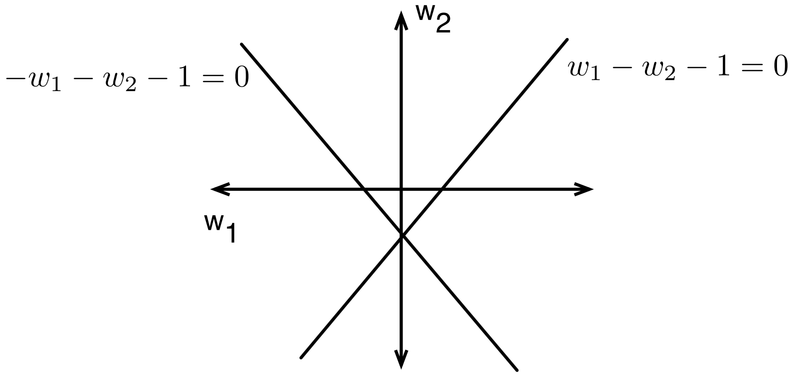
CSC321 Winter 2015 — Intro to Neural Networks
Solutions for night midterm

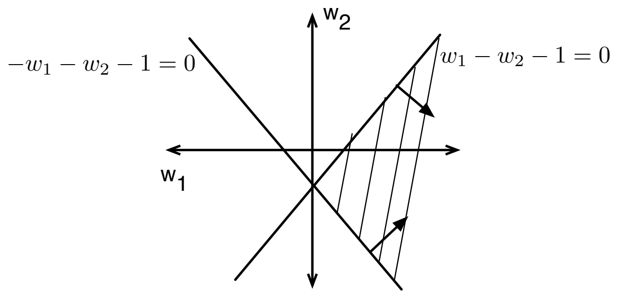
Unless otherwise specified, half the marks for each question are for the answer, and half are for an explanation which demonstrates understanding of the relevant concepts.

1. (1 mark) Suppose we want to train a perceptron with weights w_1 and w_2 and a fixed bias $b = -1$. Sketch the constraints in weight space corresponding to the following training cases. (The decision boundaries have already been drawn for you, so you only need to draw arrows to indicate the half-spaces.) Shade the feasible region or indicate that none exists. You do not need to justify your answer.

   $\mathbf{x} = (1, -1)$, $t = 1$
   $\mathbf{x} = (-1, -1)$, $t = 0$



   Solution



2. (1 mark) Suppose we have a fully connected, feed-forward network with no hidden layer, and 5 input units connected directly to 3 output units. Briefly explain why adding a hidden layer with 8 *linear* units does not make the network any more powerful (as opposed to not having a hidden layer).

   Solution. The network with 8 linear units computes a composition of two linear functions, which is linear. But the network with no hidden layer can compute any linear function, so it is at least as powerful as the network with a hidden layer. (In fact, the two are equally powerful, but the question doesn't ask for this.)

3. (1 mark) Suppose we have a network with linear hidden units, so that each hidden unit computes its activation h_i as

$$h_i = \sum_j w_{ij} x_j,$$

where the x_j are the input values and the w_{ij} are the weights. Let the matrix $\mathbf{X}$ represent the input values for a mini-batch of training examples (rows correspond to training examples and columns correspond to input dimensions). Let $\mathbf{W}$ represent the weight matrix, where the (i, j) entry connects input j to hidden unit i. Write a matrix expression which computes the hidden activations on this mini-batch, and specify what the rows and columns of the result correspond to. You do not need to justify your answer.

Solution. The activations are computed by $\mathbf{X}\mathbf{W}^\top$. Rows correspond to training cases, and columns correspond to output units. $\mathbf{W}\mathbf{X}^\top$ is also correct, if rows correspond to output units and columns correspond to training cases.

4. (1 mark) In stochastic gradient descent, each pass over the dataset requires the same number of arithmetic operations, whether we use minibatches of size 1 or size 1000. Why can it nevertheless be more computationally efficient to use minibatches of size 1000?

