Machine Learning A (2025)
Home Assignment 1, Reference Solution

Submission deadline: 11 September 2025, 22:00

1 Make Your Own, first example (10 points)

[By Ole, 2024, edited by Monika 2025]

1.1

An example of profile information that could be used as features for a classifier is

- Years studied,
- Grades of all the prior courses,
- Number of completed relevant courses prior to MLA

The sample space, $\mathcal{X}$, would be

$$\mathcal{X} = \mathbb{N} \times \{-3, 0, 2, 4, 7, 10, 12\} \times \mathbb{N}$$

Note that the correct grade for 0 and 2 is 00 and 02.

1.2

The label space includes all the possible final grades a student can receive in MLA. Since we use the Danish grading system, this would be $\mathcal{Y} = \{-3, 0, 2, 4, 7, 10, 12\}$.

1.3

You can choose either the *squared loss*

$$\ell(y', y) = (y' - y)^2,$$

or *absolute loss*

$$\ell(y', y) = |y' - y|$$

where y' is the predicted label and y is the true label.

However, it would not be the best choice to choose *zero-one loss*.

$$\ell(y', y) = \mathbb{I}(y' \neq y) = \begin{cases} 1, & \text{if } y' \neq y \\ 0, & \text{otherwise} \end{cases}$$

as this is used for binary classification. Additionally, it is not the most suitable loss measure for this problem to use *zero-one loss*, since we would like a predicted label of 2 to have higher loss than a predicted label of 7 if the correct label is 10.

1.4

The feature space is contained within $\mathbb{R}^d$ and we are therefore able to use the euclidean distance as the distance function

$$d(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\| = \sqrt{\sum_{j=1}^{d} (x_j - x'_j)^2} = \sqrt{(\mathbf{x} - \mathbf{x}')^T (\mathbf{x} - \mathbf{x}')}$$

1.5

You evaluate the performance of the algorithm by computing the loss using the loss function stated above on a validation set which is of the same distribution as the training set.

1.6

A potential issue could be that a student cannot be represented by this feature space. As it would be the case if a student enrolls in this course as her first one and therefore has no prior grades. However, this might be handled by assigning the student with the average grade of other students.

Another potential issue could be that the distribution of student grades changes, so that our training set is no longer of the same distribution as the one we apply the algorithm on. This could be the case if the course was opened up for students from other study programs. The best solution might be to start over and repeat the process of obtaining data and training the algorithm.

Another reason could be that if the students were given their grade at the beginning of the course, they would stop studying, which would change the knowledge level (and thus the grades if the students would be tested) at the end of the course. A solution to this problem could be to apply selective grading.

Make Your Own, second example (10 points)

[By Frederik, 2023, edited by Monika 2025]

1. **Profile Information and Sample Space X:**
   - Profile info: Grades in prior courses, study program.
   - Sample space $\mathcal{X} = \{-3, 0, 2, 4, 7, 10, 12\} \times \{-3, 0, 2, 4, 7, 10, 12\} \times \{\text{"Computer Science"}, \text{"Math"}, \text{"Chemistry"}, \text{"Physics"}, \text{"Other"}\}$
2. **Label Space Y:**
   - Label space: Final grade of the student in the Machine Learning course $\mathcal{Y} = \{-3, 0, 2, 4, 7, 10, 12\}$.
3. **Loss Function $\ell(y', y)$:**
   - Loss function $\ell(y', y)$: Mean squared error (MSE) between predicted grade y' and actual grade y.
4. **Distance Measure $d(x, x')$:**
   - Distance measure $d(x, x')$: Euclidean distance between feature vectors x and x'. For the study program part of the sample space we add an additional fixed distance $\delta_{i,j}$, where i and j are study programs.
     If the two students have the same program, then $\delta_{i,j} = 1$. Otherwise, $\delta_{i,j} = 0$.
5. **Algorithm Evaluation:**
   - Evaluate algorithm using mean squared error (MSE) on a *validation set* to measure prediction accuracy.
6. **Deployment Challenges and Solutions:**
   - Challenges: New data distribution, privacy concerns.
   - Solutions: Regularly update the algorithm with new data, ensure data privacy through anonymization and encryption.

