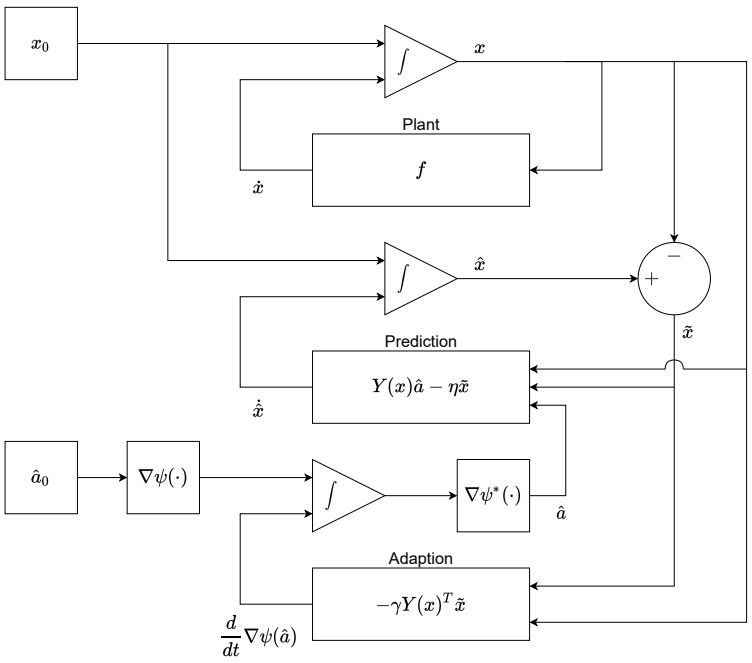
Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Engineering
Department of Mechanical and Industrial Engineering

Johannes Gierctsky Nilssen

Implicit Regularization in Machine Learning for Adaptive Control

Master's thesis in Engineering and ICT
Supervisor: Olav Egeland
June 2021



NTNU
Norwegian University of
Science and Technology

Johannes Gierctksky Nilssen

Implicit Regularization in Machine Learning for Adaptive Control

Master's thesis in Engineering and ICT
Supervisor: Olav Egeland
June 2021

Norwegian University of Science and Technology
Faculty of Engineering
Department of Mechanical and Industrial Engineering

NTNU
Norwegian University of
Science and Technology

Abstract

Modern machine learning utilizes highly overparameterized models that are able to perfectly fit the training data while still performing well on the test set. New research pins this ability to the first order gradient methods used to optimize the networks. Both theoretical and empirical research demonstrate that the optimization methods have *implicit bias* effectively regularizing the learned models. In a recent article, Boffi and Slotine explores how the implicit regularization phenomenon in machine learning can be transferred to adaptive control with stability guarantees. The current thesis examines the similarities between machine learning and adaptive control. Dynamic prediction of a Hamiltonian system is used to demonstrate how gradient based adaption laws impose regularization on the learned model.

Sammendrag

Moderne maskinlæring bruker svært overparameteriserte modeller som kan tilpasses treningsettet feilfritt og fortsatt prestere godt på testsettet. Ny forskning knytter denne evnen til gradientmetodene brukt for å optimalisere nettverkene. Både teoretisk og empirisk forskning viser at optimeringsmetodene regulariserer den lærte modellen implisitt. I en nylig artikkel utforsker Boffi og Slotine hvordan implisitt regularisering i maskinlæring kan overføres til adaptiv regulering med stabilitetsgaranti. Denne oppgaven undersøker likheter mellom maskinlæring og adaptiv regulering. Dynamisk prediksjon av et Hamiltonsk system brukes for å illustrere hvordan gradientmetoder implisitt regulariserer de lærte parametrene.

Contents

Chapter 1

Introduction

The field of adaptive control emerged in the 1950s to design autopilots for the high-performance fighter jets of the cold war era. High-performance aircraft operate at varying speeds and altitudes which require adaption in the controller. Since the 1950s, adaptive control has grown into a mature field with a rich set of techniques [1].

Machine learning has the last decade seen rapid development with the increasing availability of large datasets and the increased computational power of modern hardware. Significant research has been put into understanding the inner workings of neural networks to understand why they perform so well. Modern machine learning methods use highly overparameterized networks which interpolate the training data while still generalizing well on test sets. This ability has been found to partly stem from the optimization techniques used to optimize the networks. In two recent papers Gunesekar[2] and Azizan [3] proves that the optimization methods used in training impose implicit regularization on the learned parameters. In the present thesis parallels are drawn between neural networks and regression based adaptive control and it is demonstrated how methods may be transferred between the two subject areas.

Firstly, the thesis provides an introduction to relevant theory in optimization and regression necessary to understand simple neural networks. Next, we shall then cover some adaption methods in adaptive control similar to the optimization methods used to train neural networks. Lastly we introduce a Hamiltonian system inspired by an experiment by Boffi and Slotine [4] which will be used to empirically demonstrate the implicit regularization imposed by the adaption laws.

1.1 A Brief Note on Notation

Lowercase Latin letters are used for vectors, uppercase Latin letters for matrices and Greek lowercase letters for scalars. Calligraphic uppercase Latin letters are usually reserved for sets. Some notable exceptions are t used for scalar time, n, m, k, i used as scalars for counting and indexing. $\mathcal{R}(\cdot)$ for range of a matrix and $\mathcal{N}(\cdot)$ for null space of a matrix.

Chapter 2

Background

2.1 Normed Spaces

A norm is a function $\|\cdot\| : \mathbb{R}^n \to \mathbb{R}^+$ satisfying the following properties

1. Nonnegativity:
$$\|x\| \geq 0, \ \forall x \in \mathbb{R}^n \tag{2.1}$$
2. Definitness:
$$\|x\| = 0 \implies x = 0 \tag{2.2}$$
3. Homogenity:
$$\|\alpha x\| = |\alpha|\|x\|, \ \forall \alpha \in \mathbb{R}, x \in \mathbb{R}^n \tag{2.3}$$
4. The triangle inequality:
$$\|x + y\| \leq \|x\| + \|y\|, \ \forall x, y \in \mathbb{R}^n \tag{2.4}$$

The ℓ_p-norms are a set of norms used in the study of finite-dimensional vector spaces like $\mathbb{R}^n$. The norm is defined as

$$\|x\|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{\frac{1}{p}}, \ p \geq 1 \tag{2.5}$$

Some notable ℓ_p-norms are the ℓ_1-norm which is the sum of absolute values

$$\|x\|_1 = \sum_{i=1}^{n} |x_i| \tag{2.6}$$

The ℓ_2 norm called the Euclidean norm which is a generalization of the Pythagorean theorem

$$\|x\|_2 = \sqrt{\sum_{i=1}^{n} x_i^2} \tag{2.7}$$

And the ℓ_∞-norm also called the max norm

$$\|x\|_\infty = \max(x_1, x_2, \ldots, x_n) \tag{2.8}$$

Hölder's inequality is an important inequality in the study of spaces where the ℓ_p-norms are defined and is defined as

$$\sum_{i=1}^{n} |x_i y_i| \leq \|x\|_p \|y\|_q, \ \frac{1}{p} + \frac{1}{q} = 1 \tag{2.9}$$

Chapter 3

Convex Optimization

This chapter serves as a brief introduction to convex optimization. Optimization methods are used to train neural networks and are therefore an important factor to the performance of the network. First we will introduce some properties of convex functions which we will later use to derive some optimization techniques used in machine learning.

3.1 Convexity and Its Properties

3.1.1 Convex Functions

A set $\mathcal{C}$ is *convex* if

$$\theta x + (1-\theta)y \in \mathcal{C},\ \forall x, y \in \mathcal{C},\ 0 \leq \theta \leq 1 \tag{3.1}$$

A function f is convex if its domain is a convex set and the following inequality is satisfied

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta)f(y),\ 0 \leq \theta \leq 1 \tag{3.2}$$

The function is *strictly convex* if strict inequality holds.

A function being convex is equivalent with the *epigraph* of the function $\operatorname{epi} f$ being a convex set. The epigraph is the area above a function $f : \mathcal{X} \to \mathcal{Y}$ defined as

$$\operatorname{epi} f = \{(x, y) \in \mathcal{X} \times \mathcal{Y} \mid y \geq f(x)\} \tag{3.3}$$

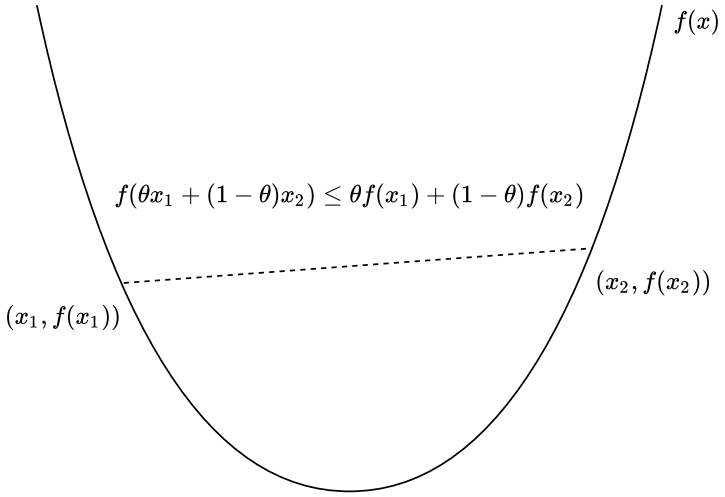


Figure 3.1: The definition of convexity. The chord between two points x_1, x_2 on function f lies above the function.

3.1.2 First Order Conditions

Another important inequality is the tangent inequality which can be proved for differentiable convex functions [5]. Starting with the definition of convexity

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta)f(y),\ 0 \leq \theta \leq 1 \tag{3.4}$$

$$f(x) \geq f(y) + \frac{f(y + \theta(x-y)) - f(y)}{\theta} \tag{3.5}$$

The limit $\lim \theta \to 0$ gives us, by the definition of the derivative

$$f(x) \geq f(y) + f'(y)(x-y) \tag{3.6}$$

Now we must prove it for the general case $f : \mathbb{R}^n \mapsto \mathbb{R}$ Let $x, y \in \mathbb{R}^n$. Define g as f restricted to the line passing between the two points x, y

$$g(\theta) = f(\theta x + (1-\theta)y),\ 0 \leq \theta \leq 1 \tag{3.7}$$

$$g'(\theta) = \langle \nabla f(\theta x + (1-\theta)y), (x-y) \rangle \tag{3.8}$$

Because f is convex g is also convex which by 3.6 gives us

$$g(1) \geq g(0) + g'(0)(1-0) \tag{3.9}$$

$$\implies f(x) \geq f(y) + \langle \nabla f(y), (x-y) \rangle \tag{3.10}$$

This is an important inequality proving that the tangent of a convex function is a global underestimator of the function. The gradient defines a supporting hyperplane to the epigraph. If the tangent has a slope of zero at a point, and it is a global underestimator, then the tangent point must be a global minimum.

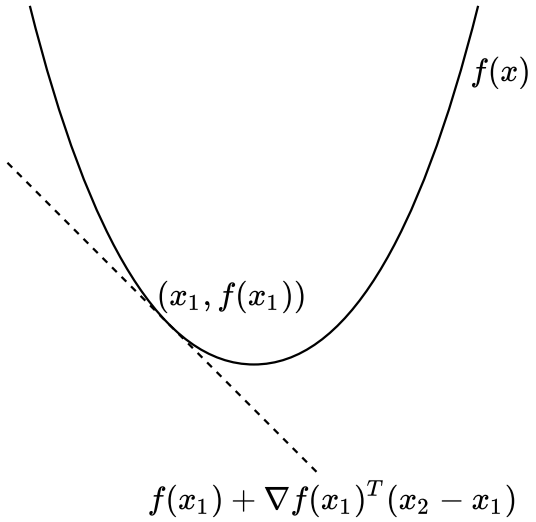


Figure 3.2: The tangent inequality for convex functions.

This inequality can be used to find a necessary condition for a global minimizer x^*. If x^* is a global minimizer we have

$$f(x) \geq f(x^*),\ \forall x \neq x^* \tag{3.11}$$

From 3.10 it follows

$$\nabla f(x^*) = 0 \tag{3.12}$$

A differentiable function f is α *strongly convex* with $\alpha > 0$ if

$$\langle \nabla f(x) - \nabla f(y), x - y \rangle \geq \alpha \|x - y\|^2 \tag{3.13}$$

for any norm $\|\cdot\|$.

3.1.3 Second Order Conditions

Consider a twice differentiable function $f : \mathcal{X} \to [-\infty, \infty]$ with the Hessian $\nabla^2 f$ defined on the entire domain of f. The function f is convex if its domain is convex and the Hessian is positive semi definite

$$\nabla^2 f(x) \geq 0,\ \forall x \in \mathcal{X} \tag{3.14}$$

The Hessian can also be used to show α-strong convexity

$$\nabla^2 f(x) \geq \alpha I, \forall x \tag{3.15}$$

3.1.4 Convexity of Norms

All norms are convex, which is easily verifiable. We begin by applying the triangle inequality to the left side

$$\|\theta x + (1-\theta)y\| \leq \|\theta x\| + \|(1-\theta)y\| \tag{3.16}$$

Then by the homogenity condition

$$\|\theta x + (1-\theta)y\| \leq \theta\|x\| + (1-\theta)\|y\| \tag{3.17}$$

Here the absolute value can be removed because $0 \leq \theta \leq 1$. It is then seen from the definition of a convex function that all norms are convex.

3.1.5 Convex Conjugate Functions

Convex conjugate functions are an important part of the mirror descent method which we will introduce later.

The concept of duality builds on the idea of paired spaces [6]. A pairing of two real linear spaces $\mathcal{X}$ and $\mathcal{Y}$ is a real-valued bilinear form $\langle x, y\rangle$ which behaves like a inner product except that $x \in \mathcal{X}$ and $y \in \mathcal{Y}$. For each y this gives us the function on $\mathcal{X}$

$$\langle \cdot, y\rangle : x \to \langle x, y\rangle \tag{3.18}$$

Then for each x

$$\langle x, \cdot\rangle : y \to \langle x, y\rangle \tag{3.19}$$

For real, finite dimensional spaces, the pairing is equivalent to the dot product. This is the case for all problems of concern in this report.

$$\implies \langle x, y\rangle = x^T y \tag{3.20}$$

A closed half-space is defined as $\{x \mid \langle x, y\rangle \leq \beta\}$ where $\beta \in \mathbb{R},\ y \neq 0$. The subset $\mathcal{C} \subseteq \mathcal{X}$ which is a closed convex set is then defined as the intersection of a collection of such closed half-spaces on $\mathcal{X}$.

