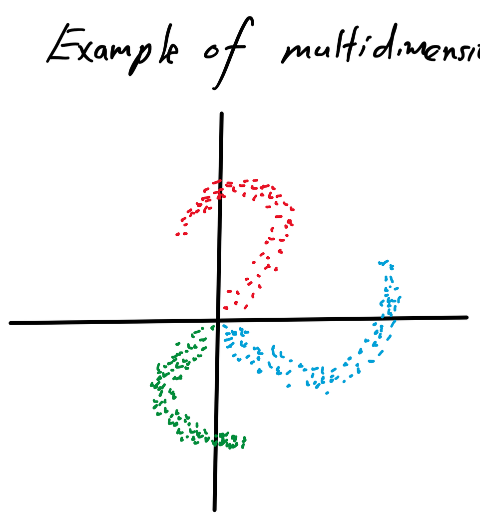
Classification and the Tendril Problem

Wednesday, July 19, 2023 10:08 AM

Example of multidimensional input/output



input D_{in} : $\vec{x}$: $[x, y]$
output D_{out} : $\vec{y}$:

- 1 value - what color it is of ? datatype
- 3 values for RGB - uint8
- 3 values for the probabilities
  $p(R)$, $p(G)$, $p(B)$

$\vec{y}$: $[s_r, s_b, s_g]$

$\vec{y} = \sum_{i=0}^{N-1} \vec{v}_i \varphi(\vec{x}_i \cdot \vec{\omega}_i + b_i)$

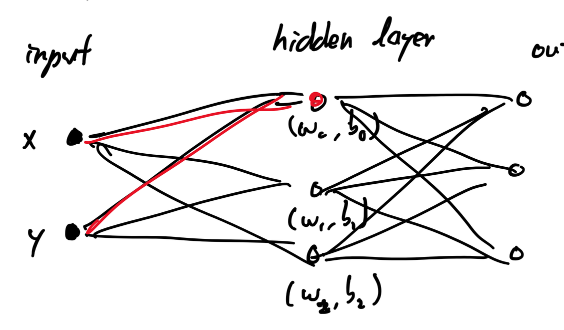
(2,)

size of $\vec{\omega}_i$ and $\vec{v}_i$?

$\vec{\omega}_i$: (2,)

$\vec{v}_i$: (3,)

example: say $N = 3$ N = # of neurons in our 1-layer NN



$w_i = (2,)$
inputs into each neuron

b_i = scalar
- # equal to # of neurons
- no relation to I/O size

$v_i = (3,)$
match output
$[s_r, s_b, s_g]$

How do we train in batches?

$\vec{x} \rightarrow (\vec{x})^M$ ← batch size
(2,) (M,2)

$\vec{y} \rightarrow (\vec{y})^M$
(3,) (3, M)

batch of tendril points

$\vec{x} = (M,2) = \begin{bmatrix} - \vec{x}_0 - \\ - \vec{x}_1 - \\ \vdots \\ - \vec{x}_{M-1} - \end{bmatrix}$

$\vec{\omega} = (2,N) = \begin{bmatrix} | & | & & | \\ \vec{w}_0 & \vec{w}_1 & \dots & \vec{w}_{N-1} \\ | & | & & | \end{bmatrix}$

$\vec{b} = (N) = [b_0 \dots b_{N-1}]$

$\underbrace{(M,2)(2,N)}_{\text{shape}}$

$\vec{V} = (N,3) = \begin{bmatrix} - \vec{v}_0 - \\ - \vec{v}_1 - \\ \vdots \\ - v_{N-1} - \end{bmatrix}$

$XW = \begin{bmatrix} x_0 w_0 & x_0 w_1 & \dots & x_0 w_{N-1} \\ \vdots & \ddots & & \\ x_{M-1} w_0 & \dots & & x_{M-1} w_{N-1} \end{bmatrix}$
(M,N)

$\varphi(XW+b) = \begin{bmatrix} \varphi(x_0 w_0 + b_0) & \dots & \varphi(x_0 w_{N-1} + b_{N-1}) \\ \vdots & \ddots & \end{bmatrix}$
(M,N)

$V = (N,3) \begin{bmatrix} v_{0,0} & v_{0,1} & v_{0,2} \\ & \ddots & \\ v_{N-1,0} & & v_{N-1,2} \end{bmatrix}$ → $\underbrace{\varphi(XW+b)}_{(M,N)} \underset{(N,3)}{V} = \underset{(M,3)}{Y}$

Loss function

$\mathcal{L}_{cross\text{-}entropy} = - \sum_{i=0}^{C-1} p_i^{(true)} \log(p_i^{(pred)})$

compares 2 probability distributions (true) vs (pred)
minimized when $\vec{p}^{(true)} = \vec{p}^{(pred)}$

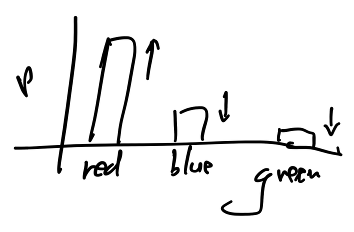
softmax cross-entropy
$\vec{s}$ → softmax → $\vec{p}$
scores probability-like

$softmax(\vec{s}) = \left[\frac{e^{s_0}}{\sum_j e^{s_j}}, \frac{e^{s_1}}{\sum_j e^{s_j}}, \frac{e^{s_2}}{\sum_j e^{s_j}} \right]$

$0 \le \frac{e^{s_k}}{\sum_j e^{s_j}} \le 1$ $\sum_k \left(\frac{e^{s_k}}{\sum_j e^{s_j}} \right) = 1 = \frac{\sum_k e^{s_k}}{\sum_j e^{s_j}}$

