RNN Derivatives and Code sample

By Mohit Kumar

Recurrent Neural Networks:ex-1:BPTT:Derivatives

Softmax Derivative

- It is fundamentally a vector function
  - So there is no derivative of softmax
  - Instead
    - Which component (output element) of softmax needs to be specified for derivative
    - Since softmax has multiple input, the derivative need to be calculated w.r.t which input element.
  - $\frac{\partial S_i}{\partial a_j}$ (partial derivative of ith output w.r.t jth input)
  - We say we compute its Jacobian Matrix.

Recurrent Neural Networks:ex-1:BPTT:Derivatives

Softmax Derivative

$$\frac{\partial S_i}{\partial a_j} = \frac{\partial \dfrac{e^{a_j}}{\sum_{K=1}^{N} e^{a_K}}}{\partial a_j}$$

Quotient Rule

$$f(x) = \frac{g(x)}{h(x)}$$

$$f'(x) = \frac{g'(x)\,h(x) - h'(x)\,g(x)}{h(x)^2}$$

- in our case

$$g_i = e^{a_i}$$

$$h_i = \sum_{K=1}^{N} e^{a_K}$$

- For h_i, no matter for which a_j the derivative is computed, the answer will always be e^{a_j}.
- For $g(i)$, the derivative w.r.t a_j is $\boxed{e^{a_j}}$ only if $i=j$, because only then $g(i)$ has a_j anywhere in it. Otherwise the derivative is $\boxed{0}$.

- When $i=j$ Softmax derivative

$$\frac{\partial \dfrac{e^{a_j}}{\sum_{k=1}^{N} e^{a_k}}}{\partial a_j} = \frac{e^{a_i}\cdot\Sigma - e^{a_j}\cdot e^{a_i}}{\Sigma^2}$$

$$= \frac{e^{a_i}}{\Sigma}\cdot\frac{\Sigma - e^{a_j}}{\Sigma}$$

$$= S_i(1-S_j)$$

- When $i\neq j$

$$\frac{\partial \dfrac{e^{a_j}}{\sum_{k=1}^{N} e^{a_k}}}{\partial a_j} = \frac{0 - e^{a_j}e^{a_i}}{\Sigma^2} = -\frac{e^{a_j}}{\Sigma}\cdot\frac{e^{a_j}}{\Sigma}$$

$$= -S_j S_i$$

$$\boxed{\begin{array}{ll} S_i(1-S_j) & i=j \\ -S_j\cdot S_i & i\neq j \end{array}}$$

eq-1

Recurrent Neural Networks:ex-1:BPTT:Derivatives

$$\boxed{\frac{\partial E}{\partial V}} = \frac{\partial E_t}{\partial \hat{y}_t} \cdot \frac{\partial \hat{y}_t}{\partial q_t} \cdot \frac{\partial q_t}{\partial V} \qquad q_t = V \cdot hs_t$$

- $E = -y_t \cdot \log \hat{y}_t$ (cross entropy loss)
- $\frac{\partial E}{\partial \hat{y}_t} = \boxed{-\frac{y_t}{\hat{y}_t}}$ eq-2
- $\hat{y}$ is the softmax function

Recurrent Neural Networks:ex-1:BPTT:Derivatives

$$\boxed{\frac{\partial E}{\partial V}} = \frac{\partial E_t}{\partial \hat{y}_t} \cdot \frac{\partial \hat{y}_t}{\partial q_t} \cdot \frac{\partial q_t}{\partial V}$$

Combining 2 & 1

$$q_t = V \cdot hs_t$$

$$\frac{\partial E}{\partial q_t} = \frac{\partial E_t}{\partial \hat{y}_t} \cdot \frac{\partial \hat{y}_t}{\partial q_t} = -\frac{y_t}{\hat{y}_t} \cdot \hat{y}_t(1-\hat{y}_t) + \sum_{i=j}\left(-\frac{y_t}{\hat{y}_t}\right)\left(-\hat{y}_t\,\hat{y}_t\right)$$

$$= -y_t + y_t\hat{y}_t + \sum_{i=j} y_t\hat{y}_t$$

$$= -y_t + \hat{y}_t \cdot \sum y_t$$

$$= \boxed{\hat{y}_t - y_t} \quad \left(\sum y_t = 1 \text{ because } y \text{ is one hot vector}\right)$$

eq-3

Recurrent Neural Networks:ex-1:BPTT:Derivatives

$$\boxed{\frac{\partial E}{\partial V}} = \frac{\partial E_t}{\partial \hat{Y}_t} \cdot \frac{\partial \hat{Y}_t}{\partial q_t} \cdot \frac{\partial q_t}{\partial V}$$

$$q_t = V \cdot hs_t$$

$$\boxed{\frac{\partial q_t}{\partial V} = \frac{\partial\, V \cdot hs_t}{\partial V} = hs_t}$$

eq-4

$$\frac{\partial E_t}{\partial \hat{Y}_t} \cdot \frac{\partial \hat{Y}_t}{\partial q_t} \cdot \frac{\partial q_t}{\partial V} = \hat{y} - y \cdot hs_t$$

