

Lecture 13 Self-Supervised Learning

sgc

March 27, 2022

1 Generative vs Self-supervised Learning

Both aim to learn from data without manual label annotation.

Generative learning aims to model data distribution $p_{data}(x)$, e.g., generating realistic images.

Self-supervised learning methods solve pretext tasks that produce good features for downstream tasks.

Learn with supervised learning objectives, e.g., classification, regression.

Labels of these pretext tasks are generated automatically

2 Pretext tasks from image transformation

Predict rotations:

Predict relative patch locations:

Solve jigsaw puzzles:

Predict missing pixels:

Image coloring:Idea: cross-channel predictions.

Video coloring: Idea: model the temporal coherence of colors in videos.

Main method :

1, Context based

2,Temporal Based

3,Contrastive Based

3 Contrastive representation learning

Give a reference score: we want : $score(f(x), f(x^+)) \gg score(f(x), f(x^-))$

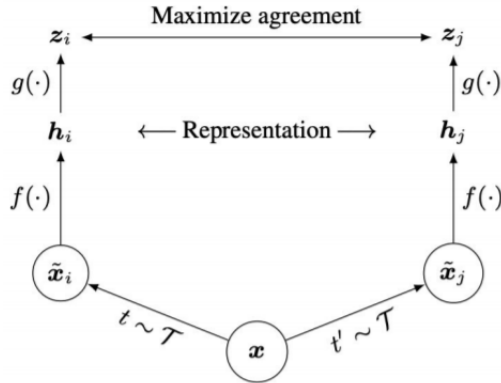
x reference sample;

x^+ positive sample;

x^- negative sample;

Given a chosen score function, we aim to learn an encoder function f that yields high score for positive pairs (x, x^+) and low scores for negative pairs (x, x^-) .

SimCLR:



SimCLR

Algorithm 1 SimCLR's main learning algorithm.

```

input: batch size  $N$ , constant  $\tau$ , structure of  $f, g, \mathcal{T}$ .
for sampled minibatch  $\{\mathbf{x}_k\}_{k=1}^N$  do
  for all  $k \in \{1, \dots, N\}$  do
    draw two augmentation functions  $t \sim \mathcal{T}, t' \sim \mathcal{T}$ 
    # the first augmentation
     $\tilde{\mathbf{x}}_{2k-1} = t(\mathbf{x}_k)$ 
     $\mathbf{h}_{2k-1} = f(\tilde{\mathbf{x}}_{2k-1})$  # representation
     $\mathbf{z}_{2k-1} = g(\mathbf{h}_{2k-1})$  # projection
    # the second augmentation
     $\tilde{\mathbf{x}}_{2k} = t'(\mathbf{x}_k)$ 
     $\mathbf{h}_{2k} = f(\tilde{\mathbf{x}}_{2k})$  # representation
     $\mathbf{z}_{2k} = g(\mathbf{h}_{2k})$  # projection
  end for
  for all  $i \in \{1, \dots, 2N\}$  and  $j \in \{1, \dots, 2N\}$  do
     $s_{i,j} = \mathbf{z}_i^\top \mathbf{z}_j / (\|\mathbf{z}_i\| \|\mathbf{z}_j\|)$  # pairwise similarity
  end for
  define  $\ell(i, j)$  as  $\ell(i, j) = -\log \frac{\exp(s_{i,j}/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(s_{i,k}/\tau)}$ 
   $\mathcal{L} = \frac{1}{2N} \sum_{k=1}^N [\ell(2k-1, 2k) + \ell(2k, 2k-1)]$ 
  update networks  $f$  and  $g$  to minimize  $\mathcal{L}$ 
end for
return encoder network  $f(\cdot)$ , and throw away  $g(\cdot)$ 
  
```

Generate a positive pair by sampling data augmentation functions

Iterate through and use each of the $2N$ sample as reference, compute average loss

*We use a slightly different formulation in the assignment. You should follow the assignment instructions.

InfoNCE loss: Use all non-positive samples in the batch as \mathbf{x}^-

Source: [Chen et al., 2020](#)

Generate two similar image from one sample and minimize the difference between them.

Generate positive samples through augmentation: cropping, color distortion, blur.

Use cos function as the score. ($s_{i,j}$ in the picture)

Extra non-linear projection improve the representation learning. Why? Use h to do downstream works.

Momentum Contrastive Learning (MoCo)

Main operation: Dictionary as a queue, Momentum update, Shuffling BN.
 Dictionary as a queue: Keep a running queue of keys, update after every epoch.
 Momentum update: $\theta_k < -m\theta_k + (1 - m)\theta_q$
 Shuffling BN:
 Now we have MoCov2 and MoCov3.
 Above we based on positive and negative instances.

Contrastive Predictive Coding(CPC) (Base on sequential/temporal orders) mainly videos:

1. Encode all samples in a sequence into vectors $z_t = g_{enc}(x_t)$
2. Summarize context into a context code c_t using an auto-regressive model (g_{ar})
3. Compute InfoNCE loss between the context c_t and future code z_{t+k} using the following time-dependent score function: $s_k(z_{t+k}W_kc_t) = z_{t+k}^tW_kc_t$, where W_k is a trainable matrix.(use c_t to predict future z_{t+k} to test if c_t is good enough)

