EXERCISES 22

DEEP LEARNING

22.1 Simple Feedforward Networks

Exercise 22.NXOR

Construct by hand a neural network that computes the XOR function of two inputs. Make sure to specify what sort of units you are using.

XOR (in fact any Boolean function) is easiest to construct using step-function units. Because XOR is not linearly separable, we will need a hidden layer. It turns out that just one hidden node suffices. To design the network, we can think of the XOR function as OR with the AND case (both inputs on) ruled out. Thus the hidden layer computes AND, while the output layer computes OR but weights the output of the hidden node negatively. The network shown in Figure S22.1 does the trick.

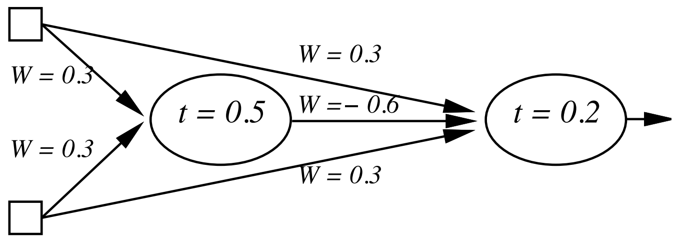


Figure S22.1 A network of step-function neurons that computes the XOR function.

Exercise 22.LNNE

Suppose you had a neural network with linear activation functions. That is, for each unit the output is some constant c times the weighted sum of the inputs.

a. Assume that the network has one hidden layer. For a given assignment to the weights $\mathbf{w}$, write down equations for the value of the units in the output layer as a function of $\mathbf{w}$ and the input layer $\mathbf{x}$, without any explicit mention of the output of the hidden layer. Show that there is a network with no hidden units that computes the same function.

b. Repeat the calculation in part (a), but this time do it for a network with any number of hidden layers.

c. Suppose a network with one hidden layer and linear activation functions has n input and output nodes and h hidden nodes. What effect does the transformation in part (a) to a network with no hidden layers have on the total number of weights? Discuss in particular the case $h \ll n$.

This exercise reinforces the student's understanding of neural networks as mathematical functions that can be analyzed at a level of abstraction above their implementation as a network of computing elements. For simplicity, we will assume that the activation function is the same linear function at each node: $g(x) = cx + d$. (The argument is the same (only messier) if we allow different c_i and d_i for each node.)

a. The outputs of the hidden layer are

$$H_j = g\left(\sum_k w_{k,j} I_k\right) = c \sum_k w_{k,j} I_k + d$$

The final outputs are

$$O_i = g\left(\sum_j w_{j,i} H_j\right) = c\left(\sum_j w_{j,i}\left(c \sum_k w_{k,j} I_k + d\right)\right) + d$$

Now we just have to see that this is linear in the inputs:

$$O_i = c^2 \sum_k I_k \sum_j w_{k,j} w_{j,i} + d\left(1 + c\sum_j w_{j,i}\right)$$

Thus we can compute the same function as the two-layer network using just a one-layer perceptron that has weights $w_{k,i} = \sum_j w_{k,j} w_{j,i}$ and an activation function $g(x) = c^2 x + d\left(1 + c\sum_j w_{j,i}\right)$.

b. The above reduction can be used straightforwardly to reduce an n-layer network to an $(n-1)$-layer network. By induction, the n-layer network can be reduced to a single-layer network. Thus, linear activation functions restrict neural networks to represent only linear functions.

c. The original network with n input and outout nodes and h hidden nodes has $2hn$ weights, whereas the "reduced" network has n^2 weights. When $h \ll n$, the original network has far fewer weights and thus represents the i/o mapping more concisely. Such networks are known to learn much faster than the reduced network; so the idea of using linear activation functions is not without merit.

[[need exercises]]

Exercise 22.188-LOFU

You would like to train a neural network to classify digits. Your network takes as input an image and outputs probabilities for each of the 10 classes, 0-9. The network's prediction is the class that it assigns the highest probability to. From the following functions, select all that would be suitable loss functions to minimize using gradient descent:

(A) The square of the difference between the correct digit and the digit predicted by your network
(B) The probability of the correct digit under your network
(C) The negative log-probability of the correct digit under your network

C only.

