Deep Learning

03 – Gradient-Based Training

Part 0: Overview

Prof. Dr. Rainer Gemulla

Universität Mannheim

Version: 2024-1

Supervised training of FNNs

- In principle: like any other ML model
- Often: empirical risk minimization (our focus)
  - Frequentist approach, obtains point estimate $\hat{\boldsymbol{\theta}}$ of FNN parameters $\boldsymbol{\theta}$
  - Use non-negative, real-valued **loss function** $L(\hat{\boldsymbol{y}}, \boldsymbol{y})$ between a prediction $\hat{\boldsymbol{y}}$ and a true answer $\boldsymbol{y}$
  - Minimize **empirical risk** = average loss on training data $\{ (\boldsymbol{x}_i, \boldsymbol{y}_i)_{i=1}^N \}$

$$R_{\text{emp}}(\boldsymbol{\theta}) = \frac{1}{N} \sum_i L(\hat{\boldsymbol{y}}_i, \boldsymbol{y}_i) \qquad \text{where } \hat{\boldsymbol{y}}_i = f(\boldsymbol{x}_i; \boldsymbol{\theta})$$

- Some common loss functions
  - Squared error (for regression)
  - Log loss / binary cross entropy (for binary / multi-label classification)
  - Cross entropy / KL divergence (for multi-class classification)
  - Hinge loss (for margin-based classification)
  - 0-1 loss / misclassification rate (for classification)
- Generally: use **cost function** $J(\boldsymbol{\theta})$
  - E.g., regularized risk to prevent overfitting

Gradient-based methods

- Gradient-based methods are dominant
  - Large datasets, many parameters
  - Many tricks used to make these methods work empirically

- General approach
  1. Construct a **batch** (e.g., a subset of examples)
  2. Compute gradients of cost function on batch
  3. Update parameters using an **optimizer**
  4. Repeat

Training Techniques

- Gradient-based methods are a *tool* to minimize some cost function (backpropagation, optimizers)
- Training FNNs successfully is also an *art*; generally, goals include
  - Improve performance of gradient-based methods
  - Reduce overfitting, improve generalizability
  - Leverage additional data
  - Reduce (task-specific) costs such as model size, computational costs, amount of required supervision, ...
- In this part of the lecture, we look at
  - Compute graphs, automatic differentiation
  - Gradient computation via **backpropagation** ("backprop"): chain rule + reuse of computations
  - Optimizers beyond plain SGD
  - Challenges in gradient-based training (vanishing/exploding gradients) and impact of architectural choices

Outline (Gradient-Based Training)

0. Overview
1. Backpropagation
2. Optimizers
3. Architecture design

Summary

- Gradient-based methods dominant for training deep learning models
- Backpropagation
  - Technique to compute gradient of a computation w.r.t. its inputs
  - Computation modeled via a compute graph
  - Chain rule + reuse of computations
  - Forward pass to compute all outputs (**forward propagation**)
  - Backward pass to compute all gradients (**backward propagation**)
  - Used to support automatic differentiation in DL frameworks
- Training deep FNNs is an art
  - Complex models, many hyperparameters, many techniques, expensive
  - Use optimizers with adaptive learning rates and momentum
  - Vanishing/exploding gradients can be problematic
  - Architectural choices matter (e.g., saturating units, residual units, skip connections)

Suggested reading

- Drori, Ch. 2, 3
- Goodfellow et al., Ch. 6, 8
- Murphy 1, Ch. 13.3, 13.4