2 Digits Classification with K Nearest Neighbors (40 points)

[by Michael, 2024]

2.1 Task #1

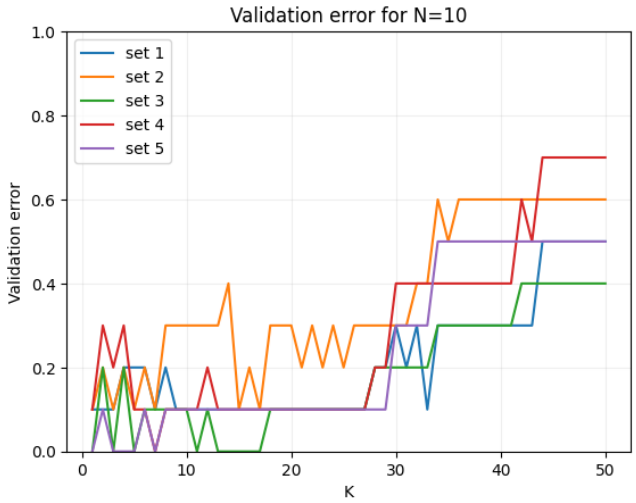
The function knn calculates the pairwise distance between each training point and test point. Thereafter the training points are sorted by distance, it is determined whether the point influences a decision towards a correct/incorrect prediction represented as either -1 or 1. By summing these together cumulatively the sign of each value k decides if it is correct or incorrect decision. In cases where k is even and a tie occurs, the prediction is counted towards the incorrect. This can be circumvented by f.x weighting the k points involved by distance to the test point. The Validation error and variances of the model is measured and plotted.

Considering these plots we see that validation error tends to decrease as n increases. The variance also decreases as n increase. This is desirable, while not a measure for increased model quality, more data points usually increase chances of representing the underlying distribution because the chances of covering different test cases increases. Using larger validation set for testing should therefore ensue results that better represent the actual loss.

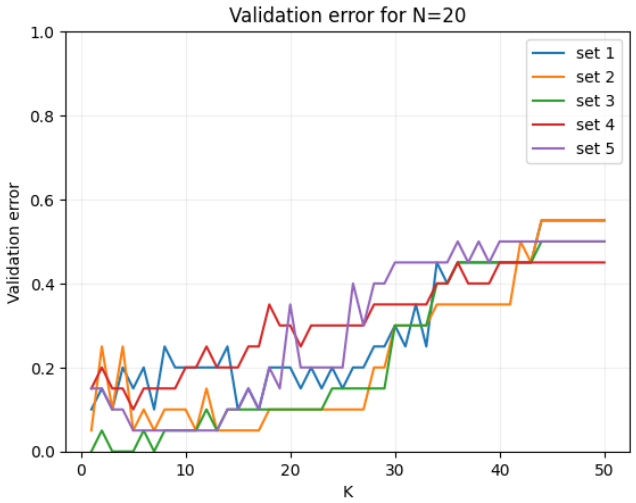
On the other hand increasing K generally lowers prediction accuracy. While you could argue similarly to increasing validation set, that higher k would cover more cases and be beneficial this does not appear to be the case. There is a trade off between weighting more representative points and representation from more points. By inspecting the 4 graphs this trade of seems to be maximized at around k=5, suggesting that few points is the most beneficial. Worth noting is that the variance is slightly higher for lower values which is less desirable, though this is mostly for lower values of n, which as discussed are less representative of the actual loss.

```
def knn(training_points, training_labels,test_points, test_labels)
    :
  #Storing lengths of train and val data
  m = training_labels.size
  n = test_labels.size
  #creating the 1 vectors that will be used and reshaping to a
    column vector
  Im = np.ones(m).reshape(m,-1) #YS: It should be the same
    argument in reshape here and in In
  In = np.ones(n).reshape(n,1)
  #Transposing getting to getcollection of column vectors that
    each represent a point
```