(A) is incorrect because it is non-differentiable. The correct digit and your model's predicted digit are both integers, and the square of their difference takes on values from the set $\{0^2, 1^2, \ldots, 9^2\}$. Losses that can be used with gradient descent must take on values from a continuous range and have well-defined gradients.
(B) is not a loss because you would like to *maximize* the probability of the correct digit under your model, not minimize it.
(C) is a common loss used for classification tasks. When the probabilities produced by a neural network come from a softmax layer, this loss is often combined with the softmax computation into a single entity known as the "softmax loss" or "softmax cross-entropy loss".

22.2 Computation Graphs for Deep Learning

Exercise 22.SOFG

A softmax layer in a neural network takes an input vector **x** and produces an output vector **y**, where

$$y_j = \frac{e^{x_j}}{\sum_{k=1^d} e^{x_k}} .$$

a. Obtain the derivatives $\partial y_j/\partial x_i$ in terms of x-values for the cases $i = j$ and $i \neq j$.
b. Re-express the derivatives in terms of y-values.
c. What can you say about the signs of the derivatives?
d. Using the indicator function $\mathbf{1}(i = j)$, which has value 1 if $i = j$ and 0 otherwise, combine your two expressions from part (b) into a single expression for the derivative $\partial y_j/\partial x_i$.

[[TBC]]

Exercise 22.SMSG

A softmax layer in a neural network takes an input vector **x** and produces an output vector **y**, where

$$y_j = \frac{e^{x_j}}{\sum_{k=1^d} e^{x_k}} .$$

Show that the sigmoid function is equivalent to a softmax with $d=2$.

[[TBC]]

Exercise 22.NNDS

This exercise asks you to implement the beginnings of a simple deep learning package.

a. Implement a data structure for general computation graphs, as described in Section 22.1, and define the node types required to support feed-forward neural networks with a variety of activation functions.
b. Write a function that computes the outputs of the computation graph given the inputs.
c. Now write a function that computes the error for a labelled example.
d. Finally, implement the local back-propagation algorithm based on Equations (22.11) and (22.12) and use it to create a stochastic gradient descent learning algorithm for a set of examples.

[[TBC]] The implementation of neural networks can be approached in several different ways. The simplest is probably to store the weights in an $n \times n$ array. Everything can be calculated as if all the nodes were in each layer, with the zero weights ensuring that only appropriate changes are made as each layer is processed.

Particularly for sparse networks, it can be more efficient to use a pointer-based implementation, with each node represented by a data structure that contains pointers to its successors (for evaluation) and its predecessors (for backpropagation). Weights $w_{j,i}$ are attached to node i. In both types of implementation, it is convenient to store the summed input $in_i = \sum_j w_{j,i}a_j$ and the value $g'(in_i)$. The code repository contains an implementation of the pointer-based variety. See the file `learning/algorithms/nn.lisp`, and the function `nn-learning` in that file.

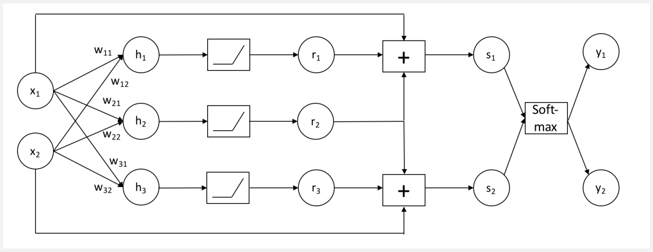
[[need exercises]]

Exercise 22.188-BAPR

Origin:

`su19_final_mesut_Backprop_SU19`

The network below is a neural network with inputs x_1 and x_2, and outputs y_1 and y_2. The internal nodes are computed below. All variables are scalar values. Note that $ReLU(x) = \max(0, x)$.



Neural Network

The expressions for the internal nodes in the network are given here for convenience:

$h_1 = w_{11}x_1 + w_{12}x_2 \qquad h_2 = w_{21}x_1 + w_{22}x_2 \qquad h_3 = w_{31}x_1 + w_{32}x_2$

$r_1 = ReLU(h_1) \qquad r_2 = ReLU(h_2) \qquad r_3 = ReLU(h_3) \qquad s_1 = x_1 + r_1 + r_2$

$s_2 = x_2 + r_2 + r_3$

$y_1 = \frac{exp(s_1)}{exp(s_1)+exp(s_2)} \qquad y_2 = \frac{exp(s_2)}{exp(s_1)+exp(s_2)}$

a. Forward Propagation

Suppose for this part only, $x_1 = 3, x_2 = 5, w_{11} = -10, w_{12} = 7, w_{21} = 2, w_{22} = 5, w_{31} = 4, w_{32} = -4$. What are the values of the internal nodes? Please simplify any fractions.