$$\boxed{\frac{\partial E}{\partial V} = \hat{y}_t - y_t \cdot hs_t}$$

eq-5

Recurrent Neural Networks:ex-1:BPTT:Derivatives

$$\boxed{\frac{\partial E}{\partial u}} = \frac{\partial E}{\partial \hat{y}_t} \cdot \frac{\partial \hat{y}_t}{\partial q_{v_t}} \cdot \frac{\partial q_{v_t}}{\partial hs_t} \cdot \frac{\partial hs_t}{\partial z_t} \cdot \frac{\partial z_t}{\partial u}$$

where $q_{v_t} = v.hs_t$

$z_t = u.x_t + w.hs_{t-1}$

$hs_t = \tanh(z_t)$

$$\boxed{\frac{\partial q_{v_t}}{\partial hs_t} = \frac{\partial\, v.hs_t}{\partial hs_t} = v}$$ eq-6

$$\boxed{\frac{\partial z}{\partial u} = \frac{\partial(u.x + w.hs_{t-1})}{\partial u} = x_t}$$ eq-8

$$\boxed{\frac{\partial hs_t}{\partial z_t} = (1 - hs_t^2)}$$ eq-7

$$\boxed{\frac{\partial E}{\partial u} = \underbrace{(\hat{y} - y)}_{eq-3} \cdot \underbrace{v}_{eq-6} \cdot \underbrace{(1 - hs_t^2)}_{eq-7} \cdot \underbrace{x_t}_{eq-8}}$$ eq-9

Recurrent Neural Networks:ex-1:BPTT:Derivatives

$$\boxed{\frac{\partial E}{\partial w}} = \frac{\partial E}{\partial \hat{y}_t} \cdot \frac{\partial \hat{y}_t}{\partial q_{v_t}} \cdot \frac{\partial q_{v_t}}{\partial hs_t} \cdot \frac{\partial hs_t}{\partial z_t} \cdot \frac{\partial z_t}{\partial w}$$

where $q_{v_t} = v . hs_t$

$z_t = u . x_t + w . hs_{t-1}$

$hs_t = \tanh(z_t)$

$$\boxed{\frac{\partial z}{\partial w} = \frac{\partial(u . x + w . hs_{t-1})}{\partial u} = hs_{t-1}}$$

eq-10

$$\boxed{\frac{\partial E}{\partial w} = \underbrace{(\hat{y} - y)}_{eq-3} . \underbrace{v}_{eq-6} . \underbrace{(1 - hs_t^2)}_{eq-7} . \underbrace{hs_{t-1}}_{eq-10}}$$

eq-11

Recurrent Neural Networks:ex-1:BPTT:Derivatives

$$\boxed{\frac{\partial E}{\partial hs_{t-1}}} = \frac{\partial E}{\partial \hat{y}_t} \cdot \frac{\partial \hat{y}_t}{\partial q_{v_t}} \cdot \frac{\partial q_{v_t}}{\partial hs_t} \cdot \frac{\partial hs_t}{\partial hs_{t-1}} \cdot$$

where $q_{v_t} = v.hs_t$

$z_t = u.x_t + w.hs_{t-1}$

$hs_t = \tanh(z_t)$

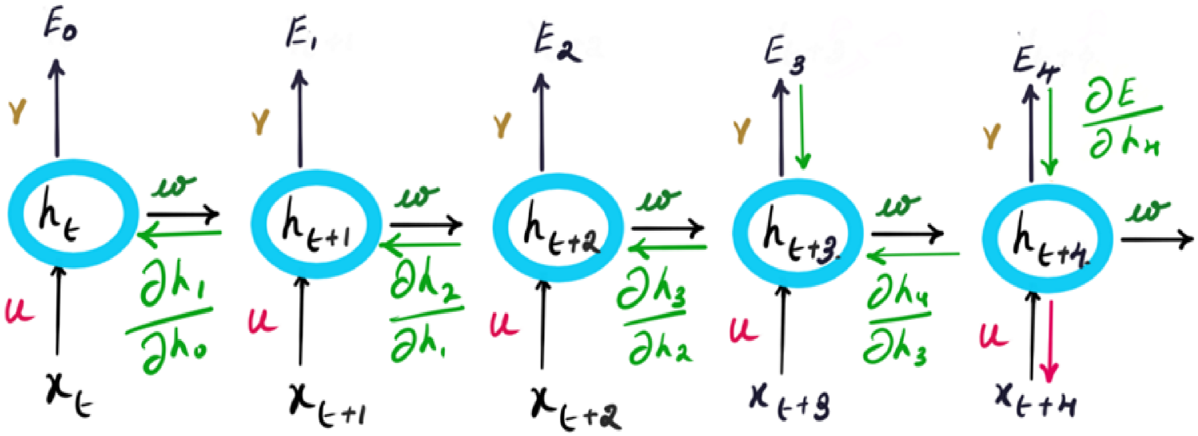
$$\boxed{\frac{\partial z}{\partial hs_{(t-1)}} = \frac{\partial(u.x + w.hs_{t-1})}{\partial u} = w}$$

