

# 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)  $a_{ij}^1 = X_{ij}; a_p^1 = X_p$   
 $c_{ij}^{(n)(k)}$  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  
  
 $\delta_p^{m(n)}$  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  
  
 $\delta_{ij}^{k(n)}$  error at layer n for map k and outputs i, j for image layer

## 3. functions

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

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

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

## 4. forward propagation equations

### 4.1. FC layer

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

### 4.2. Conv layer

“convolution” of data from feature map l into feature map k :

[a]

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

$$i \leq I^{(n)} - R + 1, n > 1, j \leq J^{(n)} - S + 1$$

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

$a(n) \rightarrow a(n+1), \text{idx}(n+1)$        $\text{idx}$  records the relationship from (n) input space to (n+1) input space,  $\text{idx}()$  begin the function to find the index

[d]

$$a_{ij}^{k(n+1)} = \text{pool}(\text{max}, a_{ij}^{l(n)})$$

activation output for map k

no activation function

no parameters, so no gradient

## 5. backward propagation equations

### 5.1. FC layer

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

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

[g]  $\Delta_{pq}^{(n)} = \sum_{m=1}^M \delta_p^{(n+1)m} * a_q^{(n)mT}$  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]

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

[4] we define :

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

[5]

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

[6]

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

[7]

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

$$i' \leq I^{(n)} - R + 1, n \geq 1, j' \leq J^{(n)} - S + 1$$

- bias term is zero since bias derivative wrt  $\Theta$  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]

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

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

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

[a] [b] -> [10]

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

$$i' \leq I^{(n)} - R + 1, j' \leq J^{(n)} - S + 1$$

- 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]

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

[12] let us compute derivative of a :

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

- 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]

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

[14]

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

[15]

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

$i' + r - 1 = i \rightarrow r = i + 1 - i'$   $s = j + 1 - j' \rightarrow$  [16]

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

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

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

[16]+[17]  $\rightarrow$  [18] recurrence formula for backpropagating errors per layer

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

$1 \leq i - i' + 1 \leq R \rightarrow -R \leq i' - i - 1 \leq -1 \rightarrow i + 1 - R \leq i' \leq i$

$r = i - i' + 1 \rightarrow i' = i - r + 1$

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

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  $I^{(n+1)} \cdot J^{(n+1)} = P^{(n+1)}$

### 5.2.1. biases

[5]  $\rightarrow$  [20] by replacing derivation /  $\Theta$ , by derivation / b

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

[21]

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

[20] + [21]  $\rightarrow$  [22]

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

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

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

## 5.3. pooling layer

### 5.2.1. upsampling

[0]

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

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

max pooling upsampling

## 6. Vector expressions

back propagation of delta

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

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)

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

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

recurrence of delta from [19]

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

gradient computation from [9]

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