Solutions for Homework 1

Bao Jinge A0214306U e0522065@u.nus.edu

1 VC-dimension of Decision Trees

1.1 a

From SSBD 18.1, we can think of a decision tree as a splitting of the instance space. Since a binary tree with height of d has at most 2^d leaves, a decision tree with depth of d also has at most 2^d leaves, namely 2^d root-to-leaf pathes. Because of every leaf of a decision tree can be seen as a result of classifying (label), every path from root to leaf can be seen as classifying a instance according to d attributes in each level.
Thus, we could always build such dataset which has d features and every feauture has a value 0 or 1. And every instance will be label with 0 or 1 randomly. There are at most 2^d kinds of instance. Obviously, every kind of instance from this dataset can be classified by a unique root-to-leaf path in this decision tree, which meets the definition of VC-dimension.
Accouding to the definition of VC-dimension, the d-height decision tree can shatter a set of 2^d instances, so the VC-dimension of decision tree of height d is at least 2^d.

1.2 b

From SSBD 18.1, we could know a tree with n nodes can be described in $n+1$ blocks, each of size $\log_2(d+3)$ bits. Here the first n blocks encode the nodes of the tree in pre-order, and the last block marks the end of the code. Here each blocks indicates whether the current node is:

1. An internal nodes of the form $\mathbb{1}[x_i = 1]$ for some $i \in [d]$
2. A leaf whose value is 1
3. A leaf whose value is 0
4. End of the code

There are $d + 3$ opinions, so we need $\log_2(d + 3)$ bits to describe each block. According to the requirements of the question (b), if the decision tree is a binary tree that maps from $\{0, 1\}^d$ to $\{0, 1\}$, than the description length of this tree is $(n + 1) \log_2(d + 3)$, which means that the binary decision tree will have at most $2^{(n+1)log_2(d+3)}$ leaves. Thus, VC-dimension of such binary tree is at most $(n + 1) \log_2(d + 3)$.

1.3 c

Because every binary decision tree is a full binary tree, if binary decision tree has n leaves, it will have $2n-1$ nodes(both internal nodes and external nodes). From the conclusion of (b), we know for each decision tree, its VC-dimension at most $2n \log_2(d + 3)$. Because we could consider a random

forest as a threshold weighted average of the k decision trees, so we could regard results of k decision trees as a k-dimension vectors and regard threhold weighted average as a linear threshold classifier from $\{0,1\}^k$ to $\{0,1\}$. Because the VC-dimension of a linear threshold classifier on k dimension space is $k+1$. According to Sauer's Lemma, we have

$$\tau_{ltc}(m) \leq \sum_{i=0}^{k+1} \binom{m}{i}$$

which m is size of set and ltc denotes linear threshold classifier. Because of every decision tree is also independent, so we have

$$\tau_{\mathcal{H}}(m) \leq \prod_{i=1}^{k} 2^{2n \log_2 (d+3)} \sum_{i=0}^{k+1} \binom{m}{i}$$

Let

$$\begin{aligned} 2^m &\leq \prod_{i=1}^{k} 2^{2n \log_2 (d+3)} \sum_{i=0}^{k+1} \binom{m}{i} \\ &\leq 2^{2kn \log_2 (d+3)} (\frac{em}{k+1})^{k+1} \end{aligned}$$

where when $m > d+1$. Thus, we get

$$\begin{aligned} m &\leq 2nk \log_2 (d+3) + (k+1) log_2(em) - (k+1) log_2(k+1) \\ &\leq O(nklog_2 d + klog_2 k) \end{aligned}$$

From the definition of VC-dimension, such random forest has upper bound of VC-dimension as $O(nklog_2 d + klog_2 k)$

2 Kernels

2.1 a

We could find $\phi(x)$ like as follows

$$\psi(x) = (1,1,1,1,1,0,0,0,...)^T$$

where there are n coordinates. The first x coordinates are 1's and other coordinates are 0's. Obviously, we can rewrite kernel as follows

$$\forall x, x' \in \{1,...,N\} K(x,x') =< \psi(x), \psi(x') >= min(x,x')$$

We could see that the answer of inner product is determined by the smaller number of 1's between both N-dimensional vectors. Consequently, K is a valid kernel.

2.2 b

2.2.1 i

Since the parameter of first representation depends on the size of training dataset m, the first representation, namely

$$w = \sum_{i=1}^{m} \alpha_i \psi(x_i)$$

will be smaller when m is much smaller than N.

2.2.2 ii

Since the parameter of second representation depends on the size of feature vector N, the second representation, namely a vector of real numbers, will be smaller when N is much smaller than m,.

2.3 c

2.3.1 i

Suppose we can represent this function, we will get system of equations as follows

$$\begin{cases} f(1) = \alpha_1 K(1,1) + \alpha_2 K(3,1) = \alpha_1 + \alpha_2 = 0 \\ f(2) = \alpha_1 K(1,1) + \alpha_2 K(3,2) = \alpha_1 + 2\alpha_2 = 0 \\ f(3) = \alpha_1 K(1,1) + \alpha_2 K(3,3) = \alpha_1 + 3\alpha_2 = 1 \end{cases}$$

Differencing between the first two equaitons, we get $\alpha_3 = 0$. Differencing between the last two equations, we get $\alpha_3 = 1$, contracting the former result. Thus we can not represent such function.

