Department of Mathematics and Computer Science
University of Southern Denmark, Odense

March 25, 2025
Marco Chiarandini

AI505 – Optimization

Sheet 05, Spring 2025

Exercises with the symbol $^+$ are to be done at home before the class. Exercises with the symbol * will be tackled in class. The remaining exercises are left for self training after the exercise class. Some exercises are from the text book and the number is reported. They have the solution at the end of the book.

Exercise 1$^+$

Learn the basics of PyTorch `https://pytorch.org/tutorials/beginner/basics/intro.html`.

Exercise 2$^+$

Write the update rule for stochastic gradient with mini-batches of size m on a generical machine learning model $y = h(x)$ and with loss function L.
Write the update formula with momentum in the case of mini-batch of size m.

Exercise 3*

In a regression task we assume $h(\mathbf{x}; \mathbf{w}) = w_0 + w_1x_1 + w_2x_2 + \ldots + w_dx_d$. For the estimation of the parameters $\mathbf{w} \in \mathbb{R}^{d+1}$ using the examples $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ we can use the least squares loss function:

$$\min R_n(\mathbf{w}) = \sum_{i=1}^{n} (h(\mathbf{x}_i; \mathbf{w}), y_i) = \min \left\|\mathbf{y} - X\mathbf{w}\right\|_2^2$$

where

$$X = \begin{bmatrix} 1 & x_{11} & x_{21} & \ldots & x_{d1} \\ 1 & x_{11} & x_{21} & \ldots & x_{d1} \\ \vdots & \ddots & & & \\ 1 & x_{1n} & x_{2n} & \ldots & x_{dn} \end{bmatrix}$$

This problem admits a closed form solution by means of the normal equations $\mathbf{w} = (X^TX)^{-1}X^T\mathbf{y}$. You find the derivation of this result in these slides from DM579/AI511.
The L_2 Regularized risk is

$$\min R_n(\mathbf{w}) + \lambda \left\|\mathbf{w}\right\|_2^2 = \sum_{i=1}^{n} (h(\mathbf{x}_i; \mathbf{w}), y_i) + \lambda \sum_{j=0}^{d} w_j^2 = \min \left\|\mathbf{y} - X\mathbf{w}\right\|_2^2 + \lambda \left\|\mathbf{w}\right\|_2^2$$

admits also a closed-form solution: $\mathbf{w} = (X^TX + \lambda I)^{-1}(X^T\mathbf{y})$.
Provide a computational analysis of the cost of computing the estimates of $\mathbf{w}$ by means of these closed-form solutions and compare these costs with the cost of carrying out the gradient descent. When is thegradient descent faster?

Exercise 4*

Consider now multiple logistic regression. In this case the hypothesis is that the probability of $y = 1$ given $\mathbf{x}$ is given by

$$h(\mathbf{x}; \mathbf{w}) = p(\mathbf{x}; \mathbf{w}) = \frac{1}{1 + e^{-(w_0+w_1x_1+\ldots+w_dx_d)}} = \frac{e^{w_0+w_1x_1+\ldots+w_dx_d}}{1 + e^{w_0+w_1x_1+\ldots+w_dx_d}}$$

The loss-likelihood function can be formulated as follows:

$$L = \prod_{i:y_i=1} p_i \prod_{i:y_i=0} (1 - p_i)$$

$$R_n(\mathbf{w}) = \log L = \sum_{i=1}^{n} y_i \log_b(p(\mathbf{x}_i)) + \sum_{i=1}^{n}(1 - y_i)\log_b(1 - p(\mathbf{x}_i))$$

$$\min R_n(\mathbf{x}; \mathbf{w}) = \sum_{i=1}^{n} y_i \log_b(p(\mathbf{x}_i)) + \sum_{i=1}^{n}(1 - y_i)\log_b(1 - p(\mathbf{x}_i))$$

which we can't formulate as a linear system. Setting $\nabla R_n(\mathbf{w}) = 0$ gives a system of transcendental equations. But this objective function is convex and differentiable.
With some tedious manipulations, the gradient for logistic regression is

$$\nabla R_n(\mathbf{w}) = X^T \mathbf{s}$$

where vector $\mathbf{s}$ has $s_i = -y_i h(-y_i \mathbf{w}^T \mathbf{x}_i)$ and h is the sigmoid function and

$$\nabla^2 R_n(\mathbf{w}) = X^T D X$$

where D is a diagonal matrix with

$$d_{ii} = h(y_i \mathbf{w}^T \mathbf{x}_i) h(-y_i \mathbf{w}^T \mathbf{x}_i)$$

It can be shown that $X^T DX$ is positive semidefinite and therefore logistic regression is convex (it becomes strictly convex if we add L_2-regularization making the solution unique).
Hence, gradient descent converges to a global optimum. Alternately, another common approach is Newton's method. Requires computing Hessian $\nabla^2 R_n(\mathbf{w}_i)$.
What is the computational cost of gradient descent and of the Newton method for the logistic regression described?

Exercise 5*

Recall that a way to measure rate of convergence is by the *limit of the ratio of successive errors*,

$$\lim_{k\to\infty} \frac{f(\mathbf{w}_{k+1}) - f(\mathbf{w}^*)}{f(\mathbf{w}_k) - f(\mathbf{w}^*)} = r$$

Different r values of give us different rates of convergence:

1. If $r = 1$ we call it a sublinear rate.
2. If $r \in (0, 1)$ we call it a linear rate.
3. If $r = 0$ we call it a superlinear rate.

Consider the following sequences, which represent the error $e_k = \left\| F(\mathbf{w}_k) - F^* \right\|$ at iteration k of an optimization algorithm:

- $e_k = 0.5^k$
- $e_k = \frac{1}{k+1}$
- $e_k = 0.1^k$
- $e_k = \frac{1}{k+1}^2$

Tasks
a) Classify the convergence rate of each sequence as linear, sublinear, superlinear, or quadratic.
b) Provide a justification for each classification by computing the ratio e_{k+1}/e_k or by using the definition of order of convergence.

Exercise 6

Consider applying gradient descent to the one-dimensional quadratic function

$$f(x) = \frac{1}{2}x^2$$

with the update rule:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha \nabla f(\mathbf{x}_k)$$

where $\nabla f(x) = x \nabla f(x) = x$. Tasks
a) Derive the update formula for $\mathbf{x}_k$.
b) Show that the error $e_k = \|\mathbf{x}_k\|$ follows an exponential decay when $0 < \alpha < 20 < \alpha < 2$. c) Compute $e_{k+1} e_k$ and determine the rate of convergence for different values of α. d) Set up a Python experiment where gradient descent is applied with different step sizes (α) and verify the theoretical convergence rate numerically.
Hint: Try $\alpha = 0.1, 0.5, 1, 1.5$ and observe how quickly the errors decrease.