WIP: PHYSICAL SENSING COUPLED WITH PHYSICS BASED MACHINE LEARNING

by

John Waczak

APPROVED BY SUPERVISORY COMMITTEE:

David J. Lary, Chair

Christopher Simmons

David Lumley

Lindsay King

Joseph Izen

Update required!

WIP: PHYSICAL SENSING COUPLED WITH PHYSICS BASED MACHINE LEARNING

by

JOHN WACZAK, BS, PhD

DISSERTATION

Presented to the Faculty of

The University of Texas at Dallas

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY IN

PHYSICS

THE UNIVERSITY OF TEXAS AT DALLAS

May 2024

ACKNOWLEDGMENTS

Update required! Make sure to include lab colleagues *and* ActivePure colleagues and a nod to Dr. Cooper, OSU teachers, and friends…

December 2012

WIP: PHYSICAL SENSING COUPLED WITH PHYSICS BASED MACHINE LEARNING

John Waczak, PhD
The University of Texas at Dallas, 2024

Supervising Professor: David J. Lary, Chair

Update required!

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

NOTE: For the proposal in particular, we should add a new section with timelines and a table of currently completed tasks.

NOTE: We want to emphasize the need for reproducible research paradigms. This motivates

NOTE: for the area of uncertainty quantification via statistical machine learning methods we should emphasize the fact that uncertainty is challenging to quantify in physical systems but in the age of big data, we need not really on the assumption of conveniently shaped normal distributions– we can do better. Instead, we let the data be our guide, deriving improved uncertainty bounds (in the statistical sense) with methods to track key diagnostic characteristics such as the relevant time scale. Further, in the domain of low-cost sensing, a key challenge is accurately quantifying uncertainty bounds for measurements as manufacturer provided estimates are almost always overly conservative (to minimize responsibility for variance, perhaps?)

NOTE: We should also mention (in our goals for the dissertation) that reproducibility is of critical importance (in physics, in machine learning, in scientific code, in analysis, etc…) with relevant statistics quoted (perhaps that nice interview from)

NOTE: explain Dr. Lary's principle of "Do the obvious thing: ask the obvious question, run the obvious experiment, perform the obvious analysis" as a guiding principle for the logical organization of this dissertation.

The goal of this thesis is advancing physical sensing in service of society to provide actionable insights. This goal is pursued by applying physics informed approaches together with a suite of sensing and computational technologies, implementing the reusable paradigm of software defined sensors, i.e. physical sensing elements wrapped in a software layer. This software layer can serve a variety of purposes such as calibration and the provision of enhanced or derived data products. It is part of a broader effort in the MINTS-AI laboratory at the University of Texas at Dallas. Where MINTS-AI is an acronym, **M**ulti-Scale **M**ulti-Use **Int**egrated **Int**elligent **Int**eractive **S**ensing in **S**ervice of **S**ociety for **A**ctionable **I**nsights.

Comprehensive environmental sensing is a timely and beneficial endeavor for a variety of reasons. The growing awareness of major environmental issues such as climate change, pollution, and habitat loss necessitates effective environmental monitoring and management. Comprehensive environmental sensing can provide real-time data on air and water quality, weather patterns, and other environmental factors, assisting in the identification and resolution of environmental issues. This assists in the development and implementation of policies and strategies aimed at reducing environmental impact and increasing sustainability. Given that, for instance, air quality can have significant effects on human health, this has particular societal value.

Exposure to air pollution has been linked to a wide range of health effects (Brook et al. 2010; Kelly and Fussell 2011; Xu et al. 2017), including respiratory and cardiovascular diseases, cancer, and adverse birth outcomes. Further, physical sensing provides valuable data and the basis for international assessments such as the Intergovernmental Panel on Climate Change (IPCC), which seeks to assess the science related to climate change and its impacts on natural and human systems (Houghton, Jenkins, and Ephraums 1990; Houghton et al. 1996, 2001; Solomon et al. 2007; Parry et al. 2007; Metz et al. 2007; Stocker et al. 2013; Field et al. 2014; Edenhofer et al. 2014; Masson-Delmotte et al. 2018; Friedlingstein et al. 2020; Huang et al. 2017).

Comprehensive sensing of the environment can improve decision-making. The real-time and accurate data provided by environmental sensors can aid in informed decision-making regarding various aspects such as traffic management, industrial regulation, and crop planning. For instance, data on air quality can be used to inform decisions about reducing pollution levels, while data on weather patterns can help farmers to plan their crops and reduce water usage. Comprehensive sensing of the environment can be instrumental in emergency response. Real-time data on weather patterns, air quality, water levels and resources, and seismic activity can help emergency responders to prepare for and respond to natural disasters such as hurricanes, floods, and earthquakes. The quick and accurate information can enable effective and timely response, potentially saving lives and reducing the impact of the disaster.

Many advances in technology have enabled the creation of comprehensive sensing systems that can monitor and analyze data from various sensors and devices in real-time. In this thesis we use a range of technologies including autonomous robotic teams (Dunbabin and Marques 2012; Rubenstein, Cornejo, and Nagpal 2014; Y. Chen et al. 2017), hyperspectral imaging (Plaza et al. 2009; Li et al. 2018; Zhu et al. 2017), mesh networks utilizing the Internet of Things (IoT) (Gubbi et al. 2013; Atzori, Iera, and Morabito 2010; Al-Fuqaha et al. 2015), machine learning (ML) (Goodfellow, Bengio, and Courville 2016; LeCun, Bengio, and Hinton 2015; Jordan and Mitchell 2015), edge computing, high-performance computing, wearable sensors and modern high-performance dynamic programming languages such as Julia (Bezanson et al. 2017) designed for numerical and scientific computing. These technologies have facilitated the collection and processing of large amounts of data from multiple sources, resulting in more accurate and comprehensive environmental monitoring.

1.1 Goals of this Dissertation

Test text

1.2 Global Change

Global change refers to the significant and long-term alterations in the Earth's physical, chemical, and biological systems, resulting from natural and human-induced processes (Edenhofer et al. 2014; Masson-Delmotte et al. 2018; United Nations 2015). This includes changes in the climate, land use, biodiversity, and biogeochemical cycles, as well as interactions among these systems. Global change can have profound impacts on natural and human systems, including altered weather patterns, sea level rise, increased frequency and severity of extreme events, loss of biodiversity and ecosystem services, and effects on human health and well-being. Understanding and managing global change is a critical challenge facing society today, requiring interdisciplinary approaches and collaboration across sectors and regions.

Global change can have a range of impacts on society, including environmental, social, and economic effects. Some of the aspects of global change that have the biggest impact on society include:

- Climate Change: Climate change, driven by human activities such as burning fossil fuels, deforestation, and land-use changes, has impacts on natural systems such as ocean acidification, sea level rise, and changes in precipitation patterns. These impacts can have cascading effects on human systems, including impacts on food security, water availability, and health.

- Biodiversity Loss: Global change can lead to the loss of biodiversity, which can have impacts on ecosystem functioning and services, such as pollination, pest control, and carbon storage. These impacts can have indirect effects on human well-being, including impacts on food security, health, and cultural heritage.

- Land Use Change: Land use change, such as deforestation, urbanization, and agriculture, can have impacts on natural systems such as soil quality, water availability, and biodiversity. These impacts can have direct and indirect effects on human systems, including impacts on food security, water availability, and cultural heritage.

- Economic and Social Inequality: Global change can exacerbate economic and social inequality, with impacts on access to resources, health, and well-being. These impacts can have cascading effects on the ability of societies to adapt and respond to global change.