2.3.2 ii

Similarly, suppose we represent this function, we will get system of equations as follows

$$\begin{cases} f(1) = \alpha_1 K(2,1) + \alpha_2 K(3,1) = \alpha_1 + \alpha_2 = 0 \\ f(2) = \alpha_1 K(2,1) + \alpha_2 K(3,2) = 2\alpha_1 + 2\alpha_2 = 0 \\ f(3) = \alpha_1 K(2,1) + \alpha_2 K(3,3) = 2\alpha_1 + 3\alpha_2 = 1 \end{cases}$$

From system of equations as above, we can get the values of α_2 and α_3

$$\begin{cases} \alpha_1 = -1 \\ \alpha_2 = 1 \end{cases}$$

This implies that such function can be represented by $\alpha_1 K(2,x) + \alpha_2 K(3,x)$ with $\alpha_1 = -1$ and $\alpha_2 = 1$.

2.4 d

First, we could count the times each word occuring in a file x_i, and encoding it's into a vector v^i

$$v^i = (v_1^i, v_2^i, ...v_M^i)^T.$$

Suppose there are M files. In this vector v^i, there are M entries, which denotes the times each word occuring in file x_i. Obviously,

$$K_s(x_i, x_j) = K_s^{'}(v^i, v^j) = \sum_{k=1}^{M} K_{min}(v_k^i, v_k^j)$$

where K_s and $K_s^{'}$ denotes the similarity function, and K_{min} denotes the kernel function given by problem, i.e. $K(x, x') = min(x, x')$. Since kernel K_s can be denoted by sum of several valid kernel, we can be sure that K_s is also a valid kernel.

3 Learn to Forex Trading

3.1 a

Suppose there are C kinds of currency. If we know the exchange rate for each of the D days into the future r_{ij}^d, we have following equation

$$\begin{cases} V(j, d-1) & = \max\limits_{k \in C} \{r_{jk}^{d-1} V(k, d)\} \\ V(j, D) & = r_{j0}^D \end{cases}$$

where $0 \leq j < C$ and $d \in \{2, ..., D\}$
We design algorithms as **Algorithm 1**:

3.2 b

3.2.1 i

Because Value Iterative Networks can be designed as Convolutional Nerual Networks, we design a CNNs as follows.
Input: Let $x = (V(0, D), V(1, D), ...V(C-1, D))^T$ as input.
Input Layer: Accept an input $C * 1 * 1$ tensor(vector) x, and output a tensor $h^{(0)} = x$
Hidden layers: There will be total $D-1$ convlution layers and maxpooling layers stacked. To illustrate clearly, we rank convolution Layers from 1 to $D-1$. All convolution layers and maxpooling layers will have same structure, thus we just clarify the first convolution layer and maxpooling layer as an example.

Convolution Layer 1: Accept the input $C * 1 * 1$ tensor(vector). Use C kinds of a $|C| * 1 * 1$ Filters. Each filter has parameters $\theta_k^{(1)} = (r_{k,0}^D, r_{k,1}^D, ..., r_{k,C-1}^D)^T$, where $0 \leq k < C$, which are also weights of this CNNs . Then we will get C features. Each feature is the inner product of $F_k =< h^{(0)}, \theta_k^{(1)} >$. In this layer, we don't use activation function. In other way, we use activation function like $g(x) = x$

Algorithm 1 Dynamic Programming Algorithm for Computing $V(0,1)$

Input: exchange rate r_{jk}, which $j,k \in [C]$
Output: the maximun amount of currency 0 V(0,1)

```
for all j ∈ C and d ∈ {1, ..., D − 1} do
    initialize V(j, d) := −∞.
end for
for j ∈ C do
    set V(j, D) := r^D_{j0}
end for
for d = D; d ≥ 2; d − − do
    for j = 0; j < C; j + + do
        for k = 0; k < C; k + + do
            V(j, d − 1) = max{V(j, d − 1), r^d_{jk} V(k, D)}
        end for
    end for
end for
return V(0, 1)
```

Maxpooling Layer 1: Accept input C features as $C \times 1 * C$ tensor F, we use operation maxpooling on every $C \times 1 \times 1$ feature and get maximun. $h_k^{(1)} = \max_{0 \le j < C} F_{kj}$, where F_{kj} denotes the j-th coordiante of the k-th feature vector. Obviously, we can get $h^{(1)} = (V(0, D-1), V(1, D-1), ..., V(C, D-1)^T$, which will be output.

Convolution Layer 2:...

Maxpooling Layer 2:...

...(Other $D-1$ convolution layers and maxpooling layers are like above)

Convolution Layer D-1:...

Maxpooling Layer D-1:...

After $D-1$ convolution layers and maxpooling layers, we get $h^{(D-1)} = (V(0,1), V(1,1), ..., V(C,1))^T$.

Output Layer: This layer accept $h^{(D-1)}$ as input, and output the first coordinate of input as output, which is also the output of this CNNs, i.e. $V(0,1)$.

3.2.2 ii

If we have information about exchange rates and feature vector x in last M days, we could built training dateset consisting of $M - D + 1$ training examples. For each training example $(x^{(i)}, y^{(i)})$, the feature vector $x^{(i)}$ can be got from observation everyday, whose label $y^{(i)}$ can be calculated using the dynamic programming algorithm in (a).

3.3 c

(1) Using supervised learning, we could use more historical information(M days) than dynamic programming algorithms(just D days) to help us get more better optimal solution. More historical information can help reduce the chance of incidents.
(2) Dynamic programming only take exchange rates into consideration, but many other factors will

influence exchange rates in real world. Actually, the weights of CNNs not only denotes just exchange rates, but also total influcence of many factors in real world. Although we can't get very precise exchange rates by weights, we may get better combination factors which have more significant influence on currency. Namely, we got more and deeper information behind exchange rates on currecy.