

CSC413: Homework 1

Tianyu Du (1003801647)

2020/02/06 at 16:02:00

1 Hard-Coding Networks

1.1 Verify Sort

Soln. The first layer performs pairwise comparison to construct indicators $\mathbb{1}\{x_1 \leq x_2\}$, $\mathbb{1}\{x_2 \leq x_3\}$, and $\mathbb{1}\{x_3 \leq x_4\}$. The second layer performs an `all()` operation on indicators from the previous layer.

$$\mathbf{W}^{(1)} = \begin{pmatrix} -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 \end{pmatrix} \quad (1.1)$$

$$\mathbf{b}^{(1)} = \begin{pmatrix} 0 & 0 & 0 \end{pmatrix} \quad (1.2)$$

So that

$$\varphi(\mathbf{h}) = \varphi(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)}) = \varphi \begin{pmatrix} x_2 - x_1 \\ x_3 - x_2 \\ x_4 - x_3 \end{pmatrix} = \begin{pmatrix} \mathbb{1}\{x_2 \geq x_1\} \\ \mathbb{1}\{x_3 \geq x_2\} \\ \mathbb{1}\{x_4 \geq x_3\} \end{pmatrix} \quad (1.3)$$

$$\mathbf{w}^{(2)} = \begin{pmatrix} 1 & 1 & 1 \end{pmatrix} \quad (1.4)$$

$$b^{(2)} = -2.5 \quad (1.5)$$

Such that $y = 1$ if and only if all components of \mathbf{h} are ones, i.e., the list is sorted. ■

1.2 Perform Sort

Proof. For this section, I am implementing a feedforward neural network performing bubble sort. Note that given $\mathbf{x} = (x_1, x_2, x_3, x_4)$, the **swapping** operation can be conducted by multiplying \mathbf{x} by a matrix.

For instance, the following $\mathbf{W}^{(12)}$ swaps the first and second elements.

$$\underbrace{\begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}}_{\mathbf{W}^{(12)}} (x_1, x_2, x_3, x_4)^T = (x_2, x_1, x_3, x_4)^T \quad (1.6)$$

Let \mathbf{I}_4 denote the identity matrix. Define the following neural network:

$$\mathbf{h}_1 := \mathbb{1}\{x_1 \leq x_2\} \mathbf{I}_4 \mathbf{x} + \mathbb{1}\{x_1 > x_2\} \mathbf{W}^{(12)} \mathbf{x} \quad (1.7)$$

$$\mathbf{h}_2 = \mathbb{1}\{x_2 \leq x_3\} \mathbf{I}_4 \mathbf{h}_1 + \mathbb{1}\{x_2 > x_3\} \mathbf{W}^{(23)} \mathbf{h}_1 \quad (1.8)$$

$$\mathbf{h}_3 = \mathbb{1}\{x_3 \leq x_4\} \mathbf{I}_4 \mathbf{h}_2 + \mathbb{1}\{x_3 > x_4\} \mathbf{W}^{(34)} \mathbf{h}_2 \quad (1.9)$$

$$\mathbf{h}_4 = \mathbb{1}\{x_1 \leq x_2\} \mathbf{I}_4 \mathbf{h}_3 + \mathbb{1}\{x_1 > x_2\} \mathbf{W}^{(12)} \mathbf{h}_3 \quad (1.10)$$

$$\mathbf{h}_5 = \mathbb{1}\{x_2 \leq x_3\} \mathbf{I}_4 \mathbf{h}_4 + \mathbb{1}\{x_2 > x_3\} \mathbf{W}^{(23)} \mathbf{h}_4 \quad (1.11)$$

$$\mathbf{y}(\mathbf{x}) = \mathbb{1}\{x_1 \leq x_2\} \mathbf{I}_4 \mathbf{h}_5 + \mathbb{1}\{x_1 > x_2\} \mathbf{W}^{(12)} \mathbf{h}_5 \quad (1.12)$$

Note: x_i s in different equations have potentially different values. For instance, the x_2 in equations (1.7) and (1.8) are potentially different, in 1.7, it represents the second component of \mathbf{x} and in 1.8 it represents the second component of \mathbf{h}_1 .

The above neural network can also be implemented using a recurrent structure, in which hidden states record the number of pairwise comparisons made. ■

1.3 Universal Approximation Theorem

1.3.1

Soln. To avoid over-using of notations, let $\varphi(y) := \mathbb{1}\{y \geq 0\}$ denote the activation function.

$$n = 2 \quad (1.13)$$

$$\mathbf{W}_0 = (1, -1) \quad (1.14)$$

$$\mathbf{b}_0 = (-a, b) \quad (1.15)$$

$$\mathbf{W}_1 = (h, h) \quad (1.16)$$

$$\mathbf{b}_1 = -h \quad (1.17)$$

Justification:

$$\varphi(\mathbf{h}) = \varphi((x - a, b - x)) \quad (1.18)$$

$$= (\mathbb{1}\{x - a \geq 0\}, \mathbb{1}\{b - x \geq 0\}) \quad (1.19)$$

$$= (\mathbb{1}\{x \geq a\}, \mathbb{1}\{x \leq b\}) \quad (1.20)$$

$$\mathbf{W}_1 \varphi(\mathbf{h}) + \mathbf{b}_1 = \mathbb{1}\{x \geq a\} h + \mathbb{1}\{x \leq b\} h - h \quad (1.21)$$

$$= \mathbb{1}\{a \leq x \leq b\} h \quad (1.22)$$

■

1.3.2

Soln. Let $\delta \in (0, 1)$ denote the ratio parameter, a higher value of δ results in a finer approximation is. In this example, take $\delta = \frac{9}{10}$.