(i) h_1 (ii) s_1 (iii) y_2

b. Back Propagation

Compute the following gradients analytically. The answer should be an expression of any of the nodes in the network $(x_1, x_2, h_1, h_2, h_3, r_1, r_2, r_3, s_1, s_2, y_1, y_2)$ or weights $w_{11}, w_{12}, w_{21}, w_{22}, w_{31}, w_{32}$. In the case where the gradient depend on the value of nodes in the network, please list all possible analytical expressions, caused by active/inactive ReLU, separated by comma.

Hint 1: If z is a function of y, and y is a function of x, the chain rule of taking derivative is: $\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y} * \frac{\partial y}{\partial x}$

Hint 2: Hint: Recall that for functions of the form $g(x) = \frac{1}{1+exp(a-x)}$, $\frac{\partial g}{\partial x} = g(x)(1 - g(x))$

(i) $\frac{\partial h_2}{\partial x_1}$ (iv) $\frac{\partial s_1}{\partial r_1}$ (vii) $\frac{\partial y_1}{\partial x_1}$

(ii) $\frac{\partial h_1}{\partial w_{21}}$ (v) $\frac{\partial s_1}{\partial x_1}$

(iii) $\frac{\partial r_3}{\partial w_{31}}$ (vi) $\frac{\partial y_2}{\partial s_2}$

c. In roughly 15 words, what role could the non-negative values in node r_2 play according to its location in the network architecture?

a. (i) 5

$(-10) * 3 + 7 * 5 = 5$

(ii) 39

$3 + ReLU((-10) * 3 + 7 * 5) + ReLU(2 * 3 + 5 * 5) = 3 + 5 + 31 = 39$

(iii) $\frac{1}{1+exp(3)}$

$s_2 = 5 + ReLU(2 * 3 + 5 * 5) = 5 + 31 = 36$

$y_2 = \frac{exp(36)}{exp(36)+exp(39)} = \frac{1}{1+exp(3)}$

b. (i) w_{21}

(ii) 0

h_1 is independent of weight w_{21}

(iii) $0, x_1$ (two possibilities)

If the ReLU is active, the result is straightforward, and thus x_1. If the ReLU is inactive, the gradient is 0

(iv) 1

(v) $1 + w_{11} + w_{21}, 1 + w_{21}, 1 + w_{11}, 1$

The ReLU with output r_1 and the ReLU with output r_2 can be active or inactive independently

(vi) y_1y_2, or $y_2(1 - y_2)$

$y_2 = \frac{exp(s_2)}{exp(s_1)+exp(s_2)} = \frac{1}{1+exp(s_1-s_2)}$

$\frac{\partial y_2}{\partial s_2} = y_2(1 - y_2) = y_1y_2 = \frac{exp(s_1-s_2)}{(1+exp(s_1-s_2))^2} = \frac{exp(s_1+s_2)}{(exp(s_1)+exp(s_2))^2}$

Please note that, due to symmetry between y_1 and y_2, $\frac{exp(s_2-s_1)}{(1+exp(s_2-s_1))^2}$ is also equivalent to the correct answer. If you don't believe it, try simplifying it!

(vii) $y_1y_2(1 + w_{11} - w_{31}), y_1y_2(1 + w_{11}), y_1y_2(1 - w_{31}), y_1y_2$ (four possibilities)

Explanation:

$y_1 = \frac{exp(s_1)}{exp(s_1)+exp(s_2)} = \frac{1}{1+exp(s_2-s_1)}$

When calculating $\frac{\partial y_1}{\partial s_1}$, this is the correct format for $g(x) = \frac{1}{1+exp(a-x)}$, because there is a negative sign is in front of s_1. e can apply the rule to simply get $\frac{\partial y_1}{\partial s_1} = y_1(1 - y_1) = y_1y_2$

But when calculating $\frac{\partial y_1}{\partial s_2}$, this is NOT in the correct format, because now there is no negative sign in front of s_2. One way to resolve this is to create a fake variable $t_2 = -s_2$, and $\frac{\partial t_2}{\partial s_2} = -1$.

