1. **[2pts]** Suppose you design a multilayer perceptron for classification with the following architecture. It has a single hidden layer with the hard threshold activation function. The output layer uses the softmax activation function with cross-entropy loss. What will go wrong if you try to train this network using gradient descent? Justify your answer in terms of the backpropagation rules.

   Solution: The relevant backprop rules are:

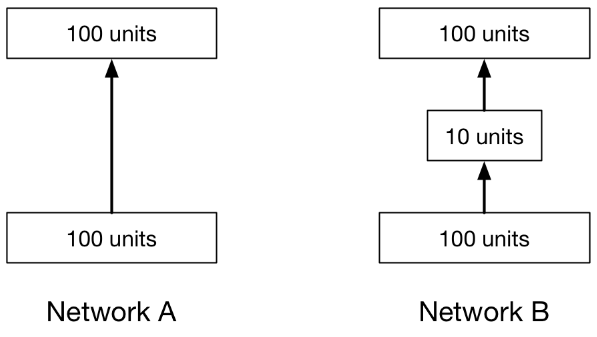
$$\begin{aligned}\overline{z_i} &= \overline{h_i}\,\phi'(z_i)\\ \overline{w_{ij}} &= \overline{z_i}\,x_j\end{aligned}$$

   From these equations, we see that if $\phi'(z_i) = 0$, then $w_{ij} = 0$. For the hard threshold activation function, the derivative is zero almost everywhere, so the gradient for the first-layer weights will be zero, and the weights will never be updated.

   Marking: We intended for you to write the backprop equation, but didn't state this clearly, so we didn't take off for it. One point for saying the gradient will be zero, and the other point for explaining why.

   Mean: 1.4/2

2. Consider the following two multilayer perceptrons, where all of the layers use linear activation functions.



   (a) **[1pt]** Give one advantage of Network A over Network B.

   Possible answers:

   - A is more expressive than B.
   - Half credit for saying it has fewer units or is easier to implement.

   (b) **[1pt]** Give one advantage of Network B over Network A.

   Possible answers:

- B has fewer connections, so it's less prone to overfitting
- B has fewer connections, so backprop requires fewer operations
- B has a bottleneck layer, so the network is forced to learn a compact representation (like an autoencoder)

Mean: 1.8/2

3. Suppose you train an ensemble of 5 networks for object recognition; all of the networks use the same architecture, but start from different random initializations. We saw that if you average the predictions of all five networks, you are guaranteed to do better in expectation (in terms of cross-entropy loss) than if you just use one of the networks.

   (a) **[1pt]** This guarantee depended on a particular property of cross-entropy loss. What is that property? (A one-word answer is sufficient, but an explanation may help you get partial credit.)

   Solution: Convexity.

   (b) **[1pt]** Does this guarantee hold if you instead average the weights and biases of the networks? Why or why not?

   Solution: No, the guarantee does not hold because the loss is not convex with respect to the weights and biases. Networks starting from different initializations might learn different hidden representations, so it makes no sense to average the weights and biases.

   Marking: Full credit for mentioning any of the following: (a) non-convexity, (b) different nets might learn different representations, (c) weight space symmatries, (d) separate basins of attraction. We also gave full credit for people who answered yes and then explained why linear models with convex loss functions are convex. (We forgot to specify that this was a multi-layer network.)

Mean: 1.2/2

4. Consider Bayesian optimization, where we are trying to minimize a function $f(\boldsymbol{\theta})$.

   (a) **[1pt]** Give one reason that Probability of Improvement is a better acquisition function than the negative predictive mean (i.e. $-\mathbb{E}[f(\boldsymbol{\theta})]$).