Without loss of generality, assume the region I on which function f is defined on to be symmetric across zero.

Let $I = [-1, 1]$, given f is symmetric, $f(-\delta) = f(\delta)$.

Define:

$$\hat{f}_1(x) = \hat{f}_0(x) + g(f(\delta), -\delta, \delta, x) \quad (1.23)$$

Note that

$$\|f - \hat{f}_1\| = \int_{-1}^1 |f(x) - \hat{f}_1(x)| \, dx \quad (1.24)$$

$$= \int_{-1}^{-\delta} |f(x)| \, dx + \int_{-\delta}^{\delta} |f(x) - \hat{f}_1(x)| \, dx + \int_{\delta}^1 |f(x)| \, dx \quad (1.25)$$

Given that $\forall x \in (-\delta, \delta)$, $f(x) > f(-\delta) = f(\delta) > 0$, it follows

$$\int_{-\delta}^{\delta} |f(x) - \hat{f}_1(x)| \, dx = \int_{-\delta}^{\delta} f(x) - \hat{f}_1(x) \, dx \quad (1.26)$$

$$= \int_{-\delta}^{\delta} f(x) \, dx - \int_{-\delta}^{\delta} \hat{f}_1(x) \, dx \quad (1.27)$$

Also, $\int_{-\delta}^{\delta} \hat{f}_1(x) \, dx > 0$ provided $\delta \neq 0$. Therefore,

$$\int_{-\delta}^{\delta} |f(x) - \hat{f}_1(x)| \, dx < \int_{-\delta}^{\delta} f(x) \, dx \quad (1.28)$$

$$= \int_{-\delta}^{\delta} |f(x)| \, dx \quad (1.29)$$

Therefore,

$$\|f - \hat{f}_1\| = \int_{-1}^{-\delta} |f(x)| \, dx + \int_{-\delta}^{\delta} |f(x) - \hat{f}_1(x)| \, dx + \int_{\delta}^1 |f(x)| \, dx \quad (1.30)$$

$$< \int_{-1}^{-\delta} |f(x)| \, dx + \int_{-\delta}^{\delta} |f(x)| \, dx + \int_{\delta}^1 |f(x)| \, dx \quad (1.31)$$

$$= \int_{-1}^1 |f(x) - 0| \, dx \quad (1.32)$$

$$= \int_{-1}^1 |f(x) - \hat{f}_0(x)| \, dx \quad (1.33)$$

$$= \|f(x) - \hat{f}_0(x)\| \quad (1.34)$$

Therefore,

$$\|f(x) - \hat{f}_1(x)\| < \|f(x) - \hat{f}_0(x)\| \quad (1.35)$$

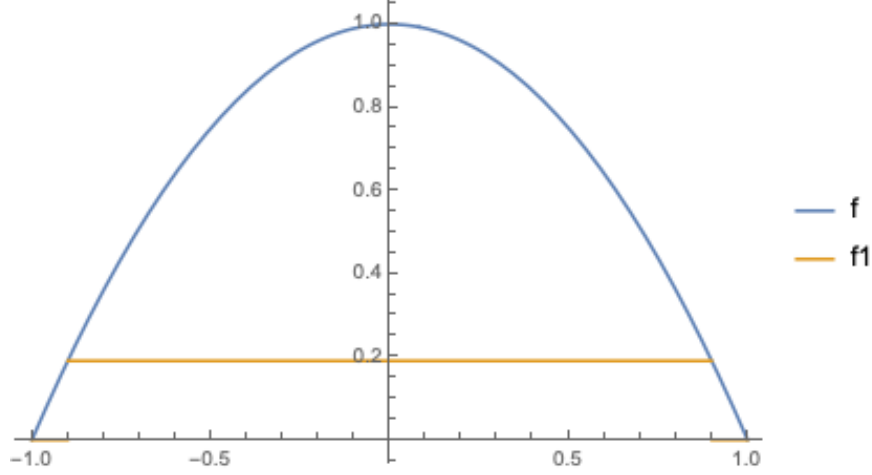


Figure 1.1: Approximated Result

■

1.3.3

Soln. **Algorithm:**

- (i) Divide $I = [-1, 1]$ into $N + 2$ sub-intervals with equal length, such that

$$I_i := \left[-1 + \frac{i}{N+2}, -1 + \frac{i+1}{N+2} \right] \quad \forall i \in \{1, 2, \dots, N\} \quad (1.36)$$

Note that the first and last sub-intervals are not used to construct g_i .

- (ii) For each i , define

$$h_i := \min_{x \in I_i} f(x) \quad (1.37)$$

$$a_i := -1 + \frac{i}{N+2} \quad (1.38)$$

$$b_i := -1 + \frac{i+1}{N+2} \quad (1.39)$$

Because $f(x) \geq 0 \quad \forall x \in I$.