- Human Health: Global change can have significant impacts on human health (World Health Organization 2018; Costello et al. 2009; Haines et al. 2006), both directly and indirectly, for example:

  - Heat-related Illness: As temperatures increase due to global warming, there is an increased risk of heat-related illness, including heat exhaustion and heat stroke, particularly in vulnerable populations such as the elderly, young children, and outdoor workers.
  - Air Pollution: Global change can lead to increased air pollution, including from sources such as wildfires and fossil fuel combustion. Exposure to air pollution can increase the risk of respiratory and cardiovascular diseases, including asthma, chronic obstructive pulmonary disease (COPD), and heart disease.
  - Vector-borne Diseases: Changes in temperature and precipitation patterns can affect the distribution and abundance of disease vectors such as mosquitoes and ticks, leading to increased risks of vector-borne diseases such as dengue fever, malaria, and Lyme disease.
  - Waterborne Diseases: Changes in precipitation patterns and water quality can increase the risk of waterborne diseases, including cholera and other diarrheal diseases.

  - Food Security: Global change can affect food production and availability, leading to food shortages and malnutrition, particularly in vulnerable populations such as children and pregnant women.

Effectively addressing these aspects of global change requires interdisciplinary approaches and collaboration across sectors and regions, as well as a commitment to sustainable development and equitable solutions. Adaptation and mitigation are two strategies for addressing global change, which differ in their focus and approach.

Adaptation involves taking measures to adjust and respond to the impacts of global change that are already occurring or are expected to occur in the future. This can include actions such as building sea walls to protect against sea level rise, developing drought-resistant crops, and improving public health infrastructure to address the increased risk of vector-borne diseases. Adaptation strategies aim to reduce the vulnerability of human and natural systems to the impacts of global change and increase their resilience.

Mitigation involves taking measures to reduce the drivers of global change, such as greenhouse gas emissions, land use change, and deforestation. This can include actions such as increasing energy efficiency, shifting to renewable energy sources, and reducing waste and consumption. Mitigation strategies aim to address the root causes of global change and reduce its severity and impact.

Both adaptation and mitigation are important strategies for addressing global change, but they differ in their focus and approach. Adaptation strategies focus on responding to the impacts of global change that are already occurring or are expected to occur in the future, while mitigation strategies focus on reducing the drivers of global change and preventing its impacts from occurring in the first place. A comprehensive approach to global change will require both adaptation and mitigation strategies, as well as efforts to promote sustainable development and equitable solutions.

1.3 The Role of Sensing

Sensing technologies can play a critical role in both adaptation and mitigation efforts by providing data and information that can inform decision-making and improve the effectiveness of strategies (United Nations Environment Programme 2017; National Research Council 2010; Centre for Ecology and Hydrology 2017).

In adaptation efforts, sensing technologies can provide real-time data on environmental conditions such as temperature, precipitation, sea level, air quality, as well as on the status and health of ecosystems and wildlife. This information can be used to inform early warning systems for natural disasters, to track the spread of vector-borne diseases, and to monitor the impacts of climate change on biodiversity and ecosystem services. Sensing technologies can also provide data on the effectiveness of adaptation measures, such as the performance of sea walls and other infrastructure.

In mitigation efforts, sensing technologies can provide data on greenhouse gas emissions and other drivers of global change, as well as on the effectiveness of mitigation measures such as renewable energy and carbon capture and storage. Sensing technologies can also be used to monitor and manage land use changes such as deforestation and urbanization, and to track the impacts of these changes on ecosystems and carbon storage.

Overall, sensing technologies can provide critical data and information for both adaptation and mitigation efforts, helping to improve decision-making and increase the effectiveness of strategies. The integration of sensing technologies with other tools such as modeling and data analysis can also help to identify new strategies and solutions for addressing global change. There are various sensing technologies and approaches used for monitoring the global environment. Here are some of the key ones:

- Remote Sensing: This technology involves using satellites and other airborne platforms to collect data on the Earth's atmosphere, land, and oceans. Remote sensing provides information on environmental parameters such as temperature, humidity, air quality, land use and land cover, and ocean temperature, salinity, and sea level (Thenkabail 2019; Buyantuyev and Wu 2017; Gamon et al. 2016; Wang et al. 2017; Pasher et al. 2019). Some examples of remote sensing include:
  - Lidar: This technology uses laser pulses to measure distance and can be used to create detailed three-dimensional maps of the environment. Lidar is commonly used to measure forest canopy height, but can also be used to measure atmospheric conditions such as cloud cover and aerosol concentrations.
  - Imaging Spectroscopy: This technology uses a combination of imaging and spectroscopy to measure the reflectance of different wavelengths of light. Imaging spectroscopy can be used to identify and map different types of vegetation and minerals, and can provide information on the health of plant communities.
  - Unmanned Aerial Vehicles (UAVs): These are remote-controlled or autonomous aircraft that can be equipped with sensors for remote and in-situ environmental monitoring. UAVs can be used for mapping and monitoring of large areas, and can collect high-resolution data on environmental conditions.
- In-Situ Sensors: These sensors are used to collect data directly from the environment at the location of interest. They can measure environmental parameters such as temperature, pressure, and humidity, as well as water quality and soil moisture. In situ sensors are commonly used in marine environments to measure ocean temperature, salinity, and other properties. Some examples of in-situ sensing include:
  - Weather Stations: These are automated weather monitoring systems that collect data on atmospheric conditions such as temperature, humidity, barometric

pressure, wind speed and direction, and precipitation. Weather stations can be installed on land or in the ocean to provide continuous monitoring of environmental conditions.

  - Ground-Based Sensors: These sensors are used to monitor the quality of air, water, and soil. They can detect and measure pollutants such as carbon dioxide, nitrogen dioxide, ozone, sulfur dioxide, and particulate matter. Ground-based sensors are installed in various locations such as cities, industrial sites, and rural areas to provide localized environmental monitoring.
  - Acoustic Sensors: These sensors are used to monitor environmental noise levels, including noise from traffic, industrial sources, and natural sources such as wind and waves. Acoustic sensors can provide information on noise levels over time and across different locations.

Overall, these sensing technologies play a critical role in monitoring the global environment and can provide valuable information for environmental research, management, and policy-making.

1.4 The Role of Computational Modelling

Computer modeling can play a valuable role in both understanding and predicting global change (J. Chen et al. 2019; Hantson et al. 2016; DeLucia et al. 2021; Oleson et al. 2013; Clark et al. 2016). For example:

- Climate Modeling: Computer models can be used to simulate the Earth's climate system and predict future climate conditions. These models can incorporate data on greenhouse gas emissions, land use changes, and other factors to project how the Earth's climate will change over time.

- Ecosystem Modeling: Computer models can be used to simulate how ecosystems will respond to changes in environmental conditions, such as changes in temperature, precipitation, and atmospheric composition. These models can help predict how changes in ecosystems will impact biodiversity, ecosystem services, and human well-being.
- Carbon Cycle Modeling: Computer models can be used to simulate the global carbon cycle, which is the exchange of carbon between the Earth's atmosphere, land, and oceans. These models can help predict how changes in carbon emissions and land use will impact atmospheric carbon dioxide concentrations and global climate.
- Air Quality Modeling: Computer models can be used to simulate air quality, including the dispersion of pollutants in the atmosphere. These models can help predict how changes in emissions and atmospheric conditions will impact air quality and human health.
- Hydrological Modeling: Computer models can be used to simulate the movement of water through the Earth's hydrological cycle. These models can help predict how changes in precipitation, land use, and other factors will impact water availability, quality, and distribution.

Overall, computer modeling can provide valuable insights into the complex processes and interactions that drive global change. These insights can inform policy decisions and help guide efforts to mitigate and adapt to the impacts of global change.

