Contrastive Learning

Blog

- Contrastive Self-Supervised Learning [Local link] (./blog-contrastive self-supervised learning.pdf) [Internet link] https://ankeshanand.com/blog/2020/01/26/contrative-self-supervised-learning.html
- 对比学习（Contrastive Learning）相关进展梳理

  [Internet link] https://zhuanlan.zhihu.com/p/141141365
- 对比学习（Contrastive Learning）:研究进展精要

  [Internet link] https://zhuanlan.zhihu.com/p/367290573
- 论文阅读 | 浅谈图上的自监督学习——对比学习

  [Internet link] https://zhuanlan.zhihu.com/p/187247235
- 深度学习中的互信息

  [Internet link] <深度学习中的互信息：无监督提取特征 - 知乎 (zhihu.com)>

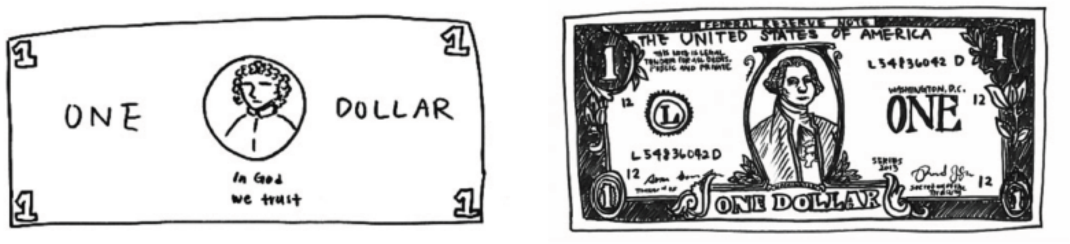
Note

Introduction

```
Contrastive self-supervised learning techniques are a promising class of methods that build representations by learning to encode what makes two things similar or different.
```

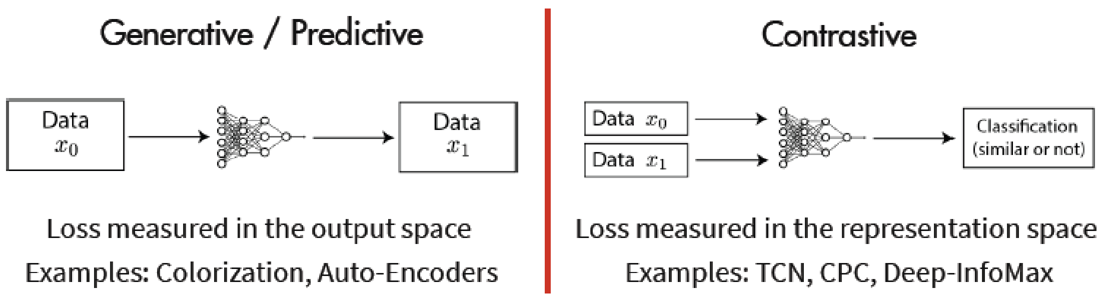
其核心思想是学会编码相似事物的不同之处。

- Example



左边是根据记忆画出来的美元，右边是照着实物画出来的，尽管我们看过很多次，依然无法完全一模一样画出来，但是我们可以有足够的特征将其与其它物体区分开。那么我们是否可以不根据元素级别的细节，只通过编码高层次特征来区分不同事物？

- Genrative vs contrastive method



- How do contrastive methods

More formally, for any data point $\boldsymbol{x}$, contrastive methods aim to learn an encoder $\boldsymbol{f}$ such that:

$$\underbrace{\operatorname{score}(f(x), f(x^+))}_{} >> \underbrace{\operatorname{score}(f(x), f(x^-))}_{}$$

- here $\boldsymbol{x}^+$ is data point similar or congruent to $\boldsymbol{x}$, referred to as a *positive* sample.
- $\boldsymbol{x}^-$ is a data point dissimilar to $\boldsymbol{x}$, referred to as a *negative* sample.
- the **score** function is a metric that measures the similarity between two features.