eq-12

$$\boxed{\frac{\partial E}{\partial hs_{t-1}} = \underbrace{(\hat{y} - y)}_{eq\text{-}3} . \underbrace{v}_{eq\text{-}6} . \underbrace{(1 - hs_t^2)}_{eq\text{-}7} . \underbrace{w}_{eq\text{-}12}}$$

eq-13

Recurrent Neural Networks:ex-1:BPTT:Derivatives



$$h_t = \tanh(u \cdot x_t + w \cdot h_{(t-1)} + bh)$$

$$y_t = \mathrm{softmax}(v \cdot ht)$$

$$\boxed{\frac{\partial E}{\partial v} = \hat{y}_t - y_t \cdot hs_t}$$ eq-5

$$\boxed{\frac{\partial E}{\partial u} = \underbrace{(\hat{y} - y)}_{eq-3} \cdot \underbrace{v}_{eq-6} \cdot \underbrace{(1 - hs_t^2)}_{eq-7} \cdot \underbrace{x_t}_{eq-8}}$$ eq-9

$$\boxed{\frac{\partial E}{\partial w} = \underbrace{(\hat{y} - y)}_{eq-3} \cdot \underbrace{v}_{eq-6} \cdot \underbrace{(1 - hs_t^2)}_{eq-7} \cdot \underbrace{hs_{t-1}}_{eq-10}}$$ eq-11

$$\boxed{\frac{\partial E}{\partial hs_{t-1}} = \underbrace{(\hat{y} - y)}_{eq-3} \cdot \underbrace{v}_{eq-6} \cdot \underbrace{(1 - hs_t^2)}_{eq-7} \cdot \underbrace{w}_{eq-12}}$$ eq-13

Recurrent Neural Networks:ex-1:BPTT:Derivatives

```python
du, dw, dv = np.zeros_like(self.u), np.zeros_like(self.w), np.zeros_like(self.v)
dbh, dby = np.zeros_like(self.bh), np.zeros_like(self.by)
dhnext = np.zeros_like(hs[0])

for t in reversed(range(c)):
    label = labels[:, t * 1:(t + 1) * 1]
    dy = np.copy(ps[t])
    labelnhot=self.onehot(label)
    dy=dy-(labelnhot)
    dv += np.dot(dy, hs[t].T)
    dby += np.sum((dy),1).reshape(self.numclasses,1)
    dh = np.dot(self.v.T, dy) + dhnext  # backprop into h
    dhraw = (1 - hs[t] * hs[t]) * dh  # backprop through tanh nonlinearity
    dbh += np.sum((dhraw),1).reshape(self.statesize,1)
    du += np.dot(dhraw, xs[t].T)
    dw += np.dot(dhraw, hs[t - 1].T)
    dhnext = np.dot(self.w.T, dhraw)

for dparam in [dw, du, dv, dbh, dby]:
    np.clip(dparam, -5, 5, out=dparam)  # clip to mitigate exploding gradients
self.state=hs[c - 1]

for param, dparam, mem in zip([self.w, self.u, self.v, self.bh, self.by], [dw, du, dv, dbh, dby],
                              [self.mw, self.mu,self.mv, self.mbh, self.mby]):
    mem += dparam * dparam
    param += -self.learningrate * dparam / np.sqrt(mem + 1e-8)  # adagrad update
```

- calculate in reverse order.

$$\boxed{\frac{\partial E}{\partial v} = (\hat{y}_t - y_t) \cdot hs_t} \quad \text{eq-5}$$

$$\boxed{\frac{\partial E}{\partial u} = \underbrace{(\hat{y} - y)}_{\text{eq-3}} \cdot \underbrace{v}_{\text{eq-6}} \cdot \underbrace{(1 - hs_t^2)}_{\text{eq-7}} \cdot \underbrace{x_t}_{\text{eq-8}}} \quad \text{eq-9}$$

$$\boxed{\frac{\partial E}{\partial w} = \underbrace{(\hat{y} - y)}_{\text{eq-3}} \cdot \underbrace{v}_{\text{eq-6}} \cdot \underbrace{(1 - hs_t^2)}_{\text{eq-7}} \cdot \underbrace{hs_{t-1}}_{\text{eq-10}}} \quad \text{eq-11}$$

$$\boxed{\frac{\partial E}{\partial hs_{t-1}} = \underbrace{(\hat{y} - y)}_{\text{eq-3}} \cdot \underbrace{v}_{\text{eq-6}} \cdot \underbrace{(1 - hs_t^2)}_{\text{eq-7}} \cdot \underbrace{w}_{\text{eq-12}}} \quad \text{eq-13}$$

- softmaxed output or $\hat{y}$
- dhnext is the portion that propagates back to step-0. 0 to begin with.

- Apply gradients.