Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Institut für Integrierte Systeme
Integrated Systems Laboratory

Department of Information Technology and Electrical Engineering

Machine Learning on Microcontrollers

227-0155-00L

Exercise 2

Good Practices for Training ML models

ETH Center for Project-Based Learning

Wednesday 24th September, 2025

1 Introduction

In this short exercise, we will see on the one hand how polynomial regression fails on noisy data while neural networks are much more robust. On the other hand, we will discuss regularization techniques that speed up learning and avoid overfitting in neural networks.

2 Fitting Noisy Polynomial Data

We have a noisy measurement of a polynomial

$$f(x) = 10 + \frac{5}{10}x - \frac{4}{10^2}x^2 + \frac{2}{10^3}x^3 + \frac{3}{10^4}x^4 - \frac{1}{10^5}x^5 + N \tag{1}$$

with noise N.

2.1 Polynomial Regression

Polynomial regression finds $y(x)$ that minimizes the Mean Squared Error (MSE) based on n samples of $f(x)$

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(y(x_i) - f(x_i)\right)^2 \tag{2}$$

where $y(x)$ is a polynomial with degree d:

$$y(x) = \sum_{i=0}^{d} c_i x^i \tag{3}$$

In this exercise, we fit $y(x)$ with various degrees d and wish to obtain coefficients c_i that are identical to the ones of equation 1.

Student Task: What minimal degree d must $y(x)$ have such that an MSE of 0 can be achieved?

Note: Having a polynomial degree greater than the one found above quickly leads to capturing the noise of the data. This phenomenon is called *overfitting* and can also happen to neural networks if a model contains too many trainable parameters.

We now turn to the `regression` code provided with this exercise. In that code, we randomly generate 100 points of data according to equation 1 above. We choose $N \in [-2, 2]$ to be uniformly distributed and fit polynomials of various degrees on the $x \in [10, 20]$ interval. Then, we see how well the obtained results generalize by extending the interval to $[8, 22]$.

Student Task: Run the `regression` code. Which polynomial degree generalizes best? Why?

2.2 Deep Neural Networks

A deep neural network consists of daisy-chaining dense network layers. An individual layer of a such a dense network with n input units and a single output is characterized by

$$y(x) = \phi\left(b + \sum_{i=1}^{n} w_i x_i\right) \tag{4}$$

where b is a learnable bias, w_i are trainable parameters and $\phi()$ is a non-linear function. In this exercise, we consider both $\phi(x) = x^2$ and the so-called *exponential linear unit (ELU)*

$$\phi(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha(e^x - 1) & \text{else} \end{cases} \tag{5}$$

with a hyperparameter α that is set to 1 per default.

Student Task: Extend equation 4 to have k outputs.

Note: In this exercise, we have a deep neural network with one single input and one single output. Layers in between (the so-called *hidden layers*) are less trivial and resemble the form obtained in the student task just above.

3 Regularization Techniques for Neural Networks

We now turn to the `regularization` code of this exercise. In this part, we generate points on a noisy quadratic function and fit a network with several layers to it. Note that there are only very vew data points available for training, which makes the task challenging for the network.

Student Task: Run the `regularization` code. Is the neural network overfitting? How can one tell?

Additionally, there are five different configurations in the `regularization` that we can set: `dropout`, `batchnorm`, `normalize`, `l1_reg` and `l2_reg`.

Student Task:

1. What do these configurations do?
2. Play around with them and see how they affect the predictions. What do you observe?

To prevent overfitting, several techniques can be used on neural networks. In the following, we discuss several of them:

Batch size By backpropagating the average of several forward propagations, the training can be parallelized and thus accelerated. Furthermore, taking the average smoothes out noise, therefore regularizing the network.

Early stopping Stop training when the validation loss is as small as possible. Return the network before it can overfit.

L1/L2 regularization We extend the loss function with either the L1 term

$$\text{MSE} + \lambda \sum_{w \in W} |w| \tag{6}$$

or with the L2 term

$$\text{MSE} + \lambda \sum_{w \in W} w^2 \tag{7}$$

The non-negative λ can be seen as a "tuning knob". This means that the learnable weights w partly account for the loss and will therefore be kept lower than without L1 or L2 regularization.

Student Task:

1. Draw the gradients of $|x|$ and x^2.
2. Explain why in opposition to L2, L1 leads to sparse weights in the network.

Dropout Units are randomly disabled. This also speeds up learning and prevents units from co-adapting.

Student Task: Run the `regularization` code with dropout enabled by passing `dropout` as a command line argument. What do you expect to observe? Do the results match your expectations?

Batch normalization Recenter and rescale the data using the following empirical mean and variance of the batch B:

$$\mu_B = \frac{1}{|B|} \sum_{x \in B} x \tag{8}$$

$$\sigma_B^2 = \frac{1}{|B|} \sum_{x \in B} (x - \mu_B)^2 \tag{9}$$

Then, the data is transformed with

$$\hat{x} = \frac{x - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}}, \quad \forall x \in B \tag{10}$$

with non-negative ε.

Student Task: Why is there an ε in equation 10?

The `regularization` code has several other mechanisms implemented. Try them out if you like!

4 Feature Extraction

Now that you are more familiar with the technical tools, let's talk about an important concept you will find extremely useful: feature extraction.

When you want to analyze a particular object, like an image or an audio track, you firstly need a digital representation of it. In the case of an image, you will have a set of numbers describing the color of each pixel, in the case of an audio track, you will have a set of amplitudes and a sampling rate.

Sometimes, the information collected to describe the object can be redundant, or there are better ways to describe the same object reducing the required amount of data. The process of reducing the dimensionality of this information into more manageable descriptors (features) is called feature extraction. This process allows to compress the initial sets of data, easing the subsequent processing that would be otherwise impractical and too resources/time consuming.

Student Task 1 (Feature Extraction): Important features you can extract from an audio file are the Mel-frequency cepstral coefficients (MFCCs), coefficients that describe a Mel-frequency cepstrum, a representation of the short-term power spectrum of the audio.

You will now extract MFCCs from an audio sample with the aid of a Jupyter Notebook. Read the *Part 2: Audio Feature Extraction* of the Notebook.

To correctly run the code inside the notebook, you need to activate your Docker environment that you have installed in the first exercise.

Follow the instructions and run the code in the Notebook. Answer the following questions.

- Imagine to store all the samples in a micro-controller. How much memory do you need? Assume that the size of each sample is $2\,\mathrm{B}$ ($16\,\mathrm{bit}$).

  $Required\ memory$ =

- How many features do you have before extracting the MFCCs? **(We count also the information about the Sampling Rate)**

  $Features_0$ =

- How many features do you have, after? **(We count also the information about the number of frames)**

  $Features_1$ =

- Suppose that a sample and a MFCC have the same size in memory. Calculate the achieved compression ratio. **(With this definition of CR, the lower the CR, the better.)**

  CR =

$\mathcal{E}$ **Congratulations! You have reached the end of the exercise.**
If you are unsure of your results, discuss with an assistant. $\mathcal{E}$