Paper summary: Contrastive representation learning: a framework and review

May 19, 2022

1 Idea in few sentances

2 Explanation of the central concept

3 Methodology

4 Initial rambly notes

4.1 Abstract

Authors systematically go over what Contrastive representation learning is.

4.2 Introduction

Representation learning refers to the process of learning a parametric mapping from raw input data domain to a feature vector or tensor, in the hope of capturing and extracting more abstract and useful concepts that can improve performance of downstream tasks. Often this includes dimensionality reduction. The goal of representation learning is for this mapping to meaningfully generalize well on new data.

A good representation has the following properties:

1. it is locally smooth
2. it is temporally and spatially coherent in a sequence of observations
3. has multiple, hierarchically organised explanatory factors which
4. are shared across tasks
5. has simple dependencies
6. is sparsly activated for a specific input

In deep learning this became that good representation are:

1. **distributed** representations can represent an exponential amount of configuration for their size
2. **abstract and invariant**
3. **disentangled representation**: each feature should be as disentangled as from another as possible

4.3 Representation learning

The process of extracting representations from observations, or inferring latent variables in a probabilistic view of a dataset, is often called **inference**. There are **generative** and **discriminative** models.

Generative models learn representations by modelling the data distribution $p(\boldsymbol{x})$. Such a model can generate realistic examples. Evaluating the conditional distribution $p(y|\boldsymbol{x})$ it done via Bayes rule.

Discriminative models model the conditional distribution $p(y|\boldsymbol{x})$ directly. Discriminative modelling consists of first the inference that extracts latent variables $p(\boldsymbol{v}|\boldsymbol{x})$ which are then used to make downstream decision from those variables $p(y|\boldsymbol{v})$.

The benefit of discriminative models are that you don't have to go through an expensive process of learning $p(\boldsymbol{x})$. That's also harder to evaluate. This is especially evident if you just want a lower dimensional distribution.

4.4 Contrastive representation learning

Intuitively, it's learning by comparing. So instead of needing data labels y for datapoints $\boldsymbol{x}$, you need to define a similarity distribution which allows you to sample a positive input $\boldsymbol{x}^+ \sim p^+(\cdot|\boldsymbol{x})$ and a data distribution for a negative input $\boldsymbol{x}^- \sim p^-(\cdot|\boldsymbol{x})$, with respect to an input sample $\boldsymbol{x}$. "Similar" inputs should be mapped close together, and "dissimilar" samples should be mapped further away in the embedding space.

Let's explain how this would work with the example of image-based instance discrimination. The goal is to learn a representation by maximizing agreement of the encoded features (embeddings) between two differently augmented views of the same images, while simultaneously minimizing the agreement between different images. To avoid model maximizing agreement through low-level visual cues, views from the same image are generated through a series of strong image augmentation methods. Let $\mathcal{T}$ be a set of image transformation operations where $t, t' \sim \mathcal{T}$ are two different transformations sampled independently from $\mathcal{T}$. There transformations include ex. cropping, resizing, blurring, color distortion or perspective distortion. A $(\boldsymbol{x}_q, \boldsymbol{x}_k)$ pair of query and key views is positive when these 2 views are created with different transformations on the same image, i.e. $\boldsymbol{x}_q = t(\boldsymbol{x})$ and $\boldsymbol{x}_k = t'(\boldsymbol{x})$, and is negative otherwise. A feature encoder $e(\cdot)$ then extracts feature vectors from all augmented data samples $\boldsymbol{v} = e(\boldsymbol{x})$. This is usually ResNet, in which case $\boldsymbol{v} \in \mathcal{R}^d$ is the output of the average pooling layer. Each v is then fed into a projection head $h(\cdot)$ made up of a small

multi-layer perceptron to obtain a metric embedding $\boldsymbol{z} = h(\boldsymbol{v})$, where $\boldsymbol{z} \in \mathcal{R}^{d'}$ with $d' < d$. All vectors are then normalized to be unit vectors. Then you take a batch of these metric embedding pairs $\{(\boldsymbol{z}_i, \boldsymbol{z}'_i)\}$, with $(\boldsymbol{z}_i, \boldsymbol{z}'_i)$ being the metric embeddings of $(\boldsymbol{x}_q, \boldsymbol{x}_k)$ of the same image are fed into the contrastive loss function which does what we said 3 times already. The general form of popular loss functon such as InfoNCE and NT-Xent is:

$$\mathcal{L}_i = -\log \frac{\exp(\boldsymbol{z}_i^T \boldsymbol{z}'_i/\tau)}{\sum_{j=0}^{K} \exp(\boldsymbol{z}_i \cdot \boldsymbol{z}'_j)/\tau} \tag{1}$$

where τ is the temperature parameter. The sum is over one positive and K negative pairs in the same minibatch.

4.5 Other stuff

This is just the first 20% of the paper. The rest goes into different application of contrastive learning and different losses, encoder and head architectures etc.

An interesting application to us is contrastive learning on sequences. A common assumption there is the slowness principle, whereby you say that a small subsequence of observations is a positive example because things don't change that quickly. Another subsequence found in some other time-slice is then a negative example. This is used in for example multi-frame Time-Contractive Network (TCN). Of course you can also do the previously described transformations on those video frames.