By the definition of $g_i(x)$, it can be shown that¹

$$f(x) \geq f_i(x) \quad \forall i \in \{1, 2, \dots, N\} \quad \forall x \in \bigcup_{i=1}^N (a_i, b_i) \quad (1.40)$$

Further,

$$f(x) = f_i(x) \quad \forall i \in \{1, 2, \dots, N\} \quad \forall x \in \left[-1, -1 + \frac{1}{N+2} \right) \cup \left(1 - \frac{1}{N+2}, 1 \right] \quad (1.41)$$

¹I am excluding those boundary points between consecutive sub-intervals, because at those points, the value of f_i spikes due to duplicate counts of indicator functions. However, while doing integral, this does not matter as the set of boundary points has measure zero.

Define

$$\mathcal{K} := \left[-1, -1 + \frac{1}{N+2}\right) \cup \left(1 - \frac{1}{N+2}, 1\right] \cup \left(\bigcup_{i=1}^N (a_i, b_i)\right) \quad (1.42)$$

Note that the set $I \setminus \mathcal{K}$ consists of all boundary points between consecutive sub-intervals. There are only finitely many such points, therefore $I \setminus \mathcal{K}$ has measure zero, and

$$\int_I |f(x) - \hat{f}_i(x)| \, dx = \int_{\mathcal{K}} |f(x) - \hat{f}_i(x)| \, dx \quad (1.43)$$

And I've shown that for every i and every $x \in \mathcal{K}$, $f(x) \geq \hat{f}_i(x)$. Consequently,

$$\int_I |f(x) - \hat{f}_i(x)| \, dx = \int_{\mathcal{K}} |f(x) - \hat{f}_i(x)| \, dx \text{ (removing measure zero set.)} \quad (1.44)$$

$$= \int_{\mathcal{K}} f(x) - \hat{f}_i(x) \, dx \quad (1.45)$$

$$= \int_I f(x) - \hat{f}_i(x) \, dx \text{ (adding back the measure zero set.) } (\dagger) \quad (1.46)$$

Define $\hat{f}_0(x) = 0$ and let $i \in \{1, 2, \dots, N\}$,

$$\|f - \hat{f}_{i+1}\| = \int_{-1}^1 |f(x) - \hat{f}_{i+1}(x)| \, dx \quad (1.47)$$

$$= \int_{-1}^1 f(x) - \hat{f}_{i+1}(x) \, dx \text{ by } (\dagger) \quad (1.48)$$

$$= \int_{-1}^{-1+\frac{i+1}{N}} f(x) - \hat{f}_{i+1}(x) \, dx + \int_{-1+\frac{i+1}{N}}^{-1+\frac{i+2}{N}} f(x) - \hat{f}_{i+1}(x) \, dx + \int_{-1+\frac{i+2}{N}}^1 f(x) - \hat{f}_{i+1}(x) \, dx \quad (1.49)$$

Further, by construction, $\hat{f}_{i+1}(x) = \hat{f}_i(x) \, \forall x \notin [a_{i+1}, b_{i+1}]$. Therefore,

$$\|f - \hat{f}_{i+1}\| = \int_{-1}^{a_{i+1}} f(x) - \hat{f}_i(x) \, dx + \int_{a_{i+1}}^{b_{i+1}} f(x) - \hat{f}_{i+1}(x) \, dx + \int_{b_{i+1}}^1 f(x) - \hat{f}_i(x) \, dx \quad (1.50)$$

$$= \int_{-1}^{a_{i+1}} f(x) - \hat{f}_i(x) \, dx + \int_{a_{i+1}}^{b_{i+1}} f(x) - \hat{f}_i(x) - g(h_{i+1}, a_{i+1}, b_{i+1}, x) \, dx + \int_{b_{i+1}}^1 f(x) - \hat{f}_i(x) \, dx \quad (1.51)$$

$$= \int_{-1}^{a_{i+1}} f(x) - \hat{f}_i(x) \, dx + \int_{a_{i+1}}^{b_{i+1}} f(x) - \hat{f}_i(x) \, dx + \int_{b_{i+1}}^1 f(x) - \hat{f}_i(x) \, dx - \int_{a_{i+1}}^{b_{i+1}} g(h_{i+1}, a_{i+1}, b_{i+1}, x) \, dx \quad (1.52)$$

$$= \int_{-1}^1 f(x) - \hat{f}_i(x) \, dx - \int_{a_{i+1}}^{b_{i+1}} g(h_{i+1}, a_{i+1}, b_{i+1}, x) \, dx \quad (1.53)$$

$$= \|f - \hat{f}_i\| - \int_{a_{i+1}}^{b_{i+1}} g(h_{i+1}, a_{i+1}, b_{i+1}, x) \, dx \quad (1.54)$$

