Mathematics for Machine Learning

Prof Willie Brink

Applied Mathematics, Stellenbosch University

Lecture 6: Continuous Optimisation

Contents of the module

Introduction

Machine learning often boils down to finding a good set of parameters for a model (e.g. a neural network or a probabilistic model).

We define an objective function, and minimise it using numerical optimisation:

- initialise and then iteratively update the parameter values.

The gradient of the objective function w.r.t. the parameters will be particularly useful in indicating which direction to update the parameters.

A step size (a.k.a. the learning rate) dictates the speed of updates.

7.1 Optimisation using gradient descent

Let $f : \mathbb{R}^d \to \mathbb{R}$ be differentiable, and consider the minimisation problem: $\min_{\boldsymbol{x}} f(\boldsymbol{x})$

Gradient descent starts with an initial guess $\boldsymbol{x}_0$, then iterates:

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i - \gamma_i \big((\nabla f)(\boldsymbol{x}_i)\big)^{\mathsf{T}}$$

For suitable step-size γ_i, the sequence $f(\boldsymbol{x}_0), f(\boldsymbol{x}_1), \ldots$ converges to a local minimum.

Gradient descent can be slow close to the minimum. For poorly conditioned problems, it may "zig-zag" as the gradients are nearly orthogonal to the shortest distance to the minimum point.

Choosing an appropriate step-size, a.k.a. learning rate, is important!

- too small, and GD can be slow
- too large, and GD can overshoot, fail to converge, or diverge

Adaptive learning rate

- When the function f increases after a gradient step, the step-size was too large. Undo the step and decrease the step-size.
- When the function f decreases, the step could have been larger. Try to increase the step-size.

Momentum

The curvature of the optimisation surface may cause GD to hop over the minimum.

Let's introduce an extra term to remember what happened in the previous iteration:

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i - \gamma_i\big((\nabla f)(\boldsymbol{x}_i)\big)^\top + \alpha(\boldsymbol{x}_i - \boldsymbol{x}_{i-1}) \quad \text{with } \alpha \in [0, 1]$$

This combination of current and previous gradients dampens oscillations in the updates.

Stochastic gradient descent

In machine learning the objective function f is often an average over training samples:

$$f(\boldsymbol{x}) = \frac{1}{N}\sum_{i=1}^{N} L_i(\boldsymbol{x}) \quad \text{and} \quad \nabla f(\boldsymbol{x}) = \frac{1}{N}\sum_{i=1}^{N} \nabla L_i(\boldsymbol{x})$$

We can get an inexpensive, unbiased estimate of the gradient by sampling m data points.

$m = N$: **batch** gradient descent

$m < N$: **mini-batch** gradient descent

$m = 1$: **stochastic** gradient descent (online)

Can be very effective in large-scale deep learning, and may enable escape from undesired stationary points. Small mini-batches also give a more noisy estimate of the gradient, which can provide regularisation.

7.2 Constrained optimisation and Lagrange multipliers

Consider the following constrained optimisation problem:

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) \quad \text{subject to} \quad g_i(\boldsymbol{x}) \leq 0, \quad i = 1, \ldots, m$$

This "primal problem" can be converted to its associated Lagrangian dual problem:

$$\max_{\boldsymbol{\lambda} \in \mathbb{R}^m} \left[\min_{\boldsymbol{x} \in \mathbb{R}^d} \left(f(\boldsymbol{x}) + \sum_{i=1}^{m} \lambda_i g_i(\boldsymbol{x}) \right) \right] \quad \text{subject to} \quad \lambda_i \geq 0, \quad i = 1, \ldots, m$$

When the inner minimisation problem is easy to solve, the overall problem is easy to solve (the outer maximisation problem involves a convex function in $\boldsymbol{\lambda}$).

7.3 Convex optimisation

C is a convex set if for any $x, y \in C$ and $\theta \in [0, 1]$, we have that $\theta x + (1 - \theta) y \in C$.

The function $f : \mathbb{R}^d \to \mathbb{R}$ whose domain is a convex set is a convex function if for all $\boldsymbol{x}, \boldsymbol{y}$ in the domain of f, and $\theta \in [0, 1]$ we have

$$f\big(\theta \boldsymbol{x} + (1 - \theta)\boldsymbol{y}\big) \leq \theta f(\boldsymbol{x}) + (1 - \theta) f(\boldsymbol{y}) \qquad \text{[a form of Jensen's inequality]}$$

A differentiable function f is convex if and only if, for any $\boldsymbol{x}, \boldsymbol{y}$ in the domain of f,

$$f(\boldsymbol{y}) \geq (\boldsymbol{x}) + \nabla_{\boldsymbol{x}} f(\boldsymbol{x})^{\mathsf{T}}(\boldsymbol{y} - \boldsymbol{x})$$

Optimisation problems involving convex functions $f(\cdot)$ and $g_i(\cdot)$ are particularly useful, since we can guarantee global optimality.

Examples: linear and quadratic programming (sections 7.3.1 and 7.3.2)