Machine Learning: HW1

Brent Garey

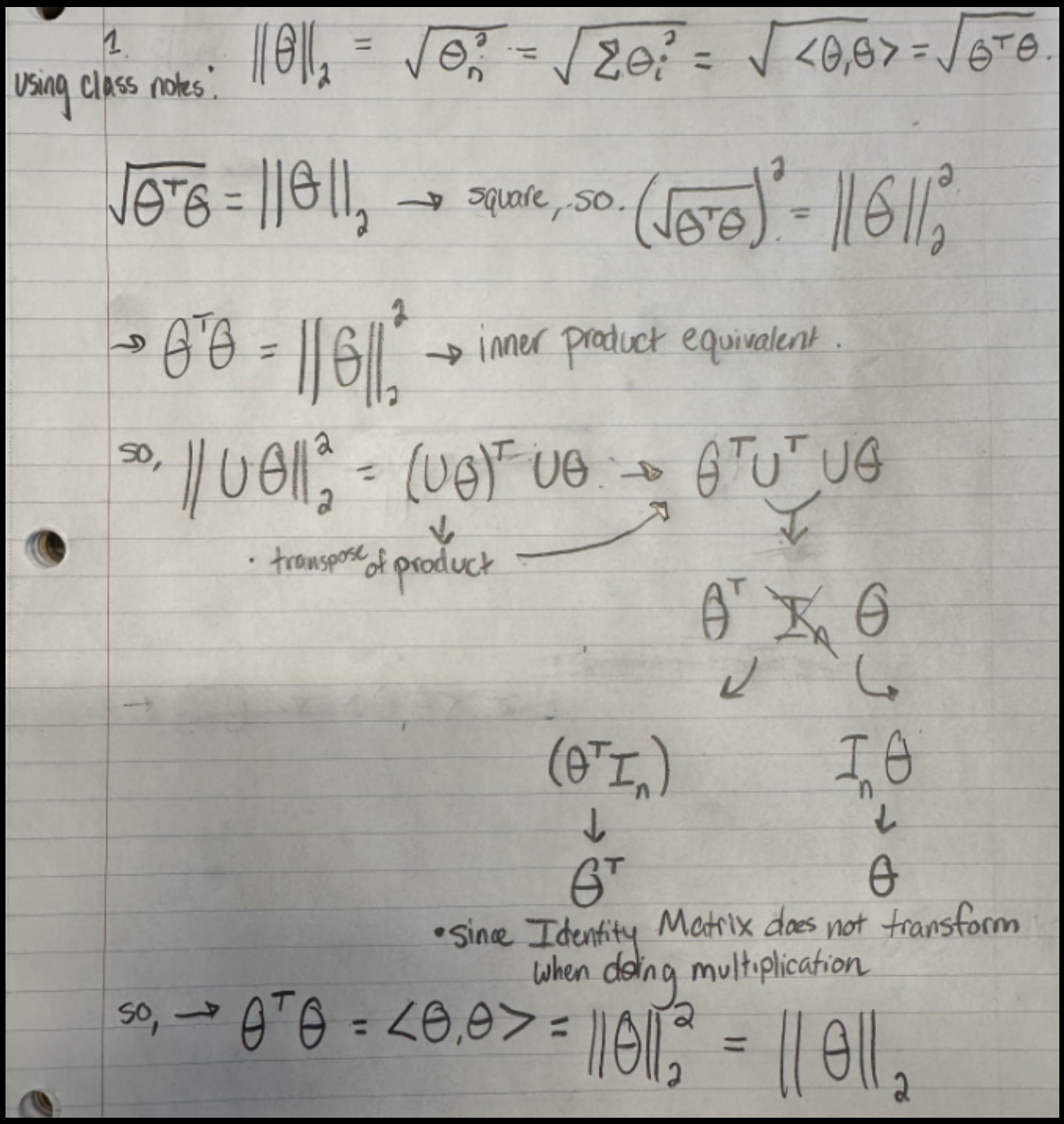
February 3rd, 2025

Contents

1 1.

Show transformation of a vector by an orthonormal matrix (U) preserves the norm (magnetude) of the vector.

answer:



2 2.