   Possible solutions:

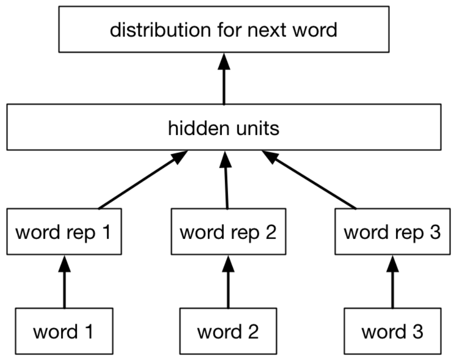
   - PI will not try the same point again, since it has zero chance of improvement
   - PI encourages exploration since points with poor mean have a higher PI the higher their variance

(b) **[1pt]** Give one reason that Expected Improvement is a better acquisition function that Probability of Improvement.

Solution: PI tends to look for incremental improvements, often nearby to points already explored. EI would rather take a smaller chance of a larger improvement, so it more strongly favors exploration.

Mean: 0.9/2

5. **[1pt]** Recall the neural language model architecture:



Why is it that the learned word representations for "professor" and "teach" are likely to be far apart? (It's not sufficient to say they are "semantically dissimilar" – your answer should mention how the representations are used in the prediction task.)

Solution: Two words will have similar representations when seeing either word in a particular context will lead you to make similar predictions. Since "professor" and "teach" have different parts of speech, the distributions of next words will be different, so the words will have dissimilar representations.

Mean: 0.6/1

6. Suppose you want to redesign the AlexNet architecture to reduce the number of arithmetic operations required for each backprop update.

   (a) **[1pt]** Would you try to cut down on the number of weights, units, or connections? Justify your answer.

   Solution: Cut down on the number of connections, since the number of arithmetic operations is proportional to the number of connections. (I.e., one operation for the forward pass, and two for the backward pass.)

(b) [**1pt**] Would you modify the convolution layers or the fully connected layers? Justify your answer.

Solution: Modify the convolution layers, since most of the connections are in these layers.

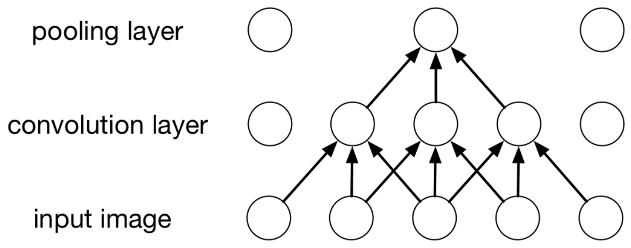
Mean: 1.0/2

7. [**2pts**] Suppose you have a convolutional network with the following architecture:

- The input is an RGB image of size 256×256.
- The first layer is a convolution layer with 32 feature maps and filters of size 3×3. It uses a stride of 1, so it has the same width and height as the original image.
- The next layer is a pooling layer with a stride of 2 (so it reduces the size of each dimension by a factor of 2) and pooling groups of size 3×3.

Determine the size of the receptive field for a single unit in the pooling layer. (I.e., determine the size of the region of the input image which influences the activation of that unit.) You may assume the receptive field lies entirely within the image. *Hint: you may want to draw a one-dimensional conv net to reason about this problem.*

Solution: The following figure shows a 1-D analogue:



In this figure, 5 of the input units feed into the pooling unit. This happens along both dimensions in the 2-D case, so the size of the receptive field is 5×5.

Marking: We couldn't come up with any reasonable way to assign partial credit for this question. Instead, we gave one free point and made this question worth one point. It was all-or-nothing, based on the answer. The intended answer was 5×5, but we also accepted $5 \times 5 \times 3$, 25, or 75.

Mean: 0.4/1

8. In this course, we've discussed five cases where you want to use backprop to compute the gradient of some function *with respect to the pixels of an image*. Describe two of these cases. For each one, explain what function we compute the gradient of and what the gradient is useful for. (A one-sentence verbal description is sufficient; you don't need to write equations.)

