NATIONAL UNIVERSITY OF SINGAPORE

CS5339: Theory and Algorithms for Machine Learning

(Semester 2: AY2018/19)

Time Allowed: 120 Minutes

INSTRUCTIONS TO CANDIDATES

a) This paper consists of **THREE (3)** questions and **EIGHT (8)** printed pages including this page. Answer all questions.

b) You have 120 minutes to earn 40 marks. Do not spend too much time on any problem. Read them all through first and attack them in the order that allows you to make the most progress.

c) You may quote results that are stated in the lecture notes or homework solutions without re-deriving them if you need the results as part of your answer.

d) Show your work, as partial credit will be given. You will be graded on the correctness and optimality of your answers, and also on your clarity. Be clear and always explain your reasoning.

e) This is an **OPEN BOOK** assessment.

f) Please write your Student Number only. Do not write your name.

Student number:

For Examiner's Use Only		
	Max Marks	Earned Marks
Problem 1	18	
Problem 2	12	
Problem 3	10	
TOTAL:	40	

Problem 1. Short Questions (18 Marks)

a) Assume that the density model used for multi-output regression is $p(\boldsymbol{y}|f(x)) = \frac{1}{Z}\exp\left(-(\boldsymbol{y}-f(x))^T\boldsymbol{A}(\boldsymbol{y}-f(x))\right)$, where $\boldsymbol{y}$ is a d-dimension vector, $\mathbf{A}$ is a d by d positive definite matrix and Z is the normalizing constant. What is the corresponding loss function for empirical risk minimization such that minimizing the empirical risk corresponds to maximizing the likelihood?

Solution:

The negative log likelihood is $(\boldsymbol{y}-f(x))^T\boldsymbol{A}(\boldsymbol{y}-f(x)) - \log Z$. Since Z is a constant, and log is a monotonic function, minimizing $(\boldsymbol{y}-f(x))^T\boldsymbol{A}(\boldsymbol{y}-f(x))$ corresponds to maximizing the log likelihood.

b) Consider representing the Boolean function $f: \{0,1\}^d \rightarrow \{0,1\}$, where $f(x)$ outputs the value 1 if $\sum_{i=1}^{d} x_i > d-1$ and $f(x)$ the value 0 otherwise. Argue that the height of a decision tree representing f is at least d, where the height is the longest path from the root to a leaf.

Solution:
Consider the instance x' where all the variables in x' take the value 1. This instance will take a particular path to the leaf of the tree If the longest path to a leaf is less than d, the number of variables encountered along the path must be less than d. Let the variable that is not encountered be x_i. Consider the class label at the leaf. If the label is 0, then x' is misclassified and the tree cannot represent $f(x)$. If the class label is 1, set $x_i = 0$ in x'. This new instance will also be classified by the same leaf, hence will be misclassified. Hence, the tree cannot represent $f(x)$ if its height is less than d.

c) Consider the quadratic kernel $K(x,x') = (1+\langle x,x'\rangle)^2$ in $\mathbb{R}^2$. Let $f(x) = 2K\big((2,3),x\big) + 3K((1,2),x)$. Let w be the weight vector representing $f(x)$ in feature space. What is the value of $\|w\|$?

Solution:
The squared norm of the weight is $2\times 2\times K\big((2,3),(2,3)\big) + 3\times 3\times K\big((1,2),(1,2)\big) + 2\times 3\times K\big((2,3),(1,2)\big) + .3\times 2\times K\big((1,2),(2,3)\big) = 4\times 14^2 + 9\times 6^2 + 12\times 9^2 = 2080$. Hence the norm is $\sqrt{2080} = 45.61$.

d) **True or False.** Let $K(x,x') = (1+\langle x,x'\rangle)^2$ be the quadratic kernel in $\mathbb{R}^2$. Consider the class of function defined by thresholding $f(x) = \sum_{i=1}^{100}\alpha_i K(x_i,x)$ where x_i are fixed but α_i are allowed to vary for $i = 1,\ldots,100$. Then the VC-dimension of this class of functions is no more than 6. Justify your answer.

