Home Assignment 1

Yu Wang(ndp689)

November 30, 2021

# Contents

# 1 Make Your Own

1) What profile information would you collect and what would be the sample space $X$?
   Profile information: their grades in *Linear Algebra*, *Calculus*, *Statistics and Probability Theory*.
   The collection of (the grade in the Linear Algebra course, the grade in the Calculus course, the grade in the Statistics and Probability Theory course) of all students in the Machine Learning course is sample space, then $X = R^3$.

2) What would be the label space $Y$?
   The collection of truth final grades of all students in the Machine Learning course is the label space, then $Y = R$.

3) How would you define the loss function $l(y', y)$?
   $l(y', y) = |y' - y|$.

4) Assuming that you want to apply K-Nearest-Neighbors, how would you define the distance measure $d(x, x')$?
   Using the Euclidian distance.

5) How would you evaluate the performance of your algorithm? (In terms of the loss function you have defined earlier.)
   If the value of the loss function $l(y', y)$ is closer to 0 (ideally, we want it to be 0), which means the performance of the algorithm is good.

6) Assuming that you have achieved excellent performance and decided to deploy the algorithm, would you expect any issues coming up? How could you alleviate them?
   Possible issues: the grading systems of the prior courses are different.
   Alleviate issues: convert grades under different grading systems into the same systems, like course grades $\in$\{A,B,C,D,E,F\}

# 2 Digits Classification with $K$ Nearest Neighbors

- Some details of my code

  How to implement K-NN?
  The implementation of the K-NN algorithm is as follows

```python
def KNN(test_vec, train_data, train_label, k):
    train_data_num = train_data.shape[1]
    dif_mat = np.tile(np.reshape(test_vec, (784, 1)),
    (1, train_data_num)) - train_data
    sqr_dif_mat = dif_mat ** 2
    sqr_dis = sqr_dif_mat.sum(axis=0)
    sorted_idx = sqr_dis.argsort() # get the index
    class_cnt = {}  # key is 5 or 6
    for i in range(k):
        tmp_class = train_label[sorted_idx[i]]
        if tmp_class in class_cnt:
            class_cnt[tmp_class] += 1
        else:
            class_cnt[tmp_class] = 1
        sortedVotes = sorted(class_cnt.items(),
        key=operator.itemgetter(1), reverse=True)
    # return the prediction of this test data
    return sortedVotes[0][0]
```

I use 'np.tile' function to create a matrix, so that we can calculate all distances between the tested data and the 100 training data in one shot. And using 'argsort' function to get the indices that would sort all the distances in ascending order.
And I use a dict (key is 5 or 6, and value is the number of K nearest neighbors labeled 5 or 6) to store the number of K nearest neighbors labeled 5 or 6. Then, we sort the dict in descending orderand return the first key in the dict as the prediction.

How to break ties in K-NN?
As the K-NN algorithm shows above, I use the dict to store the number of K nearest neighbors labeled 5 or 6. If the value of 5 and the value of 6 are the same, my code will choose the first key in the dict to be the prediction. Because the first one is the nearest neighbor's label and the tested data is more similar to the neighbor.

How to compute the validation error?
I use zero-one loss as the loss function. Then, the validation error is:

$$\hat{L}(h_{KNNS_t}, S_{val}) = \frac{1}{n}\sum_{i=1}^{n} l(h_{KNNS_t}(X_i), Y_i)$$