InfoNCE loss

$$\mathcal{L}_N = -\mathbb{E}_X\left[\log \frac{\exp\left(f(x)^T f\left(x^+\right)\right)}{\exp\left(f(x)^T f\left(x^+\right)\right) + \sum_{j=1}^{N-1} \exp\left(f(x)^T f\left(x_j\right)\right)}\right]$$

- 如何定义目标函数？最简单的一种就是上面提到的内积函数，另外一中 triplet 的形式就是 $l = max(0, \eta + s(x, x^+) - s(x, x^-))$ ，直观上理解，就是希望正例 pair 和负例 pair 隔开至少 η 的距离，这一函数同样可以写成另外一种形式，让正例 pair 和负例 pair 采用不同的 s 函数，例如，$s(x, x^+) = \| \max(0, f(x) - f(x^+) \|$ ，$s(x, x^+) = \| \max(\eta, f(x) - f(x^-) \|$。
- 如何构建正例和负例？针对不同类型数据，例如图像、文本和音频，如何合理的定义哪些样本应该被视作是 x^+ ，哪些该被视作是 x^- ，；如何增加负例样本的数量，也就是上面式子里的 N ？这个问题是目前很多 paper 关注的一个方向，因为虽然自监督的数据有很多，但是**设计出合理的正例和负例 pair，并且尽可能提升 pair 能够 cover 的 semantic relation，才能让得到的表示在 downstream task 表现的更好。**

Contrastive predictive coding (CPC)

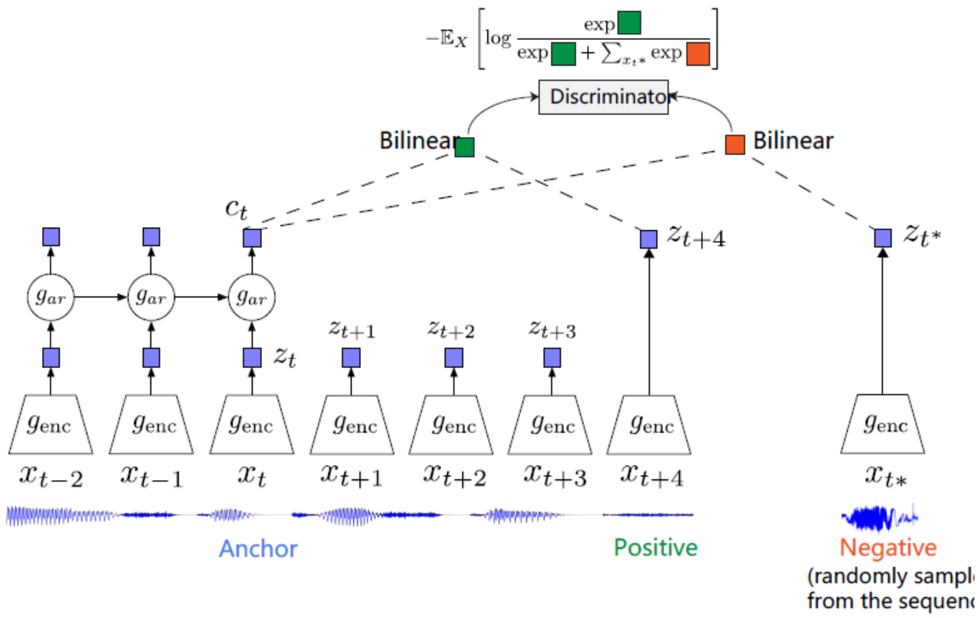
- **Problem**: Improving representation learning requires features that are less specialized towards solving a single supervised task.
- **Intuition**: learn the representations that encode the underlying shared information between different parts of the (high-dimensional) signal.
- **Method**: CPC learns representations by encoding information that's shared across data points multiple time steps apart, discarding local information. These features are o□en called "slow features": features that don't change too quickly across time. Examples include identity of a speaker in an audio signal, an activity carried out in a video, an object in an image etc.

The contrastive task in CPC is set as follows. Let $\{x_1, x_2, ..., x_N\}$ be a sequence of data points, and x_t be an anchor data point. Then,

- x_{t+k} will be a positive sample for this anchor.
- A data point x_{t^*} randomly sampled from the sequence will be a negative sample.

CPC makes use of multiple ks in a single task to capture features evolving at different time scales.

When computing the representation for x_t, we can use an autoregressive network that runs on top of the encoder network to encode the historical context.

