FLT Seminar Series[1], Session 3

Feature Learning Theory for Sequential Data

Chen Yanbo[2]

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

Jun. 15st, 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

FLT for Seq2Seq Tasks

- Highlights from our last sessions
  - **what** is feature learning theory? (Session 1)
  - A simplified example regarding a binary classification task (Session 2)
- Another simplified example: **how** FLT analyzes sequential tasks?
  - How to characterize sequential tasks.
  - A simplified FLT analysis setting for sequential tasks.
  - The proofs and techniques. (Delayed to Session 4).

Table of Contents

1 Highlights from our last sessions

2 How FLT analyzes Sequential Tasks?

What is feature learning theory? (Session 1)

What is “feature”?

- Features are extracted from raw data and are used for specific tasks.
  - higher-level feature ≈ data *representation* (used for classification)
  - lower-level feature ≈ data *pattern* (e.g., edges and shapes)

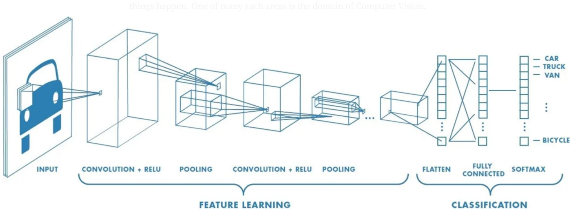

Figure: Higher- and lower-level features in CNN-based image classification

What is feature learning theory? (Sessions 1 & 2)

How FLT characterizes the features?

- FLT analyzes **features** w.r.t. *specified data*, *NN*, and *task*, e.g.,
  - Sparse coding data model: $x = \boldsymbol{M}z + \xi,\ y = sign(\langle w^*, z\rangle)$.
  - Supervised binary classification $\longrightarrow$ find the best NN parameter that minimizes the empirical risk.
- **Learning**: find NNs that extract useful features for classification.
  - $w_i^{(t)}$ is randomly initialized and updated by GD.

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

A bird's-eye view summary of FLT (Sessions 1 & 2)

4 core elements: ML uses a specified *algorithm* to find the best model in a *parameterized hypothesis class* according to the performance of the model on the *data*, concerning the *evaluation* standard.

$\Downarrow$

FLT *specifies* the learning task and explore the **dynamics** of training.

$\Downarrow$

Dynamics: how the *parameters of the NN* iterate from random initialization (noise) to *useful features* w.r.t. a specific task?

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, ...

Takeaways

The Philosophy of FLT

- **Specify everything**
  - Create a "virtual environment" to play around with
  - There is a trade-off between triviality & tractability
- **Seek symmetry**
  - The symmetric structures (e.g., self-similarities of GD steps and the orthonormal assumption of $\boldsymbol{M}$) in FLT frameworks make it sufficient to analyze a single part of a symmetric system rather than all the parts.
- **Programmatic thinking**
  - Find definitions and simplify them according to real-world practice
  - There are many parameters in the analysis that require careful tuning

The triviality & tractability trade-off

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

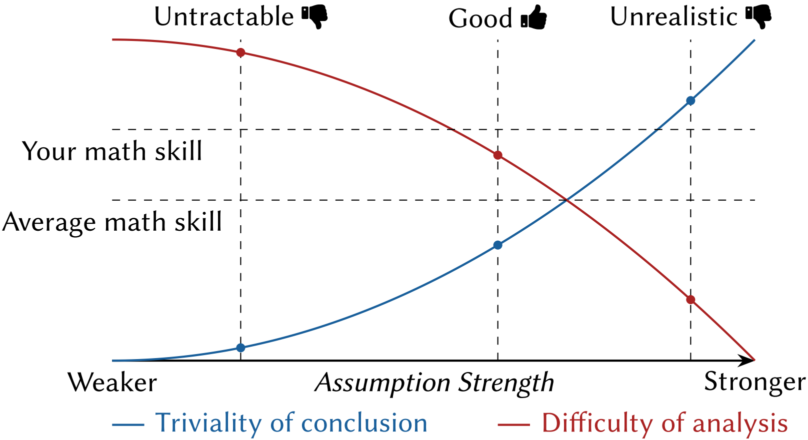

Figure: There is a trade-off between triviality and tractability.

Table of Contents

1 Highlights from our last sessions

2 How FLT analyzes Sequential Tasks?

A brief introduction to sequential tasks

There are two major applications of deep learning, i.e.,

- Computer vision (CV): image or video data (characterized as matrices).
- Natural language processing (NLP): corpus of token sequences.

The sequential tasks (4 core elements) in NLP

- *Data.* Token sequences $(v_1, v_2, \cdots, v_t)$, where the tokens i $(i \in [t])$ are chosen from an (abstract) vocabulary set $\mathcal{V}$.
- *Hypothesis class.* RNN-like, transformer-like, and others (e.g., mamba)
- *Algorithm.* similar to those in CV.
- *Evaluation.* It is task-specific, e.g.,
  - Sentiment Analysis: similar to image classification.
  - Machine Translation: similar to image reconstruction.
  - Question Answering (QA): can collapse to classification.

The Philosophy of FLT

Philosophy No.4., There is no free lunch in feature learning

There are several no-free-lunch (NFL) theorems in machine learning, e.g.,

- "*... if an algorithm performs well on a certain class of problems, then it necessarily pays for that with degraded performance on the set of all remaining problems.*" (cited from NFL's wiki)

Similarly, we can state the (informal) NFL theorem for FLT as follows.

