concept-learning-exercises

March 26, 2019

1 18. Learning from Examples

18.1 [infant-language-exercise]Consider the problem faced by an infant learning to speak and understand a language. Explain how this process fits into the general learning model. Describe the percepts and actions of the infant, and the types of learning the infant must do. Describe the subfunctions the infant is trying to learn in terms of inputs and outputs, and available example data.

18.2 Repeat Exercise Section **??** for the case of learning to play tennis (or some other sport with which you are familiar). Is this supervised learning or reinforcement learning?

18.3 Draw a decision tree for the problem of deciding whether to move forward at a road intersection, given that the light has just turned green.

18.4 We never test the same attribute twice along one path in a decision tree. Why not?

18.5 Suppose we generate a training set from a decision tree and then apply decision-tree learning to that training set. Is it the case that the learning algorithm will eventually return the correct tree as the training-set size goes to infinity? Why or why not?

18.6 [leaf-classification-exercise]In the recursive construction of decision trees, it sometimes happens that a mixed set of positive and negative examples remains at a leaf node, even after all the attributes have been used. Suppose that we have p positive examples and n negative examples.

1. Show that the solution used by DECISION-TREE-LEARNING, which picks the majority classification, minimizes the absolute error over the set of examples at the leaf.

2. Show that the **class probability** $p/(p+n)$ minimizes the sum of squared errors.

18.7 [nonnegative-gain-exercise]Suppose that an attribute splits the set of examples E into subsets E_k and that each subset has p_k positive examples and n_k negative examples. Show that the attribute has strictly positive information gain unless the ratio $p_k/(p_k+n_k)$ is the same for all k.

18.8 Consider the following data set comprised of three binary input attributes (A_1, A_2, and A_3) and one binary output:

Example	A_1	A_2	A_3	$Output y$
$\mathbf{x}_1$	1	0	0	0
$\mathbf{x}_2$	1	0	1	0
$\mathbf{x}_3$	0	1	0	0
$\mathbf{x}_4$	1	1	1	1
$\mathbf{x}_5$	1	1	0	1

Use the algorithm in Figure Section **??** (page Section **??**) to learn a decision tree for these data. Show the computations made to determine the attribute to split at each node.

18.9 Construct a data set (set of examples with attributes and classifications) that would cause the decision-tree learning algorithm to find a non-minimal-sized tree. Show the tree constructed by the algorithm and the minimal-sized tree that you can generate by hand.

18.10 A decision *graph* is a generalization of a decision tree that allows nodes (i.e., attributes used for splits) to have multiple parents, rather than just a single parent. The resulting graph must still be acyclic. Now, consider the XOR function of *three* binary input attributes, which produces the value 1 if and only if an odd number of the three input attributes has value 1.

1. Draw a minimal-sized decision *tree* for the three-input XOR function.

2. Draw a minimal-sized decision *graph* for the three-input XOR function.

18.11 [pruning-DTL-exercise]This exercise considers χ^2 pruning of decision trees (Section Section **??**).

1. Create a data set with two input attributes, such that the information gain at the root of the tree for both attributes is zero, but there is a decision tree of depth 2 that is consistent with all the data. What would χ^2 pruning do on this data set if applied bottom up? If applied top down?

2. Modify DECISION-TREE-LEARNING to include χ^2-pruning. You might wish to consult Quinlan [@Quinlan:1986] or [@Kearns+Mansour:1998] for details.

18.12 [missing-value-DTL-exercise]The standard DECISION-TREE-LEARNING algorithm described in the chapter does not handle cases in which some examples have missing attribute values.

1. First, we need to find a way to classify such examples, given a decision tree that includes tests on the attributes for which values can be missing. Suppose that an example $\mathbf{x}$ has a missing value for attribute A and that the decision tree tests for A at a node that $\mathbf{x}$ reaches. One way to handle this case is to pretend that the example has *all* possible values for the attribute, but to weight each value according to its frequency among all of the examples that reach that node in the decision tree. The classification algorithm should follow all branches at any node for which a value is missing and should multiply the weights along each path. Write a modified classification algorithm for decision trees that has this behavior.