Now $y_1 = \frac{1}{1+exp(s_2-s_1)} = \frac{1}{1+exp(-s_1+s_2)} = \frac{1}{1+exp(-s_1-t_2)}$, which is the correct format again. At this point, $\frac{\partial y_1}{\partial s_2} = \frac{\partial y_1}{\partial t_2}\frac{\partial t_2}{\partial s_2} = [y_1(1 - y_1)](-1) = -y_1y_2$

As seen in v, $\frac{\partial s_1}{\partial x_1} = 1 + w_{11} + w_{21}, 1 + w_{21}, 1 + w_{11}, 1$

Similarly, $\frac{\partial s_2}{\partial x_1} = w_{21} + w_{31}, w_{21}, w_{31}$

In the end,

$$\begin{aligned}\frac{\partial y_1}{\partial x_1} &= \frac{\partial y_1}{\partial s_1}\frac{\partial s_1}{\partial x_1} + \frac{\partial y_1}{\partial s_2}\frac{\partial s_2}{\partial x_1} \\ &= y_1y_2 * \{1 + w_{11} + w_{21}, 1 + w_{21}, 1 + w_{11}, 1\} \\ &+ (-y_1y_2) * \{w_{21} + w_{31}, w_{21}, w_{31}\}\end{aligned}$$

Which can be expanded to this list (note that things can cancel out): $y_1y_2(1+w_{11}-w_{31}), y_1y_2(1+w_{11}), y_1y_2(1+w_{11}+w_{21}-w_{31}), y_1y_2(1-w_{31}), y_1y_2, y_1y_2(1+w_{21}-w_{31}), y_1y_2(1+w_{11}-w_{21}-w_{31}, y_1y_2(1+w_{11}-w_{21}), y_1y_2(1+w_{11}-w_{31}), y_1y_2(1+-w_{21}-w_{31}, y_1y_2(1-w_{21}), y_1y_2(1-w_{31})$

But, note that the activity of r_2 actually affect both $\frac{\partial s_1}{\partial x_1}$ and $\frac{\partial s_2}{\partial x_1}$! So we need to remove 6 contradicting terms, including $y_1y_2(1+w_{11}+w_{21}-w_{31})$, $y_1y_2(1+w_{21}-w_{31})$, $y_1y_2(1+w_{11}-w_{21}-w_{31})$, $y_1y_2(1+w_{11}-w_{21})$, $y_1y_2(1-w21-w_{31})$, $y_1y_2(1-w_{21})$.

The easier way to think about this is, that if r_2 is activated, then the gradient will cancel out from the two sides. If r_2 is deactivated, it won't appear at all.

We are left with 6 terms: $y_1y_2(1+w_{11}-w_{31}), y_1y_2(1+w_{11}), y_1y_2(1-w_{31}), y_1y_2, y_1y_2(1+w_{11}-w_{31}), y_1y_2(1-w_{31})$

Removing the duplicates, we are left with the final 4 terms: $y_1y_2(1+w_{11}-w_{31}), y_1y_2(1+w_{11}), y_1y_2(1-w_{31}), y_1y_2$

c. Smoothing the inputs into the soft-max functions. (anything reasonable is acceptable)

Exercise 22.188-MTXB

Origin:

`fa17_final_MLQuestionEasier`

In this question we will perform the backward pass algorithm on the formula

$$f = \frac{1}{2}\left\|\mathbf{A}\mathbf{x}\right\|^2$$

Here, $\mathbf{A} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$, $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$, $\mathbf{b} = \mathbf{A}\mathbf{x} = \begin{bmatrix} A_{11}x_1 + A_{12}x_2 \\ A_{21}x_1 + A_{22}x_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$, and $f = \frac{1}{2}\left\|\mathbf{b}\right\|^2 = \frac{1}{2}\left(b_1^2 + b_2^2\right)$ is a scalar.

a. Calculate the following partial derivatives of f.

(i) Find $\frac{\partial f}{\partial \mathbf{b}} = \begin{bmatrix} \frac{\partial f}{\partial b_1} \\ \frac{\partial f}{\partial b_2} \end{bmatrix}$.

b. Calculate the following partial derivatives of b_1.

(i) $\left(\frac{\partial b_1}{\partial A_{11}}, \frac{\partial b_1}{\partial A_{12}} \right)$

(ii) $\left(\frac{\partial b_1}{\partial A_{21}}, \frac{\partial b_1}{\partial A_{22}} \right)$

(iii) $\left(\frac{\partial b_1}{\partial x_1}, \frac{\partial b_1}{\partial x_2} \right)$

c. Calculate the following partial derivatives of f.

