Problem Set 1

Xujian Liang GTID: 903421366

1 For almost every case we have discussed where we are doing supervised learning, we have assumed a deterministic function. Imagine instead a world where we are trying to capture a non-deterministic function. In this case, we might see training pairs where the x value appears several times, but with different y values. For example, we might be mapping attributes of humans to whether they are likely to have had chicken pox. In that case, we might see the same kind of person many times but sometimes they will have had chicken pox, sometimes not. We would like to build a learning algorithm that will compute the probability that a particular kind of person has chicken pox. So, given a set of training data where each x is mapped to 1 for true or 0 for false:

1. Derive the proper error function to use for finding the ML hypothesis using Bayes Rule. You should go through a similar process as the one used to derive least squared error in the lessons

if we set error function is N(0, σ^2) Gaussian noise. And just duplicated the process during the lecture:

$$h_{ML} = \underset{h}{argmax} P(h|D)$$
$$= \underset{h}{argmax} P(D|h)$$
$$= argmax \prod_i P(d_i|h)$$
$$= argmax \prod_i e^{\frac{-\frac{1}{2}(d_i - h(x_i))^2}{\sigma^2}}$$
$$= argmax - \sum_i (d_i - h(x_i))^2$$
$$= argmin \sum_i (d_i - h(x_i))^2$$

we finally derived least squared error in the lessons.

2. Compare and contrast your result to the rule we derived for a deterministic function perturbed by zero-mean gaussian noise. What would a normal neural network using

sum of squared errors do with these data? What if the data consisted of x,y pairs where y was an estimate of the probability instead of 0s and 1s?

Nueral network will treat sum of squared errors as loss function during the training period.

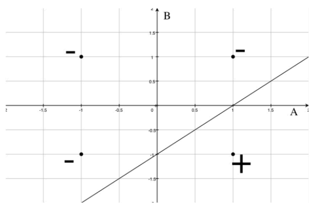
We will change the perception to fit proability instead of 0s and 1s.

2 Design a two-input perceptron that implements the boolean function $A \wedge \neg B$. Design a two-layer network of perceptrons that implements $A \oplus B$ ($\oplus$ is XOR).

The requested perceptron has 3 inputs: A, B and constant 1. The values of A and B are 1 or -1.

A	B	$A \wedge \neg B$
-1	-1	-1
-1	1	-1
1	-1	1
1	1	-1

One of the correct decision surfaces is shown below:



$$A - B - 1 = 0$$

Since A = 1 and B = -1 is negative. Thus we got

W0 = -1(for constant), W1 = 1(for A) , W2 = -1(for B)

For the second question:

Expressing A XOR B in terms of other logical conectives:

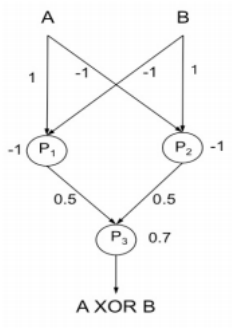
$$A\ XOR\ B = (A \wedge \neg B) \vee (\neg A \wedge B)$$

Defining the perceptrons P1 and P2 for$(A \wedge \neg B)$ and $(\neg A \wedge B)$.

Composing the outputs of P1 and P2 into a perceptron P3 that implement o(P1) ∨ o(P2) .

Since we have define P1 above, we could redefine it into B = A – 1, we will define P2 in the similar way A = B – 1

And we can also perceptron for P3. B = -A + 1



3.Derive the perceptron training rule and gradient descent training rule for a single unit with output o, where $o = w0 + w1x1 + w1x\ 2\ 1 + + wnxn + wnx^2_n$. What are the advantages of using gradient descent training rule for training neural networks over the perceptron training rule?

For perceptron rule, we need to repeat the following step while error exist

$$w_i = w_i + \Delta w_i$$

$$\Delta w_i = \eta(y - y')\times x_i$$

$$y' = \sum x_i\, w_i = \{a \geq 0\}$$

$$E = 1/2\times \sum_i (y - a)^2$$

y represent target and y’ represents output from our model.

