Problem Set 2: Scaling with Sampling

CS4787/5777 — Principles of Large-Scale ML Systems

This problem set is assigned on September 11, 2023 and is due about two weeks later on September 27, 2023 at 11:59PM. You may work in groups of up to four students. Submit your solutions on Gradescope.

Problem 1: Empirical Risk Minimization. Consider a machine learning task where we want to train a model h that maps examples $x \in \mathbb{R}^d$ to labels $y \in \{-1, 1\}$. We do so by selecting from a hypothesis class h_w parameterized by weights $w \in \mathbb{R}^d$, defined by

$$h_w(x) = \text{sign}\left(x^T w\right).$$

(Recall here that $x^T w$ denotes the dot product of the vectors x and w.) The sign function is $\text{sign}(\cdot)$: $x \mapsto x/|x|$ with the convention $0/0 = 0$. To train this model given a dataset of n labeled examples $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$, we solve the *empirical risk minimization* problem

$$\begin{aligned} \text{minimize: } & \frac{1}{n}\sum_{i=1}^{n} \log\left(1 + \exp\left(-y_i \cdot x_i^T w\right)\right) + \frac{\lambda}{2}\|w\|^2, \\ \text{over: } & w \in \mathbb{R}^d, \end{aligned}$$

where $\lambda > 0$ is a hyperparameter.

1(a). The given empirical risk minimization problem is an example of one of the following types of machine learning models:

(A) Soft-margin support vector machine

(B) Hard-margin support vector machine

(C) k-Nearest neighbors classifier

(D) Perceptron

(E) Ordinary least squares

(F) Logistic regression

Which one is it?

1(b). One way to solve this empirical risk minimization problem is to use gradient descent on the loss. Define the loss function $\ell : \mathbb{R}^d \to \mathbb{R}$ to be the function

$$\ell(w) := \frac{1}{n}\sum_{i=1}^{n} \log\left(1 + \exp\left(-y_i \cdot x_i^T w\right)\right) + \frac{\lambda}{2}\|w\|^2.$$

Let's try to understand some things about this empirical risk minimization problem. Answer each of the following questions, and provide a brief justification.

(i) Is the objective ℓ convex? (Recall that a function f is convex if it is "bowl-shaped" i.e. if any line segment drawn between two points on its graph lies above the graph.)

(ii) Is ℓ strongly convex? If so, can you find a simple value for its constant of strong convexity μ? (Recall that a twice-differentiable function f is μ-strongly convex if its second directional derivative along any direction is always no less than μ, i.e., $\frac{\partial^2}{\partial \alpha^2} f(w + \alpha u) \geq \mu$ for any $u, w \in \mathbb{R}^d$ with $\|u\| = 1$.)

(iii) Is the solution of this empirical risk minimization problem necessarily unique?

(iv) Is the gradient $\nabla \ell$ Lipschitz continuous? If so, can you find a simple data-independent value for its constant of Lipschitz continuity L? (Recall that a function F is L-Lipschitz continuous if $\|F(u) - F(v)\| \leq L\|u - v\|$ for any $u \in \mathbb{R}^d$.)

1(c). Write out the expression for the update step of gradient descent with learning rate α on this problem. You will need to derive the gradient to do this.

1(d). Now, suppose you wanted to use stochastic gradient descent (SGD) for this task to make learning more scalable. (For this problem set, just assume the baseline SGD algorithm with no minibatching, i.e. a batch size of 1.) Write out the expression for the update step of SGD with step size α on this problem. You will need to derive the gradient to do this.

1(e). Your classmate, Iago, claims that gradient descent for this task is guaranteed to converge asymptotically to some local minimum of the loss ℓ, as long as the step size is $\alpha = \frac{1}{4\lambda}$. Is Iago correct? If so, prove it. If not, find a counterexample.

1(f). Iago also claims that if gradient descent (i.e. batch size n, the whole training set) for this task is guaranteed to converge asymptotically to a local minimum of the loss ℓ for *some* fixed step size $\alpha > 0$, then *stochastic* gradient descent with that same step size α (and batch size 1) is also guaranteed to converge asymptotically to a local minimum of the loss ℓ with probability 1. Is Iago's second claim correct? If so, prove it. If not, find a counterexample.

Problem 2: Computational Costs of Learning Algorithms.

In this problem, we will explore the computational cost of solving the ERM task we saw in Problem 1 using Gradient descent and SGD. Each of the parts of this problem asks you to evaluate how many floating point operations are needed for a particular task. Note that you may need to use any of the following types of operations:

- Additions ($a + b$)
- Subtractions ($a - b$)
- Multiplications ($a \times b$)
- Divisions (a/b)

Besides, we may assume that operations such as $\log(\cdot)$, $\exp(\cdot)$, and $\text{sign}(\cdot)$ could cost one unit of floating-point operation. Be sure to show your work, including a break down of the floating-point operations used by type. Also note that your answer may depend on the problem size constants n and d. Note that there are multiple possible correct answers here, which differ depending on how accumulators are initialized in the implementations of the various operations: any answer with a good justification will be accepted.

2(a). Suppose that we have already trained a model h_w from the ERM task of Problem 1. We are going to produce and output a prediction for an example x using h_w . Suppose that we are going to do this on a CPU using ordinary floating-point arithmetic. How many total floating-point operations are required to compute this prediction, $h_w(x)$?

2(b). Now suppose that we want to know the empirical risk associated with a given weight vector. That is, we want to compute $\ell(w)$. How many total floating-point operations are required to compute this prediction, $\ell(w)$?

2(c). Now imagine that we are going to compute gradient descent for the Problem 1 task on a CPU using ordinary floating-point arithmetic. Using your answer of 1(c), how many total floating-point operations are required to compute one update step of gradient descent?

2(d). Now imagine that we are going to compute SGD (with batch size 1) for the Problem 1 task on a CPU using ordinary floating-point arithmetic. Using your answer of 1(d), how many total floating-point operations are required to compute one update step of SGD?

2(e). Now, let's look at a concrete task. Suppose that you are training a model where the training examples have dimension $d = 784$, and there are $n = 60000$ total training examples. For single-precision arithmetic, the H100 GPU accelerator has 60 teraFLOPS ($60 \cdot 10^{12}$ floating-point operations per second) maximum performance. Assume that you are using single-precision arithmetic to run gradient descent on this GPU. Using your answer from 2(c), what is the maximum number of iterations of gradient descent you could run in an hour on this device, assuming that you are bound only by FLOPS limitations?