FTML practical session 3: 2023/05/24

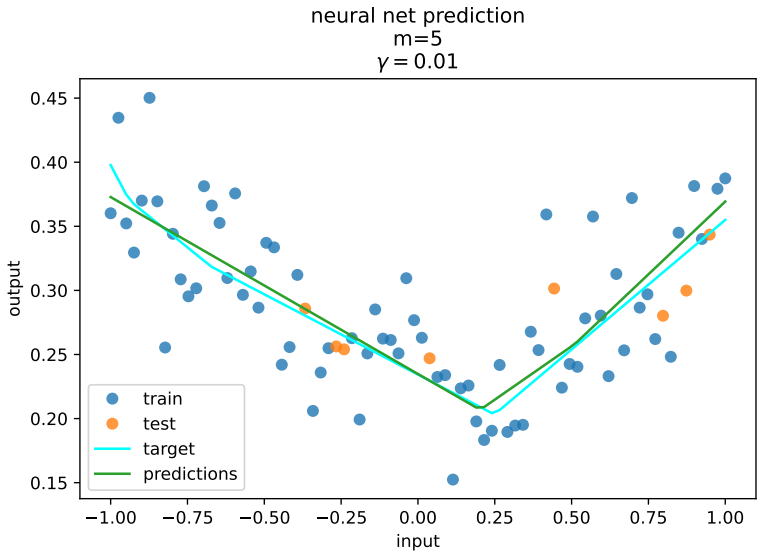

TABLE DES MATIÈRES

1 SIMPLICITY BIAS OF NEURAL NETWORKS

Introduction

This exercise assumes a basic knowledge of neural networks. We will witness that with some neural networks, it is unlikely to overfit the data and illustrate this with networks using the ReLU activation. In order to have some visual feedback, we will, use $\mathcal{X} = \mathbb{R}$ and $\mathcal{Y} = \mathbb{R}$ and the data to be learned are like the ones in 1.

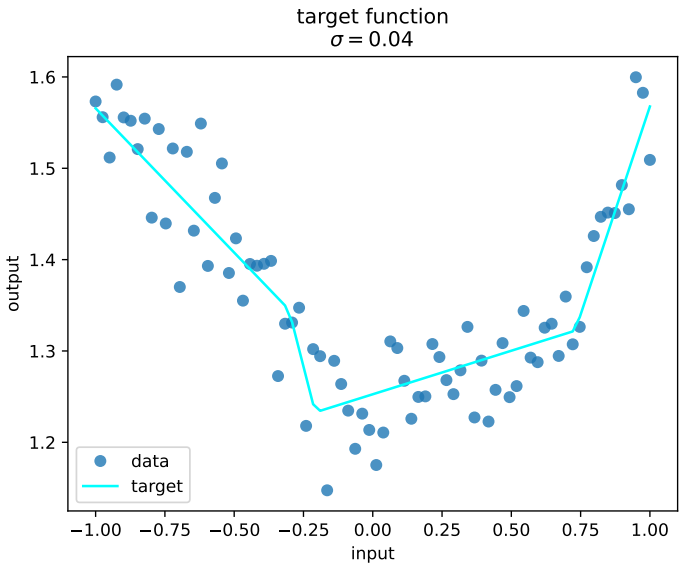

FIGURE 1 – Example target function and dataset

Let m be the number of neurons in the hidden layer, and n the number of samples in the dataset. We will see that with the specific type of networks and datasets used here, even when $m > n$ (which implies that the number of parametrers of the network is larger than n, since it is of order at least $\mathcal{O}(dm)$, d being the input dimension), no overfitting happens. This is related to the "interpolation regime", a contemporary topic in machine learning research. You can see an exmaple of such a phenomena in figure 2.

