sta314 HW1

shimmy.nau

September 2021

1 Question 1

1.1 1a

```
# -*- coding: utf-8 -*-



def process_data(data, labels):
    """
    Preprocess a dataset of strings into vector representations.

    Parameters
    ----------
        data: numpy array
            An array of N strings.
        labels: numpy array
            An array of N integer labels.

    Returns
    -------
    train_X: numpy array
        Array with shape (N, D) of N inputs.
    train_Y:
        Array with shape (N,) of N labels.
    val_X:
        Array with shape (M, D) of M inputs.
    val_Y:
        Array with shape (M,) of M labels.
    test_X:
        Array with shape (M, D) of M inputs.
    test_Y:
        Array with shape (M,) of M labels.
    """
```

```
# Split the dataset of string into train, validation, and test
# Use a 70/15/15 split
# train_test_split shuffles the data before splitting it
# Stratify keeps the proportion of labels the same in each split

# -- WRITE THE SPLITTING CODE HERE --
tr_X, ts_X, tr_Y, ts_Y = train_test_split(
    data, labels, test_size = 0.3, stratify = labels)


Val_X, tes_X, Val_Y, tes_Y = train_test_split(
    ts_X, ts_Y, test_size = 0.5, stratify = ts_Y)
#print(Val_Y)

# Preprocess each dataset of strings into a dataset of feature vectors
# using the CountVectorizer function.
# Note, fit the Vectorizer using the training set only, and then
# transform the validation and test sets.

# -- WRITE THE PROCESSING CODE HERE --
vectorizer = CountVectorizer()
training_X = vectorizer.fit_transform(tr_X)
testing_X = vectorizer.transform(tes_X)
valing_X = vectorizer.transform(Val_X)


# Return the training, validation, and test set inputs and labels

# -- RETURN THE ARRAYS HERE --
return training_X, tr_Y, valing_X, Val_Y, testing_X, tes_Y
```

1.2 1b

```
# -*- coding: utf-8 -*-

def select_knn_model(train_X, val_X, train_Y, val_Y):
    """
    Test k in {1, ..., 20} and return the a k-NN model
    fitted to the training set with the best validation loss.

    Parameters
    ----------
        train_X: numpy array
            Array with shape (N, D) of N inputs.
        train_X: numpy array
            Array with shape (M, D) of M inputs.
        train_Y: numpy array
            Array with shape (N,) of N labels.
        val_Y: numpy array
            Array with shape (M,) of M labels.

    Returns
    -------
    best_model : KNeighborsClassifier
        The best k-NN classifier fit on the training data
        and selected according to validation loss.
      best_k : int
        The best k value according to validation loss.
    """

    lst_of_models = []
    for k in range(1,21):
        # want values from 1:20
        neigh = KNeighborsClassifier(n_neighbors = k)
        t = neigh.fit(train_X, train_Y)
        s = neigh.score(val_X, val_Y)


        lst_of_models.append((s, t, k)) # k- 1 because indexing starts at 0 for
    print(lst_of_models)
    return max(lst_of_models,key = lambda t: t[0])[1], max(lst_of_models,key = la
# the max function returns the tuple with the highest s value,
#this is because of the key
```

1.2.1 output

```
Selected K: 9 Test Acc: 0.6591836734693878
```

1.3 1c

results
Selected K: 19 Test Acc: 0.746938775510204
the metric = 'cosine' seems to compute the distance, by considering the cosine of the angle between the two vectors, using the dotproduct definition. Therefore,

$$\cos\theta = \frac{a \cdot b}{\|a\| \, \|b\|}$$

This is a much more accurate measure because of the size of our vectors, taking the euclidean distance between two different vectors loses it's meaning. Further, it is important to realize, that by the definition of $a{\cdot}b$ when there is an increase in one of the vectors in the count of a certain word, and the other vector also has at least that word occuring once, then this term in the sum will have greater weight in turn causing $\cos(\theta)$ to be larger. Furthermore, by my personal understanding of fake news, repetition of words would be higher therefore resulting in a closer angle between the different vectors as these values will have more weight when considering the dot product and hence the angle.

2 Question 2

2.1 a

since we know that x_i, y_i are all independent, therefore we know that all $z_i = (x_i - y_i)^2$ are also independent. As a result, we are able to apply linearity of expectation. since we know that there are d terms in R therefore, we can find the expectation of

$$R = \sum_{i=1}^{i=d} \mathbb{E}(Z_i)$$

.

similarly, since we know that since each z_i is indepedent as outlined above, we are also able to apply linearity of variance to determine that the variance is

$$var(R) = \sum_{i=1}^{i=d} var(Z_i)$$

3 Question 3

note
For the purpose of simplicity in typescripting we will ignore the $\frac{1}{2}$ coefficient in front of both sides of the equation as it will not make a different in the inequality as it can be factored out of the sum on the left side of the equation per rules of sums, and therefore, we can multiply both sides of inequality by two in order to get rid of the coefficient.

