Subject: Stanford CS229 Machine Learning, Lecture 16, Self-supervised Learning

Date: from December 2, 2025 to January 5, 2026

Contents

Stanford CS229 Machine Learning, Self-supervised learning, 2022, Lecture 16

Link on YouTube: Stanford CS229 Machine Learning, Self-supervised learning, 2022, Lecture 16

Introduction

Introduction

本节主要讨论在标签稀缺但无标签数据极其丰富的现实条件下，如何通过自监督方式学习高质量表示，并将其迁移到下游任务中，重点包括：1. Self-supervised Learning 的基本框架，2. Adaptation 的两种典型方式，3. Contrastive Learning (对比学习), 4. 大语言模型 (LLM).

Self-supervised Learning

Basic Concept

网络架构、数据、硬件是深度学习训练的三大组成部分. 其中对于训练数据而言，在现实场景中有标签数据相对无标签数据是非常少的，为使模型"见过"更多数据，一个想法是先在大量无标签数据上进行无监督学习再针对特定任务使用有标签数据进行监督学习. 这种现代化训练方法被称为 **Self-supervised learning**.

Pre-training. 预训练(pre-train) 是指在大量的无标签数据上进行无监督学习以训练一个大型模型.

Adaptation. 将预训练模型应用于各种特定下游任务(downstream task)，通常使用监督学习进一步微调(tuning).

Note 1. *Adaptation* 也被称为 *transfer learning*. 现代的 *trasfer learning (TL)* 使用无标签数据进行预训练，但 *2000* 年左右 *TL* 还是使用监督学习进行预训练，并且上下游任务相似.

Foundation model. 也被称为 pre-trained model [1].

Pretraining

Data. $\{x^{(1)}, \dots, x^{(n)}\}$，此处为无标签数据.
Model. $\phi_\theta : x \mapsto \phi_\theta(x) \in \mathbb{R}^m$，将数据（可能是文本、图像等）映射为向量.

Note 2. ϕ_θ 可以被称为 *representation / feature / embedding*，在 *kernel method* 中被称为 *feature map*，但区别是 *kernel method* 中 ϕ 是给定的，此处的 ϕ_θ 是一个神经网络，θ 需要学习.

Loss. $L_{\text{pre}}(\theta) = \frac{1}{n}\sum_{i=1}^{n} \ell_{\text{pre}}(\theta, x^{(i)})$. 对于不同的任务 $\ell_{\text{pre}}(\theta, x^{(i)})$ 的定义也不相同.

Optimize. $\hat{\theta} = \arg\min_\theta L_{\text{pre}}(\theta)$，并将 $\hat{\theta}$ 称为 pretrained model.

Adaptation

Adaptation. 作用于下游(监督学习)任务数据集 $\{(x^{(1)}_{\text{task}}, y^{(1)}_{\text{task}}), \dots, (x^{(n_t)}_{\text{task}}, y^{(n_t)}_{\text{task}})\}$，其中 $n_t =$ # downstream task examples. 当 $n_t = 0$ 时被称为 **zero-shot learning**；当 $n_t \neq 0$ 但

仍较小(例如 10) 时被称为 **few-shot learning**.

Linear probe. Adaptation 的最简单的方式是冻结预训练模型，只在其输出表示上训练一个线性模型，即 linear probe:

$$\text{Prediction model: } \hat{y} = w^\top \phi_{\hat{\theta}}(x),\ w \in \mathbb{R}^m \Rightarrow \min_w \frac{1}{n_t} \sum_{i=1}^{n_t} \ell_{\text{task}}\left(y_{\text{task}}^{(i)},\ w^\top \phi_{\hat{\theta}}(x_{\text{task}}^{(i)})\right). \quad (1)$$

以计算机视觉(Computer Vision) 任务为例，在有标签数据集(例如 ImageNet) 上学习一个神经网络 $u^\top \phi_\theta(x)$，其中 u 为最后一层线性层，ϕ_θ 为最后一层前所有层，此时 ϕ_θ 即作为 pretrained model.

