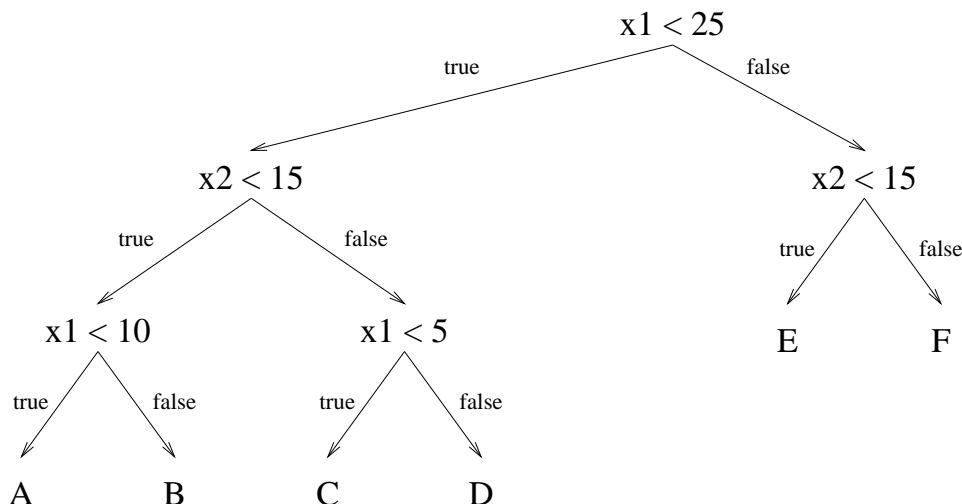
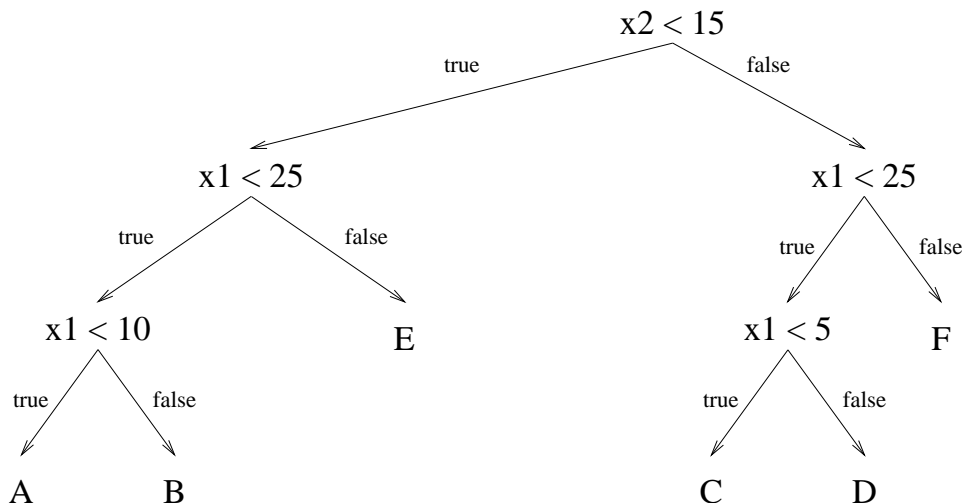


CS534 — Homework Assignment 4 — Solutions

1. Consider the following decision tree:



Give another decision tree that is syntactically different but defines the same decision boundaries. This demonstrates that the space of decision trees is syntactically redundant. Is this redundancy a statistical problem (i.e., does it affect the accuracy of the learned trees)? Is it a computational problem (i.e., does it increase the computational complexity of finding an accurate tree)?



This redundancy is not likely to be a statistical problem, because it has no effect on the expressive power of decision trees. That is, the same space of decision trees can be represented with or without the redundancy. The redundancy is likely to be a computational advantage, because it makes it easier for an imperfect greedy heuristic to find a good solution. For depth-first search methods (such as our greedy algorithm), the ideal search space is one such that every path leads to a solution. Redundancy increases the chance that any random sequence of node expansions will lead to a good tree.