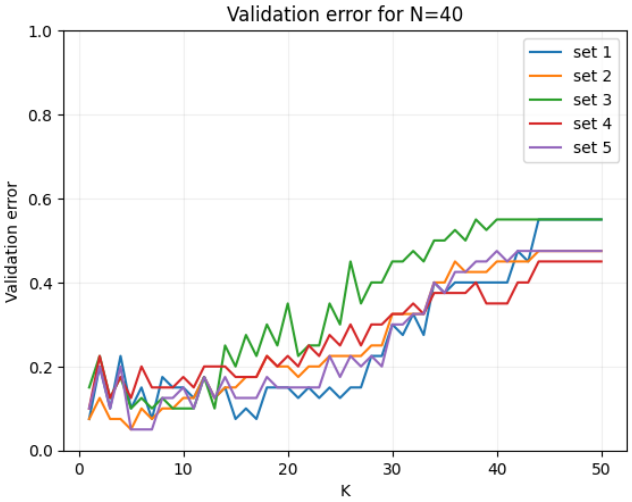
```
training_points = training_points.T # (d,m)
test_points = test_points.T # (d,n)

#Vectorized calculations of distances getting matrix D of
 pairwise distances
#Calculating first term of the equation
term1 = np.matmul(training_points.T, training_points) #YS: If we
  release the code, it would be better to have more meaningful
 names than "term1", etc.
term11 = np.diag(term1).reshape(m,1)
term12 = np.matmul(term11, In.T)

#second term
term2 = np.matmul(training_points.T, test_points)

#third term
term3 = np.matmul((test_points).T, test_points)
term31 = np.diag(term3).reshape(n,1)
term32 = np.matmul(Im, term31.T)
D = term12 - 2*term2 + term32 # (m, n)

#sorting indices of training points by distance to the test
 points
sorted_indices = np.argsort(D, axis=0)
#getting the labels of the sorted points
sorted_labels = training_labels[sorted_indices]
#Determining wheter each training point matches the correct
 label of each test point.
correlations = sorted_labels== test_labels.reshape(-1, n)
#converting from true or false to 1 or -1
correlations = np.where(correlations, 1, -1)
#Calling cumsum to sum up the training points result in a
 correct of incorrect classification
#As it is cumulative the prediction for all values of K are
 calculated
cumsum_correlation = np.cumsum(correlations, axis = 0)

#getting the validation error by calculating the mean with the
 incorrect classifications
validation_error = np.mean(cumsum_correlation <= 0, axis=1)
return validation_error
```



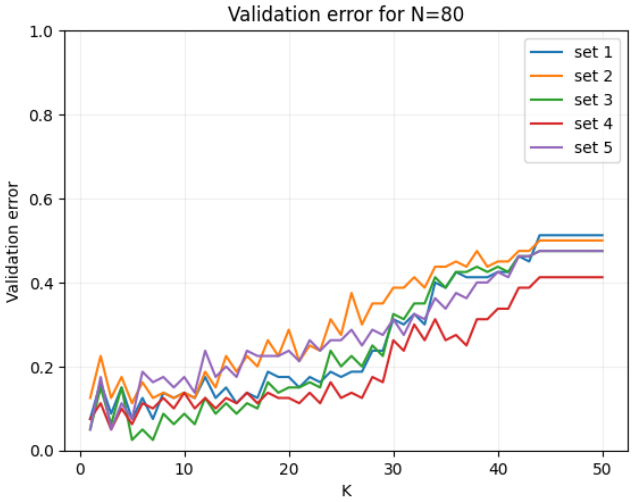
(a) Validation error n=10



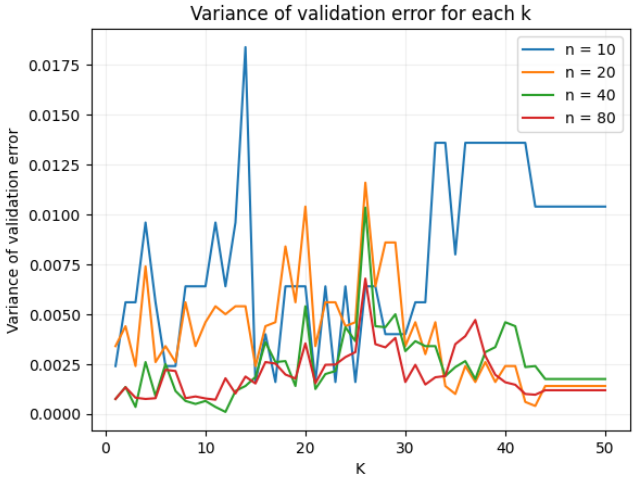
(b) Validation error n=20



(c) Validation error n=40



(d) Validation error n=80



(e) Variance of validation error

3 Linear Regression (50 points)

[by Hippolyte, 2024]