2. 1* $\frac{\partial b^T X\theta}{\partial \theta} = X^T b$

1. $\frac{\partial b^T X\theta}{\partial \theta} = \frac{\partial (b^T)(X\theta)}{\partial \theta} = \frac{\partial (X\theta)^T (b)}{\partial \theta}$

• since in class. $\frac{\partial (a^T x)}{\partial x} = \frac{\partial (x^T a)}{\partial x}$

so, $\frac{\partial (X\theta)^T (b)}{\partial \theta} = \frac{\partial\ \theta^T X^T (b)}{\partial \theta}$

↳ multiplicative transpose property.

in class: $\frac{\partial (x^T a)}{\partial x} = a$, so:

$$\frac{\partial \theta^T X^T b}{\partial \theta} = X^T b$$

2. * $\frac{\partial \|X\theta - b\|_2^2}{\partial \theta} = 2X^T(X\theta - b)$

- set $\ell(\theta) = \|X\theta - b\|_2^2 = \left\| \begin{pmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_n^T \end{pmatrix} \begin{pmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{pmatrix} - \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix} \right\|_2^2$

transpose matrix X to get all features of a single x_n

$$\left\| \begin{pmatrix} x_1^T\theta_1 \\ x_2^T\theta_2 \\ \vdots \\ x_n^T\theta_n \end{pmatrix} - \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix} \right\|_2^2 = \left\| \begin{pmatrix} \theta_1^T x_1 \\ \theta_2^T x_2 \\ \vdots \\ \theta_n^T x_n \end{pmatrix} - \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix} \right\|_2^2$$

Since $\frac{\partial(x^T\theta)}{\partial\theta} = \frac{\partial(\theta^T x)}{\partial\theta}$, transpose property rearranges order.

→ we now have vector operations, simplify.

$$\left\| \begin{pmatrix} \theta_1^T x_1 - b_1 \\ \theta_2^T x_2 - b_2 \\ \vdots \\ \theta_n^T x_n - b_n \end{pmatrix} \right\|_2^2 \rightarrow \text{L2 norm definition "sum of squares"} \sum_{i=1}^{N} (\theta^T x_i - b_i)^2$$

$$\frac{\partial}{\partial \theta} \sum_{i=1}^{N} (\theta^T x_i - b_i)^2 =$$

take derivative using chain rule.

$$\rightarrow n(f(x))^{n-1} \cdot f'(x)$$

$f'(x) \rightarrow \frac{\partial}{\partial \theta}$

plug in: $2(\theta^T x_i - b_i) \cdot x_i$

- since x_i is features of index i of x, we can once again claim $x_i = X^T$

Simplify. $2x_i(\theta^T x_i - b) = 2X^T(\theta^T X^T - b)$

transpose multiplicative property.

$$= 2X^T((\theta x)^T - b) = 2X^T(X\theta - b)$$

3 3.

3. Gradient Descent: $f(x_1, x_2) = (x_1 - x_2)^2 + (x_1 - 1)^2 + (x_2 - 1)^2$

partial Derivatives: $\begin{pmatrix} \partial f/\partial x_1 \\ \partial f/\partial x_2 \end{pmatrix} = \begin{pmatrix} 4x_1 - 2x_2 - 2 \\ -2x_1 + 4x_2 - 2 \end{pmatrix}$

1. Use learning rate $\rho = 1$.

iteration 1: $f(x_1, x_2) = (0,0) = \theta^{(0)}$

$\begin{pmatrix} 4(0) - 2(0) - 2 \\ -2(0) + 4(0) - 2 \end{pmatrix} = \begin{pmatrix} -2 \\ -2 \end{pmatrix}$ → gradient → not 0, so has NOT converged.

↳ update position using step size & gradient:

$(\theta^{(0)} - \rho(-2), \theta^{(0)} - \rho(-2)) \rightarrow (0 - 1(-2), 0 - 1(-2))$

$= (2, 2)$ → updated position after 1 iteration.

iteration 2: $f(x_1, x_2) = (2,2) = \theta^{(1)}$

$\begin{pmatrix} 4(2) - 2(2) - 2 \\ -2(2) + 4(2) - 2 \end{pmatrix} = \begin{pmatrix} 2 \\ 2 \end{pmatrix}$ → gradient
↳ not converged!