Note that for every i , for every $x \in [a_i, b_i]$, $g(h_i, a_i, b_i, x) > 0$. Therefore, $\int_{a_i}^{b_i} g(h_i, a_i, b_i, x) dx > 0$. Hence,

$$\|f - \hat{f}_{i+1}\| = \|f - \hat{f}_i\| - \int_{a_{i+1}}^{b_{i+1}} g(h_{i+1}, a_{i+1}, b_{i+1}, x) dx \quad (1.55)$$

$$> \|f - \hat{f}_i\| \quad (1.56)$$

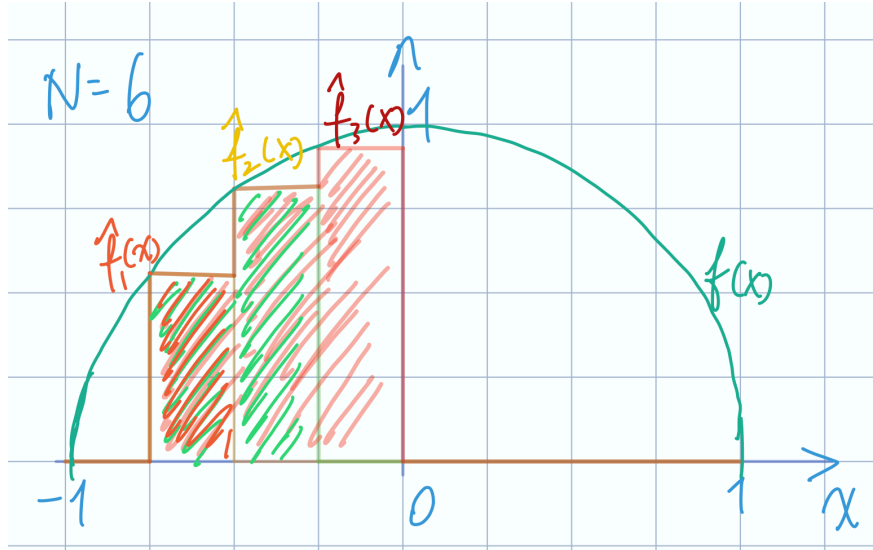


Figure 1.2: Approximated Results for $N = 6$

1.3.4

Soln. Not required.

2 Backprop

2.1 Computational Graph

2.1.1

Soln. The computational graph can be drawn as:

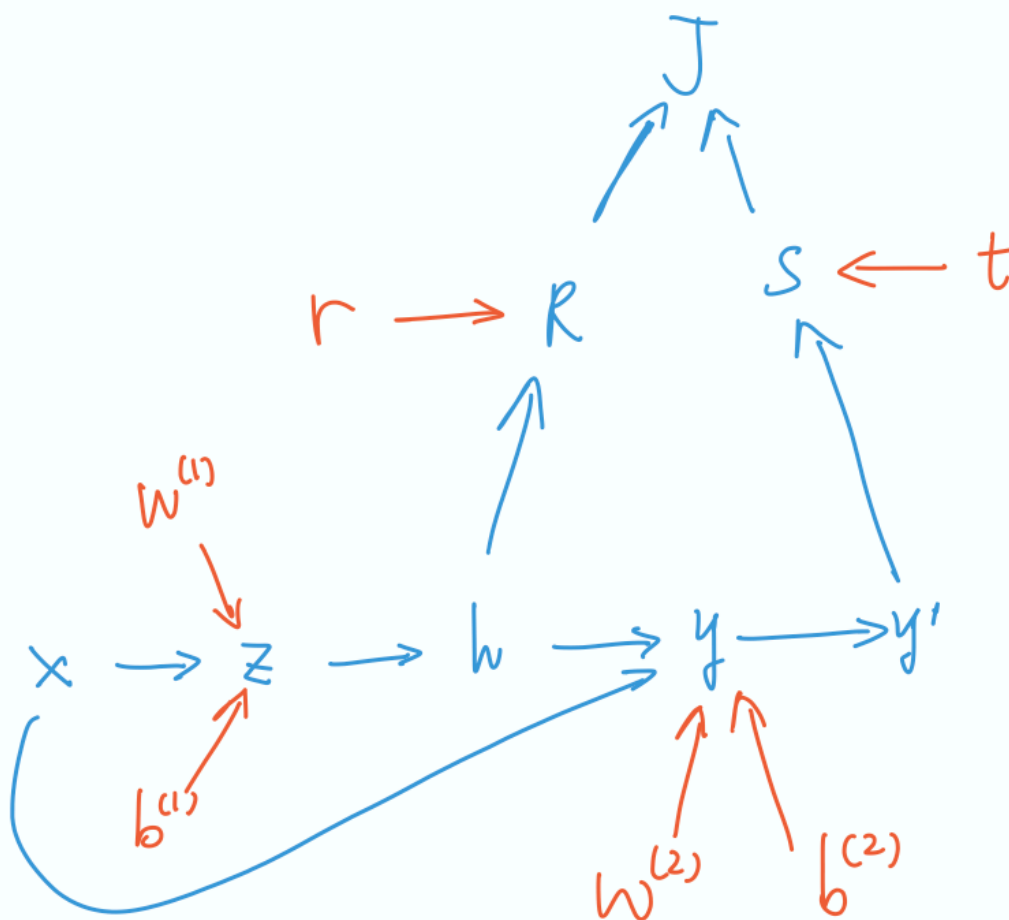


Figure 2.1: Computational Graph

■

2.1.2

Soln. For individual derivatives:

$$\overline{\mathcal{J}} = \frac{\partial \mathcal{J}}{\partial \mathcal{J}} = 1 \quad (2.1)$$

$$\overline{\mathcal{R}} = \frac{\partial \mathcal{J}}{\partial \mathcal{R}} = \overline{\mathcal{J}} \mathbf{1} = 1 \quad (2.2)$$

$$\overline{\mathcal{S}} = \overline{\mathcal{J}}(-1) = -1 \quad (2.3)$$

$$\overline{\mathbf{r}} = \overline{\mathcal{R}} \frac{\partial \mathcal{R}}{\partial \mathbf{r}} = \mathbf{h}^T \overline{\mathcal{R}} \quad (2.4)$$

