CS/DSC/AI 391L: Machine Learning

Homework 1 - Theory

Patrick Brown

Lecture: Prof. Adam Klivans
Keywords: Boolean functions, mistake bounds, PAC learning

Instructions

Please either typeset your answers (LaTeX recommended) or write them very clearly and legibly and scan them, and upload the PDF on edX. Legibility and clarity are critical for fair grading.

Problem 1

Often in binary classification we are interested in the differences in the output of our current classifier, g, and an unknown function f that we are trying to learn. It is common in these cases to examine the quantity produced by $f(x)g(x)$ for a given input x. For this problem, let D be an arbitrary distribution on the domain $\{-1,1\}^n$, and let $f, g : \{-1,1\}^n \to \{-1,1\}$ be two Boolean functions.

Part (a) [6 points]

Prove that

$$\mathbb{P}_{x\sim D}[f(x) \neq g(x)] = \frac{1 - \mathbb{E}_{x\sim D}[f(x)g(x)]}{2}.$$

Proof:

Let's consider the event $f(x) \neq g(x)$. This occurs when either $f(x) = 1$ and $g(x) = -1$, or $f(x) = -1$ and $g(x) = 1$. In both cases, $f(x)g(x) = -1$.

Conversely, when $f(x) = g(x)$, we have $f(x)g(x) = 1$.

Therefore, we can write:

$$\begin{aligned}\mathbb{E}_{x\sim D}[f(x)g(x)] &= 1 \cdot \mathbb{P}[f(x) = g(x)] + (-1) \cdot \mathbb{P}[f(x) \neq g(x)] \\ &= \mathbb{P}[f(x) = g(x)] - \mathbb{P}[f(x) \neq g(x)]\end{aligned}$$

Since $\mathbb{P}[f(x) = g(x)] + \mathbb{P}[f(x) \neq g(x)] = 1$, we can substitute $\mathbb{P}[f(x) = g(x)] = 1 - \mathbb{P}[f(x) \neq g(x)]$:

$$\begin{aligned}\mathbb{E}_{x\sim D}[f(x)g(x)] &= (1 - \mathbb{P}[f(x) \neq g(x)]) - \mathbb{P}[f(x) \neq g(x)] \\ &= 1 - 2\mathbb{P}[f(x) \neq g(x)]\end{aligned}$$

Rearranging this equation:

$$\mathbb{P}[f(x) \neq g(x)] = \frac{1 - \mathbb{E}_{x\sim D}[f(x)g(x)]}{2}$$

Which is what we wanted to prove.

Part (b) [4 points]

Would this still be true if the domain were some other domain (such as $\mathbb{R}^n$, where $\mathbb{R}$ denotes the real numbers, with say the Gaussian distribution) instead of $\{-1,1\}^n$? If yes, justify your answer. If not, give a counterexample.

Note: Only the domain changes here. The output is still boolean.

Answer:

No, this would not necessarily be true for other domains such as $\mathbb{R}^n$.

Counterexample:

Consider $f, g : \mathbb{R} \to \{-1, 1\}$ defined as:

$$f(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ -1 & \text{if } x < 0 \end{cases}$$

$$g(x) = \begin{cases} 1 & \text{if } x > 0 \\ -1 & \text{if } x \leq 0 \end{cases}$$

Let D be the standard normal distribution $N(0, 1)$.

In this case, $f(x) \neq g(x)$ only when $x = 0$, which has probability 0 under the normal distribution. So $\mathbb{P}_{x\sim D}[f(x) \neq g(x)] = 0$.

However, $\mathbb{E}_{x\sim D}[f(x)g(x)] < 1$ because there's a non-zero probability that $x < 0$ or $x > 0$.

Therefore, $\frac{1-\mathbb{E}_{x\sim D}[f(x)g(x)]}{2} > 0 = \mathbb{P}_{x\sim D}[f(x) \neq g(x)]$.

This counterexample shows that the equality doesn't hold for all domains and distributions.

Problem 2 [10 points]

Let f be a decision tree with t leaves over the variables $x = (x_1, \ldots, x_n) \in \{-1, 1\}^n$. We can write f as a multivariate polynomial $p(x_1, \ldots, x_n)$ such that for every input $x \in \{-1, 1\}^n$, $f(x) = p(x)$. Here's how:

1. For each leaf in the decision tree, create an "indicator polynomial":
   - Start with 1.
   - For each decision node on the path from the root to the leaf:
     - If the path goes left (xi = -1), multiply by $\frac{1-x_i}{2}$.
     - If the path goes right (xi = 1), multiply by $\frac{1+x_i}{2}$.
   - Multiply the result by the leaf's value (1 or -1).
2. Sum all these indicator polynomials to get $p(x_1, \ldots, x_n)$.

This polynomial $p(x_1, \ldots, x_n)$ will:

- Evaluate to 1 for paths leading to TRUE leaves.
- Evaluate to -1 for paths leading to FALSE leaves.
- Evaluate to 0 for all other paths.

The resulting polynomial $p(x_1, \ldots, x_n)$ will be equivalent to the decision tree f for all inputs $x \in \{-1, 1\}^n$.

Problem 3 [10 points]

Compute a depth-two decision tree for the training data in table 1 using the Gini function, $C(a) = 2a(1 - a)$ as described in class.

