11-785 05/05/2020

L4.5 - Backpropagation calculation

- continue S.2019 slides L4 from ⊛
- Focus on understanding

- Forward pass: Present input; compute affine combos z_j and activations
- Backward pass: Back through net, use forward pass computations to compute derivatives.

Special cases

- Assumed some things
- 1. 'Independent' computation
  2. Affine combination of inputs
  3. Diff. activations

Ⓝ: Think in terms of ~~topological~~ influence as routes through a topology

Special case

1. vector activation vs (scalar activation)

(*) Diagram

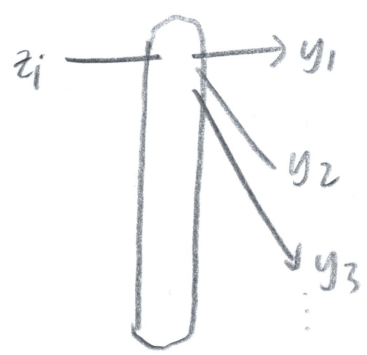
- A Renders activation multivariate vector function
- Has consequences for 'influence'

- $\frac{\partial Div}{\partial z_i^{(k)}} = \sum_j \frac{\partial Div}{\partial y_j^{(k)}} \frac{\partial y_j^{(k)}}{\partial z_i^{(k)}}$

- Softmax activation is an example of a vector activation

Softmax activ.: (as special case vector act.)

$$y_i = \frac{e^{z_i}}{\sum_j e^{z_j}} \quad \text{(softmax)}$$



$$\frac{dy_i}{dz_i} = \frac{d}{dz_i} e^{z_i} \cdot \frac{1}{\sum_j e^{z_j}} + e^{z_i} \frac{d}{dz_i} \left(\frac{1}{\sum_j e^{z_j}} \right)$$

$$= 6 \frac{e^{z_i}}{\sum_j e^{z_j}} + \frac{e^{z_i}(-e^{z_i})}{\left(\sum_j e^{z_j}\right)^2} \rightarrow y_i - y_i^2$$

$$= y_i - \frac{e^{z_i}}{\sum_j e^{z_j}} \cdot \frac{e^{z_i}}{\sum_j e^{z_j}}$$

$\frac{dy_i}{dz_i} = y_i - y_i^2$ (non diagonal path)

$$\frac{dy_i}{dz_k} = e^{z_i} \frac{d}{dz_k} \frac{1}{\sum_j e^{z_j}} = \frac{e^{z_i}(-e^{z_k})}{\left(\sum_j e^{z_j}\right)^2} = -y_i y_k \quad \text{(dicgonal path)}$$

- can compute derivative of each of ys to any z_j
- softmax output activation derivative varies depending on whether dicgonal or nondicgonal path.
- use Kronecker delta to compress notation

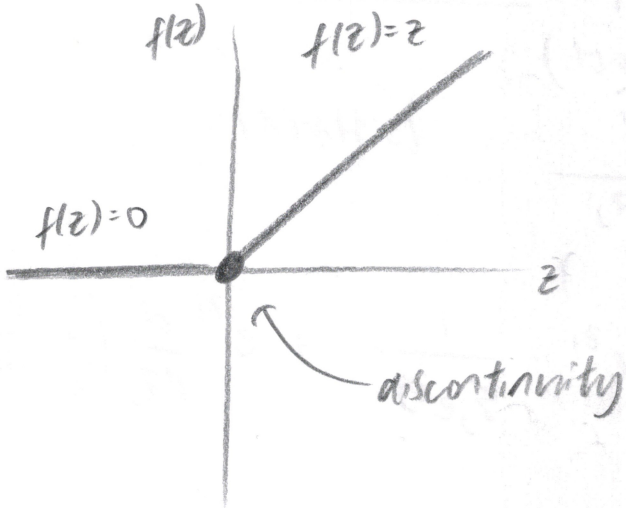
$$\frac{\partial Div}{\partial z_i^{(k)}} = \sum_j \frac{\partial Div}{\partial y_j^{(k)}} y_i^{(k)} (\delta_{ij} - y_j^{(k)})$$

(R)(W)(9) - multiplicative networks - review

Special case 3 - non-differentiable activ.

- Assumed activations diff. everywhere

RELU:



(*) subgradients → review

(W)(R) - check you understand definition

- Relu prime → use subgradients to get around discontinuity

- subgradient and max

- max → use subgradients

(Q): I am confused by inclusion of averaging factor $\frac{1}{|\xi \times 3|}$ $\frac{1}{T}$ $\frac{1}{N}$ in actual error

(B) Acknowledges 'average error/loss' rather than actual (?) → is this really used?