2. In the basic decision tree algorithm, we choose the feature/value pair with the maximum mutual information as the test to use at each internal node of the decision tree. Suppose we modified the algorithm to choose at random from among those feature/value combinations that had non-zero mutual information, but that we kept all other parts of the algorithm unchanged.

(a) Prove that if a splitting feature/value combination has non-zero mutual information at an internal node, then at least one training example must be sent to each of the child nodes.

Proof: We prove the contrapositive of the statement (which also proves the original statement) which is “if all examples are sent to one of the child nodes then the mutual information is zero”. Let (p, n) be the number of positive and negative examples at the internal node. Assume that the feature/value split under consideration sends all the examples to the left child. Then the child will also have (p, n) examples, and the right child will receive $(0, 0)$. Now we plug into the formula for mutual information. The mutual information will be

$$H(p, n) - [1.0H(p, n) + 0.0H(0, 0)] = 0$$

This completes the proof of the contrapositive and hence of the original statement.

(b) What is the maximum number of leaf nodes that such a decision tree could contain if it were trained on m training examples?

The maximum number would be obtained if each training example were alone in its own leaf: m leaves. We can have no leaves with zero training examples, because each split has non-zero mutual information and therefore sends at least one example to each child. (Actually, let m' be the number of distinct training examples. Then no tree can have more than m' leaves.)

(c) What is the maximum number of leaf nodes that a decision tree could contain if it were trained on m training examples using the original maximum mutual information version of the algorithm? Is it bigger, smaller, or the same as your answer to (b)?

Using maximum mutual information, in the worst case we could also get one example in each leaf node. (or m' distinct examples and leaves).

(d) How do you think this change would affect the accuracy of the decision trees produced on average? Why?

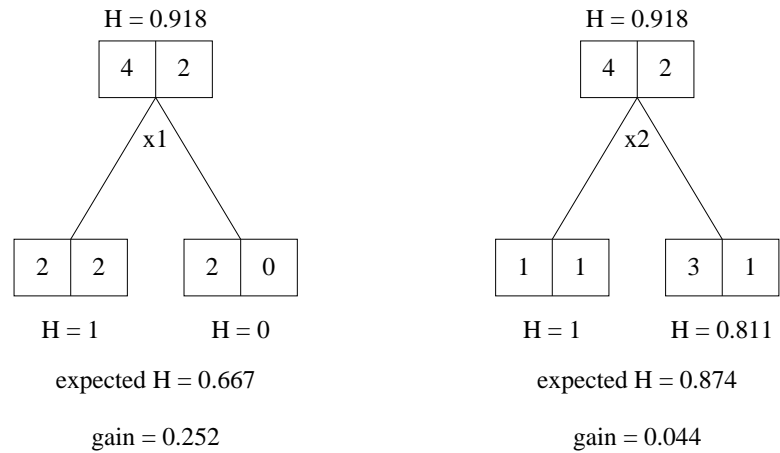
Although in the worst case, both decision trees will have the same size, I would guess that in the average case, using randomized splits would give lower accuracy, particularly if there are irrelevant or noisy features in the data. A random split is more likely to split on an irrelevant or inappropriate feature. This will have the unfortunate result of subdividing the data randomly. This effectively creates two learning problems equivalent to the unsplit learning problem, but each has only half as much data. We know that the more data we have available, the more accurate the hypothesis will be on the average. Conversely, the less data we have available, the less accurate the hypothesis will be on the average.

3. Consider the following training examples:

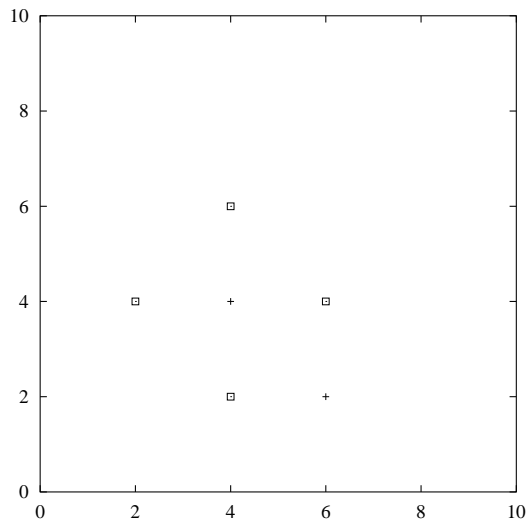
x_1	x_2	y
0	1	0
1	1	0
0	0	1
1	0	0
0	1	0
0	1	1

