Technical Report:

WEI FENG

HUST

Sept 1st, 2025

**Abstract**

.

**Contents**

**[1 Overview](#bookmark1) 2**

**[2 Model and Framework](#bookmark2) 2**

**[3 Data and Formatting](#bookmark3) 2**

**[4 Training Setup](#bookmark4) 3**

**[5 Evaluation Setup](#bookmark5) 4**

**[6 Experimental Results](#bookmark6) 5**

**[7 Discussion and Insights](#bookmark7) 5**

**[8 Reproducibility and Engineering Details](#bookmark8) 6**

**[9 Conclusion](#bookmark9) 6**

**Keywords:**

**1 Overview**

梯度下降法是优化神经网络参数的核心算法，而其高效实现依赖于快速、准确地计算损失函数对于数百万乃至数十亿参数的梯度。反向传播算法正是实现这一目标的关键。本报告基于讲座内容，旨在提供一个从数学基础到算法实现，再到工程实践的完整视角。

报告首先从矩阵微积分入手，阐明如何利用雅可比矩阵高效地表示多变量函数的导数，并将单变量链式法则推广到多维情况。接着，报告引入计算图作为表示任意复杂函数的有力工具，并在此框架下详细讲解反向传播算法的工作原理。算法的核心在于“局部梯度”的概念以及通过递归应用链式法则实现的计算复用，这极大地提高了梯度计算的效率。报告通过一个手算示例，直观演示了梯度在计算图中的反向流动与在多路径节点的累加方式。最后，报告探讨了算法在现代框架中的实现方式、数值验证方法，并深入讨论了理解这些细节对于在实际应用中有效设计和调试神经网络的重要意义。

**2 Model and Framework**

讲座中用于推导的核心神经网络模型是一个简单的单隐藏层全连接网络，其数学定义如下：

h = f(z)

z = Wx+b

x∈ 是输入向量（例如，一个词或句子的向量表示）。

W∈是输入层到隐藏层的权重矩阵。

b∈ 是隐藏层的偏置向量。

z∈ 是隐藏层的线性预激活值。

f 是逐元素应用的激活函数（如 Sigmoid, Tanh, ReLU），因此 h∈ 是隐藏层的激活后输出。u∈是隐藏层到输出层的权重向量。

s∈R 是网络的最终输出得分。

该模型的框架是理解反向传播的范例。目标是计算标量输出 s 对于所有参数 (u,W,b) 的梯度。计算过程基于链式法则，并被系统地组织成计算图。计算图由节点（代表变量或操作）和边（代表数据流）组成。例如，该模型的计算图可表示为：输入 x 与权重 W 相乘，加上偏置 b 得到 z，经激活函数 f 得到 h，最后与 u 做点积得到 s。反向传播算法正是在这个图上逆向进行，计算梯度。

**3 Data and Formatting**

讲座内容侧重于算法原理，因此使用的数据是高度抽象和符号化的。输入向量 x 被示例性地表示为：

x = [Xmuseums, Xin, XParis, Xare, Xamazing]

这表示一个与文本“museums in Paris are amazing”相关的特征向量，其中每个 Xword 代表对应单词的特征值。

在实际应用中，数据需要经过严格的格式化处理：

数值化：所有数据（图像、文本、音频）必须转换为数值张量。例如，文本需通过词嵌入技术转换为浮点数向量。

批处理：为提升计算效率，训练通常以小批量进行。因此，输入 x 会变为矩阵 X∈。

归一化：对输入数据进行归一化或标准化是常见的前处理步骤，以加速训练并提高稳定性.