Solution:

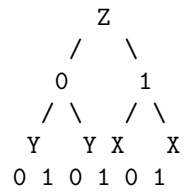
We compute the Gini impurity for each feature (X, Y, Z) to determine the root node:

- Weighted Gini for X: 0.444
- Weighted Gini for Y: 0.445
- Weighted Gini for Z: 0.400

Z provides the best split, so it becomes the root node. For the second level:

- When Z=0, the best split is on Y
- When Z=1, the best split is on X

The resulting tree structure:



Accuracy Calculation:

- Leaf 1 (Z=0, Y=0): Predict negative, 45/60 correct
- Leaf 2 (Z=0, Y=1): Predict positive, 45/60 correct
- Leaf 3 (Z=1, X=0): Predict positive, 60/90 correct
- Leaf 4 (Z=1, X=1): Predict positive, 45/40 correct

Total correct predictions: 45 + 45 + 60 + 45 = 195 Total examples: 250

Overall Accuracy:

The overall accuracy of the depth-two decision tree on the training data is:

$$\frac{195}{250} = 0.78 \text{ or } 78\%$$

Problem 4 [10 points]

PAC Learning Algorithm for Threshold Functions

Let's define a simple algorithm to learn the threshold function:

1. Request $m = O(\frac{1}{\epsilon} \log \frac{1}{\delta})$ training examples.
2. Sort the examples in ascending order based on their x-values.
3. Find the largest x-value with label -1, call it x_-.
4. Find the smallest x-value with label 1, call it x_+.
5. Set the threshold $\hat{\theta} = \frac{x_- + x_+}{2}$.
6. Return the hypothesis $h_{\hat{\theta}}(x)$.

Justification

This algorithm is simple and efficient:

- Sorting takes $O(m \log m)$ time, which is $O(\frac{1}{\epsilon} \log \frac{1}{\epsilon} \log \frac{1}{\delta})$.
- Finding x_- and x_+ takes linear time $O(m)$.
- The algorithm uses only $m = O(\frac{1}{\epsilon} \log \frac{1}{\delta})$ examples as required.

The algorithm will output a classifier with error at most ϵ with probability at least $1 - \delta$ because:

- With high probability, the gap between x_- and x_+ will be at most ϵ in the underlying distribution.
- Any threshold in this gap will have error at most ϵ.
- The Chernoff bound guarantees that $m = O(\frac{1}{\epsilon} \log \frac{1}{\delta})$ samples are sufficient to achieve this with probability at least $1 - \delta$.

Therefore, this algorithm satisfies the PAC learning requirements for the given concept class C.

Problem 5 [6 points]

In this problem, we show that the existence of an efficient mistake-bounded learner for a class $\mathcal{C}$ implies an efficient PAC learner for $\mathcal{C}$.

(a) Lower bound on k

To determine a lower bound on k such that $\mathbb{P}[\text{err}(h') > \epsilon] \leq \delta'$, we use Hoeffding's inequality. For a fixed hypothesis h', we want:

$$\mathbb{P}[|\text{err}(h') - \widehat{\text{err}}(h')| > \epsilon] \leq \delta'$$

Where $\widehat{\text{err}}(h')$ is the empirical error on k samples. Using Hoeffding's inequality:

$$\mathbb{P}[|\text{err}(h') - \widehat{\text{err}}(h')| > \epsilon] \leq 2e^{-2k\epsilon^2}$$

Setting this less than or equal to δ' and solving for k:

$$k \geq \frac{1}{2\epsilon^2} \ln\left(\frac{2}{\delta'}\right)$$

This gives us the lower bound on k.

(b) Bounding all possible hypotheses

To bound all possible hypotheses, we need to account for the number of mistakes algorithm A can make. Since A has a mistake bound t, it can produce at most $t+1$ different hypotheses.

We use the union bound to ensure that the probability of any of these hypotheses having error $> \epsilon$ is at most δ. Set $\delta' = \frac{\delta}{t+1}$ in the previous bound:

$$k \geq \frac{1}{2\epsilon^2} \ln\left(\frac{2(t+1)}{\delta}\right)$$

Now, m should be large enough to contain t blocks of size k. So:

$$m \geq tk \geq \frac{t}{2\epsilon^2} \ln\left(\frac{2(t+1)}{\delta}\right)$$

(c) PAC learner description and proof

Here's a high-level pseudocode for algorithm B:

```
Algorithm B(epsilon, delta, t):
    k = ceil((1 / (2*epsilon^2)) * ln(2(t+1)/delta))
    m = t * k
    Draw m examples from D
    Initialize A
    For each example (x, y) in the m examples:
        Let h be A's current hypothesis
        If h(x) != y:
            Update A with (x, y)
    Return A's final hypothesis
```

Proof of PAC learning:

1. We've shown that $m \geq \frac{t}{2\epsilon^2} \ln\left(\frac{2(t+1)}{\delta}\right)$ examples are sufficient.
2. With probability at least $1-\delta$, all $t+1$ possible hypotheses produced by A will have true error at most ϵ if their empirical error on a block of k examples is 0.
3. A makes at most t mistakes, so it will produce at most $t+1$ hypotheses.
4. If A makes fewer than t mistakes, its final hypothesis will have seen at least k examples without making a mistake, ensuring its true error is at most ϵ with high probability.
5. Therefore, with probability at least $1-\delta$, B outputs a hypothesis with true error at most ϵ.

This satisfies the PAC learning requirements, with a finite lower bound on m that depends on ϵ, δ, and t.