CS 4670 PA5 Written Part

Assigned April 28, 2016

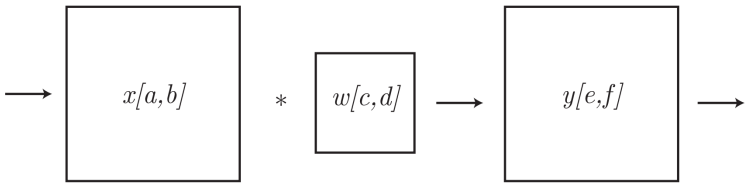
Background. In this problem, we will derive an update rule for a convolutional neural network (ConvNet). A neural network consists of a number of layers, each layer having parameters w. One type of layer is a "convolutional" layer that convolves an input x with a set of weights w to produce an output y:

$$y = x \star w \tag{1}$$

Here, x is a 2D grayscale image, w is a 2D array of weights, and y is an output 2D grayscale image. We will assume zero padding for all convolution. We can write this as an equation (the definition of convolution):

$$y[e, f] = \sum_a \sum_b x[a, b] \cdot w[e - a, f - b] \tag{2}$$

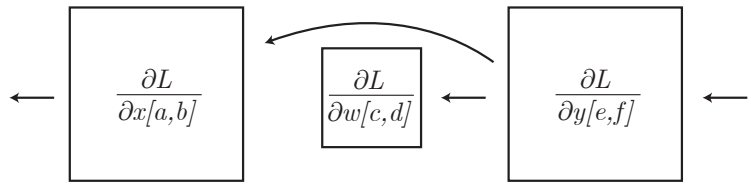
or we can write it as a block diagram:



The block diagram is useful to visualize the relative sizes of each layer. To avoid confusing indices, we will index the input as $x[a, b]$, the weights as $w[c, d]$, and the output as $y[e, f]$.

Backpropagation. To "learn" the weights for a ConvNet, it is necessary to compute the gradient of some loss function L with respect to the weights inside the network. L measures "how good" the current weights are, and we can use the gradient $\frac{\partial L}{\partial w}$ to update the weights to be slightly better (gradient descent).

The goal of backpropagation is to calculate $\frac{\partial L}{\partial x[a,b]}$ and $\frac{\partial L}{\partial w[c,d]}$ in terms of $\frac{\partial L}{\partial y[e,f]}$, where L is some function that depends on y (and y depends on x and w). Note that x is a 2D image, so $\frac{\partial L}{\partial x}$ is a 2D image of the same size (and similarly for w and x). We can write this as a block diagram to see the sizes of each image:



Note that we don't need to know L, because we can use the chain rule to relate partial derivatives together:

$$\frac{\partial L}{\partial x[a, b]} = \sum_e \sum_f \frac{\partial L}{\partial y[e, f]} \cdot \frac{\partial y[e, f]}{\partial x[a, b]} \tag{3}$$

Computing y is called the "forward" pass (because the computation proceeds left to right) and computing $\frac{\partial L}{\partial x}$ and $\frac{\partial L}{\partial w}$ is the "backward pass" because it proceeds right to left.

Note: An answer with no derivation will receive no credit.

A. Convolution layer.

1. For a convolution layer that computes

$$y = x \star w$$

Show that the backwards pass is the same as the forwards pass, but with a weight kernel that is flipped vertically and horizontally. Specifically, derive the gradient with respect to the input x and show that it is:

$$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial y} \star \tilde{w}$$

where $\tilde{w}$ is a flipped version of w both vertically and horizontally: $\tilde{w}[c, d] = w[-c, -d]$.

Hint: start from Equation 3.

2. Similarly, show that the gradient with respect to the weights w is also a flipped convolution:

$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial y} \star \tilde{x}$$

where $\tilde{x}$ is a flipped version of x: $\tilde{x}[a, b] = x[-a, -b]$.

Hint: remember that $x \star w = w \star x$.

B. Fully connected layer.

1. Consider another type of layer that computes matrix multiplication:

$$y = Wx$$

where x is an input 1D column vector, W is a 2D weight matrix, and y is an output 1D column vector. We call this "fully connected" because every entry in y depends on every entry in x.

Derive the backward pass for this layer and write your answer in the form:

$$\frac{\partial L}{\partial x} = A^T B$$

Hint 1: convert all matrix expressions to operations on scalars (e.g., write out the definition of the matrix product), manipulate everything as scalars, and then convert back to matrix expressions in the last step.

Hint 2: here is the chain rule for $\frac{\partial L}{\partial x}$:

$$\frac{\partial L}{\partial x_i} = \sum_j \frac{\partial L}{\partial y_j} \frac{\partial y_j}{\partial x_i}$$

Hint 3: remember that matrix multiplication is defined as:

$$E = CD$$
$$E_{ij} = \sum_k C_{ik} D_{kj}$$

where C, D, E are matrices.

Hint 4: remember that $(A^T)_{ij} = A_{ji}$.

Note: While you do not need to derive this, the derivation for the weight gradient ($\frac{\partial L}{\partial W}$) is similar to this derivation.

C. Softmax Loss layer.

1. Consider a Softmax Loss layer (softmax followed by cross-entropy loss). For a single input example, the loss L is given by:

$$L = -\log p_y = -\sum_i t_i \log p_i$$

(using the natural logarithm).

The "target" probability distribution t encodes the ground truth y as a "one hot" vector:

$$t = [0\ 0 \ldots 1 \ldots 0]$$

with entry y set to 1, and M total entries. For example, if we had $M = 4$ categories and $y = 1$, then $t = [0\ 1\ 0\ 0]$.

The predicted probability distribution p is computed from the raw category scores s, using the softmax function:

$$p_i = \frac{e^{s_i}}{\sum_j e^{s_j}}$$

Show that the backwards pass for this layer is:

$$\frac{\partial L}{\partial s} = p - t$$

Hint 1: Use back-propagation methods to simplify the derivation. Rather than plug the expression for p into L, which can get messy, keep the derivation simple by using the chain rule:

$$\frac{\partial L}{\partial s_i} = \sum_k \frac{\partial L}{\partial p_k} \frac{\partial p_k}{\partial s_i} \tag{4}$$

Hint 2: In the sum above (Equation 4), first consider which terms are nonzero.