Machine Learning
Home Assignment 1

Mao Xiqing (tls868)

1 Make Your Own

1. What profile information would you collect and what would be the sample space $\mathcal{X}$?
   The sample space $\mathcal{X}$ could be grades in prior courses, grades of assignments, attendance, previous education areas...

2. What would be the label space $\mathcal{Y}$?
   The label space could be pass/failed ($\mathcal{Y} = \{0, 1\}$), or just like Danish 7-point grading scale from -3 to 12 ($\mathcal{Y} = \{-3, 0, 2, 4, 7, 10, 12\}$).

3. How would you define the loss function $\ell(Y', Y)$?
   The loss function is defined as $\ell(Y', Y) = (Y' - Y)^2$, where Y' is prediction grades.

4. How would you evaluate the performance of your algorithm? (In terms of the loss function you have defined earlier.)
   Input new data (it is different from training set but come from the same distribution) as test set to compute $\ell(Y', Y)$ again. We expect the loss function to be very small (the predicted value is closer to the true value), which means better performance.

5. 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?
   Maybe this algorithm runs very slowly, I can optimise it in following ways:

   (a) Use better data structures and algorithms to reduce the need for loops.

   (b) Use space to save time.

   (c) Vectorization.

   Besides, the model may also overfit, so training set must be i.i.d.

2 Illustration of Markov's and Chebyshev's Inequalities

Source code for this question can be found in the notebook **`MCInequalities.ipynb`**.

2.1

1. Make 1,000,000 repetitions of the experiment of drawing 20 i.i.d. Bernoulli random variables $X_1, ...X_{20}$ (20 coins) with bias $\frac{1}{2}$.
   This case could be regard as 20 times Bernoulli trials with parameter $\frac{1}{2}$, which obeys binomial distribution, so I implemented this event by the following code:

```
k = 1000000
x = np.random.binomial(n=20,p=0.5,size=k)
mean = x / 20
```

2. Plot of the empirical frequency of observing $\frac{1}{20}\sum_{i=1}^{20} X_i \geq \alpha$ for $\alpha \in \{0.5, 0.65, ..., 1\}$ is shown on Figure 1.

3. Explain why the above granularity of α is sufficient. I.e., why, for example,taking α= 0.51 will not provide any extra information about the experiment.
   Because for probability $\in [0.5 \sim 1]$, it is enough to plot a continuous curve by granularity = 0.05. Taking granularity = 0.01 will only increase amount of calculation, except for making the curve look smoother.

4. Plot of the Markov's bound on $\mathbb{P}(\frac{1}{20}\sum_{i=1}^{20} X_i \geq \alpha)$ is shown on Figure 1.

5. Plot of the Chebyshev's bound on $\mathbb{P}(\frac{1}{20}\sum_{i=1}^{20} X_i \geq \alpha)$ is shown on Figure 1.

6. Compare the three plots.
   It is obvious that Markov's inequality gives us a pretty loose bound, which may not really useful in this case, while Chebyshev's inequality is much tighter than Markov's. However, when ϵ is approximate to bias$(0.5 \sim 0.6)$, the probability will be exceeds 1. This is true for equation itself, but it has no practical significance.

7. For $\alpha = 1$ and $\alpha = 0.95$ calculate the exact probability $\mathbb{P}(\frac{1}{20}\sum_{i=1}^{20} X_i \geq \alpha)$.
   For $\alpha = 1, k = \alpha \times 20 = 20, p = 0.5$,

$$\mathbb{P}(\frac{1}{20}\sum_{i=1}^{20} X_i \geq \alpha) = \binom{20}{k} p^k (1-p)^{20-k} = \binom{20}{20} p^{20}(1-p)^0 \approx 9.5367 \times 10^{-7} \quad (1)$$

   For $\alpha = 0.95, k = 19, p = 0.5$,

$$\mathbb{P}(\frac{1}{20}\sum_{i=1}^{20} X_i \geq \alpha) = \binom{20}{k} p^k (1-p)^{20-k} = \binom{20}{19} p^{19}(1-p)^1 \approx 1.9073 \times 10^{-5} \quad (2)$$

