
Number recognition for use in a Sudoku solver

Abstract

The goal of this work is to implement number recognition for a Sudoku solver. First we will describe the generation of synthetic data, in this case labelled images of numbers. Then we compare different machine learning techniques for supervised learning using the generated data. Once the training is complete, images of real Sudokus are captured using the webcam. The different methods then try to correctly predict each number.

1. Introduction

Number and letter recognition is key in many real-world applications. Digitalizing print media, automated sorting of letters, or recognition of signs in robotics or driver assistance systems require well performing algorithms for character recognition. In the following we analyze the performance of different techniques for number recognition, namely Neural Networks (NN), Support Vector Machines (SVM) and Logistic Regression (LR), by using them in an automated Sudoku solver.

2. Overview

Figure 2 shows the basic data flow and the participating data manipulators. First of all we generate data by sampling position, rotation, size and color of a number and place it based on these values on a white 20x20 pixel image. Examples of 9 such generated images are shown in Figure 1. The corresponding labels are defined by the name of the folder the images are stored in. These images are preprocessed (applying greyscale) by an octave script and written to an input matrix and a target vector. All pixels of one image are encoded in one row of the input matrix and the corresponding label is stored in the row of the target vector. The data is dumped to a “cache.mat” file which allows us to apply many different tools and programming lan-

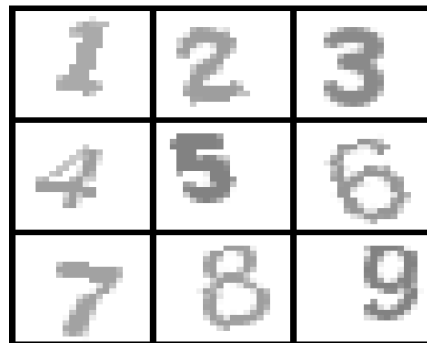


Figure 1. Examples of generated numbers

guages to the dataset. In our case we use it in python and octave. The input matrix $X^{n \times 400}$ and the target

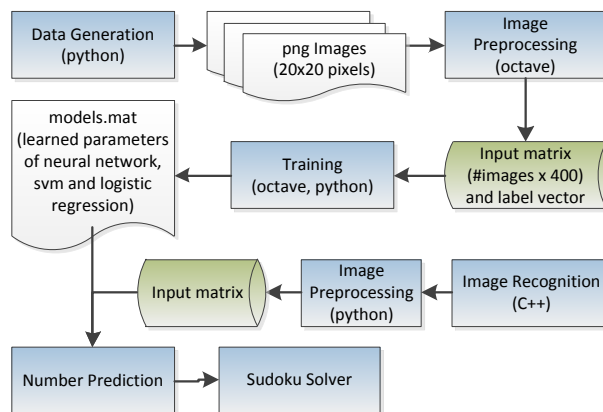


Figure 2. Overview

vector y are the inputs for training NN, SVM and LR, as described in Section 3. After training, the learned models/parameters are saved in “.mat” files that can be loaded by the number prediction. The other input for number prediction is an input matrix of the same form as X with a row entry for each Sudoku field that was detected to contain a number. This implies that we do not train our models to detect empty fields. We could reliably detect empty fields by performing some preprocessing steps on the images (filling borders, adjusting thresholds etc.) thus applying a machine learning technique does not seem appropriate for this task.

3. Training

3.1. Neural Network

Applying a NN seems promising as it was already successfully applied to the quite similar problem of detecting handwritten digits. [ref!](#) For the NN we choose a simple architecture with 25 hidden sigmoidal units (+ bias unit), as shown in figure 3. The size of the input layer is 400 (given by the 20 x 20 pixel images) and the size of the output layer is defined by the number of classes, in this case the numbers 1 to 9. This simple architecture turned out to perform pretty well compared to more complex ones with a higher number of hidden layers. Training is performed with error backpropagation, thus minimizing the error function

$$E(w) = \sum_{n=1}^N E_n(w)$$

with $E_n(w) = \frac{1}{2} \sum_k (y_{nk} - t_{nk})^2$

and $y_k = \sum_i w_{ki} x_i$

subject to w . For NNs we test two implementations.

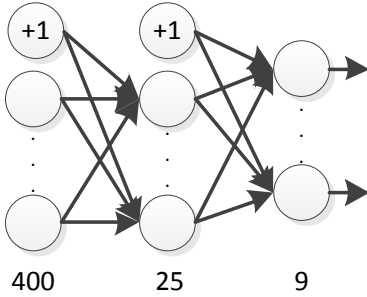


Figure 3. Neural Network with one hidden layer

On the one hand we use a partly self-developed octave code for two-layer NNs which was written as part of the stanford machine learning course¹, on the other hand we include PyBrain (Schaul et al., 2010) for comparison and easier evaluation of more sophisticated network architectures. To find a good value for the regularization parameter λ we perform a grid search with k-fold cross-validation which gives us a good performance when choosing $\lambda = 1$.