3.1 Implement LinReg (5 points)

Below, we present an implementation of the linear regression based on the course lecture notes. We point out that for simplicity this code only handle one-dimensional dataset due to the parameters of the 'reshape'.

```
# A linear regression function, takes as input a dataset
    consisting of a list of pairs (x, y).
# Return a couple (w, b), parameter of the affine linear model
    resulting from the linear regression.
def lin_reg_1D(data):
    X = np.concatenate((data[:, 0].reshape(-1, 1), np.ones(len(
    data)).reshape(-1, 1)), axis=1)
    Y = data[:, 1].reshape(-1, 1)

    # Compute the linear model, as presented in the lecture note:
    (X^T X)^-1 X^T Y
    res = np.matmul(np.matmul(np.linalg.inv(np.matmul(X.T, X)), X.
    T), Y)

    return res
```

We plot in Figure 2, the result of an immediate application of the algorithm to the PCB dataset. This model result in a MSE equal to 24.80106432.

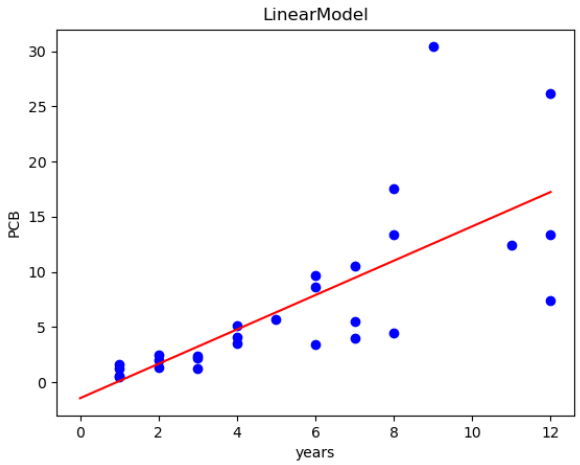


Figure 2: The linear regression applied to the PCB dataset.

3.2 Build a non-linear model (10 points)

We now learn the model of the form $h(x) = \exp(ax + b)$ where the learn the parameters $a = 0.25912824$ and $b = 0.03147247$, resulting in an MSE equal to 77.7699548. We also plot the linear model in the modified dataset S' in Figure 3 and the non-linear model in the original PCB dataset in Figure 4.

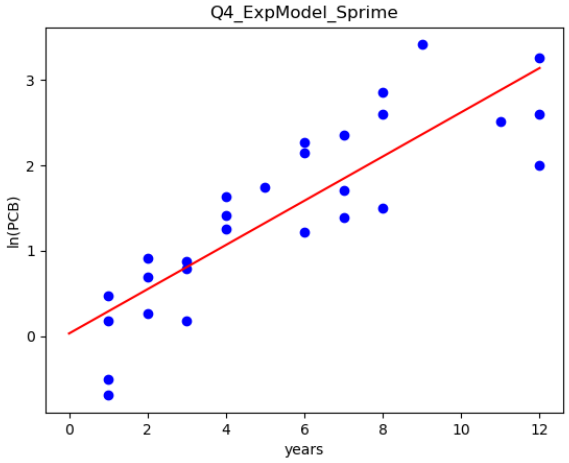


Figure 3: The linear regression applied to the dataset S'.

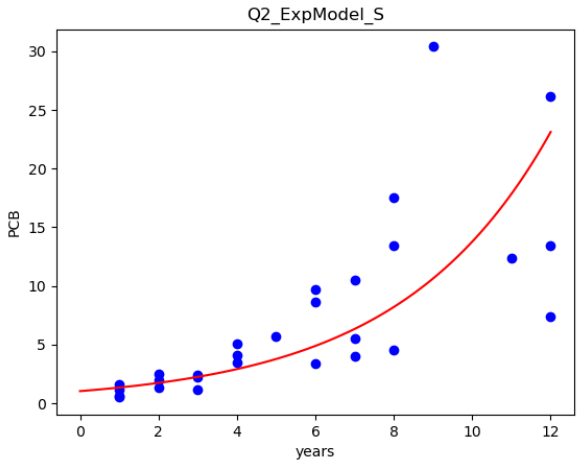


Figure 4: The non-linear model and the PCB dataset.