1.5 Key Technologies

1.5.1 Julia for Scientific Computing

Julia is designed to combine the ease of use and high-level abstractions of languages like Python with the performance of compiled languages like C++, achieving a unique combination of speed and productivity for numerical and scientific computing. Julia is a high-level, high-performance programming language designed for numerical and scientific computing. It combines the ease of use and readability of Python with the speed and efficiency of Fortran or C. Julia has a wide array of scientific computing functionality, making it a powerful language for numerical analysis, data science, and engineering. It has built-in support for arrays and linear algebra, as well as packages for differential equations, optimization, probabilistic programming, data analysis and visualization, parallel and distributed computing, and machine learning. Julia's combination of performance, expressiveness, and flexibility make it an excellent choice for scientific and engineering applications, allowing for high-level abstractions and rapid prototyping, while still providing low-level control and efficient execution.

Here are some examples of what can be done easily in Julia that may not be as easy or efficient in other widely used scientific computing languages such as Python or Fortran:

- Multiple dispatch: Julia has a powerful multiple dispatch system that allows for generic programming and efficient function overloading. This allows for more flexible and expressive code compared to traditional object-oriented programming (OOP) in Python. Multiple dispatch allows a function to behave differently based on the types and/or number of arguments passed to it. In other words, the behavior of a function can be 'dispatched' based on the specific types and/or number of arguments passed to it.
- Just-in-time (JIT) compilation: Julia's JIT compiler translates high-level Julia code into optimized machine code, making Julia programs run nearly as fast as C or Fortran. In contrast, Python code is interpreted, and Fortran requires pre-compilation.
- Distributed computing: Julia has built-in support for distributed computing, making it easy to parallelize and scale up computations across multiple processors or machines. This is not as easy to do in Python or Fortran.
- Units and Error Propagation: The Units package in Julia provides a powerful and flexible framework for handling physical units in computations, useful for error propagation

and dimensional analysis, helping to ensure that the results are accurate, consistent, easy to interpret, and dimensionally consistent.

- Built-in unit testing: Julia has a built-in testing framework that makes it easy to write and run unit tests for your code, ensuring that it works correctly.
- ISO characters: Julia supports the use of Greek and other ISO characters in variable and function names, which can make code more readable and expressive, especially in mathematical or scientific contexts.
- Interactive data visualization: Julia has a number of powerful data visualization packages, such as Plots.jl and Makie.jl, that allow for interactive, high-performance data visualization.
- Package management: Julia has a sophisticated package manager that makes it easy to install, manage, and use third-party packages in your code. This is not as easy to do in Fortran, and while Python has a package manager, Julia's package manager is faster and more reliable.
- Inline C/Fortran/Python/R/Matlab code: Julia allows for inline C, Fortran, Python, R or Matlab code, making it easy to use existing libraries and code written in these languages without having to rewrite everything in Julia.

1.5.2 Scientific and Physics-based Machine Learning

Scientific machine learning (SciML) refers to the application of Machine Learning (ML) techniques to scientific problems, where the goal is not only to make predictions but also to gain insights into the underlying physical processes (Raissi, Perdikaris, and Karniadakis 2019; Rackauckas et al. 2020; Carleo et al. 2019). SciML involves the integration of domain-specific knowledge and physical models with data-driven techniques, and it has the potential to revolutionize many areas of science and engineering. In this thesis we explore the use of Physics-based machine learning (PBML) (Raissi and Karniadakis 2021; Wu and Zhang 2021) for a variety of applications.

Recent examples include a paper by (Raissi, Perdikaris, and Karniadakis 2019) that introduces a physics-informed neural network (PINN) framework for solving nonlinear partial differential equations, a paper by (Rackauckas et al. 2020) that proposes a universal differential equation (UDE) approach to scientific machine learning, and a review article by (Carleo et al. 2019) that discusses the use of machine learning in various fields of physics, including condensed matter physics, high-energy physics, and quantum physics. PBML has several advantages over purely data-driven approaches, including:

- Improved generalization: PBML models incorporate prior knowledge of the underlying physics, resulting in models that are more interpretable and generalizable. This enables the models to make accurate predictions even with limited training data.
- Incorporation of physical constraints: PBML models can be designed to incorporate physical constraints, such as conservation laws, which can help to ensure physically consistent predictions.
- Improved interpretability: PBML models are more interpretable than purely data-driven models since they are designed to incorporate physical principles. This can enable scientists and engineers to gain deeper insights into the underlying mechanisms of the systems they are studying.
- Reduced data requirements: PBML models require less training data than purely data-driven models since they leverage the physics-based priors, reducing the need for large datasets to train accurate models.
- Better extrapolation: PBML models are better equipped to extrapolate beyond the training data since they incorporate knowledge of the underlying physics, enabling them to make more accurate predictions in new and unseen scenarios.

Overall, PBML has several advantages over purely data-driven approaches, including improved generalization, reduced data requirements, better extrapolation, incorporation of physical constraints, and improved interpretability, making it a valuable tool for scientific and engineering applications.

1.6 Machine Learning, Physics-Informed Machine Learning, Scientific Machine Learning

1.6.1 overview of types of ML

1.6.2 explain NN (i.e. neural network as a function + universal approximator)

1.6.3 explain Physics Informed NN(s)

1.6.4 explain SciML + UDE framework

1.6.5 use RLC circuit instead of Spring mass to introduce the concept

1.6.6 methods I've developed

- Ensembling and Super Learners
- Self Organizing Maps
- Generative Topographic Mapping

NOTE: perhaps add section on increasing concerns about indoor air quality (use EPA RFI document). I think ideally we should break down the introduction by sensing domain

CHAPTER 2

PHYSICAL CONTEXT

test text

more test text....

2.1 Pollution and Particulate Matter

2.2 Optical Properties of Aqueous Solutions

2.3 Chemical Reaction Kinetics

2.4 Photolysis

2.5 Indoor Air Quality

2.5.1 Better Indoor Air Quality Management To Help Reduce COVID-19 and Other Disease Transmission in Buildings: Technical Assistance Needs and Priorities to Improve Public Health

"Ventilation, filtration, and air cleaning in buildings are essential components of a multilayered approach to preventing disease transmission, including COVID-19."

This statement serves as the foundation for The Clean Air in Buildings Challenge and this RFI. We completely agree with this premise, but we believe the 'order of operations' - among ventilation, filtration, and air cleaning - should be the primary focus of inquiry, and effective and safe air cleaning innovations should be a focus deserving full consideration. We can reconcile multiple, often competing goals by informing and empowering schools and commercial buildings to adopt a Clean First' mentality (enVerid 2022b, 2022a), which we will explore in this response, as well as examine the ‘art of the possible’ by taking a critical look at the 21st century innovations we believe will be essential to the practical pursuit and achievement of The Clean Air in Buildings Challenge goals/objectives. We will demonstrate the necessary considerations of a Clean First’ (enVerid 2022b, 2022a) approach as well as highlight its additional benefits in this response, which include but are not limited to:

- Reduced mechanical system energy consumption.
- Reductions in healthcare-acquired infections (HAIs) that are statistically significant in healthcare settings.
- Real-world testing/proof of lower microbiological burden (total and culturable bacteria, fungi, and their spores) in addition to improved ventilation/filtration alone.
- Reduced school absenteeism.
- Increased resiliency to future pandemics and communicable diseases.
- Reduction in airborne communicable disease and allergen loads.

The real challenges posed by global environmental change, combined with rising utility costs as a result of inflation and geopolitical shocks to energy supply chains, have created an implicit dichotomy between climate and public health goals. As Prof. Bahnfleth has said,

2.6 Hyperspectral Imaging

The following are notes from the Manolakis textbook that I originally kept here. NOTE: we will need to either make new figures or correctly cite these for attribution.

