CSC413 Homework 1

1.1 Verify Sort

let h_1 detect $x_1 \le x_2 \Leftrightarrow x_2 - x_1 \ge 0$
let h_2 detect $x_2 \le x_3 \Leftrightarrow x_3 - x_2 \ge 0$
let h_3 detect $x_3 \le x_4 \Leftrightarrow x_4 - x_3 \ge 0$
let y detect $h_1 = h_2 = h_3 = 1$

$h = W^{(1)}x + b^{(1)}$
$3\times1 \quad 3\times4 \; 4\times1 \quad 3\times1$

$$W^{(1)} = \begin{bmatrix} -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 \end{bmatrix}$$

$$b^{(1)} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$y = w^{(2)T}h + b^{(2)}$
$1\times1 \quad 1\times3 \; 3\times1 \quad 1\times1$

$$w^{(2)} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

$$b^{(2)} = -2.5$$

1.2 Perfom Sort

Suppose we have now 1. the verify_sort that checks for the sortedness
and
2. a permutation neural network, which takes in $[x_1, x_2, x_3, x_4]$ and output all 24 permutations,

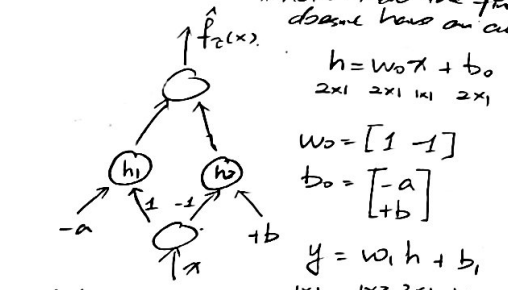
then, for p in permutations:
  if verify_sort(p) == 1:
    return p.

1.3 Universal Approximation theorem

1.3.1 bump function

if $a \le x \le b$, return h

$a \le x$ and $x \le b$. $\Rightarrow$ requires 2 hidden units.
otherwise return 0
* note that the final layer doesn't have an activation function



$a \le x$ | $x \le b$
$x \ge a$. | $-x \ge -b$.
$x - a \ge 0$ | $-x + b \ge 0$

$h = W_0 x + b_0$
$2\times1 \quad 2\times1 \; 1\times1 \quad 2\times1$

$$W_0 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

$$b_0 = \begin{bmatrix} -a \\ +b \end{bmatrix}$$

$y = w_1 h + b_1$
$1\times1 \quad 1\times2 \; 2\times1 \quad 1\times1$

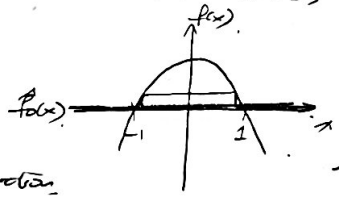
$$w_1 = \begin{bmatrix} h \\ h \end{bmatrix}^T$$

$b_1 = -h$

$w_{11}\overset{1}{x_1} + w_{12}\overset{1}{b_2} - b_1 = h$.

$w_{11} + w_{12} - b_1 = h$.

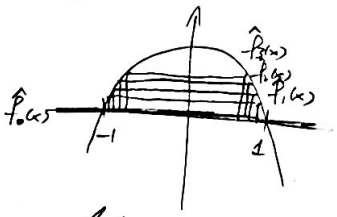
1.3.2 $f(x) = -x^2 + 1$, $\hat{f}_0(x) = 0$



find $\hat{f}_2(x) = \hat{f}_0(x) + g(h_1, a_1, b_1, x)$.
$= \hat{f}_0(x) + h \cdot I(a \le x \le b)$

let a_1 be a value slightly greater than -1, say $a = -0.999$
let b_1 be a value slightly smaller than 1. say $b = 0.999$
let h_1 be $f(a)$.

