Homework 2 TTIC 31250

Eddie Shim
(worked with Hanqi Zhang)
eddieshim@uchicago.edu

04/27/2020

Exercise 1

Part a

Show that for any constant $c \geq 0$, cK_1 is a legal kernel. That is, show how to define ϕ in terms of ϕ_1 such that $\phi(x) \cdot \phi(x') = cK_1(x, x')$.

Let $\phi_1(x) \rightarrow \mathbb{R}^N$ define the mapping for the kernel K. We have the following:

$$\begin{aligned} K_1(x, x') &= \phi_1(x) \cdot \phi_1(x') \\ \text{Let } \phi(x) &= \frac{\phi_1(x)}{\sqrt{c}} \\ cK_1(x, x') &= \sqrt{c}\phi(x) \cdot \sqrt{c}\phi(x') \end{aligned}$$

Thus, the mapping $\sqrt{c}\phi$ is a legal kernel.

Part b

Show that the sum, $K_1 + K_2$, is a legal kernel. That is show how to define ϕ in terms of ϕ_1 and ϕ_2 such that $\phi(x) \cdot \phi(x') = K_1(x, x') + K_2(x, x')$.

Let $\phi_1(x) \rightarrow \mathbb{R}^N$ define the mapping for the kernel K_1 and let $\phi_2(x) \rightarrow \mathbb{R}^N$ define the mapping for kernel K_2. We have the following:

$$\begin{aligned}
K_1(x, x') &= \phi_1(x) \cdot \phi_1(x') \\
K_2(x, x') &= \phi_2(x) \cdot \phi_2(x') \\
\text{Let } \phi(x) &= \langle \phi_1(x), \phi_2(x)) \rangle \\
\phi(x) \cdot \phi(x') &= K_1(x, x') + K_2(x, x')
\end{aligned}$$

Thus the mapping $\phi(x)$ is our desired legal kernel.

Part c

Show that the product, $K = K_1 K_2$, is a legal kernel. That is, show how to define ϕ in terms of ϕ_1 and ϕ_2 such that $\phi(x) \cdot \phi(x') = K_1(x, x')K_2(x, x')$.

Let $\phi_1(x) \to \mathbb{R}_1^N$ define the mapping for the kernel K_1 and let $\phi_2(x) \to \mathbb{R}_2^N$ define the mapping for kernel K_2. We have the following:

$$\begin{aligned}
\text{Let } K(x, x') &= \sum_i \phi_i(x)\phi_i(x') \\
\text{Thus: } K^{(1)}(x, x')K^{(2)}(x, x') &= \sum_{N_1} \phi_{N_1}^{(1)}(x)\phi_{N_1}^{(1)}(x') * \sum_{N_2} \phi_{N_2}^{(2)}(x)\phi_{N_2}^{(2)}(x') \\
&= \sum_{N_1, N_2} \phi_{N_1}^{(1)}(x)\phi_{N_2}^{(2)}(x)\phi_{N_1}^{(1)}(x')\phi_{N_2}^{(2)}(x') \\
\text{Now define: } \phi_{N_1 N_2}(z) &= \phi_{N_1}^{(1)}(z)\phi_{N_2}^{(2)}(z) \\
\text{Thus: } K^{(1)}(x, x')K^{(2)}(x, x') &= \sum_{N_1 N_2} \phi_{N_1 N_2}(x)\phi_{N_1 N_2}(x')
\end{aligned}$$

Thus the mapping $\phi_{N_1 N_2}(z)$ is our desired legal kernel.

Problem 2

Assuming that at least one of the N hypotheses has error at most $\epsilon/2$, give an explicit bound (without O notation) on a size for the test set that is sufficient so that with probability at least 1-$\delta/2$, the hypothesis that performs best on the test set has error at most ϵ. Prove that we can convert algorithm A to algorithm

B. Define algorithm A with at least 1/2 chance of producing a hypothesis of error at most $\epsilon/2$ when given m labeled examples drawn from distribution D and labeled by a target function $f \in C$. Define algorithm B as having at least 1-δ probability of producing a hypothesis of error at most ϵ.

As given by the problem statement, we first run A on $N = lg(2/\delta)$ different sets of m random examples. This guarantees that with probability at least 1- $\delta/2$, at least one of the N hypotheses produced has error at most $\epsilon/2$. Then we test the N hypotheses produced on a new test set, choosing the one that performs best.

Define T as the test set, $err_{D_T}(h)$ as the error on the test set, and h_T^* as the optimal hypothesis in T. Let's take the optimal hypothesis h from the training set, namely the one that has $err_{D_{train}}(h) < \epsilon/2$.

The Chernoff bounds states:

$$Pr[X/m > p(1+\alpha)] \leq e^{-mp\alpha^2/3}$$

Applying the Chernoff and union bound to our optimal train hypothesis h:

$$1-(\delta/4) = 1 - Pr[err_T(h) > (\epsilon/2)(1+\frac{1}{2})] \geq 1 - |H|e^{-|S|\frac{\epsilon}{2}\frac{1}{2}^2/3}$$

$$1-(\delta/4) = Pr[err_T(h) \leq \frac{3}{4}\epsilon] \geq 1 - |H|e^{-|S|\frac{\epsilon}{24}}$$

(Property 1) Therefore, if $|S| \geq \frac{24}{3}(ln|H| + ln(\frac{4}{\delta}))$, then with prob $\geq 1-(\delta/4), err_T(h) \leq \frac{3}{4}\epsilon$

For all other hypothesis h' in the training set that are not the optimal hypothesis h with $err_{D_{train}}(h) > \epsilon$, we can apply the Chernoff and union bound again:

$$1-(\delta/4) = 1 - Pr[err_T(h') \leq \epsilon(1-\frac{1}{4})] \geq 1 - |H|e^{-|S|\frac{\epsilon}{2}\frac{1}{4}^2/2}$$

