Machine Learning A
Home Assignment 1

Jakob Schauser, pwn274

September 2021

1 Make Your Own

1.1 What profile information would you collect and what would be the sample space X?

The grades in prior courses seems like a good bet (especially in courses with similar curricula). The workload in other courses the student takes alongside the course should also be added. Maybe the grades of people in the students study groups could also be of help. This would be a sample space along the lines of $\{-3, 0, 2, 4, 7, 10, 12\}^4 \times \mathbb{R}$

1.2 What would be the label space Y?

Either a grade in the danish 12-scale $\{-3, 0, 2, 4, 7, 10, 12\}$ or a float (say $x \in [0; 1]$) that can be translated into one.

1.3 How would you define the loss function L(y,y)?

Going for something simple, a Mean Square Error between predicted and truth could work. A penalizing term for mistakenly failing/passing could also be added.

1.4 Assuming that you want to apply K-Nearest-Neighbors, how would you define the distance measured d(x,x')?

I don't think I have the intuition on when to use one or the other yet, so I will say the squared Euclidan or Taxi-cap since I have tried implementing these before. (Bonus: I have always loved the way DnD handles distances which is simply $d(x, y, x', y') = max(abs(x - x'), abs(y - y'))$ and I am sure it has some applications, but I am not sure when it would be useful)

1.5 How would you evaluate the performance of your algorithm? (In terms of the loss function you have defined earlier.)

Splitting into a training and a validation set, then simply using the Mean-Squared-Error could be enlightening. This will not show if there is a bias in the model for grading lower or higher in general, so I would maybe pair this with an average error.

1.6 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?

Yes, issues would come up! The idea is horrible! First of all, if everyone knows their grades in advance, no one would do any extra work. Secondly, external circumstances could change the actual deserved grade. Also, I am neither a philosophist or a lawyer, but unless perfectly corrected for, the model might have some strong biases (racism, sexism, exchange-student-ism or something like that) which might make the models decisions straight up illegal.

2 Digits Classification with K Nearest Neighbors

2.1 General thoughts on the implementation

As the distance metric was hinted to simply be the squared Euclidean, the only choice that was needed was the validation error. Here I chose the *zero-one loss*. This is both simple to implement and telling as a validation-metric, as it doubles as number of correctly guessed per batch.

The most important part of my code is undoubtedly the following snippet:

```
# Return the k-NN predictions for all k at the same time
def knn(i):
    # After some matrix-gymnastics on paper I came up with this way to
   find all distances to a digit
    dists = dist(np.array(dat[:N]).T, np.tile(dat[i],(N,1)).T)
    dists = np.diag(dists)

    # Sorting by proximity and only looking at the first 50
    nbs = labels[np.argsort(dists)][:50]

    # Taking the hint and creating a running average
    guesses = np.cumsum(nbs)/np.arange(1,len(nbs)+1)

    # Exploit the fact that we have a binary problem by simply using
   rounding as a way of voting
    return np.round(guesses)
```

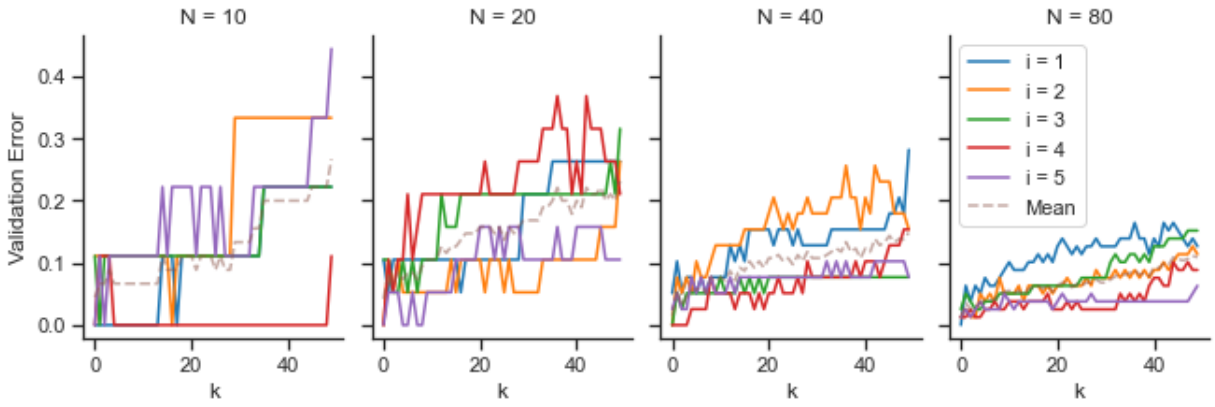

Figure 1: The validation error for digit classification as a function of neighborhood size. The plots correspond to a specific sample size each, and every line is a different sample.

