Lecture 4: Feature Perspective, Taylor Expansion, and Initialization

Jincheng Ou

Fall 25

Contents

1 Introduction

This lecture explores the theoretical underpinnings of neural network optimization and training dynamics. We adopt a "linear perspective" on neural networks using Taylor expansions to derive optimization algorithms like Adam (SignSGD). Furthermore, we analyze the critical roles of data standardization and weight initialization to maintain expressive power and prevent gradient issues.

2 The Linear Perspective of Neural Networks

2.1 Two Perspectives on Linearity

There are two primary ways to view a neural network through a linear lens:

1. **The Feature Extractor View:** The network layers preceding the final layer act as a complex, non-linear feature extractor. The final layer is simply a linear model acting on these learned features.
2. **The Taylor Expansion View:** We consider the network $f(x, \theta)$ as a function of its parameters θ. We can analyze the training dynamics by linearizing the network around the current parameters.

2.2 Taylor Expansion and "Lazy Training"

Consider a network $f : \mathbb{R}^m \to \mathbb{R}$ parameterized by $\theta \in \mathbb{R}^m$. To understand how the network output changes as we update parameters from θ to $\theta + \Delta\theta$, we apply a Taylor expansion:

$$f(x, \theta + \Delta\theta) = f(x, \theta) + \langle \nabla_\theta f(x, \theta), \Delta\theta \rangle + O(||\Delta\theta||^2) \tag{1}$$

- The term $\langle \nabla_\theta f, \Delta\theta \rangle$ represents the linear approximation of the network update.
- The gradient entries $\nabla_\theta f = \left(\frac{\partial f}{\partial \theta}\right)^T$ can be interpreted as "features" of the data relative to the parameters.

Lazy Training Assumption: This approximation relies on $\Delta\theta$ being small. The "Lazy Training" assumption posits that in many practical scenarios, weights move by very small amounts during training, keeping the first-order Taylor approximation valid within a local neighborhood.

3 Revisiting Optimization Algorithms

3.1 The Fundamental Tension

When designing descent algorithms, we face a trade-off:

- **Convergence Speed:** We want large steps ($||\Delta\theta||$) to converge quickly.
- **Approximation Validity:** We need small steps to ensure the Taylor approximation holds.

3.2 Deriving Adam (SignSGD) via L_∞ Constraints

To resolve this tension, we model the optimization step as maximizing the descent direction subject to a constraint on the step size.

Let $\mathcal{L}$ be the loss function. We aim to find the update step $\Delta\theta$ that minimizes the linearized loss:

$$\Delta\theta^* = \underset{\Delta\theta}{\operatorname{argmin}} \left[\mathcal{L}(\theta) + \langle \nabla_\theta \mathcal{L}, \Delta\theta \rangle\right] \tag{2}$$

Subject to a constraint on the step size. If we choose the L_∞ **norm** (infinity norm) for this constraint:

$$||\Delta\theta||_\infty \leq \eta \quad \implies \quad \max_j |\Delta\theta_j| \leq \eta \tag{3}$$

The optimization problem decouples element-wise because the L_∞ constraint applies to each coordinate independently. We solve for each coordinate j:

$$\Delta\theta_j^* = \underset{|\Delta\theta_j| \leq \eta}{\operatorname{argmin}} \left(\nabla_\theta \mathcal{L}\right)_j \cdot \Delta\theta_j \tag{4}$$

To minimize the inner product $\nabla\mathcal{L} \cdot \Delta\theta$, we must choose $\Delta\theta_j$ to have the opposite sign of the gradient component $(\nabla\mathcal{L})_j$ and maximal magnitude η:

- If $(\nabla\mathcal{L})_j > 0$, set $\Delta\theta_j = -\eta$.
- If $(\nabla\mathcal{L})_j < 0$, set $\Delta\theta_j = +\eta$.

Thus, the optimal step is:

$$\Delta\theta = -\eta \cdot \operatorname{sign}(\nabla_\theta \mathcal{L}) \tag{5}$$

This is exactly **SignSGD**. This derivation provides a theoretical motivation for **Adam** (which can be viewed as a smoothed version of SignSGD) as the optimal descent algorithm under an L_∞ trust region constraint.