2.6.1 Hyperspectal Imaging Sensors

- **Hyperspectral Sensors** aka imaging spectrometers
  - scanning mechanism
  - imaging system

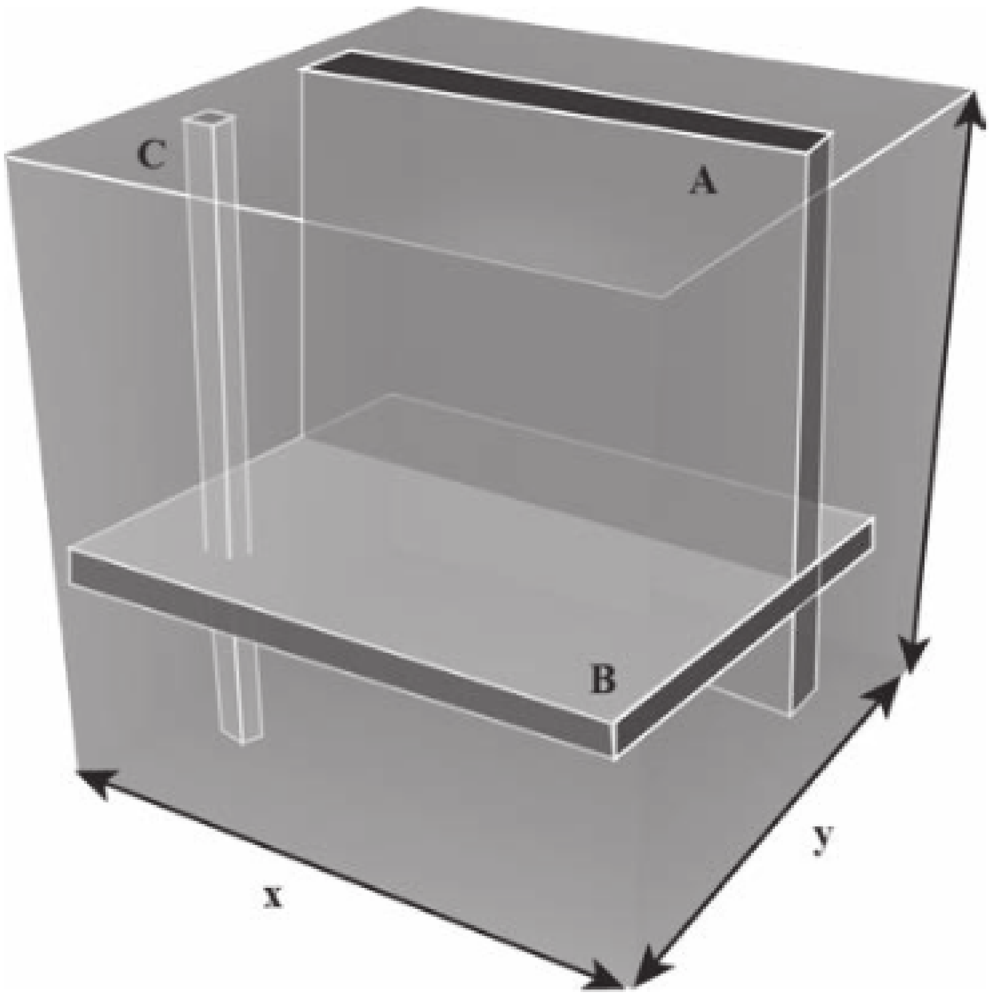
  - spectrometer

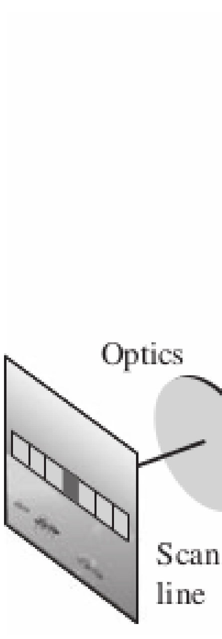
- 3 types of resolution

1. spatial
2. spectral
3. radiant
4. (temporal?)

2.6.1.1 Spectral-Spatial Data Collection and Organization

- Data collected into **Data Cube**
  - 2 spatial dimensions, 1 spectral dimension





- Different types of rigs:
  - Pushbroom scanner (ours)
  - Staring System
  - Fourier Transform Imaging Spectrometer (FTIS)

2.6.1.2 Spatial Sampling

- ground resolution elements are mapped to picture elements (pixels)
- **IFOV**: Instantaneous Field of View

- **Cross track dimension** the projection of the long axis of the slit (i.e. the axis of the pushbroom sensors)
- **Along track dimension** the direction accumulated by traveling
- **Ground Sample Distance** physical size of projected pixel element

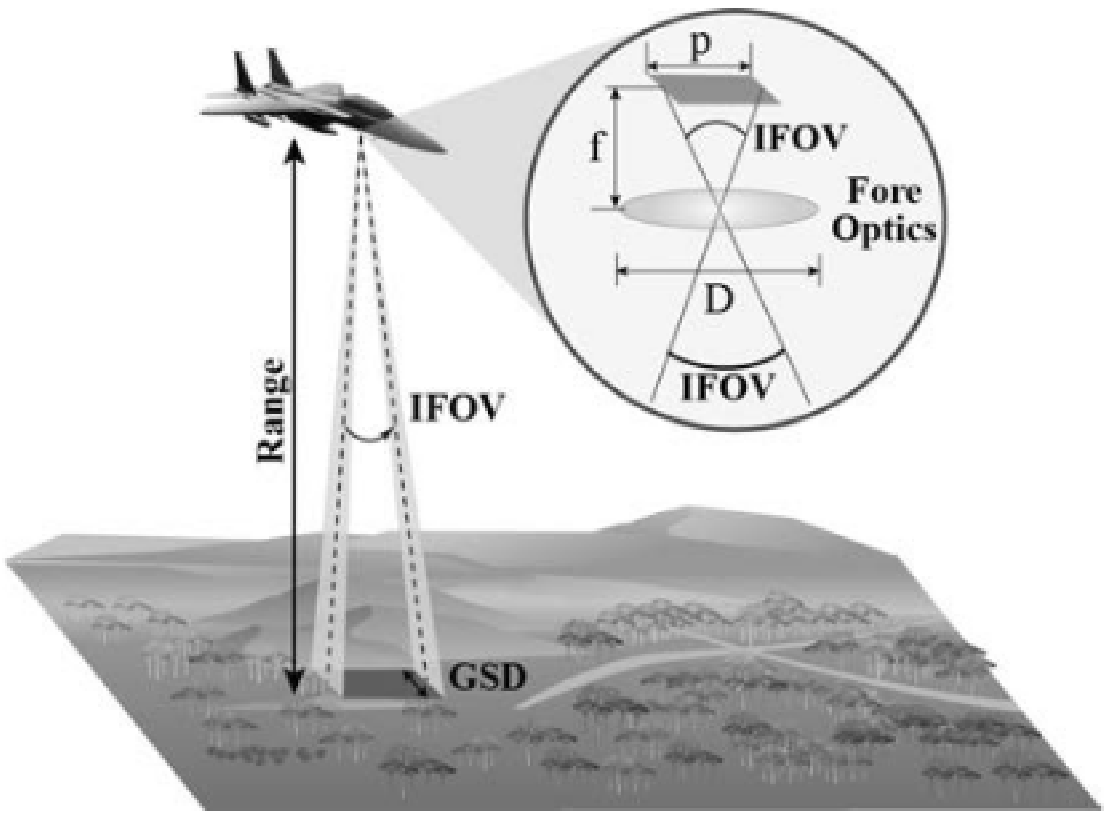


Figure 2.1: viewing

2.6.1.3 Spectral Sampling

- Recovery of spectral info is *imperfect* due to finite sampling
- **Spectral Response Function** is the weighing function that describes the wavelengths that are transmitted to a particular spectral sample

2.6.1.4 Radiometric Sampling

- detector transforms radiant power to electrical signal
- electrical signal converted to numbe via analog-to-digital converter
- photon detectors

2.6.1.5 Signal Consideratiosn

Strength of signal is determined by: - *Terrain composition* → affects amount of radiant energy reflected/emitted from ground resolution element - *Range* Intensity drops off by inverse square law. Further you are away, the worse the signal - *Spectral Bandwidth* output signal of detector element is proportional to spectral bandwidth of the detector - *Instantaneous Field of View* Decreasing IFOV increases spatial resolution but weakens the signal - *Dwell Time* the time required to sweep the IFOV across the ground resolution element, i.e. the time-on-pixel. Longer dwell time → more accumulated photons → more signal.

