NATIONAL UNIVERSITY OF SINGAPORE

CS5339: Theory and Algorithms for Machine Learning

(Semester 2: AY2017/18)

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	24	
Problem 2	8	
Problem 3	8	
TOTAL:	40	

Problem 1. Short Questions (24 Marks)

a) Assume that the density model used for regression is $p(y|f(x)) = \frac{1}{Z}\exp(-|y - f(x)|^d)$, where d is a fixed positive integer 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 $|y - f(x)|^d - \log Z$. Since Z is a constant, and log is a monotonic function, minimizing $|y - f(x)|^d$ corresponds to maximizing the log likelihood.

b) Assume that we are doing *early stopping* when learning using gradient descent. We set aside a validation set of size *m*, run *k* iterations of gradient descent and select the function that minimizes the empirical 0-1 loss on the validation set among the *k* functions found during the iterations of gradient descent. Assume that we are able to achieve a bound of ϵ for the difference between empirical and expected loss using *m* validation examples with probability $1 - \delta$. Now assume that we are running 100 times as many iterations (i.e. 100*k* iterations). We would like to have the same bound ϵ on the difference between empirical and expected loss with the same probability $1 - \delta$ using *Cm* validation examples, for some value *C*. Derive the value of *C*.

Solution:

The sample complexity for uniform convergence with $|H| = k$ is $m(\epsilon, \delta) \leq \frac{\log(\frac{2k}{\delta})}{2\epsilon^2}$.

With $|H| = 100k$, the sample complexity becomes $m'(\epsilon, \delta) \leq \frac{\log(\frac{2k}{\delta}) + \log(100)}{2\epsilon^2}$.

Hence $C = \frac{m'(\epsilon,\delta)}{m(\epsilon,\delta)} = 1 + \frac{\log(100)}{\log(\frac{2k}{\delta})}$.

c) Does the sample complexity for learning with deep neural networks with *E* weights suffer from the curse of dimensionality? Explain.

Solution:

No. The VC-dimension is $O(E \log E)$ which does not have dependence on dimensionality.

d) Argue that a disjunction of d variables can be represented using a decision tree with d+1 leaves. Use that to argue that a CNF with k clauses over d variables can be represented using a linear threshold combination of k decision trees of size no more than d+1.

Solution:

Assume that the variables are $x_1, x_2, \ldots, x_d$. One way to represent a decision tree is to have the positive child for x_i be a leaf with value 1 for i = 1 to d. The negative child for x_i would be a node with variable x_{i+1} for i = 1 to d-1 and be a leaf with value 0 for $i = d$. A CNF with k clauses is a conjunction of k clauses, hence can be represented as a conjunction of k decision trees with size no more than d+1 leaves. Finally, a conjunction can be represented with a linear threshold function with weights 1 for each input and k-0.5 for the threshold.

e) Assume that we would like to learn a degree k polynomial function in d variables for large k and d. Consider two methods: (i) constructing a feature from each possible monomial and learning a linear function of the features, or (ii) using kernel methods with a polynomial kernel. Which method would be preferred and why?

Solution:

For large k and d, the number of monomials grow exponentially with k and d. In contrast, the number of kernel in kernel methods grow linearly with the sample size m. In most practical situations in high dimensions, m would be much smaller than the number monomials, hence the kernel method is likely preferable.

f) Consider getting a corrupted transmission of a m by n image where only a fraction of pixels randomly distributed in the image is received and the other pixels are missing. It is known that the image can be well approximated as a low rank matrix of size m by n. Exploit this knowledge to design a learning algorithm for reconstructing the complete image. Describe the representation you would use and the optimization problem you would solve.

Solution:

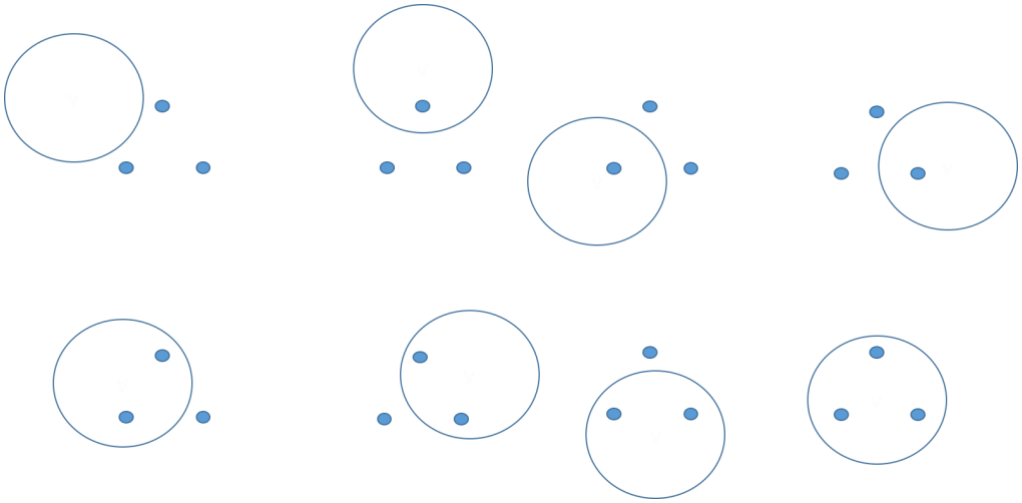
Given that the image has a low rank representation, one natural representation is to use matrix factorization. In this case, we approximate the image I with the product of two matrices UV^T where U is a m by k matrix and V^T is a k by n matrix. Each observed pixel $I(i,j)$ is approximated by $u_i^T v_j$ where u_i is the i-th row of U and v_j is the j-th column of V^T. We can use a loss function, e.g. the square loss to compute the loss between each observe pixel and its approximation and learn the matrices U and V to minimize the sum of the losses on the observed pixels.