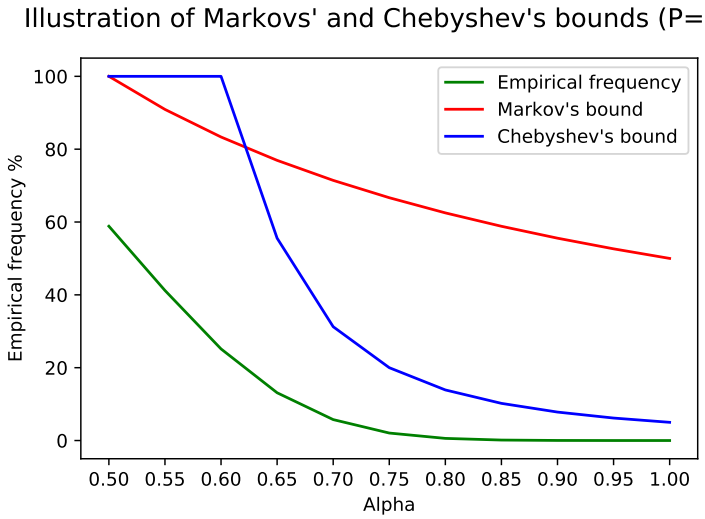

Figure 1: This figure shows Markov's and Chebyshev's bound of the same experiment and compare them with empirical frequency.

2.2

The plot of the same question but with bias 0.1 (i.e., $\mathbb{E}[X_1] = 0.1$) and $\alpha \in \{0.1, 0.15,...,1\}$ is shown on Figure 2.

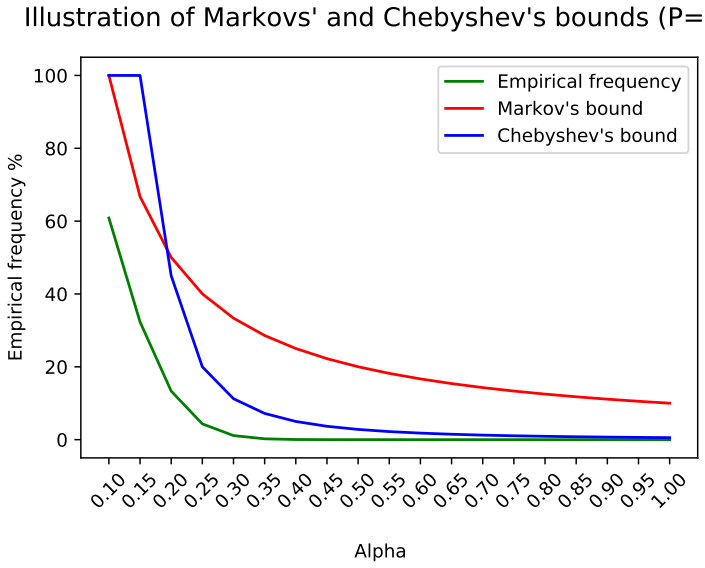

Figure 2: This figure shows Markov's and Chebyshev's bound of the same experiment but with a smaller ϵ, and compare them with empirical frequency.

For $p = 0.1$, when $\alpha = 1$ and 0.95, the exact probability of $\mathbb{P}(\frac{1}{20}\sum_{i=1}^{20} X_i \geq \alpha)$ is 1×10^{-20} and 1.8×10^{-8} respectively.

2.3

Discussion: When bias=0.1 and α starting from 0.1, curves of those two bounds and empirical frequency drop very fast, and both of two bounds seem to be tighter than when bias=0.5 on Figure 1, but Markov's bounds is still looser than Chebyshev's, which means the estimation by Chebyshev's inequality is more accurate. The Chebyshev's bound is still exceeds 1 when ϵ approach to bias ($0.1 \sim 0.15$). This means inequality would be meaningless if ϵ is smaller than expectation.

Overall, Chebyshev's bound is tighter than Markov's, and it will greater than 1 when ϵ is close to bias.

3 Tightness of Markov's Inequality

Let X be a random variable

$$X = \begin{cases} k \geq 1, & P = \frac{1}{k}, \\ 0, & P = 1 - \frac{1}{k}, \end{cases} \tag{3}$$

and let ε^*=k, now Markov's Inequality become

$$\mathbb{P}(X \geq k) = \frac{\mathbb{E}(X)}{k} = \frac{1}{k} \tag{4}$$

The inequality is tight in the sense that equality can hold for a given X, as X only takes values 0 and k.

4 Digits Classification with Nearest Neighbors

Source code for this question can be found in the notebook **`knn.ipynb`**.