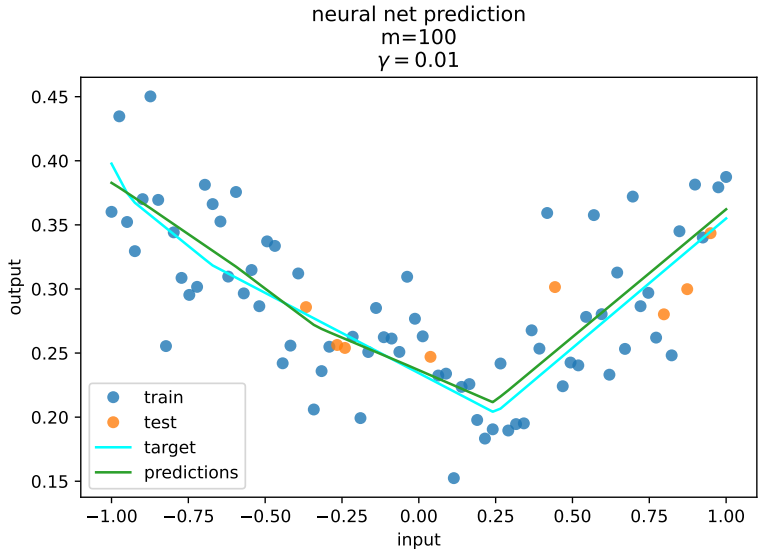

FIGURE 2 – Although the network has a high capacity, it does not overfit the data.

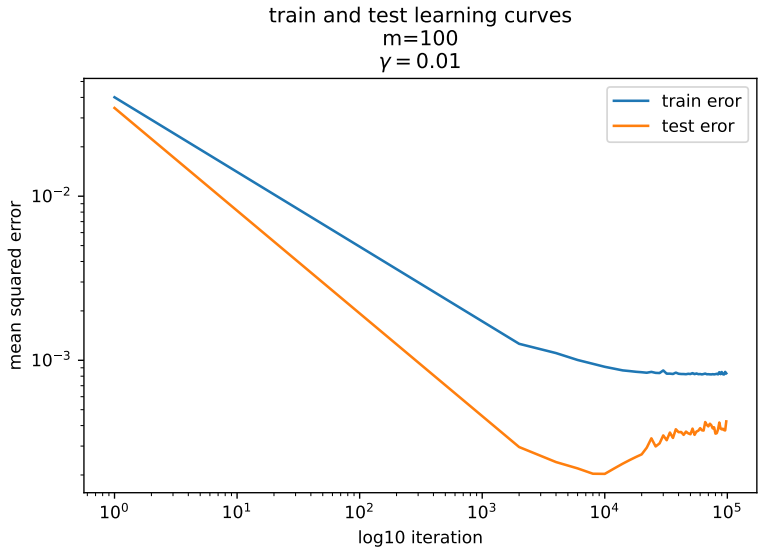

FIGURE 3 – Learning curves, same network as in figure 2

1.1 Definition of the neural network

We use the following architecture :

— $\mathcal{X} = \mathbb{R}$, $\mathcal{Y} = \mathbb{R}$, the loss is the squared loss.

— In order to add an intercept we add a component to the inputs $x \in \mathbb{R}^d$ and build a vector $(x, 1) \in \mathbb{R}^{d+1}$. The same operation is applied to the hidden layer.

— The hidden layer contains m neurons. The matrix w_h containing the weigths between the input layer and the hidden layer in $\mathbb{R}^{d+1,m}$ (the $d+1$ comes from the previous point)

— nonlinearity : ReLU (noted σ). Also applied to the output.

— The output of the network is a single real number, hence the matrix θ containing the weights between the hidden layer and the output layer is a vector in $\mathbb{R}^{m+1}$.

— Let $h \in \mathbb{R}^{m+1}$ be the vector representing the output of the hidden layer. Then, the output of the neural network writes :

$$\hat{y} = f(x) = \sigma(\langle \theta, h \rangle) \tag{1}$$

1.1.1 *Ressources*

Useful ressources :

— https://playground.tensorflow.org/

— http://yann.lecun.com/exdb/publis/pdf/lecun-98b.pdf

— [LeCun et al., 1998]

1.2 Implementation

Train neural networks with varying number m of neurons in the (unique) hidden layer in order to observe the simplicity bias.

You will need to compute the gradients of the loss function with respect to the network parameters. There will be a gradient with respect to θ, noted $\nabla_\theta l(\theta, w_h)$, and a gradient with respect to w_h, noted $\nabla_{w_h} l(\theta, w_h)$.

You will also need to edit the main SGD algorithm.

For the implementation, you are free to either compute the gradients manually and write the computations in numpy, or to use automatic differentiation with libraries like torch of tensorflow.

1.2.1 *Python files*

The template files are more useful if you use the numpy / manual approach.

— **main.py** : main file, runs SGD on the neural network. You need to fix the algorithm.

— **utils.py** : computes the forward passes and the gradient computations. You need to fix the computations.

