CS260 HW2

Shruti Sharan, UID: 405228029

1 Problem 1

(a) (10 pt) If classifier A has smaller training error than classifier B, then classifier A will have smaller generalization (test) error than classifier B.

False
Training and testing error of classifiers are largely dependant on the size of the dataset that is being considered. Usually training data is much larger in size than test data. A lower training error is expected when a method easily overfits to the training data, yet, poorly generalizes. Since we have no information about the sizes of the dataset, we cannot assume that just because Classifier A has smaller training error than classifier B, it will have a smaller generalization error too. If classifier B has a larger test set and smaller training set as compared to Classifier A, then this result will not hold true.

(b) (10 pt) The VC dimension is always equal to the number of parameters in the model.

False
The Vapnik-Chervonenkis dimension, VC(H), of a hypothesis space H defined over an instance space X, is the size of the largest finite subset of X that is shattered by H. For N points/parameters, the VC dimension is the largest value of N for which the growth function, $m_H(N) = 2^N$. Thus the value of VC Dimension is equal to the number of parameters, only when the model is able to shatter N parameters. Otherwise the condition holds that $N \leq d_{VC}(H)$.
For example, the sin function is an infinite-VC function with just one parameter. This example directly contradicts the given statement and hence is False.

(c) (10 pt) For non-convex problems, gradient descent is guaranteed to converge to the global minimum.

False
Gradient Descent is an algorithm which is designed to find optimal points of a given function, but these optimal points are not necessarily the global minimum. For a function which is convex, there is only one point at which the differen-

tial becomes zero, which is also the global minimum and thus the algorithm is guaranteed to converge. But for non-convex problems, this doesn't hold true. This is because of multiple reasons. A function may have many local minimas where the algorithm might get stuck. A local minima, may or may not be the global minima for non-convex functions. Other cases where gradient descent may not converge to a global minimum is if there are saddle points. These are points on the surface of the graph of a function where the slopes (derivatives) in orthogonal directions are all zero (a critical point) but it is not a local extremum of the function. Saddle points are common in non-convex functions. Very flat regions (Plateau effect) in the graph of the function also results in a similar problem. In all the above mentioned cases, the algorithm fails to converge to a global minima, despite having gradient equal to zero.
Another factor for non-convex functions is its varying curvature, which often leads to misbehavior of the gradient descent algorithm. Many times this causes the algorithm to jump into bad trajectories which take an arbitrarily long time, to get out of as the algorithm keeps oscillating inside the trajectory.
Since gradient descent is the most widely used algorithm for solving optimisation problems, non-convex problems become very hard to solve, since the algorithm doesn't guarantee convergence.

2 Problem 2.

(a) (20 pt) Which of the following is not a possible growth function $m_H(N)$ for some hypothesis set? (1)$2^N(2)2^{\sqrt{N}}(3)1(4)N^2N+2$ **(5)none of the other choices**.

The function $2^{\sqrt{N}}$ is not a possible growth function.
The growth function is the maximum number of ways into which n points can be classified. For some hypothesis set H,each input maps to a label output {+1, -1}. A dichotomy is a set of sampled data points from H that assign a label to the corresponding data points. The growth function counts the maximum number of dichotomies on n points. This value is the $max|H(x_1, x_2, ..., x_N)|$, and can at most be equal to 2^N. Thus option (1) 2^N) is a valid growth function.

If we consider a case in which all the points are of one class only, i.e. the labels for all the points are the same, then there is only one way to shatter them, with all the points lying only on one side of the classifier. In this case the growth function is equal to 1. Therefore option (3) is also valid.

A break point occurs if no data set of size k can be shattered by H. In this case the break point in equal to k. In the case of occurrence of a break point, we can show that the growth function can be bounded by a polynomial of N, where N is the number of points. By definition, we know,

$$m_H(N) \leq \sum_{i=0}^{k-1} \binom{N}{i}$$

The term N choose i, is a polynomial with degree (k-1) and thus, option (4) $N^2 - N + 2$ is also a valid option. We an verify this by putting k=3 and see that the condition holds.

Option (2) $2^{\sqrt{N}}$ is not valid, since it violates the above stated conditions. Also it is proved that there is no case of a growth function which lies in between a polynomial and an exponential function. If there are no break points, the function is bounded by 2^N, otherwise it is controlled by a polynomial. Thus $2^{\sqrt{N}}$ is not a valid growth function.

3 Problem 3

(10 pt) The gradient descent algorithm cannot be directly applied since the objective function is non- differentiable. Discuss why the objective function is non-differentiable.

