Machine Learning A (2025)
Home Assignment 1

Simon Henriksen, xwj436

Contents

1 Preprocessing

1.1

1. Based on the euclidean distance measure the distance from how the data is represented in the matrix

```
import numpy as np

matrix = np.array([[47, 35], [22, 40]])
matrix_scaled = np.array([[47, 35000], [22, 40000]])
C = np.array( [21,36])
C_scaled = np.array([21,36000])

d = np.linalg.norm(matrix - C, axis=1)
d_scaled = np.linalg.norm(matrix_scaled - C_scaled, axis=1)
print(d)
print(d_scaled)

-----------------------------------
Output

[26.01922366  4.12310563]
[1000.3379429 4000.000125 ]
```

According to the measurement of the original data, C shouldn't be granted credit due to B being it's 1-NN.

When the y feature is scaled to thousand dollars, the distance grow much faster and dominating the x_i feature. This results in A being the closest 1-NN to C saying C should be granted credit.

1.2

1. The mean vector is given by

$$\bar{x} = \frac{1}{N} X^\top \mathbf{1},$$

When we subtract the mean, we can write

$$Z = X - \mathbf{1}\bar{x}^\top.$$

Insert $\bar{x}^\top = \frac{1}{N}\mathbf{1}^\top X$:

$$Z = X - \mathbf{1}\left(\frac{1}{N}\mathbf{1}^\top X\right).$$

This can be rewritten by moving $1/N$ outside the parenthesis is the same as saying:

$$Z = X - \tfrac{1}{N}\mathbf{1}\mathbf{1}^\top X.$$

$X = I * X$ and then we've a common factor in X which we factorize out so it ends up as the desired expression.

$$Z = \Big(I - \tfrac{1}{N}\mathbf{1}\mathbf{1}^\top\Big) X = \gamma X.$$

2. X

1.3

1. We first note that

$$\mathrm{Var}(\hat{x}_1) = 1,$$

so $\hat{x}_1$ has unit variance and is equal to the true x_1.

Next,

$$\mathrm{Var}(\hat{x}_2) = (1 - \epsilon^2) \cdot 1 + \epsilon^2 \cdot 1 = 1,$$

since both $\hat{x}_1$ and $\hat{x}_2$ have unit variance, and $x_2 = \sqrt{1-\epsilon^2}\,\hat{x}_1 + \epsilon \hat{x}_2$.

To compute the covariance, we plug in the coefficients

$$a = 1, \quad b = 0, \quad c = \sqrt{1-\epsilon^2}, \quad d = \epsilon,$$

with $V = \hat{x}_1$ and $W = \hat{x}_2$.

Then

$$\mathrm{Cov}(\hat{x}_1, x_2) = \mathrm{Cov}\big(\hat{x}_1,\, cV + dW\big).$$

The cross-term with dW vanishes, because V and W are independent with zero mean. Thus we obtain

$$\mathrm{Cov}(\hat{x}_1, x_2) = c\,\mathrm{Var}(\hat{x}_1) = \sqrt{1-\epsilon^2} \cdot 1 = \sqrt{1-\epsilon^2}.$$

2. X

3. I plug the correlated x into the linear function

$$f(x) = w_1 x_1 + w_2 x_2,$$

which gives

$$f(x) = w_1 \hat{x}_1 + w_2\big(\sqrt{1-\epsilon^2}\,\hat{x}_1 + \epsilon \hat{x}_2\big).$$

We use the distributive law on the second term to multiply w_2 with both parts, and then factorize with respect to $\hat{x}_1$. This yields

$$f(x) = \big(w_1 + w_2\sqrt{1-\epsilon^2}\big)\hat{x}_1 + (w_2\epsilon)\hat{x}_2.$$

We want to determine the minimum weights, since we are asked to find the minimum value of C according to the weights. The target function is

$$f(x) = \hat{x}_1 + \hat{x}_2,$$

so we solve for w_1 and w_2 such that our model matches this target.

From the second condition:

$$w_2 \epsilon = 1 \quad \Rightarrow \quad w_2 = \frac{1}{\epsilon}.$$

From the first condition:

$$w_1 + w_2\sqrt{1-\epsilon^2} = 1 \quad \Rightarrow \quad w_1 = 1 - \frac{\sqrt{1-\epsilon^2}}{\epsilon}$$

The expressions for w_1 and w_2 above are the ones consistent with minimizing C.

4. As ϵ converges towards zero, w_1 and w_2 diverge towards $-\infty$ and $+\infty$, respectively. Since x_2 is highly correlated with x_1 (it contains $\hat{x}_1$ inside itself), the contribution from $\hat{x}_1$ will dominate, and the linear model will extract almost no information about $\hat{x}_2$ through x_2. To avoid this loss of information, the value of C must increase, allowing for larger weights in order to recover the signal from $\hat{x}_2$.

