Research Paper

Character recognition of handwritten digits

0. Motivation:

While being impressed by today's standard of large language learning models and artificial intelligence, I wanted to investigate to which degree and under which circumstances a simple model like logistic regression for classification can still recognise with a high accuracy distorted representations of the data.

To conduct this experiment I used handwritten digits provided by the MNIST-Database, represented as 28x28 gray-scale images, values ranging from 0 to 255.

The Python code and other files are uploaded to GitHub (cf. 1) and are freely accessible.

1. What is Python and why did I choose it?

Python is a beginner friendly, syntactically comfortable and useful programming language. Due to the simplistic syntax the code is very easily maintained and more productive. It uses indented spaces as a form of code segmentation, differentiating levels of the application, such as loops, functions, classes, etc, while other languages such as Java, JavaScript or C use curly brackets, which often cause problems in debugging – determining problems and fixing them. In addition, Python is considered to be a high level language since it is quite closely related to the English language and is therefore easy to understand, even if the syntax or even semantics of the language are unknown. Although Python is based on the C programming language, which on the one hand is more complicated and allows greater control of hardware-related decisions, making faulty code much more dangerous, is also a lot faster, it does not inherent the speed of C. This is due to the intention of the developers to create a simple, yet useful, language that takes over a great portion of the work itself (cf. 2).

This drawback in performance is also due to the fact that Python needs to be interpreted before it can be compiled, meaning that the code is translated to an intermediate low-level language before it is translated to executable machine code, whereas C is directly compiled. But unlike its parent language, Python is much more secure and stable and is supported by a large community. There exist around 13,000 freely accessible libraries, providing ready-made implementations of nearly every aspect of computing and development (cf. 2).

The libraries used in this paper include Numpy, Scikit-Learn and Matplotlib. Numpy especially deals with vector and matrix operations and the storage and organization of large amounts of data, while Scikit-Learn provides implementations of machine learning algorithms, like logistic regression,

which is of interest to this paper, and Matplotlib was used to graphically display the data received from the experiment.

2. Preparation of the data:

The data used to train the model is from the MNIST-Database, which provides a labeled set of handwritten digits in the form of 785-item arrays, including at the first position the label and the rest representing the gray-scale representation.

The exact files containing the data although were not from the database's website but instead publicly available on GitHub (cf. 1), since the database's source was no longer available. The data was stored in two CSV-files and extracted using Python's built-in method for manipulating external documents. This resulted in a list of 60,000 digit representations for training and 10,000 digit representations for testing. In order to simplify the computations the values were normalised to a unit range. In addition, a new array was created in which only the labels were stored and the first position in the arrays was set to 0, therefore not contributing to the model.

Since the method for extracting text from a file in Python includes new lines and other indentations, these needed to be sorted out and corrected. This was done by splitting the data and excluding the last element - the new line - first and then looping over each element and dividing it by 255. (figure 1).

The training set and test set received from this preparation were directly used to create and evaluate the model, only in the testing of the generalisational ability was the test set altered in the later described fashion.

3. Machine learning:

Generally two kinds of learning algorithms, the problems they solve, are differentiated: regression and classification. While regression is used to find a functional relationship between the input vectors and the output – mostly one-dimensional as value estimations - that they are directly mapped to, classification aims to sort the input examples into a predefined set of classes by mapping the input vector to a conditional probability of that class (cf. 3, p. 35-38). Oftentimes, like in the case of logistic regression, this is done by using a regression algorithm and applying an activation function, that maps the regressed values to probabilities.

There exist different kinds of model types and techniques used to solve different problems. Some models are parametric, like linear or logistic regression, and some are non-parametric, i.e. k-nearest-neighbors, abbreviated kNN. While both types of models directly learn from the given input data, parametric models extract parameters from the data, variable values that map a given input to the output, and therefore don't need to store the data after the learning process, non-parametric models often rely on the training set while determining the output. For example kNN finds the nearest k examples, in terms of euclidean distance, and takes the most probable – often appearing – result. All of the previous examples are supervised algorithms, meaning the output of any given training vector is already given and part of the training process. There also exist other types of learning techniques, like unsupervised or semi-supervised learning. These techniques can also often times be necessary, especially when the data is not at all, or very rarely, labeled or nearly no supervision is possible in the

training process (cf. 3, p. 34-42). Luckily, the training data from the MNIST-Database is completely labeled, so logistic regression, which is a classification algorithm – the name is due to historical circumstances – is chosen to be inspected in terms of generalisational ability.

