CMPUT 466 Assignment 3

Arun Woosaree

October 4, 2021

Problem 1

$$J = \sum_{m=1}^{M} \left(\sum_{i=0}^{d} w_i x_i^{(m)} - t^{(m)} \right)^2 + \sum_{i=0}^{d} w_i^2$$

can be written in matrix form as:

$$J = ||\mathbf{X}\mathbf{w} - t||_2^2 + \mathbf{w}^\top \mathbf{w}$$

$$J = \mathbf{w}^\top \mathbf{X}^\top \mathbf{X}\mathbf{w} - \mathbf{w}^\top \mathbf{X}^\top \mathbf{t} - \mathbf{t}^\top \mathbf{X}\mathbf{w} + \mathbf{t}^\top \mathbf{t} + \mathbf{w}^\top \mathbf{w}$$

$$J = \mathbf{w}^\top \mathbf{X}^\top \mathbf{X}\mathbf{w} - 2\mathbf{t}^\top \mathbf{X}\mathbf{w} + \mathbf{t}^\top \mathbf{t} + \mathbf{w}^\top \mathbf{w}$$

$$\nabla J(\mathbf{w}) = 2\mathbf{X}^\top \mathbf{X}\mathbf{w} - 2(\mathbf{t}^\top \mathbf{X})^\top + 2\mathbf{w}$$

$$\nabla J(\mathbf{w}) = 2\mathbf{X}^\top \mathbf{X}\mathbf{w} - 2\mathbf{X}^\top \mathbf{t} + 2\mathbf{w}$$

$$\nabla^2 J(\mathbf{w}) = 2\mathbf{X}^\top \mathbf{X} + 2 \geq 0$$

Because $\mathbf{X}^\top \mathbf{X} \in \mathbb{R}$ and it's ≥ 0, $\nabla^2 J(\mathbf{w}) \geq 0$

So, J is convex in $\mathbf{w}$.

Because J is convex in $\mathbf{w}$, we can set its first derivative to $\mathbf{0}$ to find the global minimum:

$$\mathbf{0} = 2\mathbf{X}^\top \mathbf{X}\mathbf{w} - 2\mathbf{X}^\top \mathbf{t} + 2\mathbf{w}$$
$$\mathbf{X}^\top \mathbf{t} = \mathbf{X}^\top \mathbf{X}\mathbf{w} + \mathbf{w}$$
$$\mathbf{X}^\top \mathbf{t} = \mathbf{X}^\top \mathbf{X}\mathbf{w} + \mathbf{w}$$
$$\mathbf{X}^\top \mathbf{t} = (\mathbf{X}^\top \mathbf{X} + I)\mathbf{w}$$
$$\boxed{\mathbf{w} = (\mathbf{X}^\top \mathbf{X} + I)^{-1} \mathbf{X}^\top t}$$

Where I is the identity matrix.

Note: this works when:

- $\mathbf{X}^\top \mathbf{X}$ is invertible (full rank)
- cannot have duplicate features, can use pseudo inverse instead
- need more samples than features. If not, do feature selection

Problem 2

Let's pick the loss function $J = w^2, w \in \mathbb{R}$. We know this is a convex function because its second derivative is two, which is greater than 0.

Let's pick an annealing scheduler such that $\alpha^{(i)} = 0.1\alpha^{(i-1)}$, and $\alpha^{(0)} = 1$.

Let's say that $w^{(0)} = 10$, so $J(w)$ starts at 100. After just a few iterations in a python program, we see that w converges to about 7.822581297348168, which is far from the optimum, where $w = 0$. This happens because α decreases so fast, that this is practically the lowest it can go, even with infinite iterations. However, this does not necessarily prove convergence.

We can prove for example that the following series converges:

$$\sum_{i=1}^{\infty} \frac{1}{10^i}$$

This is the pattern that the annealing scheduler for α follows. If we prove that the sum is bounded, then that helps to convince us that the gradient descent algorithm would decrease infinitely, but not to the optimum.

The partial sum formula for this series is:

$$\frac{1}{9} \times \frac{10^n - 1}{10^n}$$

We can take the limit as $n \to \infty$ to find where the sum converges:

$$\lim_{n\to\infty} \frac{1}{9} \times \frac{10^n - 1}{10^n}$$

$$\lim_{n\to\infty} \frac{1}{9} \times \left(\frac{10^n}{10^n} - \frac{1}{10^n}\right)$$

$$\lim_{n\to\infty} \frac{1}{9} \times \left(1 - \frac{1}{10^n}\right)$$

$$= \frac{1}{9}$$

, which is $< \infty$

Now, if at each iteration, α is multiplied by a constant value k, the total sum would be: $k\frac{1}{9}$.

This value is still $< \infty$.

Actually, in the gradient descent algorithm, the new w is updated as follows in each iteration:

$$\alpha = 0.1\alpha$$
$$w^{(i)} = w^{(i-1)} - \alpha(2 * w^{(i-1)})$$

Because the loss function selected is convex, and we chose $w^{(0)} = 10$, the gradient descent algorithm will always subtract an amount equal to $\alpha \times 2w^{(i-1)}$ in this example. Another result of the loss function being convex and the initial value we chose for w is that $w^{(i)} < w^{(i-1)}$ because we chose a small enough α, and w must decrease to minimize J. If we continue the gradient descent algorithm indefinitely, the total amount subtracted will be some amount less than $2 \times 10 \times \frac{1}{9}$. (Because 10 is the largest value w will ever be in this example.)

i.e., the value that the gradient descent algorithm will converge to will be some value greater than $10 - \frac{20}{9}$, or some value greater than $7.777\ldots$

In practice, this value appears to be about 7.82 (not proven). Either way, we have shown an example where the gradient descent algorithm converges to a value that is not the minimum. Even after infinite iterations, w will never be less than $10 - \frac{20}{9}$, which is far from the optimal $w = 0$. $\square$