Semester projet report

Ylann Rouzaire

May 29, 2020

This semester project can be separated in two distinct parts : the investigation of the dynamics of learning in a Teacher/Student Kernel Regression framework on one hand and the investigation of the behaviour of a noiseless hard-margin SVM classifier in presence of a gap at the interface between the 2 classes.

Each work will constitute a section of this report, organised as follows : in a first part, I shall explain the mathematical setting of the framework. The second part is the actual code and its explanation, available publicly on GitHub. The README.md if self-contained so it is not necessary to duplicate the information. Then come the figures and main numerical results. The last part sums up the analytical developments aiming at proving the scaling laws observed numerically.

This project was supervised by Matthieu Wyart and Stefano Spigler.

Contents

1 Dynamics Kernel Regression

1.1 Settings

Kernel : A kernel is a function from $\mathbf{R}^d$ x $\mathbf{R}^d$ to $\mathbf{R}$. We will only deal with isotropic translation-invariant kernel $k(x, x') = k(||x - x'||)$ of the Matérn family :

$$f_\nu(h) = \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\sqrt{2\nu} \frac{h}{\rho} \right)^\nu K_\nu \left(\sqrt{2\nu} \frac{h}{\rho} \right), \tag{1}$$

where Γ is the Euler Gamma function, K_ν is the modified Bessel function of the second kind, $\rho = 1$ is the characteristic length scale and $\nu > 0$ is the smoothness parameter (the larger the smoother, $\nu = \infty \leftrightarrow$ Gaussian kernel).

Teacher/Student framework : The idea is the following : the Teacher generates data according to its own "law" K_T. A subset of size P of that data called the *training set* is passed to the Student. From it and thanks to its "law" K_S, the Student has to infer the underlying "law" K_T. Its performance is measured by the test error ϵ , defined as a quadratic loss between the prediction of the Student and true values at several *new/unseen* points generated by the Teacher.

$$\epsilon = \epsilon\left(P, \nu_T, \nu_S\right) = \mathrm{E}_T \, \frac{1}{P_{test}} \sum_{\mu=1}^{P_{test}} |\hat{Z}^S(x_\mu) - Z^T_{test}(x_\mu)|^2 \tag{2}$$

where E_T is the expectation over the Teacher random process.

In this project, Teacher kernels are chosen among the Matérn family (1) and the Student kernel will always be the Laplace kernel :

$$\text{Laplace}(x, x') = \text{Matérn}\left[\nu = 1/2\right](x, x') = \exp(-||x - x'||) \tag{3}$$

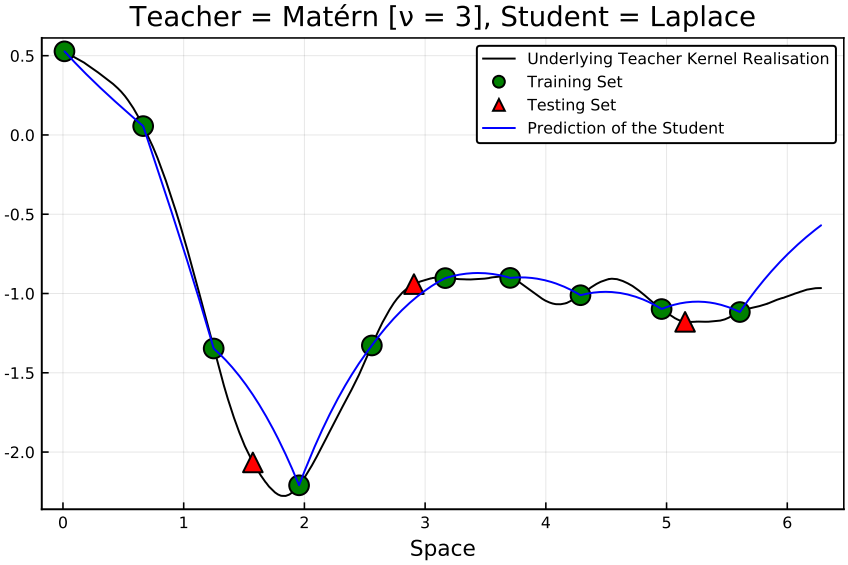


Figure 1: Illustration of the Teacher/Student framework in the context of Matérn kernel Regression. The test error ϵ is the sum over all testing set of the square distance between the red triangles and value of the blue curve at the same x coordinate.

Generation of the data by the Teacher kernel :

$$Z^T \sim \mathcal{N}(0, K_T) \tag{4}$$

so that

$$\mathrm{E}\ Z^T(x) = 0 \text{ and } \mathrm{E}\ Z^T(x)Z^T(x') = K_T(x, x') \tag{5}$$

Prediction of the Student kernel :

$$\hat{Z}^S(x_0)(t) = \sum_{\mu=1}^{P} a_\mu(t) K_S(x_\mu, x_0) = a(t) \cdot k_S(x_0) \tag{6}$$

where $a(t)$ is a vector of weights of size P changing over time during the learning procedure. The different optimisation routines implemented in the code are described in the README.md file of the project.