— **generate_data.py** : generates the data, you don't need to edit it.

1.2.2 *Initialization*

You can use the following type of initialization.

— θ is initialized uniformly in $[-\frac{1}{\sqrt{m}}, -\frac{1}{\sqrt{m}}]^{m+1}$

— Each column of w_h, that belongs to $\mathbb{R}^2$, is initialized on the sphere of radius $\frac{1}{\sqrt{m}}$.

1.2.3 *Derivatives*

As the derivative of ReLU, you can use the heaviside function.
https://numpy.org/doc/stable/reference/generated/numpy.heaviside.html
https:/numpy.org/doc/stable/reference/generated/numpy.maximum.html

1.2.4 *Learning rate*

You will need to experiment with the learning rate γ in order to observe learning and the simplicity bias. As the learning algorithm and dataset generation are stochastic, you might observe different outputs.

Note that learning does not always happen, depending on the hyperparameters and dataset. In figure 4, the network has not been able to approximate the target function.

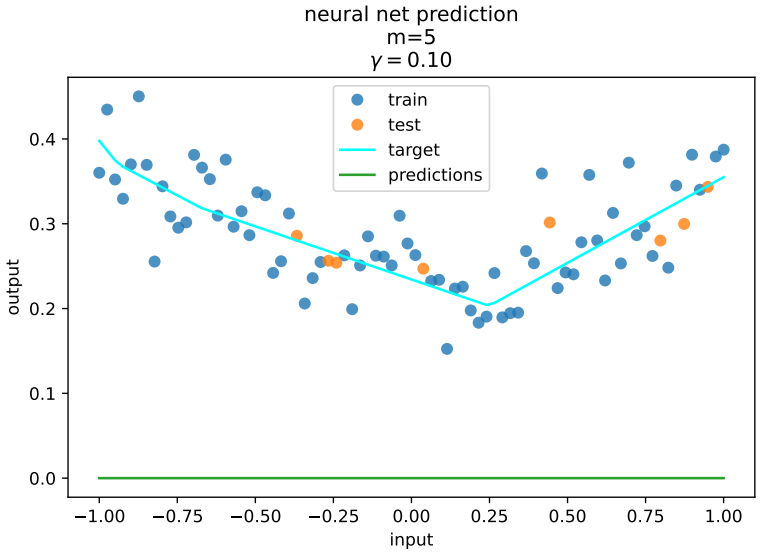

FIGURE 4 – This network has not been able to learn the data.

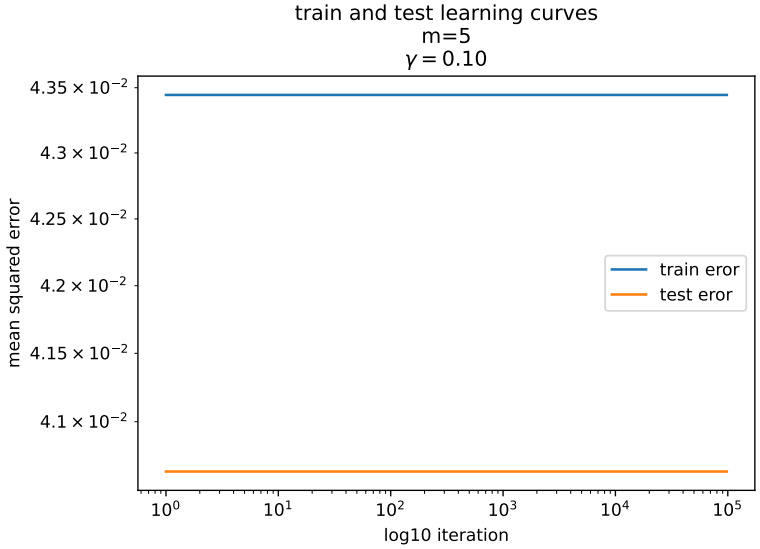

FIGURE 5 – Same network as in 4

1.3 Conclusion

These neurons tend to not overfit, although some of them have a number of parameters way larger than of the minimal space containing the target function.

See more in this post : https://francisbach.com/quest-for-adaptivity/

2 QUANTITATIVE EVALUTATION OF THE BENEFIT OF RIDGE REGRESSION

The goal of this exercise is to have a more concrete representations of situations where Ridge regression is useful. We will state some theoretical resutlts, and then find datasets for which these results apply.

