- 18.22 (FIXME) Suppose you have 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.

  - Assume the network has one hidden layer. For a given assignment of the weights *w,* write down equations for the value of the units in the output layer as a function of *w* and the input layer *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.

  - Repeat the previous calculation, but this time for a network with any number of hidden layers.

**18.22** 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 function restrict neural networks to represent only linearly 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.

- 18.23 (FIXME) Suppose that the training set contains only a single example, repeated 100 times. In 80/100, the output value is 1; in 20/100 the output is 0. What will a backpropagation algorithm predict for this example, assuming that it reaches the global optimum? (Hint: to find the global optimum, differentiate the error function and set it to zero)

**18.23** 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.

- 18.25 (FIXME) Consider the problem of separating N data points into positive and negative examples using a linear separator. Clearly, this can always be done for N=2 points on a line of dimension d=1, regardless of how the points are labeled of where they are located (unless the points are in the same place).

    - Show that it can always be done for N=3 points on a plane of d=2, unless they are collinear.

    - Show that it cannot always be done for N=4, d=2.

    - Show that it can always be done for N=4, d=3, unless they are coplanar.

    - Show that it cannot always be done for N=5, d=3.

**18.25** The main purpose of this exercise is to make concrete the notion of the *capacity* of a function class (in this case, linear halfspaces). It can be hard to internalize this concept, but the examples really help.

**a.** Three points in general position on a plane form a triangle. Any subset of the points can be separated from the rest by a line, as can be seen from the two examples in Figure S18.3(a).

**b.** Figure S18.3(b) shows two cases where the positive and negative examples cannot be separated by a line.

**c.** Four points in general position on a plane form a tetrahedron. Any subset of the points can be separated from the rest by a plane, as can be seen from the two examples in Figure S18.3(c).

**d.** Figure S18.3(d) shows a case where a negative point is inside the tetrahedron formed by four positive points; clearly no plane can separate the two sets.