Now consider the function $f : \mathcal{X} \to \mathbb{R}$. We define the conjugate function $f^* : \mathcal{Y} \to \mathbb{R}$ which is defined as the smallest set $\mathcal{C}$ that approximates the epigraph of f with closed half-spaces. The conjugate of the function f is then defined as

$$f^*(y) = \sup_{x \in \mathcal{X}} \{\langle x, y\rangle - f(x)\} \tag{3.21}$$

If f is convex and continuously differentiable we can solve for the supremum by setting the gradient with respect to x equal to zero

$$0 = \nabla_x \left(\langle x, y\rangle - f(x)\right) \tag{3.22}$$

which results in

$$\nabla f(x) = y \tag{3.23}$$

We also have that

$$f(x) = \sup_{y \in \mathcal{Y}} \{\langle x, y \rangle - f^*(y)\} \tag{3.24}$$

which means the supremum is achieved when

$$\nabla f^*(y) = x \tag{3.25}$$

If f and f^* are strictly convex their gradients are unique at each point. The two gradients then form a one-to-one correspondence between the primal space $\mathcal{X}$ and the gradient space $\mathcal{Y}$. In the strictly convex case we have that

$$\nabla f^*(\nabla f(x)) = x \tag{3.26}$$

$$\nabla f(\nabla f^*(y)) = y \tag{3.27}$$

From the definition of the conjugate function we obtain *Fenchel's inequality*

$$f^*(y) = \sup_{x \in \mathcal{X}} \{\langle x, y \rangle - f(x)\} \tag{3.28}$$

$$\geq \langle y, x \rangle - f(x) \tag{3.29}$$

$$\implies f(x) + f^*(y) \geq \langle x, y \rangle, \ \forall x, y \tag{3.30}$$

3.2 Optimization Methods

In this section we introduce some numerical methods for unconstrained optimization called *descent methods*. Descent methods describe a sequence x_k indexed by the subscript $k = 1, 2, ...$

$$x_{k+1} = x_k + \gamma_k \Delta x_k \tag{3.31}$$

where Δx_k is the *step direction* and $\gamma_k > 0$ the *step size*. Because we want to descent to the minimum of the function we impose the condition

$$f(x_k) < f(x_{k+1}) \tag{3.32}$$

For convex functions we have that

$$(\nabla f(x_k))^T \underbrace{(x_{k+1} - x_k)}_{=\Delta x_k} \geq 0 \implies f(x_{k+1}) \geq f(x_k) \tag{3.33}$$

Therefore the step direction Δx_k must satisfy

$$(\nabla f(x_k))^T \Delta x_k < 0 \tag{3.34}$$

3.2.1 Gradient Descent

The simplest way to satisfy the above inequality is by setting the step direction equal to the negative gradient $\Delta x = -\nabla f(x)$ this leads to the *gradient descent* method.

$$x_{k+1} = x_k - \gamma_k \nabla f(x_k) \tag{3.35}$$

Gradient descent is simple yet effective. The method exhibits approximately linear convergence for convex functions, but is sensitive to the condition number of the Hessian $\nabla^2 f(x)$.

Algorithm 1: Gradient Descent

initialize x_0;
repeat
  $x_{k+1} \leftarrow x_k - \gamma_k \nabla f(x_k)$;
until $\|x_{k+1} - x_k\|_2 \leq$ *tolerance*;
return x_k;

3.2.2 Bregman Divergence

The Bregman divergence [7] measures the approximation error of the first order Taylor approximation of $\psi(\cdot)$ around x_2 at x_1.

$$d_\psi(x_1 \| x_2) = \psi(x_1) - \psi(x_2) - \langle \nabla\psi(x_2), x_1 - x_2 \rangle \tag{3.36}$$

Because the tangent of a convex function is a global underestimator the Bregman divergence is positive semidefinite if ψ is convex, but it is not guaranteed to be symmetric and does not satisfy the triangle inequality. The choice of potential ψ makes the Bregman divergence a more general way to measure distance than the Euclidean norm which is often used in optimization.

3.2.3 Mirror Descent

The mirror descent algorithm was first presented by Nemirovsky and Yudin [8]. Mirror descent uses the gradient of a strictly convex and continuously differentiable potential function $\nabla\psi$ to transform the problem from the primal space $\mathcal{X}$ to the gradient space $\mathcal{Y}$ which we will call the *mirrored domain*. Because ψ is strictly convex and continuously differentiable the gradient is unique at each point in $\mathcal{X}$ and the conjugate gradient at every point in $\mathcal{Y}$.

Let $\mathcal{X}$ be a real Banach space with norm $\|\cdot\|$. And let $\mathcal{Y} = \mathcal{X}^*$ be a real Banach space with norm $\|\cdot\|_*$. It is assumed that $\mathcal{X}$ is reflexive such that $(\mathcal{X}^*)^* = \mathcal{X}$.

In the problems of concern to this thesis we always have that $\mathcal{X} = \mathcal{Y} = \mathbb{R}^n$. We still keep the distinction in this section to make it clear when we operate in the primal and mirrored domain.

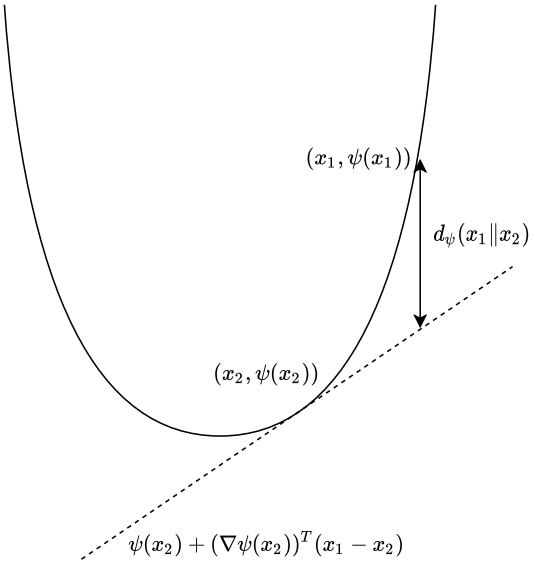


Figure 3.3: The Bregman divergence between x_1 and x_2 on the convex function ψ

Consider the potential $\psi : \mathcal{X} \to \mathbb{R}$, with the following properties

$$\psi(x) \geq 0 \tag{3.37}$$
$$\psi(0) = 0 \tag{3.38}$$
$$\psi \text{ is strictly convex} \tag{3.39}$$
$$\nabla\psi(x) \text{ is uniformly continuous and bounded } \forall x \in \mathcal{X} \tag{3.40}$$

Consider the problem minimizing the objective function $f : \mathcal{X} \to \mathbb{R}$. We define a descent method in the mirrored domain $\mathcal{Y}$ with the step direction Δy equal to the negative gradient of the objective function $f(x) = f(\nabla\psi^*(y))$

$$\Delta y = -\nabla f(\nabla\psi^*(y)) \tag{3.41}$$

which defines the mirror descent method

$$y_{k+1} = y_k - \gamma_k \nabla f(\nabla\psi^*(y_k)) \tag{3.42}$$

The mapping $\nabla\psi^*$ carries $\mathcal{Y}$ into $\mathcal{X}$, and ∇f maps $\mathcal{X}$ back to $\mathcal{Y}$. The method can also be defined with respect to x

$$\nabla\psi(x_{k+1}) = \nabla\psi(x_k) - \gamma_k \nabla f(x_k) \tag{3.43}$$

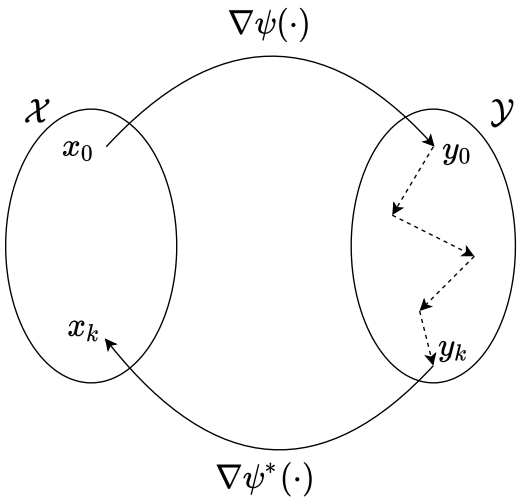


Figure 3.4: Mirror descent. Notice how the optimization steps are taken in the mirrored domain.

Algorithm 2: Mirror Descent

initialize x_0;
$y_0 \leftarrow \nabla\psi(x_0)$;
repeat
  $y_{k+1} \leftarrow y_k - \gamma_k \nabla f(\nabla\psi^*(y_k))$;
until $\|x_{k+1} - x_k\|_2 \leq$ *tolerance*;
$x_k \leftarrow \nabla\psi^*(y_k)$;
return x_k

The ability to choose mirror function ψ gives us a wealth of alternatives for optimizing functions and the choice of mirror can significantly impact the convergence properties of the method. By choosing the potential $\psi = \frac{1}{2}\|\cdot\|_2^2$ we get the gradient $\nabla\psi(x) = x$. This shows that gradient descent is a special case of mirror descent.

It should be noted that the optimization movement takes place in the mirrored domain. The motion on $\mathcal{X}$ is the projection of the main movement on $\mathcal{Y}$. This fact is overlooked in gradient descent because the potential results in the unit transformation $\nabla\psi(x) = x$ resulting in $\mathcal{X}$ being identified with $\mathcal{Y}$ giving the illusion of the optimization steps taking place in the primal domain.

Another definition of mirror descent from [9] which uses a projection with the Bregman divergence is

$$x_{k+1} = \arg\min_x \left\{ \nabla f(x_k)^T (x - x_k) + \frac{1}{\gamma_k} d_\psi(x \| x_k) \right\} \tag{3.44}$$

The minimizing x_{k+1} can be found by solving

$$0 = \nabla_{x_{k+1}} \left(\nabla f(x_k)^T (x_{k+1} - x_k) + \frac{1}{\gamma_k} d_\psi(x_{k+1} \| x_k) \right) \tag{3.45}$$

$$= \nabla f(x_k) + \frac{1}{\gamma_k} (\nabla \psi(x_{k+1}) - \nabla \psi(x_k)) \tag{3.46}$$

$$\nabla \psi(x_{k+1}) = \nabla \psi(x_k) - \gamma_k \nabla f(x_k) \tag{3.47}$$

which leads to the mirror descent method

$$y_{k+1} = y_k - \gamma_k \nabla f(\nabla \psi^*(x_k)) \tag{3.48}$$

3.2.4 ℓ_p Norms as Mirrors

Looking at the requirements to the potential function ψ used in mirror descent, we immediately see that all ℓ_p-norms satisfy them apart from the ℓ_1-norm. The ℓ_1-norm is convex but not strictly convex. It is also not differentiable at the origin.

Setting the potential equal to a squared ℓ_p-norm has some desirable properties. In order to use the potential we have to derive the gradient and the conjugate gradient of the norms. First we derive the conjugate

$$\psi = \frac{1}{2} \| \cdot \|_p^2 \tag{3.49}$$

$$\psi^*(y) = \sup_x \{ \langle y, x \rangle - \psi(x) \} \tag{3.50}$$

$$= \sup_x \left\{ y^T x - \frac{1}{2} \|x\|_p^2 \right\} \tag{3.51}$$

Hölders inequality gives

$$y^T x \leq \sum_{i=1}^{n} |y_i x_i| \leq \|y\|_p \|x\|_q, \; \frac{1}{p} + \frac{1}{q} = 1 \tag{3.52}$$

$$\sup_x \left\{ y^T x - \frac{1}{2} \|x\|_p^2 \right\} \leq \|y\|_q \|x\|_p - \frac{1}{2} \|x\|_p^2 \tag{3.53}$$

Taking the derivative of the right hand side and setting it equal to zero then solving for $\|x\|_p$ to find the maximizing argument.

$$\frac{\partial}{\partial \|x\|_p} \|y\|_q \|x\|_p - \frac{1}{2} \|x\|_p^2 = \|y\|_q - \|x\|_p = 0 \tag{3.54}$$

$$\tag{3.55}$$

which gives

$$\|y\|_q = \|x\|_p \tag{3.56}$$

Inserting back we find the maximum

$$\psi^*(y) = \sup_x \left\{ y^T x - \frac{1}{2}\|x\|_p^2 \right\} \leq \|y\|_q \|x\|_p - \frac{1}{2}\|x\|_p^2 \tag{3.57}$$

$$\leq \frac{1}{2}\|y\|_q^2 \tag{3.58}$$

To show the other inequality, let x be any vector with $y^T x = \|x\|_p \|y\|_q$, scaled so that $\|x\|_p = \|y\|_q$. Then we have, for this x,

$$y^T x - \frac{1}{2}\|x\|_p^2 = \|y\|_q \|x\|_p - \frac{1}{2}\|x\|_p^2 \tag{3.59}$$

$$= \frac{1}{2}\|y\|_q^2 \tag{3.60}$$

which shows that $\psi^*(y) \geq \frac{1}{2}\|y\|_q^2$ and therefore

$$\psi^* = \frac{1}{2}\|\cdot\|_q^2 \tag{3.61}$$

The gradient of the ℓ_p-norm is

$$\frac{\partial}{\partial x_j}\|x\| = \frac{\partial}{\partial x_j}\left(\sum_{i=0}^{n} |x_i|^p \right)^{\frac{1}{p}} \tag{3.62}$$

The chain rule gives

$$\frac{\partial}{\partial x}|x| = \frac{x}{|x|} \tag{3.63}$$

$$\frac{\partial}{\partial x_j}\left(\sum_{i=0}^{n} |x_i|^p \right)^{\frac{1}{p}} = \frac{1}{p}\left(\sum_{i=0}^{n} |x_i|^p \right)^{\frac{1}{p}-1} p|x_j|^{p-1}\frac{x_j}{|x_j|} \tag{3.64}$$

$$= \left(\left(\sum_{i=0}^{n} |x_i|^p \right)^{\frac{1}{p}-1} \right)^{1-p} |x_j|^{p-2} x_j \tag{3.65}$$

$$= \|x\|_p^{1-p} |x_j|^{p-2} x_j \tag{3.66}$$

$$= x_j \frac{|x_j|^{p-2}}{\|x\|_p^{p-1}} \tag{3.67}$$

In vector form this becomes

$$\nabla\|x\|_p = x \circ \frac{|x|^{p-2}}{\|x\|_p^{p-1}} \tag{3.68}$$

where $\circ$ is element-wise multiplication.

Using the chain rule we can easily find the gradient of the squared p-norm

$$\nabla\frac{1}{2}\|x\|_p^2 = \|x\|_p \nabla\|x\|_p \tag{3.69}$$

$$= x \circ \left(\frac{|x|}{\|x\|_p} \right)^{p-2} \tag{3.70}$$

3.2.5 Stochastic Gradient Methods

For larger optimization problems, computing the gradient for all data points simultaneously becomes computationally intractable. This is generally the case for problems in machine learning. In these cases, a stochastic version of gradient methods is used. The stochastic modification pulls a random sample from the data set and computes the gradient for the single data point and performs an optimization step. For real world datasets there will be considerable noise in individual data points which weakens the convergence properties for stochastic gradient methods.

