Convnet backpropagation

1. indexes

i,j : image pixel coordinates	from 1 to I,J
k : feature map	from 1 to K
l : other layer feature map	from 1 to L
m : dataset item	from 1 to M
n : layer	from 1 to N
p : layer size	from 1 to P
q : previous layer size	from 1 to Q
r,s : filter pixel coordinates	from 1 to R,S

2. variables

$X_{i,j}^{(m)}$ or $X_p^{(m)}$	dataset inputs
$y_p^{(m)}$	dataset targets
$a_{ij}^{(n)}$ or $a_p^{(n)}$	layer outputs (activation) $a^1_{ij} = X_{ij}; a^1_p = X_p$
$c_{ij}^{(n)(k)}$	convolution output for map k at layer n
$z_p = (\theta_{pq})^T * a_p + b_p$	weighted sum of inputs of neuron i
$\Theta_{pq}^{(n)}$	parameters weights of FC layer n+1, p=index in layer n, q=index in layer n, not including bias
$b^{(n)}$	bias at layer n for FC layer
$\delta^{m(n)}{}_p$	error at layer n for sample m and output p for FC layer
$\Theta_{rs}^{kl(n)}$	parameters weights of convnet at layer n+1 for computing feature maps k from F.M. l of layer n
$z_{ij}^{k(n)}$	net input to the activation function at layer n
$b^{k(n)}$	bias at layer n for map k connection to all maps of previous layer
$\delta^{k(n)}{}_{ij}$	error at layer n for map k and outputs i,j for image layer

3. functions

g(z)	activation function $a = g(z)$
$g(z) = \dfrac{1}{1+e^{-z}}$	sigmoïd activation function

$$g(z_i) = \frac{e^{z_i}}{\Sigma_{j=1}^{j=I} e^{z_j}}$$ softmax activation function

$$C(\Theta, X, y) = -\sum_{m=1}^{m=M} \sum_{k=1}^{k=K} 1(y^{(m)} = k) log \frac{e^{z_k^{(m)}}}{\Sigma_{j=1}^{j=I} e^{z_j^{(m)}}}$$ output layer cross-entropy cost function

4. forward propagation equations

4.1. FC layer

$$a^{(n+1)} = g(\theta^{(n)T} * a^{(n)} + b^{(n)})$$ neuron activation with bias $b^{(n)}$

4.2. Conv layer

"convolution" of data from feature map l into feature map k :

[a]

$$c_{ij}^{kl(n+1)} = \sum_{r=1}^{r=R^{(n)}} \sum_{s=1}^{s=S^{(n)}} a_{i+r-1,j+s-1}^{l(n)} \theta_{rs}^{kl(n)}$$

$i<=I^{(n)}-R+1, n>1, j<=J^{(n)}-S+1$ convolution output, with bias

[b]

$$z_{ij}^{k(n+1)} = b^{k(n)} + \sum_{l=1}^{l=L} c_{ij}^{kl(n+1)}$$ net input to the activation function

[c]

$$a_{ij}^{k(n+1)} = g(z_{ij}^{k(n+1)})$$ sigmoid activation before pooling

4.3 Pooling layer

max pooling layer

a(n) -> a(n+1),idx(n+1) idx records the relationship from (n) input space to (n+1) input space, idx() begin the function to find the index

[d]

$$a_{ij}^{k(n+1)} = pool(max, a_{ij}^{l(n)})$$ activation output for map k

no activation function

no parameters, so no gradient

5. backward propagation equations

5.1. FC layer

[e] $\delta_p^{m(N)} = a_p^{m(N)} - y_p^m$ error at the output layer

[f] $\delta^{m(n)} = \theta^{(n)T} * \delta^{m(n+1)}. * g'(z^{m(n)}); n < N$ error propagation

[g] $\Delta_{pq}^{(n)} = \Sigma_{m=1}^{m=M} \delta_p^{(n+1)m} * a_q^{(n)mT}$ layer to layer error summed on all samples

