1 point

1. One regularization technique is to start with lots of connections in a neural network, and then remove those that are least useful to the task at hand (removing connections is the same as setting their weight to zero). Which of the following regularization techniques is best at removing connections that are least useful to the task that the network is trying to accomplish?

- [ ] Early stopping
- [ ] L2 weight decay
- [ ] Weight noise
- [x] L1 weight decay

1 point

2. Why don't we usually train Restricted Boltzmann Machines by taking steps in the exact direction of the gradient of the objective function, like we do for other systems?

- [ ] Because it's unsupervised learning (i.e. there are no targets), there is no objective function that we would like to optimize.
- [ ] That would lead to severe overfitting, which is exactly what we're trying to avoid by using unsupervised learning.
- [x] That gradient is intractable to compute exactly.

1 point

3. When we want to train a Restricted Boltzmann Machine, we could try the following strategy. Each time we want to do a weight update based on some training cases, we turn each of those training cases into a full configuration by adding a sampled state of the hidden units (sampled from their distribution conditional on the state of the visible units as specified in the training case); and then we do our weight update in the direction that would most increase the goodness (i.e. decrease the energy) of those full configurations. This way, we expect to end up with a model where configurations that match the training data have high goodness (i.e. low energy).

However, that's not what we do in practice. Why not?

- [x] High goodness (i.e. low energy) doesn't guarantee high probability.
- [ ] The gradient of goodness for a configuration with respect to the model parameters is intractable to compute exactly.
- [ ] Correctly sampling the state of the hidden units, given the state of the visible units, is intractable.
- [ ] That would lead to severe overfitting, which is exactly what we're trying to avoid by using unsupervised learning.

1 point

4. CD-1 and CD-10 both have their strong sides and their weak sides. Which is the main advantage of CD-10 over CD-1?

- [ ] CD-10 is less sensitive to small changes of the model parameters.
- [ ] The gradient estimate from CD-10 has less variance than the gradient estimate of CD-1.
- [x] CD-10 gets its negative data (the configurations on which the negative part of the gradient estimate is based) from closer to the model distribution than CD-1 does.
- [ ] The gradient estimate from CD-10 takes less time to compute than the gradient estimate of CD-1.
- [ ] The gradient estimate from CD-10 has more variance than the gradient estimate of CD-1.

1 point

5. CD-1 and CD-10 both have their strong sides and their weak sides. Which are significant advantages of CD-1 over CD-10? Check all that apply.

- [ ] The gradient estimate from CD-1 has more variance than the gradient estimate of CD-10.
- [ ] CD-1 gets its negative data (the configurations on which the negative part of the gradient estimate is based) from closer to the model distribution than CD-10 does.
- [x] The gradient estimate from CD-1 has less variance than the gradient estimate of CD-10.
- [x] The gradient estimate from CD-1 takes less time to compute than the gradient estimate from CD-10.

1 point

6. With a lot of training data, is the perceptron learning procedure more likely or less likely to converge than with just a little training data?

Clarification: We're not assuming that the data is always linearly separable.

- [ ] More likely.
- [x] Less likely.

1 point

7. You just trained a neural network for a classification task, using some weight decay for regularization. After training it for 20 minutes, you find that on the validation data it performs much worse than on the training data: on the validation data, it classifies 90% of the data cases correctly, while on the training data it classifies 99% of the data cases correctly. Also, you made a plot of the performance on the training data and the performance on the validation data, and that plot shows that at the end of those 20 minutes, the performance on the training data is improving while the performance on the validation data is getting worse.

What would be a reasonble strategy to try next? Check all that apply.

- [x] Redo the training with fewer hidden units.
- [x] Redo the training with more weight decay.
- [ ] Redo the training with less weight decay.
- [ ] Redo the training with more hidden units.
- [ ] Redo the training with more training time.

1 point

8. If the hidden units of a network are independent of each other, then it's easy to get a sample from the correct distribution, which is a very important advantage. For which systems, and under which conditions, are the hidden units independent of each other? Check all that apply.

- [ ] For a Sigmoid Belief Network where the only connections are from hidden units to visible units (i.e. no hidden-to-hidden or visible-to-visible connections), when we condition on the state of the visible units, the hidden units are independent of each other.
- [x] For a Sigmoid Belief Network where the only connections are from hidden units to visible units (i.e. no hidden-to-hidden or visible-to-visible connections), when we don't condition on anything, the hidden units are independent of each other.
- [ ] For a Restricted Boltzmann Machine, when we don't condition on anything, the hidden units are independent of each other.
- [x] For a Restricted Boltzmann Machine, when we condition on the state of the visible units, the hidden units are independent of each other.