2. Now modify the information-gain calculation so that in any given collection of examples C at a given node in the tree during the construction process, the examples with missing values for any of the remaining attributes are given "as-if" values according to the frequencies of those values in the set C.

18.13 [gain-ratio-DTL-exercise]In Section Section **??**, we noted that attributes with many different possible values can cause problems with the gain measure. Such attributes tend to split the examples into numerous small classes or even singleton classes, thereby appearing to be highly relevant according to the gain measure. The **gain-ratio** criterion selects attributes according to the ratio between their gain and their intrinsic information content—that is, the amount of information contained in the answer to the question, "What is the value of this attribute?" The gain-ratio criterion therefore tries to measure how efficiently an attribute provides information on the correct classification of an example. Write a mathematical expression for the information content of an attribute, and implement the gain ratio criterion in DECISION-TREE-LEARNING.

18.14 Suppose you are running a learning experiment on a new algorithm for Boolean classification. You have a data set consisting of 100 positive and 100 negative examples. You plan to use leave-one-out cross-validation and compare your algorithm to a baseline function, a simple majority classifier. (A majority classifier is given a set of training data and then always outputs the class that is in the majority in the training set, regardless of the input.) You expect the majority classifier to score about 50% on leave-one-out cross-validation, but to your surprise, it scores zero every time. Can you explain why?

18.15 Suppose that a learning algorithm is trying to find a consistent hypothesis when the classifications of examples are actually random. There are n Boolean attributes, and examples are drawn uniformly from the set of 2^n possible examples. Calculate the number of examples required before the probability of finding a contradiction in the data reaches 0.5.

18.16 Construct a *decision list* to classify the data below. Select tests to be as small as possible (in terms of attributes), breaking ties among tests with the same number of attributes by selecting the one that classifies the greatest number of examples correctly. If multiple tests have the same number of attributes and classify the same number of examples, then break the tie using attributes with lower index numbers (e.g., select A_1 over A_2).

	A_1	A_2	A_3	A_y	y
$\mathbf{x}_1$	1	0	0	0	1
$\mathbf{x}_2$	1	0	1	1	1
$\mathbf{x}_3$	0	1	0	0	1
$\mathbf{x}_4$	0	1	1	0	0
$\mathbf{x}_5$	1	1	0	1	1
$\mathbf{x}_6$	0	1	0	1	0
$\mathbf{x}_7$	0	0	1	1	1
$\mathbf{x}_8$	0	0	1	0	0

18.17 Prove that a decision list can represent the same function as a decision tree while using at most as many rules as there are leaves in the decision tree for that function. Give an example of a function represented by a decision list using strictly fewer rules than the number of leaves in a minimal-sized decision tree for that same function.

18.18 [DL-expressivity-exercise]This exercise concerns the expressiveness of decision lists (Section Section **??**).

1. Show that decision lists can represent any Boolean function, if the size of the tests is not limited.

2. Show that if the tests can contain at most k literals each, then decision lists can represent any function that can be represented by a decision tree of depth k.

18.19 [knn-mean-mode] Suppose a 7-nearest-neighbors regression search returns $ {7, 6, 8, 4, 7, 11, 100} $ as the 7 nearest y values for a given x value. What is the value of $\hat{y}$ that minimizes the L_1 loss function on this data? There is a common name in statistics for this value as a function of the y values; what is it? Answer the same two questions for the L_2 loss function.

18.20 [knn-mean-mode] Suppose a 7-nearest-neighbors regression search returns $ {4, 2, 8, 4, 9, 11, 100} $ as the 7 nearest y values for a given x value. What is the value of $\hat{y}$ that minimizes the L_1 loss function on this data? There is a common name in statistics for this value as a function of the y values; what is it? Answer the same two questions for the L_2 loss function.

18.21 [svm-ellipse-exercise] Figure Section **??** showed how a circle at the origin can be linearly separated by mapping from the features (x_1, x_2) to the two dimensions (x_1^2, x_2^2). But what if the circle is not located at the origin? What if it is an ellipse, not a circle? The general equation for a circle (and hence the decision boundary) is $(x_1 - a)^2 + (x_2 - b)^2 - r^2 = 0$, and the general equation for an ellipse is $c(x_1 - a)^2 + d(x_2 - b)^2 - 1 = 0$.