Chapter 4

Regression

Now that we have covered optimization we move to the second technique neural networks are built on, regression. Regression analysis is concerned with identifying the underlying pattern of data. In this chapter we will introduce some simple regression problems and demonstrate how they can be solved by the optimization methods introduced in the previous chapter. The concept of regularization will be presented and a basic nonlinear neural network is used to show the effect of explicit and implicit regularization.

4.1 Linear Regression

A regression problem is an unconstrained optimization problem where the objective is to find a function $\hat{f}(x)$ that fits data points $(x_i, y_i),\ i = 1, ..., n$ from an unknown function $y_i = f(x_i)$ [10]. The simplest form of regression is linear regression where we assume the data points are the output of a linear function $y = Xa$ where X is the augmented data matrix consisting of stacked input vectors with an 1 added to the end for bias. The problem is then reduced to finding the parameter vectors $\hat{a}$ that transforms X to y, which is equivalent to minimizing the error $X\hat{a} - y$ through some norm $\|\cdot\|$.

$$\min_{\hat{a}} \|X\hat{a} - y\| \tag{4.1}$$

$$X = \left[\begin{array}{c|c} x_1^T & 1 \\ x_2^T & 1 \\ \vdots & \vdots \\ x_m^T & 1 \end{array}\right] \in \mathbb{R}^{m\times(n+1)} \tag{4.2}$$

$$x_i \in \mathbb{R}^n,\ \hat{a} \in \mathbb{R}^{n+1},\ y \in \mathbb{R}^m \tag{4.3}$$

A problem on this form is convex and always has at least one optimal solution. If $y \in \mathcal{R}(X)$ there exists a solution with zero error [5].

The most common approach to regression is the method of least squares which is defined as

$$\min_{\hat{a}} \|X\hat{a} - y\|_2^2 \tag{4.4}$$

This can be solved analytically for the optimal $\hat{a}^*$ assuming $X^T X$ is full rank. We define the objective function L, called the *loss function* in regression as

$$L(\hat{a}) = \|X\hat{a} - y\|_2^2 = a^T X^T X a - 2y^T X y + y^T y \tag{4.5}$$

We set the gradient with respect to $\hat{a}$ of the loss function to zero and solve

$$\nabla L(\hat{a}) = 0 \tag{4.6}$$

$$2X^T X\hat{a}^* - 2X^T y = 0 \tag{4.7}$$

$$\hat{a}^* = (X^T X)^{-1} X^T y \tag{4.8}$$

Because the problem is convex and unconstrained this can be efficiently solved with gradient algorithms

$$\hat{a}_{k+1} = \hat{a}_k - \gamma \nabla f(\hat{a}) \tag{4.9}$$

4.2 Regularization

For regression problems that result in underdetermined equations there will exist multiple sets of optimal parameters. We call these regression problems *overparameterized*. An extra objective can then be added to the minimization problem to penalize certain choices of parameters in order to "shrink" the problem and obtain a single optimal solution [5] [10]. This method is called *regularization*.

In Tikhinov regularization a squared ℓ_2-norm penalization of the parameters, weighted by $\lambda > 0$ is added to the objective function. The objective then becomes

$$\min_{\hat{a}} \|X\hat{a} - y\|_2^2 + \lambda \|\hat{a}\|_2^2 \tag{4.10}$$

which can be analytically solved the same way as the classic least squares problem

$$\nabla f(\hat{a}) = 2X^T X a - 2X^T y + 2\lambda \hat{a} = 0 \tag{4.11}$$

$$\implies (X^T X + \lambda I)\hat{a} = X^T y \tag{4.12}$$

$$\hat{a}^* = (X^T X + \lambda I)^{-1} X^T y \tag{4.13}$$

By adding $\lambda > 0$ to the diagonal of the squared data matrix $X^T X$ it is guaranteed to be invertible and the problem always has an analytical solution.

Because we penalize the objective with the size of the squared parameters the solution will favour parameter vectors that are small in the Euclidean norm, even over perfect solutions if λ is sufficiently large. In other words regularization trades lower variance for higher bias.

The least absolute shrinkage and selection operator (LASSO) technique adds a ℓ_1-norm penalization to the loss function weighted by the hyperparameter $\lambda > 0$

$$\min_{\hat{a}} \|X\hat{a} - y\|_2^2 + \lambda\|\hat{a}\|_1 \tag{4.14}$$

The ℓ_1-norm penalization of the parameters will lead to the sparsest solution being favored. That is the solution with the most parameters of value 0.

In an estimation setting regularization can be thought of as the mathematical equivalent of Occam's Razor which states that the simplest solution is most likely correct. The regularization term can also be thought of as representing the cost of parameters. In a robotics setting few parameters might be equivalent to few actuators and small parameters equal to less power used in actuators.

Because the ℓ_1-norm is not strictly convex a linear combination of the two penalty terms can be used in order to make the loss function strictly convex. This technique is called *elastic net* regularization.

$$\min_{a} \|X^T\hat{a} - y\|_2^2 + \lambda_1\|\hat{a}\|_1 + \lambda_2\|\hat{a}\|_2 \tag{4.15}$$

4.3 Nonlinear Regression

Now consider the regression problem where $f(x_i) = y_i$ is a nonlinear function. The linear basis used in linear regression is no longer sufficient. We replace the linear basis by a vector of nonlinear *basis functions* with x as input.

$$Y(x)^T\hat{a} \approx f(x) \tag{4.16}$$

The problem of minimizing the approximation error can then be formulated as

$$\min_{\hat{a}} \|Y(x)^T\hat{a} - y\| \tag{4.17}$$

The basis functions can either be engineered from domain knowledge or more generic functions can be used. Examples of popular nonlinear basis functions are the logistic function $\frac{1}{1+e^{-x}}$, $\tanh(x)$ and the rectified linear function which will be introduced soon.

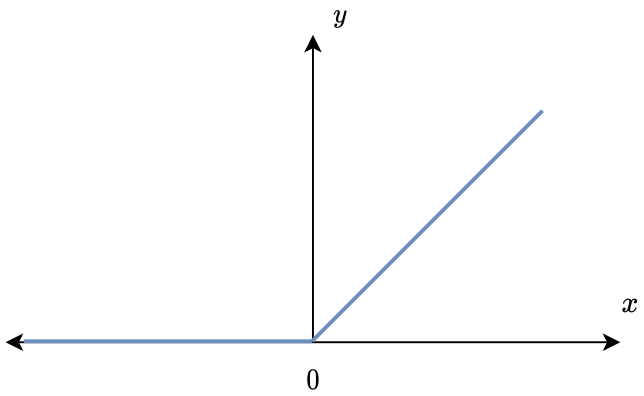


Figure 4.1: The Rectified Linear Unit (ReLU)

4.3.1 Approximating Functions With a Nonlinear Basis

By using a nonlinear basis of arbitrary size we should be able to fit any continuous function with arbitrary precision. We will now show how a nonlinear regressor with a ReLU basis can be optimized to fit a function using gradient descent. This regression technique is also called a feed forward neural network, or more specifically a single layer perceptron. The network is an adaption of the neural network introduced in [10] where ReLU is used as activation function instead of the logistic function.

$$\text{ReLU}(x) = \max(x, 0) \tag{4.18}$$

ReLU is nonlinear, well behaved, and is easy to differentiate. The derivative is undefined at the origin, but this can simply be set equal to 1 or 0. In order to obtain sparse solutions we define the derivative at $x = 0$ to 0.

$$\frac{\partial}{\partial x}\text{ReLU}(x) = \mathbb{I}(x) = \begin{cases} 1, & x > 0 \\ 0, & x \leq 0 \end{cases} \tag{4.19}$$

where $\mathbb{I}$ is called the indicator function.

We use two parameter vectors $\hat{a}$ and $\hat{b}$ called gain and bias, respectively. Giving us the regressor

$$\hat{f}(x) = Y(x\mathbb{1} + \hat{b})^T \hat{a} = \hat{y} \tag{4.20}$$

$$\tag{4.21}$$

$$Y(x\mathbb{1} + \hat{b}) = \begin{bmatrix} \text{ReLU}(x + \hat{b}_1) \\ \text{ReLU}(x + \hat{b}_2) \\ \vdots \\ \text{ReLU}(x + \hat{b}_d) \end{bmatrix} \tag{4.22}$$

$$x, y \in \mathbb{R},\ \hat{a}, \hat{b} \in \mathbb{R}^d \tag{4.23}$$

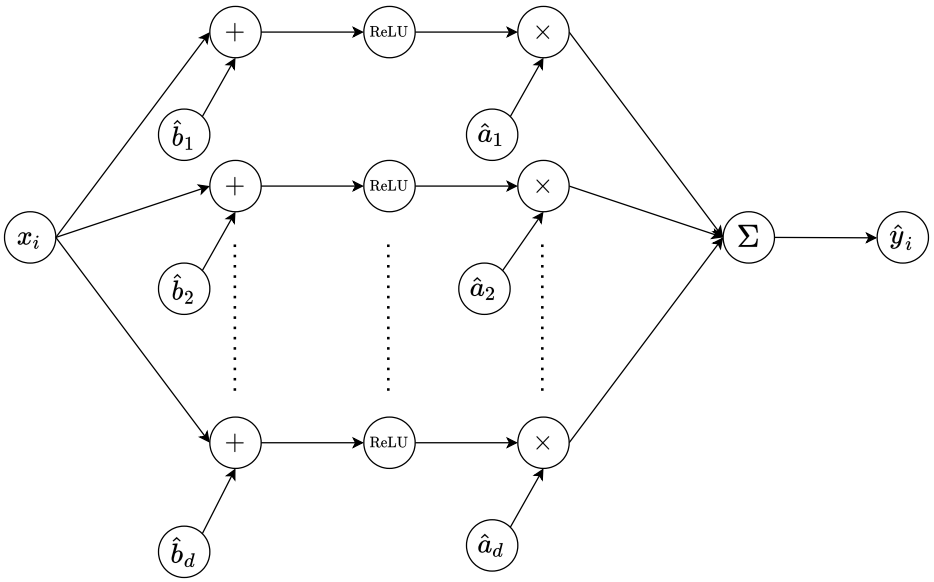


Figure 4.2: Computational graph of a single-layer perceptron with ReLU activation functions.

where $\mathbb{1}$ is a column vector of 1s of appropriate dimension.

In order to fit the regressor to the datapoint tuples (x_i, y_i), we use gradient descent with respect to the parameters on a squared estimation error loss function. Defining the optimization problem

$$L = \frac{1}{2}(\hat{y} - y)^2 \tag{4.24}$$

$$\min_{\hat{a},\hat{b}\in\mathbb{R}^d} L = \min_{\hat{a},\hat{b}\in\mathbb{R}^d} \sum_{i=1}^{n} \frac{1}{2}(Y(x_i\mathbb{1} + \hat{b})^T\hat{a} - y_i)^2 \tag{4.25}$$

Taking the partial derivative of the loss function with respect to the parameter vectors gives us the step directions for gradient descent

$$\nabla_{\hat{a}} L = \sum_{i=1}^{n} -(y_i - Y(x_i\mathbb{1} + \hat{b})^T\hat{a})Y(x_i\mathbb{1} + \hat{b}) \tag{4.26}$$

$$\nabla_{\hat{b}} L = \sum_{i=1}^{n} -(y_i - Y(x_i\mathbb{1} + \hat{b})^T\hat{a})\nabla_b Y(x_i\mathbb{1} + \hat{b})^T\hat{a} \tag{4.27}$$

$$\nabla_{\hat{b}} Y(x_i\mathbb{1} + \hat{b}) = \begin{bmatrix} \mathbb{I}(x + \hat{b}_1) & 0 & \dots & 0 \\ 0 & \mathbb{I}(x + \hat{b}_2) & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \dots & 0 & \mathbb{I}(x + \hat{b}_d) \end{bmatrix} \tag{4.28}$$

While the squared loss function is convex with respect to $\hat{y}_i$, it is generally not convex with respect to the parameters for nonlinear regressors. Yet experience shows that gradient based optimization methods are still effective at optimizing the parameters for overparameterized networks. This is something we will discuss in the next section.

4.4 The Loss Landscape of Overparameterized Regressors

Visualizing the high dimensional loss function of an overparameterized nonlinear regressor it would be fair to assume the function is rather ill-conditioned and has many local minima making gradient algorithms converge to suboptimal parameters if they even converge at all.

Recent research on the other hand paints a brighter picture. Hardt [11] demonstrates that overparameterized neural networks are able to fit random noise with zero loss using gradient algorithms. This shows that gradient algorithms are effective at optimizing regressors even for the hardest problems as long as the regressor is sufficiently overparameterized.

For linear networks theoretical proof [2] exists showing that for sufficiently overparametrized networks all minima are in fact global minima. There will exist multiple parameter vectors interpolating the dataset, resulting in zero loss. If we stack all the parameters into the vector a given the loss function L we can define this set as

$$\mathcal{A} = \{a \mid L(a) = 0\} \tag{4.29}$$

For regressors with a basis consisting of one function, like the ReLU regressor in the previous section, this is especially true. Because permuting the parameter tuples (a_i, b_i) would result in an identical network.

Cooper [12] argues that the set $\mathcal{A}$ even forms a continuous smooth manifold in parameter space. For an regressor with $d \gg n$ the manifold is either empty or a submanifold of dimension $d - n$.

4.5 Implicit Regularization in Overparametrized Regressors

Even in the case where there exists an infinite amount of parameters yielding zero loss, gradient algorithms will converge to a single solution. Which solutions will they converge to and why? Is there any way to alter the optimization algorithm in order to make the algorithm choose parameters with certain properties?

The first question is investigated by Gunesekar [2]. He demonstrates that minimizing the loss with gradient descent does not take us to just any global

minimum, but the global minimum minimizing some regularizer defined implicitly by the optimization algorithm. We call this phenomenon *implicit regularization*. Using gradient descent on a linear regressor provably makes the parameters converge to the minimum ℓ_2-norm solution

$$\hat{a}^*_{\text{GD}} = \arg\min_{a \in \mathcal{A}} \|a\|_2 \tag{4.30}$$

This is an important result because it shows that the choice of optimization algorithm is just as important as the choice of basis and parameterization in a regressor.

As we already know gradient descent is equivalent to mirror descent with $\psi = \frac{1}{2}\|\cdot\|_2^2$. Azizan [3] extends Gunesekars result and shows that by using mirror descent to optimize an overparameterized regressor the method implicitly regularizes the choice of parameters by the potential function ψ. Azizan also demonstrates through experiments that mirror descent imposes approximate regularization on nonlinear regressors. If the initial conditions of mirror descent are close enough to the manifold of global minima the algorithm will converge to the solution approximately closest to the initial condition measured by the Bregman divergence

$$\hat{a}^*_{\text{MD}} = \arg\min_{a \in \mathcal{A}} d_\psi(a \| a_0) \tag{4.31}$$

For highly overparameterized regressors "all" initial conditions are close to the manifold $\mathcal{A}$. This means that the choice of potential and initial conditions can be used as design parameters.