[h] $\frac{\partial}{\partial_{\Theta_{pq}^{(n)}}} J(\theta) = 1/M * \Delta_{pq}^{(n)}$ gradient between output i and input j at layer l

5.2. Conv layer

5.2.1. parameters

[3]

$$\frac{\partial C(\Theta, b, a_{i'\in 1..I^{(n+1)} j'\in 1..J^{(n+1)}}^{(n+1)k'\in 1..K^{(}n+1)})}{\partial_{\Theta_{ij}^{kl(n)}}} = \sum_{k'=1}^{k'=K^{(n+1)}} \sum_{i'=1}^{i'=I^{(n+1)}} \sum_{j'=1}^{j'=J^{(n+1)}} \frac{\partial C}{\partial z_{i'j'}^{k'(n+1)}} . \frac{\partial z_{i'j'}^{k'(n+1)}}{\partial_{\Theta_{ij}^{kl(n)}}}$$

[4] we define :

$$\boxed{\delta_{i'j'}^{k'(n+1)} = \frac{\partial C}{\partial z_{i'j'}^{k'(n+1)}}}$$

[5]

$$\frac{\partial C}{\partial_{\Theta_{ij}^{kl(n)}}} = \sum_{k'=1}^{k'=K^{(n+1)}} \sum_{i'=1}^{i'=I^{(n+1)}} \sum_{j'=1}^{j'=J^{(n+1)}} \delta_{i'j'}^{k'(n+1)} . \frac{\partial z_{i'j'}^{k'(n+1)}}{\partial_{\Theta_{ij}^{kl(n)}}}$$

[6]

$$\frac{\partial z_{i'j'}^{k'(n+1)}}{\partial_{\Theta_{ij}^{kl(n)}}} = \sum_{l'=1}^{l'=L} \frac{\partial c_{i'j'}^{k'l'(n+1)}}{\partial_{\Theta_{ij}^{kl(n)}}}$$

[7]

$$\frac{\partial z_{i'j'}^{k'(n+1)}}{\partial_{\Theta_{ij}^{kl(n)}}} = \sum_{l'=1}^{l=L} \sum_{r=1}^{r=R^{(n)}} \sum_{s=1}^{s=S^{(n)}} \frac{\partial (a_{i'+r-1, j'+s-1}^{l'(n)} \theta_{rs}^{k'l'(n)})}{\partial_{\Theta_{ij}^{kl(n)}}}$$

$i' <= I^{(n)} - R+1, n>1, j' <= J^{(n)} - S+1$

- bias term is zero since bias derivative wrt ϴ is zero
- derivative of $a^{(n)}$ wrt $\Theta^{(n)}$ is zero
- remaining term is $a^{(n)}$ times derivative of $\Theta^{(n)}$ which is 1 only if indexes are i,j,k,l are equal

[8]

$$\frac{\partial z_{i'j'}^{k'(n+1)}}{\partial_{\Theta_{ij}^{kl(n)}}} = a_{i'+i-1, j'+j-1}^{l(n)} . 1(k = k')$$

[8]+[5] -> [9] **computes gradient of parameters from error at level n+1**

$$\boxed{\frac{\partial C}{\partial_{\Theta_{ij}^{kl(n)}}} = \sum_{i'=1}^{i'=I^{(n+1)}} \sum_{j'=1}^{j'=J^{(n+1)}} \delta_{i'j'}^{k(n+1)}.a_{i'+i-1,j'+j-1}^{l(n)}}$$

and for i=j=$K^{(n)}$=l=1 where have :

$$\frac{\partial C}{\partial_{\Theta_{11}^{k(n)}}} = \sum_{i'=1}^{i'=I^{(n+1)}} \sum_{j'=1}^{j'=J^{(n+1)}} \delta_{i'j'}^{k(n+1)}.a_{i',j'}^{(n)} = \delta^{k(n+1)} * a^{(n)}$$

$$\frac{\partial C}{\partial_{\Theta_{11}^{k(n)}}} = \sum_{i'=1}^{i'=I^{(n+1)}} \sum_{j'=1}^{j'=J^{(n+1)}} \delta_{i'j'}^{k(n+1)}.a_{i',j'}^{(n)} = \delta^{k(n+1)} * a^{(n)}$$