Problem 2. Boosting Circles (8 Marks)

Consider the class of circles in the plane $\mathbb{R}^2$, $C = \{c_{p,r}(x): p \in \mathbb{R}^2, r \in \mathbb{R}\}$ where $c_{p,r}(x) = 1$ if $\|x - p\| \leq r$ and $c_{p,r}(x) = -1$ otherwise.

a) Show that the VC-dimension of *C* is at least 3. (You may draw all possible cases of the shattering.)

Solution: Eight possible functions with the same three points shown.



b) Assume that the target function can be represented by thresholding the function $f(x) = \frac{1}{3}\sum_{i=1}^{3} c_{pi,ri}(x)$, where the circles formed by $c_{p1,r1}(x)$ and $c_{p2,r2}(x)$ are disjoint and the circle formed by $c_{p3,r3}(x)$ contains the other two circles. Assume that the input domain is a subset of the circle formed by $c_{p3,r3}(x)$ (i.e. $c_{p3,r3}(x)$ is effectively a constant function with value 1 for all inputs). How many rounds of boosting would AdaBoost require to classify *m* examples correctly using a weak learner that minimizes the error using functions from *C* in each iteration of boosting?

Solution:

The margin of $f(x)$ is 1/3. Hence the weak learner is a γ-weak learning algorithm for $\gamma = 1/6$. AdaBoost requires at most $T = \frac{\log m}{2\gamma^2} = 18 \log m$ iterations to classify *m* examples correctly.

c) The VC-dimension of *C* is known to be exactly 3. Give a bound for the Rademacher complexity of ensembles of functions formed from functions in *C* on a sample with *m* data points.

Solution:

From Sauer's lemma, the number of functions induced by *C* on *m* points is at most $\left(\frac{em}{3}\right)^3$. From Massart's lemma, we get that the Rademacher complexity is at most $\sqrt{\frac{6\log(em/3)}{m}}$. Finally, the Rademacher complexity of the convex hull, and hence ensembles formed from *C* is the same as the Rademacher complexity of *C*.

Problem 3. Convolutional Neural Network (8 Marks)

Consider a convolutional neural network with a single convolution layer followed by a fully connected layer. The architecture is as follows:

- The input is a signal $s \in \mathbb{R}^n$, i.e. a vector of *n* real numbers.
- There are *k* convolution kernels in the convolution layer and each kernel has dimension *d*.
- In the detection stage of the convolution layer, the non-linearity is the threshold function (same non-linearity as in the linear threshold function).
- Max pooling is done within a neighbourhood of size 3 for the output of each kernel after the detector stage but there is no subsampling. Hence the output for each kernel after the detector and pooling stages is a vector of length *n*.

Each of the *k* outputs of the kernels after the max pooling stage is a vector of length *n*. This is flattened into a vector of length *kn* and used as input to a linear threshold unit to give the final output of the network.

a) Show that the VC-dimension of this network is $O(k(n+d)\log(k(n+d)))$.

Solution:

A single kernel after thresholding can form at most $\left(\frac{em}{d}\right)^d$ functions for *m* input points by Sauer's lemma. However, each of the *m* points are signals of length *n*. As the kernel is applied at every point of the signal, it creates *mn* points. This gives a bound of $\left(\frac{emn}{d}\right)^d$ possible functions. There are *k* kernels, giving a total of no more than $\left(\frac{emn}{d}\right)^{kd}$ possible functions. The max pooling stage does not have any parameters, hence does not increase the number of functions. Finally, the output layer has (*kn*+1) weights giving a total of no more than $\left(\frac{emn}{d}\right)^{kd}\left(\frac{em}{kn+1}\right)^{kn+1}$ possible functions. Solving for $2^m \leq \left(\frac{emn}{d}\right)^{kd}\left(\frac{em}{kn+1}\right)^{kn+1}$, we get $m \leq kd\log(emn) - kd\log d + (kn+1)\log em - (kn+1)\log(kn+1) \leq kd\log(em) + kd\log(n) + (kn+1)\log(em)$. By lemma A.2. of the textbook, this gives $m = O((kd+kn)\log(kd+kn))$.

b) The CNN uses parameter tying, i.e. only *k* kernels are used, where each kernel is applied at all locations of the signal. Consider the case where the parameters are not tied, i.e. the kernel parameters are different at each location of the signal, effectively giving *kn* distinct kernels overall. Give the VC-dimension of this network and discuss how it compares to the VC-dimension of the network with parameter tying.

Solution:

From lecture, we know that the VC dimension of a linear threshold network with $|E|$ weights is $O(|E|\log|E|)$. The presence of the max pooling layer does not change this bound as it does not increase the number of possible functions. The number of edges is $O(ndk+kn)$, hence the VC dimension is $O((nkd+kn)\log(nkd+kn))$. The part of the VC dimension attributed to the convolution layer is at least *d* times larger compared to when parameter tying is used.

c) Give the time complexity of computing the output of the network given an input signal *s* in *O*-notation. Comment on whether parameter tying give computational

advantages over a network without parameter tying for the purpose of computing the output of the network.

Solution:

Each filter is run through n points requiring $O(nd)$ time. For k filters we get $O(knd)$ time. For the fully connected part, the computation time is $O(kn)$, so the total time is $O(knd)$. Computed this way, there is no computational advantage in parameter tying.
Students are not expected to know this but it is actually possible to speed up the filter computation (with parameter tying) through using fast Fourier transform. FFT allows the filter output to be computed in $O(n \log n)$ time which may have advantages if $\log n$ is substantially smaller than d.

END OF PAPER