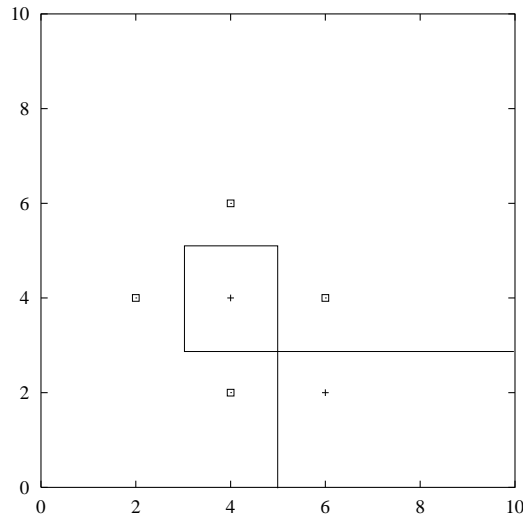
What feature would be chosen for the split at the root of a decision tree using the Mutual Information criterion? Show your work.

The decision tree will choose x_1 :



4. Consider the set of training examples shown in the diagram below.
- Draw the decision boundaries for the nearest neighbor algorithm assuming that we are using standard Euclidean distance to compute nearest neighbors. A plus indicates a positive example and a small square indicates a negative example.





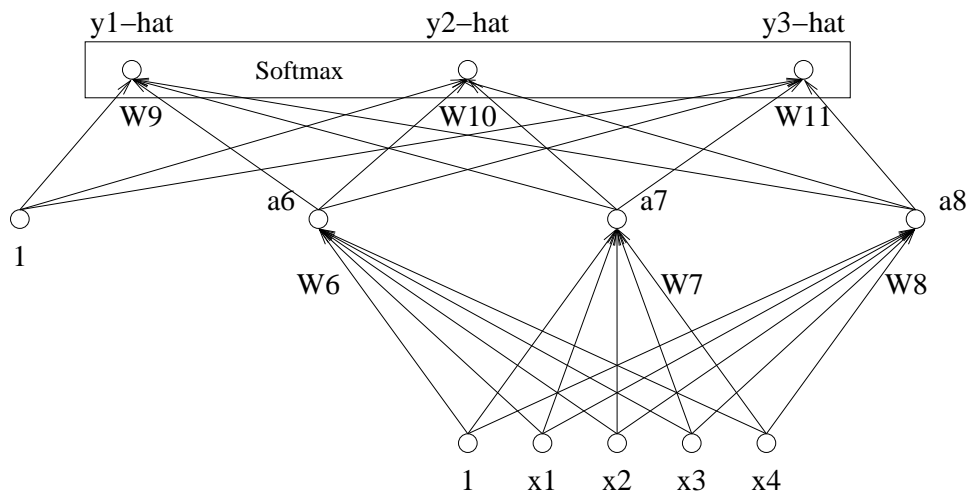
b. How will the point (8, 1) be classified by the nearest-neighbor classifier?

Positive

c. How will the point (8, 8) be classified?

Negative

5. Lets consider an alternative neural network diagram different from the lectures. In this diagram, a neural network uses a softmax activation function for its output layer. Its outputs can be interpreted as posterior probabilities $P(y|x)$ for a categorical target variable y . Consider the following neural network with three output units. The softmax activation function is defined as: $\hat{y}_i = \frac{\exp(x_i)}{\sum_{j=1}^3 \exp(x_j)}$ (as opposed to what we saw in class $\hat{y}_i = \frac{1}{1+\exp(-x_i)}$), where x_i is the net input input the activation function of the output node i . Note that $\hat{y}_1 + \hat{y}_2 + \hat{y}_3 = 1$, making it a valid posterior probability. In this problem, we will compute the derivatives needed for the backpropagation algorithm for this kind of network.