(* =sum/product of a and delta)

[a] [b] -> [10]

$$\frac{\partial z_{i'j'}^{k(n+1)}}{\partial z_{ij}^{l(n)}} = \sum_{l'=1}^{l=L^{(n)}} \sum_{r=1}^{r=R^{(n)}} \sum_{s=1}^{s=S^{(n)}} \frac{\partial\left(a_{i'+r-1,j'+s-1}^{l'(n)} \theta_{rs}^{kl'(n)}\right)}{\partial z_{ij}^{l(n)}}$$

$i'<=I^{(n)}-R+1, n>1, j'<=J^{(n)}-S+1$

- like [7] bias is eliminated
- derivative of $\Theta^{(n)}$ wrt $z^{(n)}$ is zero
- remaining terms are derivative of $a^{(n)}$ wrt $z^{(n)}$ times $\Theta^{(n)}$
- z_{ij} is defined in coordinate space n-1

[11]

$$\frac{\partial z_{i'j'}^{k(n+1)}}{\partial z_{ij}^{l(n)}} = \sum_{l'=1}^{l=L^{(n)}} \sum_{r=1}^{r=R^{(n)}} \sum_{s=1}^{s=S^{(n)}} \theta_{rs}^{kl'(n)}.\frac{\partial\left(a_{i'+r-1,j'+s-1}^{l'(n)}\right)}{\partial z_{ij}^{l(n)}}$$

[12] let us compute derivative of a :

$$\frac{\partial\left(a_{i'j'}^{l'(n)}\right)}{\partial z_{ij}^{l(n)}} = g'\left(z_{ij}^{l(n)}\right)$$

- derivative of a wrt to z is 0 if not l=l'
- derivative of a wrt to z is 0 if not (i,j)=(i'',j'')
- i',j' are in coordinate space n
- i,j are coordinates in image space n

[13]

$$\frac{\partial\left(a_{i'j'}^{l'(n)}\right)}{\partial z_{ij}^{l(n)}} = 1\left(l = l', (i,j) = (i',j')\right) g'\left(z_{ij}^{l(n)}\right)$$

[14]

$$\frac{\partial z_{i'j'}^{k(n+1)}}{\partial z_{ij}^{l(n)}} = \sum_{l'=1}^{l=L^{(n)}} \sum_{r=1}^{r=R^{(n)}} \sum_{s=1}^{s=S^{(n)}}$$
$$\theta_{rs}^{kl'(n)}.1(l = l', (i,j) = (i'+r-1, j'+s-1))g'(z_{ij}^{l(n)})$$

[15]

$$\frac{\partial z_{i'j'}^{k(n+1)}}{\partial z_{ij}^{l(n)}} = \sum_{r=1}^{r=R^{(n)}} \sum_{s=1}^{s=S^{(n)}} \theta_{rs}^{kl(n)}.1((i,j) = (i'+r-1, j'+s-1))g'(z_{ij}^{l(n)})$$

i'+r-1=i -> r=i+1-i' s=j+1-j' -> [16]

$$\frac{\partial z_{i'j'}^{k(n+1)}}{\partial z_{ij}^{l(n)}} = \theta_{i-i'+1,j-j'+1}^{kl(n)}.g'(z_{ij}^{l(n)})$$

[17] **chain rule** applied to derivatives of cost function C yields

$$\frac{\partial C}{\partial z_{ij}^{l(n)}} = \sum_{i'=1}^{i'=I^{(n+1)}} \sum_{j'=1}^{j'=J^{(n+1)}} \sum_{k=1}^{k=K^{(n+1)}} \frac{\partial C}{\partial z_{i'j'}^{k(n+1)}} \frac{\partial z_{i'j'}^{k(n+1)}}{\partial z_{ij}^{l(n)}}$$