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

The main results can be seen in Figure 1. The validation error is not only generally lower for higher N's, but also less dependant on the exact data looked at, as can both be seen qualitatively as the spread between the different i-values and in the variance-plot (Figure 2).

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

Slightly surprising at first, in the clean sample, it is pretty clear that a lower k corresponds to a better predictor. This is most likely because the 5's almost always lies closer to other 5's than any 6's in feature space (and vice versa) but not by a large margin. Enlarging the neighborhoods would therefore only increase the possibility of wrong digits "contaminating" the voting.

2.4 Discuss how corruption magnitude influences the prediction accuracy of K-NN and the optimal value of K.

Looking at Figure 3 we see a different picture emerge than we did in Figure 1. In the clean sample the optimal value of k was "as low as possible", but as the corruption increases a 'valley' emerges, where the optimal number of neighbors seems to lie around 20. This seems more in line with my intuition of the Jury Theorem[1] kicking in, where the decision made by multiple predictors are more reliant than any single one. But as k rises too high, the neighborhood becomes diluted by other digits, which gives rise to the apparent 'valley'.

[1] `wikipedia.com/Jury_theorem`

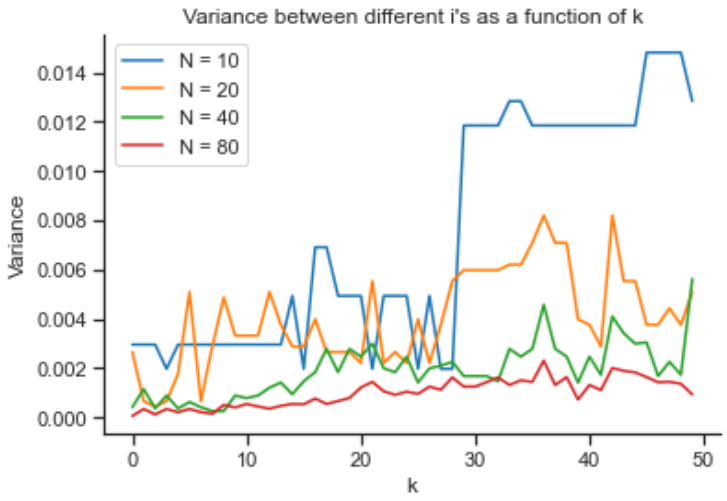

Figure 2: The variance as a function of number of neighbors k.

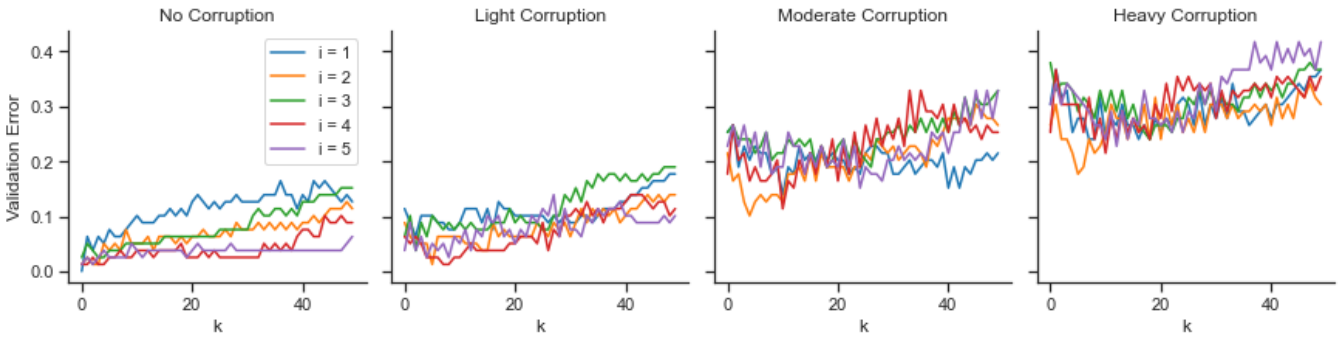

Figure 3: A comparison of the validation errors for the differently corrupted digits, all with a simple-size of N = 80.