Finetune. Adaptation 的另一种方法是微调(finetune) pretrained model 的参数 $\hat{\theta}$:

$$\begin{aligned} &\text{Prediction model: } \hat{y} = w^\top \phi_\theta(x) \\ &\text{Optimize: both } w \text{ and } \theta \text{ on downstream task} \\ &\text{Initialization: } \theta \leftarrow \hat{\theta} \text{ (from pretraining)}, w \leftarrow w_0 (\text{random}) \end{aligned} \quad (2)$$

Contrastive Learning

Data Augmentation

数据增强 (data augmentation) 是通过特定方式使用原始图像生成不同"视角/视图（views）"并将其作为新的数据用于模型训练. 常见的数据增强方式包括：

1. 平移(translation)，翻转(flip)，旋转(rotation)，缩放(scale)
2. 增加噪点(add pixel noise)，改变颜色(color)、亮度(brightness)、对比度(contrast)

Contrastive Learning

数据增强在监督学习中可以增大数据集并且让模型对于这些改变具有不变性，其也可以应用于无监督学习中. 设原始样本 x 经过两次随机数据增强后得到两视图 $\hat{x}, \tilde{x}$，称 $(\hat{x}, \tilde{x})$ 为正样本 (positive pair). 另一随机样本 z 经数据增强后得到视图 $\hat{z}$，称 $(\hat{x}, \hat{z})$ 为负样本 (negative / random pair). 那么在对比学习的目标在于：

1. 让正样本对在表征空间里更接近，即 $\phi_\theta(\hat{x})$ 与$\phi_\theta(\tilde{x})$ 接近.
2. 让负样本对在表征空间里更远离，即 $\phi_\theta(\hat{x})$ 与 $\phi_\theta(\hat{z})$ 远离. (防止坍缩 (collapse)[a])

SIMCLR. Contrastive learning 的代表性开创工作是 SIMCLR [2]. 其想法是取一个 batch 的原始样本 $x^{(1)}, x^{(2)}, \ldots, x^{(B)}$，对每个样本做两次独立的数据增强得到 $\hat{x}^{(1)}, \ldots, \hat{x}^{(B)}$ 和 $\tilde{x}^{(1)}, \ldots, \tilde{x}^{(B)}$. 定义相似度分数:

$$s_{i,j} \triangleq \phi_\theta(\hat{x}^{(i)})^\top \phi_\theta(\tilde{x}^{(j)}), \quad (3)$$

则第 i 个样本的对比损失定义为

$$\ell_i = -\log \frac{\exp(s_{i,i})}{\exp(s_{i,i}) + \sum_{j \neq i} \exp(s_{i,j})}, \quad (4)$$

总损失为

$$\mathcal{L} = \sum_{i=1}^{B} \ell_i = -\sum_{i=1}^{B} \log \frac{\exp(\phi_\theta(\hat{x}^{(i)})^\top \phi_\theta(\tilde{x}^{(i)}))}{\exp(\phi_\theta(\hat{x}^{(i)})^\top \phi_\theta(\tilde{x}^{(i)})) + \sum_{j \neq i} \exp(\phi_\theta(\hat{x}^{(i)})^\top \phi_\theta(\tilde{x}^{(j)}))}. \tag{5}$$

可以看出增大正样本相似度 ($s_{i,i}$) 会让损失变小，增大负样本相似度 ($s_{i,j}, i \neq j$) 会让损失变大. 这与 constrastive learning 的两个目标相符.

Note 3. 此处的损失函数式*(5)* 并未使用标签信息，因此其为 *unsupervised learning / self-supervised learning*.

[a] 当模型坍缩至一点时 $\phi(\hat{x})$ 与 $\phi(\tilde{x})$ 距离为 0，即条件 1 满足，因此为保证模型不坍缩还需要满足第 2 点.

Large Language Model

Large Language Model

语言模型 (language model) 通常处理文本序列，记为 $(x_1, \ldots, x_T), x_i \in \{1, \ldots, V\}$，其中 T 为序列长度，$\{1, \ldots, V\}$ 为词汇表 (vocabulary).