(i) $\left(\frac{\partial f}{\partial A_{11}}, \frac{\partial f}{\partial A_{12}} \right)$

(ii) $\left(\frac{\partial f}{\partial A_{21}}, \frac{\partial f}{\partial A_{22}} \right)$

(iii) $\left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2} \right)$

d. Now we consider the general case where $\mathbf{A}$ is an $n \times d$ matrix, and $\mathbf{x}$ is a $d \times 1$ vector. As before, $f = \frac{1}{2} \|\mathbf{A}\mathbf{x}\|^2$.

(i) Find $\frac{\partial f}{\partial \mathbf{A}}$ in terms of $\mathbf{A}$ and $\mathbf{x}$ only.

(ii) Find $\frac{\partial f}{\partial \mathbf{x}}$ in terms of $\mathbf{A}$ and $\mathbf{x}$ only.

a. (i) $\begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$

b. (i) (x_1, x_2)

(ii) $(0, 0)$

(iii) (A_{11}, A_{12})

c. Calculate the following partial derivatives of f.

(i) $(x_1 b_1, x_2 b_1)$

(ii) $(x_1 b_2, x_2 b_2)$

(iii) $(A_{11} b_1 + A_{21} b_2, A_{12} b_1 + A_{22} b_2)$

d. (i) $\mathbf{A}\mathbf{x}\mathbf{x}^\top$

(ii) $\mathbf{A}^\top \mathbf{A}\mathbf{x}$

22.3 Convolutional Networks

[[need exercises]]

22.4 Learning Algorithms

Exercise 22.TSOH

Suppose that a training set contains only a single example, repeated 100 times. In 80 of the 100 cases, the single output value is 1; in the other 20, it is 0. What will a back-propagation network predict for this example, assuming that it has been trained and reaches a global optimum? (*Hint:* to find the global optimum, differentiate the error function and set it to zero.)

This question is especially important for students who are not expected to implement or use a neural network system. Together with 20.15 and 20.17, it gives the student a concrete (if slender) grasp of what the network actually does. Many other similar questions can be devised.

Intuitively, the data suggest that a probabilistic prediction $P(Output = 1) = 0.8$ is appropriate. The network will adjust its weights to minimize the error function. The error is

$$E = \frac{1}{2}\sum_i (y_i - a_i)^2 = \frac{1}{2}[80(1-a_1)^2 + 20(0-a_1)^2] = 50O_1^2 - 80O_1 + 50$$

The derivative of the error with respect to the single output a_1 is

$$\frac{\partial E}{\partial a_1} = 100a_1 - 80$$

Setting the derivative to zero, we find that indeed $a_1 = 0.8$. The student should spot the connection to Ex. 18.8.

Exercise 22.NNRE

The neural network whose learning performance is measured in Figure **??** has four hidden nodes. This number was chosen somewhat arbitrarily. Use a cross-validation method to find the best number of hidden nodes.

This is just a simple example of the general cross-validation model-selection method described in the chapter. For each possible size of hidden layer up to some reasonable bound, the k-fold cross-validation score is obtained given the existing training data and the best hidden layer size is chosen. This can be done using the AIMA code or with any of several public-domain machine learning toolboxes such as WEKA.

Exercise 22.BPRE

Section 22.4.1 gives a generic formula for propagating gradients in computation graphs, embodied in Equations (22.11) and (22.12). Apply this process to the simple network in Figure 22.3 in order to rederive the gradient expressions in Equations (22.4) and (22.5).

[[need exercises]]

22.5 Generalization

Exercise 22.188-GENA
Origin:

`sp13_final_ml-generalization`

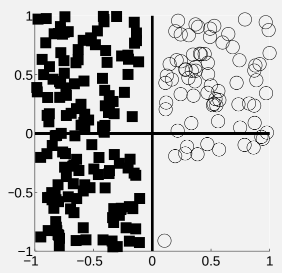
We consider the following different classifiers for classification of samples in a 2-dimensional feature space.

PNoBias	Linear perceptron *without* a bias term (features $\begin{bmatrix} x_1 & x_2 \end{bmatrix}^\intercal$)
PBias	Linear perceptron with a bias term (features $\begin{bmatrix} 1 & x_1 & x_2 \end{bmatrix}^\intercal$)
PQuad	Kernel perceptron with the quadratic kernel function $K(x, z) = (1 + x \cdot z)^2$
PCutoff	Kernel perceptron with the kernel function $K(x, z) = \max\{0, 0.01 - \|\|x - z\|\|_2\}$ ($\|\|a - b\|\|_2$ is the Euclidean distance between a and b)
1NN	1-nearest neighbor classifier
3NN	3-nearest neighbor classifier

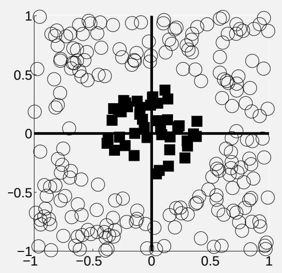
In each of the plots below you are given points from two classes, shown as filled rectangles and open circles. For each plot, list all each classifier that will be able to perfectly classify all of the training data (or, if none, mark "None of these will classify the data perfectly").

