1. Instructions

Homework Partners: None

2. Perceptron Training

Assume a three input perceptron plus bias (it outputs 1 if b + P i wi ∗ xi > 0, else 0). Assume a learning rate c of 1 and initial weights all 1: ∆wi = c(t − z) ∗ xi, where t is the true label and z is the predicted label. Show weights after each pattern in Table 1 until the result converges. Use an Excel sheet (attach your Excel sheet to the homework). Iterate over the training samples from top to bottom.

x1	x2	x3	t
1	0	1	0
1	1	0	0
1	0	1	1
0	1	1	1

1st Pass/Epoch

x_1	x_2	x_3	t	w1	w2	w3	b	net	z	w_1	w_2	w_3	b_updated
1	0	1	0	1	1	1	1	3	1	0	1	0	0
1	1	0	0	0	1	0	0	1	1	-1	0	0	-1
1	0	1	1	-1	0	0	-1	-2	0	0	0	1	0
0	0	1	1	0	0	1	0	1	1	0	0	1	0

2nd Pass/Epoch

1	0	1	0	0	0	1	0	1	1	-1	0	0	-1
1	1	0	0	-1	0	0	-1	-2	0	-1	0	0	-1
1	0	1	1	-1	0	0	-1	-2	0	0	0	1	0
0	0	1	1	0	0	1	0	1	1	0	0	1	0

This will never converge. The reason for this is that we have two rows with the exact same input, but differing labels. Therefore, with every pass of the data, when we reach the same row again, the weights will change, and this will be a never-ending process since the model's weights cannot fit perfectly/optimize for this case.

3. Input Validation

A SickBit health sensor produces a stream of readings from 20 different sensors (think blood pressure, heart rate body temperature, etc.). List two techniques you could use to check whether the stream of data coming from the sensors are valid or not. Write one or two sentences to describe each approach.

1. **Descriptive Statistics/Throughput:** To check for incomplete data, we can use descriptive statistics and check the number of values that are streaming over a period of time. If we see in a drop in the number of values over a period of time (throughput), then we know that there is something wrong with our streaming data. This works because streaming data, in this case, is sequential in nature, and we want to use the sequential nature of the data as a means of validating the data quality.
2. **Mean/Median from Historic Data:** Use past measurements to calculate mean/median values for each of the sensor readings (or, in some cases, can use expert's opinion on bottom and top values) and constantly check the incoming stream of data to make sure they fall within +/- 10% of those ranges, or look at the deviation in the moving mean. You can also update the mean/median values on the fly to account for new incoming data. Potential pitfall: If too much consistently invalid data flows in and we decide to update the mean/median, then the values can get skewed problems. Therefore, using historic data approved by experts is the best approach here.
3. **Lower/Upper Quartiles:** Alternatively, you could use past measurements to set a lower quartile and upper quartile threshold and ensure values fall within this range.
4. **Other Statistical Tests**: If we use some sort of ETL tool to transform the streaming data into categorical variables, we could use things like the chi-squared test to check for correlations/independence in the data: no relationship exists on the categorical variables in the population; they are independent
5. **Live Dashboards**: This is a popular way of looking at and validating streaming data. It often involves either live updates on charts from the data lake, or alternatively, a live update on sensors, their healths, and processed dumps of their logs in real-time. If things are working correctly, then the odds are that the values the sensors produce are also valid.

4. Distributions

Galton measured the heights of individuals in 200 families, each of which included one mother, one father, and a varying number of adult sons. The three histograms of heights in Figure 1 depict the distributions for all mothers, fathers, and adult sons. All bars are 2 inches wide. All bar heights are integers. The heights of all people in the data set are included in the histograms.

(a)

(i) 60-62: 6% * 2 = 12%
62-64: 14% * 2 = 28%
Therefore 60-64%: **40%**

(ii) **Unknown**: Since the histogram measures intervals of 2 inches each, we can't determine an odd number like '67' because we don't know the distribution of 66 and 67 in an interval from 66-68.

