1. **Empirical Risk Minimization.**

Consider a binary classification task where we want to train a classifier 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

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

To train this classifier given a dataset of n labeled examples $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$, we solve the *empirical risk minimization problem*

$$\min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \max(0, 1 - y_i x_i^T w) + \lambda \|w\|^2$$

(a) The empirical risk minimization problem is an example of a Soft-margin support vector machine learning model.

(b) Derive an expression for the gradient of the loss. More formally, if we define $f : \mathbb{R}^d \to \mathbb{R}$ to be the function

$$f(w) = \frac{1}{n} \sum_{i=1}^{n} \max(0, 1 - y_i x_i^T w) + \lambda \|w\|^2$$

find a formula for $\nabla f(w)$.

Solution.
We note that functions of the form $max(0, 1 - z)$ can be expressed as

$$max(0, 1 - z) = \begin{cases} 1 - z & z \leq 1 \\ 0 & z > 1 \end{cases} \tag{1}$$

and thus the derivative with respect to z can be computed piece wise to obtain

$$\frac{d}{dz} max(0, 1 - z) = \begin{cases} -z & z \leq 1 \\ 0 & z > 1 \end{cases} = \mathbb{1}(z \leq 1)(-z) \tag{2}$$

We also note that for $w \in \mathbb{R}^d$

$$\nabla \|w\|^2 = \nabla \left(\sum_{i=1}^{d} w_i^2 \right) = 2w$$

Using the reasoning above, we obtain

$$\nabla f(w) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}(y_i x_i^T w_i \leq 1)(-y_i x_i) + 2\lambda w$$

as the gradient of f. ■

(c) Answer each of the following of the following question

i. Is the objective f convex?

Solution.
f is convex if f' is monotonically non-decreasing of f'' is non-negative. Assuming $\lambda \geq 0$ and taking $\nabla^2 f$ we get

$$\nabla^2 f(w) = 2\lambda I_{d\times d}.$$

Since this is a diagonal matrix with all entries 2λ it is positive semi-definite by assumption on λ. Thus f is convex. ■

ii. Is f strongly convex? If so, what is its constant of strong convexity?

Solution.
A function f is strongly convex if

$$u^T \nabla^2 f(x) u \geq \mu ||u||^2$$

for some $\mu > 0$. We now note that $\nabla^2 f(x) = 2\lambda I_{d\times d}$, so

$$2\lambda u^T u = 2\lambda \|u\|^2$$

It follows that $\mu = 2\lambda$ ■

iii. Is the gradient ∇f Lipschitz continuous? If so, what is its constant of Lipschitz continuity?

Solution.
No, since the gradient of f is not itself a continuous function. ■

(d) Write out the expression for the update step of gradient descent on this problem, using the formula derived in 1(c).

Solution.
The update step of gradient descent is the following:

$$\begin{aligned} w_{t+1} &= w_t - \alpha_t \cdot \nabla f(w_t) \\ &= w_t - \alpha_t \cdot \left(\frac{1}{n} \sum_{i=1}^{n} \mathbb{1}(y_i x_i^T w_i \leq 1)(-y_i x_i) + 2\lambda w_t \right) \end{aligned}$$

where α_t is the step size. ■

(e) Write out the expression for the update step of SGD on this problem, using the formula you derived in 1(c).

Solution.
The update rule is the following:

$$\begin{aligned} w_{t+1} &= w_t - \alpha_t \cdot \nabla f_{\tilde{i}}(w_t) \\ &= w_t - \alpha_t \cdot \left(\mathbb{1}(y_{\tilde{i}} x_{\tilde{i}}^T w_{\tilde{i}} \leq 1)(-y_{\tilde{i}} x_{\tilde{i}}) + 2\lambda w_t \right) \end{aligned}$$

where we pick sample $\tilde{i}$ uniformly at random from the training examples and α_t is the step size. ■

2. **Computational Costs of Learning Algorithms**

Note that the following operations may need to be used

- Additions (a+b)
- Subtractions (a-b)
- Multiplications (a× b)
- Division (a/b)
- Comparisons (a=b, a≥b, etc)

Show a break down of the floating-point operations used by type. Answers may depend on the problem size constants n and d.

(a) Suppose that we have already trained a classifier h_w from the ERM task of Problem 1. We are going to classify an example x using h_w. Suppose that we are going to do this on a CPU using ordinary floating point arithmetic. How many floating point operations are required to compute prediction $h_w(x)$?

Solution.
To classify an example $x \in \mathbb{R}^d$, we take the dot product of x and w, $\langle x, w \rangle$, and then compare the result to 0.
Taking the dot product between two vectors in $\mathbb{R}^d$ involves taking d multiplications and $d-1$ additions. Classifying the result is a comparison with zero. That gives

$$\begin{aligned} d &\quad \text{(multiplications)} \\ d-1 &\quad \text{(additions)} \\ 1 &\quad \text{(comparison)} \end{aligned}$$

for a total of $2d$ flops. ■

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

Solution.
Recall that

$$f(w) = \frac{1}{n} \sum_{i=1}^{n} \max(0, 1 - y_i x_i^T w) + \lambda \|w\|^2 .$$

First consider the terms

$$\max(0, 1 - y_i x_i^T w) + \lambda \|w\|^2 .$$

Analyzing the first part, we see that we must calculate $1 - y_i x_i^T w$ and compare it to 0. The calculation of $1 - y_i x_i^T w$ involves the inner product between two d dimensional vectors, x_i, w, a multiplication with the label y_i, and a subtraction from 1. In total, this is $2d + 2$ flops, where we have

$$\begin{aligned} d+1 &\quad \text{(multiplications)} \\ d-1 &\quad \text{(additions)} \\ 1 &\quad \text{(subtraction)} \\ 1 &\quad \text{(comparison)} \end{aligned}$$