$\underline{W} \leftarrow \underline{W} - \eta \nabla_{\underline{W}} Err$

---

training via backprop algo pseudocode → deserves place in supplement notes

---

move to vector formulation (*) (W)(A3) → review notation

(W): (*) understood this; more compact

- simple conceptually
- fast matrix libraries (e.g. numpy)

(*) $\underline{z}_k = \underline{W}_k \underline{y}_{k-1} + \underline{b}_k$

$\underline{y}_k = f_k(\underline{z}_k)$

(W) this is key; everything before is fine; but if you want to commit anything to memory

---

(*)

$$\underline{Y} = f_N(\underline{W}_N f_{N-1}(\cdots f_2(\underline{W}_2 f_1(\underline{W}_1 \underline{x} + \underline{b}_1) + \underline{b}_2)\cdots) + \underline{b}_N)$$

---

init $\underline{y}_0 = \underline{x}$

$k = 1, \ldots, N$

$\underline{z}_k = \underline{W}_k \underline{y}_{k-1} + \underline{b}_k$

$\underline{y}_k = f_k(\underline{z}_k)$

output: $\underline{Y} = \underline{y}_N$

(*) $Div(\underline{Y}, d)$

- $Div\left[f_N(\underline{W}_N f_{N-1}(\dots f_2(\underline{W}_2 f_1(\underline{W}_1 \underline{X} + \underline{b}_1) + \underline{b}_2)\dots) + \underline{b}_N),\ d\right]$

Jacobians → derivative of vector fn. with respect to vector input

$\underline{y} = \underline{f}(\underline{z})$ i.e. $\begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix} = f\left(\begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_D \end{bmatrix}\right)$

- perturb-calc rule

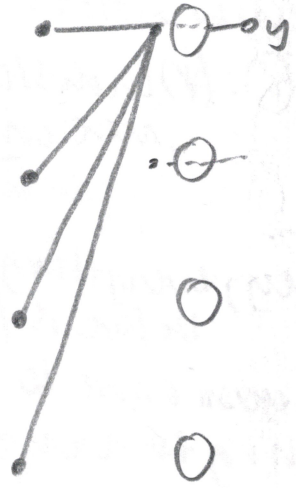
$\Delta y = J_y(\underline{z})\,\Delta \underline{z}$ $\quad J_y(\underline{z}) = \begin{bmatrix} \frac{\partial y_1}{\partial z_1} & \frac{\partial y_1}{\partial z_2} & \dots & \frac{\partial y_1}{\partial z_D} \\ \frac{\partial y_2}{\partial z_1} & \frac{\partial y_2}{\partial z_2} & \dots & \frac{\partial y_2}{\partial z_D} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial y_m}{\partial z_1} & \frac{\partial y_m}{\partial z_2} & \dots & \frac{\partial y_m}{\partial z_D} \end{bmatrix}$ - Remember I know

---

- Jacobian for scalar vs for vector activations

(*) Notice distinction

scalar activation

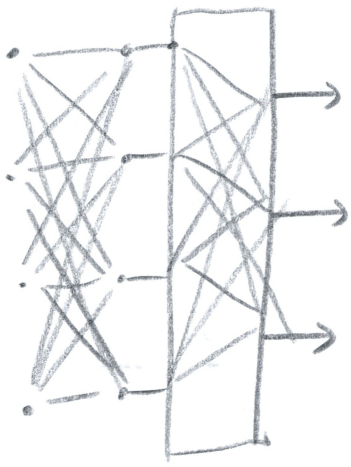


$J_y(\underline{z}) = \begin{bmatrix} \frac{dy_1}{dz_1} & & \\ & \ddots & \\ & & \frac{dy_D}{dz_D} \end{bmatrix}$

$y_i = f(z_i)$ $\quad J_y(\underline{z}) = \begin{bmatrix} f'(z_1) & & \\ & \ddots & \\ & & f'(z_D) \end{bmatrix}$

---

vector activations



$$J_{\underline{y}}(\underline{z}) = \begin{bmatrix} \frac{\partial y_1}{\partial z_1} & \cdots & \frac{\partial y_1}{\partial z_0} \\ \vdots & & \\ \frac{\partial y_m}{\partial z_1} & \cdots & \frac{\partial y_m}{\partial z_0} \end{bmatrix}$$

(W) check Jacobian notation (that you understand)
Remember → highly targetted learning

$J_{\underline{z}}(\underline{x})$ - Jacobian of $\underline{z}$ wrt $\underline{x}$ (BR notation)

- chain rule applied to Jacobians

- A cosmetic difference

(W): use ~~peter~~/understand BR way of understading chain rule

---

W(R) - Jacobian / scalar fm of vector m/ - familiarity

---

special cases (R)

- scalar (W)(RS): dimensionality ??

- dimensionality for higher-order calc.

(R): requires supplement

(*) Review and ensure you understand backprop in vector form, including dimensionality considerations

(W): once you understand
3 (*) calculations in vector-matrix form; repeated a, recursive application of same formula

---

Backward pass pseudocode
- check

- Most difficult lecture * (BR)

- Review; review; review

- BR: vector variant of backprop - you must be fluent.

- convergence of SGD } next lecture
- generalisation
- interp of output