3.1 Logistic regression:

Logistic Regression is based on linear regression to which an activation function, in this case the logistic or also called sigmoid function is applied, which maps the domain to the range [0;1] (figure 2):

$$h(z) = (e\text{^}z)/(1+e\text{^}z)$$

The output from the linear regression of some input vector $x = [1, x_1, x_2, \ldots, x_p]^T$,

$$z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p = \boldsymbol{\beta}^T \mathbf{x},$$

is the input for the logistic function. The weight β_i determines the importance of the component x_i for the specific class, positive values correspond to a positive influence on the class, while negative values decrease the probability for this class. The sigmoid function has the benefit of being nearly linear near 0 and rapidly pushing higher and lower values to either 0 or 1 (cf. 4, p.25-27).

Statistically, classification amounts to predicting the conditional probability of some class given the input:

$$p(y, \mathbf{x}).$$

In the case of binary classification - classes 0 and 1 - this results in two probabilities to be learned $P(y=0, \mathbf{x})$ and $P(y=1, \mathbf{x})$. By the laws of probability theory these two terms add up to 1:

$$P(y=0, \mathbf{x}) + P(y=1, \mathbf{x}) = 1.$$

Therefore simplifying the learning process to just one model, since the other can simply be calculated by subtracting the known value from 1. Every multi-class classification problem can be interpreted as a binary classification by using the one-versus-all method.

It is important to note that since linear regression is used as a starting point of logistic regression, the decision boundary, the hyperplane where the prediction changes from one class to another, is always linear. This results in logistic regression being called a linear classifier. This introduces bias into the evaluation since it is assumed that the data inherits some kind of linearity. Even in multi-class classification the boundary between each two classes is a linear hyperplane.

$$p(y=1, x) = p(y=0, x)$$

$$e\text{^}(\beta^T x)/(1+e\text{^}(\beta^T x)) = 1/(1+e\text{^}(\beta^T x)) \leftrightarrow e\text{^}(\beta^T x) = 1 \leftrightarrow \beta^T x = 0$$

This last expression represents the linear hyperplane (cf. 4, p.28f.).

The threshold by which the resultant class is chosen is a hyperparameter α, a parameter not directly trained in the training phase, but otherwise determined (section 3.3), such that:

$$\text{decision}(x) = \{1 \text{ if } P(y=1, x) > \alpha; 0 \text{ otherwise}$$

3.1.1 Logistic regression for more than two classes:

To generalise logistic regression for more than two classes K = n , with n bigger than 2, there exist two important methods: vanilla and one-hot encoding. The assumption is made, that every example corresponds to only one class, meaning only singular predictions are of interest.

Vanilla encoding is based on the one versus all tactic, creating a binary classifier in which all other classes except the one explicitly trained for are “the rest”, i.e. 0 if the explicit class is represented by 1. Using this method the output from the classifier is a probability of this input being of this specific class. To eventually determine the most probable class, a classifier would need to be trained for each class separately using this method. So for a 10-class classification as for handwritten digits, 10 classifiers for each class, i.e. 0, 1, 2, etc, need to be trained.

In contrast to this method stands one-hot encoding. This method supposes the output from the classifier would be a vector of probabilities relating to each class. Therefore all elements of this vector would add up to 1. To receive this vector from a linear regression, a different activation function, in this case the softmax function is used:

$$\text{softmax}(\mathbf{z}) = 1/(\text{sum(from } j=1 \text{ to } K)\ e^{z_j}) * [e^{z_1}, e^{z_2}, \ldots, e^{z_k}]$$

in which $\mathbf{z}$ is a K-dimensional input vector $\mathbf{z} = [z_1, z_2, \ldots, z_K]$. The classification output of the softmax regression, would be a K-dimensional vector with the class-component set to 1 while all other will be set to 0. A vector of this type is called a one-hot vector. While this method seems different from the one-against-all method, it is actually quite similar, but includes all parameters in one operation instead of creating K different classifiers. Although it might not deliver the same probabilities, since the softmax function does differ from the sigmoid function in terms of mapping (cf. 4: p. 30f., cf. 5: p. 84f.).

One-hot encoding will not be used in this paper, because it requires higher computational efforts that the test machine is not able to uphold. It is also important to note that the softmax solution is slightly over-parameterised compared to binary logistic regression and for a small amount of classes the one-versus-all method is preferred.

3.1.2 Training the logistic model:

The goal is to learn $\boldsymbol{\beta}$ from the training data $T = \{(\mathbf{x}_i, y_i)\}^n_{i=1}$ using the maximum likelihood approach, ultimately maximising the likelihood-function with respect to the weight vector $\boldsymbol{\beta}$ or as used in this paper minimising the cross-entropy loss function:

$$\beta = \arg_\beta \min L_{CE}(...)$$

$$L_{CE} = \ldots$$

A loss function in regression is commonly described by the distance – metric error - between the prediction - the probability of the class - and the true label.