$$\overline{\mathbf{y}'} = \overline{\mathcal{S}} \frac{\partial \mathcal{S}}{\partial \mathbf{y}'} = \mathbf{e}_t^T \overline{\mathcal{S}} \quad (2.5)$$

$$\overline{\mathbf{y}} = \overline{\mathbf{y}'} \frac{\partial \mathbf{y}'}{\partial \mathbf{y}} = \text{softmax}'(\mathbf{y})^T \overline{\mathbf{y}'} \quad (2.6)$$

$$\overline{\mathbf{h}} = \overline{\mathcal{R}} \frac{\partial \mathcal{R}}{\partial \mathbf{h}} + \overline{\mathbf{y}} \frac{\partial \mathbf{y}}{\partial \mathbf{h}} = \overline{\mathcal{R}} \mathbf{r} + \mathbf{W}^{(2)T} \overline{\mathbf{y}} \quad (2.7)$$

$$\overline{\mathbf{b}^{(2)}} = \overline{\mathbf{y}} \quad (2.8)$$

$$\overline{\mathbf{W}^{(2)}} = \overline{\mathbf{y}} \mathbf{h}^T \in \mathbb{R}^{N \times N} \quad (2.9)$$

$$\overline{\mathbf{z}} = \overline{\mathbf{h}} \mathbf{1} \{ \mathbf{z} \geq 0 \} \quad (2.10)$$

$$\overline{\mathbf{W}^{(1)}} = \overline{\mathbf{z}} \mathbf{x}^T \in \mathbb{R}^{N \times N} \quad (2.11)$$

$$\overline{\mathbf{b}^{(1)}} = \overline{\mathbf{z}} \quad (2.12)$$

$$\overline{\mathbf{x}} = \mathbf{W}^{(1)T} \overline{\mathbf{z}} + \overline{\mathbf{y}} \quad (2.13)$$

where \mathbf{e}_t denotes the one-hot vector in \mathbb{R}^N in which the t^{th} element is one. ■

2.2 Vector-Jacobian Product (VJPs)

2.2.1

Soln.

$$\mathbf{f}(\mathbf{x}) := \mathbf{v} \mathbf{v}^T \mathbf{x} \quad (2.14)$$

$$\implies J = \frac{d}{d\mathbf{x}} \mathbf{f}(\mathbf{x}) = \mathbf{v} \mathbf{v}^T \quad (2.15)$$

$$\implies J = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 4 & 6 \\ 3 & 6 & 9 \end{pmatrix} \quad (2.16)$$

■

2.2.2

Soln. Time cost: n^2 . Memory cost: $\frac{(n+1)n}{2}$ (only need to store elements above the diagonal (including the diagonal), because J is symmetric). Therefore, both time and memory cost are $\mathcal{O}(n^2)$. ■

2.2.3

Soln.

$$J^T \mathbf{y} = \mathbf{v} \mathbf{v}^T \mathbf{y} \quad (2.17)$$

$$= \mathbf{v}(\mathbf{v}^T \mathbf{y}) \quad (2.18)$$

$$= \underbrace{\mathbf{v} \underbrace{\langle \mathbf{v}, \mathbf{y} \rangle}_{\text{step 2}}}_{\text{step 1}} \quad (\dagger) \quad (2.19)$$

$$\implies \mathbf{z} \mathbf{v} \sum_{i=1}^3 v_i y_i \quad (2.20)$$

$$\implies \mathbf{z}^T = 6 \mathbf{v}^T \quad (2.21)$$

$$\implies \mathbf{z}^T = [6, 12, 18] \quad (2.22)$$

where the inner product operation has time cost n and its output takes memory cost 1. Let $\alpha = \langle \mathbf{v}, \mathbf{y} \rangle \in \mathbb{R}$, the vector scalar multiplication operation $\mathbf{v}\alpha$ has time cost n and its output has memory cost n .

The overall time cost for (\dagger) is therefore $2n$ and memory cost is $n + 1$.

So that both the memory and time costs are in $\mathcal{O}(n)$. ■

3 Linear Regression

3.1 Driving the Gradient

Soln.

$$\frac{d}{d\hat{\mathbf{w}}} \frac{1}{n} (X\hat{\mathbf{w}} - \mathbf{t})^2 = \frac{d}{d\hat{\mathbf{w}}} \frac{1}{n} \|X\hat{\mathbf{w}} - \mathbf{t}\|_2^2 \quad (3.1)$$

$$= \frac{2}{n} (X\hat{\mathbf{w}} - \mathbf{t})^T X \quad (3.2)$$

$$\implies \nabla_{\mathbf{w}} \mathcal{J} = \frac{2}{n} X^T (X\hat{\mathbf{w}} - \mathbf{t}) \quad (3.3)$$