形状约定：讲座中强调了一个重要的工程实践——形状约定，即参数梯度的形状应与参数本身的形状保持一致。这使得梯度可直接用于参数更新（如 θ←θ−αJ），而无需繁琐的reshape操作。

**4 Training Setup**

神经网络的训练是一个迭代过程，旨在通过梯度下降最小化损失函数 J(θ)J(θ)，其中 θθ 代表所有参数。基于反向传播的训练设置包含以下关键步骤：

前向传播：按照计算图的拓扑顺序，从输入节点开始，计算每个节点的输出值。对于每个节点，保存其输入和输出值，因为它们将在反向传播中被用于计算局部梯度。对于我们的模型，前向传播顺序为：x→z = Wx+b → h=f(z)→s=→J(s,y),其中 y 是真实标签。

损失计算：将网络输出 s 与真实标签 y 进行比较，通过损失函数（如交叉熵损失用于分类，均方误差用于回归）计算标量损失值 J。该损失值衡量了当前网络预测的不准确程度。

反向传播：这是训练的核心。

初始化：从损失节点开始，设置 ∂J / ∂J = 1。

递归应用链式法则：按照与前向传播相反的顺序遍历节点。对于每个节点，根据其“上游梯度”（即损失对该节点输出的梯度）和该操作的“局部梯度”（即该节点输出对其输入的梯度），计算“下游梯度”（即损失对该节点输入的梯度）。核心公式为：

∂J / ∂x = ∂J / ∂y⋅∂y / ∂x ​

其中 y 是节点的输出，x 是节点的输入。在矩阵形式下，即为雅可比矩阵的乘法。

关键优化——计算复用：讲座重点强调了误差信号 δ 的概念。在计算 ∂s / ∂b和 ∂s / ∂W时，都共享了相同的上游梯度 δ=∂s / ∂h·∂h / ∂z=·f′(z)。这避免了重复计算，使得反向传播的计算复杂度与前向传播同阶。

参数更新：在获得所有参数的梯度 ∇θJ∇θ​J 后，使用优化算法（如随机梯度下降SGD）更新参数：

= − αJ

其中 α 是学习率，控制更新的步长。

**5 Evaluation Setup**

梯度计算正确性可以通过中心差分公式来近似：

IMG_256

若解析梯度与数值梯度接近，则实现正确。评估标准通常为比较两者的相对误差：

算法优点是实现简单，是验证梯度正确性的“黄金标准”。缺点是计算速度极慢，因为需要 perturb 每个参数并重新计算损失，不适用于大规模模型的日常训练，仅用于调试和验证。

**6 Experimental Results**

讲座中以函数为例进行手算演示：

f(x,y,z) = (x+y)·max(y,z)

在点 (1,2,0) 处，通过计算图反向传播得到：

∂f / ∂x = 2, ∂f / ∂y=5,∂f / ∂z = 0​

验证了梯度在分支节点处的累加特性。

**7 Discussion and Insights**

计算复用：反向传播中，高层梯度可复用至低层，避免重复计算如误差信号 δ = ·f′(z)

形状约定：梯度形状应与参数形状一致，便于 SGD 实现

节点类型：

加法节点：梯度均匀分配

乘法节点：梯度交换输入

Max 节点：梯度仅传递至最大输入

**8 Reproducibility and Engineering Details**

现代深度学习框架（如 PyTorch、TensorFlow）通过自动微分实现反向传播。每个算子需实现 forward 和 backward 方法，简单实现为：

class MultiplyGate:

def forward(self, x, y):

self.x, self.y = x, y

return x \* y

def backward(self, dz):

dx = self.y \* dz

dy = self.x \* dz

return dx, dy

**9 Conclusion**

理解梯度计算与反向传播的数学原理是设计和调试神经网络的基础。尽管现代框架已实现自动微分，掌握其底层机制仍对模型优化、调试与创新至关重要。

**Acknowledgments**

**References**

[1] Lecture: Neural net learning: Gradients by hand (matrix calculus) and algorithmically (the backpropagation algorithm)

[15] Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. spacy:

Industrial-strength natural language processing in python. <https://spacy.io/>, 2020.

Accessed: 2025-08-10.

[16] Hernan E. Grecco and contributors. Pint: Python units library. [https://github.com/](https://github.com/hgrecco/pint) [hgrecco/pint](https://github.com/hgrecco/pint). Accessed: 2025-08-10.