- *If a type of network structure (e.g., CNN) performs well on a certain class of data (e.g., image data), then it necessarily pays for that with degraded performance on other classes of data (e.g., sequential data).*

In contemporary FLT research, the NN-data pairs are almost fixed, e.g.,

- Convolution $\Longleftrightarrow$ Image (Patch-like data)
- Attention $\Longleftrightarrow$ Sequential data
- Linear-like NN $\Longleftrightarrow$ Linear-like data

Transformers

The architecture of transformers (cf. Transformer Explainer)

- The input token sequence $(v_1, v_2, \cdots, v_t) \in \mathcal{V}^t$ is often the output of a tokenizer, each of which is represented by a unique index $\leq |\mathcal{V}|$.
- Given $i \in [t]$, the token v_i is embedded into a high-dimensional space $x := \mathrm{emb}(v_i) \in \mathbb{R}^d$. Most theoretical analyses start from here!
- The embedded sequence $(x_1, x_2, \cdots, x_t)$ is processed through multiple transformer blocks. Each of these blocks includes
  - Multiple attention modules (i.e., the QKV calculation), if MHA,
  - Attention mask, if causal/auto-regressive,
  - Output projection, if MHA,
  - Other regularization (e.g., layer norm, residual, drop-out, ...).
- The output of the final transformer layer is further processed through an MLP and an LM head module to obtain the next token x_{t+1}.

Transformers, but in theoretical analysis

The transformers can be simplified as follows.

The simplified transformers

- Directly consider $x \in \mathbb{R}^d$ as raw input. (omit the embeddings)
- Only one simplified transformer block with
  - Only one attention head,
  - Attention mask, if causal/auto-regressive (basically unchanged)
  - Output projection, MLP, and LM head are integrated together
  - Omit other regularization.
- The next-token generation task is often characterized as classification, or even binary classification.

The simplified transformer refers to single-layer (non-)linear transformers with only one attention head

Hypothesis Class (i.e., Network Structure)

The attention module [Sakamoto & Sato, ICML 2025]

Given input $X = (x_1, x_2, \cdots, x_t)^T \in \mathbb{R}^{t\times d}$, a single-head *self attention layer* $f_{SA} : \mathbb{R}^{t\times d} \longrightarrow \mathbb{R}^{t\times m}$ is given by

$$f_{SA}(X) = \mathrm{softmax}\left(XW_Q(XW_K)^\top\right) XW_vW_o \tag{1}$$

Denote $W = W_Q W_K^\top$ and $v = W_V W_o \in \mathbb{R}^{d\times m}$, then

$$f_{SA}(X) = \mathrm{softmax}\left(XWX^\top\right) Xv \tag{2}$$

In a binary classification task, let $m = 1$ and consider the query sequence as a single [CLS] token p. The final model is

$$\text{(final model) } f(X) = \mathrm{softmax}\left(p^\top WX^\top\right) Xv = v^\top X^\top \mathrm{softmax}(XW^\top p) \tag{3}$$

- Learnable parameters: W and p. The linear classifier v is set fixed.

Data Distribution

Gaussian mixture distribution with positional label noise

Let $\mu_{+1}, \mu_{-1} \in \mathbb{R}^d$ be fixed *class signal vectors* such that $\|\mu_{+1}\|_2 = \|\mu_{-1}\|_2 = \mu$ and $\langle \mu_{-1}, \mu_{+1} \rangle = 0$. For each token x_i, the clean label $Y_i^\star$ of x_i is sampled uniformly from $\{\pm 1\}$. The label noise vectors ϵ_i sampled from $N(0, \sigma_\epsilon^2 I_d)$.

Moreover, the tokens are split into three groups:

- Relevant token. $x_1 = \mu_{Y^\star} + \epsilon_1$
- Weakly relevant tokens. $x_i = \rho\mu_{Y^\star} + \epsilon_i$ with $\rho \ll 1$.
- Irrelevant tokens. $x_i = \epsilon_i$.

Remark:

- Compare Gaussian mixture with sparse coding (in Session 2).
- The expressiveness is controlled by the signal-noise ratio $\mu/(\sigma_\epsilon\sqrt{d})$.

Evaluation

Loss function and empirical risk

We consider the standard *logistic loss* (again, in a broadcasted sense)

$$\mathrm{Loss}\,(W, p) := \log\left(1 + \exp\left(-Y_j f(X_j)\right)\right) \tag{4}$$

for fixed (X_j, Y_j), and the corresponding *empirical risk*

$$\mathrm{Risk}_\tau\left(W^{(\tau)}, p^{(\tau)}\right) := \frac{1}{N}\sum_{j\in[N]}\left(\mathrm{Loss}_\tau\left(W^{(\tau)}, p^{(\tau)}\right)\right) \tag{5}$$

The **training goal** is to find the best W and p that minimizes the empirical risk (i.e., the ERM training paradigm).

Algorithm

Gradient descent with random initialization

For each $i \in [m]$, the update rules of W^{τ} and ∇_p are

$$\begin{aligned} W^{(\tau+1)} &\leftarrow W^{\tau} - \alpha \nabla_W \operatorname{Risk}_\tau \left(W^{(\tau)}, p^{(\tau)} \right), \\ p^{(\tau+1)} &\leftarrow p^{\tau} - \alpha \nabla_p \operatorname{Risk}_\tau \left(W^{(\tau)}, p^{(\tau)} \right), \end{aligned} \tag{6}$$

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

$$\boldsymbol{w}_i^{(0)} \sim \mathcal{N}(0, \sigma_1^2 \boldsymbol{I}). \tag{7}$$

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*).

Compared with Session 2 [Allen-Zhu & Li, 2005.10190v4]

- The first step of FLT is to *specify* the learning task, including network structure, data assumption, loss, and algorithm. ()
- The loss and algorithm are almost unchanged. (Recall the NFL theorem in FLT)
- Data model: from sparse coding model to Gaussian mixture with positional label noise
- Network structure: from two-layer ReLU network to one-layer single-head attention model

Thanks for your participation!

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