Anthony Poerio (adp59@pitt.edu)

**CS1674: Homework 11 - Written**

**Due:** 12/7/2016, 11:59pm 

**PART I**

1. **Compute network activations using fixed input and weights.**

First, we need to compute Z2. This can be done with the following formula (after expanding the summation and removing the bias value):

**a2** = (w(1)21 \* x1) + (w(1)22 \* x2) + (w(1)23 \* x3) + (w(1)24 \* x4)

**a2** = (0.02 \* x1) + (0.25 \* x2) + (0.4 \* x3) + (0.3 \* x4)

**a2** = (0.02 \*10) + (0.25 \* 1) + (0.4 \* 2) + (0.3 \* 3)

**a2** = 2.15

Next, perform tanh activation.

**Z2** = tanh(**a2**)

**Z2** = tanh(**2.15**)

**Z2 = 0.9732**

**PART II**

**What is the output size resulting from convolving a 35x35 image with a filter of 15x15, using:**

* 1. **Stride 1 and no padding**
     1. Formula to use: (N-F) / stride + 1
  + N=35
  + F=15
  + Stride=1
  + Padding=0
  + Formula = (35-15) /1 + 1
    - 20/1 + 1 = 21
      * Therefore, **filter size = 21x21**
  1. **Stride 1 and padding 1**
     1. Formula to use: (N-F) / stride + 1
  + N=35
  + F=15
  + Stride=1
  + Padding = 2 dimensions \* 1 px = (2\*1)
  + Formula = ( (35+(2\*1))-15) /1 + 1
    - 22/1 + 1 = 23
      * Therefore, **filter size = 23x23**
  1. **Stride 2 and padding 3** 
     1. Formula to use: (N-F) / stride + 1
  + N=35
  + F=15
  + Stride=1
  + Padding = 2 dimensions \* 3 px = (2\*3)
  + Formula = ( (35+(2\*3))-15) /1 + 1
    - (41-15)/1 + 1
    - 26/1 + 1 = 27
      * Therefore, **filter size = 27x27**

**PART III**

**Computing outputs of convolutions**

* 1. **First, show the output of applying convolution**

To start, let’s figure out the dimensions of our resulting matrix. The formula to use is: (N-F) / Stride + 1

And we know this about the data set:

* + Padding=0
  + Stride=0
  + N=9
  + F=3
    - Therefore:
      * = (N-F) / Stride + 1
      * = (9-3)/2 + 1
      * = 6/2 + 1
      * = 3 + 1
      * = 4
      * **= 4x4 matrix**

With this in mind, we can calculate the values for each cell of our matrix, given our **Image** and **Filter**, as specified by the assignment prompt.

Calculating by hand, I got:

[-2 -2 -1 0; 0 0 -3 -1; 0 0 0 -3; 0 0 0 0]

Note, that I am doing a convolution by hand, using the formula on slide 26, here:

<https://people.cs.pitt.edu/~kovashka/cs1674/vision_04_filters_texture.pdf>

* 1. **Second, show the output of applying a Rectified Linear Unit (ReLU) activation**

Next, we apply a ReLU activation, which yields a matrix of all 0’s. More specifically, our matrix becomes: [0 0 0 0; 0 0 0 0; 0 0 0 0; 0 0 0 0]

This is because ReLU transforms any value that is <= 0 to 0.

And, all of our values in the convolution matrix produced in part (a) are <= 0.

* 1. **Third, show the output of applying max pooling over 2x2 regions**

This yields a matrix with almost all 1’s and only one 0, in the top rightmost cell:

[1 1 1 0; 1 1 1 1; 1 1 1 1; 1 1 1 1]

This is because max-pooling takes the MAX value from each region. And the only values in our matrix are 0’s and 1’s. AND 🡪 the top rightmost 2x2 region is the only one that does not have a 1 contained in the 2x2 filter.

**PART IV**

**Compute 2 types of loss functions: SVM and Softmax. Use three sets of weights W, each will result in a different set of scores, S 🡪 for 4 image examples. Goal is: determine which set of weights results in the smallest SVM loss, and which set results in the smallest Softmax loss.**

* + 1. **SVM\_LOSS,** Results are:
* W1\_Loss=6.5676
* **W2\_Loss=4.8363 🡪 W2 is has the SMALLEST loss.**
* W3\_Loss=7.7445
  + 1. **SOFTMAX\_LOSS,** Results are (all ended up negative):
* **W1\_Loss= -22.1060 🡪 W1 is has the SMALLEST loss.**
* W2\_Loss= -7.6050
* W3\_Loss= -19.1920

**PART V**

**Compute the numerical gradient of W1.**

h=0.0001 yields this gradient vector for me:

G =

1.0e+03 \*

-7.0926

-7.5459

-2.7593

-6.7960

-6.5500

-1.6251

-1.1890

-4.9826

-9.5964

-3.4029

-5.8517

-2.2371

-7.5117

-2.5500

-5.0586

-6.9898

-8.9080

-9.5919

-5.4712

-1.3852

-1.4919

-2.5741

-8.4062

-2.5418

-8.1418

-2.4342

-9.2916

-3.4988

-1.9650

-2.5098

-6.1594

-4.7319

-3.5156

-8.3073

-5.8516

-5.4962

-9.1709

-2.8574

-7.5710

-7.5363

-3.8035

-5.6772

-0.7575

-0.5385

-5.3070

-7.7907

-9.3391

-1.2981

-5.6872

-4.6929

-0.1180

-3.3702

-1.6208

-7.9418

-3.1112

-5.2843

-1.6555

-6.0188

-2.6287

-6.5398

-6.8911

-7.4805

-4.5044

-0.8372

-2.2888

-9.1324

-1.5228

-8.2572

-5.3824

-9.9603

-0.7808

-4.4258

-1.0655

-9.6180

-0.0453

-7.7481

-8.1720

-8.6859

-0.8434

-3.9968

-2.5977

-7.9997

-4.3131

-9.1055

-1.8175

-2.6370

-1.4544

-1.3597

-8.6919

-5.7960

-5.4976

-1.4485

-8.5293

-6.2196

-3.5085

-5.1315

-4.0171

-0.7587

-2.3982

-1.2322

And computing the SVM Loss on each example yields these loss values for me:

* Loss\_X1 = 1.2224e+03
* Loss\_X2 = 0
* Loss\_X3 = 1.4240e+04
* Loss\_X4 = 8.8725e+03

**Then, after updating the data with h=0.001, I see the following results.**

* **Loss\_X1 = 1.3312**
* **Loss\_X2 = 0.6210**
* **Loss\_X3 = 0.0966**
* **Loss\_X4 = 1.0404**