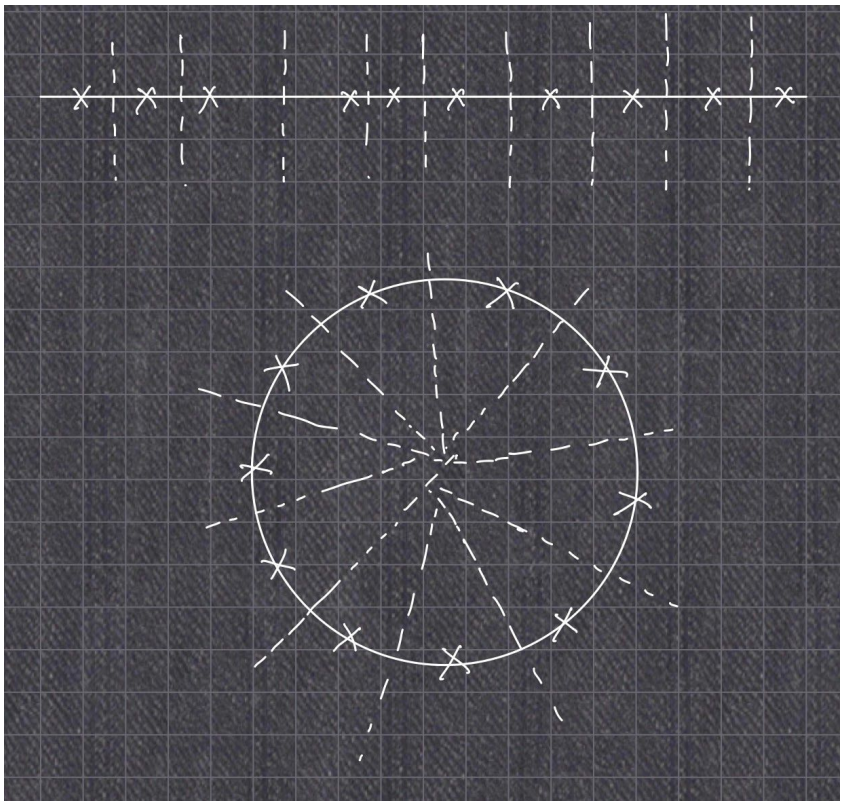
(iii) **Unknown**: We don't actually know the number of sons since it is 'varying' number of each family, therefore we cannot compute an exact number of this question.

(iv) P(X>60) = 1-P(X<60) = 100% - 2%*2 = 96%
Therefore, the percentage is 96%*(200) = **192 mothers**

(b) The total percentage is 2*8% + 2*2% = 20%. Since this will be spread out over 4 inches and the y-axis represents *percent per inch*, we divide this by 4 to get **5%**.

(c) The percentage of sons that are taller than all of the mothers is **between 20% and 48%.** The lower limit can be found as follows: All mothers are less than 72 inches. Therefore, 2*8% + 2*2% = 20% of the sons (the 72-74 bucket, as well as the 74-76 bucket) than all mothers. For the 70-to-72 bin, there is a sense of ambiguity that gives rise to the upper limit. If all the sons in that bucket, which equals 2*14% = 28% of sons in that bin, are taller than all mothers in the corresponding bin for the mothers' histogram, then 2*8% + 2*2% + 2*14% = 48% of the sons would be taller than all mothers. This situation arises if the mother in that bucket are all of height 70 inches, while the sons in the same bucket are all of height 71 inches.

5. Voronoi



(For the 10 points on the circle, if the points were equally spaced on the circumference of the circle, then all the boundaries (two areas created by adjacent perpendicular bisectors) would ‘exactly’ meet in the center. In my case, I was unable to space them equally, so I tried my best to draw bisectors, which as expected, did not intersect/meet at the center of the circle.

Things in common:

1. (n-1) lines between the n points -> 9 lines in between 10 points
2. n Voronoi cells/regions/boundaries of separation.
3. Each line is equidistant from two adjacent points (the perpendicular bisector)
4. All points on the perpendicular bisector are equidistant from the adjacent points through which the perpendicular bisector runs.

6. Augmentation

a) I think it could be interesting to cross *cp* and *restecg*. Crossing these together could actually point to rather abnormal conditions, like (0,0) -> asymptomatic and showing probable or definite left ventricular hypertrophy by Estes' criteria, which is definitely an abnormality, and the hope is that by crossing these features, the model will essentially create a new, more specific categorical variable which may show much more dramatic/important instances just the individual ones themselves. It's even more interesting because both of these are negatively correlated, but after crossing, it could actually show a strong positive correlation.

b) There are two ways we could create new features from latitude and longitude
   i) Bounding box split into 10-25 distinct regions using max/min of latitudes and longitudes in the dataset. Then, we represent each region with a unique number, and one-hot encode it before passing it through our machine learning model. Therefore, instead of placing advantage of individual values, the model may be able to learn that certain distinct/nonoverlapping "regions" are more important than others.
   ii) Cluster (using k-means or x-means clustering) with k=10, then come up with bounding boxes/regions and labels for each region based on data point density. This could indirectly help in dimensionality reduction since we can only use regions with high density while marking all other shallow density regions under the same label. However, this solution is not scalable but would work well if we had a predefined region such as a part of a city, an entire city, or a state.

c) If we use a linear model like linear regression, we essentially want to create a dataset such that:

$x\theta_1 + y\theta_2$ = Z, where theta 1 and theta 2 are coefficients to be learned.

Let the two labels for Z (assuming a binary task) be A and B.

A -> negative values, B -> positive values.

X	Y	Z
-2	4	A
4	-3	A
4	3	B
3	4	B
2	-4	A
-4	3	B

However, if we created a feature cross ‘W’ = XY, then that has a linear relationship with the target variable and would be weighted VERY heavily in each prediction. We would probably be able to see if this in Python if we simulated this using any linear model and then tried model.get_coef().