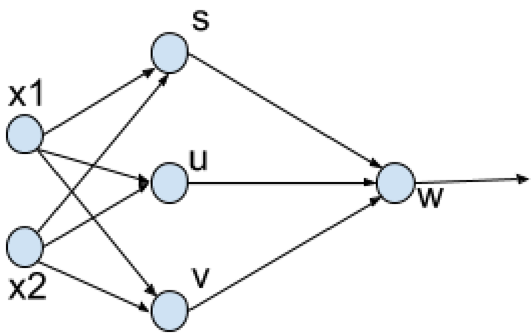
In this document we will study how to train a convolutional neural network. First we see how to train a single layer neural network shown below.



The loss is given by $f = ((w_1, w_2, w_3)^T(\sigma(s^T x), \sigma(u^T x), \sigma(v^T x)) - y)^2$ where **σ**(x) is an activation function such as sigmoid or relu. In this document we let **σ**(x) be the sigmoid activation: $\sigma(x) = 1/(1 + e^{-x})$.

We optimize the loss f with gradient descent by initializing all weights to random. We then update each weight w with $w = w - \eta df/dw$ until the loss converges. This is the same as moving the weight vector in the direction of the negative gradient which gives the optimal direction to decrease the objective.

We call ***η*** the learning rate. This is usually a small value such as 0.1 when the search starts and we change it to a smaller value as the number of epochs proceed. In other words we want to take smaller step sizes as we approach the local minima.

Thu we need only first derivatives to optimize a neural network's parameters to reach a local minima.

In order to calculate the update equations let $z_1 = \sigma(s^T x) = \sigma(s_1 x_1 + s_2 x_2)$. This means I can write f as $f = ((w_1, w_2, w_3)^T(z_1, z_2, z_3) - y)^2$. Then

$df/dw_1 = 2\sqrt{(f)}z_1$

$df/ds_1 = (df/dz_1)(dz_1/ds_1)$

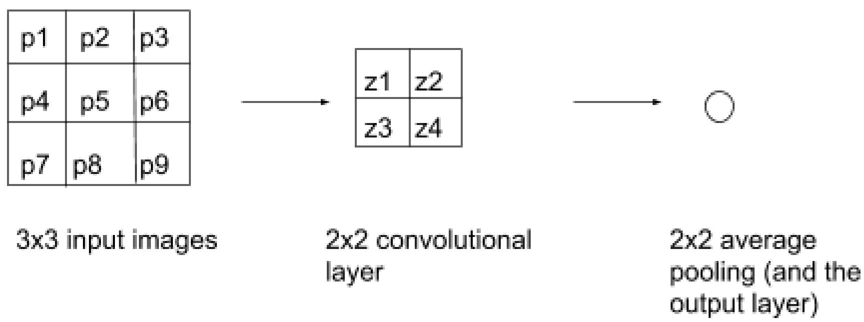
where

$df/dz_1 = 2\sqrt{(f)}w_1$ and

$dz_1/ds_1 = \sigma(s^T x)(1 - \sigma(s^T x))x_1$ since $d\sigma/df(x) = \sigma(f(x))(1 - \sigma(f(x))df/dx$

In the same way we calculate the update weights for the other parameters $w_2, w_3, s_2, u_1, u_2, v_1, v_2$.

Now consider a simple convolutional neural network shown below. In this network our input images are 3x3 and we have one 2x2 convolutional layer with average pooling



We define the loss as the squared difference between the final layer and desired output:

$$f = ((z_1 + z_2 + z_3 + z_4)/4 - y)^2$$

We optimize this in the same way as we do a single layer network. We start with random weights and update each with the gradient (first derivatives). Let our convolutional filter be

c1	c2
c3	c4

Then $z_1 = \sigma(c_1 p_1 + c_2 p_2 + c_3 p_4 + c_4 p_5)$ where $\boldsymbol{\sigma}$(x) is the sigmoid activation. We also have $z_2 = \sigma(c_1 p_2 + c_2 p_3 + c_3 p_5 + c_4 p_6)$.

Note that f is actually a function of c1,c2,c3, and c4. Therefore I can write f as

$$f = ((\sigma(c_1 p_1 + c_2 p_2 + c_3 p_4 + c_4 p_5) + \sigma(c_1 p_2 + c_2 p_3 + c_3 p_5 + c_4 p_6) +$$
$$\sigma(c_1 p_4 + c_2 p_5 + c_3 p_7 + c_4 p_8) + \sigma(c_1 p_5 + c_2 p_6 + c_3 p_8 + c_4 p_9))/4 - y)^2$$

We then have

$$df/dc_1 = \sqrt{f}/2(dz_1/dc_1 + dz_2/dc_1 + dz_3/dc_1 + dz_4/dc_1)$$

where

$dz_1/dc_1 = \sigma(c_1p_1 + c_2p_2 + c_3p_4 + c_4p_5)(1 - \sigma(c_1p_1 + c_2p_2 + c_3p_4 + c_4p_5))p_1$

$dz_2/dc_1 = \sigma(c_1p_2 + c_2p_3 + c_3p_5 + c_4p_6)(1 - \sigma(c_1p_2 + c_2p_3 + c_3p_5 + c_4p_6))p_2$