3.2. Support Vector Machines

The next technique we apply to the discussed problem are Support Vector Machines (Bishop, 2006) which try

¹<http://www.ml-class.org>

to solve the following optimization problem:

$$\min_{w,b} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^n \xi_i$$

subject to $y_i(w^T \phi(x_i) + b) - 1 + \xi_i \geq 0$
and $\xi_i \geq 0$.

Using a radial basis function (RBF) as kernel we perform a grid search over C and γ to find those parameters that provide highest accuracy on the test set. We use the open source library “libsvm” (Chang & Lin, 2011) wrapped by some octave scripts for setting up data sets and performing grid search with k-fold cross-validation. Grid search with cross validation resulted in accuracies illustrated in Figure 4. From the best parameter pairs we choose the one that performs best on the validation set, which gives us $C = 20$ and $\gamma = 0.1$.

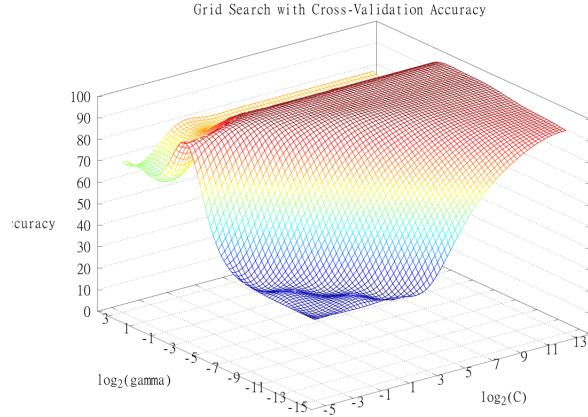


Figure 4.

3.3. Logistic Regression

Finally we apply Logistic Regression to the datasets. Again we use a (vectorized) octave implementation for learning and prediction. Therefore we minimize the (regularized) error function

$$E(w) = \frac{1}{n} (-y^T \ln(\Phi w) + (1 - y^T) \ln(1 - \Phi w)) + \frac{\lambda}{2n} w^T w$$

with respect to w . Minimization is performed by an algorithm² that was provided during the stanford ma-

²<http://sprinkler.googlecode.com/svn/trunk/regression/fmincg.m>

Table 1. Classification accuracies for Neural Network, SVM and Logistic Regression on different data sets.

DATA SET	NN	SVM	LR	WINNER
D1	95.9± 0.2	96.7± 0.2	96.7± 0.2	?
D2	83.3± 0.6	80.0± 0.6	?	?
D3	61.9± 1.4	83.8± 0.7	?	?
DVAL	74.8± 0.5	78.3± 0.6	?	?

chine learning course. Grid search and k-fold cross-validation recommend a value for the regularization parameter λ .

4. Image Recognition and Preprocessing

Image recognition is written in C by extensive use of the open source library OpenCV³. Images captured by the webcam are continously analyzed to find a Sudoku square. Images are transformed to greyscale, then an adaptive threshold algorithm is used to preprocess the image for contour detection by OpenCV. OpenCV returns a list of contours which are further checked for certain criteria like having four corners or being of certain size. Since the contours are not detected precise enough lines are fitted through six checkpoints on each edge in order to achieve subpixel accuracy. The content enclosed by the corners is perspectively warped to fit a predefined square size. Finally, the thresholded Sudoku field is saved to disk. A python script is then used to split the whole field into 9x9 subfields with each subfield being flood filled around the corners for noise reduction. The format of the subfields now equals to the one provided by the generated images which allows us to apply the trained NN/SVM/LR models and, finally, apply a Sudoku solver.

5. Evaluation

To evaluate the performance of different approaches we define the following data sets:

- d1: 4500 generated images
- d2: ~ 1000 gathered images
- d3: $D1 \cup D2$
- dVal: images of a captured Sudoku

6. Future Work

- include information if a sudoku is unsolvable
- online learning

7. References

References

- Bishop, Christopher M. (*Pattern Recognition and Machine Learning*) By Bishop, Christopher M. (Author) Hardcover on (08 , 2006). Springer, 8 2006.
- Chang, C.C. and Lin, C.J. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011.
- Schaul, T., Bayer, J., Wierstra, D., Sun, Y., Felder, M., Sehnke, F., Rückstieß, T., and Schmidhuber, J. Pybrain. *The Journal of Machine Learning Research*, 11:743–746, 2010.

³<http://opencv.willowgarage.com/wiki/>