   (a) **[1pt]** The first example:

   (b) **[1pt]** The second example:

   Possible solutions:

   - We compute the gradient of the activations of a unit in order to visualize which parts of the image cause that unit to activate.
   - In inceptionism (i.e. DeepDream), we use backprop to update the image to increase the activations of the units which are already highly activated. (Technically we're not computing derivatives of a function here. Any reasonable explanation of DeepDream was accepted.)
   - In doing style transfer, we optimize the image to minimize a combination of squared difference between some layer's activations for the generated image and the photograph (i.e. the content penalty), and the squared difference between the second-order statistics of the generated image and the painting (i.e. the style penalty).
   - When generating adversarial images, we perturb the image in the direction that maximizes the probability of misclassification. (Other similar criteria are possible also.)
   - When training a GAN, the backprop computations for the generator involve computing the derivatives of the discriminator's output with respect to the generated image.
   - One way to do image completion is to optimize for an image which has high likelihood under some particular generative model.

   Marking: Even though we asked for the function we're computing the gradients of, we gave full credit for any answer that gave a good explanation (even if it didn't specify the function). We also gave full credit for explaining how image gradients can be used for edge detection — this was a reasonable misinterpretation of the question.

   Mean: 1.1/2

9. We considered two different models for binary images: the mixture of Bernoullis (MoB) and restricted Boltzmann machine (RBM).

   (a) **[1pt]** Give one advantage of an RBM over an MoB. (It's not sufficient to say that it gets higher likelihood or produces better samples — you should explain why it can model the data better.)

   Possible solutions:

   - It learns a distributed representation.
   - It's better at modeling soft constraints.

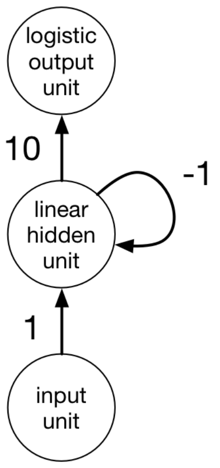
   (b) **[1pt]** Give one advantage of an MoB over an RBM.

   Possible solutions:

   - Posterior inference is computationally tractable.
   - Computing the likelihoods is computationally tractable.
   - It's easy to understand what the model is doing by visualizing the mean pixels for each class.
   - You can use the class labels when fitting the model.
   - A discrete set of classes may be more interpretable than a distributed representation.

   Mean: 1.4/2

10. **[2pts]** Determine what the following recurrent network computes. More precisely, determine the function computed by the output unit at the final time step; the other outputs are not important. All of the biases are 0. You may assume the inputs are integer valued and the length of the input sequence is even.



Solution: This network outputs 1 if the sum of the even-numbered inputs is larger than the sum of the odd-numbered inputs, and 0 if it is less.

Marking: Full credit for stating the above solution, even without justification. For incorrect or incomplete answers, here is our scheme for partial credit:

- 1 point for writing out the RNN computations to determine the recurrence
- 1/2 point for solving the recurrence
- 1/2 point for understanding the significance of the logistic nonlinearity for the output unit

Mean: 1.3/2

11. **[2pts]** Consider a residual network built out of residual units, where the residual function is given by an MLP with one hidden layer:

$$\begin{aligned}\mathbf{z} &= \mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)} \\ \mathbf{h} &= \phi(\mathbf{z}) \\ \mathbf{y} &= \mathbf{x} + \mathbf{W}^{(2)}\mathbf{h}\end{aligned}$$

Give a way of setting the weights and biases such that the derivatives will not explode or vanish. Briefly explain your answer, but you do not need to provide a detailed derivation.

