Convolution

Input is of dimension (C, H, W), each filter is of size (C, HH, WW), output responding to the filter is of size (H_out, W_out). X_out refers to the output of convo layer, input of next layer. X_in is the input of convo layer.

Output at a specific position

$$out = \sum_{c \in C} \sum_{i \in HH} \sum_{j \in WW} X\_in_{cij} * W_{cij} + b$$

For each filter, the weight affects all the value in the output layer. According to chain rule,

$$\begin{aligned} \frac{dError}{dw_{mn}} &= \sum_{i \in H_{out}} \sum_{j \in W_{out}} \frac{dError}{dX\_out_{ij}} * \frac{dX\_out_{ij}}{dw_{mn}} \\ &= \sum_{i \in H_{out}} \sum_{j \in W_{out}} dout_{ij} * dX\_in_{(i*stride+m)(j*stride+n)} \end{aligned}$$

$$\begin{aligned} \frac{dError}{db} &= \sum_{i \in H_{out}} \sum_{j \in W_{out}} \frac{dError}{dX\_out_{ij}} * \frac{dX\_out_{ij}}{db} \\ &= \sum_{i \in H_{out}} \sum_{j \in W_{out}} dout_{ij} \end{aligned}$$

Each filter can be calculated independently and with batch processing, it means aggregation of gradient.

Max pooling

Forward of max pooling is simple, the output is the max of all the value in receptive field.

If the input is not maximum in the receptive field, the gradient is 0 since it does not contribute to the output. If the node input is the maximum, its gradient is equal to the responding dout.

Dropout

During forward propagation, if a node is dropped out, output is zero. Otherwise the output is equal to 1/(1-p) * input.

If the node is dropped, its gradient is 0 since it does not contribute to the output. Else the gradient is equal to 1/(1-p) * dout.

Back propagation with regularization

The gradient of regularization can be calculated independently.

$$Error = Softmax\_Error + regularization\_error$$

$$\frac{dError}{dw} = \frac{dsoftmax\_error}{dw} + \frac{dregularization\_error}{dw}$$

$$\frac{dregularization\_error}{dw} = reg * w$$

$$\frac{dregularization\_error}{db} = 0$$

First part can be calculated using the formula above.

Results

Each epoch contains 200 iterations.
Accuracy for reg = 0 and dropout rate = 0

Epoch	0	1	2	3	4	5	6	7
Training	0.514000	0.834000	0.872000	0.863000	0.856000	0.888000	0.885000	0.900000
Validation	0.525000	0.825000	0.860000	0.870000	0.890000	0.880000	0.890000	0.890000

Accuracy for reg=0.01 and dropout rate = 0

Epoch	0	1	2	3	4	5	6	7
Training	0.496000	0.833000	0.811000	0.842000	0.895000	0.860000	0.900000	0.873000
Validation	0.575000	0.845000	0.815000	0.820000	0.855000	0.825000	0.865000	0.855000

No obvious improvement is observed with reg set to 0.01. This is because the regularization parameter is not large enough to avoid overfitting.

Accuracy for reg=0.01 and dropout rate = 0.4

Epoch	0	1	2	3	4	5	6
Training	0.506000	0.842000	0.840000	0.887000	0.858000	0.875000	0.906000
Validation	0.470000	0.865000	0.880000	0.890000	0.895000	0.870000	0.895000

With dropout, the overfit problem is greatly alleviated. The reason is that with dropout, a node has to learn more robust features and therefore improves the performance of the network.