After reading dataset by `numpy.loadtxt`, I use `numpy.reshape` to make dataset more readable. Then I merged dataset and label with numpy function `numpy.hstack`, and I use pandas' function to select rows form dataset and then split validation set from training set:

```
trainset = labelledtrain[:round(labelledtrain.shape[0]*0.8)] #first 80%
valset = labelledtrain.iloc[-round(labelledtrain.shape[0]*0.2):] #last 20%
```

I picked the images of the corresponding label by following code:

```
def genlabel(dataset,dg1,dg2):
    digital = dataset[dataset[0]==dg1]
    digital = digital.append(dataset[dataset[0]==dg2])
    label = digital[0] #label
    label = np.array(label)
    return label

def gendataset(dataset,dg1,dg2):
    digital = dataset[dataset[0]==dg1]
    digital = digital.append(dataset[dataset[0]==dg2])
    digital_train = digital.drop([0],axis=1)
    digital_train = digital_train.to_numpy()
    return digital_train
```

I implemented Nearest Neighbors learning algorithm by following code:

```
def knn(newX, dataset, labels, K):
    sqdiff = (newX - dataset)**2
    d = sqdiff.sum(axis=1)
    euc_d = d**0.5
    sort = d.argsort() # sort and get index
    nearlabels = [labels[i] for i in sort[0 : K]]  # nearest label
    label = collections.Counter(nearlabels).most_common(1)[0][0] # majority
   vote of K, select the first one
    return label
```

I use the following code to calculate the error rate:

```
##########  errcount ##########
def err(inX, data, labels1, labels2):
    errcount = []
    for k in range(1,34,2):
        error = 0
        for x in range(inX.shape[0]):
            answer = knn(inX[x], data, labels1, k)
            if answer != labels2[x]:
            error += 1
        errcount.append(error)
    return errcount

##########  errpercentage ##########
def errper(error, tset):
    error2 = pd.DataFrame(data = error, columns = ["Error rate %"])
    errpercent = error2 / float(tset.shape[0]) *100
    return errpercent
```

4.1 Comparison results on validation error and test error

4.1.1 Result on digits '0' and '1'.

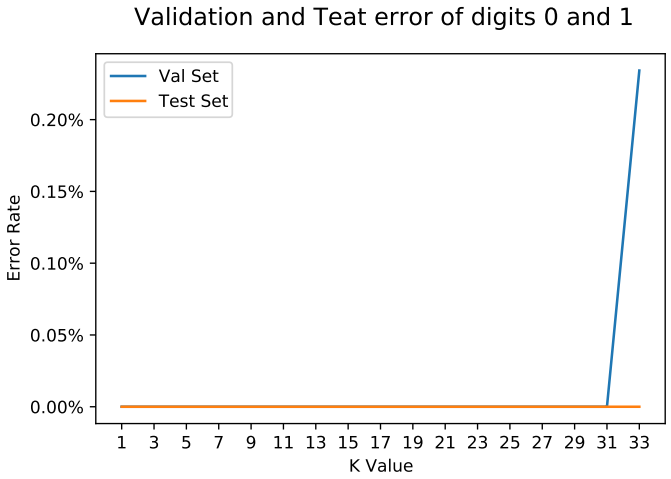

Figure 3: This figure shows the error rate of validation set and test set for digits '0' and '1'.

4.1.2 Result on digits '0' and '8'.

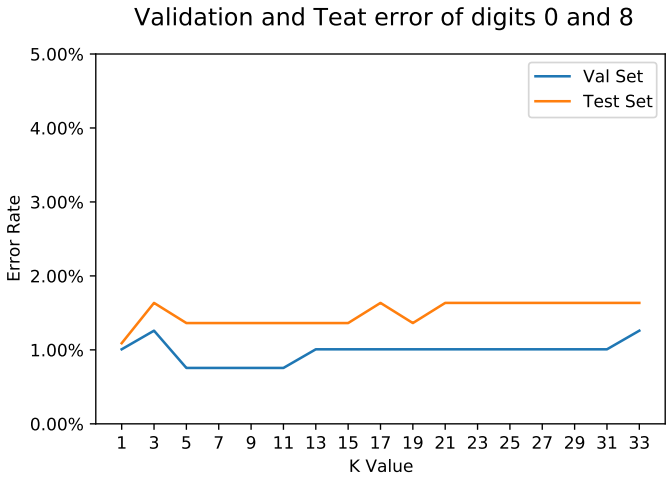

Figure 4: This figure shows the error rate of validation set and test set for digits '0' and '8'.