4 Data Standardization and Expressive Power

Standardizing data (e.g., zero mean, unit variance) is crucial for numerical precision and conditioning. However, in Deep Learning, it is also essential for utilizing the expressive power of non-linearities.

4.1 The ReLU "Elbow" Problem

Consider a simple ReLU unit $f(x) = \text{ReLU}(wx + b)$. We typically initialize $w, b \sim \mathcal{N}(0, 1)$. The non-linearity (the "elbow") occurs when the argument is zero:

$$wx + b = 0 \implies x = -\frac{b}{w} \tag{6}$$

If w, b are independent standard normal variables, the ratio $R = -b/w$ follows a **Cauchy Distribution**. The Cauchy PDF is $p(x) = \frac{1}{\pi(1+x^2)}$. Key properties:

- It is heavy-tailed with undefined mean and infinite variance.
- However, 50% of the probability mass lies between $[-1, 1]$ and 90% between $[-7, 7]$.

4.2 Consequences of Unscaled Data

If input data is not standardized (e.g., values in range $[100, 200]$) while weights are standard normal:

- The probability of a ReLU "elbow" landing in the data range $[100, 200]$ is approx 0.1%.
- **Result:** For almost all inputs x, the neuron is either always 0 (inactive) or always linear ($wx + b$).

Mathematically, if no non-linear corners are hit, the hidden layer output h becomes a linear function of x:

$$h_i(x) = \alpha_i x + \beta_i \quad \text{or} \quad h_i(x) = 0 \tag{7}$$

This leads to a **low-rank feature matrix**, effectively collapsing the deep network into a linear model and losing expressive power.

5 Initialization Strategies

Even if inputs are standardized, we must ensure weights are initialized such that activations remain standardized (mean ≈ 0, variance ≈ 1) as they propagate through deep layers.

Why not initialize to zero? If weights $W = 0$, activations are zero. More critically, gradients with respect to weights depend on the input to the weights. If inputs are zero, gradients are zero, and weights never update (Symmetry breaking problem).

5.1 Xavier (Glorot) Initialization

Designed for linear or Tanh/Sigmoid activations. **Goal:** Maintain unit variance of activations. Let $y = \sum_{i=1}^{d_{in}} w_i x_i + b$. Assume x_i, w_i are independent with zero mean.

$$\begin{aligned}
\text{Var}(y) &= \text{Var}\left(\sum_{i=1}^{d_{in}} w_i x_i\right) && (8)\\
&= \sum_{i=1}^{d_{in}} \text{Var}(w_i x_i) && (9)\\
&= \sum_{i=1}^{d_{in}} E[w_i^2]E[x_i^2] \quad (\text{since } E[w] = E[x] = 0) && (10)\\
&= d_{in} \cdot \text{Var}(w) \cdot \text{Var}(x) && (11)
\end{aligned}$$

If we want $\text{Var}(y) = \text{Var}(x)$, we require:

$$d_{in} \cdot \text{Var}(w) = 1 \implies \text{Var}(w) = \frac{1}{d_{in}} \tag{12}$$

Thus, Xavier initialization draws weights from $w \sim \mathcal{N}(0, \frac{1}{d_{in}})$.

5.2 He (Kaiming) Initialization

Designed specifically for **ReLU** networks. **Observation:** ReLU outputs are zero for half the inputs (assuming symmetric distribution around 0). This effectively halves the variance propagation. The "effective" fan-in is $d_{in}/2$.

To compensate and maintain unit variance, we need double the variance in the weights:

$$\text{Var}(w) = \frac{2}{d_{in}} \tag{13}$$

Thus, He initialization draws weights from $w \sim \mathcal{N}(0, \frac{2}{d_{in}})$.

Method	**Target Activation**	**Variance Formula**
Xavier (Glorot)	Tanh / Linear / Sigmoid	$\text{Var}(W) = \frac{1}{\text{fan\_in}}$
He (Kaiming)	ReLU / LeakyReLU	$\text{Var}(W) = \frac{2}{\text{fan\_in}}$

Table 1: Summary of Initialization Schemes