1.3.3

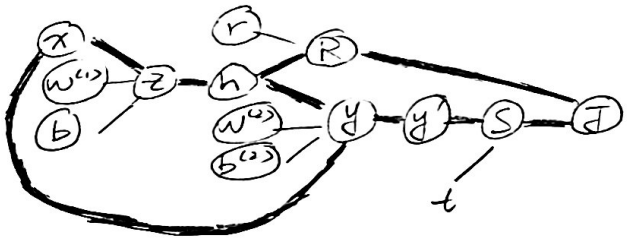


let a_i be a value slightly greater than a_{i-1}.
let b_i be a value slightly smaller than b_{i-1}.
let h_i be $f(a_i)$
keep doing this until a_i and b_i meets at 0

2. Backprop

2.1 Computational Graph.

2.1.1



2.1.2, assuming $M=N$.

→ $z = w^{(1)}x + b^{(1)}$ (K×1 = K×N N×1 K×1)

→ $h = \text{ReLU}(z)$ (K×1 = K×1) ⟶ $R = r^T h$ (1×1 = 1×K K×1)

→ $y = w^{(2)}h + b^{(2)} + x$ (M×1 = M×K K×1 M×1 N×1)

→ $y' = \text{softmax}(y)$ (M×1 = M×1) ⟶ $S = I(t=k)\,y_k'$ (1×1)

2.2 Vector-Jacobian Products (VJPs).

$f(x) = vv^Tx$ (n×1 = n×n n×1)

2.2.1

$v = \begin{bmatrix}1\\2\\3\end{bmatrix}$.

$vv^T = \begin{bmatrix}1\\2\\3\end{bmatrix}\begin{bmatrix}1&2&3\end{bmatrix} = \begin{bmatrix}1&2&3\\2&4&6\\3&6&9\end{bmatrix}$

$vv^Tx = \begin{bmatrix}1&2&3\\2&4&6\\3&6&9\end{bmatrix}\begin{bmatrix}x_1\\x_2\\x_3\end{bmatrix} = \begin{bmatrix}x_1+2x_2+3x_3\\2x_1+4x_2+6x_3\\3x_1+6x_2+9x_3\end{bmatrix} = f$

$J = \begin{bmatrix}\frac{\partial f_1}{\partial x_1}&\frac{\partial f_1}{\partial x_2}&\frac{\partial f_1}{\partial x_3}\\ \frac{\partial f_2}{\partial x_1}&\frac{\partial f_2}{\partial x_2}&\frac{\partial f_2}{\partial x_3}\\ \frac{\partial f_3}{\partial x_1}&\frac{\partial f_3}{\partial x_2}&\frac{\partial f_3}{\partial x_3}\end{bmatrix} = \begin{bmatrix}1&2&3\\2&4&6\\3&6&9\end{bmatrix}$. ✓

2.2.2

time cost: n^2

memory cost: n

2.2.3

$J = \frac{\partial f}{\partial x} = vv^T$ (3×3)

$J^Ty = (vv^T)^Ty = vv^Ty$

v^Ty requires ~ n multiply-additions

$v(v^Ty)$ requires n multiply

So time cost is linear, memory cost is also linear.

method: $J^Ty = vv^Ty$.

e.g. $J^Ty = vv^Ty = \begin{bmatrix}1\\2\\3\end{bmatrix}\begin{bmatrix}1&2&3\end{bmatrix}\begin{bmatrix}1\\1\\1\end{bmatrix} = \begin{bmatrix}1\\2\\3\end{bmatrix}\cdot 6 = \begin{bmatrix}6\\12\\18\end{bmatrix}$ ✓

2.1.2 $\bar{J} = 1$

$\bar{S} = \frac{\partial J}{\partial S}^T\bar{J} = 1\bar{J}$ ✓

$\bar{y'} = \frac{\partial S}{\partial y'}^T\bar{S} = 1I(t=k)$ ✓

$\bar{y} = \frac{\partial y'}{\partial y}^T\bar{y'} = \text{softmax}'(y)\bar{y'}$ (M×1) ✓