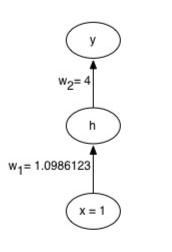
1 point

9. What is the purpose of momentum?

- [x] The primary purpose of momentum is to speed up the learning.
- [ ] The primary purpose of momentum is to reduce the amount of overfitting.
- [ ] The primary purpose of momentum is to prevent oscillating gradient estimates from causing vanishing or exploding gradients.

1 point

10. Consider a Restricted Boltzmann Machine with 2 visible units v_1, v_2 and 1 hidden unit h. The visible units are connected to the hidden unit by weights w_1, w_2 and the hidden unit has a bias b. An illustration of this model is given below.



The energy of this model is given by: $E(v_1, v_2, h) = -w_1 v_1 h - w_2 v_2 h - bh$. Recall that the joint probability $P(v_1, v_2, h)$ is proportional to $\exp(-E(v_1, v_2, h))$.

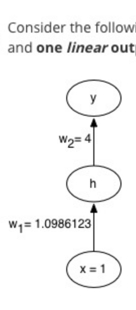
Suppose that $w_1 = 1, w_2 = -1.5, b = -0.5$. What is the conditional probability $P(h = 1 | v_1 = 0, v_2 = 1)$? Write down your answer with at least 3 digits

after the decimal point.

0.119

1 point

11. Consider the following feed-forward neural network with **one *logistic* hidden neuron** and **one *linear* output neuron**.



The input is given by $x = 1$, the target is given by $t = 5$, the input-to-hidden weight is given by $w_1 = 1.0986123$, and the hidden-to-output

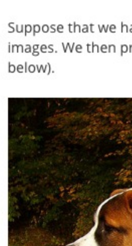
weight is given by $w_2 = 4$ (there are no bias parameters). What is the cost incurred by the network when we are using the **squared error cost**?

Remember that the squared error cost is defined by $\text{Error} = \frac{1}{2}(y - t)^2$. Write down your answer with at least 3 digits after the decimal point.

2.000

1 point

12. Consider the following feed-forward neural network with **one *logistic* hidden neuron** and **one *linear* output neuron**.



The input is given by $x = 1$, the target is given by $t = 5$, the input-to-hidden weight is given by $w_1 = 1.0986123$, and the hidden-to-output

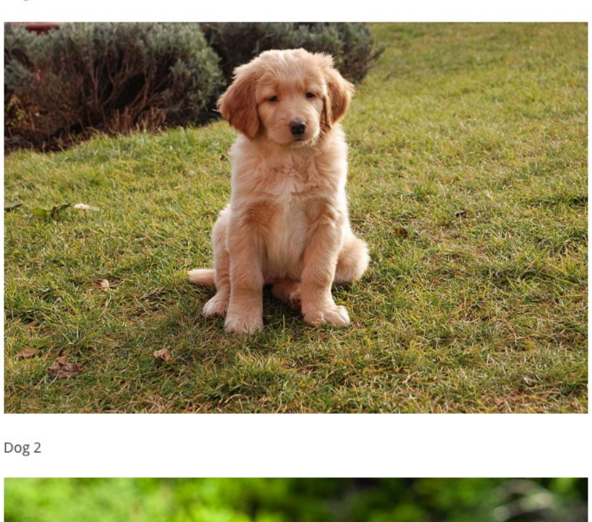
weight is given by $w_2 = 4$ (there are no bias parameters). If we are using the **squared error cost** then what is $\frac{\partial \text{Error}}{\partial w_1}$, the derivative of the error with respect to w_1? Remember that the squared error cost is defined by

$\text{Error} = \frac{1}{2}(y - t)^2$. Write down your answer with at least 3 digits after the decimal point.