Solution:
True. The kernel expands out to only 6 features $\big(1,\sqrt{2}x_1,\sqrt{2}x_2,\sqrt{2}x_1x_2,x_1^2,x_2^2\big)$. As VC-dimension of linear functions with 6 parameters is no more than 6, the VC-dimension of the function class is no more than 6.

e) **True or False.** Consider using the following class for doing weak learning for classifying strings: $H = \{1_s(x): s \text{ is a substring of length } d\} \cup \{f(x) = 1\}$, where $f(x) = 1$ is a constant function taking the value 1, and $1_s(x)$ is an indicator function that takes the value 1 when string *s* is a substring in *x* and takes the value 0 otherwise. Assume that the target function *t* is a disjunction of *k* indicator functions $1_{s1}(x) \vee \ldots \vee 1_{sk}(x)$. Then there is a $O(1/k)$-weak learning algorithm that uses *H* for learning *t*. Justify your answer.

Solution:
True. We know that if there is a convex combination of a set of {-1,1}-valued functions with margin 2γ then there is a γ-weak learning algorithm that uses the set of functions. We first convert each of $1_{si}(x)$ into a set of {-1,1}-valued function $\bar{1}_{si}(x)$ by setting the output to -1 when a function produces 0. We now show that there is a convex combination of $\bar{1}_{si}(x), i = 1, \ldots, k$ and the bias with margin $O(1/k)$. We first construct a linear function before normalizing the weight to have a convex combination. Set the output weight of $\bar{1}_{si}(x)$ 1 and the bias to *k*-1. When the string *x* is labeled positive, one or more of $\bar{1}_{si}(x), i = 1, \ldots, k$ has output 1, while no more than *k*-1 has output -1. Adding the bias *k*-1, the value of the function is at least 1. When the string *x* is labeled negative all of $\bar{1}_{si}(x), i = 1, \ldots, k$ has output -1. Adding the bias *k*-1, the output of the function is -1. The sum of output weights and bias is 2*k*-1. Normalizing the margin is at least 1/(2*k*-1).

f) **True or False.** The function class *H* contains only a *single* Boolean function of *d* variables, namely the Parity function. The *Rademacher complexity* of *H* is at least 2^d as representing Parity using a decision tree requires a tree of height *d* and such decision trees have VC-dimension at least 2^d. Justify your answer.

Solution:
False. The Rademacher complexity of a class that contains a single function is 0 as described in lectures.

Problem 2. Recurrent Neural Networks (12 Marks)

Consider a recurrent neural network that takes in sequences $x_1, x_2, \ldots, x_n$, where $x_i \in \mathbb{R}$ as shown below.

Each vector of hidden units has dimension *k*. There are *k* by *k* weights forming a weight matrix $\boldsymbol{W}$ between $\boldsymbol{h}_i$ and $\boldsymbol{h}_{i+1}$ and a vector $\boldsymbol{U}$ of length *k* parameterizing the connections between x_i and $\boldsymbol{h}_i$. The *j*-th hidden variable at time *i* is defined by $h_{i,j} = \sigma\left(\sum_{l=1}^{k} W_{j,l} h_{i-1,l} + U_j x_i\right)$, where σ is an activation function. There is only one binary output *y* which has h_n as input and is parameterized by weight vector *w* of length *k*: $y = \sum_{j=1}^{k} w_j h_{n,j}$. For simplicity, none of the units or output has bias. In the following, give efficient solutions in *O*-notation in terms of the parameters of the problem, *n* and *k*.

a) Assume that all parameters are represented in a computer using 64 bits for each parameter. What is the sample complexity of PAC learning this function class? Justify your solution.

Solution:
The parameters is consists of the parameters *W*, *U* and *w*. This gives the number of parameters $p = k^2 + 2k = O(k^2)$. The number of functions $|H|$ is no more than 2^{64p}. The sample complexity for PAC learning is $O\left(\frac{\log|H|/\delta}{\epsilon}\right) = O\left(\frac{p/\delta}{\epsilon}\right) = O\left(\frac{k^2/\delta}{\epsilon}\right)$.

b) In the usual recurrent neural networks, the same parameters *W* and *U* are used for every time instance *i*. Assume instead that a different set of parameters *W* and *U* are used for each time instance *i*. Assume that all parameters are still represented in a computer using 64 bits for each parameter. What is the sample complexity for PAC learning this function class?