Minimising the cross-entropy function is a convex optimisation problem. Unfortunately, the resultant vector-valued equation has no closed form solution, so it can only be numerically approximated by a numerical algorithm, for example using gradient descent (next section).

3.2 Stochastic gradient descent:

(from stanford)

Stochastic Gradient Descent relies upon the fact, that the loss function (negative of likelihood function) is convex, meaning there exist only one minimum or maximum value and no local extrema, therefore the algorithm is guaranteed to approach a global extrema instead of getting stuck near a local extrema.

The idea behind gradient descent is described in my previous paper “Character recognition and turing tests” (2025)

The algorithm itself calculates the gradient for each training example with respect to each weight component, through the equation:

$$\partial \, \mathrm{LCE}(\hat{y}, y)/\partial \, w_j = -(y - \hat{y})x.$$

The algorithm is named as stochastic because it uses real-world training examples to calculate the gradient and apply the change to the weight vector. This is then done for each input vector individually or using multiple at once, called batch or mini-batch training, depending on the size of the batch. Using a batch method may increase the computations but decreases the amount of times the weight vector is updated so as to create smoother updates of the values. The gradient of a batch is the average gradient of all included input vectors, resulting a matrix valued expression for calculating the gradient.

3.3 Hyperparameter tuning:

Hyperparameters can be defined as variable values, non-altered in the training phase, that affect the performance or learning rate of a given model. These are often times chosen prior to the learning phase or individual models are trained with differing hyperparameters so as too choose the best performing one to continue the process.

If the hyperparameters are not chosen prior to the learning phase, they can be determined by two methods: grid search and random search. Grid search generally involves more supervision, the values that are tried will be provided by the supervisor. This means that if no value of the provided ones are ideal, either the process continues or a non-ideal one is chosen to start the training phase. Random search on the other hand tries different values chosen from a distribution, for example the normal distribution (cf. 3: p.291-295).

The hyperparameters that needed to be determined were the tolerance – stopping criteria for the training, and the threshold for determining the class from the probability. In determining these hyperparameter, I used grid search and provided in the beginning more differing and later on more similar values to get a semi-ideal precision, I chose to stop when the accuracy was as high as possible, while having a precision over 90 % (metrics are explained in the next section). I chose this

criteria since the accuracy of predicting every example as false would lead to an accuracy of approximately 90 % since the classes are nearly distributed equally, 10 % for each class.

3.4 Evaluation/performance metrics and confusion matrix

Using the predictions of the model in relation to the actual labels, it is possible to sort each prediction in one of four classes, these include true negative (TN), false negative (FN), false positive (FP) and true positive (TP). The names are direct reflections of their meaning, so they will not be explained further. The two-dimensional relation between the labels and the predictions is often times represented in a matrix called the confusion matrix, in which the four cells represent one of the previous classes (cf. 3: p. 272f., figure 4).

To evaluate a given model there exist four main metrics: accuracy, precision, recall and F1 score. All of these metrics rely on some of the four classes given in the confusion matrix. Accuracy is a measure of the overall correct predictions in relation to all predictions:

$$\text{Accuracy} = TP + TN / TP + TN + FP + FN$$

In trying to achieve a high accuracy, correct predictions of both types are equally important. There can also be instances in which a true positive value might be much more important than a true false value. Such circumstances might include indicators for cancer, it is important to catch as many cancerous patients as possible even if it includes people who do not show any signs of cancer. In such a case it becomes more important to catch every patient than to reduce medical costs. If such predictions become important a better measure would be recall, also called sensitivity. Recall represents the correctly predicted true examples in comparison to every true example:

$$\text{Recall} = TP / TP + FN$$

Precision is similar to recall but measures the correctly predicted true examples in comparison to all examples predicted as true:

$$\text{Precision} = TP / TP + FP$$

As can be seen from the equations, precision also includes negative examples that were classified as positive while recall only focuses on positively labeled examples.

Lastly, the F1 score tries to achieve a balanced optimisation of both precision and recall. It is defined as the harmonic mean of precision and recall:

$$\text{F1 score} = 2*\text{Precision}*\text{Recall} / (\text{Precision} + \text{Recall})$$

(cf. 3: p. 274f.).

