Lecture 23: Representation Learning

Instructor: Aditya Bhaskara *Scribe: Andrew Fraser*

CS 5966/6966: Theory of Machine Learning

January 1st, 2017

Abstract

In this lecture we discuss common concepts in representation learning, comparing strategies using supervised and unsupervised learning. We discuss what makes a representation effective, as well as common examples of representation learning in NLP and graphs using word embeddings.

1 Review of Last Class

1.1 *Analyzing gradient descent*

For infinitely wide neural nets, the kernel remains "fixed". Therefore, neural network learning is equivalent to kernel regression. [Jacot et al.]

Recall: We can do this by viewing gradient descent as kernel regression for a time-varying kernel.

1.2 *Neural Networks as Feature Learning or Representation Learning*

Neural Networks transform inputs into feature space embeddings. Suppose layers $f_1, f_2, \ldots f_{n-1}$, then layer f_{n-1} captures the most important "features" or "semantic properties" of the data that are used to predict f_n. Very often, f_{n-1} is relatively sparse, even if x is high dimensional. This is an excellent way to represent x in a lower space.

These are very useful even beyond classification (e.g. image captions, transfer learning). We can use f_{n-1} and learn how to predict image features based upon it, known as image captioning.

Consider another strategy called transfer learning. Suppose we have trained a neural network for one task. Then, we use it on another similar task that has insufficient training data. We can hypothesize that the same neural net could still be useful on another similar task with not much additional training.

2 What makes a representation good?

Below is a list of generally good traits that a representation can have:

- "Auto-encoder" view: Representation should allow us to "approximately recover" input (i.e. it is somewhat invertible). This tends to hold true often for unsupervised learning. If information is invertible, then as much information is being "represented" as possible.
- Information "bottleneck": noise should not affect representation, but a large change should.

- Contrastive (for classification). Large changes result in large differences in output.
- "Disentangled" or orthogonal. Very easy to determine between two inputs.
- Sparse "explanations" for phenomena.
- Hierarchically organized, explanatory.
- Leverage domain knowledge. This is especially important in recent times.

Supervised learning is contrastive, has better accuracy for solving specific tasks, and requires no special training. However, it tends to have issues generalizing well. In supervised learning, we come in with prior knowledge about the data and can be contrastive about different targets. This is excellent for tasks where we know exactly what we want to solve.

Unsupersvised learning generalizes to many tasks and does not require careful data collection. However, it is unclear what approach to take to learning or what the initial objective is. Unsupervised learning only gives us a map, it's up to us to determine how to use the map. It is often a focus for research for this reason, as there are many new approaches that can be applied.

3 Unsupervised Representation Learning

3.1 *Classic approaches (somewhat linear)*

This includes manual feature engineering, autoencoders, and sparse coding.

3.1 definition. **Sparse coding** is a way to encode many vectors in a smaller space. Given input $x_1, x_2, \ldots, x_n \in \mathbb{R}^d$, we want a dictionary $v_1, \ldots, v_m \in \mathbb{R}^d$ such that for each x_i, $x_i \approx \sum_{j=1}^{m} a_j^{(i)} v_j$.

This approach is nice for the early layers of a neural network, but tends to be less useful for deeper networks.

3.2 *Modern approach*

This includes self-supervised learning, invariances, and data augmentation.

Neural networks are excellent at supervised learning, can we leverage supervised learning to improve unsupervised learning? This is called self-supervised learning. Very highly active area of research.

Example: NLP tasks (fill in the blanks). We have a lot of unlabeled data. We mask a small portion of the data (example: one word in a sentence) and then try to predict the word using the rest of the sentence. This can also be applied to images where we mask a few pixels and then learn them.

Rotation prediction: An image and its slightly rotated version should have the same representation. This is an example of an invariance.

4 Representation Learning in NLP

1950s result in NLP

These approaches are based upon **Firth's Hypothesis**: "A word is defined by

the company it keeps". The general strategy is to look at sequences of words, otherwise known as "N-grams".

4.1 DEFINITION. This approach is called **Latent Semantic Indexing**. An "N-gram" is an N-length sequence of words. For each w_i, look at how frequently it occurs with w_j within N words. $\alpha_{i,j}$ is the frequency with which w_i and w_j appear within the same N-gram.

Consider the matrix M containing all $\alpha_{i,j}$. By SVD low-rank approximation, $M \approx UV$, where $U \in \mathbb{R}^{Nxk}$ and $V \in \mathbb{R}^{kxN}$. Then, for any $\alpha_{i,j}$, we have vectors u_i, u_j with the property that $\alpha_{i,j} =< u_i, u_j >$.

This is very useful for keeping low-rank approximations of any $\alpha_{i,j}$. Nowadays, we don't usually build this full matrix. Instead, this is used as the basis for a neural network model.

5 Representation Learning in Graphs

Suppose we have a graph G. Spectral clustering and performing the SVD on an adjacency matrix are two approaches to analyzing G.

These two strategies have a problem: they are based directly on the adjacency matrix and therefore can only look at direct neighborhoods. Neighborhoods tell us that two points are similar, but this does not capture all similarity information in the graph. A vertex two or three neighbors away from vertex v should still be similar to v, but is not captured by adjacency matrix directly.

Solution: view a graph as producing N-grams. Perform many random walks on the graph to determine N-grams. Then, treat these N-grams as an NLP problem and learn what makes the vertices similar.

We may say a bit more on this in the next lecture, as we ran out of time.