2.7 Remote Sensing

mention different types of satellite data platforms (mostly optical), differences in orbits, coverage, etc… Also good to discuss the increasing use of drones for a variety of applications including intelligent agriculture, geophysics, mapping, etc…

- **Remote Sensing**: data acquisition, processing, and interpretation of images, and related data, obtained from aircraft and satellites that record the interaction between matter and electromagnetic radiation
- **Source**: the source of electromagnetic radiation, e.g. the sun, black-body radiation, microwave radar, etc…
- **Atmospheric Radiation**: The EM radiation propagating through the atmosphere. Moderated by various processes including absorption and scattering

- **Earth's Surface Interation**: Amount and spectral distribution of radiation emitted/reflected by the earth's surface. This depends on
  - physical properties of the matter
  - wavelength of EM radiation that is sensed

2.7.1 Infrared Sensing Phenomenology

- Main passive sources of EM radiation for remote sensing are light emitted by the sun and the self-emission via black-body radiation of objects due to their temperature.

2.7.1.1 Sources of Infrared Radiation

- **spectral radiant exitance** power per unit area emitted by the sun. We can treat this as a black body with temperature $5800K$, maximum emittance at $\lambda = 0.50\ \mu m$.
- The Earth is ~$300K$ with maximum spectral radiant emittance at $\lambda = 9.7\ \mu m$. This is known as the **thermal infrared**

2.7.1.2 Atmospheric Propagation

- Key parameter is the **path length** of atmosphered traveled through before it arrives at the remote sensing system. Main effects are:
  - **Atmospheric Scattering**: diffusion of radiation by particles in the atmosphere
  - **Absorption**
- Useful remote sensing spectral regions are obtained via the **Transmission Spectrum**.
  - **Reflective Range**: $0.35 - 2.5\ \mu m$. Dominated by solar illumination
  - **Water Absorption**: $0.2 - 2.5\ \mu m$.
- **Atmospheric Windows**: Regions of low atmospheric absorption

2.7.1.3 Reflectance and Emissivity Spectra

There are three processes that occur when EM radiation meets and interface:

1. **Reflection**: Solar illumination dominates here. Consequently, this part of the spectrum is used to characterize the surface
    - *Specular Reflectors*: Flat surfaces that act like mirrors, i.e. $\theta_i = \theta_r$.
    - *Diffuse (Lambertian) Reflectors*: Rough surfaces that reflect uniformly in all directions.
    - *Real Reflectors*: Somewhere between the specular and diffuse.
2. **Absorption**
3. **Transmission**

- Fractions vary as a function of λ
- Remote sensing usually cares about *diffuse* reflectors because this is the dominant type of most materials (water being an exception).
- Reflectance of a material is characterized by its **Reflectance Spectrum**, that is, the percent of incident light reflected as a function of wavelength.
    - Dips in reflectance spectrum are called **absorption features**
    - Peaks are called **Reflectance Peaks**
- **Emissivity Spectrum**: The ratio of radiant emittance at a given temperature to the radiant emittance of a black body at the same temperature.

2.8 Solar Geometry

An explanation of relevant solar angles as well as their determination (i.e. the code I ported to Julia from Matlab script Dr. Lary supplied). We should also comment on the importance of

CHAPTER 3

PHYSICAL SENSING

3.1 Low Cost Sensors for (Outdoor) Air Quality

The outdoor part here is optional. Here I seek to provide specific details of the classes of low cost sensors used for PM (optical particle counters based on Mie scattering), thermistors (for tempature), humidity sensors, pressure sensors, gas sensors, etc... This can be a rough outline for those that specifically are used in my research projects.

3.2 Sensing Chemical Concentration in Aqueous Environments

Here we can give an overview of the sensors utilized on the boat for the robot team studies... We should include any information about the uncertainties. We can also comment on their use in fresh v.s. salt water environments.

3.3 Coordinated Robot Teams for Data Acquisition

Here we can discuss the drone, boat, and walking robot as well as long range communication setup with wifi router, transfering of files, MQTT protocol, and real time mapping with Grafana.

3.4 The HEART Chamber

For specifics of sensing equipment, see EPA RFI we did for Mr. Urso

CHAPTER 4

COMPUTATIONAL TOOLS

4.1 The Julia Programming Language

Overview of key features of Julia that made it attractive for this work:

- reproducibility
- composability
- readability (greek symbols as variables and operators)
- speed

4.2 Reproducible Research

principles & implementation: version control, CI-CD, test-driven-development, containerization

We should also discuss how *this dissertation is built* using quarto and github workflows. Similarly, we can discuss the process of building quarto templates via pandocs to target specific journals as well as pdf, web, and slides formats.

4.3 Eigen-stuff and the Singular Value Decomposition

Begin with a general description of the utility of eigen-stuff type analysis. Then explain how the SVD is the natural extension of this to non-square systems (rectangular matrices). Further expand by illustrating the application to principal component analysis. Make sure to comment on the general utility of decomposing a function/vector/signal into a (infinite) linear combination of (stationary) modes with simple time evolution.

4.4 Optimization Methods

Describe standard gradient descent and it's utility for machine learning. Expand to briefly describe the extensions used in our work:

- ADAM
- BFGS
- LBFGS

4.5 High Performance Computing

Provide an overview of relevant concepts in high performance computing i.e.

- slurm
- multi-threading
- parallelization (distribured computing)
- Memory management (i.e. preallocating data containers and writing functions that mutate, not allocate)

CHAPTER 5

THEORETICAL TOOLS

5.1 Automatic Differentiation

perhaps this can be moved to the Computational tools section instead.

Use Chris Rackauckas and Chris Olah blogs to start with chain rule and justify automatic differentiation. Then describe reverse mode via gradient tapes and the pullback operator.

5.2 Dual Numbers

Describe dual numbers because they are cool (and by extension, forward mode AD)

5.3 Embedding Theorems

Discuss Taken's embedding theorem and other relevant information for the time-series work.

5.4 Koopman Operator Theory

Give an overview of continuous and discrete Koopman operator theory.

