UNIVERSITY OF TORONTO
Faculty of Arts and Science
APRIL 2018 EXAMINATIONS
CSC321H1S

Duration — 3 hours
No Aids Allowed

Name: ______________________________

Student number: ______________________

This is a closed-book test. It is marked out of 35 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 "justify your answer" or "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.

.

Q1:	______	/ 2
Q2:	______	/ 1
Q3:	______	/ 1
Q4:	______	/ 2
Q5:	______	/ 2
Q6:	______	/ 1
Q7:	______	/ 3
Q8:	______	/ 2
Q9:	______	/ 1
Q10:	______	/ 2
Q11:	______	/ 4
Q12:	______	/ 2
Q13:	______	/ 2
Q14:	______	/ 2
Q15:	______	/ 2
Q16:	______	/ 3
Q17:	______	/ 2
Q18:	______	/ 1

Final mark: ______ / 35

1. **[2pts]** Recall that multilayer perceptrons are universal for the set of functions mapping binary-valued input vectors to binary valued outputs.

   (a) **[1pt]** What do we mean by universal?

   (b) **[1pt]** If multilayer perceptrons are universal, why do we still consider other architectures?

2. **[1pt]** Give an example of a data augmentation technique that would be useful for classifying images of cats vs. dogs, but not for classifying handwritten digits. Briefly explain your answer.

3. **[1pt]** Suppose we have a grayscale image represented as an array, where larger values denote lighter pixels. What is the effect when we convolve it with the following kernel?

$$\begin{pmatrix} 0 & -1 & 0 \\ -1 & -4 & -1 \\ 0 & -1 & 0 \end{pmatrix}$$

4. **[2pts]** The learning rate is an important parameter for gradient descent.

   (a) **[1pt]** Briefly describe something that can go wrong if we choose too high a learning rate for *full batch* gradient descent.

   (b) **[1pt]** Briefly describe something that can go wrong if we choose too high a learning rate for *stochastic* gradient descent, but is not a problem in the full batch setting.

5. **[2pts]** Suppose we are training an RNN language model using teacher forcing (the method you implemented in Assignment 3).

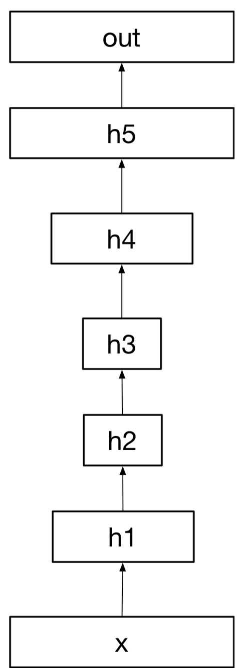
   (a) **[1pt]** What are the inputs to the network at training time?

   (b) **[1pt]** What are the inputs to the network at test time?

6. [**1pt**] Here is a modified version of code from Programming Assignment 2. The methods `downconv1`, `rfconv`, etc. implement convolution layers. Add edges to the diagram to represent the network architecture this implements. You don't need to justify your answer.

```
class MyNet(nn.Module):
    ...

    def forward(self, x):
        self.h1 = self.downconv1(x)
        self.h2 = self.downconv2(self.h1)
        self.h3 = self.rfconv(self.h2)
        self.h4 = self.upconv1(torch.cat([self.h3, self.h2], 1))
        self.h5 = self.upconv2(self.h4)
        self.out = self.finalconv(torch.cat([self.h5, x], 1))
        return self.out
```



7. **[3pts]** Recall the (multivariate) linear regression model:

$$y = \mathbf{w}^\top \mathbf{x} + b$$
$$\mathcal{L}(y, t) = \frac{1}{2}(y - t)^2$$

Your job is to implement full batch gradient descent in NumPy. In particular, suppose we are given an $N \times D$ NumPy array `X` representing all the training inputs, and an N dimensional NumPy vector `t` representing the targets, where N is the number of data points, and D is the input dimension. The weights are represented with a D-dimensional NumPy vector `w`, and the biases are represented with a scalar `b`. The learning rate is given as `alpha`.

Write NumPy code which implements one iteration of batch gradient descent. It should be vectorized, i.e. it should not involve a `for`-loop. You don't need to show your work, but doing so may help you get partial credit.

8. **[2pts]** Suppose we have a convolution layer which takes as input an array $\mathbf{x} = \begin{pmatrix} x_1 & x_2 & x_3 \end{pmatrix}$ and convolves $\mathbf{x}$ with the kernel $\begin{pmatrix} 2 & -1 \end{pmatrix}$. This layer has a linear activation function. The output is an array of length 4.

   Now let's design a fully connected layer which computes the same function. It has a linear activation function and no bias, so it computes $\mathbf{y} = \mathbf{W}\mathbf{x}$, where the output $\mathbf{y}$ is a vector of length 4. Give the 4×3 weight matrix $\mathbf{W}$ which makes this fully connected layer equivalent to the convolution layer above. You don't need to justify your answer, but doing so may help you get partial credit.

