FLT Seminar Series[1], Session 2

How Feature Learning Theory Works?

Chen Yanbo[2]

Ph.D. Candidate
School of Computer Science, Wuhan University
430072, Wuhan, Hubei, China

Jun. 1st, 2025

[1]This project is open for collaboration. For details, see our project page at https://github.com/yanboc/feature-learning-theory.

[2]Contact: yanboch@126.com.

Outline

An Quick Start to FLT

- Highlights from our last session: **what** is feature learning theory?
  - Terminologies: what are feature and learning, respectively?
  - A bird's-eye view summary of FLT
- A simplified example: **how** FLT works?
  - The **theoretical framework** of FLT
  - The **theoretical goal** of FLT

Table of Contents

What is *feature learning theory (FLT)*?

What are *feature* and *learning*, respectively?

Terminologies

- We focus on **features** in DL (w.r.t. data, NNs, and specific tasks); the main goal of DL is to *find NNs that extract useful features from data.*

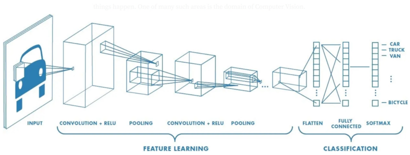

Figure: Higher- and lower-level features in CNN-based classification

What is *feature learning theory (FLT)*?

What are *feature* and *learning*, respectively?

Terminologies

- Machine **learning** uses a specified *algorithm* to find the best model in the *hypothesis class* (e.g., NNs) according to the performance of the model on the *data*, concerning the *evaluation* standard.

Table: The *four core elements* of the ML&DL paradigm

	Theoretical	**In Practice**
Data	vectors and matrices	tensor
Hypothesis Class	functions and mappings	multi-layer NN
Algorithm	optimization	optimizer, LR, ...
Evaluation	loss function, regularization	CE, MSE, ...

A bird's-eye view summary of FLT

Machine learning uses a specified *algorithm* to find the best model in the *hypothesis class* (e.g., NNs) according to the performance of the model on the *data*, concerning the *evaluation* standard.

$\Downarrow$

FLT *specifies* the learning task (*network structure*, *data assumption*, *loss*, and *algorithm*) and explore the ***dynamics*** of training.

$\Downarrow$

Dynamics: how the *parameters of the NN* iterate from random initialization (noise) to *useful features* capable of accurate classification/regression?

Table of Contents

How FLT works? (1/2)

Step 1. FLT *specifies* the learning task

(*network structure*, *data assumption*, *loss*, and *algorithm*)

Theoretical framework [Allen-Zhu & Li, 2005.10190v4]

- **Hypothesis Class**: 2-layer (symmetric)-ReLU network $f(x; w)$
- **Data**: orthogonal feature + sparse coding model

$$x = \boldsymbol{M}z + \xi, \;\; y = sign(\langle w^*, z\rangle)$$

- **Algorithm**: GD with random initialization

$$w^{(t+1)} = w - \eta \nabla Loss(f(x, w), y)$$

- **Evaluation**: logistic loss for classification.

Intuition: specifying the learning task is like creating a **virtual environment** to *play around* with.

The triviality & tractability trade-off

Specifying the learning task is a **tricky job**

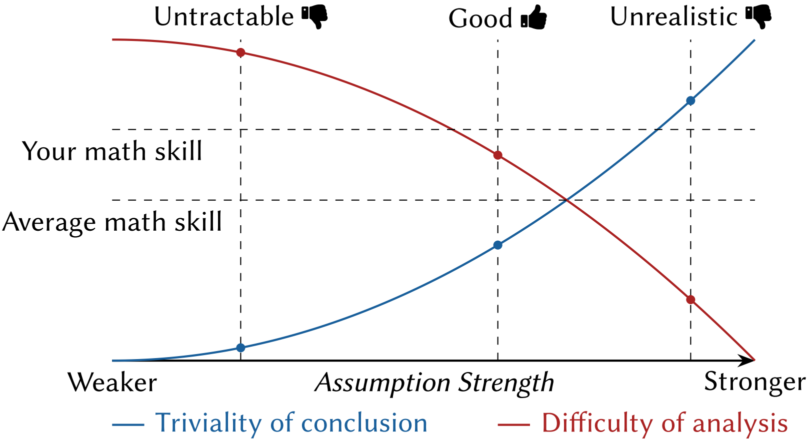

Figure: There is a trade-off between triviality and tractability.

Data Distribution

We consider a supervised binary classification task.

