Architectures

I. Architectures as priors on function space

Change of paradigm
- classical ML: design features beforehand
  vs. deep ML: meta-design of features \(\rightarrow\) design architectures able to produce
  the features we think about
- an architecture = a parameterized family

Architecture: prior on the function

- priors as constraints: what is expressible with this architecture?
- must (reasonably) networks have already a huge expressive power?
- probability: easy to reach?
- with random weights: reasonable function or not?
- how architectures yield good features "naturally" of random projections

Lo neural network initialized by non-linear random projections

In some cases: most of the performance is due to the architecture, not to the training

Lo Extreme Machine Learning: learn only the last layer

Lo quality of layers with random weights
Lo quantify the information flow: information theory \(\rightarrow\) "information bottleneck"
Bias of the architecture
  - move to Fourier space (CNN)
  - stacking convolutional layers
    - Convolutions: $\mathbb{R}^p \rightarrow \mathbb{R}^q \rightarrow \mathbb{R}^r \rightarrow \cdots$
    - Convolutional layers map pixels
  - stacking fully connected layers
    - Linear: $\mathbb{R}^p \rightarrow \mathbb{R}^q \rightarrow \cdots$
    - $y = \sum_{i=1}^{M_1} W_{1,i} x_i$
    - $y = \sum_{i=1}^{M_2} W_{2,i} x_i$...
    - $y = \sum_{i=1}^{M_n} W_{n,i} x_i$
  - expressive power: the same
  - functional space

Random initialization:
  - random, according to which law? To choose a law yielding good properties
  - *exploding/vanishing gradient*
    - $x \rightarrow \cdots \rightarrow y$
    - High $\gamma$ big
    - $\gamma \approx |\text{magifying factor}|$
    - $L$ layers
    - $\sum_{k=1}^{L} \| \frac{\partial J}{\partial W_k} \|^2 < 1$ ~ prevent exploding
    - $L \gamma < 1$: $y \approx 0$ ~ function $e^y$
    - $L \gamma > 1$: $y \propto e^y$

  - *Xavier G. W. initialization*
    - $w \sim \mathcal{U}[-\sqrt{\frac{2}{n}}, \sqrt{\frac{2}{n}}]$ for $n > 1$
    - $w \sim \mathcal{N}(0, \sigma^2=\frac{2}{n})$ for $n = 1$
  - justification:
    - $x_i$: iid of mean $0$, standard deviation $\sqrt{\frac{1}{n}}$
    - $w$: iid of mean $0$, variance $1$
    - neuron computes: $y = \sum_i w_i x_i$
    - statistical properties?
    - $\mathbb{E}_w \mathbb{E}_x [y] = \mathbb{E}_w \mathbb{E}_x [\sum_i w_i x_i] = 0$
Let variance: $E[w^2] = \frac{\sum_{i=1}^{n} E[w_i^2]}{n} = \frac{\sum_{i=1}^{n} E[w_i^2]}{n}$

$= \frac{\text{variance}}{1 - \text{bias}}$

Input $x \xrightarrow{\text{linear}} y$

$\text{variance} < 1$

=> $\sigma = 1 \rightarrow \text{no exploding/vanishing} \Rightarrow \text{reasonable initializations}$

+ non-linear activation: $y = \sigma(\sum w_i x_i + b)$

=> $\text{ReLU}$

 divides variance by 2

-> correct with a factor $\sqrt{2}$

- bias $b$? $\rightarrow$ initialize to 0

- other initialization laws (similar properties): He, xavier

[Bay, Hinton, 2015] 

$\frac{df}{dx}$

- Jacobian properties

$\frac{df}{dx}$

-> at initialization, $\frac{df}{dx}$

- Jacobian matrix: $\text{data adapts}$

$\text{weight}$

- $n_2$: number of neurons in layer $l$

- $\text{var}(\frac{df}{dx}) \times \text{Fried} = \frac{1}{n_2}$

- $\text{var}(\frac{df}{dx}) = \frac{E[(\frac{df}{dx})^2]}{n_2}$

$\beta E \text{trace} \leq x \leq e^\frac{E}{n_2}$

$\rightarrow$ better to choose equal widths (than one then layer)

$\rightarrow$ put sufficiently-many neurons in each layer

Designing architectures easy to train

- "deep" networks: not that deep in terms of optimization

$B \rightarrow \text{DNN} \rightarrow y \rightarrow \text{loss}$

- $\text{do we need small number of layers}$

- $\text{do we need long $\text{deep}$ network}$

II. Architecture zoo

For vision problems: CNN $\rightarrow$ share local filters $\Rightarrow$ invariance to translation

- Hierarchical model $\Rightarrow$ much greater generality power

Typical block: 2-Block

- input image $\rightarrow$ 1-Block $\rightarrow$ 1-Block $\rightarrow$ Fully connected
Recurrent networks:
- Basic RNN: \( h_{t+1} = f(h_t, x_t) \)
- LSTM
- GRU
- Minimal RNN

Scale & resolution
- classification: tasks in computer vision; pyramidal approach
- segmentation
- regression / high input/output aim

Depth & mixing blocks

- memory
- leaky: \( h_{t+1} = h_t + f(h_t, x_t) \)
- gates

\( \Rightarrow \) LSTM
GRU
Minimal RNN

Pyramidal

U66
AlexNet
LeNet

Loss the details

U-net

E - segmentation
**RedNet:**

\[ x' = f_1(x) \]
\[ x = x + f(x) \]
\[ y_j = \frac{\partial L}{\partial \theta} = \frac{\partial L}{\partial f_3} \times \frac{\partial f_3}{\partial f_2} \times \frac{\partial f_2}{\partial x} \times \frac{\partial x}{\partial \theta} \]

**Highway networks**
-> some idea + attention

**Orthogonal matrices**
-> train a Feedforward network: 1D layers

**Inception**

\[ \text{Start training: } \text{Loss}(y) + \text{Loss}(y') + \text{Loss}(y'') \]
\[ \text{when optimization fails} \]

**Attention**

\[ y = w_x a + w_y b \]
\[ \text{Linear combination} \]
\[ y = \frac{w_x}{F_x(c)} a + \frac{w_y}{F_y(c)} b \]
\[ \alpha = (1 - x) \]
\[ \gamma \in [0,1] \]
\[ y = \alpha a + (1 - \alpha) b \]
\[ \text{interpolation} \]
\[ y = \text{softmax}(c) \]
\[ \text{sigmoid} \]
Attention (query case)

\[ \sum_{i} w_i \mathbf{v}_i \]

Self-attention:

\[ \text{softmax} \]

Multitask:

\[ \mathbf{U} = \mathbf{M} \times \mathbf{N} \times \mathbf{P} \]

Graph-NN (CNN)

\[ \sum w_{i} x_{i} \]

- analogy of CNN for graph
- computer vision
- any graph
for each node: \( F(a_v, \{ \text{neighbors} \}) \)

\[
\begin{align*}
y_v &= 2a_v + \sum_{e \in \text{neighbors}(v)} w_e (e) a_i \\
& \quad \text{(towards attention mechanism)}
\end{align*}
\]

GAT: graph attention network

very flexible architecture

\[
\begin{align*}
\text{set of nodes} & \xrightarrow{F} \text{new values for nodes} \\
\text{set of edges} & \xrightarrow{F} \text{new values for edges}
\end{align*}
\]

keep the same graph

\( G \to \text{graph classification?} \)

\[
\begin{align*}
\text{input: graph} & \xrightarrow{\text{decrease regularly}} \text{size of the graph} \\
& \quad \text{or FC}
\end{align*}
\]