3.3 An example of non-equal arg min (10 points)

In this question we provide an example where the following inequality is observed:

$$\arg\max_{a,b} \sum_i (y_i - h(x_i))^2 \neq \arg\max_{a,b} \sum_i (\log(y_i) - (ax_i + b))^2$$

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

To that end, we use defined a dataset $S^{\neq}$ with two points: $S^{\neq} := \{(0, y_1), (0, y2)\}$, it's now quite easy to choose two value for y_1 and y_2 such that the argmin for b will be different in the left-hand and right-hand, only b matters here, since all x are equal to 0 and thus the expressions are independent of the parameter a.

For the sake of visualisation we choose the arbitrary $y_1 := 0.4$ and $y_2 := \exp(1.01)$ and plot in figure 5 the two expressions as a function of b, we observe that the argmin for b is indeed different.

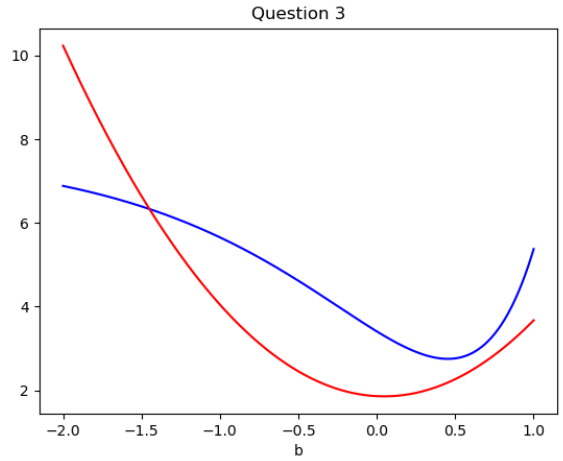


Figure 5: Both expressions (left-hand in blue, right-hand in red) as a function of b in the dataset $S^{\neq}$.

3.4 Plots (5 points)

The plot is provided in Figure 3.

3.5 Compute R^2 (10 points)

We compute the coefficient of determination R^2 for the non-linear model and the original dataset and obtain $R^2 = 0.35701357$.

Now we briefly discuss R^2, we first observe that $R^2 = 1$ if and only if the model perfectly fit all elements of the dataset, additionally by definition $R^2 \leq 1$. Then we observe that $R^2 = 0$ if the model predict the average value $\bar{y}$. Finally we observe that R^2 can be negative by definition, for example if the model is not fitted for the dataset but arbitrarily wrong. The coefficient R^2 would characterise the performance of a model over dataset with an increased 'fit' the closer R^2 is to 1.

3.6 Another non-linear model (10 points)

We finally learn the non-linear model $h(x) = \exp(a\sqrt{x} + b)$, the parameters learnt are $a = 1.1986063$ and $b = -1.19475082$ and the MSE is equal to 29.35919865. We provide the plot over the dataset S' from the previous questions in Figure 6.

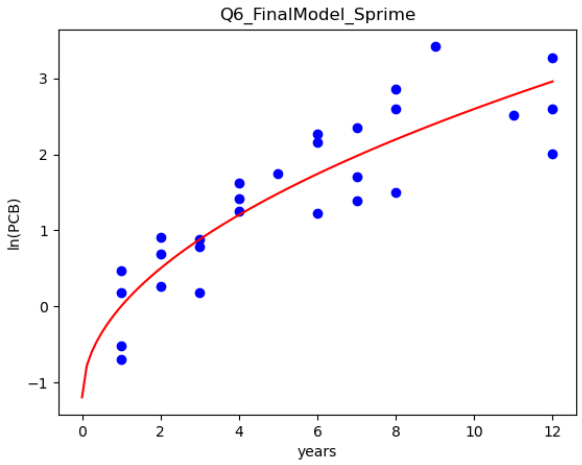


Figure 6: The final model and the dataset S'.

Now we compute the coefficient R^2 over the dataset with normal –linear– inputs and the logarithmic scale targets (dataset S'). We have for the previous exp based model $R^2 = 0.73139154$ and for the last model we have 0.78610565, giving the intuition of a better fit for the last model as can be observed in the respective figures.