Sparse Coding Model, SCM (1/2)

Consider the training data $x \in \mathbb{R}^d$ generated from $x = \boldsymbol{M}z + \xi$ for *coding matrix* $\boldsymbol{M} \in \mathbb{R}^{d \times d}$, *hidden vector* $z \in \mathbb{R}^d$, and *noise vector* $\xi \in \mathbb{R}^d$ such that

- $\boldsymbol{M} = (\boldsymbol{M}_1, \boldsymbol{M}_2, \cdots, \boldsymbol{M}_d)$ is a unitary (i.e., orthonormal) matrix.
- z is a sparse vector in the sense that $\|z\|_0 = \Theta\left(\sqrt{d}\right)$.
- for simplicity, we let $\xi \sim \mathcal{N}(0, \sigma_0 \boldsymbol{I})$ in this seminar

Remarks:

- The *sparsity* of z and the *magnitude* of ξ are *non-negligible* to guarantee the expressiveness of SCM.
- More choices of (implicit) data distributions (delayed to FLT-3).

Data Distribution

Sparse Coding Model, SCM (2/2)

The label of x is decided by the hidden vector z and a *labeling function* $w^\star$.

$$y = \mathrm{sign}(\langle w^\star, z\rangle). \tag{1}$$

For simplicity, assume that $|w_i^\star| = \Theta(1)$ for all $i \in [d]$ (i.e., *balanced* setting).

learning goal: predict the label of x

$\Downarrow$

We need to find f such that $f(\boldsymbol{M}z + \xi) \approx \mathrm{sign}(\langle w^\star, z\rangle)$.

The Philosophy of FLT

Philosophy No.1., **Symmetry**

In the specified learning tasks, *the data, networks, and algorithms must embody certain symmetries or self-similarities.* For instance,

- the coding matrix $\boldsymbol{M}$ is orthonormal,
- the labeling function $w^\star$ is balanced,
- the self-similarity within the GD algorithm.

In FLT, we only need to analyze a single part of the symmetric system, rather than all the parts.

Hypothesis Class (i.e., Network Structure)

learning goal: predict the label of x

$\Downarrow$

We need to find f such that $f(\boldsymbol{M}z + \xi) \approx \operatorname{sign}(\langle w^\star, z\rangle)$.

Natural intuition

A natural intuition is letting

$$f(x) = \langle w^\star, \boldsymbol{M}^\top x\rangle = \langle w^\star, z\rangle + \langle \boldsymbol{M}w^\star, \xi\rangle. \tag{2}$$

The non-negligible magnitude of ξ would lead to inaccuracy.

$\Downarrow$

Linear models are *not complicated (expressive) enough* to characterize SCM.

(cf. the expressiveness of NN, VC-dimension theory, the overfitting phenomenon, regularization & the Occam's Razor principle.)

Hypothesis Class (i.e., Network Structure)

Two-layer (symmetric) ReLU network

We consider the following network

$$f_t(x; \boldsymbol{w}^{(t)}) = \sum_{i=1}^{m} \left(\mathrm{ReLU}\left(\langle w_i^{(t)}, x \rangle - b_i^{(t)} \right) - \mathrm{ReLU}\left(-\langle w_i^{(t)}, x \rangle - b_i^{(t)} \right) \right)$$

(optimized to) $f(x) \approx \sum_{i=1}^{n} w_i^\star \left(\mathrm{ReLU}\left(\langle \boldsymbol{M}_i, x \rangle - b_i \right) - \mathrm{ReLU}\left(-\langle \boldsymbol{M}_i, x \rangle - b_i \right) \right)$

parameterized by $\boldsymbol{w}^{(t)} := \left(w_{[m]}^{(t)}, b_{[m]}^{(t)} \right)$, where m denotes the width of f_t.

Remarks:

- The ReLU activation is smoothified (using a mollifier), omitted here.
- **How to obtain Eq. (14)?** It can neither be more complicated nor simpler (cf. Figure 2). Pure tricks or intuition, maybe.
- Over-parameterization & Thresholding.

The Philosophy of FLT

- Philosophy No.1., Symmetry.

Philosophy No.2., **Programmatic Thinking**

When performing feature learning analysis, *one should think and act like a programmer, rather than a mathematician.* For instance,

- **Programmatic definitions**. Find intuitions and definitions from practical code and PyTorch documentation!
- **Programmatic tuning**. There are many parameters in the analysis, e.g., m and σ_0, that require careful tuning.
- **Programmatic workflow**. FLT undergoes the entire training process, starting with random initialization and stopping by the attainment of an accurate classifier.

In FLT, we only commit the *minimum necessary changes* to a practical training process of NNs.

Loss function and empirical risk

We consider the standard *logistic loss*

$$\mathrm{Loss}_t\left(\boldsymbol{w}^{(t)}; x, y\right) := \log\left(1 + \exp\left(-y f_t\left(x; \boldsymbol{w}^{(t)}\right)\right)\right) \tag{3}$$

and the corresponding *empirical risk*

$$\mathrm{Risk}_t\left(\boldsymbol{w}^{(t)}\right) := \frac{1}{N}\sum_{j\in[N]}\left(\mathrm{Loss}_t\left(\boldsymbol{w}^{(t)}; x_j, y_j\right)\right) \tag{4}$$

The **training goal** is to find the best $\boldsymbol{w}$ that minimizes the empirical risk (i.e., the ERM training paradigm).