1.2 Features of the code

The Julia code can be found here : `github.com/Rouzaire/Kernel_Regression`.
The README.md file is self-explanatory and contains general information on the code and its structure. The code itself is commented and adds more precise details.

1.3 Results and Figures

The first checkup of the code was to recover the scaling for ϵ versus P found in [1] :

$$\epsilon \sim P^{-\beta}, \text{ with } \beta = \frac{1}{d} \min(\alpha_T - d, 2\alpha_S) \tag{7}$$

My main contribution at this stage is that my code is faster than what was previously coded.

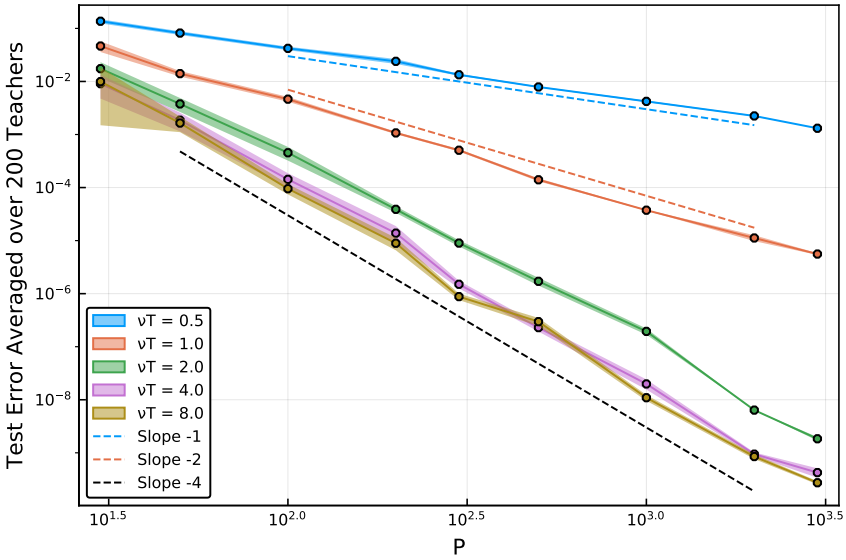
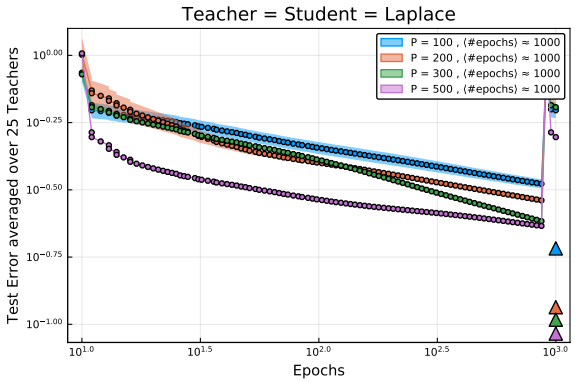


Figure 2: Learning curves for several Teacher kernels, here for $d = 1$. Student kernel is fixed to Laplace, cf. eq (3). One recovers the scaling predicted in [1].

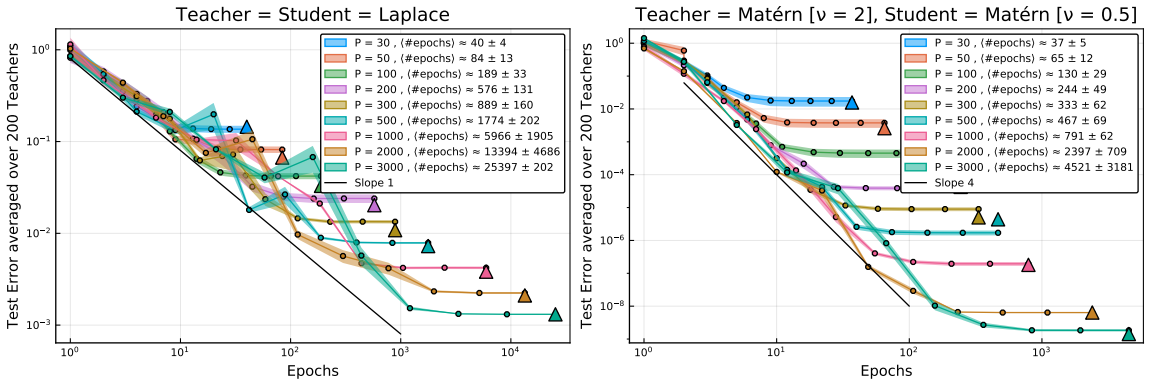
Then I could investigate the dynamics of the test error during the optimization procedure. The conclusion is that for the Gradient Descent algorithm, the dynamics is very monotonic and exhibits no overfit : the test error is always decreasing and its limit is the exact test error, plotted with triangles in Fig. (3) and (4) :



(a)

(b)

Figure 3: Optimization : Gradient Descent.