By setting the initial conditions to zero, which by definition makes the potential zero $\psi(0) = 0$ we get:

$$\hat{a}^*_{MD} = \arg\min_{a \in \mathcal{A}} \psi(a) \tag{4.32}$$

which gives us the interpolating parameters minimizing the potential.

4.6 Some Results in Regularized Regression

In this section we illustrate how different optimization techniques affect the nonlinear regression problem introduced in Section 4.3.1 through some experiments.

For the experiment we will approximate the nonlinear function $f(x) = x + \sin(x)$ over the domain $x \in [0, 10]$ with the nonlinear regressor introduced in Section 4.3.1. We use a basis consisting of ReLU functions with individual gain and bias parameters. We draw two independent sets of 10 samples of the function from a uniform distribution over the domain. One set for optimizing the regressor, and one for testing the fit of the function approximation. These

sets are called training set and test set respectively. The discrepancy between training and test loss is called generalization error and tells us how well the regressor is able to approximate the underlying pattern of the data.

All experiments use the same training and test set, and the same initial conditions. The initial bias parameters β are drawn from a uniform distribution over the domain of x. The initial conditions for the gain α are drawn from a uniform distribution over the domain $\alpha_0 \in [-0.1, 0.1]$. A small initial gain makes the model more "linear" and thus well-behaved at the start of the optimization period resulting in a less jagged end result.

The regressor has dimension $d = 200$ making it 20 times overparameterized as 10 ReLU functions is sufficient for perfect interpolation of the training data. Each ReLU function has independent gain and bias resulting in a total of 400 parameters.

The optimization is run for $1.2 \cdot 10^5$ steps with a step length of $2 \cdot 10^{-4}$ for both parameter vectors.

The nonlinear regressor is implemented in Python using the matrix library `numpy` and the plotting library `matplotlib`.

4.6.1 Gradient Descent

For the first experiment optimize the regressor with standard gradient descent.

4.6.2 Regularized Gradient Descent

For the next two experiments we regularize the regressor with the ℓ_1 and ℓ_2-norms. The weighting of the regularization term can be found in Table 4.1.

	λ_α	λ_β
ℓ_1	0.005	0.020
ℓ_2	0.03	0.1

Table 4.1: Regularization weight parameters.

4.6.3 Mirror Descent for Sparse Estimation

In the last function fitting experiment we inspect the effect of using mirror descent with $\psi = \frac{1}{2}\|\cdot\|^2_{1.01}$ as mirror. From the last section we expect this to perform similar to the explicitly ℓ_1-norm regularized regressor.

4.6.4 Results

The experiment illustrates how different optimization algorithms converge to different global minima. Comparing the parameter histograms of the different

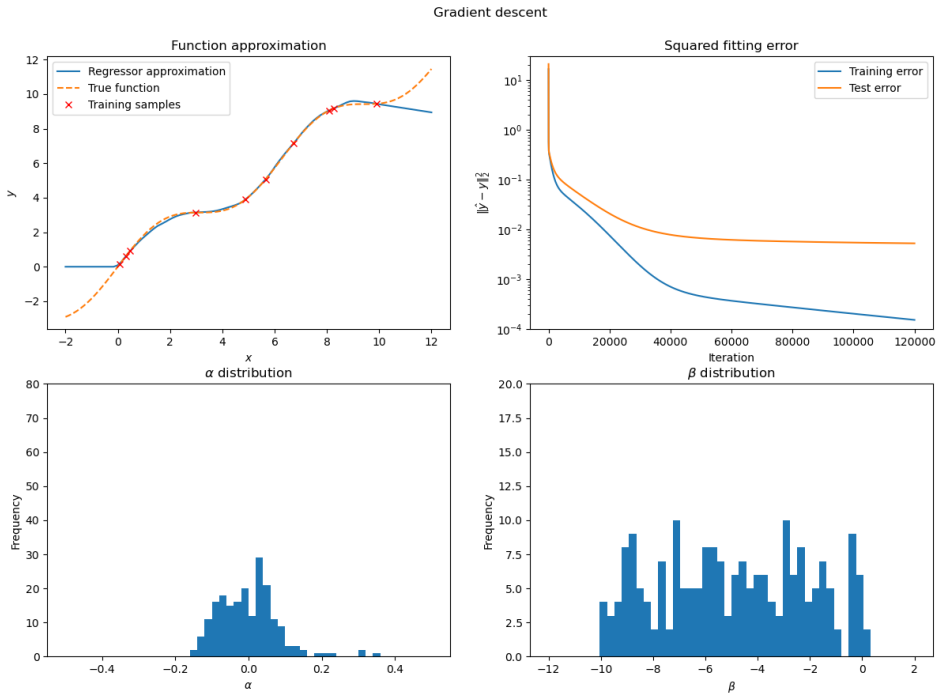


Figure 4.3: Results of optimizing the function fitting regressor with gradient descent.

methods it is clear that they have converged to different minima. When comparing the test error of the different networks in Figure 4.7 it is apparent that the different minima score the same when tested on unseen data. This indicates that the different minima are in fact all global minima.

Inspecting the different parameter histograms the implicit regularization of gradient descent and ℓ_1 mirror descent becomes apparent. The α parameter distribution for gradient descent seem to be approximately normally distributed while the mirror descent histogram shows a sparse solution where one third of the Relu functions are deactivated by setting the gain to zero.

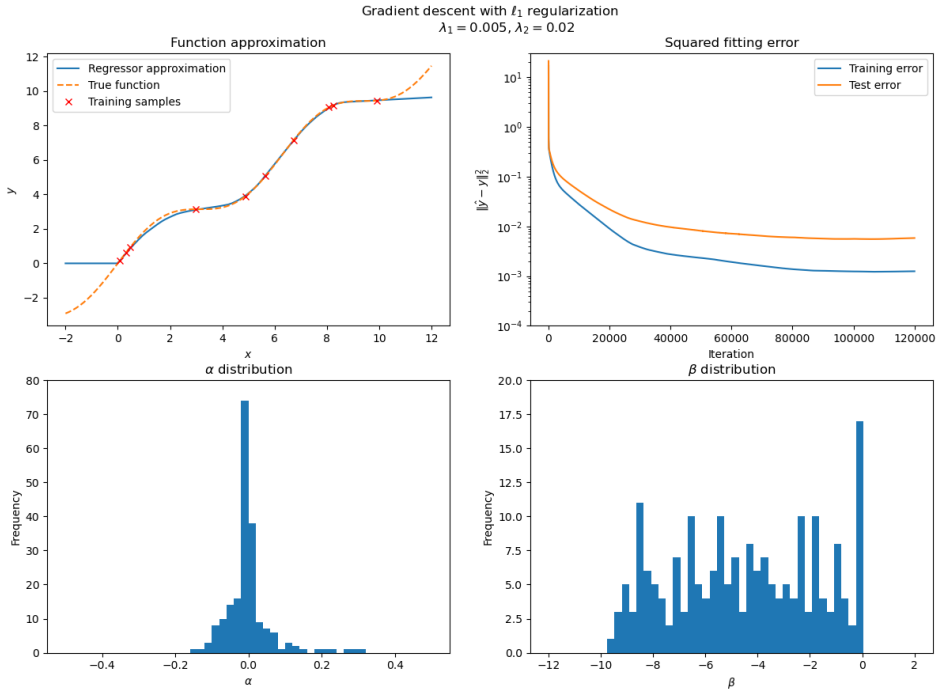


Figure 4.4: Results of optimizing the function fitting regressor with gradient descent regularized by a ℓ_1-norm penality.

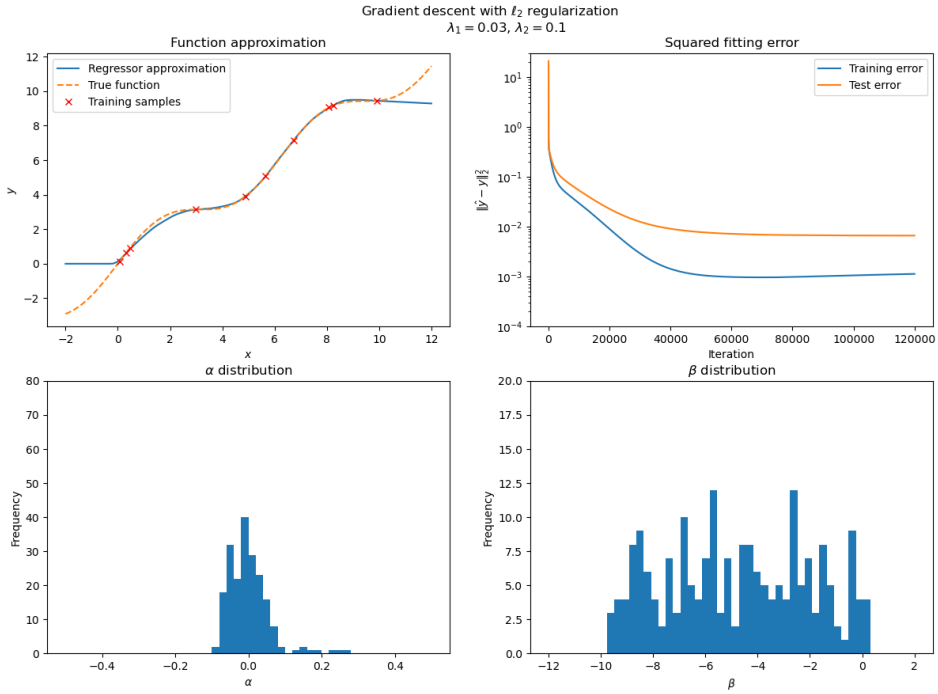


Figure 4.5: Results of optimizing the function fitting regressor with gradient descent regularized by a ℓ_2-norm penality.

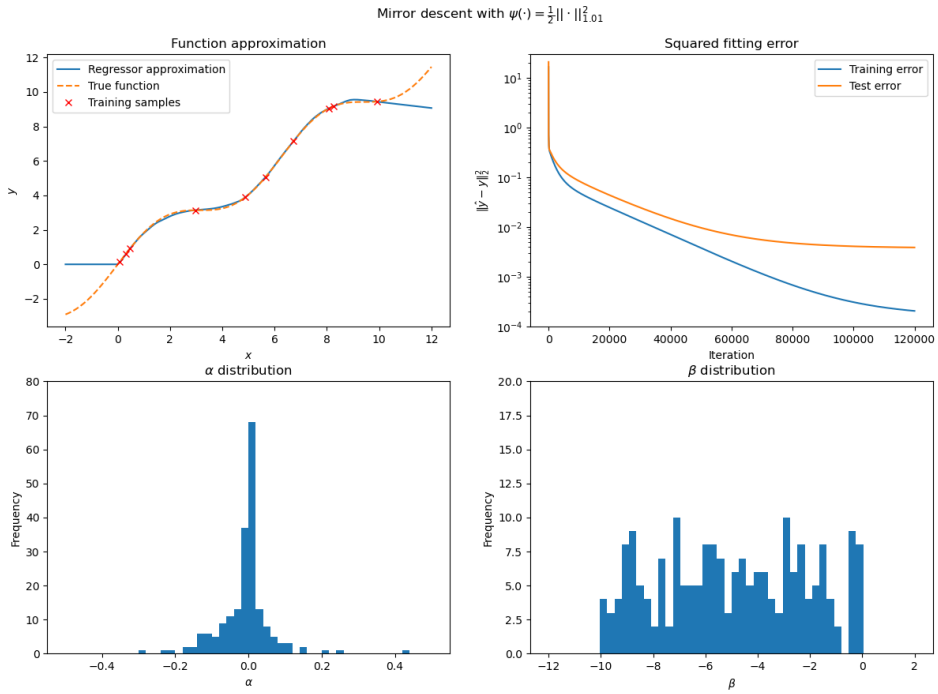


Figure 4.6: Results of optimizing the function fitting regressor with mirror descent with an approximated squared ℓ_1 norm mirror.

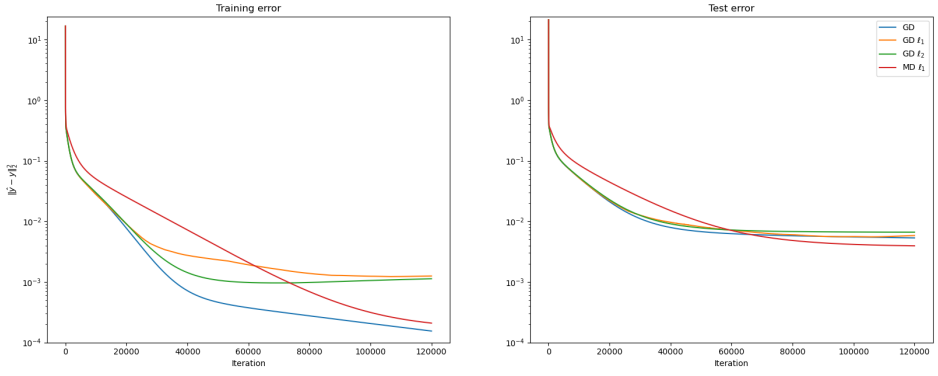


Figure 4.7: Fitting error of the different optimization algorithms for the nonlinear regression problem plotted on a logarithmic scale.

Chapter 5

Adaptive Control

The objective of adaptive control is stable simultaneous learning and control of a dynamical system. In this chapter we will introduce some methods for learning parametric models that are similar to the regression problems in the previous chapter. Adaption laws similar to the optimization methods already covered will be presented. Lastly we show that the implicit regularization in regression extends to adaptive nonlinear control.

The key feature of an adaptive control algorithm is the *adaption law*. Adaption is an online learning problem concerned with learning the dynamics of a system. In practice this is done with a parametric model where the parameters are learned by stochastic gradient inspired methods. There are considerable similarities between adaptive control and machine learning. One important difference is that while machine learning is mainly concerned with the end result after optimizing the network, adaptive control needs to guarantee that the learned parameters result in a stable system at all times.

5.1 Linear Parametric Models

An assumption usually made in adaptive control is that the unknown nonlinear dynamics depends linearly on a set of unknown parameters a

$$\dot{x} = f(x, a, t) = Y(x)a \tag{5.1}$$

$$Y : \mathbb{R}^n \to \mathbb{R}^{n \times d},\ \hat{a} \in \mathbb{R}^d \tag{5.2}$$

By modeling the system as a linear combination of basis functions of the state $Y(x)$ and an estimated parameter vector $\hat{a}$ we can approximate the dynamics of the system

$$\dot{\hat{x}} = \hat{f}(x, \hat{a}, t) = Y(x)\hat{a} \tag{5.3}$$

$$\tag{5.4}$$

The equation is linear in the parameters $\hat{a}$ which is crucial for how we design the adaption law. The estimated dynamics are on the same form as the nonlinear regressor already introduced illustrating the similarities between machine learning and adaptive control.