1. Expand out the equation for the circle and show what the weights w_i would be for the decision boundary in the four-dimensional feature space (x_1, x_2, x_1^2, x_2^2). Explain why this means that any circle is linearly separable in this space.

2. Do the same for ellipses in the five-dimensional feature space $(x_1, x_2, x_1^2, x_2^2, x_1 x_2)$.

18.22 [svm-exercise] Construct a support vector machine that computes the xor function. Use values of +1 and –1 (instead of 1 and 0) for both inputs and outputs, so that an example looks like $([-1, 1], 1)$ or $([-1, -1], -1)$. Map the input $[x_1, x_2]$ into a space consisting of x_1 and $x_1\, x_2$. Draw the four input points in this space, and the maximal margin separator. What is the margin? Now draw the separating line back in the original Euclidean input space.

18.23 [ensemble-error-exercise] Consider an ensemble learning algorithm that uses simple majority voting among K learned hypotheses. Suppose that each hypothesis has error ϵ and that the errors made by each hypothesis are independent of the others'. Calculate a formula for the error of the ensemble algorithm in terms of K and ϵ, and evaluate it for the cases where $K = 5$, 10, and 20 and $\epsilon = 0.1$, 0.2, and 0.4. If the independence assumption is removed, is it possible for the ensemble error to be *worse* than ϵ?

18.24 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.

18.25 A simple perceptron cannot represent xor (or, generally, the parity function of its inputs). Describe what happens to the weights of a four-input, hard-threshold perceptron, beginning with all weights set to 0.1, as examples of the parity function arrive.

18.26 [linear-separability-exercise] Recall from Chapter Section **??** that there are 2^{2^n} distinct Boolean functions of n inputs. How many of these are representable by a threshold perceptron?

18.27 Consider the following set of examples, each with six inputs and one target output:

$\mathbf{x}_1$	1	1	1	1	1
$\mathbf{x}_2$	0	0	0	1	1
$\mathbf{x}_3$	1	1	1	0	1
$\mathbf{x}_4$	0	1	0	0	1
$\mathbf{x}_5$	0	0	1	1	0
$\mathbf{x}_6$	0	0	0	1	0
T	1	1	1	1	1

1. Run the perceptron learning rule on these data and show the final weights.

2. Run the decision tree learning rule, and show the resulting decision tree.

3. Comment on your results.

18.28 [perceptron-ML-gradient-exercise] Section Section **??** (page Section **??**) noted that the output of the logistic function could be interpreted as a *probability* p assigned by the model to the proposition that $f(\mathbf{x})=1$; the probability that $f(\mathbf{x})=0$ is therefore $1-p$. Write down the probability p as a function of $\mathbf{x}$ and calculate the derivative of $\log p$ with respect to each weight w_i. Repeat the process for $\log(1-p)$. These calculations give a learning rule for minimizing the negative-log-likelihood loss function for a probabilistic hypothesis. Comment on any resemblance to other learning rules in the chapter.

18.29 [linear-nn-exercise]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.

1. 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.

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

3. 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 $h \ll n$.

18.30 Implement a data structure for layered, feed-forward neural networks, remembering to provide the information needed for both forward evaluation and backward propagation. Using this data structure, write a function NEURAL-NETWORK-OUTPUT that takes an example and a network and computes the appropriate output values.

18.31 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.)

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

18.33 [embedding-separability-exercise] 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 or where they are located (unless the points are in the same place).

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

2. Show that it cannot always be done for $N=4$ points on a plane of dimension $d=2$.

3. Show that it can always be done for $N=4$ points in a space of dimension $d=3$, unless they are coplanar.

4. Show that it cannot always be done for $N=5$ points in a space of dimension $d=3$.

5. The ambitious student may wish to prove that N points in general position (but not $N+1$) are linearly separable in a space of dimension $N-1$.