**e.** Proof omitted.

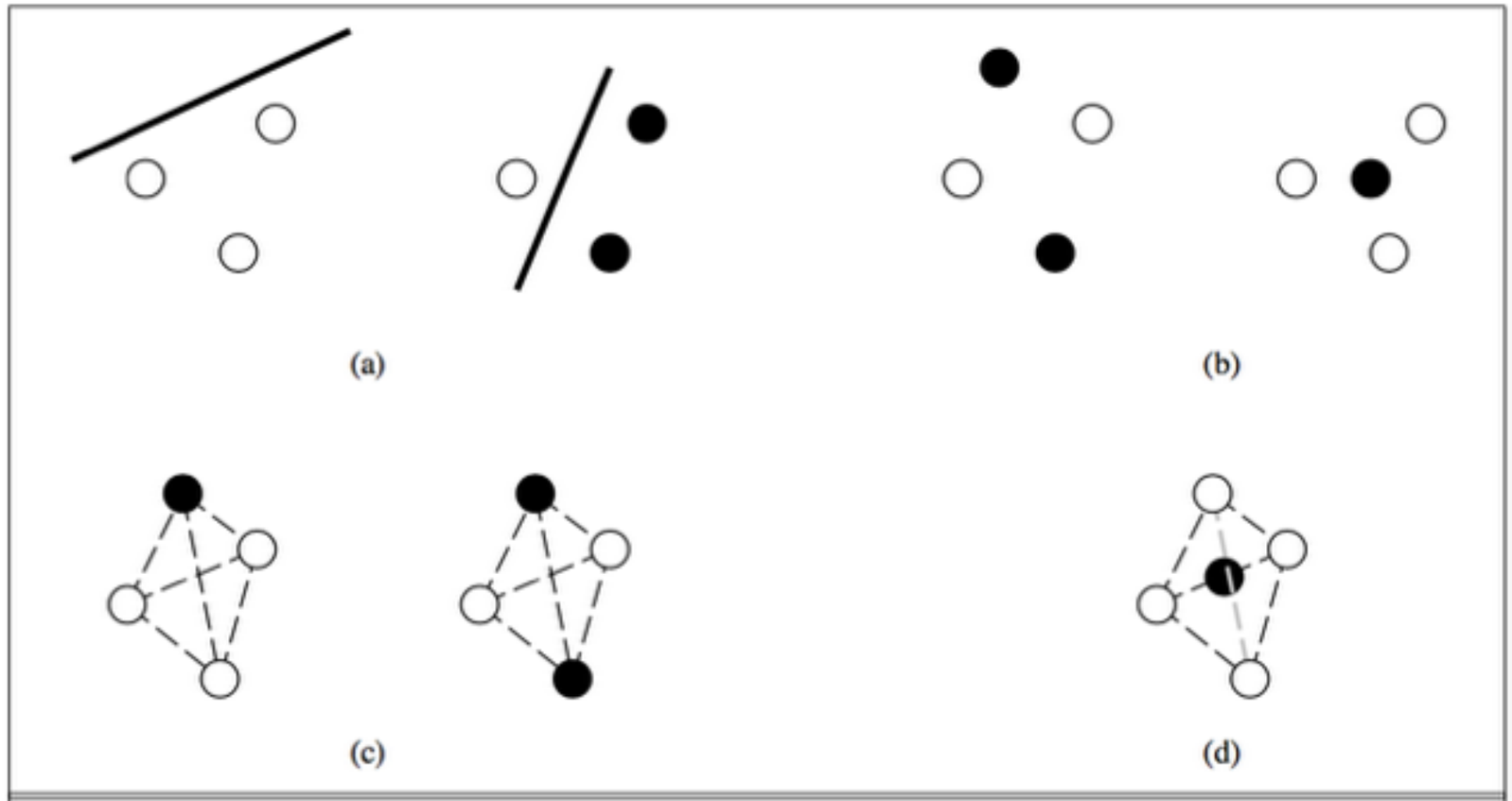
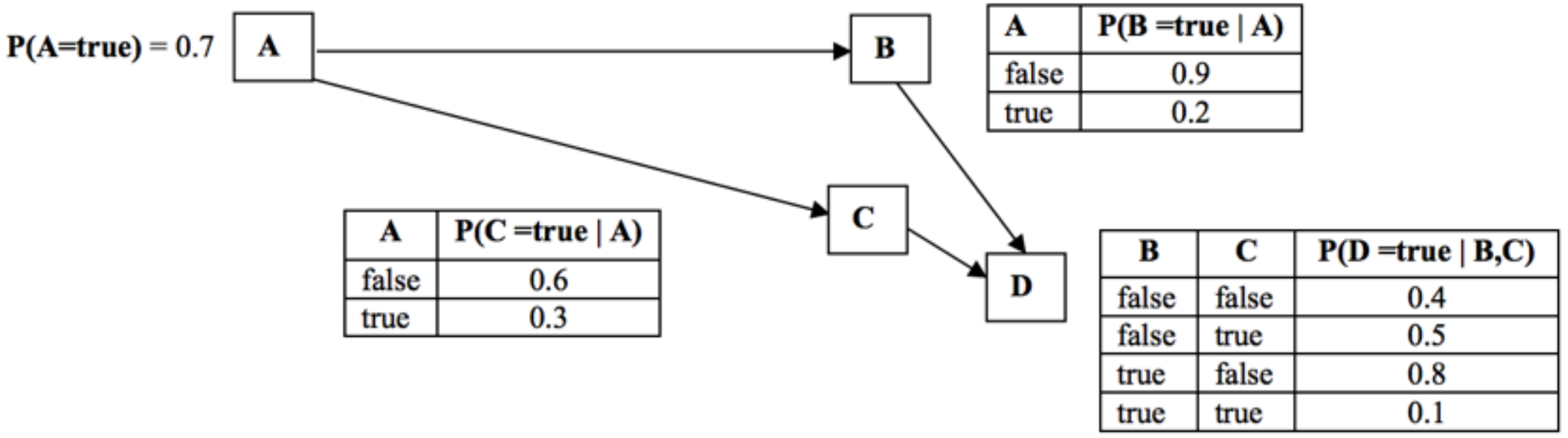



**Figure S18.3** Illustrative examples for VC dimensions.

- (FIXME) Define these variables:
  - F = had the flu
  - S = had the flu shot
- Assume:
  - P(F) = 0.75
  - P(S) = 0.5
  - P(F|S) = 0.1
- Given that you find out that someone had the flu, what is the probability that they had the flu shot?

- (FIXME) Consider the following Bayes net, where all values are Boolean-valued.




| A | P(B =true \| A) |
|---|---|
| false | 0.9 |
| true | 0.2 |

| A | P(C =true \| A) |
|---|---|
| false | 0.6 |
| true | 0.3 |

| B | C | P(D =true \| B,C) |
|---|---|---|
| false | false | 0.4 |
| false | true | 0.5 |
| true | false | 0.8 |
| true | true | 0.1 |

- What is the probability that A and B are true but C and D are false?
- What is the probability that B is true, A is false and D is true?
- What is the probability that B is true given that A is false and D is true?

- (FIXME) Consider the following training examples, where four Boolean-valued variables (X1 - X4) are used to predict a Boolean-valued output (Y):
  - 0 1 1 0 — 1
  - 1 0 1 0 — 0
  - 1 0 0 1 — 1
  - 0 1 1 1 — 0
- Which feature would be chosen as the root node of a decision tree, if information gain was used as the scoring function?
- Draw a single-layer perceptron for this task, initializing the weights and biases to -0.1.
- Show how the weights and biases would be changed after processing the first training example above. Use a learning rate of 0.5.