5.2 Sliding Control

Before we introduce the gradient adaption laws we need to define the sliding variable s [13]

$$s = \left(\frac{d}{dt} + \lambda\right)^{n-1} \tilde{x} \tag{5.5}$$

where n is the order of the system, λ a strictly positive constant, and $\tilde{x} = x - \hat{x}$ the prediction error.

For a second order system consisting of position and velocity this would give us

$$s = \left(\frac{d}{dt} + \lambda\right) \tilde{x} \tag{5.6}$$

$$= \dot{\tilde{x}} + \lambda\tilde{x} \tag{5.7}$$

which is a weighted sum of position and velocity error. Now the problem of perfect estimation $\hat{x} = x$ is equivalent to remaining on the surface defined by $s = 0$. s defines a linear differential equation with equilibrium point $\hat{x}^* = 0$. s therefore represents the performance of the estimator. The parameterization replaces a nth-order tracking problem by a 1st-order stabilization problem.

The tracking problem can now be solved by choosing an adaption law such that

$$\frac{1}{2}\frac{d}{dt}s^2 \leq -\gamma|s| \tag{5.8}$$

Causing the system to converge to and slide along the surface $s = 0$.

5.3 Gradient Methods for Linear Parametric Models

In order to control the system we need to estimate parameters that minimizes the prediction error. The prediction error variable for linear parametric models is defined as

$$\dot{\tilde{x}} = \dot{\hat{x}} - \dot{x} \tag{5.9}$$

$$= Y(x)\hat{a} - Y(x)a \tag{5.10}$$

$$= Y(x)\tilde{a} \tag{5.11}$$

A simple and stable technique for minimizing the prediction error is using a gradient descent like algorithm [13]

$$\dot{\hat{a}} = -\gamma Y(x)^T s \tag{5.12}$$

where $\gamma > 0$ is the scalar adaption gain. This adaption law is referred to as the Slotine-Li adaption law.

If the parameter estimation is on-line this is the continuous time equivalent of the stochastic gradient descent algorithm introduced earlier.

5.4 Mirror Descent Based Adaption Law

As demonstrated in the chapter on optimization the gradient descent method is a special case of mirror descent. We now present a mirror descent based adaption law introduced by Boffi and Slotine [4] from which the Slotine-Li adaption law is a special case. The law can be derived by replacing the adaption error term $\frac{1}{2}\tilde{a}^T P^{-1}\tilde{a},\ P \geq 0$ in the Lyapunov like function by the Bregman divergence for a strictly convex potential ψ

$$V = \frac{1}{2}s^T s + \frac{1}{2}d_\psi(a\|\hat{a}) \tag{5.13}$$

The derivative of the Bregman divergence is

$$\frac{d}{dt}d_\psi(a\|\hat{a}) = \tilde{a}^T\nabla^2\psi(\hat{a})\dot{\hat{a}} \tag{5.14}$$

which can be used to prove the stability of the adaption law

$$\dot{\hat{a}} = -\gamma\left(\nabla^2\psi(\hat{a})\right)^{-1} Y(x)^T s \tag{5.15}$$

The inverted Hessian is cumbersome to compute and we will therefore perform the adaption in the mirrored domain by using the identity $\nabla^2\psi(\hat{a})\dot{\hat{a}} = \frac{d}{dt}\nabla\psi(\hat{a})$

$$\frac{d}{dt}\nabla\psi(\hat{a}) = -\gamma Y(x)^T s \tag{5.16}$$

where $\gamma > 0$ is the adaption gain. The parameters can then be recovered by using the conjugate gradient of the potential $\nabla\psi^*(\nabla\psi(\hat{a})) = \hat{a}$.

This is the continuous time equivalent of stochastic mirror descent. Using this adaption law gives us the ability to introduce implicit regularization into the adaption law by choice of the potential ψ and initial condition $\hat{a}_0$ while still guaranteeing stability, something that cannot easily be done with explicit regularization.

As we recall mirror descent converges to the solution closest to the initial conditions measured in Bregman divergence for linear regressors. Azizan showed that this can be approximately achieved for nonlinear regressors as well.

5.5 Persistent Excitation

Tho goal of the adaptive controller is to minimize the error and this does not necessarily require the correct parameter estimate. In order for the estimated parameter to converge to the true parameters the system must satisfy a condition called *persistent excitation* [13].

To obtain the correct parameters we need to solve

$$\|Y\tilde{a}\|_2^2 = 0 \tag{5.17}$$

$$\tilde{a}^T Y^T Y \tilde{a} = 0 \tag{5.18}$$

$Y^T Y$ is positive semidefinite but not positive definite as Y will never be full rank at any time instant. But if the trajectory is complicated enough the time average of Y will have full rank. Then the estimated parameters will converge to the true parameters $\hat{a} \to a$ if the following condition is met

$$\frac{1}{\tau}\int_t^{t+\tau} Y^T Y \geq \alpha I \tag{5.19}$$

$$\forall t \geq t_o,\ \exists t_0 \geq 0,\ \exists \tau > 0,\ \exists \alpha > 0 \tag{5.20}$$

which means Y will be positive definite on average over the time interval $[t, t+\tau]$. If the signal is persistently exciting, the parameter error will converge to 0 exponentially.

5.6 Implicit Regularization in Adaptive Control

The parameters need only fit the unknown dynamics along the system trajectory in order to achieve zero error. This means that for trajectories that are not persistently exciting there exists possible infinite choices of parameters that give zero error, even for underparameterized systems. For overparameterized regressors the set of parameters resulting in zero error is not unique regardless of the trajectory.

Again we define the set of parameters $\mathcal{A}$ that perfectly interpolate the dynamics along the trajectory at all time time instants t

$$\mathcal{A} = \{\hat{a} \mid Y(x)\hat{a} = f(x),\ \forall t\} \tag{5.21}$$

Consider the regressor matrix $Y(x_d(t))$ along the desired trajectory $x_d(t)$ with null space $\mathcal{N}(Y(x_d(t))$ and the true parameter vector a. Then for all vectors $\delta a(t) \in \mathcal{N}(Y(x_d(t))$ we have $(a + \delta a(t)) \in \mathcal{A}$. For the case were there exist an infinite amount of interpolating parameters. Which parameters will the adaption law converge to?

Building on the work of Azizan [3], Boffi and Slotine [4] developed a continuous time proof for implicit regularization in linearly parameterized systems with the mirror descent based adaption law.

Let θ be a constant vector of parameters. The time derivative of the Bregman divergence is

$$\frac{d}{dt}d_\psi(\theta\|\hat{a}) = -\left(\frac{d}{dt}\nabla\psi(\hat{a})\right)^T(\theta-\hat{a}) \tag{5.22}$$

inserting the mirror descent adaption law we get

$$\frac{d}{dt}d_\psi(\theta\|\hat{a}) = s^TY(x)(\theta-\hat{a}) \tag{5.23}$$

integrating both sides gives us

$$d_\psi(\theta\|\hat{a}_0) = d_\psi(\theta\|\hat{a}) + \int_0^t s^TY(x(\tau))(\hat{a}(\tau)-\theta)d\tau \tag{5.24}$$

If $\theta \in \mathcal{A}$ then $Y(x)\theta = f(x)$. The integral is then independent of θ. Assuming that $\hat{a}$ converges to some optimal parameter $\hat{a}^* \in \mathcal{A}$ we take the limit $t \to \infty$

$$d_\psi(\theta\|\hat{a}_0) = d_\psi(\theta\|\hat{a}^*) + \int_0^t s^TY(x(\tau))\hat{a}(\tau) - f(x(\tau))d\tau \tag{5.25}$$

The only dependence on θ on the right hand side is in the Bregman divergence term. Because this equation holds for any constant θ the minimizing argument of the two Bregman divergences must be identical. The minimizing argument for the right hand side is simply $\theta = \hat{a}^*$. We can then conclude that

$$\hat{a}^* = \arg\min_{\theta\in\mathcal{A}} \; d_\psi(\theta\|\hat{a}^*) \tag{5.26}$$

$$= \arg\min_{\theta\in\mathcal{A}} d_\psi(\theta\|\hat{a}_0) \tag{5.27}$$

This tells us that the adaption law converges to the optimal $\hat{a}^*$ closest to the initial condition $\hat{a_0}$ measured in Bregman divergence. Like in the case with mirror descent we can set the initial conditions to zero $\hat{a}_0 = 0$ then by definition the initial potential is zero $\psi(0) = 0$ which result in

$$\hat{a}^* = \arg\min_{\theta\in\mathcal{A}} \; \psi(\theta) \tag{5.28}$$

which is equal to regularization by the potential ψ.

Chapter 6

Hamiltonian Systems

This chapter introduces the theory of Hamiltonian systems from a dynamical system perspective. The classic nonlinear three-body system is presented and different regressors for learning its parameters are found. Dynamic prediction of the three-body problem will later be used to empirically demonstrate the implicit regularization in the adaption laws from the previous chapter.

6.1 Introduction

Consider a mechanical system with n degrees of freedom. Newton's second law can be used to derive the equations of motion as a system of second-order differential equations in $\mathbb{R}^n$ which can be transformed to a first-order system in $\mathbb{R}^{2n}$. If the forces originate from a potential function, like Newtonian gravitational forces, or a spring, the system can be described by a single scalar function of the system state called the Hamiltonian [14]. The equations of motion for a Hamiltonian system is described as $2n$ first-order differential equations in the canonical states q, p. Here q represents the position of the mass and p represents the momentum $p = m\dot{q}$. The Hamiltonian is the sum of potential and kinetic energy in the system.

The time evolution of the states are defined by Hamilton's equations of motion

$$\frac{d}{dt}q = \frac{\partial \mathcal{H}}{\partial p} \tag{6.1}$$

$$\frac{d}{dt}p = -\frac{\partial \mathcal{H}}{\partial q} \tag{6.2}$$

The structure of the time evolution is called *symplectic*. The system can be represented by the symplectic coordinates r. The Hamiltonian dynamics are

then written as

$$\frac{d}{dt}r = J\nabla_r \mathcal{H}(r) \tag{6.3}$$

$$r = \begin{bmatrix} q \\ p \end{bmatrix} \in \mathbb{R}^{2n} \tag{6.4}$$

$$J = \begin{bmatrix} 0 & I_n \\ -I_n & 0 \end{bmatrix} \tag{6.5}$$

We call p and q conjugate variables. This structure is an important property of Hamiltonian systems and will be used throughout this thesis.

For autonomous Hamiltonian systems another interesting property arises when we inspect the time evolution of the Hamiltonian function

$$\dot{\mathcal{H}}(r) = \frac{\partial \mathcal{H}}{\partial t} + (\nabla_r \mathcal{H})^T \dot{r} \tag{6.6}$$

$$\dot{r} = J\nabla_r \mathcal{H} \tag{6.7}$$

$$\dot{\mathcal{H}}(r) = (\nabla_r \mathcal{H})^T J (\nabla_r \mathcal{H}) \tag{6.8}$$

$$= [\nabla_q \mathcal{H}, \nabla_p \mathcal{H}]\, J \begin{bmatrix} \nabla_q \mathcal{H} \\ \nabla_p \mathcal{H} \end{bmatrix} \tag{6.9}$$

$$= (\nabla_q \mathcal{H})^T \nabla_p \mathcal{H} - (\nabla_p \mathcal{H})^T \nabla_q \mathcal{H} \tag{6.10}$$

$$= 0 \tag{6.11}$$

The system energy does not vary with time, in other words the energy of autonomous Hamiltonian systems are conserved. This means the Hamiltonian is constant and only depends on initial conditions.

6.1.1 Linear Parameterization of Hamiltonian Dynamics

Suppose that the Hamiltonian of a system can be written as a linear parameterization

$$\mathcal{H} = Y_{\mathcal{H}}(r)^T a \tag{6.12}$$

for the column vector $Y_{\mathcal{H}}$. Then the dynamics are given by

$$\dot{r} = J\nabla_r \mathcal{H}(r) \tag{6.13}$$

$$= J\nabla_r Y_{\mathcal{H}}(r)^T a \tag{6.14}$$

Let the regressor matrix be defined by

$$Y(r) = J\nabla_r Y_{\mathcal{H}}(r)^T \tag{6.15}$$

This gives the dynamics

$$\dot{r} = Y(r)a \tag{6.16}$$

6.1.2 Dynamic Prediction of Hamiltonian Systems

Inspired by one of the experiments by Boffi and Slotine in [4] we will use dynamic prediction of a Hamiltonian system as the setting for our experiments.

If the Hamiltonian of a system is unknown it can be approximated by a linear combination of nonlinear basis functions. Consider the approximation

$$\hat{\mathcal{H}}(r, \hat{a}) = Y_{\mathcal{H}}(r)^T \hat{a} \tag{6.17}$$

By using the symplectic structure and the linear parameterization we can now estimate the dynamics of the symplectic coordinates with the prediction law

$$\dot{\hat{r}} = Y(r)\hat{a} - \eta\tilde{r} \tag{6.18}$$

where $\eta > 0$ is the feedback gain. Assuming the true system is of the form

$$\dot{r} = Y(r)a \tag{6.19}$$

We define the error dynamics

$$\dot{\tilde{r}} = Y(r)\tilde{a} - \eta\tilde{r} \tag{6.20}$$

Now gradient based adaption laws can be used to estimate the parameter vector a.

$$\dot{\hat{a}} = -\gamma Y(r)^T \tilde{r} \tag{6.21}$$

6.2 The Harmonic Oscillator

A simple Hamiltonian system is the harmonic oscillator. An example of an harmonic oscillator is a frictionless spring-mass system where q is the displacement from the springs equilibrium position, m the mass at the end of the spring and k the spring constant. The Hamiltonian of the system is defined as

$$\mathcal{H}(q, p) = \frac{1}{2m}p^2 + \frac{k}{2}q^2 \tag{6.22}$$

This gives the linear dynamics

$$\dot{q} = \frac{\partial \mathcal{H}}{\partial p} = \frac{1}{m}p \tag{6.23}$$

$$\dot{p} = -\frac{\partial \mathcal{H}}{\partial q} = -kq \tag{6.24}$$

$$\implies \dot{r} = \begin{bmatrix} 0 & \frac{1}{m} \\ -k & 0 \end{bmatrix} r \tag{6.25}$$

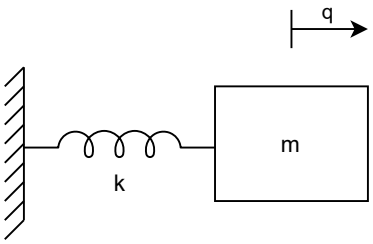


Figure 6.1: A simple harmonic oscillator with a spring and a mass without damping or external forces.

