Introduction to Deep Learning

by Boris Hanin

June 12, 2020
Deep Learning Tutorial

Goal: ① What is a NN?
   ② How NNs are used?
   ③ What kinds of choice are made by engineers?
   ④ Main use cases and failure modes?

Supervised Learning

① Data Acquisition: obtain dataset
   \[ D = \{ (x_k, f(x_k)) \}_{k=1}^K \] (f unknown)
   - e.g., \( x_k \) is image; \( f(x_k) = \{ 1, 0 \} \), \( x_k \) has 0 or

② Model Selection: choose model
   \[ x \rightarrow \mathcal{M}(x; \Theta) \]
   where \( \Theta \) = vector of params
   - e.g., \( \mathcal{M}(x; \Theta) = \alpha_1 x_1 + \ldots + \alpha_n x_n \), \( \Theta = (\alpha) \)

③ Optimization: Use \( D \) to obtain setting \( \Theta^* \) s.t.:\[ f(x_k) \approx \mathcal{M}(x_k; \Theta^*) \]
   e.g., linear regression

④ Testing: See how well \( \Theta^* \) performs on unseen data:
   \[ f(x) \approx \mathcal{M}(x; \Theta^*) ? \]

Salient Features:

- input dim \( \gg 1 \) (10^2 \approx 10^6)
- \#params \( \gg \#data \gg 1 \)
- NNs both interpolate and extrapolate
Neural Nets:

- Neural nets are built of "neurons"
- $x = (x_1, ..., x_n) \mapsto z(x; \Theta) = \sigma(b + x_1w_1 + ... + x_nw_n)$
- $\Theta = (b, w_1, ..., w_n)$

- "Def" A neural network is a collection of neurons and wiring diagram

$$\mathcal{N}(x; \Theta)$$
In practice, NNs have “layers”:

- Input $x_1, x_2$
- 1st layer
- 2nd layer
- Output $x_3$

- Layers $\rightarrow$ hierarchical reps
- $x_1 \rightarrow x_1^{(1)} \rightarrow x_1^{(2)} \rightarrow N(x_j; \theta)$

- Typical Use:
  - $\mathbb{R}^{n_0}$, $n_0 > 1$

1. **Data Acquisition**: $D = \{(x_k, f(x_k))\}$
2. **Architecture Selection**: choose $\sigma$’s, wiring diagram, depth, width
3. **Randomly initialize**: $\Theta = \{W, b\}$
4. **Testing**: Draw new $(x, f(x))$ and check whether $N(x; \Theta^*) \approx f(x)$

Empirically: deeper is better*

- Optimize $\Theta$ by gradient descent on

\[
L(\Theta) = \frac{1}{|D|} \sum_{x_1} l_x(\Theta); \quad l_x(\Theta) = 1 f(x) - N(x; \Theta)^2
\]
Main Uses:

1. NLP
   - $x_k = "the$ cat is big"
   - $f(x_k) = "le$ chat est grand"
   - Google Translate
   - Siri
   - Chat Bots

2. Computer Vision
   - $x_k = \text{image}$

3. Reinforcement Learning
   - $x_k =$ state of system
     (e.g., position of chess board)
   - $f(x_k) =$ reward $-$ max action
     (e.g., best next move)

- Self-Driving Cars
- Facial Rec
- Exploration by $\text{AU}$
How to choose $\beta_2$, $\beta_1$?

**Key:** How to choose $\beta_2$, $\beta_1$?

<table>
<thead>
<tr>
<th>Schedules</th>
<th>Intuition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\beta_2: \frac{\text{noise}}{\text{epochs}}$</td>
<td>for all $\beta_2$, fast but noisy</td>
</tr>
<tr>
<td>$\beta_1: \frac{\text{accuracy}}{\text{epochs}}$</td>
<td>for all $\beta_1$, slow but accurate</td>
</tr>
</tbody>
</table>

$\beta_2$:
- $eta_1$ is inversely related to $\beta_2$.
- Large $\beta_1$ = noisy but fast.
- Small $\beta_1$ = accurate but slow.

$\beta_2$:
- Periodic steps $(\lambda = \text{const})$.
- Every $2\pi$.

$\beta_1$:
- Single steps.
- Every epoch.

GD:
\[ \Delta w = - \eta \frac{\partial}{\partial w} J(w) \]

Compute $\frac{\partial J}{\partial w}$ using the chain rule.

GD:
\[ \frac{\partial J}{\partial w} \approx \frac{1}{b} \sum_{i=1}^{b} \frac{\partial J}{\partial w_i} \]

Replace SGD by \( \frac{1}{b} \sum_{i=1}^{b} \frac{\partial J}{\partial w_i} \).

**Intuition:**
- Small $\beta_2$ as noisy but fast.
- Large $\beta_2$ as accurate but slow.

**Intuition:**
- Small $\beta_1$ as fast but noisy.
- Large $\beta_1$ as accurate but slow.

**Intuition:**
- Batch size: small batches mean less computation.
- Batch size: small batches mean less computation.

**Intuition:**
- Small $\beta_1$ as fast but noisy.
- Large $\beta_1$ as accurate but slow.
Architecture Selection:

- Best architecture is always data-dependent:
  - NLP $\leftarrow \begin{cases} \text{Transformer-based} \\ \text{Recurrent (CLSTM + Attention)} \end{cases}$
  - CV $\leftrightarrow$ Convolutional, Residual

- Still leaves many choices:
  - Details of wiring (width, depth...)
  - Choice of $\sigma$ (= ReLU)
  - How to initialize and to optimize

- Empirical: deep is good* 

* = but often less stable
Residual Network:

\[ x \rightarrow N_1 \rightarrow N_2 \rightarrow \ldots \rightarrow N_j \]

output = \[ x + N_1(x) + N_2(x + N_1(x)) + \ldots \]

Intuition: \[ N_j \] is the \( j \)th order correction to \[ x \mapsto x \]

ConvNets: inputs are nxn RGB

Key: Images are hierarchical and local

R G B all neurons \( \rightarrow \) 1st

Look for same pattern
Challenges

1. New Use Cases:
   - PDE (fluids, physics, chemical...)
   - Biology (genomics)

2. Distribution Shift (nature of data change):
   - Change in hardware
   - Sunny vs. cloudy
   - Issue: NNs tend to be brittle