Possible solutions:

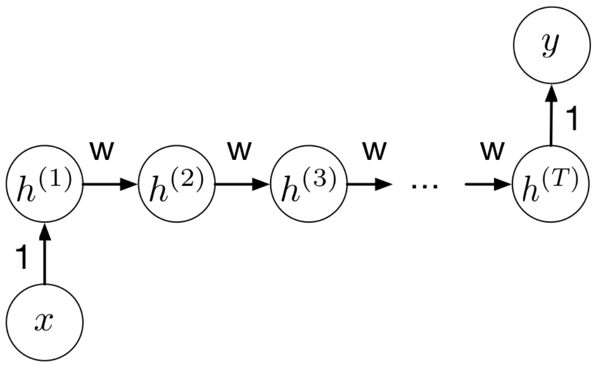
- Set $\mathbf{W}^{(2)}$ to zero so that it computes the identity function.
- Set either $\mathbf{W}^{(1)}$ or $\mathbf{W}^{(2)}$ to zero so that the Jacobian is the identity.
- Variants of the above where the weights are just *close* to 0.

Marking: One point for recognizing that we'd like to have identity Jacobian or compute the identity function, without saying how to do it.

Mean: 0.9/2

12. Consider the following RNN, which has a scalar input at the first time step, makes a scalar prediction at the last time step, and uses a shifted logistic activation function:

$$\phi(z) = \sigma(z) - 0.5.$$



(a) **[1pt]** Write the formula for the derivative $\overline{h_t}$ as a function of $\overline{h_{t+1}}$, for $t < T$. (Use z_t to denote the input to the activation function at time t. You may write your answer in terms of σ', i.e. you don't need to explicitly write out the derivative of σ.)

Solution:

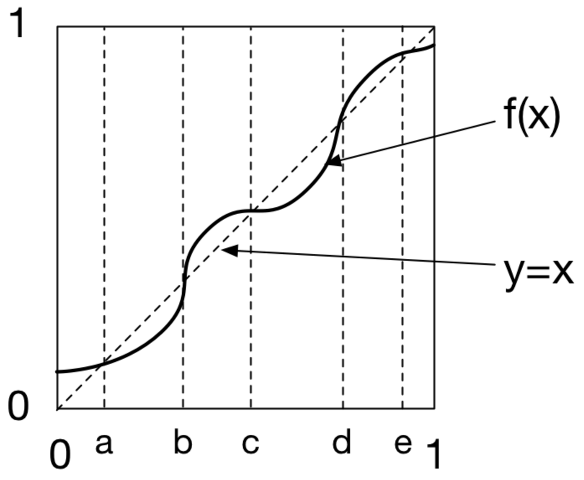
$$\overline{h_t} = \overline{h_{t+1}}\, \sigma'(z_t)\, w$$

(b) **[1pt]** Suppose the input to the network is $x = 0$. Notice that $h_t = 0$ for all t. Based on your answer to part (a), determine the value α such that if $w < \alpha$, the gradient vanishes, while if $w > \alpha$, the gradient explodes. You may use the fact that $\sigma'(0) = 1/4$.

Solution: Set $\alpha = 4$, since this is the value for which $\overline{h_t} = \overline{h_{t+1}}$.

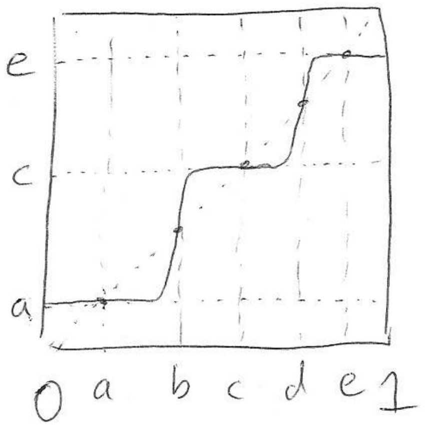
Mean: 1.6/2

13. **[2pts]** Consider the following function $f(x)$:



Here, a, b, c, d, e represent real values between 0 and 1. Sketch the function $f^{(100)}(x)$, i.e. f iterated 100 times. Label any relevant values on the axes.