We now introduce the summation and division by n **to the first part**, we now have

$$\begin{aligned} n(d+1) &\quad \text{(multiplications)} \\ n(d-1) + (n-1) &\quad \text{(additions)} \\ n &\quad \text{(subtraction)} \\ n &\quad \text{(comparison)} \\ 1 &\quad \text{(division)} \end{aligned}$$

where all the results from prior were multiplied by n and then we had to account for the $n-1$ additions the summation produces as well as the division by n. The total flop count is now $2nd + 3n$.
Now, analyzing the second term, $\lambda\|w\|^2$, involves taking the inner product of w with itself and then multiplying by λ. In total, this is $2d$ flops where we have

$$\begin{aligned} d+1 &\quad \text{(multiplications)} \\ d-1 &\quad \text{(additions)} \end{aligned}$$

Adding these two together introduces one more addition as well as the components of each part, for a total of $2nd + 2d + 3n + 1$ flops where we have (in total),

$$\begin{aligned} n(d+1) + (d+1) &\quad \text{(multiplications)} \\ n(d-1) + (n-1) + (d-1) + 1 &\quad \text{(additions)} \\ n &\quad \text{(subtraction)} \\ n &\quad \text{(comparison)} \\ 1 &\quad \text{(division)} \end{aligned}$$

Thank you for coming to my TED talk. ■

(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(d), how many total floating-point operations are required to compute one update step of gradient descent?

Solution.
We found in part 1d the update rule to be:

$$\begin{aligned} w_{t+1} &= w_t - \alpha_t \cdot \nabla f(w_t) \\ &= w_t - \alpha_t \cdot \left(\frac{1}{n} \sum_{i=1}^{n} \mathbb{1}(y_i x_i^T w_i \leq 1)(-y_i x_i) + 2\lambda w_t \right) \end{aligned}$$

We simplify this in an attempt to be more efficient by combining like terms.

$$w_{t+1} = w_t(1 - 2\lambda\alpha_t) + \frac{\alpha_t}{n} \sum_{i=1}^{n} \mathbb{1}(y_i x_i^T w_i \leq 1)(y_i x_i)$$

The first term, $w_t(1 - 2\lambda\alpha_t)$, has a total of $d + 3 flops$ where we have

$$\begin{aligned} d + 2 &\quad \text{(multiplications)} \\ 1 &\quad \text{(subtraction)} \end{aligned}$$

The term inside the summation, $\mathbb{1}(y_i x_i^T w_i \leq 1)(y_i x_i)$, has a total of of $3d + 2$ flops where we have

$$\begin{aligned} 2d + 2 &\quad \text{(multiplications)} \\ d - 1 &\quad \text{(additions)} \\ 1 &\quad \text{(comparison)} \end{aligned}$$

We now introduce the summation, which multiplies the immediate above result by n and introduces $(n-1)d$ additions, giving a total of $n(3d+2)+(n-1)d$ flops where we have

$$\begin{aligned} n(2d + 2) &\quad \text{(multiplications)} \\ n(d - 1) + (n - 1)d &\quad \text{(additions)} \\ n &\quad \text{(comparisons)} \end{aligned}$$

and finally we compute the division, $\frac{\alpha_t}{n}$, and multiply the result with the above and add that to the first term to obtain a total of $4nd + 2n + 2d + 4$ flops where we have

$$\begin{aligned} n(2d + 2) + d + (d + 2) &\quad \text{(multiplications)} \\ n(d - 1) + (n - 1)d + d &\quad \text{(additions)} \\ n &\quad \text{(comparisons)} \\ 1 &\quad \text{(subtraction)} \\ 1 &\quad \text{(division)} \end{aligned}$$

■

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

Solution.
SGD has update rule

$$\begin{aligned} w_{t+1} &= w_t - \alpha_t \cdot \nabla f(w_t) \\ &= w_t - \alpha_t \cdot \left(\mathbb{1}(y_{\tilde{i}} x_{\tilde{i}}^T w_{\tilde{i}} \leq 1)(-y_{\tilde{i}} x_{\tilde{i}}) + 2\lambda w_t \right) \\ &= w_t(1 - 2\lambda\alpha_t) + \mathbb{1}(y_{\tilde{i}} x_{\tilde{i}}^T w_{\tilde{i}} \leq 1)(\alpha_t y_{\tilde{i}} x_{\tilde{i}}) \end{aligned}$$

where $\tilde{i}$ is chosen uniformly at random between 1 and n inclusive. Considering the choosing of the sample as a free operation, we split the analysis into two parts. The first term, $w_t(1 - 2\lambda\alpha_t)$, has a total of $d + 3$ flops where we have

$d + 2$	(multiplications)
1	(subtraction)

The second term, $\mathbb{1}(y_{\tilde{i}} x_{\tilde{i}}^T w_{\tilde{i}} \leq 1)(\alpha_t y_{\tilde{i}} x_{\tilde{i}})$, has a total of $3d + 3$ flops where we have

$2d + 3$	(multiplications)
$d - 1$	(additions)
1	(comparison)

Adding the first term with the second, it is clear we also have an additional d additions giving a total of $5d + 6$ flops where we have

$3d + 5$	(multiplications)
$2d - 1$	(additions)
1	(subtraction)
1	(comparison)

■

(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 NVIDIA V100 GPU has 14 teraFLOPS ($14 \cdot 10^{12}$ 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?

Solution.

We have a total of

$$4nd + 2n + 2d + 4$$

where $n = 60000$ and $d = 784$. This is a total of 188281572 flops. Each iteration will take:

$$\frac{188281572}{14 \cdot 10^{12}} \approx 1.34 \times 10^{-5} \text{ seconds}$$

In an hour, this is about 2.67×10^8 iterations. ■