Example:

$p^{(true)} = [1, 0, 0]$ $p^{(pred)} = [0.75, 0.2, 0.05]$



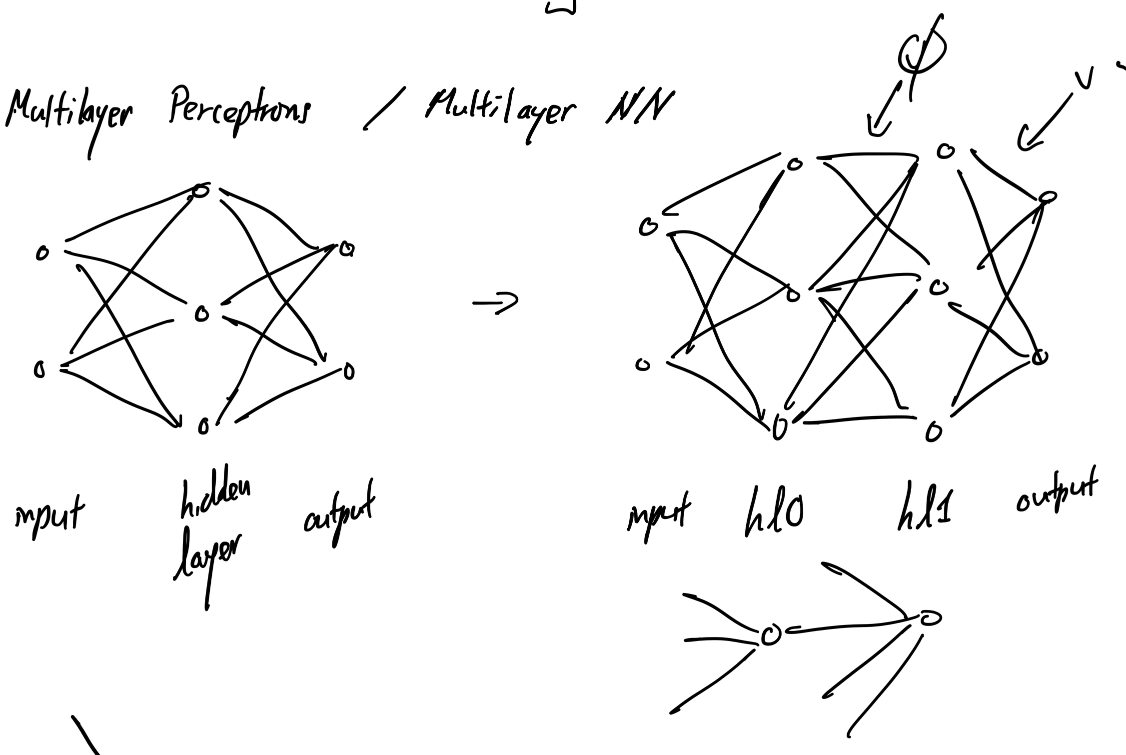
mygrad: softmax_crossentropy(scores, labels) ←
(M, D_{out}) (M)

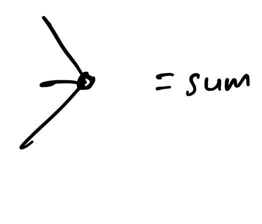
$\begin{bmatrix} s_{0,0} & s_{0,1} & \dots \\ \vdots & & \vdots \\ s_{N-1,0} & & s_{N-1,2} \end{bmatrix} \rightarrow \begin{bmatrix} p_{0,0} & p_{0,1} & \dots \\ & \vdots & \end{bmatrix}$

$p^{(pred)}$
$p^{(true)}$ $\frac{1}{M} \sum_{i=0}^{M-1} \mathcal{L}(p_i^{(pred)}, p_i^{(true)})$

$\begin{bmatrix} R \\ B \\ G \\ G \\ \vdots \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ & \vdots & \end{bmatrix}$

Multilayer Perceptrons / Multilayer NN





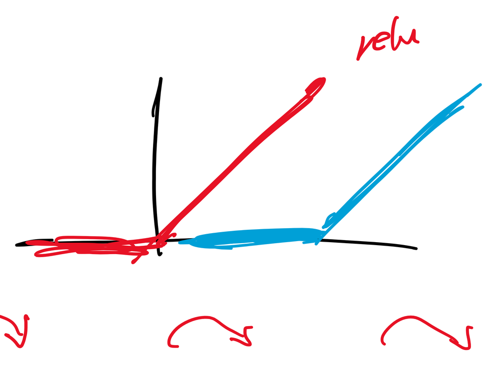
$\underbrace{\varphi(x_i w_i + b_i)}_{\text{one neuron}}$

$l_0 = [\varphi(x_0 w_0 + b_0)\ \varphi(x_1 w_1 + b_1) \dots \varphi(x_{N-1} w_{N-1} + b_{N-1})]$ ←
$= \varphi[XW_0 + b_0]$

$l_1 = \varphi[l_0 W_1 + b_1]$ - previous layers' output are now used as input

l_i = layer i for one input
L_i = layer i for one batch

one datum → $Y = \sum_{i=0}^{N-1} v_i \underbrace{\varphi(l_0 \cdot w_i + b)}_{l_1}$
sum and scale



$X = (M, D_{in})$
$W_0 = (D_{in}, N_0)$ $b_0 = (N_0)$
$W_1 = (N_0, N_1)$ $b_1 = (N_1)$
$V = (N_1, D_{out})$