4. Experiment and evaluation

To evaluate the models in terms of generalising, classifying distorted images, some of the image’s components will be randomly changed to either 0 or 1, so the minimum or maximum values , entire rows and columns will be changed to 0 or 1 and lastly every component will be increased and decreased until every component reaches 1 or 0 respectively. Each of these methods is implemented

on entire sets and are shown in figures 5 and 6. Because the image representations are not divided in individual rows or columns the components are changed iteratively.

Although the logistic regression algorithm was implemented in Python and is shown in figure 7, to evaluate the model the Scikit-Learn implementation was used, since it achieves a higher accuracy in less time, such that the hyperparameters were able to be tuned, which with my own model was not possible in the given time.

In order to evaluate each model the positive and negative recall (section 3.4), so the correct positive predictions and the correct negative predictions, were applied and will be supplied in each of the following examples.

4.1 Hyperparameter tuning of the model

As already mentioned, the tolerance and the threshold were the parameters needed to be tuned for the Scikit-Learn implementation. In this case the tolerance supplies the stopping criteria to the optimisation algorithm of logistic regression, whereas the threshold is used to optimise the classification after the regression is complete.

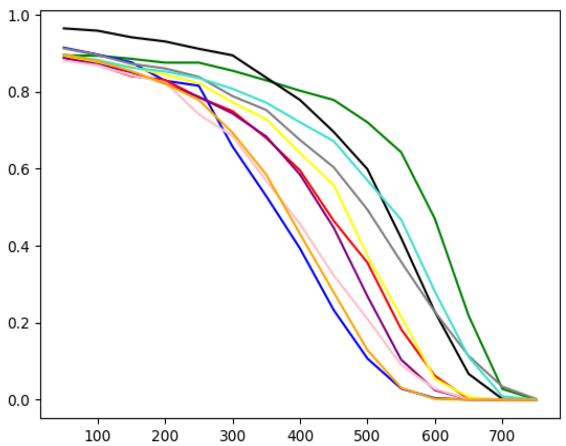
Ultimately, the tolerances for the models, in order of increasing label, so 0, 1, 2, etc, that were determined can be seen in table 1. The threshold to achieve the mentioned criteria in section 3.3 are also listed in table 1, as well as the positive and negative recall values.

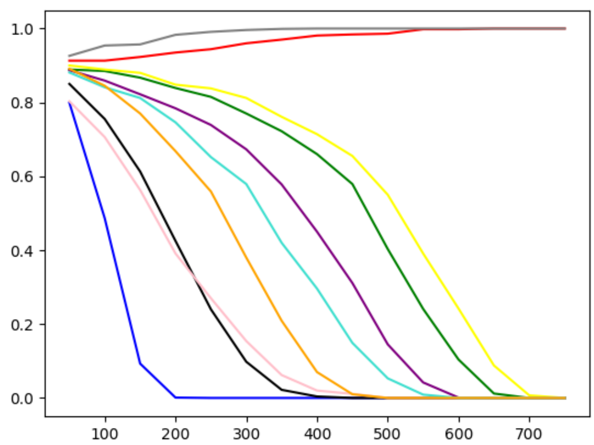
It is already apparent, that some of the models, especially for 1, 4, 7 and 9, seem to determine quite high values in the regression, such that the threshold is similar to the highest possible threshold of 1. These models over-estimate the probability for a high number of test examples, so their performance is questionable, although with these determined thresholds they gave some of the most impressive results.

Positive label	0	1	2	3	4	5	6	7	8	9
tolerance	0.05	0.007	0.03	0.03	0.005	0.0007	0.03	0.0007	0.05	0.009
threshold	0.72	0.98	0.88	0.86	0.98	0.92	0.9	0.994	0.63	0.97
Pos. and neg. recall	0.9;0.976	0.926;0.99	0.905;0.905	0.9;0.909	0.91;0.947	0.907;0.878	0.903;0.966	0.901;0.979	0.903;0.797	0.911;0.82

Table. 1: hyperparameters and performance metrics

4.2 Random component change





4.3 Brightness shift

4.4 Column and line change

5. Conclusion

Sources:

1: Code repository: https://github.com/JoeWriedt/Character-Recognition/tree/main

2: https://brochure.getpython.info/media/releases/psf-python-brochure-vol.-i-final-download.pdf/at_download/file

3: Practical Machine Learning with Python, D. Sarkar, R. Bali, T. Sharma, 2018

4:

4: Carnegie Mellon University Chapter 12 Lectures: https://www.stat.cmu.edu/~cshalizi/uADA/12/lectures/ch12.pdf

5: Speech and Language Processing. Daniel Jurafsky & James H. Martin 2024, https://web.stanford.edu/~jurafsky/slp3/ed3book.pdf