Let us begin by defining a random variable D. Let us consider D to be a distributed so that$P(D = h_i) = \frac{1}{m}$ where $i = 1, \cdots, m$. Therefore, we can notice that

$$E[D] = \frac{\sum_{i=1}^{i=m} h_i}{m} \tag{1}$$

as this is a discrete RV. Now let us begin by considering the left hand side of the inequality, $L(\overline{h}(x), t)$. If we expand the left side of the inequality, $\frac{1}{2}(\overline{h}(x) - t)^2$ we end up with

$$(\overline{h}(x)^2 - 2\overline{h}(x)t + t^2) \tag{2}$$

now if we further expand this $\overline{h}(x)$ we can notice that $\overline{h}(x)^2 = (\frac{1}{m}\sum_{i=1}^{m} h_i(x))^2 = E[D]^2$ as per (1). Therefore, if we combine this with (2), we end up with

$$E[D]^2 - 2tE[D] + t^2 \tag{3}$$

Now let us consider the right side of the inequality. Let us first notice that the expanding of each term within the sum is

$$h_i(x)^2 - 2th_i(x) + t^2 \tag{4}$$

We can therefore split up the sum as per rules of summations and bring the $\frac{1}{m}$ inside each sum as it is independent of i. Therefore, after expanding the left side of the equation we end up with

$$\sum_i (\frac{1}{m}h_i(x)^2) - 2t\sum_i (\frac{1}{m}h_i(x)) + \cancel{m}\frac{1}{\cancel{m}}t^2 \tag{5}$$

Now we can notice that since this is a discrete RV, $E(D^2) = \frac{1}{m}\sum_1^m h_i(x)^2$. Therefore, we are again able to further rewrite (5) to become

$$E[D^2] - 2t(E[D]) + t^2 \tag{6}$$

Therefore we have reduced our inequality to (6) and (3). Therefore, we can notice that since the terms t^2 and $-2tE[D]$ are common to both equations we can remove them from the inequality. Therefore our inequality further reduced to

$$E[D]^2 \leq E[D^2] \tag{7}$$

Which we know to be true as the hint outlines that $E[D]^2 - E[D]^2 \geq 0$ which implies that $E[D]^2 \leq E[D^2]$ as desired. $\square$

4 Question 4

4.1 4a

Let us begin by noticing that there is only 4 possible predictors for y which we will call $y_1, \cdots, y_4$

$$y_1(x) = x \text{ identity function} \quad (1)$$
$$y_2(x) = 1 - x \text{ inverts the values} \quad (2)$$
$$y_3(x) = 1 \text{ constant function to 1} \quad (3)$$
$$y_4(x) = 0 \text{ constant function to 0} \quad (4)$$

. Now let us notice that since we are dealing with the L_{0-1} function and x takes the values of $(0, 1)$ then we can notice the following[1].

$$E[E[T \mid X] = E[\sum_T t \cdot P(T = t \mid X)] \quad (5)$$
$$= \sum_X [\sum_T t \cdot P(T = t \mid X = x)]P(X = x) \quad (6)$$
$$= \sum_X \sum_T t \cdot P(t, x) \text{ per the definition of joint probability} \quad (7)$$

Now since we know that the conditional probability of $p(t \mid x)$ are equal, we can use the values from the L_{0-1} as t implying that our desired function outlined in the question is identical to (7) and therefore equivalent to (5). Therefore, let us computer (5) for all the possible predictors. We can notice that for all the predictors, L_{0-1} will take the value of 0 and the value of 1 half the time, therefore, it doesn't matter if we go through each option or just compute it all at once. We can notice this is will be true by noticing if we consider (7) that since we increment over the X's and the T's, we can notice that they will align only half the time, and therefore, we can replace $t = \mathbb{I}(t \neq (y(x))$ which is the equivalent to the 0-1 lost function in our case, and realise that t = 1 half the time and 0 half the time. Therefore, regardless of our value of (y(x)) it will always be equal to t half the time and not equal to t half the time. Therefore We can notice from (5) that we Can calcualte

$$E[\sum_{x \in \{0,1\}} xP(x = x \mid y)] = E(0 + \frac{1}{2}) = \frac{1}{2}$$

as the expectation of a contestant is that contestant. Therefore, we have shown that for all predictors $y_1, \cdots, y_4$ $E[l_{0-1}] = \frac{1}{2}$ as desired. $\square$.