2 Hoefddings bound

2.1

1.
```
# Calculating E[S]
EX = (-2 + 0.8 + 1) / 3
ES = n_hoeffding * EX

# deducting ES from epsilon
epsilon = 22 - ES

# Hoeffding bound for generic range [a, b]
def hoeffding_inequality (epsilon, n, range):
    return math.exp(-2 * (epsilon ** 2) / (n * (range ** 2)))

# Calculate Hoeffding bound
bound = hoeffding_inequality(epsilon, n_hoeffding, width)
print(f"Hoeffding bound: {bound:.6f}")

-------------------------------
Output: Hoeffding bound: 0.161029
```

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

3.1

1.

```
samples = np.random.binomial(1, p, size=(n_reps, n))
means = samples.mean(axis=1)

# alpha vaerdier
alpha_values = np.arange(0.5, 1.01, 0.05)

# beregn empirisk frekvens
freqs = [(means >= alpha_values).mean() for alpha_values in
alpha_values]
```

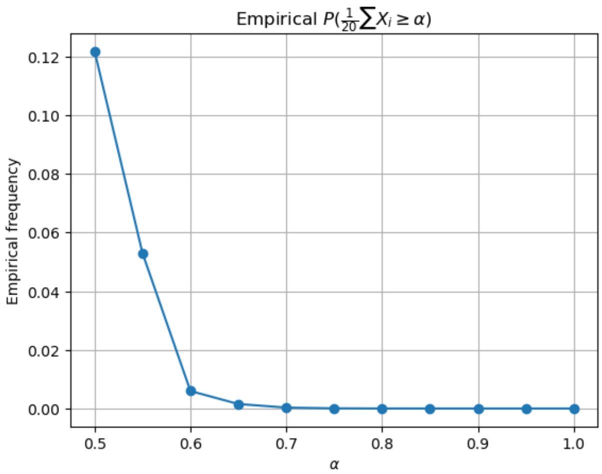

Figure 1: Emperical risk inequality for Bernoulli sample

The emperical probalility decreaces as alpha increases. Saying, the probability of the sample values being greater or equal to alpha drops drasticly from alpha = 0.5 to zero probability when alpha = 0.70 according to the empercial probability.

2. As we're having an discrete number in both the denorminator and numerator it implies that the result will be in the support of $0.05, 0.10, 0.15...0, 95$. Therefore, changing alpha to $0, 51$ instead of $0, 50$ would not make any difference as the 0.01 increase would not have any affect as it doenst cross any value in the support.

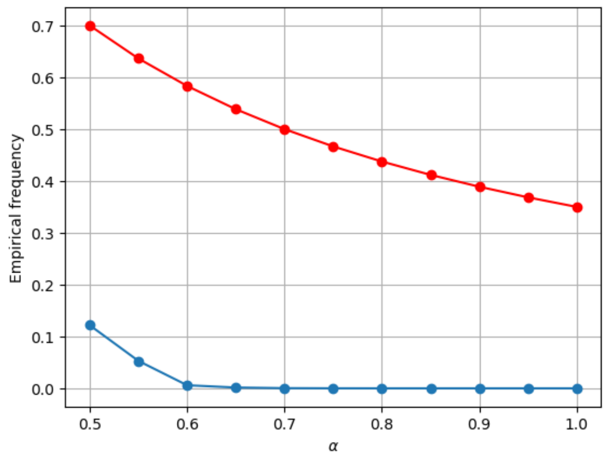

Figure 2: Emperical (blue) and Markovs(red)

3.

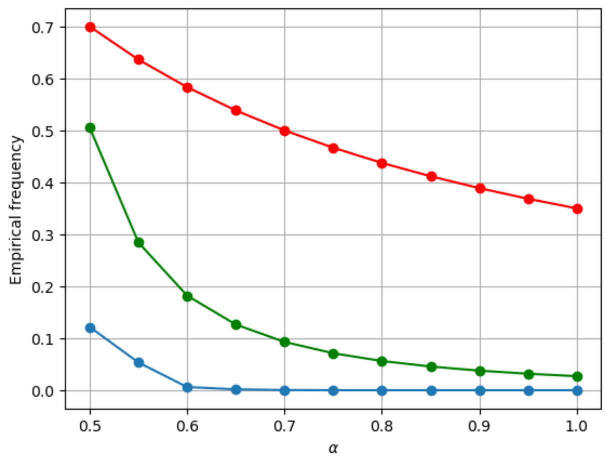

Figure 3: Emperical(blue), Markovs(red), Chebyshev (green)

4.