$\bar{R} = \frac{\partial J}{\partial R}^T\bar{J} = 1\bar{J}$ ✓

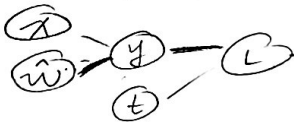
$\bar{h} = \frac{\partial R}{\partial h}^T\bar{R} + \frac{\partial y}{\partial h}^T\bar{y} = r\bar{R} + w^{(2)T}\bar{y}$ (K×1 = K×1 K×M M×1) ✓

$\bar{z} = \frac{\partial h}{\partial z}^T\bar{h} = \bar{h}\,\max(0,z)\frac{1}{z}$ (K×1) ✓

$\bar{x} = \frac{\partial z}{\partial x}^T\bar{z} = w^{(1)T}\bar{z} + \frac{\partial y}{\partial x}^T\bar{y}$ (N×1 = N×K K×1) ✓

3. Linear Regression.

3.1 Deriving the Gradient.



$y = x\hat{w}$ (n×1 = n×d d×1)

$L = (y-t)^2$ (1×1)

$J = \frac{1}{n}(y-t)^2$ (1×1)

$\bar{J} = 1$.

$\bar{L} = \frac{1}{n}$.

$\bar{y} = \frac{\partial L}{\partial y}^T\bar{L} = \frac{1}{n}2(y-t) = \frac{2}{n}(y-t)$

$\bar{\hat{w}} = \frac{\partial y}{\partial \hat{w}}^T\bar{y} = \frac{2}{n}x^T(y-t) = \frac{2}{n}x^T(x\hat{w}-t)$ (d×1 = d×n n×1) ✓

3.2 Underparametrized Model.

3.2.1

$\bar{\hat{w}} = 0 = x^T(y-t) = x^T(x\hat{w}^*-t)$ (d×n n×1)

$= x^Tx\hat{w}^* - x^Tt = 0$ (d×d d×1, d×n n×1)

$x^Tx\hat{w}^* = x^Tt$

$\hat{w}^* = (x^Tx)^{-1}x^Tt$ (d×1 = d×d d×n n×1) ✓

3.2.2

$\hat{w}^* = (x^Tx)^{-1}x^Tx w^*$

$= (x^Tx)^{-1}x^Tx w^*$

$= w^*$

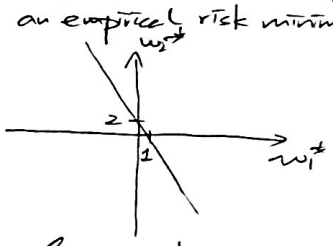
So $(w^{*T}x - \hat{w}^{*T}x)^2 = 0$ ✓

3.3 Overparametrized Model: 2D Example

3.3.1.

$$\nabla \tilde{w} = 0 = X^T(y - t) = X^T(X\hat{w}^* - t)$$
$$X^T X \hat{w}^* = X^T t$$
$$X X^T X \hat{w}^* = X X^T t$$
$$X\hat{w}^* = t$$
$$\begin{bmatrix} 2 \\ 1 \end{bmatrix} [w_1^* \; w_2^*] = 2$$
$$2w_1^* + w_2^* = 2$$

any (w_1^*, w_2^*) satisfy this relationship is an empirical risk minimizer.



✓

line: $w_2^* = 2 - 2w_1^*$.

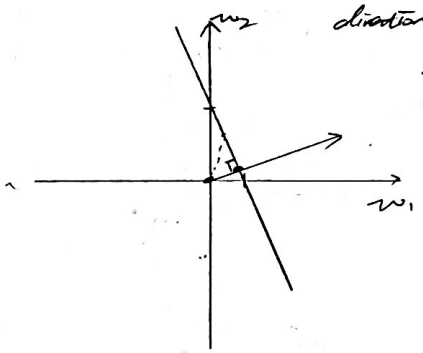
3.3.2

when $\hat{w}(0) = 0$, find direction of gradient.

$$\tilde{w} = \frac{2}{1}\begin{bmatrix} 2 \\ 1 \end{bmatrix}(0 - 2)$$
$$= -4\begin{bmatrix} 2 \\ 1 \end{bmatrix} = \begin{bmatrix} -8 \\ -4 \end{bmatrix}$$

after normalization:

$\tilde{w} = \begin{bmatrix} -2/\sqrt{5} \\ -1/\sqrt{5} \end{bmatrix}$, direction of gradient descent is $\begin{bmatrix} 2/\sqrt{5} \\ 1/\sqrt{5} \end{bmatrix}$.



direction doesn't change.