- a. Write down the log likelihood objective function $J(\mathbf{w})$ for this network, where \mathbf{w} is the concatenation of $W6, W7, W8, W9, W10$, and $W11$. You may assume that each training

example has the form (\mathbf{x}, y) , where $\mathbf{x} = (1, x_1, x_2, x_3, x_4)$ and $y = (y_1, y_2, y_3)$. There are only three possible y values: $y = (1, 0, 0)$, $y = (0, 1, 0)$, and $y = (0, 0, 1)$.

$$J(\mathbf{w}) = \sum_i \sum_k -y_{i,k} \log P(y_{i,k} | \mathbf{x}_i)$$

where $P(y_{i,k} | \mathbf{x}) = \hat{y}_{i,k}$ is the predicted probability of class k for training example i . I have included a minus sign so that our goal is to minimize J , just as it was for squared error loss.

Let $Z = \exp a_9 + \exp a_{10} + \exp a_{11}$ be the normalizer of the softmax. Then we can rewrite J as

$$\begin{aligned} J(\mathbf{w}) &= \sum_i -y_{i,1} \log \hat{y}_1 - y_{i,2} \log \hat{y}_2 - y_{i,3} \log \hat{y}_3 \\ &= \sum_i -y_{i,1} \log \frac{e^{a_9}}{Z} - y_{i,2} \log \frac{e^{a_{10}}}{Z} - y_{i,3} \log \frac{e^{a_{11}}}{Z} \\ &= -y_{i,1} a_9 - y_{i,2} a_{10} - y_{i,3} a_{11} + (y_{i,1} + y_{i,2} + y_{i,3}) \log Z \\ &= -y_{i,1} a_9 - y_{i,2} a_{10} - y_{i,3} a_{11} + \log Z \end{aligned}$$

b. Compute the partial derivative

$$\frac{\partial J(\mathbf{w})}{\partial w_{9,6}}$$

Let's consider just one training example (\mathbf{x}, y) and drop the subscript i . The objective function becomes

$$J(\mathbf{w}) = -y_1 a_9 - y_2 a_{10} - y_3 a_{11} + \log Z$$

where exactly one of the y_k is non-zero.

$$\frac{\partial J(\mathbf{w})}{\partial w_{9,6}} = -y_1 \frac{\partial a_9}{\partial w_{9,6}} - y_2 \frac{\partial a_{10}}{\partial w_{9,6}} - y_3 \frac{\partial a_{11}}{\partial w_{9,6}} + \frac{\partial \log Z}{\partial w_{9,6}}$$

Now a_{10} and a_{11} do not involve $w_{9,6}$, so these partial derivatives are zero. Hence we just have

$$\frac{\partial J(\mathbf{w})}{\partial w_{9,6}} = -y_1 \frac{\partial a_9}{\partial w_{9,6}} + \frac{\partial \log Z}{\partial w_{9,6}}$$

Now $a_9 = W_9 \cdot A$, where A is the vector of hidden unit activations. Expanding the dot product, $a_9 = \sum_u w_{9,u} a_u$. From this, we obtain:

$$\frac{\partial J(\mathbf{w})}{\partial w_{9,6}} = -y_1 a_6 + \frac{\partial \log Z}{\partial w_{9,6}}$$

Now let's compute $\partial \log Z / \partial w_{9,6}$:

$$\begin{aligned} \frac{\partial \log Z}{\partial w_{9,6}} &= \frac{1}{Z} \frac{\partial Z}{\partial w_{9,6}} \\ &= \frac{1}{Z} \frac{\partial \exp a_9 + \exp a_{10} + \exp a_{11}}{\partial w_{9,6}} \end{aligned}$$

Only the first term (a_9) depends on $w_{9,6}$, so we obtain

$$\begin{aligned}\frac{\partial \log Z}{\partial w_{9,6}} &= \frac{1}{Z} \frac{\partial \exp a_9}{\partial w_{9,6}} \\ &= \frac{\exp a_9}{Z} \frac{\partial a_9}{\partial w_{9,6}} \\ &= \hat{y}_1 a_6\end{aligned}$$