↳ update: $(\theta^{(1)} - 1(2), \theta^{(1)} - 1(2)) = (0,0)$ = updated position

iteration 3: $f(x_1, x_2) = (0,0) = \theta^{(2)}$

$\begin{pmatrix} 4(0) - 2(0) - 2 \\ -2(0) + 4(0) - 2 \end{pmatrix} = \begin{pmatrix} -2 \\ -2 \end{pmatrix}$ → gradient
↳ Not converged!

update: $(\theta^{(2)} - 1(-2), \theta^{(2)} - 1(-2)) = (2,2)$

3 pt 2: $\rho = 0.5$,

iteration 1: $(x_1, x_2) = (0,0) = \theta^0$

$\begin{pmatrix} 4(0)-2(0)-2 \\ -2(0)+4(0)-2 \end{pmatrix} = \begin{pmatrix} -2 \\ -2 \end{pmatrix}$ → not converged.

update position: $(0-0.5(-2), 0-0.5(-2)) = \underline{(1,1)}$

iteration 2: $(x_1, x_2) = (1,1) = \theta^1$

$\begin{pmatrix} 4(1)-2(1)-2 \\ -2(1)+4(1)-2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$ → converged! (make no adjus

update position yields: $(1-0.5(0), 1-0.5(0)) = \underline{(1,1)}$

iteration 3: $(x_1, x_2) = (1,1) = \theta^2$

→ gradient $= \begin{pmatrix} 0 \\ 0 \end{pmatrix}$

update: $(1-0.5(0), 1-0.5(0)) = \underline{(1,1)}$.

3 pt 3: $\rho = 2$

iteration 1: $(x_1, x_2) = (0,0) = \theta^0$

$\begin{pmatrix} 4(0)-2(0)-2 \\ -2(0)+4(0)-2 \end{pmatrix} = \begin{pmatrix} -2 \\ -2 \end{pmatrix}$ → not converged.

update position: $(0-2(-2), 0-2(-2)) = \underline{(4,4)} = \theta^1$.

iteration 2: $(x_1, x_2) = (4,4) = \theta^1$

$\begin{pmatrix} 4(4)-2(4)-2 \\ -2(4)+4(4)-2 \end{pmatrix} = \begin{pmatrix} 6 \\ 6 \end{pmatrix}$ → not converged.

update position: $(4-2(6), 4-2(6)) = \underline{(-8,-8)} = \theta^2$

iteration 3: $(x_1, x_2) = (-8,-8) = \theta^2$

$\begin{pmatrix} 4(-8)-2(-8)-2 \\ -2(-8)+4(-8)-2 \end{pmatrix} = \begin{pmatrix} -18 \\ -18 \end{pmatrix}$ → not converged.

update position: $(-8-2(-18), -8-2(-18)) = \underline{(28,28)} = \theta^3$.

Analysis:

1. Learning rate = 1:

Gradient descent has NOT converged in 3 iterations. With this step size, GD has reached a local minima where it will constantly swap between (-2,-2) and (2,2) every other iteration, NEVER converging. The step size is slightly too large, where the ideal convergence criteria lays somewhere between the step size = 1 and 0 (0.5) since 1 step in either direction steps OVER the optimal convergence criteria of gradient = 0. The next iteration will always take the same step back OVER, resulting in a back and forth.