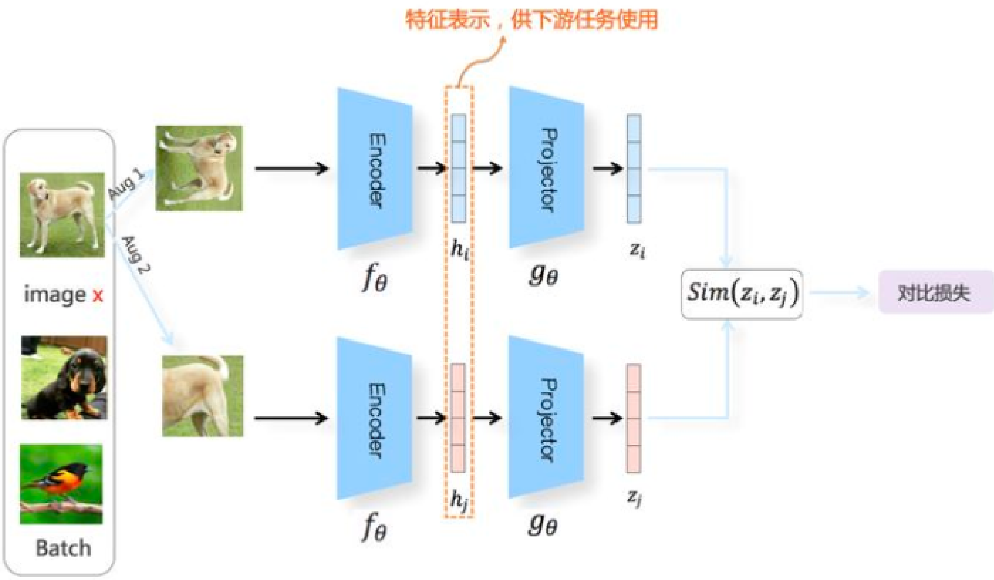


SimCLR (Hinton)

- 

正例构造方法如上图所示。对于某张图片，我们从可能的增强操作集合 T 中，随机抽取两种：$t_1 \sim T$ 及 $t_2 \sim T$，分别作用在原始图像上，形成两张经过增强的新图像 $\langle x_1, \ x_2 \rangle$，两者互为正例。训练时，Batch内任意其它图像，都可做为 x_1 或 x_2 的负例。这样，对比学习希望习得某个表示模型，它能够将图片映射到某个投影空间，并在这个空间内拉近正例的距离，推远负例距离。也就是说，迫使表示模型能够忽略表面因素，学习图像的内在一致结构信息，即学会某些类型的不变性，比如遮挡不变性、旋转不变性、颜色不变性等。SimCLR证明了，如果能够同时融合多种图像增强操作，增加对比学习模型任务难度，对于对比学习效果有明显提升作用。

有了正例和负例，接下来需要做的是：构造一个表示学习系统，通过它将训练数据投影到某个表示空间内，并采取一定的方法，使得正例距离能够比较近，负例距离比较远。在这个对比学习的指导原则下，我们来看SimCLR是如何构造表示学习系统的。

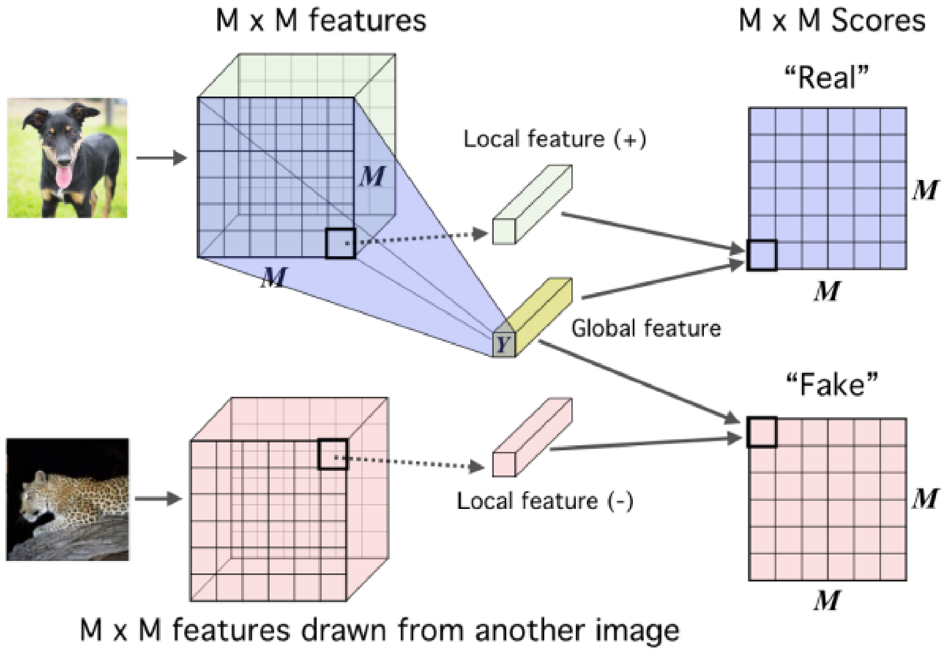
- 

- $$\ell_{i,j} = -\log \frac{\exp(\mathrm{sim}(z_i, z_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(\mathrm{sim}(z_i, z_k)/\tau)}$$

- **Algorithm 1** SimCLR's main learning algorithm.