And our goal is try to minimize E.

The gradient descent rule specifies how the weights are to be changed at each step of the learning procedure so that the prediction error of the unit decreases the most.

$$\frac{\partial E}{\partial w_i} = \sum_{x \in X} (out_x - o_x) \frac{\partial}{\partial w_i} (out_x - (w_0 + w_1 x_{1x} + w_1 x_{1x}^2 + \cdots + w_n x_{nx} + w_n x_{nx}^2))$$

$$= \sum_{x \in X} (out_x - o_x)(-x_{ix} - x_{ix}^2)$$

The advantage of gradient descent rule is that it is simple. It is easy for us to find the solution.

4. Explain how you can use Decision Trees to perform regression? Show that when the error function is squared error, then the expected value at any leaf is the mean. Take the Boston Housing dataset (https://archive.ics.uci.edu/ml/datasets/Housing) and use Decision Trees to perform regression.

The main difference between a regression tree and a classification tree is the how you measure the "badness" of a node. There are various ways to do it for both regression and classification trees. For regression trees, you could use sum of squared error (like OLS regression) or median absolute deviation (like quantile regression) or some other function.

To use Decision Trees to perform regression, we divide the predictor space into J distinct and non-overlapping regions, R1, . . . , Rn. For every observation that falls into the region Ri, we make the same prediction, which is the mean of the response values for the training observations in Ri. When picking the regions Rm, we would like to pick the regions Rm to minimize the squared error. So when splitting a tree, here is the different thing compared with classification we need to find a splitting attribute that maximizes the decrease in the error of the
tree resulting from this splitting.

I cannot access the Housing data through the URL.

5. Suggest a lazy version of the eager decision tree learning algorithm ID3. What are the advantages and disadvantages of your lazy algorithm compared to the original eager algorithm?

Start building the tree like in ID3(use information gain to decide on which attribute in to split) but build the tree only on the branch that has the value of the attribute in the query. In this way, you construct only one path in the tree, the one that can answer your query(the other ones are irrelevant for this query anyway). The big advantage is that you not build parts of the tree you don't need (by moving the computation from training phase to classification phase but saving exponential time in this way) and If the number of queries is small(when compared to the number of attributes for example) you can answer them faster. This becomes not so advantageous over a big number of queries. Since this version of lazy-ID3 gives exactly the answer the original ID3 gives there are no other disadvantages.

6. Imagine you had a learning problem with an instance space of points on the plane and a target function that you knew took the form of a line on the plane where all points on one side of the line are positive and all those on the other are negative. If you were constrained to only use decision tree or nearest-neighbor learning, which would you use? Why?

I will choose decision tree. The difference between KNN and decision tree is that decision tree is an eager learner however KNN is a lazy learner. For training part, decision tree will take lots of time to train the model but it will be very quick during query. KNN will save the time in train part but spend lots of time in query time. What's more, for points around the plane, it is hard for KNN to classify those points very clearly since maybe there are lots of points around our target has similar distance. It's not hard for decision tree to build a hyperplane to divide this space into two sides.

7 Give the VC dimension of the following hypothesis spaces. Briefly explain your answers. 1. An origin-centered circle (2D) 2. An origin-centered sphere (3D)

1. An origin-centered circle(2d):
   VC dimension of an origin-centered circle(2d) is 2. It is possible to shatter 2 points with an origin-centered circle. However, not set of 3 points can be shattered

with an origin-centered circle(2D). If there's a set of points at same radii r1 < r2 < r3 from the origin, there will be no function to label r1 and r3 with + and r2 with -.

2. AN origin-centered sphere(3d)
   Same as 2D. If there's a set of points at same radii r1 < r2 < r3 from the origin, there will be no function to label r1 and r3 with + and r2 with -.