CS 839: Theoretical Foundations of Deep Learning **Spring 2022**

Lecture 2 Theory of Representation Learning I

Instructor: Yingyu Liang *Date: March 29^{th}, 2022* *Scriber: Nicholas Roberts*

1 Background on Representation Learning

Early deep learning researchers faced a number of significant challenges when it came to finding the right tricks for getting their systems to work: a shortage of a lack of heuristics for designing deep learning algorithms, a lack of clarity with regard to the "correct" way to train their models, and finally, a lack of labeled data (though, as usual, they had plenty of unlabeled data). To make progress on some of these problems, early researchers proposed *unsupervised representation learning* as a means to make the most out of their data by warm-starting their training procedures with unlabeled data before training on what little labeled data they had. Researchers found that these techniques resulted in more performant models than those resulting from training on labeled data alone.

Nowadays, representation learning is still commonly used as a pre-training step. In many cases, however, this pre-training step is done at a large scale by organizations who can afford significant investments in compute (e.g. Google), and these pre-trained models are (sometimes) made publicly available to researchers and practitioners to use for their own downstream tasks. Domains which have enjoyed the most benefit from modern representation learning are computer vision, natural language processing, and even more recently, combinations of data modalities from both of these fields – leading to state-of-the-art results on downstream tasks in all of these domains.

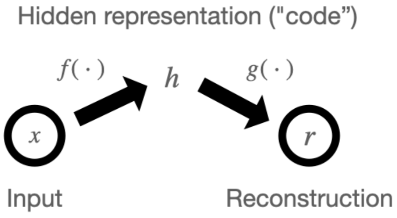
So what exactly is representation learning and what are its goals? Generally, the goal of representation learning is to learn a *representation function* via unsupervised learning which can be used in downstream supervised learning tasks. Surprisingly, the unlabeled data that was used to train the representation function need not be from the exact same distribution that the downstream model is trained on (though it should probably be at least vaguely related). For example, you might have a large corpus of unlabeled text from Wikipedia that you would use to train your representation function, and your downstream task is to classify movie reviews by their star ratings – these are different distributions, but the representation function ideally will have captured something fundamental about language that can be useful for movie review classification. Even more surprisingly, a downstream model using high-quality representations doesn't need to be complicated either – often, linear models trained on high-quality representations of their data are competitive with state-of-the-art deep learning methods.

In fact, a good representation is one with discriminative power – i.e., one that can be used to train a downstream linear classifier on labeled data. Assuming your downstream task is k-class classification, in the overly-optimistic ideal case, a perfect representation function would be able to map all of the examples of each of the k classes to one of k vectors. That is, the ideal scenario would be that we could guarantee that our unsupervised representations would be linearly separable for a downstream classifier. This seems hard. What have people done to move toward this idealized scenario? A recent method in computer vision is to apply various forms of data augmentation to images, and train the representation function to have the same representation regardless of the augmentation strategy applied to it. Though simple, this technique results in such high-quality representations that a downstream linear classifier trained on ImageNet can compete with state-of-the-art deep learning methods – this is a technique called “contrastive learning.”

2 Examples of Representation Learning

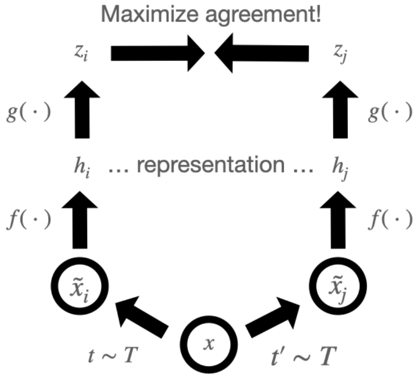
In this section, we will discuss a few popular examples of representation learning.

