Midterm for CSC321, Intro to Neural Networks
Winter 2017, night section
Tuesday, Feb. 28, 6:10-7pm

Name: ______________________________

Student number: ____________________

This is a closed-book test. It is marked out of 15 marks. Please answer ALL of the questions. Here is some advice:

- The questions are NOT arranged in order of difficulty, so you should attempt every question.
- Questions that ask you to "briefly explain" something only require short (1-3 sentence) explanations. Don't write a full page of text. We're just looking for the main idea.
- None of the questions require long derivations. If you find yourself plugging through lots of equations, consider giving less detail or moving on to the next question.
- Many questions have more than one right answer.

Final mark: ______ / 15

1. Consider a convolution layer. The input consists of 6 feature maps of size 20×20. The output consists of 8 feature maps, and the filters are of size 5×5. The convolution is done with a stride of 2 and zero padding, so the output feature maps are of size 10×10.

   For both parts, you can leave your expression as a product of integers; you do not need to actually compute the product. You do not need to show your work, but doing so can help you receive partial credit.

   (a) **[1pt]** Determine the number of weights in this convolution layer.

   (b) **[1pt]** Now suppose we made this a fully connected layer, but where the number of input and output units are kept the same as in the network described above. Determine the number of weights in this layer.

2. **[2pts]** Recall that the learning rate is an example of a hyperparameter which must be tuned. Alice wants to tune the learning rate by doing a grid search over values and choosing the one which achieves the lowest training error. Bob tells her it's important to tune all hyperparameters on a separate validation set. Who is right? Justify your answer.

   Note that there are good arguments for either side, so you will receive full credit as long as you justify your answer well.

3. **[2pts]** Your job is to design a multilayer perceptron which receives three binary-valued (i.e. 0 or 1) inputs x_1, x_2, x_3, and outputs 1 if exactly two of the inputs are 1, and outputs 0 otherwise. All of the units use a hard threshold activation function:

$$z = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{if } z < 0 \end{cases}$$

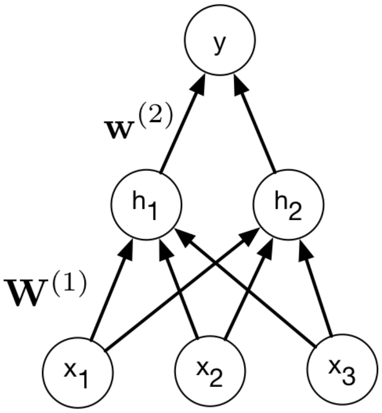
   Specify weights and biases which correctly implement this function. You do not need to explain your solution. *Hint: one of the hidden units should activate if 2 or more inputs are on, and the other should activate if all of the inputs are on.*

$$\mathbf{W}^{(1)} = \underbrace{\begin{pmatrix} \quad & \quad & \quad \\ \quad & \quad & \quad \end{pmatrix}}_{2\times 3 \text{ matrix}}$$

$$\mathbf{b}^{(1)} = (\qquad\qquad)$$

$$\mathbf{w}^{(2)} = (\qquad\qquad)$$

$$b^{(2)} =$$



4. **[3pts]** You want to train the following model using gradient descent. Here, the input x and target t are both scalar-valued.

$$\begin{aligned} z &= w_0 + w_1 x + w_2 x^2 \\ y &= 1 + e^z \\ \mathcal{L} &= \frac{1}{2}(\log y - \log t)^2. \end{aligned}$$

Determine the backprop rules which will let you compute the loss derivative $\partial \mathcal{L} / \partial w_2$.

Your equations should refer to previously computed values (e.g. your formula for $\overline{z}$ should be a function of $\overline{y}$). You do not need to show your work, but it may help you get partial credit. The dummy step has been filled in for you.

$$\overline{\mathcal{L}} = 1$$

$$\overline{y} =$$

$$\overline{z} =$$

$$\overline{w_2} =$$

5. **[2pts]** Suppose we are training a linear regression model using gradient descent with momentum. The update rules are as follows:

$$
\begin{aligned}
p_j &\leftarrow \mu p_j - \frac{\alpha}{N} \sum_{i=1}^{N} x_j^{(i)} (y^{(i)} - t^{(i)}) \\
w_j &\leftarrow w_j + p_j
\end{aligned}
$$

Now suppose that, as usual, the inputs are stored as an $N \times D$ matrix $\mathbf{X}$, where N is the number of data points and D is the input dimension. The targets and predictions are represented as N-dimensional vectors $\mathbf{t}$ and $\mathbf{y}$, respectively. The weights and momentum vector are represented as D-dimensional vectors $\mathbf{w}$ and $\mathbf{p}$, respectively. Write the vectorized form of these update rules, i.e. mathematical expressions which could be translated into NumPy without requiring `for`-loops.

You may assume $\mathbf{y}$ has already been computed. You do not need to show your work, though it may help you receive partial credit.

$$\mathbf{p} \leftarrow$$

$$\mathbf{w} \leftarrow$$

6. **[1pt]** Write the formula for the logistic function $\sigma(z)$. You do not need to justify your answer or explain anything.

$$\sigma(z) =$$

7. **[1pt]** Recall that the perceptron algorithm cycles through the training examples, applying the following rule:

```
$z^{(i)} \leftarrow \mathbf{w}^T \mathbf{x}^{(i)}$
If $z^{(i)} t^{(i)} \leq 0$,
    $\mathbf{w} \leftarrow \mathbf{w} + t^{(i)} \mathbf{x}^{(i)}$
```

(Recall that the targets take values in $\{-1, 1\}$.) Suppose we make the inequality strict in the conditional, i.e. we update the weights only if $z^{(i)} t^{(i)} < 0$. What would go wrong? You may assume the weights are initialized to 0. *Hint: what happens on the first training example?*

8. **[2pts]** Suppose the word representations $\mathbf{r}_1$ and $\mathbf{r}_2$ are both unit vectors. Show that the Euclidean distance $\|\mathbf{r}_1 - \mathbf{r}_2\|$ is a monotonically decreasing function of the dot product $\mathbf{r}_1^\top \mathbf{r}_2$. *Hint: start by expanding out the formula for squared Euclidean distance.*