4.1.3 Result on digits '5' and '6'.

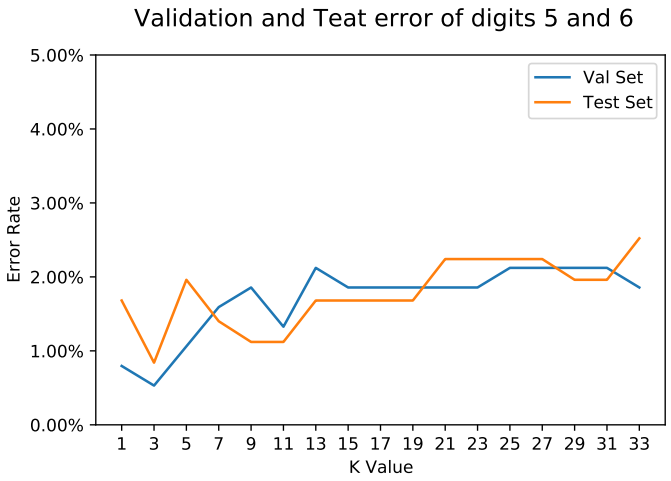

Figure 5: This figure shows the error rate of validation set and test set for digits '5' and '6'.

4.2 Discussion

1. How well does the validation error match the test error?
   They match well to some extent, and they have similar trends. For example, in Figure 3, both of them are simple horizontal lines(except when $K > 31$), and in Figure 5, both of them have the same slightly rising trend.

2. How closely does the best value of K according to the validation error match the best value of K according to the test error?
   They match very closely. Even the best value of K on validation and test set for digits '0','1' and '5','6' is exactly the same in Figure 3 and Figure 5 (K=1 and 3 respectively). Although the best value of K on validation and test set for digits '0' and '8' are different, they are also very close (K=5 and 1 respectively). See the table below for summary.

Table 1: the best value of K for validation and test set

Digital / Dataset	**0,1**	**0,8**	**5,6**
Validation Set	1∼31	5∼11	3
Test set	1∼33	1	3

3. How the validation and test errors behave as a function of K?
   Overall, it is not wise to make k value very small because because it may lead to overfitting, especially when K = 1. The error rate will increase as the K value increases after best the k value. These behavior can only be observed on Figure 5 because task on distinguishing '0','1' and '0','8' are relatively simple.

4. Does the best value of K depend on the diculty of the task and how?
   Yes. Figure 3 shows almost two exactly same lines at 0.00% of validation and test set for digits '0' and '1', and the best k value is just 1, which means '0' and '1' are very easy to tell. However, in Figure 4 and Figure 5, the error rates of '0','8' and '5', '6' are both higher than those of digits '0' and '1', which fluctuate between around $0.5\% \sim 1.5\%$ and $0.5\% \sim 2.4\%$ respectively, indicating that '5' and '6' are more difficult to tell than '0' and '1', and '0' and '8' is somewhere in between.

5 Nearest Neighbors for Multiclass Classification

Algorithm K Nearest Neighbors for Multiclass Classification with $\mathcal{Y} = \{0, 1, ..., 9\}$
1. **Input**: Labelled training set (X, Y): $\{(x_1, y_1), ..., (x_n, y_n)\}$ and unclassified data x'.
2. Calculate the distances $d_i = d(x_i, x')$.
3. Sort each d_i in ascending order and get their index: `argsort()`
4. Find k nearest labels: `nearlabels = [`$y[i]$ `for i in index[0 : K]]`
5. Label of x' is the majority one in `nearlabels`

6 Linear regression

6.1 Implementation of linear regression

I implemented linear regression using the pseudoinverse as introduced in the lecture. My implementation can be found in the notebook **`Linear Regression.ipynb`**.

I computed the parameters of a regression model in the following way:

```
array1 = np.ones((28,1)) # gen matrix with 1
x1 = np.hstack((x,array1)) # add 1 to matrix x
def reg(x, y):
    xstar = np.dot(np.linalg.pinv(np.dot(x.T,x)),x.T) #pseudo-inverse
    wmin = np.dot(xstar,y)
    return wmin
```

6.2 Building the first model

Figure 6 shows the loaded data. To fit a model of the form

$$h(x) = \exp(ax + b)$$

with parameters $a, b \in \mathbb{R}$, I transformed the output data (the y values) by applying the natural logarithm first:

```
yln = np.log(y) # y -> lny
```

Then I use the algorithm described in Section 6.1 to model $\ln y$ by $ax + b$.

```
lg = reg(x1,yln)
w = lg[0][0]
b = lg[1][0]
yhat = w * x + b
```

I found $a = 0.2591282395640716$ and $b = 0.03147246971447598$.

I calculated MSE in the follow way:

```
def mse(x, y):
    n = x.shape[0]
    MSErr = np.sum((x - y)**2)/n
    return MSErr
mse1 = mse(yln,yhat)
```

The MSE of this model is 0.29853492489386185.