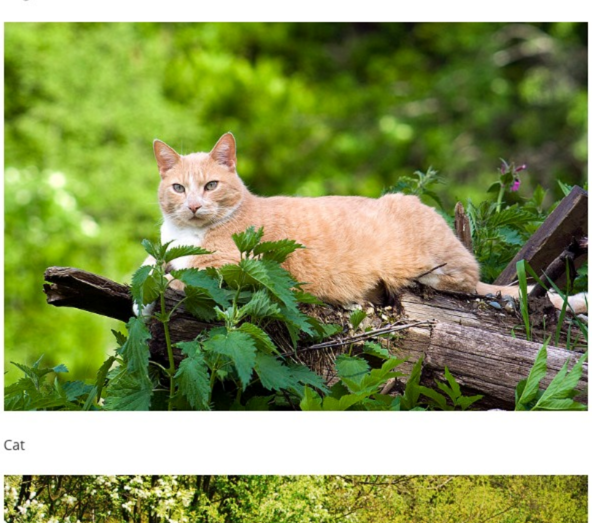
-1.500

1 point

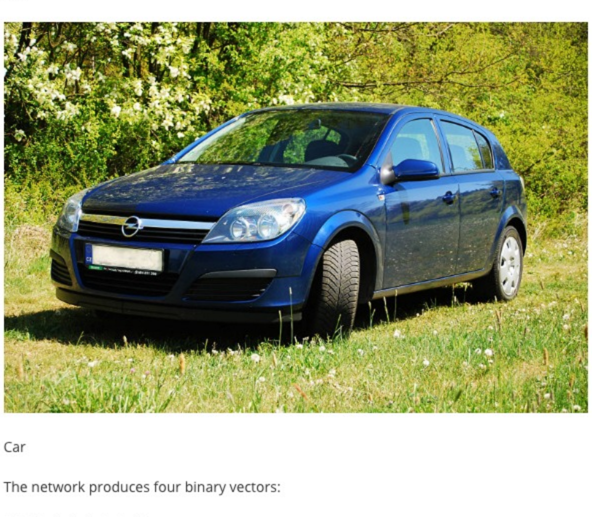
13. Suppose that we have trained a **semantic hashing** network on a large collection of images. We then present to the network four images: two dogs, a cat, and a car (shown below).



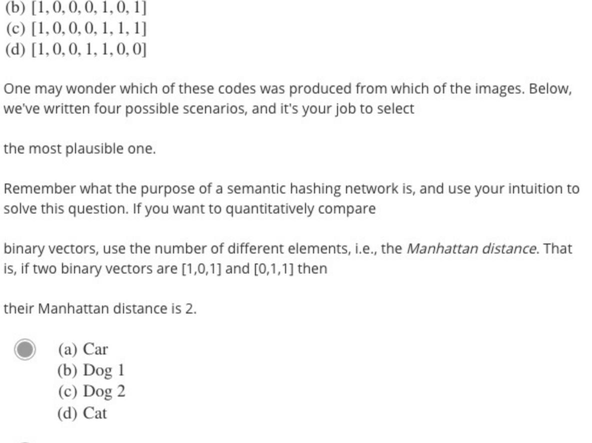
Dog 1



Dog 2



Cat



Car

The network produces four binary vectors:

(a) $[0, 1, 1, 1, 0, 0, 1]$
(b) $[1, 0, 0, 0, 1, 0, 1]$
(c) $[1, 0, 0, 0, 1, 1, 1]$
(d) $[1, 0, 0, 1, 1, 0, 0]$

One may wonder which of these codes was produced from which of the images. Below, we've written four possible scenarios, and it's your job to select

the most plausible one.

Remember what the purpose of a semantic hashing network is, and use your intuition to solve this question. If you want to quantitatively compare

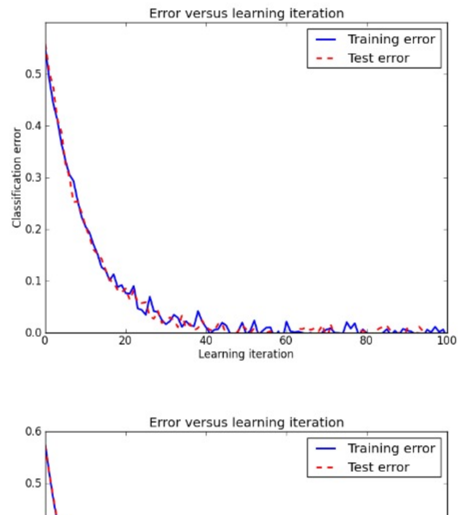
binary vectors, use the number of different elements, i.e., the *Manhattan distance*. That is, if two binary vectors are [1,0,1] and [0,1,1] then

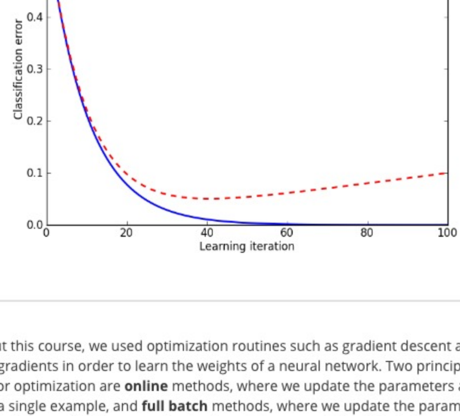
their Manhattan distance is 2.