Note 4. 语言模型处理文本的细度 *(granularity)* 并非是单词，而是 ***token***，即 x_i 并不一定是简单的一个单词，其也有可能是单词的一部分. 例如一些频繁出现的单词其本身可以作为一个 *token*，但对于如 *suboptimal* 这样的单词可能会被拆为 *sub* + *opimal* 作为两个 *token*.

Language model. 语言模型是对序列联合分布的概率模型 $p(x_1, \ldots, x_T)$，由于每个 token 有 V 种可能性，因此共有 V^T 种可能，即该分布的支持集大小为 V^T. 为解决 V^T 过大的问题，引入 链式法则分解 (chain rule):

$$p(x_1, \ldots, x_T) = p(x_1)\, p(x_2|x_1)\, p(x_3|x_1, x_2) \cdots p(x_T|x_1, \ldots, x_{T-1}) = \prod_{t=1}^{T} p(x_t|x_1, \ldots, x_{t-1}), \tag{6}$$

因此语言建模可转化为学习一系列条件概率（下一 token 分布）：

$$(x_t \mid x_1, \ldots, x_{t-1}), \quad t = 1, \ldots, T. \tag{7}$$

Embedding. 由于机器无法直接理解自然语言，因此需要将 token i 转化为向量 $e_i \in \mathbb{R}^{d_1}$，即嵌入 (embedding). 再经过黑盒模型 ϕ_θ (一般为 Transformer) 处理后得到 $c_{i+1} \in \mathbb{R}^{d_2}$ (如图1). 在使用 c_t 预测 $p(x_t \mid x_1, \ldots, x_{t-1})$ 时可以将其建模为多分类任务:

$$\begin{bmatrix} p(x_t = 1 \mid x_1, \ldots, x_{t-1}) \\ \vdots \\ p(x_t = V \mid x_1, \ldots, x_{t-1}) \end{bmatrix} = \mathrm{softmax}(z_t) = \mathrm{softmax}(W_t c_t) \in \mathbb{R}^V \tag{8}$$

其中 $z_t = W_t\, c_t$ 被称为 logits，$W_t \in \mathbb{R}^{V \times d_2}$，$\mathrm{softmax}(u) = \left[\frac{\exp(u_1)}{\sum_{i=1}^{V} \exp(u_i)}, \ldots, \frac{\exp(u_V)}{\sum_{i=1}^{V} \exp(u_i)}\right]^\top$.

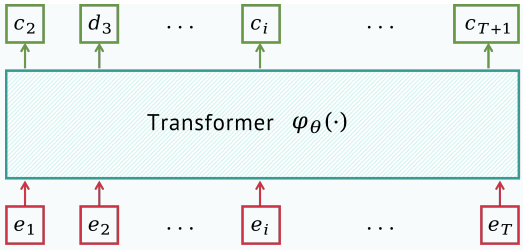

Figure 1: Large language model embedding.

Loss. 定义 $p_t = \text{softmax}(W_t\, c_t)$，则则整体损失是对每个位置交叉熵损失的求和：

$$\text{loss}(W, \theta, e) = \sum_{t=1}^{T} (\text{cross-entropy loss at position } t) = \sum_{t=1}^{T} -\log p_t(x_t). \tag{9}$$

Zero-shot. 零样本学习充分利用 LLM 的 generalization 的能力，其将任意任务转换为问题，在让模型得到一个回答后通过模型的推理能力不断生成后续的 token，从而完成整个 sequence 的生成.

In-context learning. 在上下文学习中，任务的例子直接通过“拼接”的方式，加入到一个大文档中（即 prompt）。模型通过理解这些例子，学习任务的规律，并且基于此做出预测. 例如给定以下几个问题和答案，我们将它们拼接成一个prompt：

Q: $2 \sim 3 =?$ A: 5

Q: $6 \sim 7 =?$ A: 13

Q: $15 \sim 2 =?$ A: ___ （模型预测）

在这种情况下，模型通过已经提供的少量样本 (“$2 \sim 3 = 5$” 和“$6 \sim 7 = 13$”)，来推测和回答新的问题$15 \sim 2 =?$.

Last update: January 5, 2026

References

[1] Rishi Bommasani. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

[2] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.