■

3.2 Under-parameterized Model

3.2.1

Soln. Assume $d < n$ so that $X^T X$ is invertible. The gradient descent algorithm converges when the gradient equals zero:

$$\frac{2}{n} (X\hat{\mathbf{w}} - \mathbf{t})^T X = 0 \quad (3.4)$$

$$\implies (X\hat{\mathbf{w}} - \mathbf{t})^T X = 0 \quad (3.5)$$

$$\implies X^T (X\hat{\mathbf{w}} - \mathbf{t}) = 0^T \quad (3.6)$$

$$\implies X^T X\hat{\mathbf{w}} - X^T \mathbf{t} = 0^T \quad (3.7)$$

$$\implies X^T X\hat{\mathbf{w}} = X^T \mathbf{t} \quad (3.8)$$

$$\implies \hat{\mathbf{w}} = (X^T X)^{-1} X^T \mathbf{t} \quad (3.9)$$

■

3.2.2

Soln. Let $\mathbf{x} \in \mathbb{R}^d$, note that $(X^T X)^{-1}$ is symmetric. Assuming target \mathbf{t} is generated by a linear process, then $\mathbf{t} = X\mathbf{w}^*$. Immediately, $\mathbf{t}^T = \mathbf{w}^{*T} X^T$.

$$(\mathbf{w}^{*T} \mathbf{x} - \hat{\mathbf{w}}^T \mathbf{x})^2 = (\mathbf{w}^{*T} \mathbf{x} - [(X^T X)^{-1} X^T \mathbf{t}]^T \mathbf{x})^2 \quad (3.10)$$

$$= (\mathbf{w}^{*T} \mathbf{x} - \mathbf{t}^T X (X^T X)^{-1} \mathbf{x})^2 \quad (3.11)$$

$$= (\mathbf{w}^{*T} \mathbf{x} - \mathbf{w}^{*T} X^T X (X^T X)^{-1} \mathbf{x})^2 \quad (3.12)$$

$$= (\mathbf{w}^{*T} \mathbf{x} - \mathbf{w}^{*T} \mathbf{x})^2 \quad (3.13)$$

$$= 0 \quad (3.14)$$

■

3.3 Over-parameterized Model: 2D Example

3.3.1

Soln. To minimize the empirical risk minimizer,

$$\min_{w_1, w_2} (w_1 x_1 + w_2 x_2 - t_1)^2 \quad (3.15)$$

$$\text{equivalently, } \min_{w_1, w_2} (2w_1 + w_2 - 2)^2 \quad (3.16)$$

Any pair of (w_1, w_2) satisfying

$$2w_1 + w_2 - 2 = 0 \quad (\dagger) \quad (3.17)$$

attains the minimum level of empirical risk (zero). Equivalently, any $\hat{\mathbf{w}}$ on the line

$$\hat{\mathbf{w}} = \begin{pmatrix} 0 \\ 2 \end{pmatrix} + t \begin{pmatrix} 1 \\ -2 \end{pmatrix} \text{ for } t \in \mathbb{R} \quad (3.18)$$

satisfies (\dagger) . Therefore, there are infinitely many empirical risk minimizers.

Equivalently, the collection of solution is

$$w_2 = -2w_1 + 2 \quad (3.19)$$

■

3.3.2

Soln. From the first part of this question we know that

$$\nabla_{\mathbf{w}} \mathcal{J} = \frac{2}{n} (X \hat{\mathbf{w}} - \mathbf{t})^T X \quad (3.20)$$

$$= \frac{2}{1} \left[\begin{pmatrix} 2 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 0 \end{pmatrix} - 2 \right] \begin{pmatrix} 2 \\ 1 \end{pmatrix} \quad (3.21)$$

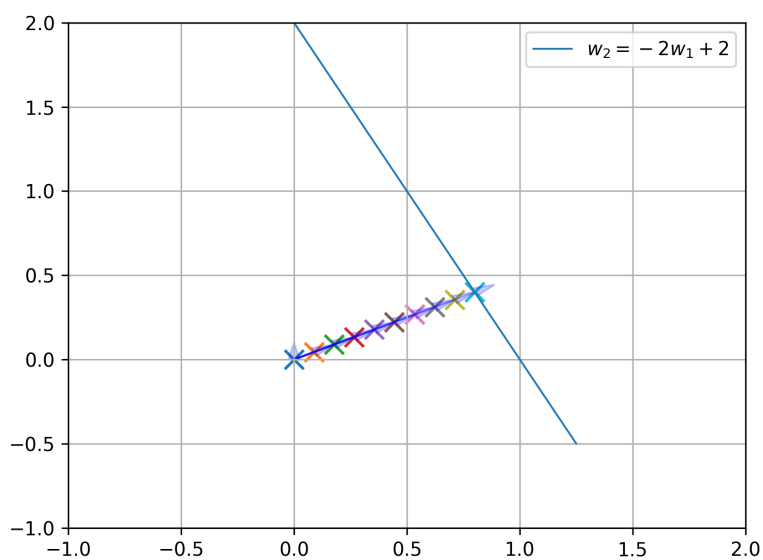
$$= \begin{pmatrix} -4 \\ -2 \end{pmatrix} \quad (3.22)$$

The unit-norm gradient is

$$\widehat{\nabla_{\mathbf{w}} \mathcal{J}} = \begin{pmatrix} -\frac{2\sqrt{5}}{5} \\ -\frac{\sqrt{5}}{5} \end{pmatrix} \quad (3.23)$$

The direction (gradient) does not change along the trajectory. Ultimately, the gradient descent algorithm converges to

$$\hat{\mathbf{w}}^* = \begin{pmatrix} \frac{4}{5} & \frac{2}{5} \end{pmatrix} \quad (3.24)$$