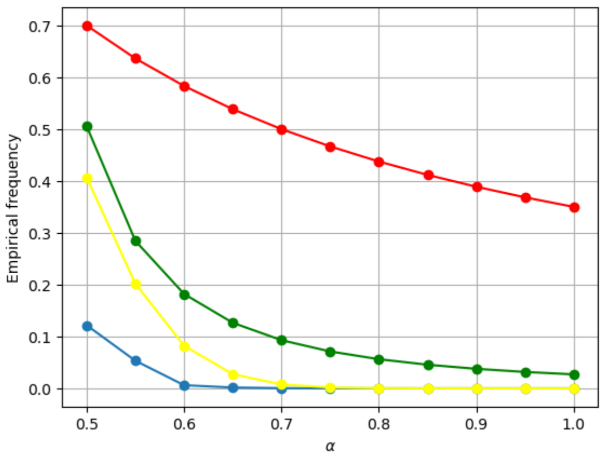

Figure 4: Emperical(blue), Markovs(red), Chebyshev (green), Hoeffding(yellow)

5.

6. Plot comment: Of the three teoretical bounds (markov, chebyshev and Hoeffding), the upper bound are much higher for red (markov) especially as it only demands non-negative random variables and the expected value. The green (chebyshevs) uses the variance which yields a tighter upperbound. Yellow (hoeffding) proves to have the tightest upperbound due to the fact of independes and use of range. This makes it decay exponentially which is much faster than 1/n for chebyshev and 1/a for markov.

7. We perform some mathematics on the expression. Firstly, we consider

$$P\left(\tfrac{S}{N} \geq \alpha\right).$$

Multiplying both sides of the inequality by N isolates S:

$$P(S \geq N\alpha).$$

Since $N = 20$, we substitute the two values of α. This gives

$$P(S \geq 20) \quad \text{and} \quad P(S \geq 19).$$

We now compute these probabilities using the binomial distribution:

$$P(S = 20) = \binom{20}{20}(0.35)^{20}(0.65)^{0} = (0.35)^{20},$$

and

$$P(S \geq 19) = \binom{20}{19}(0.35)^{19}(0.65)^1 + (0.35)^{20}.$$

Notice that the probability for $P(S \geq 20)$ is included as part of the expression for $P(S \geq 19)$.

To compute these numerically, I used the `scipy` module in Python. The results are

```
from scipy.stats import binom

p = 0.35
n = 20

prob_alpha1 = binom.sf(19, n, p)
prob_alpha095 = binom.sf(18, n, p)

print("P(S/20 >= 1) =", prob_alpha1)
print("P(S/20 >= 0.95) =", prob_alpha095)

Output
-------------------------
P(S/20 >= 1) = 7.609583501588048e-10
P(S/20 >= 0.95) = 2.9025125641771556e-08
```

This means that obtaining all 20 Bernoulli trials as successes is virtually impossible. Achieving 19 out of 20 successes is relatively more likely, but still extremely unlikely.

3.2

1. Optional

4 Experiment Design

4.1

1. As stated, we're looking for a hypothesis within 20 different models. To ensure the picking the model with the smallest difference in loss from the real loss distribution we choose the union bound inequality.

$$\Pr\left(L(h_i) - \hat{L}_n(h_i) > \varepsilon\right) \leq e^{-2n\varepsilon^2}.$$

Applying the union bound over 20 models gives

$$20\, e^{-2n\varepsilon^2}.$$

To figure out the correct n, we've to solve $20\,e^{-2n\varepsilon^2} \leq \delta$, which yields

$$n \;\geq\; \frac{1}{2\varepsilon^2}\,\ln\!\left(\frac{20}{\delta}\right).$$

Our epsilon and delta are both 0.05 as we want the value to be with with probability of 0.95.

$$n \geq \frac{1}{2(0.05)^2}\ln\!\left(\frac{20}{0.05}\right) = \frac{1}{0.005}\,\ln(400) = 200 \cdot 5.99146 \approx 1198.29 \;\Rightarrow\; n = 1199.$$

This means 1199 patients have to be kept out of the training data to ensure the model hypothesis picked in the union-bound inequality doesn't underestimate the true loss by more than 0.05.

2. To determine m, m has to be isolated according to the same inequality, the union-bound. Therefore we solve for m, $m\,e^{-2n\varepsilon^2} \leq \delta$, which yields

$$m \leq \delta e^{2n\varepsilon^2}.$$

Epsilon and delta obtain the same values as in the previous task. We calculate our m:

$$\begin{aligned}
2n\varepsilon^2 &= 2 \cdot 1140 \cdot 0.05^2 = 2280 \cdot 0.0025 = 5.7,\\
e^{5.7} &\approx 298.8674,\\
m_{\max} &= \delta\,e^{2n\varepsilon^2} = 0.05 \cdot 298.8674 \approx 14.94.
\end{aligned}$$

This means, the maximum m, number of teams, is 14 with a 0.95 probability.