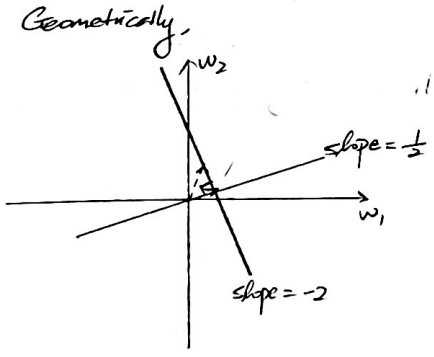
solution: find intersection of $w_2 = 2 - 2w_1$ and $w_2 = \frac{1}{2}w_1$.

$$\frac{1}{2}w_1 = 2 - 2w_1$$
$$\frac{1}{2}w_1 + 2w_1 = 2$$
$$w_1 = \frac{4}{5}, \; w_2 = \frac{2}{5}.$$

✓

3.3.3.

Geometrically,



Since the slopes are negative reciprocals, the gradient descent solution is perpendicular to the line of all empirical risk minimizers.

Suppose there exists another solution on the line,

Using Pythagorean, $a^2 + b^2 = c^2$.

for $b \neq 0$,
$c^2 > a^2$.
$c > a$

/

3.4 Overparametrized Model: General Case.

3.4.1

$$\nabla \tilde{w} = 0 \Rightarrow \underset{n \times d}{X} \underset{d \times 1}{\hat{w}^*} = \underset{n \times 1}{t}$$ empirical risk minimizer.

$$\nabla \tilde{w}(0) = \frac{2}{n} X^T(X \cdot 0 - t)$$
$$= \underset{d \times n}{\frac{2}{n} X^T} \underset{n \times 1}{t}$$

gradient direction is a linear combination of rows of X^T. the coefficients are the n elements in t.

$\Rightarrow$ gradient is spanned by the rows of X^T ✓

find $\hat{w}^*$ that ① satisfies $X\hat{w}^* = t$
② $\hat{w}^*$ spanned by rows of X^T

let $c \in \mathbb{R}^{n \times 1}$ be the coefficients

$$X X^T c = t$$
$$c = (X X^T)^{-1} t$$
$$\text{so, } \hat{w}^* = X^T (X X^T)^{-1} t$$ ✓

3.4.2

$$(\hat{w}^* - \hat{w}_1)^T \hat{w}^*$$
$$= (\hat{w}^{*T} - \hat{w}_1^T)\hat{w}^*$$
$$= \hat{w}^{*T}\hat{w}^* - \hat{w}_1^T \hat{w}^*$$
$$= (t^T (X X^T)^{-T} X)(X^T (X X^T)^{-1} t) - \hat{w}_1^T X^T (X X^T)^{-1} t$$
$$= t^T (X X^T)^{-1} t - \hat{w}_1^T X^T (X X^T)^{-1} t$$
$$= t^T (X X^T)^{-T} t - t^T (X X^T)^{-1} t$$
$$= t^T \left((X X^T)^{-T} - (X X^T)^{-1} \right) t = 0$$

(inverse of symmetric matrix is also symmetric)

so, $(\hat{w}^* - \hat{w}_1)$ is orthogonal to $\hat{w}^*$.

using the same reasoning of pythagorean, the norm of $\hat{w}^*$ is the smallest.

✓

3.3 Benefit of Overparameterization

3.3.1

code snippet:

```
if d>n:
    w = A_expand.T @ linalg.inv(A_expand @ A_expand.T) @ z
else:
    w = linalg.inv(A_expand.T @ A_expand) @ A_expand.T @ z
return w
```

No, overparameterization doesn't always lead to overfitting. ✓