n $\in \{10, 20, 40, 80\}$

## 2.1 Task 1

1. What can you say about fluctuations of the validation error as a function of n?

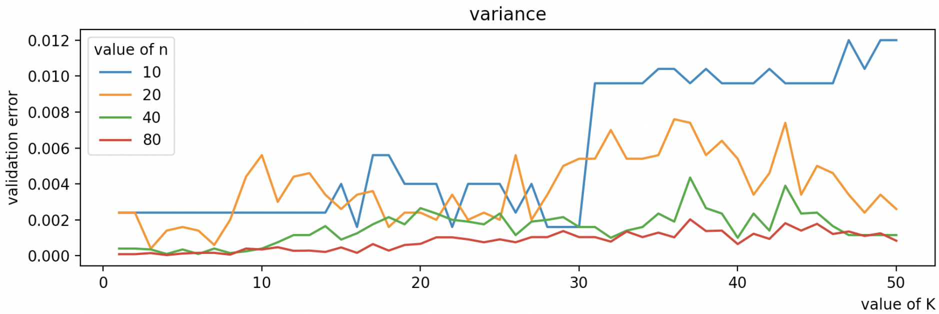


Figure 1: the variance of the validation error

According to the Figure 1, the fluctuation of the validation error is smaller with the increase of n.

2. What can you say about the prediction accuracy of K-NN as a function of K?

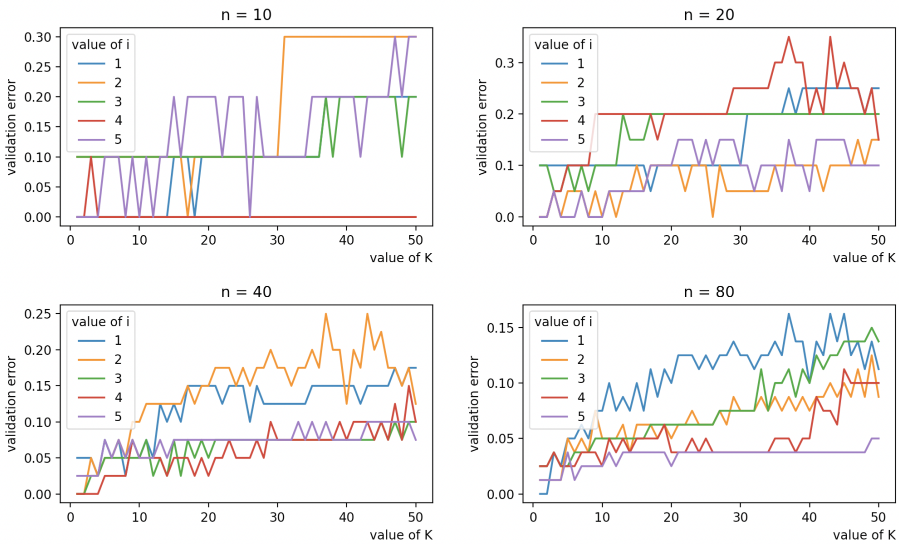


Figure 2: the validation error in different value of n

According to the Figure 2, the prediction accuracy of K-NN decreases with the increase of K.