The simple dynamics can be solved analytically with respect to the displacement q, which gives

$$q(t) = q_0 \cos\left(\sqrt{\frac{k}{m}}\, t\right) \tag{6.26}$$

The Hamiltonian of the harmonic oscillator is a linear combination of terms depending on displacement q and momentum p this makes the Hamiltonian *separable*. We will later refer to the function $U(q)$ as the Hamiltonian potential and $T(p)$ the Hamiltonian momentum.

$$\mathcal{H} = T(p) + U(q) \tag{6.27}$$

$$U(q) = \frac{k}{2}q^2 \tag{6.28}$$

$$T(p) = \frac{1}{2m}p^2 \tag{6.29}$$

6.3 The Three-body Problem

A well known Hamiltonian system with nonlinear dynamics is the three-body system. The system consists of three point masses interacting trough Newtonian gravitational forces. The system is a classic mechanics problem studied for centuries and has, unlike the harmonic oscillator, no general solution making numerical simulations necessary to study the time evolution [14].

The system is described by the Hamiltonian

$$\mathcal{H} = -\frac{gm_1m_2}{\|q_1 - q_2\|_2} - \frac{gm_2m_3}{\|q_2 - q_3\|_2} - \frac{gm_3m_1}{\|q_3 - q_1\|_2} + \frac{p_1^2}{2m_1} + \frac{p_2^2}{2m_2} + \frac{p_3^2}{2m_3} \tag{6.30}$$

where m_i is the mass of particle i and g the gravitational constant. We note that the Hamiltonian is separable and linear in the mass and gravitational parameters.

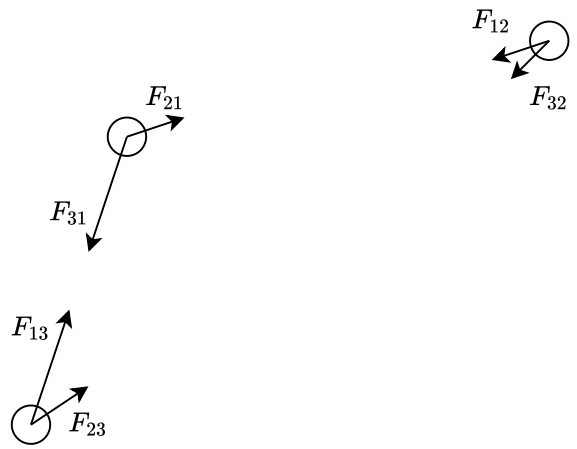


Figure 6.2: Three particles in 2D space affected by Newtonian gravitational forces.

By solving the Hamilton equations for the canonical coordinates we get

$$\dot{q} = \frac{\partial \mathcal{H}}{\partial p} = \begin{bmatrix} \frac{1}{m_1} p_1 \\ \frac{1}{m_2} p_2 \\ \frac{1}{m_3} p_3 \end{bmatrix} \tag{6.31}$$

$$\dot{p} = -\frac{\partial \mathcal{H}}{\partial} = \begin{bmatrix} -\frac{q_1 - q_2}{\|q_1 - q_2\|_2^3} - \frac{q_1 - q_3}{\|q_1 - q_3\|_2^3} \\ -\frac{q_2 - q_1}{\|q_2 - q_1\|_2^3} - \frac{q_2 - q_3}{\|q_2 - q_3\|_2^3} \\ -\frac{q_3 - q_1}{\|q_3 - q_1\|_2^3} - \frac{q_3 - q_2}{\|q_3 - q_2\|_2^3} \end{bmatrix} \tag{6.32}$$

Simulations show that the system is unstable for most initial conditions. Still there exists a plethora of periodic colissionless orbits in the plane where the system is well behaved [15] [16] [17]. These periodic solutions are particularly useful to investigate the stability and precision of numerical methods because errors compounding over time will make the simulation unstable and make the simulated system diverge quickly from the true solution. We will therefore use the three body system to test the different adaption laws.

6.3.1 A Model Based Regressor for the Three-Body Problem

Given the system Hamiltonian a model based regressor can be derived to estimate the system parameters. We formulate the Hamiltonian as a linear com-

bination of basis functions and parameters

$$Y_{\mathcal{H}}(r) = \begin{bmatrix} p_1^2 \\ p_2^2 \\ p_3^2 \\ -\frac{1}{\|q_1-q_2\|_2} \\ -\frac{1}{\|q_2-q_3\|_2} \\ -\frac{1}{\|q_1-q_3\|_2} \end{bmatrix}, \; a = \begin{bmatrix} \frac{1}{2m_1} \\ \frac{1}{2m_2} \\ \frac{1}{2m_3} \\ gm_1m_2 \\ gm_2m_3 \\ gm_3m_1 \end{bmatrix} \tag{6.33}$$

$$Y(r) = \begin{bmatrix} 2p_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2p_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2p_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\frac{q_1-q_2}{\|q_1-q_2\|_2^3} & 0 & -\frac{q_1-q_3}{\|q_1-q_3\|_2^3} \\ 0 & 0 & 0 & \frac{q_1-q_2}{\|q_1-q_2\|_2^3} & -\frac{q_2-q_3}{\|q_2-q_3\|_2^3} & 0 \\ 0 & 0 & 0 & 0 & \frac{q_2-q_3}{\|q_2-q_3\|_2^3} & \frac{q_1-q_3}{\|q_1-q_3\|_2^3} \end{bmatrix} \tag{6.34}$$

6.3.2 An Overparameterized Regressor for the Three-Body Problem

In a recent article Boffi and Slotine [4] describes a basis for estimating the three-body system consisting of physically motivated functions often appearing in physical systems.

The regression vector $Y_{\mathcal{H}}$ now consists of quadratic and quartic functions of the state of each particle in addition to the gravitational potential between the particles in powers of 1, 2, and 3. The regression basis then consists of 21 functions representing kinetic energy, spring potential, gravitational potential and higher order terms.

This basis has more expressive power than needed to describe the three-body system along periodic solutions, which only explore a small subset of the state space. The overparameterization should be enough to examine the implicit regularization in mirror descent based adaption laws.

$$Y_{\mathcal{H}}(r) = \begin{bmatrix} q_1^2 \\ q_2^2 \\ q_3^2 \\ q_1^4 \\ q_2^4 \\ q_3^4 \\ p_1^2 \\ p_2^2 \\ p_3^2 \\ p_1^4 \\ p_2^4 \\ p_3^4 \\ -\frac{1}{\|q_1-q_2\|_2} \\ -\frac{1}{\|q_1-q_3\|_2} \\ -\frac{1}{\|q_2-q_3\|_2} \\ -\frac{1}{\|q_1-q_2\|_2^2} \\ -\frac{1}{\|q_1-q_2\|_2^2} \\ -\frac{1}{\|q_1-q_3\|_2^2} \\ -\frac{1}{\|q_2-q_3\|_2^3} \\ -\frac{1}{\|q_1-q_3\|_2^3} \\ -\frac{1}{\|q_2-q_3\|_2^3} \end{bmatrix}, \; a = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \frac{1}{2m_1} \\ \frac{1}{2m_2} \\ \frac{1}{2m_3} \\ 0 \\ 0 \\ 0 \\ gm_1m_2 \\ gm_2m_3 \\ gm_3m_1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{6.35}$$

$$
Y(r) = \left[\begin{array}{cccccccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 2p_1 & 0 & 0 & 4p_1^3 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 2p_2 & 0 & 0 & 4p_2^3 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2p_3 & 0 & 0 & 4p_3^3 \\
-2q_1 & 0 & 0 & 0 & -4q_1^3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -2q_2 & 0 & 0 & 0 & -4q_2^3 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -2q_3 & 0 & 0 & 0 & -4q_3^3 & 0 & 0 & 0 & 0 & 0
\end{array}\right.
$$

$$
\left.\begin{array}{cccccccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-\frac{q_1-q_2}{\|q_1-q_2\|_2^3} & 0 & 0 & -\frac{q_1-q_3}{\|q_1-q_3\|_2^3} & -2\frac{q_1-q_2}{\|q_1-q_2\|_2^4} & 0 & -2\frac{q_1-q_3}{\|q_1-q_3\|_2^4} & 0 & -3\frac{q_1-q_2}{\|q_1-q_2\|_2^5} & 0 & 0 & -3\frac{q_1-q_3}{\|q_1-q_3\|_2^5} \\
\frac{q_1-q_2}{\|q_1-q_2\|_2^3} & -\frac{q_2-q_3}{\|q_2-q_3\|_2^3} & 0 & 2\frac{q_1-q_2}{\|q_1-q_2\|_2^4} & -2\frac{q_2-q_3}{\|q_2-q_3\|_2^4} & 0 & 3\frac{q_1-q_2}{\|q_1-q_2\|_2^5} & -3\frac{q_2-q_3}{\|q_2-q_3\|_2^5} & 0 & 0 & 0 & 0 \\
0 & \frac{q_2-q_3}{\|q_2-q_3\|_2^3} & \frac{q_1-q_3}{\|q_1-q_3\|_2^3} & 0 & 2\frac{q_2-q_3}{\|q_2-q_4\|_2^4} & 2\frac{q_1-q_3}{\|q_1-q_3\|_2^4} & 0 & 3\frac{q_2-q_3}{\|q_2-q_3\|_2^5} & 3\frac{q_1-q_3}{\|q_1-q_3\|_2^5} & 0 & 0 & 0
\end{array}\right] \tag{6.36}
$$

Chapter 7

Experiments

Now that the theoretical material has been covered we can move on to the experiments. Inspired by Boffi and Slotine [4] we will numerically simulate how different adaption laws and parameterizations perform on the three-body system in a dynamic prediction setting. The purpose of the simulations is to empirically verify the implicit regularization in parameters learned by gradient methods for nonlinear dynamical systems.

7.1 Periodic Trajectories

For the simulations in this chapter we will use the adaptive dynamic prediction method introduced in the previous chapter. Before we can simulate our system we need to find initial conditions that result in a periodic trajectory that can be simulated in reasonable time on a desktop computer.

Boffi and Slotine uses a periodic trajectory for their experiment as well, but does not reveal the initial conditions of the system for the simulations. Multiple initial conditions from the orbits discovered by Šuvakov and Šinović [16] have been tested. Unfortunately the trajectories tested all exhibit the slingshot phenomenon where particles pass close to each other resulting in extreme gravitational forces making the trajectories to stiff for the simulator to solve in reasonable time.

A well behaved trajectory is the celebrated figure-8 trajectory discovered by Chenciner and Montgomery [15]. The particles keep sufficient distance to prevent the trajectory from becoming too stiff for the integrator. We will therefore use this trajectory in the simulations.

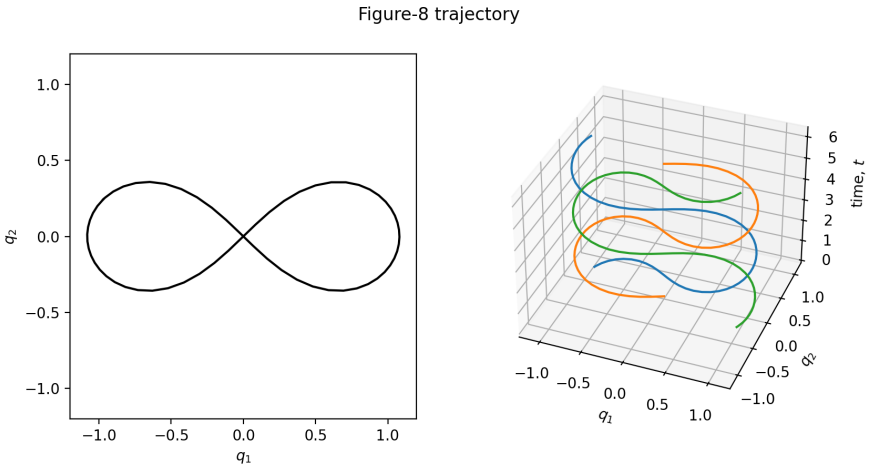


Figure 7.1: The figure-8 trajectory used in the simulations.

7.2 Model Based Dynamic Prediction

The first experiment is an implementation of the model based regressor introduced in the previous chapter. The experiment serves as a performance baseline for the dynamic prediction law under ideal conditions and an exact regression basis for the three-body system in the plane.

The experiment uses the dynamic predictor introduced in the previous chapter together with the Slotine-Li adaption law. The simulation is implemented in Python with the matrix library `numpy` using the `solve_ivp` integrator from the `scipy` library, which uses a RK45 algorithm with dynamic step size. For the plots the library `matplotlib` is used. Python as well as the libraries are free to use and open source.

Similarly to Boffi and Slotine we use the adaption gain $\gamma = 3.5$ and the feedback gain $\eta = 5$.

The initial conditions and true parameters of the system are set to produce the figure-8 trajectory from [15]:

$$g = m_i = 1 \tag{7.1}$$
$$q_1(0) = [-0.97000436, 0.24308753]^T \tag{7.2}$$
$$q_2(0) = [0, 0]^T \tag{7.3}$$
$$q_3(0) = -q_1(0) \tag{7.4}$$
$$p_1(0) = [0.4662036850, 0.4323657300]^T \tag{7.5}$$
$$p_2(0) = [-0.93240737, -0.86473146]^T \tag{7.6}$$
$$p_3(0) = p_1(0) \tag{7.7}$$

The predictor starts with the correct initial conditions $\hat{r}_0 = r_0$. The parameter

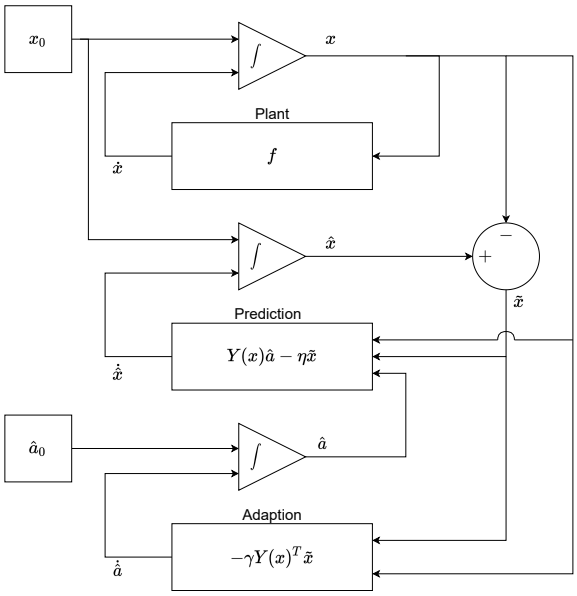


Figure 7.2: Block diagram for the dynamic predictor with the Slotine-Li adaption law

estimates are initialized at zero $\hat{a}_0 = 0$. The true parameters for the modeled basis in this setting is

$$a = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & 1 & 1 & 1 \end{bmatrix}^T \tag{7.8}$$

7.3 Implicit Regularized in Dynamic Prediction with an Overparameterized Function Basis

For the next two experiments the same simulator and system trajectory as in the model based experiment is used. The adaption law is changed from the Slotine-Li law to the mirror descent based adaption law and we use the overparameterized regression basis.