Plugging this back into the original derivation, we have

$$\begin{aligned}\frac{\partial J(\mathbf{w})}{\partial w_{9,6}} &= -y_1 a_6 + \frac{\partial \log Z}{\partial w_{9,6}} = -y_1 a_6 + \hat{y}_1 a_6 \\ &= a_6(\hat{y}_1 - y_1)\end{aligned}$$

As with sigmoidal neural networks, it is useful to define

$$\delta_9 = \hat{y}_1 - y_1$$

c. Compute the partial derivative

$$\frac{\partial J(\mathbf{w})}{\partial w_{6,3}}$$

This is the same as for squared error loss, assuming that we have redefined δ as above:

$$\delta_6 = a_6(1 - a_6) \sum_{u=9}^{11} w_{u,6} \delta_u.$$

d. Generalize your answers to (b) and (c) and write the pseudo-code for the backpropagation algorithm using them.

- **Forward pass.** Compute a_u and \hat{y}_v for hidden units u and output units v .
- **Compute Errors.** Compute $\delta_v = (\hat{y}_v - y_v)$ for each output unit.
- **Compute Hidden Unit Deltas.** Compute $\delta_u = a_u(1 - a_u) \sum_v w_{v,u} \delta_v$.
- **Compute Gradient.** Compute $\frac{\partial J_i}{\partial w_{u,j}} = \delta_u x_{ij}$ for input-to-hidden weights. Compute $\frac{\partial J_i}{\partial w_{v,u}} = \delta_v a_{iu}$ for hidden-to-output weights.
- **Take Gradient Descent Step**

$$W := W - \eta \nabla_W J(\mathbf{x}_i)$$

6. Cubic Kernels. In class, we showed that the quadratic kernel $K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j + 1)^2$ was equivalent to mapping each \mathbf{x} into a higher dimensional space where

$$\Phi(\mathbf{x}) = (x_1^2, x_2^2, \sqrt{2}x_1x_2, \sqrt{2}x_1, \sqrt{2}x_2, 1)$$

for the case where $\mathbf{x} = (x_1, x_2)$. Now consider the cubic kernel $K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j + 1)^3$. What is the corresponding Φ function (again, for the special case where $\mathbf{x} = (x_1, x_2)$)?

Let $\mathbf{x}_i = (x_{i1}, x_{i2})$ and $\mathbf{x}_j = (x_{j1}, x_{j2})$. Then the kernel is

$$\begin{aligned}
K(\mathbf{x}_i, \mathbf{x}_j) &= (x_{i1}x_{j1} + x_{i2}x_{j2} + 1)^3 \\
&= (x_{i1}x_{j1} + x_{i2}x_{j2} + 1)^2 \cdot (x_{i1}x_{j1} + x_{i2}x_{j2} + 1) \\
&= (x_{i1}^2x_{j1}^2 + 2x_{i1}x_{i2}x_{j1}x_{j2} + x_{i2}^2x_{j2}^2 + 2x_{i1}x_{j1} + 2x_{i2}x_{j2} + 1) \cdot (x_{i1}x_{j1} + x_{i2}x_{j2} + 1) \\
&= x_{i1}^3x_{j1}^3 + 3x_{i1}^2x_{j1}^2 + 3x_{i1}x_{j1} + \\
&\quad 3x_{i1}^2x_{j1}^2x_{i2}x_{j2} + 6x_{i1}x_{j1}x_{i2}x_{j2} + 3x_{i1}x_{j1}x_{i2}^2x_{j2}^2 + \\
&\quad 3x_{i2}x_{j2} + 3x_{i2}^2x_{j2}^2 + x_{i2}^3x_{j2}^3 + 1 \\
&= (x_{i1}^3, \sqrt{3}x_{i1}^2, \sqrt{3}x_{i1}, \sqrt{3}x_{i1}^2x_{i2}, \sqrt{6}x_{i1}x_{i2}, \sqrt{3}x_{i1}x_{i2}^2, \sqrt{3}x_{i2}, \sqrt{3}x_{i2}^2, x_{i2}^3, 1) \cdot \\
&\quad (x_{j1}^3, \sqrt{3}x_{j1}^2, \sqrt{3}x_{j1}, \sqrt{3}x_{j1}^2x_{j2}, \sqrt{6}x_{j1}x_{j2}, \sqrt{3}x_{j1}x_{j2}^2, \sqrt{3}x_{j2}, \sqrt{3}x_{j2}^2, x_{j2}^3, 1)
\end{aligned}$$