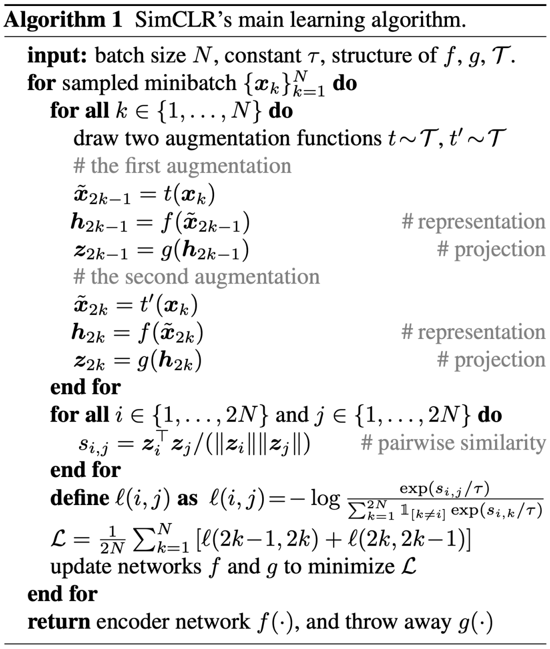
Autoencoders One of the simplest forms of representation learning are *autoencoders* [1]. Autoencoders map the input to some hidden representation and from the hidden represen-taion, attempt to reconstruct the input. More precisely, neural network-based autoencoders comprise two components: an encoder network $h = f(x)$, which acts as our representation function, and a decoder network $r = g(h)$. Basic autoencoders are trained to minimize a loss function of the form: $L(x, r) = L(x, g(f(x)))$. Typically, the vector h is of lower dimension than the input, so as to avoid learning trivial representation functions such as the identity – this is called under-completeness. Other constraints, including the sparsity of h, can be included in the loss function. When f, g are linear functions and when L is the $L2$-loss, the resulting autoencoder is essentially PCA. On the other hand, when f, g are non-linear functions, the result is considered to be a form of non-linear dimensionality reduction.



Contrastive Learning Another example of representation learning is *contrastive learning*, and in particular, we will discuss a recent exemplary contrastive learning technique called SimCLR [2]. This technique is motivated by the fact that for images, data augmentation typically does not change the class label. The goal is to move representations of different data augmentation transformations, denoted $t, t' \sim T$, of the same image closer together using a “contrastive” loss. In practice, the output layer itself is not used as part of the final representation function, and a previous layer is used (in other words, there is a “projection head” that gets discarded after pre-training). The training routine and contrastive loss function are given in the figure below.

Masked Self-Supervised Learning A third example of representation learning are *masked self-supervised learning* techniques, an example of which is BERT [3]. This is a

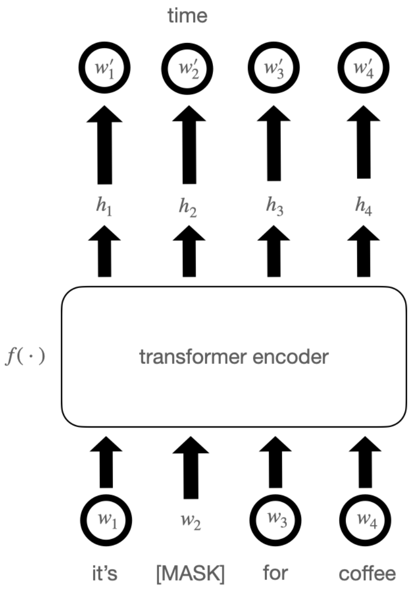




Algorithm 1 SimCLR's main learning algorithm.

```
input: batch size $N$, constant $\tau$, structure of $f$, $g$, $\mathcal{T}$.
for sampled minibatch $\{\boldsymbol{x}_k\}_{k=1}^N$ do
    for all $k \in \{1, \ldots, N\}$ do
        draw two augmentation functions $t \sim \mathcal{T}$, $t' \sim \mathcal{T}$
        # the first augmentation
        $\tilde{\boldsymbol{x}}_{2k-1} = t(\boldsymbol{x}_k)$
        $\boldsymbol{h}_{2k-1} = f(\tilde{\boldsymbol{x}}_{2k-1})$                # representation
        $\boldsymbol{z}_{2k-1} = g(\boldsymbol{h}_{2k-1})$                    # projection
        # the second augmentation
        $\tilde{\boldsymbol{x}}_{2k} = t'(\boldsymbol{x}_k)$
        $\boldsymbol{h}_{2k} = f(\tilde{\boldsymbol{x}}_{2k})$                    # representation
        $\boldsymbol{z}_{2k} = g(\boldsymbol{h}_{2k})$                        # projection
    end for
    for all $i \in \{1, \ldots, 2N\}$ and $j \in \{1, \ldots, 2N\}$ do
        $s_{i,j} = \boldsymbol{z}_i^\top \boldsymbol{z}_j / (\|\boldsymbol{z}_i\| \|\boldsymbol{z}_j\|)$      # 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 $g(\cdot)$
```

family of techniques which are typically applied to text, and involve learning a representation function for words given the context (i.e., the surrounding words). This is usually done by training a language model to fill-in missing words in a sentence. In language modeling problems such as this, the output is typically a softmax over the entire vocabulary. The resulting representation function is the encoder portion of the network shown below, which is usually a transformer-based architecture.