(Property 2)Therefore, if $|S| \geq \frac{64}{\epsilon}[ln|H| + ln(\frac{4}{\delta})]$, then with prob $\geq 1-(\delta/4)$, $err_T(h') > \frac{3}{4}\epsilon$

Thus we continue by setting $|S| \geq \frac{64}{\epsilon}[ln|H| + ln(\frac{4}{\delta})]$ From (Property 1) and the theorem described on Lecture 5 slide 10, we know that with prob $\geq 1-(\delta/4), err_T(h_T^*) \leq \frac{3}{4}\epsilon$ because h has

$$err_T(h) \leq \frac{3}{4}\epsilon \implies err_T(h_T^*) \leq \frac{3}{4}\epsilon$$

because h_T^* is the best performing hypothesis in T.
From (Property 2) we know that every hypothesis h' with $err_{D_{train}}(h') > \epsilon$, with prob $\geq 1-(\delta/4)$, so

$$err_T(h') > 3/4 \implies \text{with prob } \geq 1-(\delta/4), err_D(h_T^*) < \epsilon$$

Therefore, we arrive at the wanted property:
When $|S| \geq \frac{64}{\epsilon}(ln(|H| + ln(\frac{4}{\delta}))$, with prob $\geq (1-\delta/4)(1-\delta/4) \geq 1-(\delta/2), err_D(h_T^*) < \epsilon$

Problem 3

Prove that in any contiguous block of trials (e.g. the 51st example through the 77th example), the number of mistakes made by the algorithm is at most $O(m + logn)$, where m is the number of mistakes made by the best expert in that block, and n is the total number of experts.

Define W_i as the total weight in our current expert system at iteration i. We know that each iteration can only equal or lower the prior round's total weight W_{i-1}, so the metric is monotonically decreasing. From the definition of step (c), notice that an expert's weight must be at least 1/8 of the average. This is because an expert's weight can only be lowered (by halving) if its weight was at least 1/4 of the average weight of experts. Therefore, at each iteration i we can calculate that the weight of the best expert is at least:

$$w_{best_i} \geq (\frac{1}{2})^m \frac{1}{8} avg(W_{i-1})$$
$$= (\frac{1}{2})^m \frac{W_{i-1}}{8n}$$

We know that on each iteration i, we are removing at most $W_{i-1}/2$, in the worst case instance that all experts are wrong. We also know that in each iteration i, we cannot reduce more than $W_{i-1}/4$ due to the condition we can only reduce experts which have at least 1/4 of the average weight of experts. Thus we are removing at least $(W/2 - W/4)$ / $2 = W/8$ each iteration so the most total weight available at iteration i is $W_{i-1}(7/8)^M$. Thus we can set up the same inequality that was shown in lecture on how to prove Weighted Majority Algorithm's bound on M:

Weight of best expert at iteration $i \leq$ Most total weight available in system at iteration i

$$(1/2)^m \frac{W_{i-1}}{8n} \leq W_{i-1}(7/8)^M$$

$$(1/2)^m \leq 8n(7/8)^M$$

$$(8/7)^M \leq 8n2^m$$

$$Mlg(8/7) \leq m + lg(8n)$$

$$M \leq 5.2(m + lg(8n))$$

Problem 4

Prove that the Perceptron algorithm may make $2^{\Omega(n)}$ mistakes in learning a decision list over n Boolean variables. To prove this, give a set S of labeled examples in $0, 1^n$ that (a) is consistent with some decision list L and yet (b) has the property that any LTF with integer weights that has zero error on S must have one weight that is exponentially large. Prove that your set S has properties (a) and (b).

From homework 1, we proved that any decision list can be converted into a linear threshold function by recursively iterating through each if-then statement backwards, and cumulatively adding x_i terms multiplied by a coefficient that scales with either summed all prior positive weights or summed the absolute value of all prior negative weights. Thus to show that we can produce a decision list that's consistent and have a weight that is exponentially large, my approach will be to adversarially create a set S in which the last weight must sum over prior terms' coefficients that scale with the size of the data. Take for instance a dataset of even size n points where we have features $X_1, X_2, ...X_n$. For each odd indexed datapoint, let its label be 0 and for each even indexed datapoint let its label data be 1. Next, mark the feature column $X_i = 1$ in the same ith row of the datapoint. This gives us a matrix of nxn size with 1s on the diagonal going from top left to bottom right. Next, for each row start on the column which is marked 1 on that row, then iterate backwards to fill in the columns $i, i-1, i-3, i-5, ..., 1$ if i is even, and $i, i-1, i-3, ..., 2$ if i is odd. Here's an example when $n = 4$:

$$S = \left[\begin{array}{cccc|c} 1 & 0 & 0 & 0 & - \\ 1 & 1 & 0 & 0 & + \\ 0 & 1 & 1 & 0 & - \\ 1 & 0 & 1 & 1 & + \end{array}\right]$$

We can guarantee (a) because we can use the way the diagonal 1s are structured, we can simply use a decision list that for any given row, if the value in $(i, i) == 1$, then + if i is even or - if i is odd. Using the algorithm described in homework 1, this S guarantees that as we recurse through our decision list backwards, the weight coefficient must exponentially grow because it has to sum all the alternating past coefficients of the opposite sign. Thus, for any even row we have:

$$|w_i| > |w_{i-1}| + |w_{i-3}| + ... + |w_1|$$

and for any odd row we have:

$$|w_i| > |w_{i-1}| + |w_{i-3}| + ... + |w_2|$$

Because the perceptron algorithms on a binary classifier can update a weight value by 1 at most each step and we know that calibrating w_i will result in an exponentially large coefficient, this shows that we have satisified condition (b).