Hence, the function $\Phi(\mathbf{x}) = (x_1^3, \sqrt{3}x_1^2, \sqrt{3}x_1, \sqrt{3}x_1^2x_2, \sqrt{6}x_1x_2, \sqrt{3}x_1x_2^2, \sqrt{3}x_2, \sqrt{3}x_2^2, x_2^3, 1)$

7. Geometry of Lines and Points. Consider the line $2x_1 + 3x_2 - 1 = 0$. What is the distance from this line to the origin (at the point where the line is closest to the origin)? Now consider a point $(x_1, x_2) = (5, 1)$. What is the distance from this point to the line?

In general, consider the line $w_1x_1 + w_2x_2 + b = 0$. What is the general formula for the distance from this line to the origin? What is the general formula for the distance from this line to some arbitrary point (u, v) ?

Let \mathbf{x}_p be the point on the line $\mathbf{w} \cdot \mathbf{x} + b = 0$ closest to the origin. We know that the vector joining \mathbf{x}_p to the origin points perpendicular to the line $\mathbf{w} \cdot \mathbf{x} + b = 0$. Let d be the distance between \mathbf{x}_p and the origin. Then we can write the origin O as

$$O = \mathbf{x}_p + d \frac{\mathbf{w}}{\|\mathbf{w}\|}$$

Solve this for \mathbf{x}_p to obtain

$$\begin{aligned}
\mathbf{x}_p &= O - d \frac{\mathbf{w}}{\|\mathbf{w}\|} \\
&= -d \frac{\mathbf{w}}{\|\mathbf{w}\|}
\end{aligned}$$

We also know that \mathbf{x}_p lies on the line $\mathbf{w} \cdot \mathbf{x} + b = 0$. So

$$\mathbf{w} \cdot \mathbf{x}_p + b = 0.$$

Let us substitute the formula for \mathbf{x}_p into this to obtain

$$\mathbf{w} \cdot \left(-d \frac{\mathbf{w}}{\|\mathbf{w}\|} \right) + b = 0$$

Recall that $\mathbf{w} \cdot \mathbf{w} = \|\mathbf{w}\|^2$, so we obtain

$$\begin{aligned}
-d\|\mathbf{w}\| + b &= 0 \\
b &= d\|\mathbf{w}\| \\
d &= \frac{b}{\|\mathbf{w}\|}
\end{aligned}$$

d could be either positive or negative, so the correct answer is $\frac{|b|}{\|\mathbf{w}\|}$.

To answer the first part of the problem, $\mathbf{w} = (2, 3)$, so $\|\mathbf{w}\| = 3.606$. Hence, the distance is $1/3.606 = 0.277$.

In general, let \mathbf{u} be an arbitrary point. Let \mathbf{x}_p be the point on the line $\mathbf{w} \cdot \mathbf{x} + b = 0$ nearest to \mathbf{u} . Then

$$\begin{aligned}\mathbf{u} - \mathbf{x}_p &= d \frac{\mathbf{w}}{\|\mathbf{w}\|} \\ \mathbf{x}_p &= \mathbf{u} - d \frac{\mathbf{w}}{\|\mathbf{w}\|}\end{aligned}$$

Substituting this into the decision boundary, we obtain

$$\begin{aligned}\mathbf{w} \cdot \left(\mathbf{u} - d \frac{\mathbf{w}}{\|\mathbf{w}\|} \right) + b &= 0 \\ \mathbf{w} \cdot \mathbf{u} - d \|\mathbf{w}\| + b &= 0 \\ d &= \frac{\mathbf{w} \cdot \mathbf{u} + b}{\|\mathbf{w}\|}\end{aligned}$$

Again, this could be either positive or negative, so the distance will be the absolute value of this formula. From this, we see that the distance between any arbitrary point \mathbf{u} and the decision boundary is obtained by plugging \mathbf{u} into the classifier $\mathbf{w} \cdot \mathbf{x} + b$ and then normalizing the result by $\|\mathbf{w}\|$. This is the margin of the decision boundary to \mathbf{u} .