5.5 Elements of Bayesian Statistics

Derive Baye's rule and other important concepts

5.6 Maximum Likelihood Estimation

5.7 KL Divergence

5.8 A Survey of Relevant Mathematical Structures

generic overview of role of mathematical structures in physics and machine learning

5.9 Uncertainty Propagation

Linear v.s. Nonlinear Techniques, measurements.jl

5.10 Kernelization

5.11 Dynamical Systems

can quote paper on origins of term *phase space*

CHAPTER 6

MACHINE LEARNING AND DATA-DRIVEN METHODS

NOTE: Good way to start is with the question: What is a model? We can discuss the differences between mechanistic models (i.e. physics equations), their free parameters (constants of nature) and other types of models such as the non-parametric, nonlinear models used in machine learning. Further, we can discuss how machine learning models often display have the desirable trait of being a *universal approximator*. This will naturally lead to a discussion of function expansions (Taylor, Fourier, other polynomial expansions, etc...) and how they scale poorly (exponential) with increasing dimension. Machine learning models essentially allow us to do the same thing: perform a function expansion given data but in a way that can scale well to arbitrary dimension (features) of data. This allows us to build predictive models without necessarily needing to prescribe *all* of the physics. From another perspective, this framework allows us to incorporate our physics knowledge from well-behaved or linearized domains in order to *fit the residual* behavior with a sophisticated data-driven approach.

discuss role in science in particular (i.e. calibration, modeling, etc...)

discuss pushback against use in science and need for methods which simultaneously provide uncertainty bounds. Also describe types of ML e.g. supervised, unsupervised, generative, etc...

this is a good place for the Chihuahua vs Muffin meme... and use this to motivate incorportating physical knowledge into the machine learning process as a key feature for scientific applications... we have more constraints!

also discuss statistical v.s. deep learning

6.1 Data Sampling

cross-validation techniques

Dr. Lary's method (from Gaussian Process Code) for representative sampling to reduce data size

6.2 Neural Networks

From a perspective of basic function composition... as in Rackauckas's blog

6.3 Decision Trees

6.4 Gaussian Process Regression

Based on my notes from this repo

6.4.1 Introduction

The following is based on the book **Gaussian Processes for Machine Learning** by *Carl Edward Rasmussen and Christopher K. I. Williams.* You can find the free online book here.[1]

To explain/derive the Gaussian Process model for regression, let's first consider a motivated example: **Linear Regression**. We will use this guiding example to derive GRP from a *weight space view.* After this derivation, will suggest a simpler, but more abstract derivation using a *function space view.*

First let's set up some data we can use for training:

[1]Carl Edward Rasmussen and Christopher K. I. Williams; The MIT Press, 2006. ISBN 0-262-18253-X.

6.4.2 Weight-Space View

6.4.2.1 Nomenclature

We consider a dataset $\mathcal{D}$ with n observations

$$\mathcal{D} = \left\{(\mathbf{x}_i, y_i) \,\middle|\, i = 1, ..., n\right\} \tag{6.1}$$

- $\mathbf{x}_i$ is the i^{th} D-dimensional input (feature) vector
- y_i is the i^{th} target

Linear regression is easily understood in terms of *linear algebra*. We therefore collect our dataset $\mathcal{D}$ into a $D \times n$ dimensional **Design Matrix**. Note that we have used a transposed definition (features are rows, records are columns) as Julia is a column-major language (like Matlab & Fortran).

$$X := \begin{pmatrix} \vdots & \vdots & & \vdots \\ \mathbf{x}_1 & \mathbf{x}_2 & \dots & \mathbf{x}_n \\ \vdots & \vdots & & \vdots \end{pmatrix} \tag{6.2}$$

and our targets into a target vector

$$\mathbf{y} := (y_1, ..., y_n) \tag{6.3}$$

so that the full training set becomes

$$\mathcal{D} := (X, \mathbf{y}) \tag{6.4}$$

6.4.2.2 Standard Linear Regression

Standard linear regression is a model of the form

$$f(\mathbf{x}) = \mathbf{x}^T \mathbf{w} \tag{6.5}$$

where $\mathbf{w}$ is the D-dimensional vector of weights. By minimizing the mean-squared-error between our model and targets, one can show that the optimal weights are given by

$$\mathbf{w} = (XX^T)^{-1}X\mathbf{y} \tag{6.6}$$

Note

This can also be easily obtained geometrically by finding the vector with the shortest distance to the hyperplane defined by the column space of X. This corresponds to solving the **normal equations**

$$XX^T\mathbf{w} = X\mathbf{y} \tag{6.7}$$

The following demonstrates this procedure on a simple dataset

Note

We can also fit a y-intercept (aka *bias*) by augmenting the design matrix X to contain an extra row with all `1`'s, i.e.

$$X[D+1,:] = (1, ..., 1) \tag{6.8}$$

6.4.3 Making it Bayesian

Standard linear regression assumes that are data $\mathcal{D}$ are perfect but we can clearly see that the above data are noisy. To account for this, we need to make our model *Bayesian* by augmenting it to consider measurement error. We define

$$f(\mathbf{x}) = \mathbf{x}^T\mathbf{w} \tag{6.9}$$

$$\mathbf{y} = f(\mathbf{x}) + \tag{6.10}$$

$$\sim \mathcal{N}(0, \sigma_n^2) \tag{6.11}$$

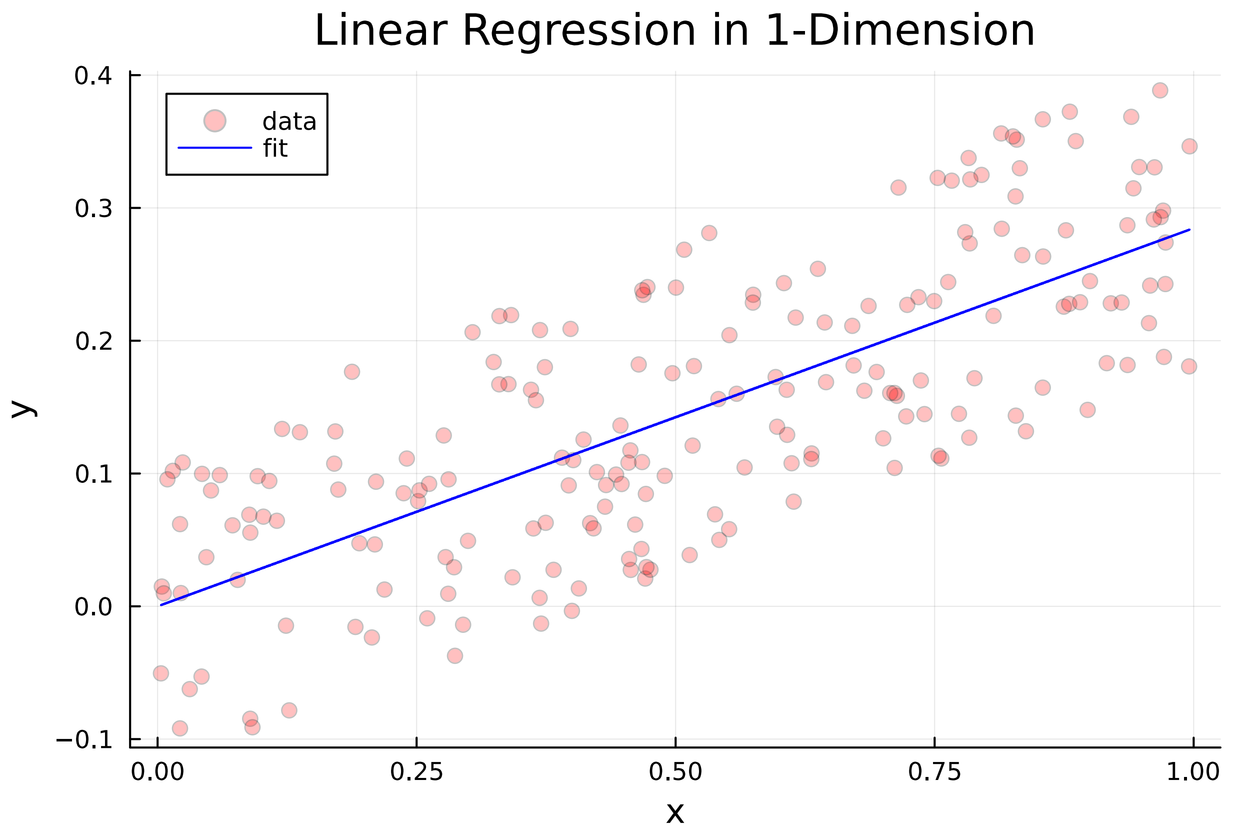


Figure 6.1: Linear Regression

or, in words, our observed values differ from the *truth* by identically, independently, distributed Gaussian noise with mean 0 and variance σ_n^2. The assumption that the noise is i.i.d. is critical because it allows us to simplify the *likelihood* function by separating out each individual contribution by our datapoints:

$$
\begin{aligned}
p(\mathbf{y}|X,\mathbf{w}) &:= \prod_i^n p(\mathbf{y}_i|\mathbf{x}_i,\mathbf{w}) && (6.12)\\
&= \prod_i^n \frac{1}{\sqrt{2\pi\sigma_n^2}} \exp\left(-\frac{(\mathbf{y}_i-\mathbf{x}_i^T\mathbf{w})^2}{2\sigma_n^2}\right) && (6.13)\\
&= \frac{1}{(2\pi\sigma_n^2)^{n/2}} \exp\left(-\frac{1}{2\sigma_n^2}|\mathbf{y}-X^T\mathbf{w}|^2\right) && (6.14)\\
&= \mathcal{N}\left(X^T\mathbf{w}, \sigma_n^2 I\right) && (6.15)
\end{aligned}
$$

To perform inference with this updated model, we apply Baye's Rule, that is:

$$p(\mathbf{w}|\mathbf{y}, X) = \frac{p(\mathbf{y}|X, \mathbf{w})p(\mathbf{w})}{p(\mathbf{y}|X)} \tag{6.16}$$

where

- $p(\mathbf{w}|\mathbf{y}, X)$ is the **posterior distribution**
- $p(\mathbf{y}|X, \mathbf{w})$ is the **likelihood**
- $p(\mathbf{w})$ is the **prior distribution**
- $p(\mathbf{y}|X)$ is the **marginal likelihood**, i.e. the normalization constant

It is now that the utility of choosing gaussian distributions for our likelihood and prior becomes clear...

$$p(\mathbf{w}|\mathbf{y}, X) \propto \exp\left(-\frac{1}{2\sigma_n^2}(\mathbf{y} - X^T\mathbf{w})^T(\mathbf{y} - X^T\mathbf{w})\right) \exp\left(-\frac{1}{2}\mathbf{w}^T\Sigma_p^{-1}\mathbf{w}\right) \tag{6.17}$$

Taking the log and expanding leads to

$$\log(p(\mathbf{w}|\mathbf{y}, X)) = \frac{1}{2}\left[\frac{1}{\sigma_n^2}\mathbf{y}^T\mathbf{y} - \frac{1}{\sigma_n^2}\mathbf{y}^T X^T\mathbf{w} - \frac{1}{\sigma_n^2}\mathbf{w}^T X\mathbf{y} + \frac{1}{\sigma_n^2}\mathbf{w}^T XX^T\mathbf{w} + \mathbf{w}^T\Sigma_p^{-1}\mathbf{w}\right] \tag{6.18}$$

$$= \frac{1}{2}\left[\mathbf{w}^T\left(\frac{1}{\sigma_n^2}XX^T + \Sigma_p^{-1}\right)\mathbf{w} - \left(\frac{1}{\sigma_n^2}\mathbf{y}^T X^T\right)\mathbf{w} - \mathbf{w}^T\left(\frac{1}{\sigma_n^2}X\mathbf{y}\right) + \mathbf{y}^T\frac{1}{\sigma_n^2}\mathbf{y}\right] \tag{6.19}$$

$$= \mathbf{w}^T A\mathbf{w} - B^T\mathbf{w} - \mathbf{w}^T B + C \tag{6.20}$$

where we have defined

$$A := \frac{1}{\sigma_n^2}XX^T + \Sigma_p^{-1} \tag{6.21}$$

$$B := \frac{1}{\sigma_n^2}X\mathbf{y} \tag{6.22}$$

$$C := \mathbf{y}^T\frac{1}{\sigma_n^2}\mathbf{y} \tag{6.23}$$

Now we can complete the square so that

$$\mathbf{w}^T A\mathbf{w} - B^T\mathbf{w} - \mathbf{w}^T B + C = (\mathbf{w} - \bar{\mathbf{w}})^T A (\mathbf{w} - \bar{\mathbf{w}}) + K \tag{6.24}$$

leading to

$$\bar{\mathbf{w}} = A^{-1}B = \frac{1}{\sigma_n^2}\left(\frac{1}{\sigma_n^2}XX^T + \Sigma_p^{-1}\right)^{-1} X\mathbf{y} \tag{6.25}$$

$$K = C - \bar{\mathbf{w}}^T A\bar{\mathbf{w}} \tag{6.26}$$

Since K does not depend on $\mathbf{w}$ directly, it may be absorbed into the normalization of $p(\mathbf{w}|\mathbf{y}, X)$. Thus we are left with

$$p(\mathbf{w}|\mathbf{y}, X) = \mathcal{N}\left(\bar{\mathbf{w}} = \frac{1}{\sigma_n^2}A^{-1}X\mathbf{y}, \Sigma = A^{-1}\right) \tag{6.27}$$

$$A = \frac{1}{\sigma_n^2}XX^T + \Sigma_p^{-1} \tag{6.28}$$

This result gives us the gaussian distriubtion over the space of possible parameter vectors $\mathbf{w}$. To use this distribution to make predictions, consider a newly supplied testpoint $\mathbf{x}_*$. We want to find

$$p(y_*|\mathbf{x}_*, \mathbf{y}, X) \tag{6.29}$$

We do this by marginalizing over our weight distribution, i.e.

$$p(y_*|\mathbf{x}_*, \mathbf{y}, X) = \int_{\mathbf{w}} p(y_*|\mathbf{x}_*, \mathbf{w})p(\mathbf{w}|\mathbf{y}, X)d\mathbf{w} \tag{6.30}$$

If we make the further assumption that testing points are i.i.d. guassian distriubted, we see that this integral is the product of two gaussians and therefore is also a guassian. To find the mean and covariance of the predictive distribution, we check

$$\bar{y}_* = \mathbb{E}[y_*] = \mathbb{E}[\mathbf{x}_*^T \mathbf{w}] = \mathbf{x}_*^T \mathbb{E}[\mathbf{w}] = \mathbf{x}_*^T \bar{\mathbf{w}} \tag{6.31}$$

$$\begin{aligned}
\text{Cov}(y_*) &= \mathbb{E}[(y_* - \bar{y}_*)(y_* - \bar{y}_*)^T] && (6.32)\\
&= \mathbb{E}[(\mathbf{x}_*^T \mathbf{w} - \mathbf{x}_*^T \bar{\mathbf{w}})(\mathbf{x}_*^T \mathbf{w} - \mathbf{x}_*^T \bar{\mathbf{w}})^T] && (6.33)\\
&= \mathbb{E}[\mathbf{x}_*^T (\mathbf{w} - \bar{\mathbf{w}})(\mathbf{w} - \bar{\mathbf{w}})^T \mathbf{x}_*] && (6.34)\\
&= \mathbf{x}_*^T \mathbb{E}[(\mathbf{w} - \bar{\mathbf{w}})(\mathbf{w} - \bar{\mathbf{w}})^T] \mathbf{x}_* && (6.35)\\
&= \mathbf{x}_*^T \text{Cov}(\mathbf{w}) \mathbf{x}_* && (6.36)\\
&= \mathbf{x}_*^T A^{-1} \mathbf{x}_* && (6.37)
\end{aligned}$$

so that

$$\boxed{p(y_* | \mathbf{x}_*, \mathbf{y}, X) = \mathcal{N}\left(\mathbf{x}_*^T \mathbf{w},\ \mathbf{x}_*^T A^{-1} \mathbf{x}_*\right)} \tag{6.38}$$

6.4.4 Doing More with Less: Kernelization

Let's take a break from our Bayesian regression and return to the standard linear regression model for a moment. The key drawback of linear models like this is, of course, that they're *linear*!. Considering that many (most?) *interesting* relationships are not linear, how can we extend our simple linear model to enable us to perform complicated non-linear fits?