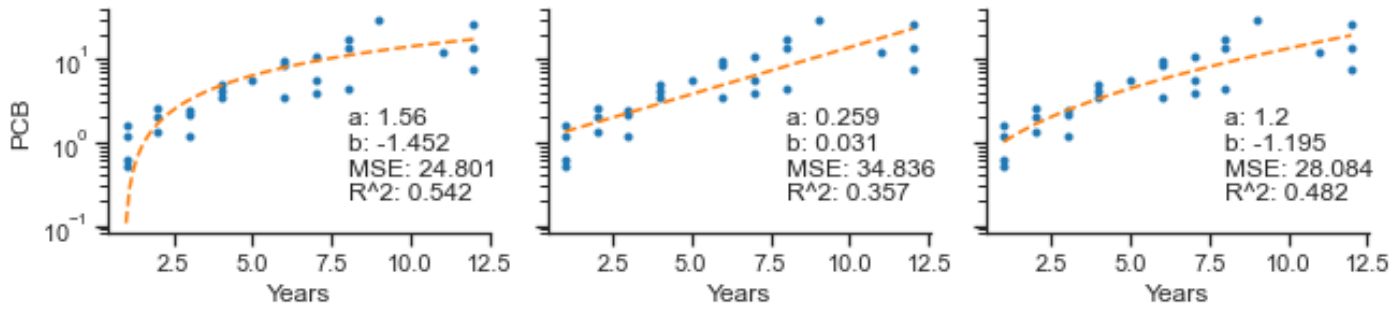

Figure 4: The three fits on top of the data. The a and b parameters along with the R^2 and MSE are written on the plot. The first is a simple linear fit, the second $\exp(ax+b)$, and the third $\exp(a\sqrt{x}+b)$.

3 Linear Regression

3.1 Implementation of linear regression

I implemented linear regression using the pseudoinverse as introduced in the lecture. My implementation can be found in the second half of the notebook.

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

```
def lin_reg(x,y):
    X = array([x,np.ones(len(x))]).T

    w = inv(X.T @ X) @ X.T @ y

    return w.T
```

As this was now defined as a function, the following sub-tasks simply consisted of transforming x and/or y.

3.2 Building the first model

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. In my code, this simply amounted to:

```
w = lin_reg(x,np.log(y))
```

Using the parameters for $ln(y)$ and inserting them into the exponential function gave me the second plot in Figure 4. It can be seen that this fit looks linear on the logarithmic scale giving us some confidence in our result. I found $a = 0.259$, $b = 0.0315$ and a Mean Square Error of 34.8.

3.3 Discussion of R^2

The R^2 is defined as:

$$1 - \frac{\sum_{i=1}^{N} (y_i - h(x_i))^2}{\sum_{i=1}^{N} (y_i - \bar{y})^2}$$

The R^2 will be 1 if and only if the fraction of the sums equal 0 which corresponds to hitting the exact answer(i.e. $y_i - h(x_i)$ is 0 for every i). If a guess is just as bad as guessing average, the R^2 will be 0, as the fraction will approximate 1. Finally we can see, that the R^2 is negative if the fit is worse than simply guessing on the average.

3.4 Building the second model

To fit a model of the form

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

with parameters $a, b \in \mathbb{R}$, I transformed the output data (the y values) by applying the natural logarithm first. I then mapped x to $\sqrt{x}$. In my code, this was not harder than the last model, as it was done by this line:

```
w = lin_reg(np.sqrt(x),np.log(y))
```

Using the parameters for $ln(y)$ and inserting them into the exponential function gave me the third plot in Figure 4. I found $a = 1.20$, $b = -1.20$ and a Mean Square Error of 28.1

We now have a lower MSE and a higher R^2, both of which are signs of a better fit. But using these metrics, the simple linear fit would be considered best so I am not sure what can be concluded from this fact.