■

3.3.3

Soln. Let $\hat{\mathbf{w}}^*$ denote the solution found using gradient descent. Note that the line of solution can be written parametrically as

$$\hat{\mathbf{w}}(t) = \begin{pmatrix} 0 \\ 2 \end{pmatrix} + t \begin{pmatrix} 1 \\ -2 \end{pmatrix} \quad (3.25)$$

The path of gradient descent starts from $\mathcal{S} = \mathbf{0}$ and the path is perpendicular to the line of solutions $\{\hat{\mathbf{w}}(t) : t \in \mathbb{R}\}$. By Pythagorean theorem, we know that the shortest path from a fixed point x to a line ℓ is the perpendicular line x and some point y on ℓ . Meanwhile, y is the point on ℓ nearest to x . The path of gradient descent is exactly such a shortest path. Therefore, $\hat{\mathbf{w}}^*$ is the solution nearest to $\mathbf{0}$. In another word, $\hat{\mathbf{w}}$ is solution with smallest Euclidean among all solutions in \mathcal{S} .

Note: this proof is rough and relies on geometric intuition, for a more formal and general proof, please refer to 3.4.2. ■

3.4 Overparameterized Model: General Case

3.4.1

Proof. Note that the solution reached by gradient descent \mathbf{w}^* can be written as a linear combination of rows of X :

$$X = \begin{pmatrix} x_{11} & \cdots & x_{1d} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{nd} \end{pmatrix} \quad (3.26)$$

$$\mathbf{w}^* = r_1 \begin{pmatrix} x_{11} \\ \vdots \\ x_{1d} \end{pmatrix} + \cdots + r_n \begin{pmatrix} x_{n1} \\ \vdots \\ x_{nd} \end{pmatrix} \quad (3.27)$$

$$= \begin{pmatrix} x_{11} & \cdots & x_{1d} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{nd} \end{pmatrix}^T \begin{pmatrix} r_1 \\ \vdots \\ r_n \end{pmatrix} \quad (3.28)$$

$$= X^T \mathbf{r} \quad (3.29)$$

The original minimization problem in \mathbb{R}^d (choosing the optimal $\mathbf{w} \in \mathbb{R}^d$) can be reduced to a minimization problem in \mathbb{R}^n (choosing the optimal $\mathbf{r} \in \mathbb{R}^n$).

The gradient descent algorithm converges if and only if zero gradient is encountered, which is characterized by the following condition:

$$\frac{d}{d\mathbf{r}} \frac{1}{n} \|XX^T \mathbf{r} - \mathbf{t}\|_2^2 = 0 \quad (3.30)$$

$$\implies \frac{2}{n} (XX^T \mathbf{r} - \mathbf{t})^T XX^T = 0 \quad (3.31)$$

$$\implies XX^T (XX^T \mathbf{r} - \mathbf{t}) = 0 \text{ (take transpose on both sides)} \quad (3.32)$$

$$\implies XX^T XX^T \mathbf{r} = XX^T \mathbf{t} \quad (3.33)$$

Because $d > n$, so $\text{rank}(XX^T) = n$ and it is invertible. Therefore,

$$(XX^T)^{-1}XX^TXX^T\mathbf{r} = (XX^T)^{-1}XX^T\mathbf{t} \quad (3.34)$$

$$\implies XX^T\mathbf{r} = \mathbf{t} \quad (3.35)$$

$$\implies \mathbf{r}^* = (XX^T)^{-1}\mathbf{t} \quad (3.36)$$

$$\implies \mathbf{w}^* = X^T(XX^T)^{-1}\mathbf{t} \quad (3.37)$$

where the solution is uniquely determined. ■

3.4.2

Proof. Let $\hat{\mathbf{w}}_1$ be another zero loss weight, then

$$X\hat{\mathbf{w}}_1 = \mathbf{t} \quad (3.38)$$

Immediately,

$$\hat{\mathbf{w}}_1^T X^T = \mathbf{t}^T \quad (3.39)$$

Evaluating

$$(\hat{\mathbf{w}} - \hat{\mathbf{w}}_1)^T \hat{\mathbf{w}} = \|\hat{\mathbf{w}}\|_2^2 - \hat{\mathbf{w}}_1^T \hat{\mathbf{w}} \quad (3.40)$$

$$= (X^T(XX^T)^{-1}\mathbf{t})^T (X^T(XX^T)^{-1}\mathbf{t}) - \hat{\mathbf{w}}_1^T \hat{\mathbf{w}} \quad (3.41)$$

$$= \mathbf{t}^T (XX^T)^{-1} X (X^T(XX^T)^{-1}\mathbf{t}) - \hat{\mathbf{w}}_1^T \hat{\mathbf{w}} \quad (3.42)$$

$$= \mathbf{t}^T (XX^T)^{-1} \mathbf{t} - \hat{\mathbf{w}}_1^T X^T (XX^T)^{-1} \mathbf{t} \quad (3.43)$$

$$= \mathbf{t}^T (XX^T)^{-1} \mathbf{t} - \mathbf{t}^T (XX^T)^{-1} \mathbf{t} \quad (3.44)$$

$$= 0 \quad (\dagger) \quad (3.45)$$

Because $\hat{\mathbf{w}}_1$ is chosen arbitrarily, let \mathcal{S} denote the space of zero-loss weights. (\dagger) suggests the optimal solution found by gradient descent $\hat{\mathbf{w}} \perp \mathcal{S}$.