## 2.2 Task 2

1. How corruption magnitude influences the prediction accuracy of K-NN and the optimal value of K?

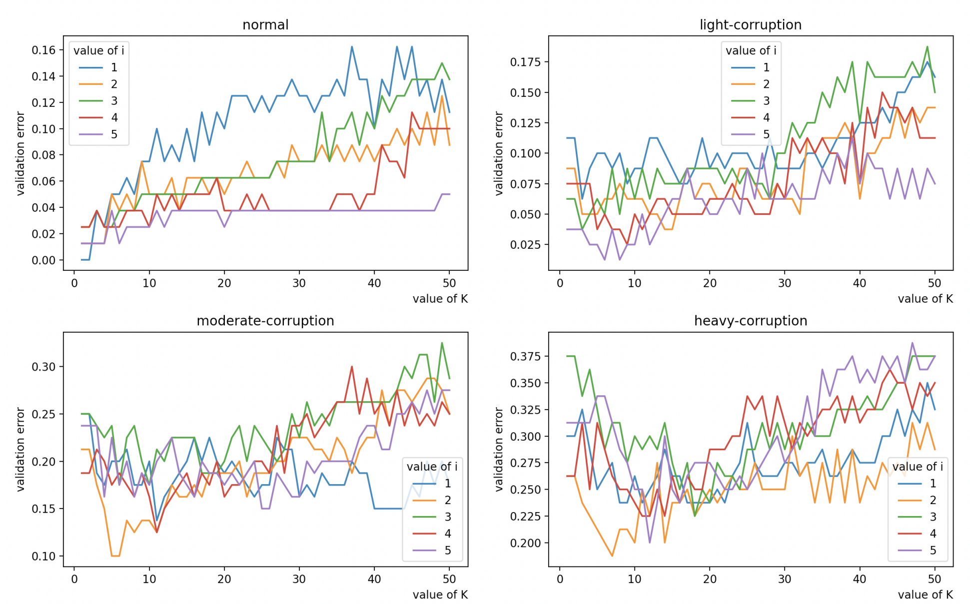


Figure 3: the fluctuations of validation error

According to the Figure 3, the heavy corruption has lower accuracy, and the light corruption has higher accuracy (Surely, the normal MNIST SUBSET has the highest accuracy).
The optimal value of K becomes larger with heavier corruption.

# 3 Illustration of Markov's, Chebyshev's, and Hoeffding's Inequalities

## 3.1 2.a Bernoulli random variables $X_1, ..., X_{20}$ with bias 0.5 and $\alpha \in$ {0.5,0.55,...,1}.

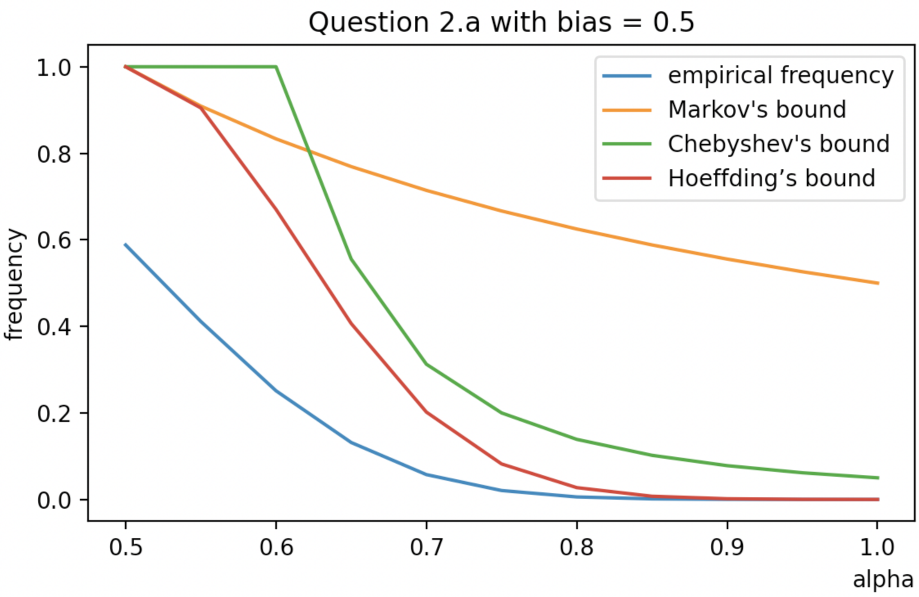


Figure 4: the empirical frequency and the upper bounds with bias = 0.5

1) Why the above granularity of $\alpha$ is sufficient?
   Because there are only some values that the mean $\frac{1}{20}\sum_{i=1}^{20} X_i$ can take (we have 20 coins, so, it has step of 0.05). When we want to compute the probability of a mean large than or equal to 0.51, which is the same as large than or equal to 0.55.

2) Check the assumptions of the Markov's Inequality is satisfied on $P(\frac{1}{20}\sum_{i=1}^{20} X_i \geq \alpha)$:
   Obviously, $\frac{1}{20}\sum_{i=1}^{20} X_i$ is non-negative, and $\alpha > 0$, so it is satisfied.