Remark:

- FLT for other training paradigms (e.g., Bayesian NN, GANs, RL, Causal Inference). (Good choices for future research!)

Algorithm

Gradient descent with random initialization

For each $i \in [m]$, the update rule of $\boldsymbol{w}_i^{(t)}$ is

$$\boldsymbol{w}_i^{(t+1)} \leftarrow \boldsymbol{w}_i^{(t)} \eta \nabla_{\boldsymbol{w}_i^{(t)}} \mathsf{Risk}(\boldsymbol{w}_i^{(t)}), \tag{5}$$

for any $t \in [T]$, where $\boldsymbol{w}$ is initialized as

$$\boldsymbol{w}_i^{(0)} \sim \mathcal{N}(0, \sigma_1^2 \boldsymbol{I}). \tag{6}$$

Remark:

- About the parameters: recall the *programmatic thinking* philosophy.
- Some neurons **have already been good enough** at initialization (cf. *concentration inequalities* & the *lottery ticket hypothesis*).

Summary

- The first step of FLT is to *specify* the learning task, including network structure, data assumption, loss, and algorithm (check it!).
- Specifying the learning task is like creating a **virtual environment** to *play around* with.
  - What is *playing around*? Acts like tuning parameters, network structures, and data assumptions.
  - How to advance an FLT proof? Just play around and observe the changes in the proofs. The *difficulty curve* of FLT is almost linear.
- The Philosophy of FLT No. 1&2
  - Design a symmetric system to reduce the complexity of analysis.
  - Think and act like a programmer, rather than a mathematician.

How FLT works? (2/2)

Step 2. FLT defines multiple **good property sets** and studies how the neurons **enter or exit** these sets. (**Dynamics!**)

What *defines* a good feature?

Recall the network structure

$$f_t(x; \boldsymbol{w}^{(t)}) = \sum_{i=1}^{m} \left(\mathrm{ReLU}\left(\langle w_i^{(t)}, x \rangle - b_i^{(t)} \right) - \mathrm{ReLU}\left(-\langle w_i^{(t)}, x \rangle - b_i^{(t)} \right) \right)$$

(optimized to) $f(x) \approx \sum_{i=1}^{n} w_i^\star \left(\mathrm{ReLU}\left(\langle \boldsymbol{M}_i, x \rangle - b_i \right) - \mathrm{ReLU}\left(-\langle \boldsymbol{M}_i, x \rangle - b_i \right) \right)$

- One good feature $\boldsymbol{w}_j^{(t)}$ should approximate the *direction* of $\boldsymbol{M}_i$.
- Multiple good features should approximate the *magnitude* of $\boldsymbol{w}_i^\star$.

Good Property Sets

FLT defines multiple levels of good property sets. We consider two of them.

Surely Good Neurons $\mathcal{S}^t_{j,sure}$ and Potentially Good Neurons $\mathcal{S}^t_{j,pot}$

Let $\mathcal{S}^t_{j,sure} \subseteq [m]$ be those neurons $i \in [m]$ satisfying

- $\langle \boldsymbol{w}_i^{(t)}, \boldsymbol{M}_j \rangle^2 \geq (c_1 + c_2)(\sigma_3^{(t)})^2 \log d,$
- $\langle \boldsymbol{w}_i^{(t)}, \boldsymbol{M}_{j'} \rangle^2 < (c_1 - c_2)(\sigma_3^{(t)})^2 \log d,$ for every $j' \neq j$,
- $\langle \boldsymbol{w}_i^{(t)}, \boldsymbol{M}_j \rangle \boldsymbol{w}_j^\star > 0.$

Let $\mathcal{S}^t_{j,pot} \subseteq [m]$ be those neurons $i \in [m]$ satisfying

- $\langle \boldsymbol{w}_i^{(t)}, \boldsymbol{M}_j \rangle^2 \geq (c_1 - c_2)(\sigma_3^{(t)})^2 \log d.$

Remarks:

- For the flexibility of the theory, more parameters are introduced.
- **Feature Learning**: The neurons $\{\boldsymbol{w}_i\}_{j \in \mathcal{S}_j}$ approximate the direction and magnitude of $\boldsymbol{M}_j$ and $\boldsymbol{w}_j^\star$.

How the neurons *enter and exit* these sets?

Theoretical Goals

The *desired principles* of neurons' entering and exiting good property sets can be summarized as follows.

Entering:

- Some of the neurons have already been in these sets at initialization.
- Neurons from lower-level sets enter higher-level sets with probability.

Exiting:

- Neurons exit lower-level sets and enter higher-level sets.
- Neurons never exit the highest-level sets.

The **main goal** of FLT analyses are to prove the above principles.

The proof techniques are postponed to FLT-3.

Thanks for your participation!

Welcome to join our WeChat group!
If this expires, please don't hesitate to contact me at yanboch@126.com.