Hence, to answer the second question, the distance to the point $(5, 1)$ is $|(2, 3) \cdot (5, 1) - 1|/\sqrt{13} = 12/\sqrt{13} = 3.328$.

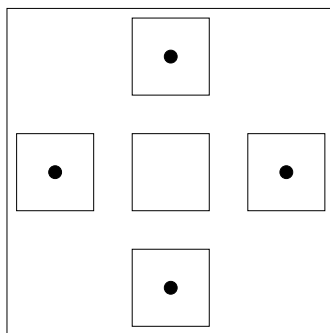
8. VC Dimension of geometric concept classes.

Consider the space of instances X corresponding to all points in the (x, y) plane. Give the VC dimension of the following hypothesis spaces:

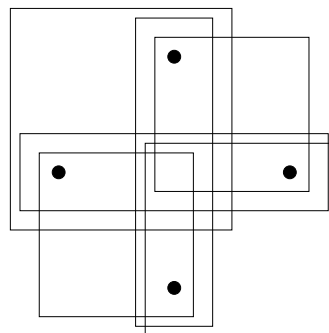
- (a) H_r = the set of all rectangles in the (x, y) plane. That is, $H = \{(a < x < b) \wedge (c < y < d) \mid a, b, c, d \in \mathbb{R}\}$.

Since it takes 4 parameters to represent a rectangle, we can guess that the VC dimension is no larger than 4. The following 4 points can be shattered:

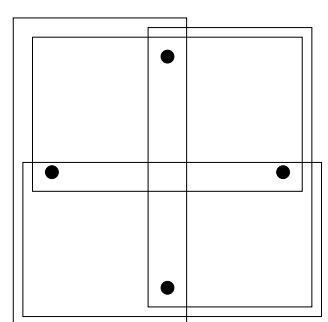
Sets of size 0, 1, and 4



Sets of size 2

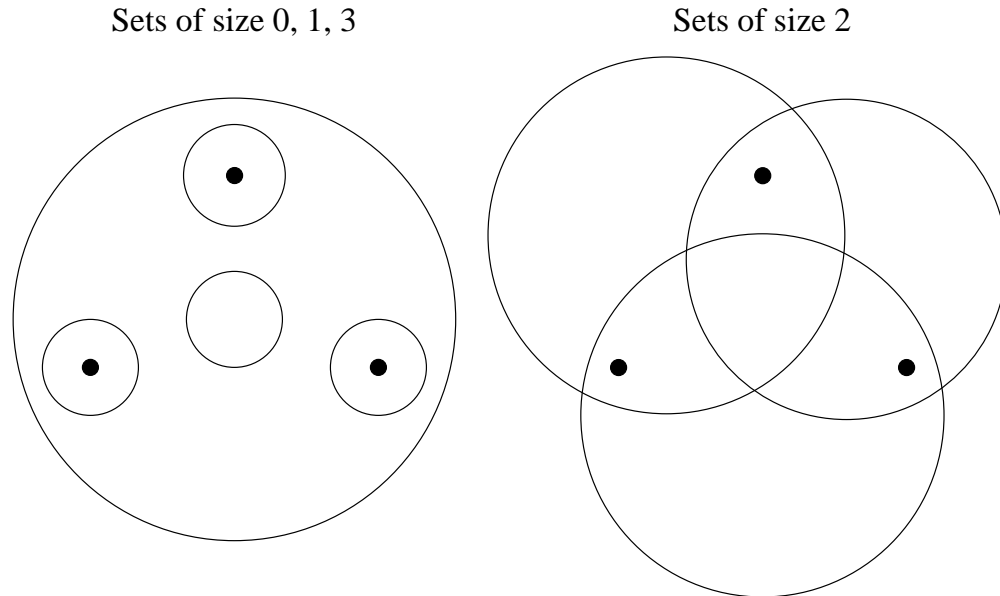


Sets of size 3



With 5 points, you can't capture all sets of 3 points.