2.1 Reminders of the theoretical results

We keep the same statistical setting as before (fixed design, linear model). We have seen in the previous classes that the excess risk is $\frac{\sigma^2 d}{n}$ for OLS (Ordinary least squares)

Definition 1. Ridge regression estimator
It is defined as

$$\hat{\theta}_\lambda = \underset{\theta \in \mathbb{R}^d}{\arg\min} \left(\frac{1}{n} \|y - X\theta\|_2^2 + \lambda \|\theta\|_2^2 \right) \tag{2}$$

Proposition. *The Ridge regression estimator is unique even if $X^T X$ is not inversible and is given by*

$$\hat{\theta}_\lambda = \frac{1}{n} (\hat{\Sigma} + \lambda I_d)^{-1} X^T y$$

Proposition. *We assume the linear model, with fixed design setting, with a Bayes estimator of θ^* and a noise with a variance of σ^2. Then, the ridge regression estimator has the following excess risk :*

$$E[R(\hat{\theta}_\lambda] - R^* = \lambda^2 \theta^{*T} (\hat{\Sigma} + \lambda I_d)^{-2} \hat{\Sigma} \theta^* + \frac{\sigma^2}{n} tr[\hat{\Sigma}^2 (\hat{\Sigma} + \lambda I_d)^{-2}] \tag{3}$$

Comments :

— We observe a bias / variance decomposition.

— We consider the bias term B :

$$B = \lambda^2 \theta^{*T} (\hat{\Sigma} + \lambda I_d)^{-2} \hat{\Sigma} \theta^* \tag{4}$$

- — The bias B increases when λ increases. It is an approximation error and does not depend on n.
- — When $\lambda = 0$ and $\hat{\Sigma}$ is invertible (which corresponds to OLS), $B = 0$.
- — When $\lambda \to +\infty$, $B \to \theta^{*T} \hat{\Sigma} \theta^*$.

— We consider the variance term V :

$$V = \frac{\sigma^2}{n} tr[\hat{\Sigma}^2 (\hat{\Sigma} + \lambda I_d)^{-2}] \tag{5}$$

- — The variance V decreases when λ increases. It is an estimation error and depends on n
- — When $\lambda = 0$ and $\hat{\Sigma}$ is invertible (which corresponds to OLS), $V = \frac{\sigma^2 d}{n}$.
- — When $\lambda \to +\infty$, $V \to 0$.
- — When $n \to +\infty$, $V \to 0$.

A natural question is whether it is possible to have a lower excess risk with Ridge regression than with OLS, which means an excess risk smaller than $\frac{\sigma^2 d}{n}$. We admit the following proposition.

Proposition. *With the choice*

$$\lambda^* = \frac{\sigma \sqrt{tr(\hat{\Sigma})}}{\|\theta^*\|_2 \sqrt{n}} \tag{6}$$

then

$$E[R(\hat{\theta}_\lambda] - R^* \leqslant \frac{\sigma\sqrt{tr(\hat{\Sigma})}\|\theta^*\|_2}{\sqrt{n}} \tag{7}$$

with

$$\hat{\Sigma} = \frac{1}{n}X^T X \in \mathbb{R}^{d,d} \tag{8}$$

Hence, the convergence to 0 in OLS is in $\frac{1}{n}$, while it is in $\frac{1}{\sqrt{n}}$ for the ridge. However, for the ridge regression, the dependence in the noise if in σ, whereas it is σ^2 for OLS. Which one is preferrable will depend on the value of the constants, and will not necessary be the "fast" rate in $\mathcal{O}(\frac{1}{n})$.

Conclusion : if, for a given setting, we have

$$\frac{\sigma\sqrt{tr(\hat{\Sigma})}\|\theta^*\|_2}{\sqrt{n}} \leqslant \frac{\sigma^2 d}{n} \tag{9}$$

then we know that there exists values for λ (such as λ^*), such as the Ridge regression estimator has better generalization properties than OLS.

2.1.1 Exercise and simulation

Find a setting (statistical values, dataset) for which equation 9 is satisfied, and verify the result with a simulation where you compare the test error of OLS and that of Ridge regression with a good choice of the regularization parameter λ.

RÉFÉRENCES

[LeCun et al., 1998] LeCun, Y. A., Bottou, L., Orr, G. B., and Müller, K.-R. (1998). Efficient BackProp. pages 9–48.