6.3 Plot the data and the regression line

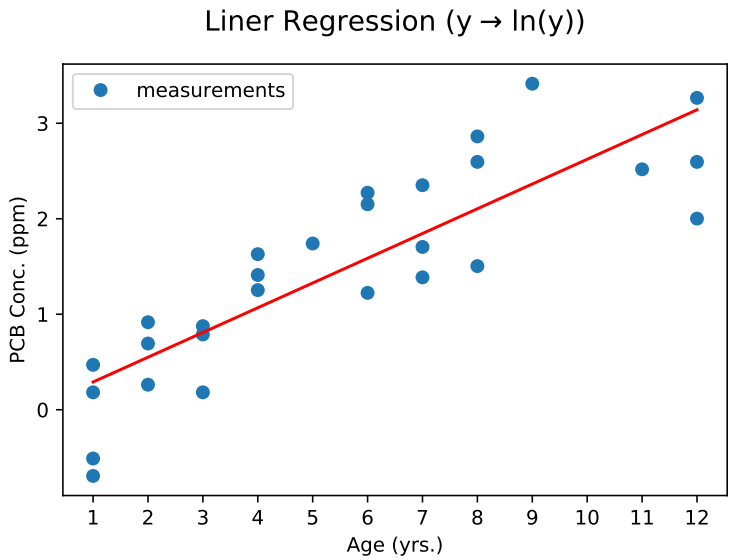

Figure 6: This figure shows the logarithm of PCB concentration vs. the age of the fish in years.

6.4 Compute the coefficient of determination

I computed R^2 in the follow way:

```
def computeR2(y, yhat):
    n = x.shape[0]
    sse = np.sum((y - yhat) ** 2)
    sst = np.sum((y - np.mean(y)) ** 2)
    R2 = 1 - sse / sst
    return R2
Rsq1 = computeR2(yln, yhat)
```

The R^2 of this model is 0.7313915439623444.

Discussion: R^2 measures how well the predicted values fits the true values. In a ideal situation, $R^2 = 1$ means that all predicted values equal to the true values. As R^2 decreases, the correlation between the predicted values and the true values of the model also decrease. If $R^2 = 0$, one possibility could be that all predicted values are equal to the average of true values, meaning that the prediction results of the model is pretty poor. There is no lower limit for the minimum value of the R^2, because the prediction can be extremely worse. Therefore, $R^2 \in (-\infty, 1]$.

6.5 Build a non-linear model

The non-linear model

$$h(x) = \exp(a\sqrt{x} + b)$$

is built by transforming the input data (they x values) in the following way:

```
xsqrt = np.sqrt(x) # x -> sqrt(x)
```

The MSE and R^2 were computed by the same way which mentioned above. The MSE of this non-linear model is 0.23772496260048076, and R^2 is 0.7861056451320498. This R^2 is slightly larger than the R^2 of the previous model(0.7313915439623444), indicating that the non-linear model has a better prediction results.

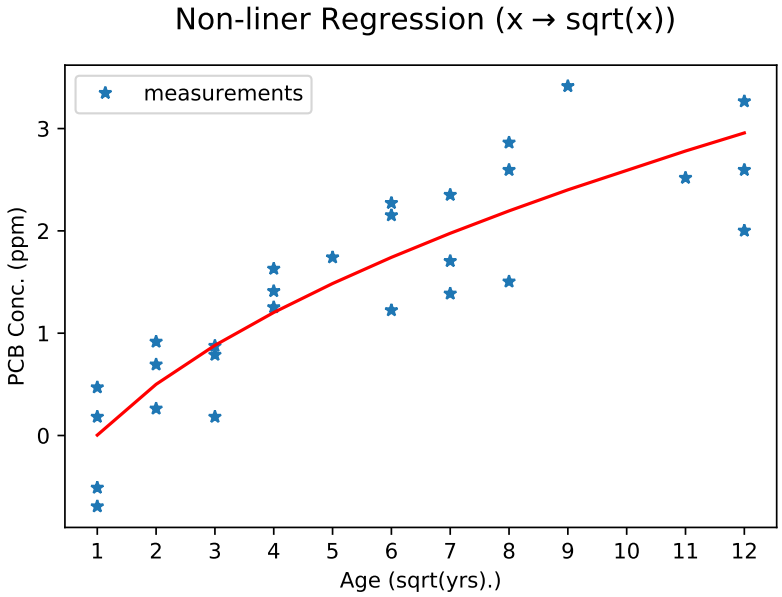

Figure 7: This figure shows the logarithm of PCB concentration vs. the age of the fish in years.