[1]help with noticing this proof from wiki

4.2 4b

Let us begin by defining the following set $P = \{p(0 \mid x), p(1 \mid x)\}$. Let us now notice that the two elements in the set are not equal per assumption and they are both greater than or equal to 0 per the definition of the probability function. Now let us define $M = max(P)$. we know that this is a well defined function as this is a finite set which has a maximum. We can now notice that $y^*(x) = arg \max_{t \in \{0,1\}} p(t \mid x)$ is a constant as it will always return the value of t such that $p(t \mid x) = M$. Now let us consider the loss function, since we know that with certainty that $y^*(x)$ will return the t which results in $p(t \mid x) = M$, therefore, since we have shown that we can equivalent use (5) from the previous question we can realize the when when $L_{0-1} = 0$ when $y^*(x) = t$. This is because the function $y^*(x)$ returns the t which maximize $p(t \mid x)$. Therefore, we know that when it returns $t = y * (x), p(t \mid x) = M$. Therefore, since we can use (5) use we can expand it and realize that we are trying to calculate $E[0 \times M) + 1 \times (1 - M)] = 1 - M$ per the expectation of a constant. The reason for the $(1 - M)$ is because we know that the sum of the probability of the two conditional probabilities must be 1 and therefore $1 - M$ is the compliment and must be strictly less than M as $p(0 \mid x) \neq p(1 \mid x)$. Now let us notice that we must, consider the four possible predictors, we can first begin by noticing that our y^* must be a constant function. Now let us consider the following table each

function	x_value	t = 0	t = 1
y_1	0	p(0$\mid x) p(x = 0)$	0
	1	0	p(1 $\mid x) p(x = 1)$
y_2	0	0	p(1 $\mid x) p(x = 0)$
	1	p(0 $\mid x) p(x = 1)$	0
y_3	0	p(0 $\mid x) p(x = 0)$	0
	1	p(0 $\mid x) p(x = 1)$	0
y_4	0	0	p(1 $\mid x) p(x = 0)$
	1	0	p(1 $\mid x) p(x = 1)$

cell in the table is the result from the different functions defined at the beginning of the previous question, and is the result of applying the definition of the join probability to result in $p(t \mid x)p(x)$. This would be expanding upon (6). Now let us now that either $p(0 \mid x)$ or $p(1 \mid x)$ is M. Therefore, we can notice that either $M = p(0 \mid x)p(x = 0) + p(0 \mid x)p(x = 1)$ or M = the same thing with replacing replace the term $p(0 \mid x)$ with $p(1 \mid x)$. Now let us consider the following set $X = \{p(x = 1), p(x = 0)\}$. We know that there is a maximum of this set, which we shall call t, however, we do not know that the two elements are not equal. Nonetheless we do know that both elements of the set add to one. Therefore, we know that $\min(X) = 1 - t$. Now let us notice that the optimal predictor would be defined as follows $(1 - M)(t) + (1 - M)(1 - t) = 1 - M$. The reason that the optimal predictor would have a value of $1 - M$, is because by (5), the L_{0-1} function will not match when $P(t \mid x)$ does not equal M, and by compliments of probability, it therefore, must be equal to 1-M. Therefore, it would suffice us to show that for all possible predictors it would be greater than 1-M. Let us notice that the other constant predictor would return M for expected error, as the predictor would match the t when we M is the other term in the product and we know by assumption that $M > 1 - M$. Therefore, we just need to show that the two other predictors have an error greater than $1 - M$. Let us notice that when we are dealing with non constant predictors, the M must appear in one term of the sum. Therefore we either have $Mt + (1 - M)(1 - t)$ or we have $M(1-t)+(1-M)t$. Now we will show that these are both greater than $1-M$.Let us first make sure to note that all terms are non-negative, which is important to ensure that certain inequalities hold. Let us begin with $Mt + (1 - m)(1 - t)$.

$$\begin{aligned} Mt + (1 - M)(1 - t) &> (1 - M)(t) + (1 - M)(1 - t) \text{ as } M > (1 - M) \text{ by assumption} && (1) \\ &= (1 - M)(t + 1 - t) && (2) \\ &= 1 - M && (3) \end{aligned}$$

Now we will show that $M(1 - t) + (1 - M)(t)$ is greater than 1-m.

$$\begin{aligned} M(1 - t) + (1 - M)(t) &> (1 - M)(1 - t) + (1 - M)(t) \text{ as } M > (1 - M) && (4) \\ &= (1 - M)(1 - t + t) && (5) \\ &= 1 - M && (6) \end{aligned}$$

Therefore, since we have shown that for all the other predictors the expected loss is greater than 1-M, and equality only holds when the predictor is the same predictor as the one desired. Therefore we have shown as desired □

4.3 5c

Let us make our distribution as such

	t= 0	t = 1
x = 0	0	0
x = 1	1	0