Note that when computing the nearest neighbors for a training data point, the training data point will be its own nearest neighbor.

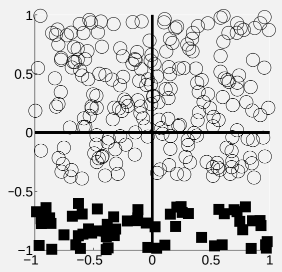
a.



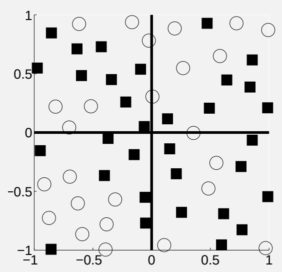
c.



b.



d.



a. PNoBias, PBias, PQuad, PCutoff, 1NN
3NN misclassifies the bottom-most circle.

b. PBias,PQuad,PCutoff,1NN,3NN
PNoBias is restricted to separators through the origin.

c. PQuad,PCutoff,1NN,3NN
The data are not linearly separable.

d. PCutoff,1NN,3NN
The decision boundary is complicated and in particular neither linear, nor quadratic. 1NN and PCutoff classify locally.

Exercise 22.188-OFUF

a. Suppose you train a classifier and test it on a held-out validation set. It gets 80% classification accuracy on the training set and 20% classification accuracy on the validation set.

(i) From what problem is your model most likely suffering? Underfitting or Overfitting?

(ii) To improve your classifier's performance on the validation set, should you add extra feature or remove some features? Please justify.

(iii) To improve your classifier's performance on the validation set, should you collect more training data or throw out some training data? Please justify.

b. Suppose you train a classifier and test it on a held-out validation set. It gets 30% classification accuracy on the training set and 30% classification accuracy on the validation set.

(i) From what problem is your model most likely suffering? Underfitting or Overfitting?

(ii) To improve your classifier's performance on the validation set, should you add extra feature or remove some features? Please justify.

(iii) To improve your classifier's performance on the validation set, should you collect more training data or throw out some training data? Please justify.

c. Your boss provides you with an image dataset in which some of the images contain your company's logo, and others contain competitors' logos. You are tasked to code up a classifier to distinguish your company's logos from competitors' logos. You complete the assignment quickly and even send your boss your code for training the classifier, but your boss is furious. Your boss says that when running your code with images and a random label for each of the images as input, the classifier achieved perfect accuracy on the training set. And this happens for all of the many random labelings that were generated.

Do you agree that this is a problem? Justify your answer.

a. (i) Overfitting

(ii) Either answer was accepted with justification.
Add extra features – adding some really good features could better capture the structure in the data. Remove some features – the model may be using the noise in the abundant feature set to overfit to the training data rather than learning any meaningful underlying structure.

(iii) Collect more data.
More data should yield a more representative sample of the true distribution of the data. Less data is more susceptible to overfitting.

b. (i) Underfitting

(ii) Add extra features.
Under the current feature representation, we are unable to accurately model the training data for the purpose of the classification task we're interested in. The classifier may be able to deduce more information about the connections between data points and their classes from additional features, allowing it to better model the data for the classification task. For example, a linear perceptron could not accurately model two classes separated by a circle in a 2-dimensional feature space, but by using quadratic features in a kernel perceptron, we can find a perfect separating hyperplane.

(iii) Collect more training data or neither.
More training data can only be a good thing. Neither was accepted, too, as given that train and hold-out validation already achieve the same performance, likely the underlying problem is not a lack of training data.

c. Yes, this is a problem.
The classifier is overfitting the training set. The fact that it had perfect accuracy with random labels suggests that it does not learn any real underlying structure in the data; it most likely essentially memorized each of the training cases.

22.6 Recurrent Neural Networks

[[need exercises]]

22.7 Unsupervised Learning and Transfer Learning

[[need exercises]]

22.8 Applications

[[need exercises]]