  input: batch size N, constant τ, structure of f, g, $\mathcal{T}$.
  for sampled minibatch $\{x_k\}_{k=1}^N$ **do**
    for all $k \in \{1, \ldots, N\}$ **do**
      draw two augmentation functions $t \sim \mathcal{T}$, $t' \sim \mathcal{T}$
      # the first augmentation
      $\tilde{x}_{2k-1} = t(x_k)$
      $h_{2k-1} = f(\tilde{x}_{2k-1})$ # representation
      $z_{2k-1} = g(h_{2k-1})$ # projection
      # the second augmentation
      $\tilde{x}_{2k} = t'(x_k)$
      $h_{2k} = f(\tilde{x}_{2k})$ # representation
      $z_{2k} = g(h_{2k})$ # projection
    end for
    for all $i \in \{1, \ldots, 2N\}$ and $j \in \{1, \ldots, 2N\}$ **do**
      $s_{i,j} = z_i^\top z_j / (\|z_i\| \|z_j\|)$ # 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 $g(\cdot)$

Deep InfoMax (Bengio)

- 

- The contrastive task behind DIM is to classify whether a pair of global features and local features are from the same image or not.
    - global features are the final output of a convolutional encoder (a flat vector, Y)
    - local features are the output of an intermediate layer in the encoder (an M x M feature map)
- The loss function for DIM looks exactly as the contrastive loss function we described above. Given an anchor image x,
    - $f(x)$ refers to the global features.
    - $f(x^+)$ refers to the local features from the same image (positive samples).
    - $f(x^-)$ refers to the local features from a different image (negative samples).

Deep graph infomax

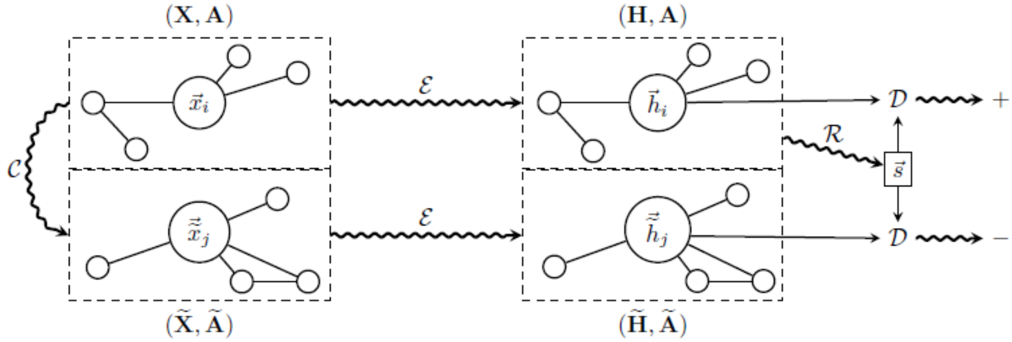
- 

Figure 1: A high-level overview of Deep Graph Infomax. Refer to Section 3.4 for more details.

- Assuming the single-graph setup (i.e., $(\mathbf{X}, \mathbf{A})$ provided as input), we will now summarize the steps of the Deep Graph Infomax procedure:

  1. Sample a negative example by using the corruption function: $(\widetilde{\mathbf{X}}, \widetilde{\mathbf{A}}) \sim \mathcal{C}(\mathbf{X}, \mathbf{A})$.
  2. Obtain patch representations, $\vec{h}_i$ for the input graph by passing it through the encoder: $\mathbf{H} = \mathcal{E}(\mathbf{X}, \mathbf{A}) = \{\vec{h}_1, \vec{h}_2, \ldots, \vec{h}_N\}$.
  3. Obtain patch representations, $\vec{\tilde{h}}_j$ for the negative example by passing it through the encoder: $\widetilde{\mathbf{H}} = \mathcal{E}(\widetilde{\mathbf{X}}, \widetilde{\mathbf{A}}) = \{\vec{\tilde{h}}_1, \vec{\tilde{h}}_2, \ldots, \vec{\tilde{h}}_M\}$.
  4. Summarize the input graph by passing its patch representations through the readout function: $\vec{s} = \mathcal{R}(\mathbf{H})$.
  5. Update parameters of $\mathcal{E}$, $\mathcal{R}$ and $\mathcal{D}$ by applying gradient descent to maximize Equation 1.

- $$\mathcal{L} = \frac{1}{N+M}\left(\sum_{i=1}^{N} \mathbb{E}_{(\mathbf{X},\mathbf{A})}\left[\log \mathcal{D}\left(\vec{h}_i, \vec{s}\right)\right] + \sum_{j=1}^{M} \mathbb{E}_{(\widetilde{\mathbf{X}},\widetilde{\mathbf{A}})}\left[\log\left(1 - \mathcal{D}\left(\vec{\tilde{h}}_j, \vec{s}\right)\right)\right]\right)$$