3) Check the assumptions of the Chebyshev's Inequality is satisfied on $P(\frac{1}{20}\sum_{i=1}^{20} X_i \geq \alpha)$:
   First, we need to convert $P(\frac{1}{20}\sum_{i=1}^{20} X_i \geq \alpha)$:

$$
\begin{aligned}
&P\left(\frac{1}{20}\sum_{i=1}^{20} X_i \geq \alpha\right) \\
&\qquad = P\left(\frac{1}{20}\sum_{i=1}^{20} X_i - E\left[\frac{1}{20}\sum_{i=1}^{20} X_i\right] \geq \alpha - E\left[\frac{1}{20}\sum_{i=1}^{20} X_i\right]\right) \\
&\qquad = P\left(\frac{1}{20}\sum_{i=1}^{20} X_i - E\left[\frac{1}{20}\sum_{i=1}^{20} X_i\right] \geq \alpha - \mu\right) \\
&\qquad \leq P\left(\left|\frac{1}{20}\sum_{i=1}^{20} X_i - E\left[\frac{1}{20}\sum_{i=1}^{20} X_i\right]\right| \geq \alpha - \mu\right)
\end{aligned}
$$

We know that $\alpha - \mu \geq 0$. When $\alpha - \mu = 0$, we cannot compute $\frac{Var[Z]}{\epsilon^2}$ in the Chebyshev's Inequality. So, I use 'try ... except' to catch the exception and let its upper bound = 1. When $\alpha - \mu > 0$, $\alpha - \mu$ is the $\epsilon$ in the Chebyshev's inequality. It is satisfied.

4) Check the assumptions of the Hoeffding's Inequality is satisfied on $P(\frac{1}{20}\sum_{i=1}^{20} X_i \geq \alpha)$:
We can use the same way in 3) to convert $P(\frac{1}{20}\sum_{i=1}^{20} X_i \geq \alpha)$. It is also satisfied.

5) Compare the four plots.
When alpha is less than or equal to 0.65, the Chebyshev's upper bound is relatively loose, and when alpha is large than or equal to 0.65, the Markov's upper bound is loose. In general, the Hoeffding's upper bound is relatively tighter than other two inequalities' upper bound, and the empirical frequency is the tightest one.

5) For $\alpha = 1$ and $\alpha = 0.95$ calculate the exact probability $P(\frac{1}{20}\sum_{i=1}^{20} X_i \geq \alpha)$
When $\alpha = 1$, the 20 coins need to be flipped to the same side with value of 1. So, we have,

$$P(\frac{1}{20}\sum_{i=1}^{20} X_i \geq 1) = P(X = 20) = 0.5^{20}$$

Similarly, when $\alpha = 0.95$, we have,

$$
\begin{aligned}
P(\frac{1}{20}\sum_{i=1}^{20} X_i \geq 0.95) &= P(X = 19) + P(X = 20) \\
&= 20 * 0.5^{19} * 0.5 + 0.5^{20} = 21 * 0.5^{20}
\end{aligned}
$$