   Hint: first write the values of each output as a linear function of the inputs. To help you check your work, if $\mathbf{x} = \begin{pmatrix} 1 & 2 & 3 \end{pmatrix}$, your answer should give $\mathbf{y} = \begin{pmatrix} 2 & 3 & 4 & -3 \end{pmatrix}$.

9. **[1pt]** Briefly explain one flaw of encoder-decoder architectures for machine translation which do not use attention, and how attention can fix it.

10. **[2pts]** Recall that in order to add a new primitive operation to Autograd, you need to define a vector-Jacobian product (VJP). To refresh your memory, here is code which defines VJPs for exponentiation and multiplication.

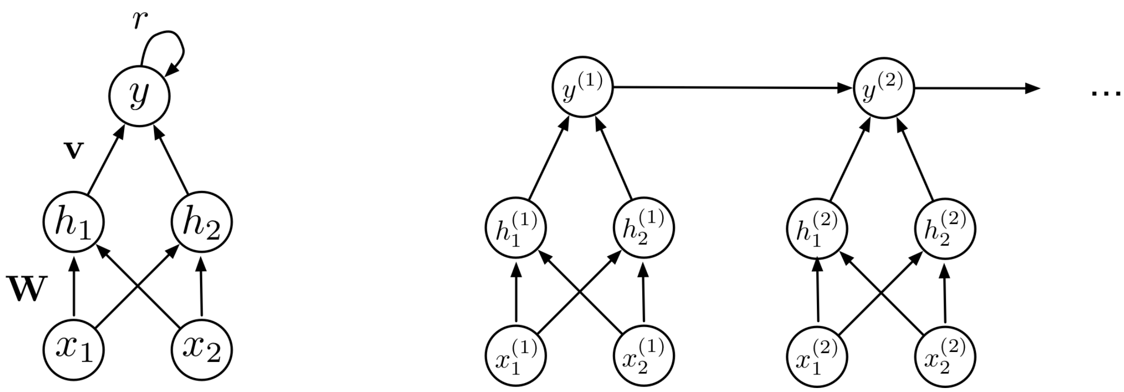
```
defvjp(exp,         lambda g, ans, x: ans * g)
defvjp(multiply,    lambda g, ans, x, y: y * g,
                    lambda g, ans, x, y: x * g)
```

The arguments to `defvjp` are the primitive op, followed by functions implementing the VJPs for each of the arguments. The arguments to the VJP function are: the output gradient `g`, the output `ans` of the op, and the arguments fed to the op.

(a) **[1pt]** Write Python code that defines a vector-Jacobian product for `sin`.

(b) **[1pt]** Write Python code that defines a vector-Jacobian product for `divide`, the function which computes the elementwise division of two arrays (i.e. `divide(x, y)` is equivalent to `x / y`). (This is floating point division, not integer division.)

11. **[4pts]** Suppose we receive two binary sequences $\mathbf{x_1} = (x_1^{(1)}, \ldots, x_1^{(T)})$ and $\mathbf{x_2} = (x_2^{(1)}, \ldots, x_2^{(T)})$ of equal length, and we would like to design an RNN to determine if they are identical. We will use the following (rather unusual) architecture, drawn with self-loops on the left and unrolled on the right:



The computation in each time step is as follows:

$$\mathbf{h}^{(t)} = \phi\left(\mathbf{W}\mathbf{x}^{(t)} + \mathbf{b}\right)$$
$$y^{(t)} = \begin{cases} \phi\left(\mathbf{v}^\top \mathbf{h}^{(t)} + r y^{(t-1)} + c\right) & \text{for } t > 1 \\ \phi\left(\mathbf{v}^\top \mathbf{h}^{(t)} + c_0\right) & \text{for } t = 1, \end{cases}$$

where ϕ denotes the hard threshold activation function

$$\phi(z) = \begin{cases} 1 & \text{if } z > 0 \\ 0 & \text{if } z \leq 0 \end{cases}$$

The parameters are a 2×2 weight matrix $\mathbf{W}$, a 2-dimensional bias vector $\mathbf{b}$, a 2-dimensional weight vector $\mathbf{v}$, a scalar recurrent weight r, a scalar bias c for all but the first time step, and a separate bias c_0 for the first time step.

