

COS 402: Artificial Intelligence

Written exercises W6
Machine learning

Fall 2013
Due: Wednesday, January 8

Approximate point values are given in brackets. Be sure to show your work and justify all of your answers. See the course home page for information on when and where to submit written exercises, and grading criteria.

1. (15) Consider the following dataset:

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	y
1	1	0	0	0	1	0	1	1
1	0	0	0	0	0	0	0	0
0	0	0	1	1	1	1	0	0
0	1	0	0	1	0	0	0	0
1	0	0	0	0	1	0	1	0
0	1	1	0	1	1	0	1	1
1	1	0	1	0	1	0	1	1
0	1	0	1	0	1	0	0	1
0	0	0	0	0	0	1	1	0

In this formulation, there are eight attributes (or features or dimensions), x_1, \dots, x_8 , each taking the values 0 or 1. The label (or class) is given in the last column denoted y ; it also takes the values 0 or 1. Notice that the label y is 1 if and only if x_2 and x_6 are both equal to 1. Since attributes and labels are $\{0, 1\}$ -valued, we can write this rule succinctly as $y = x_2x_6$. In general, such a product of any number of attributes is called a *monomial*. (This includes the “empty” monomial, which, being a product of no variables, is always equal to 1.)

Throughout this problem, you can assume that the attributes and labels are all $\{0, 1\}$ -valued. Also, let n be the number of attributes (for instance, $n = 8$ in the example above). Assume, as usual, that the training and test examples are generated independently at random according to the same distribution.

- Describe a simple algorithm that, given a dataset, will efficiently (in polynomial time) find a monomial consistent with it, assuming that one exists.
- What is the total number of monomials that can be defined on n attributes?
- Suppose you applied your algorithm to the dataset above, and that a consistent monomial was found. Use the bound derived in class (or the results in R&N) to compute an upper bound on the generalization error of this monomial. (“Generalization error” is the same as what R&N calls simply the “error” or “error rate” in Section 18.5.) Derive a bound that holds with 95% confidence (so that $\delta = 0.05$).
- Continuing the last question in which your algorithm is applied to data with $n = 8$ attributes, how many training examples would be needed to be sure the generalization error of a consistent monomial is at most 10% with 95% confidence?

2. (32) Consider the following dataset consisting of five training examples followed by three test examples:

x_1	x_2	x_3	y
<i>training</i>			
−	+	+	−
+	+	+	+
−	+	−	+
−	−	+	−
+	+	−	+
<i>test</i>			
+	−	−	?
−	−	−	?
+	−	+	?

There are three attributes (or features or dimensions), x_1 , x_2 and x_3 , taking the values + and −. The label (or class) is given in the last column denoted y ; it also takes the two values + and −.

Simulate each of the following four learning algorithms on this dataset. In each case, show the final hypothesis that is induced, and show how it was computed. Also, say what its prediction would be on the three test examples.

For parts b and c, be sure to see the errata for R&N Chapter 18 below.

- The *decision tree algorithm* discussed in class and R&N. For this algorithm, use the information gain (entropy) impurity measure as a criterion for choosing an attribute to split on. Grow your tree until all nodes are pure, but do not attempt to prune the tree.
- AdaBoost*. For this algorithm, you should interpret label values of + and − as the real numbers +1 and −1. Use decision stumps as weak hypotheses, and assume that the weak learner always computes the decision stump with minimum error on the training set weighted by D_t . (Recall that a decision stump is a one-level decision tree; see R&N p. 750.) Run your boosting algorithm for three rounds.
- Support vector machines*. For this algorithm, you should interpret both label and attribute values of + and − as the real numbers +1 and −1. For this problem you should use a simplified version of the SVM where the separating hyperplane is restricted to pass through the origin (i.e. the bias b is set to zero). See the errata below for an explanation how this simplified SVM differs from the one we did in class (and the one in R&N). The errata also corrects some typos in R&N. You can use the additional information that the first three examples are support vectors, but the others are not, so that α_4 and α_5 are both zero in R&N Eq. (18.13). This means that you can maximize this equation over α_1 , α_2 and α_3 using calculus. (Note that if any of these variables turn out to be negative, there's a problem.) When you have found a solution vector \mathbf{w} , check it by showing that, for each labeled training example (\mathbf{x}, y) , we have $y(\mathbf{w} \cdot \mathbf{x}) \geq 1$, and that equality holds for the support vectors, i.e., the first three examples. (The notation here is as in class and R&N.) You do not need to use a “kernel,” just a regular inner product, as in Eqs. (18.13) and (18.14).
- Neural networks*. For this algorithm, use a single-layer neural net consisting of just a single perceptron at the output, no hidden layers, and the three features at the input level. Attribute values of + and − should be interpreted as the real numbers +1 and −1, while label values of + and − should be interpreted as 1 and 0. You can disregard the “bias weight” (denoted

$w_{0,j}$ in R&N Figure 18.19), i.e., assume it is fixed to be zero. Assume that the neural net is trained for a single “epoch” that runs through the training data once in the order given. Use a learning rate of $\alpha = 0.1$, and start with all weights equal to zero. For g , use the standard sigmoid function given in Figure 18.17(b).

Errata for R&N Chapter 18

There are a few important errors in R&N.

First of all, in Figure 18.34, the second to last line is written ambiguously. It should read:

$$\mathbf{z}[k] \leftarrow \log[(1 - \text{error})/\text{error}].$$

(Actually, however, I would encourage you to use the pseudocode and notation for AdaBoost given in class — see also Algorithm 1.1 of the optional reading provided on the “Schedule & Readings” webpage.)

Also, the paragraph describing SVM’s at the very bottom of page 745 continuing at the top of 746 contains a few typos.

First, the equation for \mathbf{w} should instead be: $\mathbf{w} = \sum_j \alpha_j y_j \mathbf{x}_j$. Also, Eq. (18.14) should instead read:

$$h(\mathbf{x}) = \text{sign} \left(\sum_i \alpha_i y_i (\mathbf{x} \cdot \mathbf{x}_i) + b \right).$$

Finally, note that R&N does not explain how to find b . We have also skipped over it in class.

For this homework you should use a simplified version of the SVM that does not have a bias parameter b . That is, the hypothesis is of the form

$$\text{sign}(\mathbf{w} \cdot \mathbf{x}).$$

rather than the form

$$\text{sign}(\mathbf{w} \cdot \mathbf{x} + b),$$

that was discussed and in R&N. The absence of the parameter b , implies that the separating hyper-plane must pass through the origin.

The through-the-origin case differs from what appears in R&N as follows: (1) the constraint $\sum_i \alpha_i y_i = 0$ is omitted; (2) the equation for \mathbf{w} needs to be corrected (as just explained); and (3) b is set to zero in Eq. (18.14).

For this assignment (including part 2c above), you should only consider the through-the-origin case.