Solution:



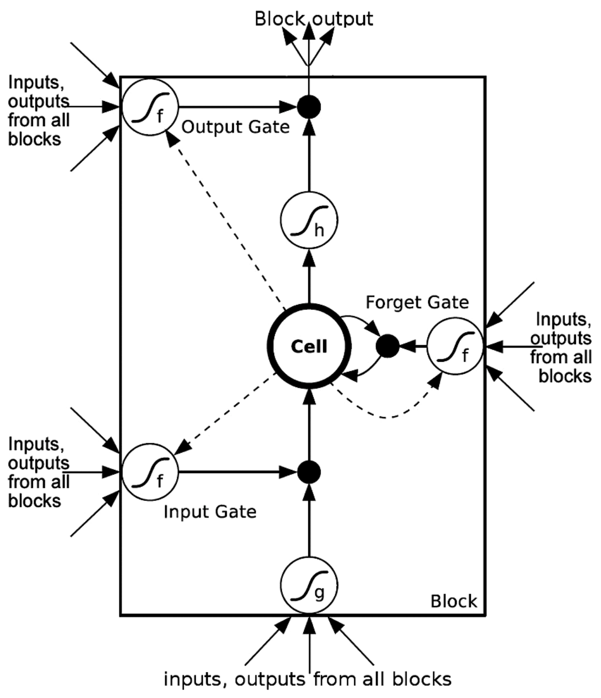
Marking: Half a point for each of the following:

- recognizing that the function should pass through (a, a), (b, b), etc.
- overall shape of the function
- correctly identifying sources and sinks

- fully correct plot

Mean: 1.3/2

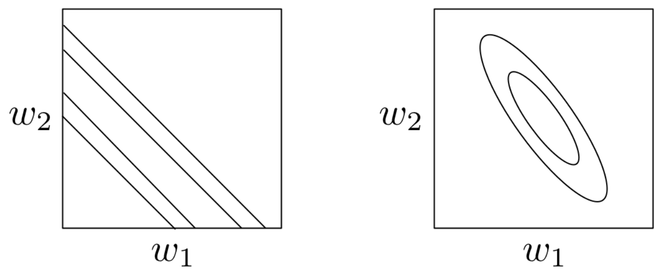
14. **[1pt]** Recall the LSTM architecture. Suppose you want the memory cell to sum its inputs over time. What values should the input gate and forget gate take? You do not need to justify your answer.


Block output
Inputs, outputs from all blocks
Output Gate
h
f
Forget Gate
Cell
Inputs, outputs from all blocks
Inputs, outputs from all blocks
Input Gate
g
Block
inputs, outputs from all blocks

Solution: input = 1, forget = 1

Mean: 0.8/1

15. **[1pt]** For linear regression with scalar-valued targets, which of the following contour plots in weight space best represents the cost function for a *single* training example? Justify your answer.



Plot A Plot B

Solution: A, because the predictions for a single example are a linear function of the weights. Changing the weights in a direction orthogonal to the input doesn't affect the predictions, and therefore doesn't affect the cost.

Mean: 0.5/1

16. Consider the problem of MAP estimation for the mean μ of a Gaussian distribution with known standard deviation σ. For the prior distribution, we will use a Gaussian distribution with mean 0 and standard deviation γ.

   (a) **[2pts]** Determine the function that we need to maximize. You do not need to determine the constant terms explicitly.
   Solution:

$$\begin{aligned}\log p(\mu \mid \mathcal{D}) &= \text{const} + \log p(\mu) + \log p(\mathcal{D} \mid \mu) \\ &= \text{const} + \log \mathcal{N}(\mu; 0, \gamma) + \sum_{i=1}^{N} \log \mathcal{N}(x^{(i)}; \mu, \sigma) \\ &= \text{const} - \frac{1}{2\gamma^2}\mu^2 - \frac{1}{2\sigma^2} \sum_{i=1}^{N} (x^{(i)} - \mu)^2\end{aligned}$$