We'll use the following strategy. We'll proceed one step at a time, and at time t, the binary-valued elements $x_1^{(t)}$ and $x_2^{(t)}$ will be fed as inputs. The output unit $y^{(t)}$ at time t will compute whether all pairs of elements have matched up to time t. The two hidden units $h_1^{(t)}$ and $h_2^{(t)}$ will help determine if both inputs match at a given time step. *Hint: have $h_1^{(t)}$ determine if both inputs are 0, and $h_2^{(t)}$ detemine if both inputs are 1.*

Continued on next page ⟶

(Question 11, cont'd) Give parameters which correctly implement this function:

$\mathbf{W} =$

$\mathbf{b} =$

$\mathbf{v} =$

$r =$

$c =$

$c_0 =$

12. **[2pts]** Suppose we have flipped a coin multiple times, and it came up heads N_H times and tails N_T times. We would like to model the coin as a Bernoulli random variable, and fit the model using maximum likelihood.

    (a) **[1pt]** Give the formula for the log-likelihood $\ell(\theta)$, where θ is the probability of heads.

    (b) **[1pt]** Solve for the maximum likelihood estimate of θ by setting $\mathrm{d}\ell/\mathrm{d}\theta = 0$.

13. **[2pts]** Recall the CycleGAN architecture for style transfer.

    (a) **[1pt]** What might go wrong if we eliminate the discriminator terms from the cost function?

    (b) **[1pt]** What might go wrong if we eliminate the reconstruction terms from the cost function?

14. **[2pts]** Recall that a GAN could, in principle, be trained using the following minimax formulation, where G is the generator function, D is the probability the discriminator assigns to the sample being data, and $\mathcal{J}_D$ and $\mathcal{J}_G$ are the cost functions for the discriminator and generator, respectively.

$$\begin{aligned}
\mathcal{J}_D &= \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}[-\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}}[-\log(1 - D(G(\mathbf{z})))] \\
\mathcal{J}_G &= -\mathcal{J}_D \\
&= \text{const} + \mathbb{E}_{\mathbf{z}}[\log(1 - D(G(\mathbf{z})))]
\end{aligned}$$

However, in practice, the generator is usually trained with a different loss function.

(a) **[1pt]** What cost function do we typically use for the generator?

(b) **[1pt]** What is the reason to use this cost function rather than the one given above?

15. **[2pts]** We've covered autoregressive generative models based on both convolutional networks and RNNs.

(a) **[1pt]** Give one advantage of using a convolutional network rather than an RNN.

(b) **[1pt]** Give one advantage of using an RNN rather than a convolutional network.

16. **[3pts]** Reversible architectures are based on a reversible block. Let's modify the definition of the reversible block:

$$\mathbf{y}_1 = \mathbf{r} \circ \mathbf{x}_1 + \mathcal{F}(\mathbf{x}_2)$$
$$\mathbf{y}_2 = \mathbf{s} \circ \mathbf{x}_2,$$

where $\circ$ denotes elementwise multiplication. This modified block is identical to the ordinary reversible block, except that the inputs $\mathbf{x}_1$ and $\mathbf{x}_2$ are multiplied elementwise by vectors $\mathbf{r}$ and $\mathbf{s}$, all of whose entries are positive.

You don't need to justify your answers for this question, but doing so may help you receive partial credit.

(a) **[1pt]** Give equations for inverting this block, i.e. computing $\mathbf{x}_1$ and $\mathbf{x}_2$ from $\mathbf{y}_1$ and $\mathbf{y}_2$. You may use / to denote elementwise division.

(b) **[1pt]** Give a formula for the Jacobian $\partial \mathbf{y} / \partial \mathbf{x}$, where $\mathbf{y}$ denotes the concatenation of $\mathbf{y}_1$ and $\mathbf{y}_2$.

(c) **[1pt]** Give a formula for the determinant of the Jacobian from part (b).

17. **[2pts]** Suppose we have an MDP with two time steps. It has an initial state distribution $p(\mathbf{s}_1)$, transition probabilities $p(\mathbf{s}_{t+1} \,|\, \mathbf{s}_t, \mathbf{a}_t)$, and deterministic reward function $r(\mathbf{s}, \mathbf{a})$. The agent is currently following a stochastic policy $\pi_{\boldsymbol{\theta}}(\mathbf{a} \,|\, \mathbf{s})$ parameterized by $\boldsymbol{\theta}$.

    (a) **[1pt]** Give the formula for the probability $p(\tau)$ of a rollout $\tau = (\mathbf{s}_1, \mathbf{a}_1, \mathbf{s}_2, \mathbf{a}_2)$.

    (b) **[1pt]** What is the function that REINFORCE is trying to maximize with respect to $\boldsymbol{\theta}$? (You can give your answer in terms of $p(\tau)$.)

18. **[1pt]** Recall that the discounted return is defined as:

$$G_t = \sum_{i=0}^{\infty} \gamma^i r_{t+i},$$

where γ is the discount factor and r_t is the reward at time t. Give the definition of the action-value function $Q^{\pi}(\mathbf{s}, \mathbf{a})$ for policy π, state $\mathbf{s}$, and action $\mathbf{a}$. You can either give an equation or explain it verbally.