Solution:
The number of parameters now depends on *n*, i.e. $p = (n-1)k^2 + nk + k = O(nk^2)$. The number of functions $|H|$ is no more than 2^{64p}. The sample complexity for PAC learning is $O\left(\frac{\log|H|/\delta}{\epsilon}\right) = O\left(\frac{p/\delta}{\epsilon}\right) = O\left(\frac{nk^2/\delta}{\epsilon}\right)$.

c) Assume that the gradient is computed using the back-propagation algorithm as described in the lecture notes. What is the amount of memory used by the algorithm, excluding the memory used for storing the parameters of the model? Justify your solution.

Solution:
For backpropagation, in the forward pass, we store the activations and also the output of the hidden units. In the backward pass, we store the δ values and the gradients of the activation functions. Altogether, these need $O(nk)$ amount of memory.

d) Assume that we are given the input sequence $x_1, x_2, \ldots, x_n$, and we would like to compute the value *y*. Describe a memory efficient algorithm for computing *y* and give the amount of memory used, excluding the memory used for storing $x_1, x_2, \ldots, x_n$ and the parameters of the model.

Solution:
To compute *y*, we only need to compute the forward pass in the graph. Once the output of the hidden layer h_i is computed, all the information at time $i' < i$ is no longer required and hence do not need to be stored. Hence only $O(k)$ amount of memory need to be used.

Problem 3. Winnow (10 Marks)

We analyse the Winnow algorithm, which can be used for online learning of disjunctions of Boolean variables. A disjunction is a concept of the form $x_{c1} \vee x_{c2} \vee \ldots \vee x_{ck}$ where $x_{c1}, x_{c2}, \ldots, x_{ck}$ are *k* Boolean variables selected from a larger set of variables $x_1, x_2, \ldots, x_n$. The Winnow algorithm works as follows:

- Each variable x_i is initialized with the weight $w_i = 1$.
- If $\sum_{i=1}^{n} w_i x_i \geq n$ the algorithm predicts 1, otherwise it predicts 0.
- If the prediction is 1 but correct label is 0, set $w_i = w_i/2$ for all *i* with x_i=1.
- If the prediction is 0 but correct label is 1, set $w_i = 2w_i$ for all *i* with x_i=1.

Let *P* be the number of mistakes where the correct label is 1 and *N* be the number of mistakes where the correct label is 0.

a) Argue that the *increase* in total weights on a mistake where the correct label is 1 is at most *n*.

Solution:
When the correct label is 1, the total weights of variables that have output 1 must be less than *n* if the prediction is incorrect. These weights are doubled, to become at most 2n. Hence the total increase is at most n.

b) Argue that the *decrease* in total weights on a mistake where the correct label is 0 is at least *n*/2.

Solution:
When the correct label is 0, the total weights of variables that have output 1 must be at least *n* if the prediction is incorrect. These weights are halved, so the total decrease is at least *n*/2.

c) Using (i) and (ii), argue that $N < 2P + 2$.

Solution:
The total weight is greater than 0 at all times. The initial value is n, the increase is at most *Pn* and the decrease is at least *Nn*/2. So the weight at any time is upper bounded by $n + Pn - Nn/2 > 0$. Rewriting that gives $N < 2P + 2$.

d) Argue that $P = O(k \log n)$ hence $P + N = O(k \log n)$.

Solution:
Each time there is a mistake when the correct label is 1, there must be a variable from $x_{c1}, x_{c2}, \ldots, x_{ck}$ whose weight is less than n. Otherwise the sum must be more than n, since the true concept is a disjunction of those variables. But each variable can only be in this position $O(\log n)$ times before its weight is at least n. As there are n variables the total number of such errors is $O(k \log n)$. Since $N < 2P + 2$, we have $P + N < O(k \log n)$.

END OF PAPER