Like in the overparameterized regressor example in Chapter 4.3.1, we want to highlight how mirror descent based algorithms regularize the estimated parameters. We also want to analyze stability and tracking error of the dynamic predictor as this is important in a controls setting.

The adaption gain and feedback gain is kept the same as in the first experiment. The initial conditions of the adaption law $\hat{a} = 0.5\mathbb{1}$ are chosen to be "close" to the exact solution because this results in more prominent regularization, as explained by Azizan [3]. For the overparameterized basis the true parameters for this experiment are

$$a = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T \tag{7.9}$$

Before the simulations start the initial adaption parameters are transformed into the mirrored domain. The parameters are transformed back for the state prediction at each time step and at the end of the simulation. All the states are stacked in a simulator state vector x and integrated.

$$x = \begin{bmatrix} r \\ \hat{r} \\ \nabla\psi(\hat{a}) \end{bmatrix}, \; \dot{x} = \begin{bmatrix} J\nabla\mathcal{H}(r) \\ Y(r)\hat{a} - \eta\tilde{r} \\ -\gamma Y(r)^T\tilde{r} \end{bmatrix} \tag{7.10}$$

Because the periodic solution only explores a small subset of the state space we do not expect the signals to be sufficiently rich for the parameters to converge to their correct values. The main interest of the experiment is the parameter distribution produced by the adaption laws, tracking error, and the estimated Hamiltonian.

The second experiment consist of two parts, each with different potential. First we use the squared ℓ_2-norm which results in the Slotine-Li adaption law. Then we use the squared ℓ_1 norm which favours sparsity in the parameters. Because the ℓ_1 norm is not strictly convex the approximation $\ell_{1.05}$ is used instead.

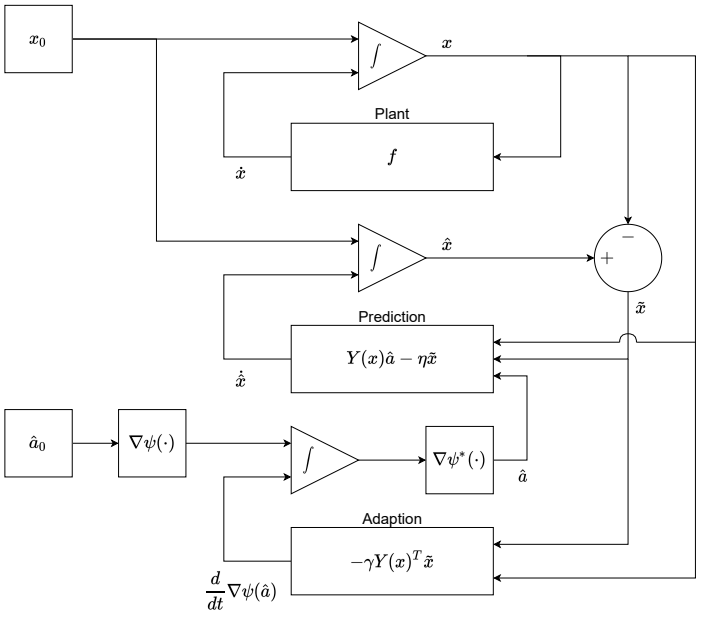


Figure 7.3: Block diagram for the dynamic predictor with the mirror descent based adaption law

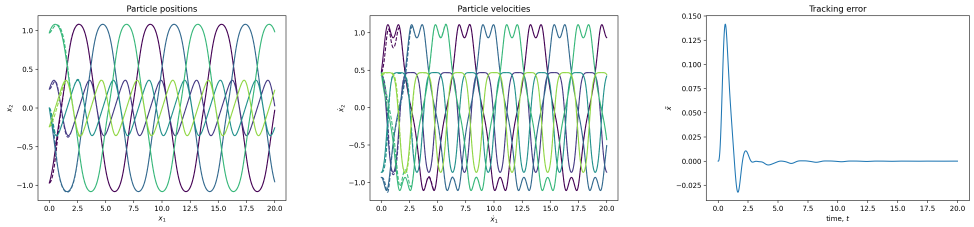
Chapter 8

Results

In this chapter the results of the experiments on the Hamiltonian three-body system is presented.

8.1 Model Based Dynamic Prediction

The first simulation confirms that the figure-8 trajectory can be followed by a dynamic predictor with minimal error. We observe that the parameters converge to the true parameters in approximately 15 time units in (Figure 8.2a). As expected the correct parameters result in zero tracking error (Figure 8.1c).



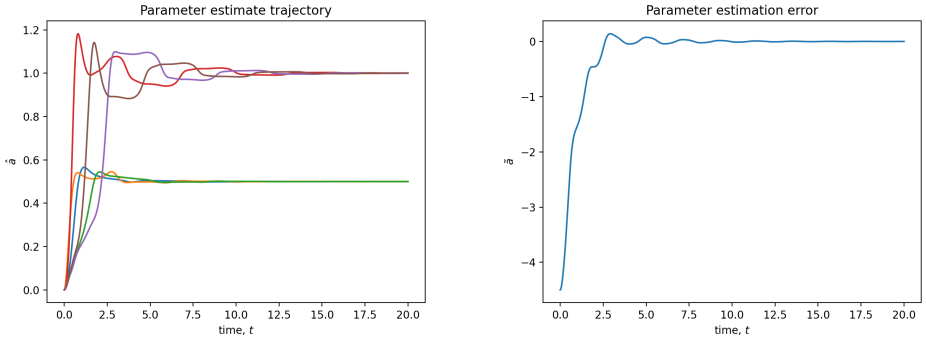
(a) **(b)** **(c)**

Figure 8.1: State trajectories and tracking error for the model based dynamic predictor. The true state in solid line and predicted state in dashed line.

8.2 Overparameterized Dynamic Prediction

8.2.1 Gradient Descent Adaption Law

The Slotine-Li adaption law converges to a set of parameters in approximately 20 time units (Figure 8.3a). In line with the theoretical results the parameters are small in the Euiclidean norm. The method finds the true parameters for the squared momentum terms at 0.5 but avoids the true potential parameters at 1.



(a) **(b)**

Figure 8.2: Parameter estimate trajectory with parameter estimation error for the model based dynamic predictor.

This is most likely because they are considered "expensive" parameters in the Euclidean norm compared to some other combination of smaller parameters that interpolate along the trajectory.

To better visualize the parameters learned by the method the estimated Hamiltonian resulting from the parameters is plotted in Figure 8.5. The plots are functions of the position and velocity of particle 1 at time $t = t_{\text{end}}$, while particle 2 and 3 are constant at their state at t_{end}. From the plot we see that the regressor is able to learn the Hamiltonian momentum perfectly apart from a constant bias. This is because the dynamic predictor uses the time derivative of the Hamiltonian and therefore a constant term in the Hamiltonian will not contribute to estimation error.

The Hamiltonian potential from the gravitational forces is fairly well approximated. The function is a good approximation along the state trajectory which is reflected in the tracking error. As the particle deviates from the figure-8 trajectory the error grows quickly as is illustrated in Figure 8.7b. As explained in the previous section the trajectory does not produce a sufficiently rich signal for the regressor to correctly learn the potential function. The function is wrong in the "unexplored" areas close to the other particles and as we move further away from them. This is not a problem because the particle does not reach that area. "Adaptive control is done on a need to know basis" — Slotine [18].

8.2.2 Mirror Descent based Adaption Law

From the results of the experiment on the ReLU regressor we expect sparsity from the $\ell_{1.05}$ mirror descent algorithm. The parameters converge more slowly than the Slotine-Li adaption law at about 100 time units (Figure 8.3c). We observe from the histogram that the parameters produced are indeed sparse

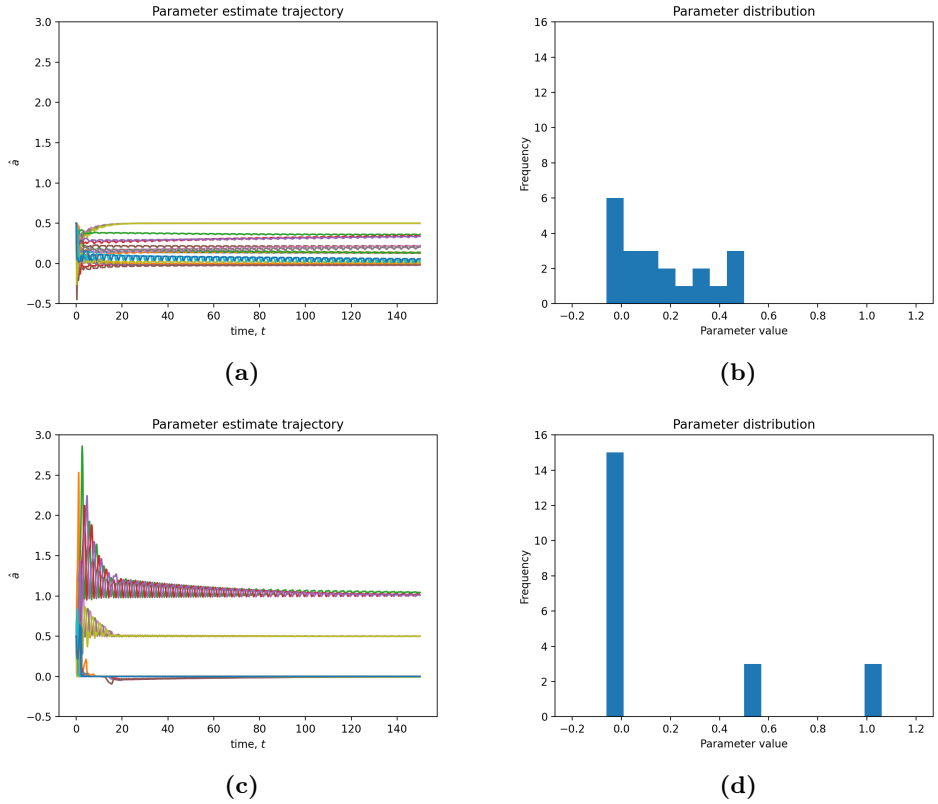


Figure 8.3: Parameter estimate trajectory and parameter distribution at time t_{end} for Slotine-Li **(a,b)** and $\ell_{1.05}$ **(c,d)**.

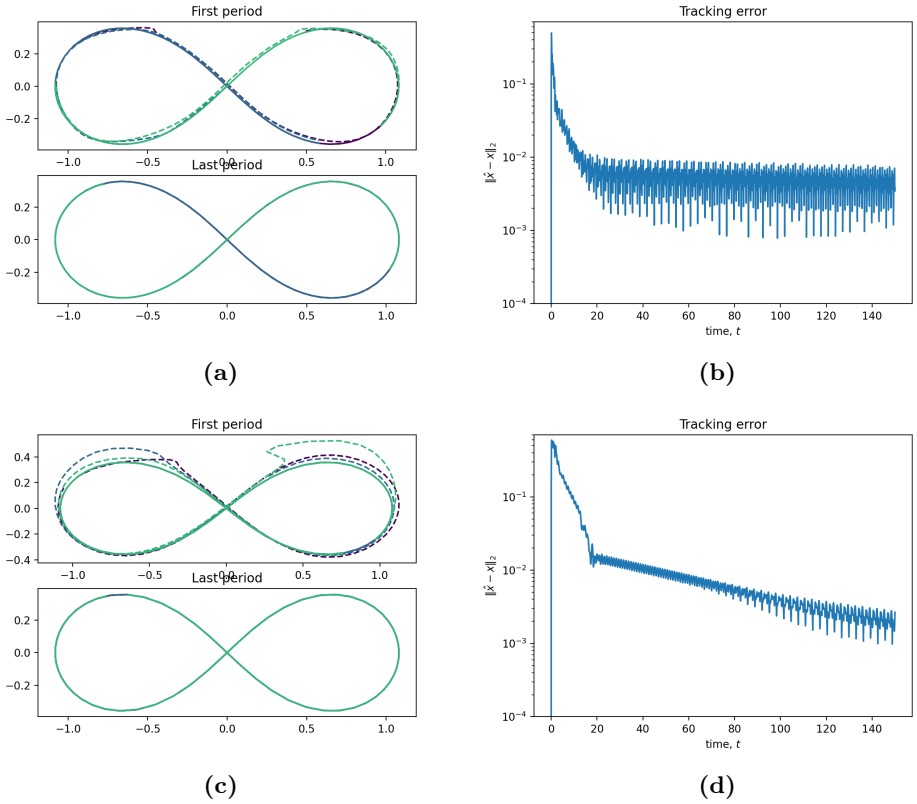


Figure 8.4: Position trajectory and tracking error on a logarithmic scale for the Slotine-Li **(a,b)** and the ℓ_1 **(c,d)** adaption laws.

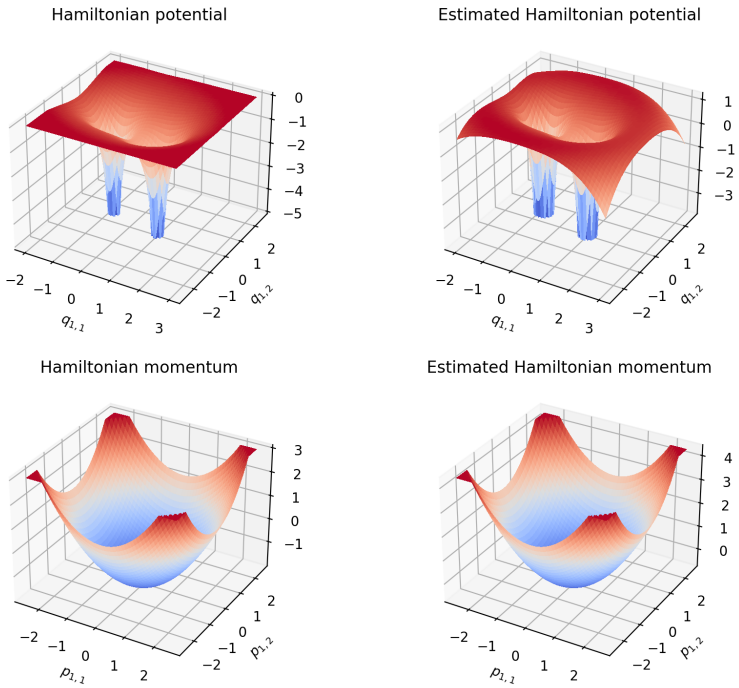


Figure 8.5: The two separable Hamiltonians learned by mirror descent based adaption law with a squared ℓ_2-norm mirror compared to the true Hamiltonians. The plots are truncated at the z-axis limits.