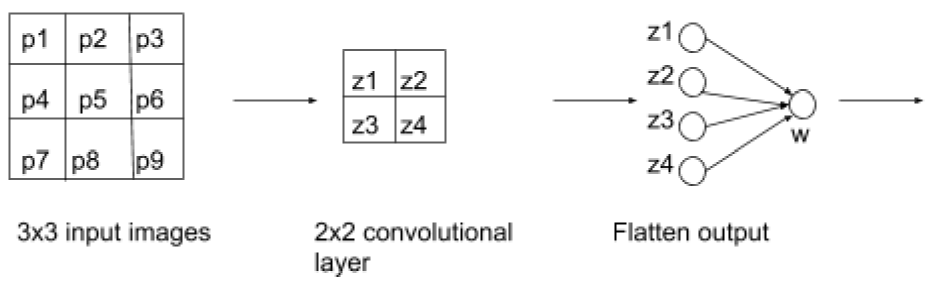
$dz_3/dc_1 = \sigma(c_1p_4 + c_2p_5 + c_3p_7 + c_4p_8)(1 - \sigma(c_1p_4 + c_2p_5 + c_3p_7 + c_4p_8))p_4$

$dz_4/dc_1 = \sigma(c_1p_5 + c_2p_6 + c_3p_8 + c_4p_9)(1 - \sigma(c_1p_5 + c_2p_6 + c_3p_8 + c_4p_9))p_5$

Similarly we calculate the gradient updates for parameters c_2, c_3, c_4.

Exercises:

1. Calculate the gradient update equations for the network below. Instead of average pooling we flatten the output of the convolution and give it to a linear classifier w. First write the loss function and then calculate first derivatives. Here w=(w1,w2,w3,w4) is a four dimensional vector.



What is the loss function?
Solution:

We start with the loss for the simpler network where we average the outputs:

$$f = ((\sigma(c_1p_1 + c_2p_2 + c_3p_4 + c_4p_5) + \sigma(c_1p_2 + c_2p_3 + c_3p_5 + c_4p_6) + \sigma(c_1p_4 + c_2p_5 + c_3p_7 + c_4p_8) + \sigma(c_1p_5 + c_2p_6 + c_3p_8 + c_4p_9))/4 - y)^2$$

We modify this for the new network in this exercise

$$f = ((z_1, z_2, z_3, z_5)^T(w_1, w_2, w_3, w_4) - y)^2$$

Now we need update equations. Writing out the loss in terms of the variables we see

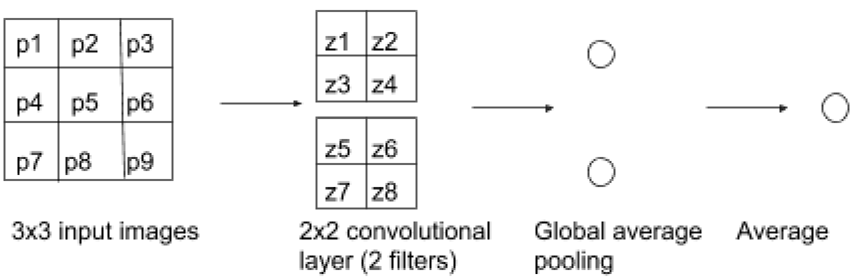
$$f = (\sigma(c_1 p_1 + c_2 p_2 + c_3 p_4 + c_4 p_5) w_1 \ + \sigma(c_1 p_2 + c_2 p_3 + c_3 p_5 + c_4 p_6) w_2 \ + \\ \sigma(c_1 p_4 + c_2 p_5 + c_3 p_7 + c_4 p_8) w_3 \ + \ \sigma(c_1 p_5 + c_2 p_6 + c_3 p_8 + c_4 p_9) w_4 \ - \ y)^2$$

$$df/dc_1 = s\sqrt{f}(w_1 \sigma(c_1 p_1 + c_2 p_2 + c_3 p_4 + c_4 p_5)(1 - \sigma(c_1 p_1 + c_2 p_2 + c_3 p_4 + c_4 p_5))p_1 + \\ w_2 \sigma(c_1 p_2 + c_2 p_3 + c_3 p_5 + c_4 p_6)(1 - \sigma(c_1 p_2 + c_2 p_3 + c_3 p_5 + c_4 p_6))p_2 + \\ w_3 \sigma(c_1 p_4 + c_2 p_5 + c_3 p_7 + c_4 p_8)(1 - \sigma(c_1 p_4 + c_2 p_5 + c_3 p_7 + c_4 p_8))p_4 + \\ w_4 \sigma(c_1 p_5 + c_2 p_6 + c_3 p_8 + c_4 p_9)(1 - \sigma(c_1 p_5 + c_2 p_6 + c_3 p_8 + c_4 p_9))p_5)$$

$$df/dw_1 = 2\sqrt{f}\sigma(c_1 p_1 + c_2 p_2 + c_3 p_4 + c_4 p_5)$$

Similarly we can calculate the updates for the other variables.

2. Calculate the gradient update equations if the network has two 2x2 convolutional filters as shown below. The output of each filter is averaged and then averaged again.



3. Calculate the gradient update equations if the network has two 2x2 convolutional filters as shown below. The output of each filter is averaged and given to a liner classifier w=(w1,w2).