### 3.2 2.b Repeat the question with $X_1, ..., X_{20}$ with bias 0.1 and $\alpha \in$ {0.1,0.15,...,1}.

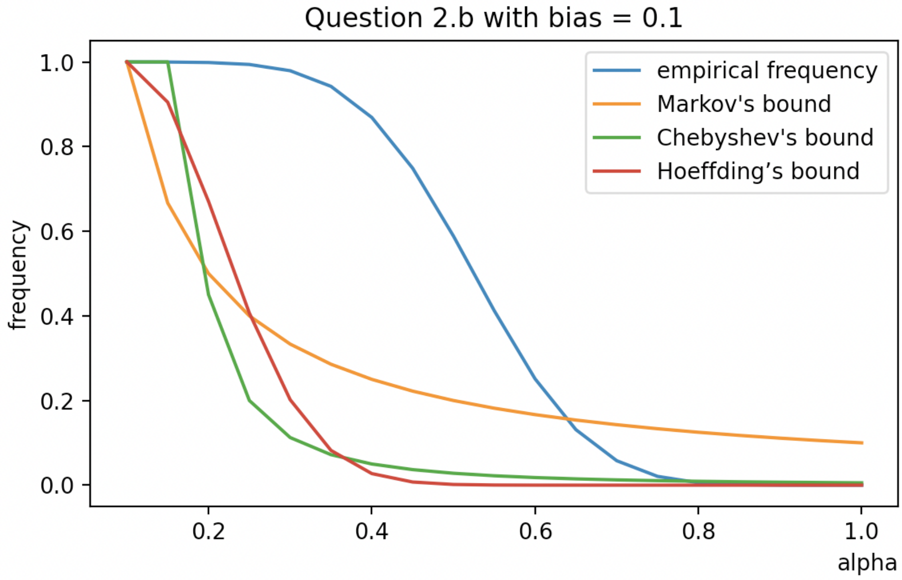


Figure 5: the empirical frequency and the upper bounds with bias = 0.1

- Compare the four plots.
  When alpha is less than or equal to 0.6, the empirical frequency is loose but the Chebyshev performs well. When alpha is large than or equal to 0.6, the Markov's upper bound is loose. In general, the Hoeffding's upper bound is relatively tighter.

## 4 Basic Linear Algebra

### 4.1 Let $h_{w,b}$ be a hyperplane given by the equation $w^T x + b = 0$. (I.e., $h_{w,b}$ is the set of points x : $w^T x + b = 0$ .) Calculate the distance from the hyperplane to the origin.

Assume $x_1$ is a point in $h_{w,b}$, and x is a point outside the hyperplane. So, the distance d from x to the hyperlane must be $|\vec{u}(x - x_1)|$ ($\vec{u}$ is a unit normal vector of $h_{w,b}$), and we know that $w^T/||w||$ is a unit normal vector of $h_{w,b}$, then,

$$d = \frac{|w^T(x - x_1)|}{||w||} = \frac{|w^T x - w^T x_1|}{||w||} = \frac{|w^T x + b|}{||w||}$$

So, the distance from the hyperplane to the origin is: $d = \frac{|b|}{||w||}$

# 5 Regression

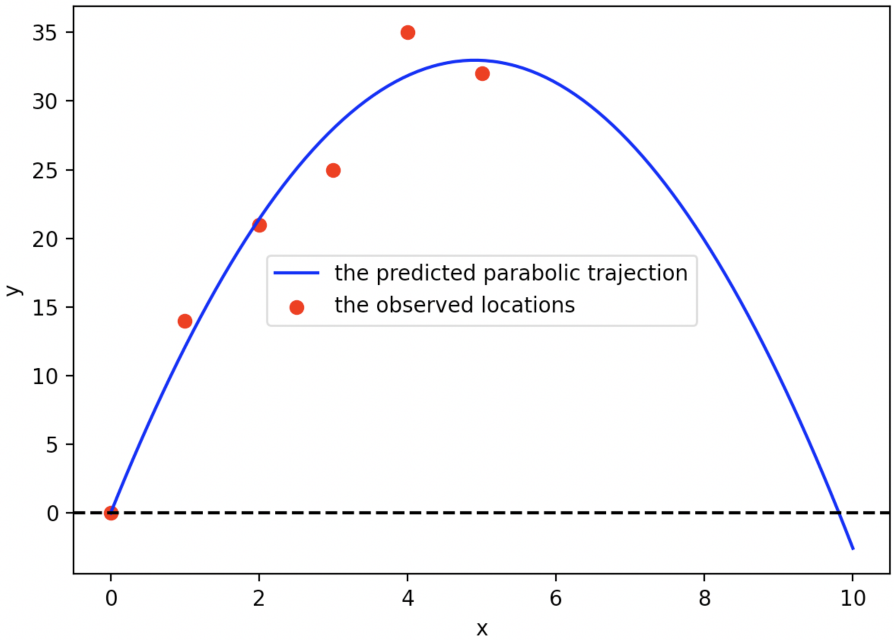


Figure 6: regression

- The parabolic trajection must pass through the origin (0,0), so, in the equation: $y = a_2 x^2 + a_1 x + a_0$, $a_0$ must be 0. So, we have,

$$(x^2, x) = \bar{x}'$$
$$(a^2, a^1) = \bar{w}$$

Then, the regression is:

$$y = \bar{w}^T \bar{x}'$$

Using the above regression, we can get the equation of the predicted parabolic trajection:

$$y = -1.36956522x^2 + 13.43913043x \tag{1}$$

- According to (1), let y = 0, then, the distance from the cannon where the cannonball is expected to fall is 9.81269839