(Figure 8.3d) with 15 parameters set very close to zero. The adaption law almost converges to the true parameters even though it uses the exact same signals as the Slotine-Li adaption law. This demonstrates that the method can be effective for system identification. The tracking ability of the predictor is also better than the standard Slotine-Li version for this simulation.

The estimated Hamiltonian potential produced by the parameters looks similar to one produced by the Slotine-Li method, but does not curve down as steeply as the particle moves away from the trajectory (Figure 8.6). In the plot of the potential we see a steep spike in one of the gravity wells. This again is because this part of the state space is unexplored. The errors in the potentials stem from higher order terms that are not zeroed out completely, with high order basis functions the error will grow quickly as we diverge from the trajectory even for very small parameters.

The Hamiltonian momentum is almost perfectly estimated by the Slotine-Li law apart from a bias, which again does not effect the predictor.

In Figure 8.7a the Hamiltonian potential estimation error has been plotted adjusted for bias. For both adaption laws the estimated potential has a very small error close to the trajectory, but diverges rapidly as the particle moves away.

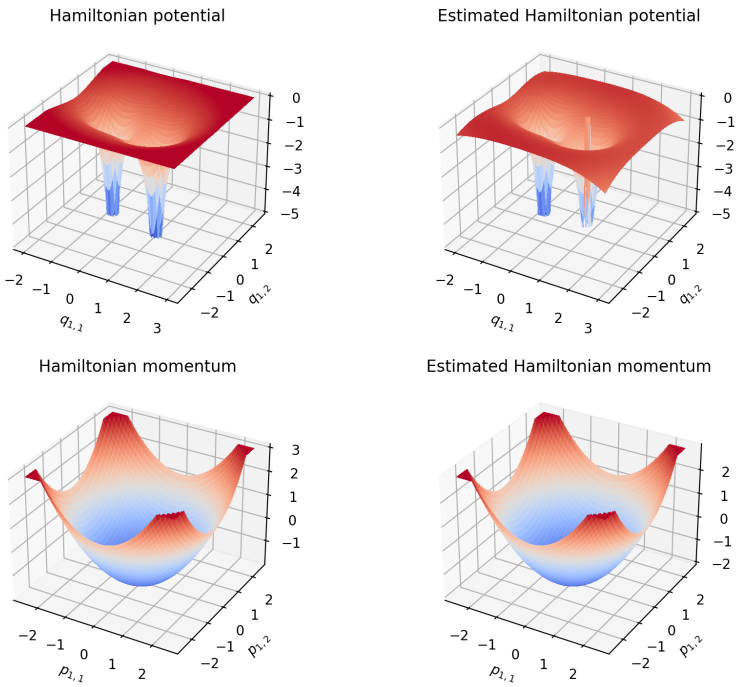
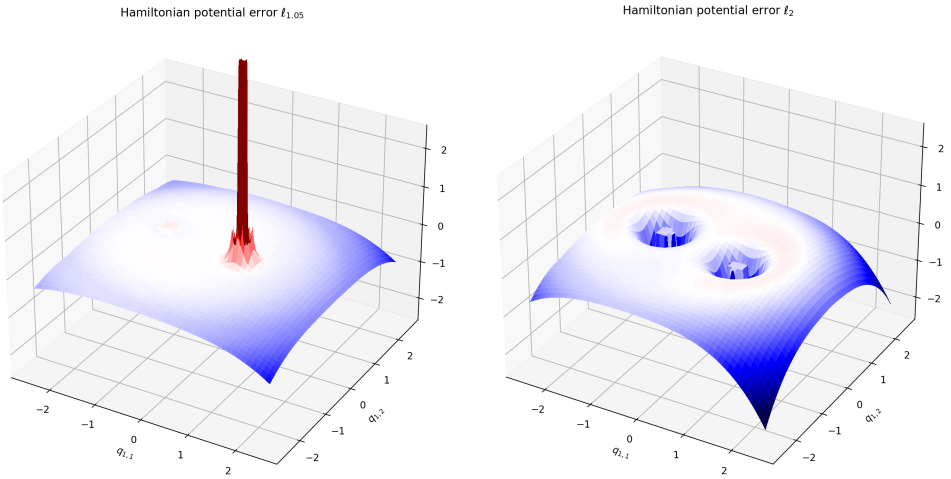


Figure 8.6: The two separable Hamiltonians learned by mirror descent based adaption law with a squared $\ell_{1.05}$-norm mirror compared to the true Hamiltonians. The plots are truncated at the z-axis limits.



(a) **(b)**

Figure 8.7: The Hamiltonian potential estimate error of the two adaption laws. The potentials have been adjusted for bias. the color map is normalized such that white corresponds to zero error.

Chapter 9

Discussion

The simulations presented in the previous chapter mostly confirm the analytical results on implicit regularization in overparameterized systems. However the regressor used in the simulations is not overparameterized enough for us to study the distribution like in [3] where the normal distribution of the parameters in the gradient descent case is very apparent. Still, the experimental results are consistent with the theoretical results as the Slotine-Li adaption law favours parameters small in the Euclidean norm and the ℓ_1 methods favours sparsity.

Interestingly, the model based regressor converges to the true parameter which suggests that the trajectory is persistently exciting for the modeled basis. The ℓ_1 adaption law with the overparameterized basis comes very close to the true parameters which points towards the trajectory being persistently exciting for the overparameterized basis as well. On the other hand, the Slotine-Li adaption law converges to a set of parameters different from the true parameters while still yielding a small tracking error. In both cases the size of the tracking error is small enough to stem from numerical errors in the integration. Thus, the small tracking errors of the methods makes it reasonable to believe that both parameter vectors are optimal along the trajectory suggesting that the local minima indeed are global minima. It is important to note that the three-body system is hard to simulate numerically and that some errors might be numerical. To further improve the simulations a symplectic integrator tailored to the three-body system may be utilized.

Another interesting modification is a larger regression basis. This would result in a larger optimal set of parameters leaving the adaption laws more room for regularization. It would also be interesting to investigate the use of generic basis functions like the ReLU basis used in the regression example.

Inspired by machine learning one could also divide the simulations into two phases. First a training trajectory to learn the parameters, and then a test-

ing trajectory with identical parameters to test the generalization of different adaption laws.

Chapter 10

Conclusion

In the present thesis we have covered the similarities between neural networks in machine learning and regression based adaption laws. More specifically the cause of implicit bias in gradient based optimization methods and how they affect regression problems have been examined. Furthermore, it has been demonstrated that the results extend to parameter estimation in adaptive nonlinear dynamical prediction both theoretically and empirically. It follows that by choice of potential function in a mirror descent based adaption law we can control the bias of the estimator to favour interpolating parameters with desirable properties. Concretely we have shown that the Slotine-Li law favours parameters small in the Euclidean norm while a mirror descent based adaption law with an approximate squared ℓ_1 potential favours sparse parameters.

Bibliography

[1] P. A. Ioannou, *Robust adaptive control*, eng. Mineola, N.Y: Dover Publications, 2012, ISBN: 9780486498171.

[2] S. Gunasekar, J. Lee, D. Soudry, and N. Srebro, "Implicit bias of gradient descent on linear convolutional networks," 2019. arXiv: 1806.00468 [cs.LG].

[3] N. Azizan, S. Lale, and B. Hassibi, "Stochastic mirror descent on overparameterized nonlinear models: Convergence, implicit regularization, and generalization," 2019. arXiv: 1906.03830 [cs.LG].

[4] N. M. Boffi and J.-J. E. Slotine, "Implicit regularization and momentum algorithms in nonlinear adaptive control and prediction," 2020. arXiv: 1912.13154 [math.OC].

[5] S. P. Boyd, *Convex optimization*. Cambridge University Press, 2004, ISBN: 9780521833783.

[6] R. T. Rockafellar, *Conjugate duality and optimization*, ser. CBMS-NSF regional conference series in applied mathematics ; Philadelphia, Pa.: Society for Industrial and Applied Mathematics (SIAM, 3600 Market Street, Floor 6, Philadelphia, PA 19104), 1974, vol. 16, ISBN: 1-61197-052-0.

[7] L. Bregman, "The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming," *USSR Computational Mathematics and Mathematical Physics*, vol. 7, no. 3, pp. 200–217, 1967, ISSN: 0041-5553. DOI: https://doi.org/10.1016/0041-5553(67)90040-7. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0041555367900407.

[8] A. Nemirovsky, D. Yudin, and E. Dawson, *Problem Complexity and Method Efficiency in Optimization*. Wiley, 1983, ISBN: 9780471103455.

[9] A. Beck and M. Teboulle, "Mirror descent and nonlinear projected subgradient methods for convex optimization," *Operations Research Letters*, vol. 31, no. 3, pp. 167–175, 2003, ISSN: 0167-6377. DOI: https://doi.org/10.1016/S0167-6377(02)00231-6. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167637702002316.

[10] T. J. Hastie, *The elements of statistical learning : data mining, inference, and prediction*, 2nd ed., ser. Springer series in statistics. New York: Springer Science + Business Media, 2009, ISBN: 9780387848570.

[11] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization," 2017. arXiv: 1611.03530 [cs.LG].

[12] Y. Cooper, "The loss landscape of overparameterized neural networks," 2018. arXiv: 1804.10200 [cs.LG].

[13] J.-J. E. Slotine, *Applied nonlinear control*, [5th printing]. Englewood Cliffs, N.J: Prentice-Hall, ISBN: 0130408905.

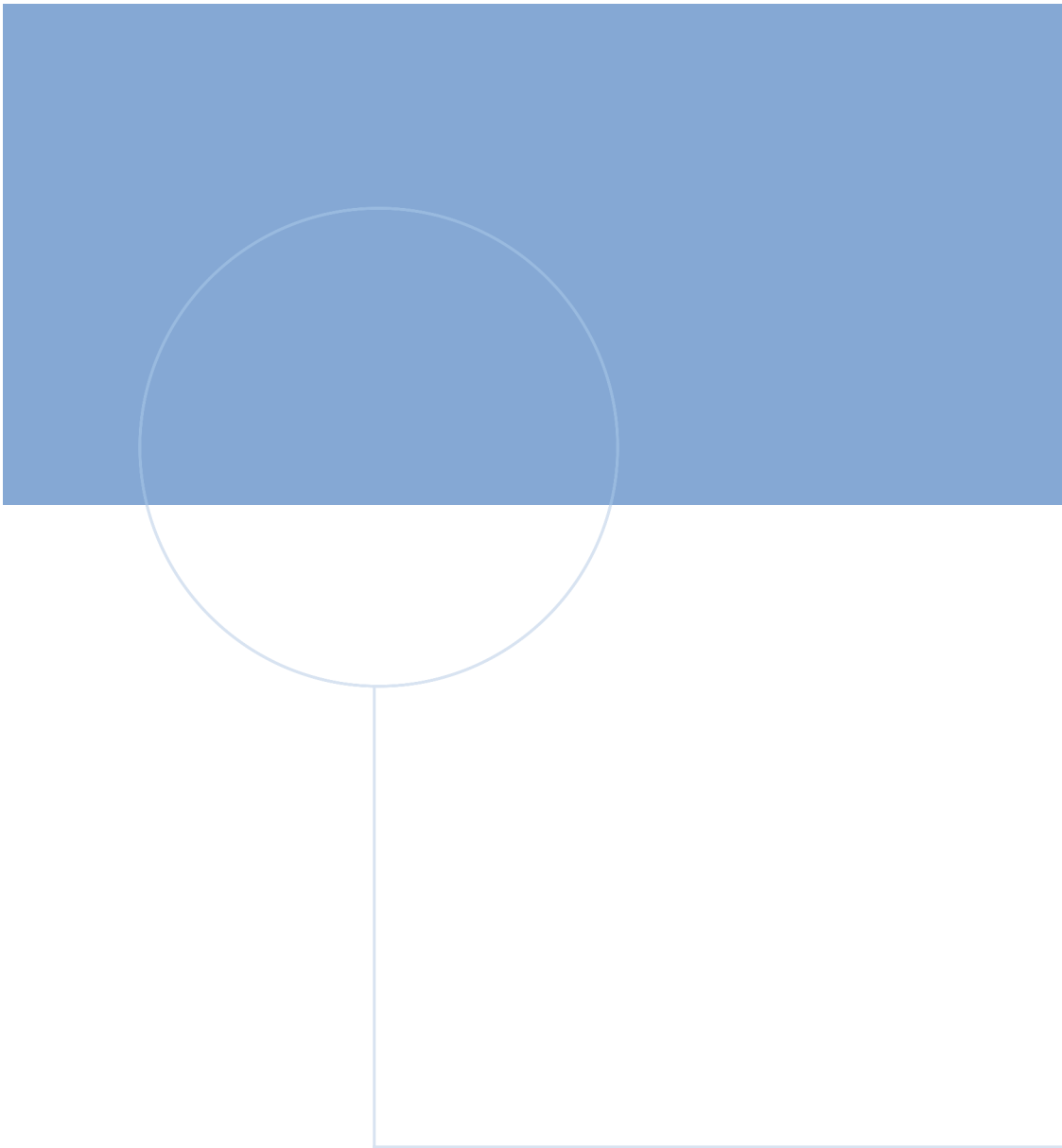
[14] K. Meyer, G. Hall, and D. Offin, *Introduction to Hamiltonian Dynamical Systems and the N-Body Problem*, eng, ser. Applied Mathematical Sciences. New York, NY: Springer New York, vol. 90, ISBN: 0387097236.

[15] A. Chenciner and R. Montgomery, "A remarkable periodic solution of the three-body problem in the case of equal masses," *arXiv Mathematics e-prints*, math/0011268, math/0011268, Oct. 2000. arXiv: math/0011268 [math.DS].

[16] M. Šuvakov and V. Dmitra Šinović, "Three classes of newtonian three-body planar periodic orbits," *Phys. Rev. Lett.*, vol. 110, p. 114 301, 11 Mar. 2013. DOI: 10.1103/PhysRevLett.110.114301. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.110.114301.

[17] X. Li, Y. Jing, and S. Liao, "Over a thousand new periodic orbits of a planar three-body system with unequal masses," *Publications of the Astronomical Society of Japan*, vol. 70, no. 4, May 2018, ISSN: 2053-051X. DOI: 10.1093/pasj/psy057. [Online]. Available: http://dx.doi.org/10.1093/pasj/psy057.

[18] *Slotine lectures on nonlinear systems, adaptive control*, Video Lecture, Accessed Jun. 3, 2021, Nonlinear Systems Laboratory, Massachusetts Institute of Technology, Fall 2013. [Online]. Available: https://www.youtube.com/watch?v=v9REipKWwyA.



NTNU

Norwegian University of
Science and Technology