   (b) **[1pt]** Determine the optimal value of μ by setting the derivative to 0. (You do not need to justify why it is a maximum rather than a minimum.)
   Solution:

$$\begin{aligned}\frac{\mathrm{d}J}{\mathrm{d}\mu} &= -\frac{\mu}{\gamma^2} + \frac{1}{\sigma^2}\sum_{i=1}^{N}(x^{(i)} - \mu) \\ &= -\left(\frac{1}{\gamma^2} + \frac{N}{\sigma^2}\right)\mu + \frac{1}{\sigma^2}\sum_{i=1}^{N} x^{(i)}\end{aligned}$$

Setting this to zero,

$$\mu = \frac{\frac{1}{\sigma^2}\sum_{i=1}^{N} x^{(i)}}{\frac{1}{\gamma^2} + \frac{N}{\sigma^2}}$$

Marking: One point for writing down the general MAP formula (in terms of log prior probability and log-likelihood). One point for writing it in terms of the Gaussian PDF and then simplifying. One point for solving for μ.

Mean: 1.7/3

17. **[1pt]** Recall that for Bayesian parameter estimation for a Bernoulli random variable, we use the beta distribution as the prior for the mean parameter θ:

$$p(\theta) \propto \theta^{a-1}(1-\theta)^{b-1}.$$

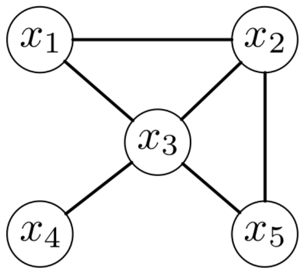
Give values of a and b for the beta prior which encode a strong prior belief that $\theta \approx 0.7$. You do not need to justify your answer.

Solution: Choose values such that a and b are large and $a/(a+b) \approx 0.7$.

Marking: Half a point if $a + b \geq 10$. Half a point if $0.6 \leq a/(a+b) \leq 0.8$.

Mean: 0.6/1

18. **[1pt]** In the following Boltzmann machine, list all of the variables which are conditionally independent of x_1 given x_3. You do not need to justify your answer.



Solution: $\{x_4\}$

Mean: 0.7/1

19. **[1pt]** In this course, we covered two network architectures whose targets are the same as their inputs. Pick one of these architectures. Name the network architecture and briefly explain why the task isn't trivial for the network to learn.

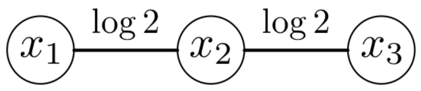
    Possible solutions:

    - For an autoencoder, the target is the same as the input. It's not trivial to learn because it needs to represent a high-dimensional input with a lower-dimensional code vector, forcing it to learn a compact representation.
    - For an RNN language model, the sampled output at one time step is fed back in as an input at the next time step. The task isn't trivial because the prediction is made for a particular word or character before it is fed in as input.

    Marking: Many people didn't understand what we meant by "explain why the task isn't trivial for the network to learn", and explained, e.g., why machine translation is hard. We gave full credit for mentioning either autoencoders or RNNs, regardless of the explanation.

    Mean: 0.9/1

20. **[2pts]** Consider the following Boltzmann machine, where all the variables take values in $\{0, 1\}$.



(This figure indicates that both edges have weight $\log 2$ and the biases are all 0.) Determine the conditional probability $\Pr(x_1 = 1 \mid x_3 = 1)$. *Hint: make a table which lists the happiness values for four configurations.*

Solution: The relevant configurations are the ones where $x_3 = 1$.

x_1	x_2	x_3	$w_{12}x_1x_2$	$w_{23}x_2x_3$	H	$\exp(H)$
0	0	1	0	0	0	1
0	1	1	0	$\log 2$	$\log 2$	2
1	0	1	0	0	0	1
1	1	1	$\log 2$	$\log 2$	$2 \log 2$	4

The conditional probability is given by:

$$\frac{1+4}{1+2+1+4} = \frac{5}{8}.$$

Marking: Full credit for the correct answer, regardless of justification. Partial credit: one point for the table, and one point for the logic relating the table to the conditional probability.

Mean: 1.4/2