Solution. Possible answers include:

- For each set of 1000 examples, the size-1 approach requires a for loop (which incurs some overhead), while the size-1000 approach requires only a single matrix multiplication.
- The size-1000 approach makes use of matrix-matrix multiplication, whose implementations are heavily optimized (e.g. for cache efficiency).

We found that students wrote other advantages of using size-1000 that we not computational. For example, the gradients will oscillate more with size-1. This is certainly true, but this is not a *computational* advantage. Another advantage is that size-1000 will have fewer weight updates, but weight updates are element-wise operations and take negligible time compared to the matrix multiplies used to compute the updates.

5. (2 marks) In class, we saw that using squared error loss $C = (y-t)^2$ with a logistic output unit can make optimization difficult because the unit can saturate, leading to a small gradient. Cross-entropy loss doesn't have this problem. Suppose that we instead use the absolute loss $C = |y - t|$ (but keep the logistic output unit). Does this have the same problem with saturation that squared error does? Justify your answer algebraically and/or by drawing a figure.

Solution. Yes, it still has the problem. Suppose, for instance, that we have a training example with target $t = 0$. The cost is $C = |y - t| = y$. For large values of the input z (i.e. very bad predictions), the cost asymptotes to $C = 1$, so the derivatives $\partial C / \partial z$ are approximately 0. This implies there is a plateau.

6. (2 marks) Briefly explain a way in which the neural probabilistic language model from Assignment 1 was doing supervised learning, and a way in which it was doing unsupervised learning. **Solution.** The model was doing supervised learning because it was trained to predict the next word as a target given the previous few words. There are two ways in which it was doing unsupervised learning:

   (a) The goal was to model the distribution over sentences, which is an unsupervised task because the sentences aren't labeled in any way.

   (b) The model learned word representations which we could then analyze and visualize. This is unsupervised learning because we didn't specify the correct word representations in advance.

7. (1 mark) Briefly explain one thing you would use a validation set for, and why you can't just do it using the test set. **Solution.** Possible things we'd use a validation set for include:

   (a) Choosing hyperparameters, e.g. learning rate or number of hidden units.

   (b) Early stopping, where we stop the training once the validation set performance gets worse.

   We can't just use the test set, because then we'd be choosing aspects of the model or algorithm based on test set performance, and the test set performance will no longer be indicative of generalization performance.

8. (2 marks) Fill in weights, biases, and initial activations for the following RNN so that it initially outputs 1, but as soon as it receives an input of 0, it switches to outputting 0 for all subsequent time steps. For instance, the input `1110101` produces the output `1110000`. All units are binary threshold units with a threshold of 0. The hidden unit has an initial value of 0. You don't need to provide an explanation, but doing so may help you receive partial credit. *Hint: in one possible solution, the hidden unit has an activation $h_t = 0$ until there's an input $x_t = 0$, at which point it switches to maintaining an activation of 1 forever. The output unit always predicts the opposite of the hidden unit, i.e. $y = 1 - h$.*

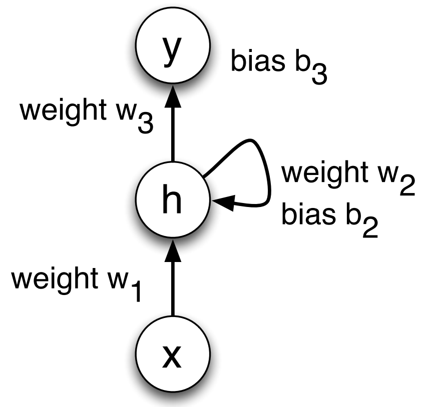
$w_1 =$ ________

$w_2 =$ ________

$b_2 =$ ________

$w_3 =$ ________

$b_3 =$ ________



Solution. A good strategy for thinking about this problem is to list the values each unit should take as a function of the units that feed into it. In particular,

h_t	y_t
0	1
1	0

h_{t-1}	x_t	h_t
0	0	1
0	1	0
1	0	1
1	1	1

One possible setting of the weights and biases which achieves these relationships is:

$w_1 = -1$

$w_2 = 1$

$b_2 = 0.5$

$w_3 = -1$

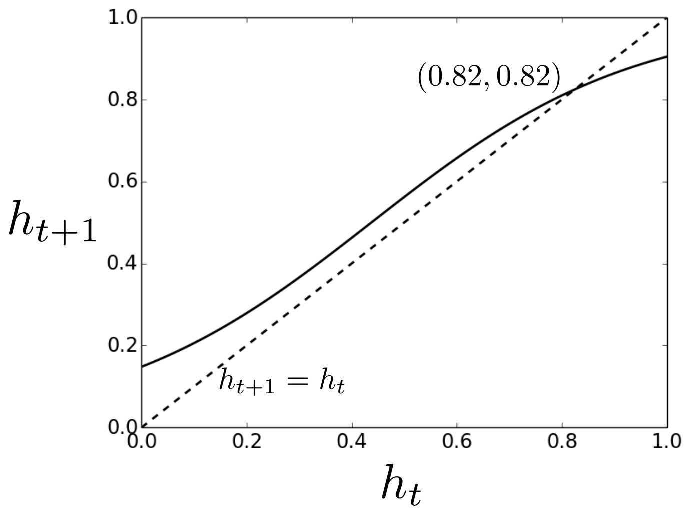
$b_3 = 0.5$

Scaling all these numbers by the same positive constant will also be a valid solution.

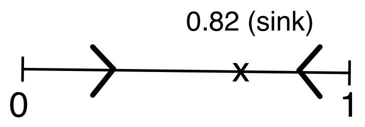
We found that several students chose numbers which led to a zero input to the threshold units. Unfortunately, we did not specify the behaviour of the unit at the threshold, that is, whether the output is 1 if the input is greater than the threshold or, greater than or equal to the threshold. So we gave full points if either of the two interpretations led to the correct result. However, the same interpretation must be used at both threshold units. In general, it is a good idea not to rely on boundary values when thinking about networks.

One mark for the relationship between h_t and y_t, and one mark for the relationship between x_t, h_{t-1}, and h_t.

9. (2 marks) Suppose we have an RNN with one hidden unit with a logistic nonlinearity, and no inputs or outputs. The unit has a bias of -1.75 and is connected to itself with a weight of 4. The figure shows the activation h_{t+1} as a function of the previous activation h_t. Draw a phase plot that summarizes the behavior of this system, and label any sources or sinks. For what values of the initial activation $r = h_0$ will we encounter exploding or vanishing gradients (with respect to r)?



Solution. When $h_t < 0.82$ we have $h_{t+1} > h_t$, and when $h_t > 0.82$ we have $h_{t+1} < h_t$. This can be summarized with the following phase plot:



Regardless of the initial value r, the activations approach 0.82. Therefore, $\partial h_T / \partial r$ will be close to 0 for large T, and we will encounter vanishing gradients for all values of r.

10. (2 marks) Alice and Bob have implemented two neural networks for recognizing handwritten digits from 16×16 grayscale images. Each network has a single hidden layer, and makes predictions using a softmax output layer with 10 units, one for each digit class.

   - Alice's network is a convolutional net. The hidden layer consists of three 16×16 convolutional feature maps, each with filters of size 5×5, and uses the logistic nonlinearity. All of the hidden units are connected to all of the output units.

- Bob's network is a fully connected network with no weight sharing. The hidden layer consists of 768 logistic units (the same number of units as in Alice's convolutional layer).

Briefly explain one advantage of Alice's approach and one advantage of Bob's approach.

Solution. The inputs to the convolution layer are a linear function of the images. In Bob's network, the hidden units can compute *any* linear function of the images; by contrast, Alice's convolutional layer is more restricted because of weight sharing and local connectivity. The advantage of Bob's network is that it is more powerful, i.e. it can compute any function Alice's network can compute, plus some additional functions. Advantages of Alice's network include:

(a) It has fewer parameters, so it is less likely to overfit.

(b) It has fewer connections, so it requires fewer arithmetic operations to compute the activations or the weight gradients.