- (b) $H_c =$ circles in the (x, y) plane. Points inside the circle are classified as positive examples. It takes only 3 parameters to specify a circle, so we can guess that the VC dimension is no larger than 3. The following 3 points can be shattered:



With 4 points, you can't get all sets of size 2 (a circle enclosing pairs most distant from one another will enclose the others as well, by definition).

9. Consider the class C of concepts of the form $(a \leq x \leq b) \wedge (c \leq y \leq d)$, where $a, b, c,$ and d are integers in the interval $[0, 99]$. Note that each concept in this class corresponds to a rectangle with integer-valued boundaries on a portion of the (x, y) plane. Hint: Given a region in the plane bounded by the points $(0, 0)$ and $(n - 1, n - 1)$, the number of distinct rectangles with integer-valued boundaries within this region is $\left(\frac{n(n-1)}{2}\right)^2$.

- (a) Give an upper bound on the number of randomly drawn training examples sufficient to assure that for any target concept c in C , any consistent learner using $H = C$ will, with probability 95%, output a hypothesis with error at most 0.15.

We will compute the size of this hypothesis space and then apply the bound

$$m \geq \frac{1}{\epsilon} \left[\ln |H| + \log \frac{1}{\delta} \right]$$

For $a \leq x \leq b$ and $x \in \{0, 1, \dots, 99\}$, there are $\frac{100 \cdot 99}{2} = 4950$ legal combinations. The same argument holds for $c \leq y \leq d$. So the total number of axis-parallel rectangles is 24,502,500. The problem specifies $\epsilon = 0.15$ and $\delta = 0.05$, so plugging in, we get (in lisp notation):

```
(* (/ 1.0 0.15) (+ (log 24502500) (log (/ 1.0 0.05)))) =
(* 6.666666666666667 (+ 17.01428775173149 2.995732273553991)) =
133.4001335019032
```

So we need at least 134 examples.

- (b) Now suppose the rectangle boundaries $a, b, c,$ and d take on *real* values instead of integer values. Update your answer to the first part of this question.

We will compute the VC dimension of the space and apply the bound

$$m \geq \frac{1}{\epsilon} [4 \lg(2/\delta) + 8d \lg(13/\epsilon)]$$

From the previous problem, part (a), the VC dimension is 4. So this becomes

$$\begin{aligned} & (* (/ 1.0 0.15) \\ & (+ (* 4 (\log (/ 2.0 0.05) 2)) \\ & (* 8 4 (\log (/ 13.0 0.15) 2.0)))) = \\ & (* 6.666666667 \\ & (+ 21.28771237954945 205.99696999383357)) = \\ & 1515.2312 \end{aligned}$$

So we need 1,516 examples.

10. Consider the class C of concepts of boolean conjunction over n boolean variables as discussed in the lectures. Prove that C is PAC-learnable.

In class we showed that the size of this hypothesis space was $|H| = 3^n$ indicating that a consistent learning algorithm need only

$$m \geq \epsilon^{-1} (n \ln 3 + \ln \delta^{-1})$$

to guarantee the PAC generalization requirements. Since this sufficient size for m is polynomial in the appropriate quantities all that remains is to show that we can find a consistent hypothesis in polynomial time.

Given a set of training examples labelled as positive and negative we will return the conjunction that includes a positive version of a feature if and only if that feature is true in all positive examples and includes the negation of a feature iff that feature is negative in all of the positive examples. Thus, if a feature is positive for some positive examples and negative for other positive examples then it is not included in the returned conjunction.

It should be easy to see that the returned conjunction will correctly label the positive examples as positive since it must be consistent with all of them. Furthermore we know that the returned conjunction must include at least all of the terms in the true underlying conjunction. This is because all of the positive examples must agree with the true underlying conjunction. From this we can infer that the returned conjunction will classify all negative examples in the training data as negative, since we know that each example must disagree with the true conjunction on at least one feature and hence will disagree with our returned hypothesis on that feature. Thus our returned hypothesis is consistent with the entire training set. Clearly we can compute the hypothesis in polynomial time which completes the proof that the hypothesis space of conjunctions is PAC-learnable.