The given objective function includes a regularization term $||W||_1$ or the L_1 norm, which is not continuously differentiable. This is because absolute value functions are not differentiable at the origin since they have a "kink", i.e. the derivative from the left does not equal the derivative from the right. Since the objective function contains a non-differentiable term, the entire function becomes non-differentiable.

(30 pt) In the class we showed that gradient descent is based on the idea of function approximation. To form an approximation for non-differentiable function, we split the differentiable part and non-differentiable part. Let $g(w) = ||Xw - y||_2^2$, as discussed in the gradient descent lecture we approximate g(w) by

$$g(w) \approx \tilde{g}(w) := g(w_t) + \nabla g(w_t)^T (w - w_t) + \frac{\eta}{2} ||w - w_t||_2{}^2$$

In each iteration of proximal gradient descent, we obtain the next iterate $(w_t + 1)$by minimizing the following approximation function:

$$w_{t+1} = argmin_w g(w) + \lambda ||w||_1$$

Derive the close form solution of w_{t+1} given w$_t$,$\nabla(w_t), \eta, \lambda$. What's the time complexity for one proximal gradient descent iteration?

The Lasso problem is defined as,

$$\underset{w}{\operatorname{argmin}} \frac{1}{2}||Xw - y||^2 + \lambda||w||_1 \tag{1}$$

For gradient descent, we do a successive approximation at each iteration, to form an approximation function of f $(\cdot)$:

$$f(w^t + d) \approx g(d) := f(w^t) + \nabla f(w^t)^T d + \frac{\eta}{2}||d||^2 \tag{2}$$

and then update the solution by $w_{t+1} \leftarrow w_t + d^*$
where,

$$\begin{aligned} d^* &= \underset{d}{\operatorname{argmin}}\, g(d) \\ &= \nabla g(d^*) \\ &= \nabla f(w^t)^T + \eta||d^*|| = 0 \end{aligned}$$

$$\implies d^* = -\frac{\nabla f(w^t)}{\eta} \tag{3}$$

Similarly, for Lasso Regression, we apply gradient approximation to the function and approximate it using the function:

$$w_{t+1} = \underset{w}{\operatorname{argmin}}\, g(w) + \lambda||w||_1 \tag{4}$$

To apply gradient descent to solve the Lasso Problem, we break it down into into a composite function,

$$\underset{w}{\operatorname{argmin}}\, f(w) := \{g(w) + h(w)\} \tag{5}$$

where $g(w) = ||Xw - y||_2{}^2$ is smooth and convex and $h(w) = \lambda||w||_1$ is convex but not differentiable everywhere.

The first part of the function, g(w) is approximated as it is, like shown before in Eq.(2). The second part of the equation, h(w) is a regularization term, and therefore we cannot apply gradient descent to it directly. For this term we approximate the function using the sub-gradient method.
Subgradients, generalize gradients to non-differentiable points by defining planes that lower bound the function.

$$\frac{\partial}{\partial w}h(w) = \lambda\frac{\partial}{\partial w}||w||_1 \tag{6}$$

$$= \begin{cases} -\lambda, & w < 0 \\ [-\lambda, \lambda], & w = 0 \\ \lambda, & w > 0 \end{cases} \tag{7}$$

Thus, putting it all together, from Eq.(3) and Eq. (7) we get three cases:

$$\begin{aligned}
case1 : w_t &< 0 \\
\nabla g(w_t) + \eta w^* - \lambda &= 0 \\
w^* &= \frac{-\nabla g(w_t) + \lambda}{\eta} \\
case2 : w_t &= 0 \\
w^* &= 0 \\
case3 : w_t &> 0 \\
\nabla g(w_t) + \eta w^* + \lambda &= 0 \\
w^* &= -\frac{\nabla g(w_t) + \lambda}{\eta}
\end{aligned}$$

This is also called the soft-thresholding operator, and is denoted by S.
Thus the final solution to the Lasso problem can be written as,

$$\begin{aligned}
w_{t+1} &= S(w_t - \frac{\nabla g(w_t) + \lambda}{\eta}) \\
where, \qquad \nabla g(w_t) &= X^T(Xw - y)
\end{aligned}$$

This is the closed form solution we get on solving the Lasso Regression problem using proximal gradient descent, given $w_t, \nabla g(w_t), \eta, \lambda$
Time complexity:
Assuming that we can pre-compute and store X^TX and X^Ty with the time complexity of $O(J^2N)$, the main computational cost in each iteration comes from calculating the gradient $\nabla g(w_t)$. The matrix that is computed at each iteration for X^TX has mostly zero (sparse). Thus for a more optimized solution, we consider only the non zero elements of X. Thus the time complexity of one iteration can be given as $\boldsymbol{O(nnz(X))}$.