Machine Learning and Neural Networks (MATH3431) Epiphany term, 2023

Handout 1: Gradient descent

Lecturer & author: Georgios P. Karagiannis georgios.karagiannis@durham.ac.uk

Aim. To introduce gradient descent, its motivation, description, practical tricks, analysis in the convex scenario, and implementation.

**This handout is still Under construction

1. Motivations

Note 1. Consider a learning problem $(\mathcal{H}, \mathcal{Z}, \ell)$. Learning may involve the computation of the minimizer $h^* \in \mathcal{H}$, where $\mathcal{H}$ is a class of hypotheses, of the empirical risk function (ERF) $\hat{R}(h) = \frac{1}{n}\sum_{i=1}^{n} \ell(h, z_i)$ given a finite sample $\{z_i; i = 1, ..., n\}$ generated from the data generating model $g(\cdot)$ and loss $\ell(\cdot)$; that is

$$w^* = \arg\min_{\forall h \in \mathcal{H}} \left(\hat{R}(h)\right) \tag{1.1}$$

If analytical minimization of (1.1) is impossible or impractical, numerical procedures can be applied; eg Gradient descent. Such approaches introduce additional errors in the solution.

2. Description

Notation 2. For the sake of simplicity and generalization, we will present Gradient descent in the following minimization problem

$$w^* = \arg\min_{\forall w \in \mathcal{H}} \left(f(w)\right) \tag{2.1}$$

where here $f : \mathbb{R}^d \to \mathbb{R}$, and $w \in \mathcal{H} \subseteq \mathbb{R}^d$. Eg, f can be an ERF.

Assumption 3. *Assume (for now) that $f : \mathbb{R}^d \to \mathbb{R}$ is differential function.*

Definition 4. The Gradient descent algorithm for the solution of the minimization problem 2.1 is in Algorithm 1

Algorithm 1 Gradient descent algorithm

- For $t = 1, 2, 3, ...T$, iterate:

$$w^{(t+1)} = w^{(t)} + \eta_t \nabla f\left(w^{(t)}\right)$$

where

$$\nabla f(w) = \left(\frac{\partial}{\partial x_1} f(w), \ldots, \frac{\partial}{\partial x_d} f(w)\right)^\top$$

is the gradient of f at .

Remark 5. As a motivation, consider the 2nd-Taylor polinomial for the approximation of $f(w)$ around for u in a small area around w

$$f(u) \approx P(u) = f(w) + \langle u - w, \nabla f(w) \rangle$$

Assuming convexity for f, it is

$$f(u) \geq \underbrace{f(w) + \langle u - w, \nabla f(w) \rangle}_{=P(u)}$$

meaning that P lower bounds f. Hence we could design an update mechanism producing $w^{(t+1)}$ that is nearby $w^{(t)}$ (small steps) and

$$f(w) \approx f\left(w^{(t)}\right) + \langle w - w^{(t)}, \nabla f\left(w^{(t+1)}\right) \rangle. \tag{2.2}$$

Hence we could recursively minimize the approximation (2.2) and the distance between the current state $w^{(t)}$ and the next w value to produce $w^{(t+1)}$; namely

$$\begin{aligned} w^{(t+1)} =& \arg\min_{\forall w} \left(\frac{1}{2} \left\| w - w^{(t)} \right\| + \eta P(w) \right) \\ =& \arg\min_{\forall w} \left(\frac{1}{2} \left\| w - w^{(t)} \right\| + \eta \left(f\left(w^{(t)}\right) + \langle w - w^{(t)}, \nabla f\left(w^{(t+1)}\right) \rangle \right) \right) \\ =& w^{(t)} + \eta_t \nabla f\left(w^{(t)}\right) \end{aligned}$$

where parameter $\eta > 0$ controls the trade off.

Remark 6. The output of GD can be (but not a exclusively), the average

$$w_{\text{GD}}^{(T)} = \frac{1}{T} \sum_{t=1}^{T} f\left(w^{(t)}\right)$$

after discarding the first few iterations of $w^{(t)}$ for stability reasons, or the best value discovered

$$w_{\text{GD}}^{(T)} = \arg\min_{\forall w_t} \left(f\left(w^{(t)}\right) \right)$$

or the last value discovered

$$w_{\text{GD}}^{(T)} = w^{(t)}$$

Note 7. GD algorithm converges to a local minimum, $w_{\text{GD}}^{(T)} \to w_*$, under different sets of regularity conditions (some weaker other stronger). In Section 4 perform an analysis.

Remark 8. The parameter η_t is called learning rate, step size, or gain. $\{\eta_t\}$ is a non-negative sequence and it is chosen by the practitioner. Regularity conditions (Note 7) often imply restrictions on the decay of $\{\eta_t\}$ which guide the practitioner to parametrise it properly. Some popular choices of learning rate $\eta_t$are:

(1) constant; $\eta_t = \eta$, where where η is a small value. The rational is that GD chain $\{w_t\}$ performs constant small steps towards the (local) minimum w_* and then oscillate around it.
(2) decreasing and converging to zero; e.g. $\eta_t = \left(\frac{C}{t}\right)^{\varsigma}$ where $\varsigma \in (0.5, 1]$ and $C > 0$. The rational is that GD algorithm at the begining starts by performing larger step to explore the area

for discovering possible minima while reducing the size of the steps with the iterations to converge to a possible w_* value.

(3) decreasing and converging to a tiny value τ_*; e.g. $\eta_t = \left(\frac{C}{t}\right)^{\varsigma} + \tau_*$ where $\varsigma \in (0.5, 1]$, $C > 0$, and $\tau_* \approx 0$. Same as previously, but the algorithm aims at oscillating around the detected local minimum.

(4) constant until an iteration T_0 and then decreasing; eg $\eta_t = \left(\frac{C}{\max(t, T_0)}\right)^{\varsigma}$, where $\varsigma \in (0.5, 1]$ and $C > 0$. The rational is that in the first stage of the iterations the algorithm may need constant larger stems for a significant number of iterations in order to explore the domain and hence the chain $\{w_t\}$ to reach the area around the (local) minimum w_*. In the second stage the chain $\{w_t\}$ may be in a close proximity to the (local) minimum w_* and hence the algorithm needs to perform smaller steps to exploit (converge to w_*). The first stage is called burn-in and it is discarded from the output of the GD algorithm.

Parameters C, ς, τ_* may be chosen based on pilot runs. A set of conditions for $\{\eta_t\}$ are those from Robbins–Monro algorithm

$$\sum_{t=1}^{\infty} \eta_t = \infty \text{ and } \sum_{t=1}^{\infty} \eta_t^2 < \infty$$

(to be seen and discussed later), satisfied by items 2 and 4.

3. Examples

4. Analysis

5. Subgradients