2. Learning rate = 0.5:

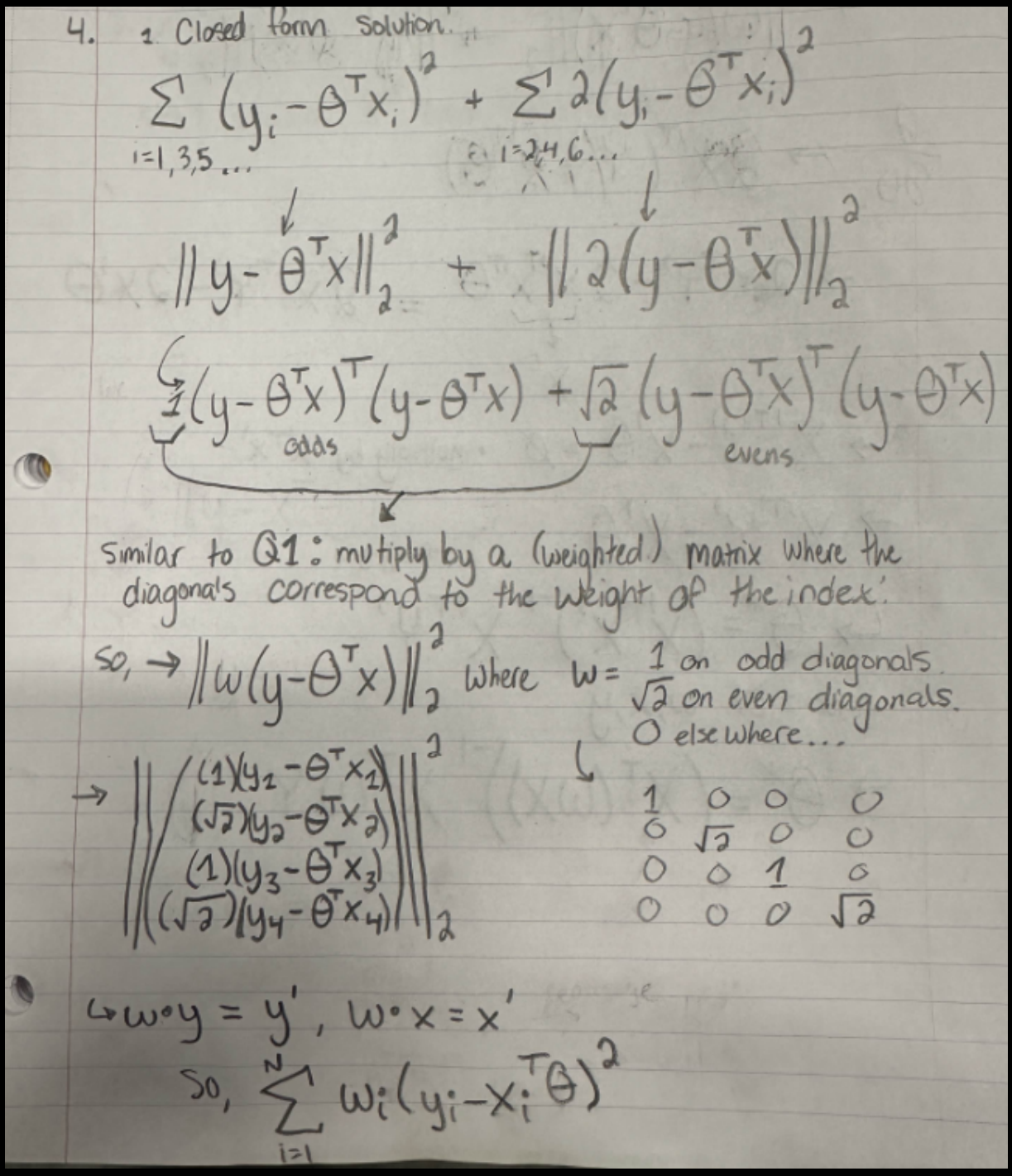
GD has converged in 2 iterations because the gradient becomes (0,0) on iteration 2. I wrote down iteration 3 just in case, but from iteration 3 and beyond the gradient will always evaluate to (0,0), so GD will never move the point from (1,1).

3. Learning rate = 2:

GD has NOT converged in 3 iterations. The step size is too large, so we achieve an oscillating pattern that will forever increase the absolute values of the gradient, with every other iteration changing signs from positive to negative. This will never converge.

4 4.

1. Closed form solution:



$\theta^* = (X^T(WX))^{-1}X^TWy$ (on the bottom of next page)

now, $\|(w(y-\theta^T x)\|_2^2 \xrightarrow{w\to y',\ x'\text{ to simplify notation}} \|y'-\theta^T x'\|_2^2$

$\frac{\partial}{\partial\theta} \rightarrow 2x'^T(y'-x'^T\theta)$

$\rightarrow 2x'^T y' - 2\underbrace{x'^T x'^T}_{x'}\theta = 2x'^T y' - 2x'\theta$

$\rightarrow x'^T y' - x'\theta = \emptyset$ · multiply by $x'^T x'$

$\rightarrow x'^T y' = x'\theta$

$\hookrightarrow \theta^* = (x'^T x')^{-1} x'^T y'$

in terms of w, x, y

$\rightarrow \theta^* = (X^T(wX))^{-1} X^T w Y$

2. Steps for gradient descent:

Step 1: Initialize θ^0 either randomly, or as 0's.

Step 2: Determine if index is odd, or even (if % 2 == 0) to determine weight of gradient in next step.

Step 3: Calculate gradient $(\nabla_\theta) = 2X^T W(X\theta - y)$:

Check for convergence! (gradient either 0 or some arbitrarily small number)

Step 4: Use gradient and some arbitrary step size to update position (find θ^{t+1})
$\theta^{t+1} = \theta^t - \alpha\nabla_\theta(\theta)$

5 5.