Let $\hat{\mathbf{w}}$ denote the solution found by gradient descent.

Let $\hat{\mathbf{w}}_\alpha \in \mathcal{S}$ and $\mathbf{v} = \hat{\mathbf{w}}_\alpha - \hat{\mathbf{w}}$.

$$\|\hat{\mathbf{w}}_\alpha\|_2^2 = \|\hat{\mathbf{w}} + \mathbf{v}\|_2^2 \quad (3.46)$$

$$= \|\hat{\mathbf{w}}\|_2^2 + 2\mathbf{v} \cdot \hat{\mathbf{w}} + \|\mathbf{v}\|_2^2 \quad (3.47)$$

$$= \|\hat{\mathbf{w}}\|_2^2 + 2(\hat{\mathbf{w}}_\alpha - \hat{\mathbf{w}}) \cdot \hat{\mathbf{w}} + \|\mathbf{v}\|_2^2 \quad (3.48)$$

$$= \|\hat{\mathbf{w}}\|_2^2 + \|\mathbf{v}\|_2^2 \text{ by } (\dagger) \quad (3.49)$$

$$\geq \|\hat{\mathbf{w}}\|_2^2 \quad (3.50)$$

Therefore,

$$\|\hat{\mathbf{w}}_\alpha\|_2^2 \geq \|\hat{\mathbf{w}}\|_2^2 \quad \forall \hat{\mathbf{w}}_\alpha \in \mathcal{S} \quad (3.51)$$

And $\hat{\mathbf{w}}$ is the element with smallest Euclidean norm among all zero-loss solutions in \mathcal{S} . ■

3.5 Benefit of Overparameterization

3.5.1

Soln. Fitting function implementation:

```
1 def fit_poly(X, d,):
2     X_expand = poly_expand(X, d=d, poly_type = poly_type)
3     if d > n: # Over-parameterized case
4         W = X_expand.T @ np.linalg.inv(X_expand @ X_expand.T) @ t
5     else: # Under parameterized case.
6         W = np.linalg.inv(X_expand.T @ X_expand) @ X_expand.T @ t
7     return W
8
```

Lines fitted and the corresponding losses using both legendre and chebyshev polynomials are presented in figures below. In both experiments, over-parameterization does **not** always lead to overfitting. In fact, models with a super high degree (say 100) models resulted similar performances compared with low degrees (In the Legendre case, high degree models actually outperformed low degree models).

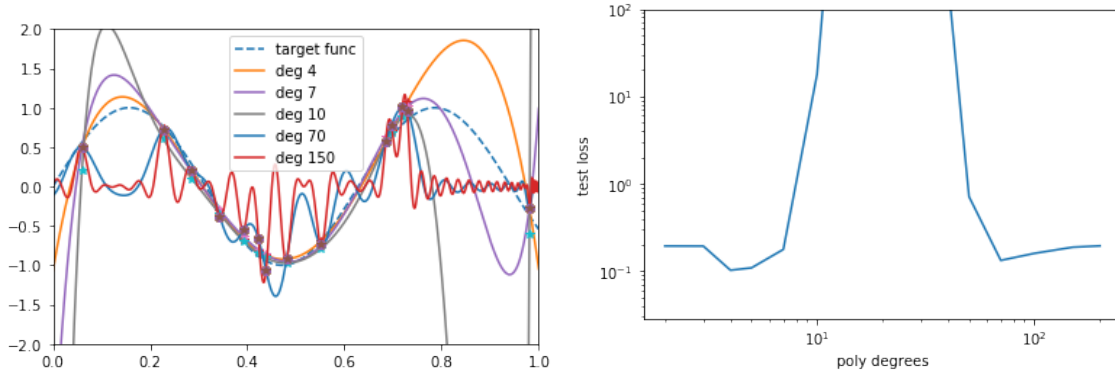


Figure 3.1: chebyshev results

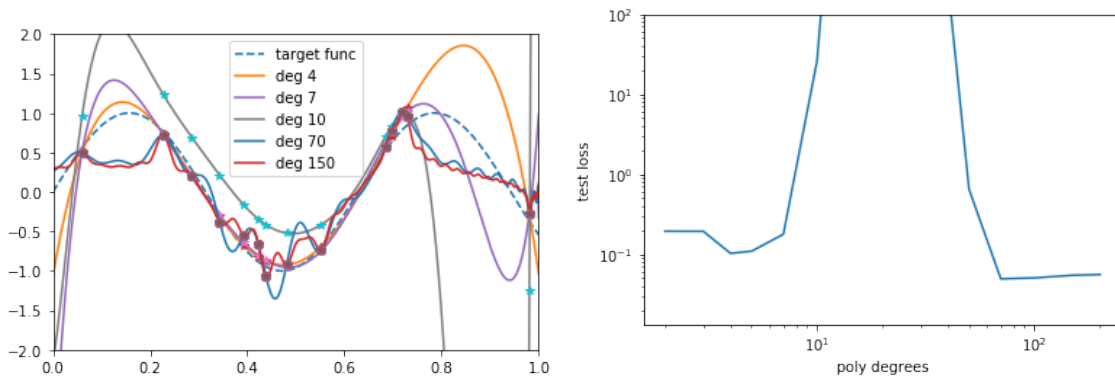


Figure 3.2: legendre results

■

3.5.2

Soln. Not required.