(a) (b)

Figure 4: Optimization : Conjugate Gradient Descent.

One can notice that for intermediate time, both for GD and CGD, the test error decays in a powerlaw. This is part of the analytic work in progress.

1.4 Analytical developments

In [3], the authors suggest that the test error can be decomposed in a sum over the modes ρ of the target function of error per mode. They provide with an expression for the error per mode : NB : $\Lambda \geq 0$ is the rigde parameter.

$$E = \sum_{\rho} E_\rho, \text{ where } E_\rho(p) = \frac{\langle \bar{\omega_\rho^2} \rangle}{\lambda_\rho} \left(\frac{1}{\lambda_\rho} + \frac{p}{\Lambda + t(p)} \right)^{-2} \left(1 - \frac{p\,\gamma(p)}{(\Lambda + t(p))^2} \right)^{-1} \tag{8}$$

$$t(p) = \sum_{\rho} \left(\frac{1}{\lambda_\rho} + \frac{p}{\Lambda + t(p)} \right)^{-1} \tag{9}$$

$$\gamma(p) = \sum_{\rho} \left(1 - \frac{p\,\gamma(p)}{(\Lambda + t(p))^2} \right)^{-2} \tag{10}$$

Note that these equations are the exact errors, there is no notion of dynamics here. We wanted to check whether for large P, these equations boil down to the scalings already found in [1] for the ridgeless case. The answer is yes and here are some intermediate results :

Without rigde : $\Lambda = 0$, for $p \gg 1$

- $t(p) \sim p^{-s} \quad s = \frac{2-\theta}{\theta-1}$
- $\gamma(p) \sim p^{-r} \quad r = \frac{3-\theta}{\theta-1}$
- $E(p) \sim p^{-\beta} \quad \beta = \frac{1}{d} \min(\alpha_T - d, 2\,\alpha_S)$

With rigde : $\Lambda \sim \mathcal{O}(1)$, for $p \gg 1$

- $t(p) \sim p^{-s} \quad s = 2 - \theta$
- $\gamma(p) \sim p^{-r} \quad r = 3 - \theta$
- $E(p) \sim p^{-\beta} \quad \beta = \frac{1}{\alpha_S} \min(\alpha_T - d, 2\,\alpha_S)$

NB : $\alpha = d + 2\,\nu > d$ so if the data is not noisy, a regression with ridge will never be optimal/better than a ridgeless regression.

2 Kernel Classification with a gap at the interface

2.1 Settings

We investigate the learning curves of a supervised *2-class hard-margin kernel classification* task when the data present many invariant. In this project, the data labels will only depend on the first component x_1 of $\vec{x} \in \mathrm{R}^d$, all other dimensions are irrelevant for labelling :

$$y(\vec{x}) = \mathrm{sgn}\ x_1$$

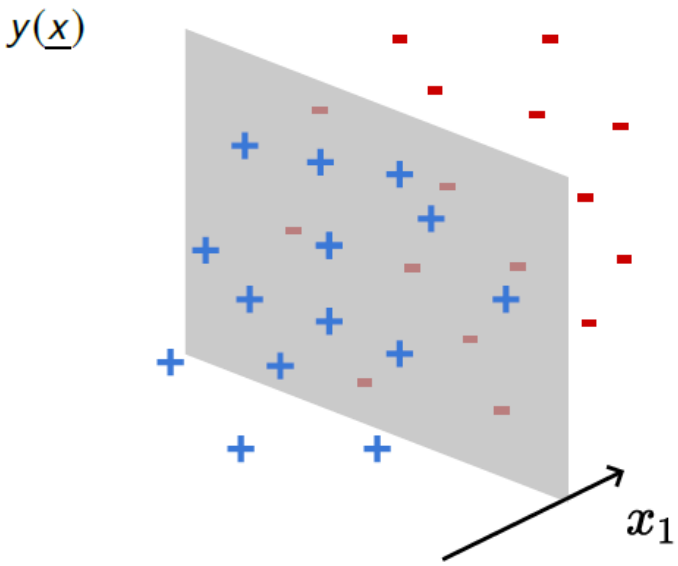


Figure 5: The label function is a step function. Image from [4].

2.2 Features of the code

The Julia code can be found here : `github.com/Rouzaire/Kernel_Classification`.
The README.md file is self-explanatory and contains general information on the code and its structure. The code itself is commented and adds more precise details.

2.3 Results and Figures

2.4 Analytical developments

Work in progress

References

[1] Spigler, Geiger and Wyart
Learning Curves of Kernel Methods, empirical data vs. Teacher/Student paradigm

[2] Jacot, Gabriel and Hongler
Neural Tangent Kernel Convergence and Generalization in Neural Networks

[3] Bordelon, Canatar and Pehlevan
Spectrum dependent learning curves in kernel regression and wide neural networks

[4] Paccolat, Spigler and Wyart
How isotropic kernels learn simple invariants