[16]+[17] -> [18] **recurrence formula for backpropagating errors per layer**

$$\boxed{\delta_{ij}^{l(n)} = \sum_{i'=1}^{i'=I^{(n+1)}} \sum_{j'=1}^{j'=J^{(n+1)}} \sum_{k=1}^{k=K^{(n+1)}} \delta_{i'j'}^{k(n+1)}.\theta_{i-i'+1,j-j'+1}^{kl(n)}.g'(z_{ij}^{l(n)})}$$

1<=i-i'+1<=R -> -R<=i'-i-1<=-1 -> i+1-R<=i'<=i

r=i-i'+1 -> i'=i-r+1

$$\boxed{\delta_{ij}^{l(n)} = \sum_{r=1}^{r=R^{(n)}} \sum_{s=1}^{s=S^{(n)}} \sum_{k=1}^{k=K^{(n+1)}} \delta_{i-r+1,j-s+1}^{k(n+1)}.\theta_{rs}^{kl(n)}.g'(z_{ij}^{l(n)})}$$

delta * theta type convolution, with reversed indexes, and using all indexes (not only valid)

Note:

if layer n+1 is FC then this formula uses $k=K^{(n+1)}=1$ and $I^{(n+1)}.J^{(n+1)}=P^{(n+1)}$

5.2.1. biases

[5] -> [20] by replacing derivation / Θ, by derivation / b

$$\frac{\partial C}{\partial_{b^{k(n)}}} = \sum_{k'=1}^{k'=K^{(n+1)}} \sum_{i'=1}^{i'=I^{(n+1)}} \sum_{j'=1}^{j'=J^{(n+1)}} \delta_{i'j'}^{k'(n+1)}.\frac{\partial z_{i'j'}^{k'(n+1)}}{\partial_{b^{k(n)}}}$$

[21]

$$\frac{\partial z_{i'j'}^{k'(n+1)}}{\partial_{b^{k(n)}}} = 1(k = k')$$

[20] + [21] -> [22]

$$\frac{\partial C}{\partial_{b^{kl(n)}}} = \sum_{k'=1}^{k'=K^{(n+1)}} \sum_{i'=1}^{i'=I^{(n+1)}} \sum_{j'=1}^{j'=J^{(n+1)}} \delta_{i'j'}^{k'(n+1)}.1(k = k')$$

[23] computes gradient of biases from error at level n+1, complements [9]

$$\boxed{\frac{\partial C}{\partial_{b^{kl(n)}}} = \sum_{i'=1}^{i'=I^{(n+1)}} \sum_{j'=1}^{j'=J^{(n+1)}} \delta_{i'j'}^{k(n+1)}}$$

5.3. pooling layer

5.2.1. upsampling

[0]

$$\delta_{i'j'}^{\prime m(n+1)} = \delta_{ij}^{m(n+1)} \text{ if } (i', j') = arg(max(d_{ij}^{(n+1)}))$$

$$\text{otherwise } \delta_{i'j'}^{\prime m(n+1)} = 0$$

max pooling upsampling

6. Vector expressions

back propagation of delta

$$\boxed{\delta_p^{m(N)} = a_p^{m(N)} - y_p^m}$$

at the output layer from [e]

in Octave : delta of size(K,M)
delta_next of size (K,M)
delta_next_img of size (I,J,K,M)

$$\boxed{\delta^{l(n)m} = \sum_{k=1}^{k=K^{(n+1)}m} (\delta^{k(n+1)m} * \theta^{kl(n)}).g'(z_{ij}^{l(n)m})}$$

where * is the regular convolution on indexes i-i',j-j'

recurrence of delta from [19]

$$\boxed{\frac{\partial C}{\partial_{\Theta_{ij}^{kl(n)}}} = \sum_{m=1}^{m=M} \sum_{k'=1}^{k'=K^{(n+1)}} \delta^{k'(n+1)m} * a^{l(n)m}}$$

gradient computation from [9]

this is almost the autocorrelation except that all indices of a are used