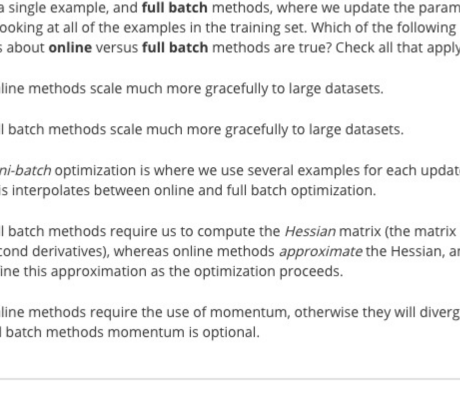
- [x] (a) Car
  (b) Dog 1
  (c) Dog 2
  (d) Cat
- [ ] (a) Dog 1
  (b) Cat
  (c) Car
  (d) Dog 2
- [ ] (a) Dog 2
  (b) Dog 1
  (c) Car
  (d) Cat
- [ ] (a) Cat
  (b) Car
  (c) Dog 2
  (d) Dog 1

1 point

14. The following plots show training and testing error as a neural network is being trained (that is, error per epoch). Which of the following plots is an obvious example of **overfitting** occurring as the learning progresses?

- [ ] 

- [ ] 

- [x] 

1 point

15. Throughout this course, we used optimization routines such as gradient descent and conjugate gradients in order to learn the weights of a neural network. Two principal methods for optimization are **online** methods, where we update the parameters after looking at a single example, and **full batch** methods, where we update the parameters only after looking at all of the examples in the training set. Which of the following statements about **online** versus **full batch** methods are true? Check all that apply.

- [x] Online methods scale much more gracefully to large datasets.
- [ ] Full batch methods scale much more gracefully to large datasets.
- [x] *Mini-batch* optimization is where we use several examples for each update. This interpolates between online and full batch optimization.
- [ ] Full batch methods require us to compute the *Hessian* matrix (the matrix of second derivatives), whereas online methods *approximate* the Hessian, and refine this approximation as the optimization proceeds.
- [ ] Online methods require the use of momentum, otherwise they will diverge. In full batch methods momentum is optional.

1 point

16. You have seen the concept of **weight sharing**, or **weight tying** appear throughout this course. For example, in dropout we combine an exponential number of neural networks with shared weights. In a convolutional neural network, each filter map involves using the same set of weights over different regions of the image. In the context of these models, which of the following statements about weight sharing is true?

- [ ] Since ordinary convolutional neural networks and non-convolutional networks with dropout both use weight sharing, we can infer that convolutional networks will generalize well to new data because they will randomly drop out hidden units with probability 0.5.
- [ ] Weight sharing implicitly causes models to prefer smaller weights. This means that it is equivalent to using weight decay and is therefore a form of regularization.
- [ ] Weight sharing introduces noise into the gradients of a network. This makes it equivalent to using a Bayesian model, which will help prevent overfitting.
- [x] Weight sharing reduces the number of parameters that need to be learned and can therefore be seen as a form of regularization.

1 point

17. In which case is unsupervised pre-training most useful?

- [ ] The data is real-valued.
- [ ] There is a lot of labeled data but very little unlabeled data.
- [ ] The data is linearly separable.
- [x] There is a lot of unlabeled data but very little labeled data.

1 point

18. Consider a neural network which operates on images and has one hidden layer and a single output unit. There are a number of possible ways to connect the inputs to the hidden units. In which case does the network have the smallest number of parameters? Assume that all other things (like the number of hidden units) are same.

- [ ] The hidden layer is locally connected, i.e. it's connected to local regions of the input image.
- [ ] The hidden layer is fully connected to the inputs.
- [ ] The hidden layer is fully connected to the inputs and there are skip connections from inputs to output unit.
- [x] The hidden layer is a convolutional layer, i.e. it has local connections and weight sharing.