to ask how can one learn without supervision. In fact, there is a sobering fact, known as *no-free-lunch theorem*, that roughly states that there is no general learning mechanism that can be used in any learning problem. Learning requires some specific knowledge about the problem at hand. Obviously, the labels provide the necessary problem-specific information for supervised learning. For unsupervised learning, however, we do not have labels. As a result, it is more crucial for unsupervised methods to encode the information necessary into the model structure and/or the features used to represent the objects of interest.

Why do we care? Do we really need unsupervised methods?

The positive answer to the question is mainly supported by the fact that labels are not easy to obtain for most problems. Obtaining the labels (annotating the data with labels) is far from trivial or cheap for most applications - even with the use of crowd-sourcing platforms. And more data is invariably helpful for machine learning models. 'More unlabeled data' is, on the other hand, not a problem in most areas of CL. The good news is that it is often possible to combine both: while using an unsupervised method to exploit the large quantities of data, we may use a relatively small amount of supervised data results better than any of the individual approaches.

Besides the cost of annotating data, sometimes we do not have a reliable way to annotate the data at hand (think about analyzing a historical language). Even if expert annotations are possible, every annotation process means introducing some amount of bias into the solution. Hence, in some cases, we may want to let 'the data speak by itself' under certain (explicit) modeling assumptions, in which case unsupervised methods are also useful.

What do we (not) do in this course?

First, the answer of the negative question: this course is not a systematic introduction to unsupervised machine learning, nor it is a systematic introduction introduction to use of unsupervised methods in computational linguistics.

In this course, the participants are expected to[3]

[3] Details of some of these steps may depend on the number of participants and/or individual agreements. In any case, however, *we will proceed quickly*!

- choose a topic relevant to computational linguistics,
- find out state-of-the art in the solution of the problem, including the use of unsupervised methods,
- introduce/discuss the topic/problem in the class,
- experiment with unsupervised solutions to the problem (both novel ideas or reproducing, reimplementing, comparing published methods are welcome),
- present your model/method/work in the class,
- write a term paper describing the work you have done.

A short list of a few topics/problems to tackle with

The following is a partial list of possible topics you may want to work on. The list is by no means close to complete, and you are welcome to bring in your own topics.

- Typical tasks in an NLP pipeline:[4]
  - tokenization / segmentation
  - text normalization
  - POS tagging
  - Learning morphology
  - Learning syntax
  - Learning semantics of words, or larger units
  - Learning representations for linguistic objects (words, phrases, documents, …)
  - Alignment, machine translation
  - Clustering/grouping documents for a particular purpose
- Topics with more (computational/quantitative) linguistics touch (or less of natural-language engineering)
  - Modeling, exploring linguistic variation (based on, geography, social class, gender, author, age, …)
  - Computational historical linguistics
  - Simulating human language processing or acquisition
  - Testing alternative theories/hypotheses in any area of linguistics
  - …

[4] You are also welcome to work on *joint* learning of one or more of these topics.

A warm-up challenge

As an entry requirement, you are presented with a few simple unlabeled data sets. You are required to *make sense of* 4 unlabeled data sets, trying to discover the underlying groups or structure in each data set. The way you solve (or cannot solve) these puzzles will not affect your grade. However, we will discuss *your* solutions during the next class session.

To start with the warm-up exercise, go to `https://classroom.github.com/a/WInXmC2E`.[5] Once you access this link and follow up the setup process, you will have a private git repository for your solution, which will include the data files, and additional information about the task.

[5] We will try GitHub classroom in this class. For this you will need a GitHub account, and some git knowledge will probably be handy.