3 Theoretical Framework for Representation Learning

While these representation learning techniques may seem complex and diverse, it is indeed possible to study them theoretically using the same set of tools. In particular, [4] proposes a unifying theoretical framework for analyzing the sample complexity of each of these methods when viewed as regularization applied to the representation, in the form of a learnable function using unlabeled data. In this setting, our focus will be on sample complexity using uniform convergence bounds, and we will ignore problems related to optimization.

Preliminaries Denote our set of unlabeled examples as $U = \{\widetilde{x}_i\}_{i=1}^{m_u}$ where $\widetilde{x}_i \sim \mathcal{U}_x$ and denote our set of labeled examples as $S = \{(x_i, y_i)\}_{i=1}^{m_l}$ with $(x_i, y_i) \sim \mathcal{D}$ and marginal $\mathcal{D}_x$. Note that $\mathcal{U}_x$ and $\mathcal{D}_x$ are not required to be the same distribution.

In all of the above examples of representation learning algorithms, the representation learning procedure proceeds as follows:

1. Learn a representation function $\phi(\cdot)$ by minimizing some loss L_r using U.
2. Learn a downstream model $\widehat{y} = f(\phi(x))$ by minimizing some other loss $L_c(\widehat{y}, y) \in [0, 1]$ using S.

Where the models for both steps are chosen from their respective hypothesis classes, denoted as $\phi \in \Phi$ and $f \in \mathcal{F}$.

Autoencoders in the representation learning framework Concretely, let's take the example of learning a downstream classifier on an autoencoder... Consider functions from the following hypothesis classes representing the encoder, decoder, and the downstream classifier, respectively:

$$h \in \mathcal{H}, g \in \mathcal{G}, f \in \mathcal{F}.$$

In this case, we have the following procedure:

1. Learn $h(\cdot)$ by minimizing $L_r = \widehat{\mathbb{E}}\|X - r\|_2^2 = \widehat{\mathbb{E}}\|X - g(h(x))\|_2^2$ using U.
2. Learn a downstream classifier $\widehat{y} = f(h(x))$ by minimizing $\widehat{\mathbb{E}}[L_c(f(h(x)), y)]$ using S.

In the first step, we're doing some loss minimization – crucially, note that this is not a classification loss and it furthermore doesn't involve labels! Let's call this the *regularization loss.* Denote the regularization loss for autoencoders over the unlabeled data distribution as $L_r(h, g; \mathcal{U}_x) = \mathbb{E}_{x \sim \mathcal{U}_x}[L_r(h, g; x)] \in [0, 1]$. Furthermore, denote $L_r(h; \mathcal{U}_x) = \min_g L_r(h, g; \mathcal{U}_x)$. After minimizing this loss, g is discarded and we use $\phi(\cdot) = h$.

Combining all of these, the representation learning problem is the following optimization:

$$\min_{f \in \mathcal{F}, h \in \mathcal{H}} L_c(f \circ h; S) \quad \text{s.t.} \quad L_r(h; \mathcal{U}_x) \leq \tau.$$

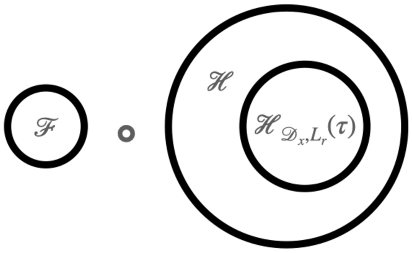
To see how this results in a regularized subset of the end-to-end fully-supervised hypothesis class $\mathcal{F} \circ \mathcal{H}$, assume $\mathcal{U}_x = \mathcal{D}_x$. Then, we can write the subset of representation functions feasible in the above optimization problem as

$$\mathcal{H}_{\mathcal{D}_x, L_r}(\tau) = \{h \in \mathcal{H} : L_r(h, \mathcal{D}_x) \leq \tau\} \subseteq \mathcal{H}.$$

Then our regularized class of functions in this representation learning framework is

$$\mathcal{F} \circ \mathcal{H}_{\mathcal{D}_x, L_r}(\tau) \subseteq \mathcal{F} \circ \mathcal{H},$$

where $\mathcal{F} \circ \mathcal{H}$ is the class of functions considered in end-to-end fully-supervised learning. This regularized subset excludes "bad" representation functions in $\mathcal{H}$. This is depicted diagrammatically below.



References

[1] Dana H. Ballard. Modular learning in neural networks. In *Proceedings of the Sixth National Conference on Artificial Intelligence - Volume 1*, AAAI'87, page 279–284. AAAI Press, 1987.

[2] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 1597–1607. PMLR, 13–18 Jul 2020.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[4] Siddhant Garg and Yingyu Liang. Functional regularization for representation learning: A unified theoretical perspective. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 17187–17199. Curran Associates, Inc., 2020.