In the parlance of machine learning, the simple solution is to do **feature engineering**. If our inital feature vector is

$$\mathbf{x} = (x_1, ..., x_n) \tag{6.39}$$

we can use our *expertise* to concoct new combinations of these features to produce the agumented vector

$$\tilde{\mathbf{x}} = (x_1, ..., x_n, x_1^2,\ sin(x_2),\ x_5 x_7 / x_4,\ ...) \tag{6.40}$$

As an example, a linear classifier is unable to distinguish points inside a circle from those outside just from the (x, y) coordinates alone. Augmenting the feature vector to include the squared radius $x^2 + y^2$ as a new feature removes this obstacle.

Note

This works because the *linear* part of linear regression only refers to the fact that our model takes *linear combinations* of feature variables to produce it's output. There is no restriction that the features themselves need to be independent variables! This same idea is what makes methods like SINDy work...

Constructing new features is often more art than science. To standardize the process, let's abstract mapping from the original feature vector $\mathbf{x}$ to the augmented vector $\tilde{\mathbf{x}}$. This is accomplished via the projection map $\phi : \mathbb{R}^D \to \mathbb{R}^N$ where

$$\mathbf{x} \mapsto \tilde{\mathbf{x}} = \phi(\mathbf{x}) \tag{6.41}$$

The result is that our linear model updates to become

$$f(\mathbf{x}) := \phi(\mathbf{x})^T \mathbf{w} \tag{6.42}$$

where the weight vector has gone from D dimensional to N dimensional.

Similarly, the normal equations for $\mathbf{w}$ update to become

$$\mathbf{w} = (\Phi\Phi^T)^{-1}\Phi\mathbf{y} \tag{6.43}$$

where $\Phi = \phi(X)$ is the $N \times n$ matrix resulting from applying ϕ columnwise to X.

The following example shows how to use such a mapping to produce a quadratic polynomial fit.

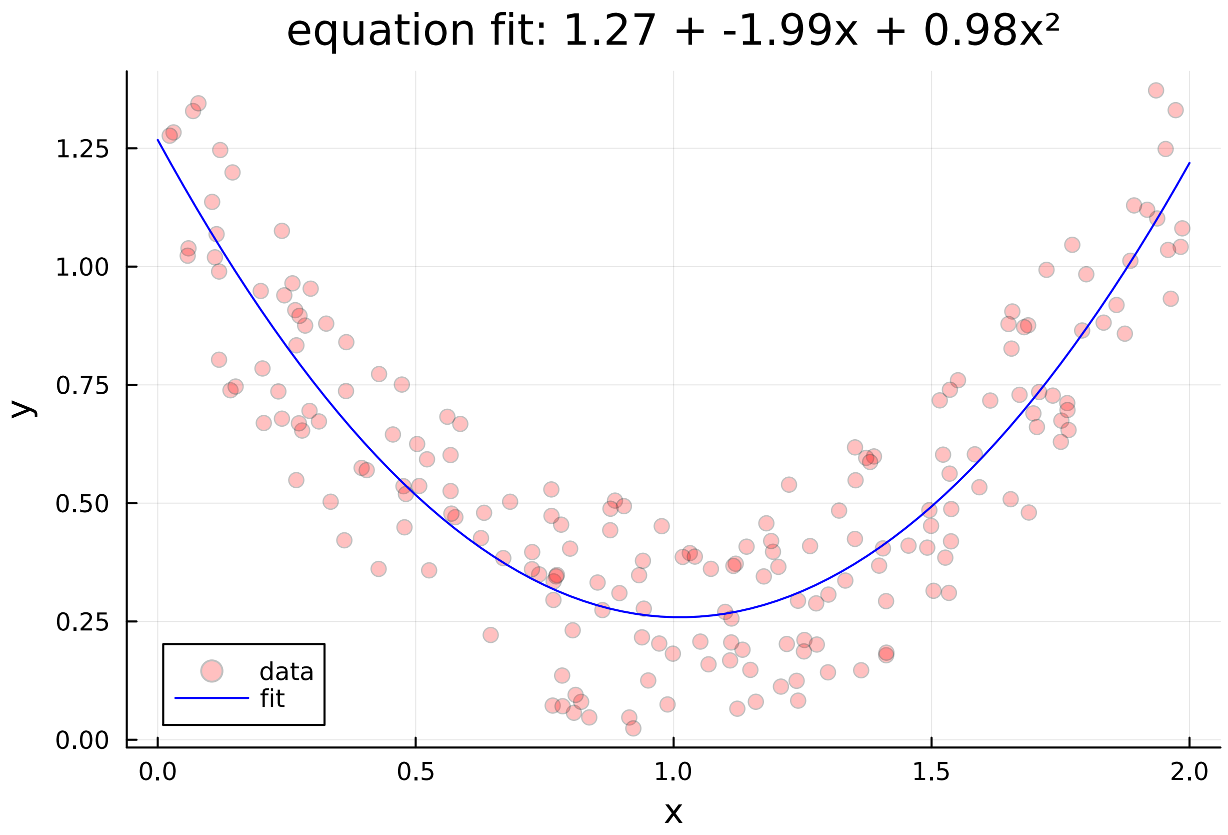


Figure 6.2: Polynomial Regression

We see from the above that our linear regression model found a great fit for a 2nd order polynomial when supplied with polynomial features.

! Important

There is a *massive* problem with this method. Our order 2 polynomial map ϕ takes us from a D dimenional feature vector to $(D+1)!$ many. This means that as we add more features to our feature map, the dimension of the resulting vector will quickly become prohibitively large.

6.4.4.1 Bayesian Regression with Feature Mappings

Let's update our Bayesian regression scheme to reflect the use of our feature projection map ϕ. First we define

$$\Phi := \phi(X) \tag{6.44}$$

$$\phi_* := \phi(\mathbf{x}_*) \tag{6.45}$$

Our predictive distribution therefore becomes

$$p(y_*|\mathbf{x}_*, X, \mathbf{y}) = \mathcal{N}\left(\frac{1}{\sigma_n^2}\phi_*^T A^{-1}\Phi\mathbf{y},\ \phi_*^T A^{-1}\phi_*\right) \tag{6.46}$$

$$A = \frac{1}{\sigma_n^2}\Phi\Phi^T + \Sigma_p^{-1} \tag{6.47}$$

Great! Now we can do our Bayesian inference with non-linear features given by ϕ.

Returning to the problem of the rapidly-growing dimensionality of our augmented feature vectors $\phi(\mathbf{x})$, we see that the computational bottleneck is the matrix inversion of A which requires we invert an $N \times N$ matrix. Our prediction (i.e. the mean) involves multiplication on the right by the n dimensional vector $\mathbf{y}$. With that in mind, perhaps we can reformulate the above into an equivalent form using at most an $n \times n$ dimensional matrix...

Let $K := \Phi^T\Sigma_p\Phi$. Observe the following:

$$\begin{aligned}
\frac{1}{\sigma_n^2}\Phi(K + \sigma_n^2 I) &= \frac{1}{\sigma_n^2}\Phi\left(\Phi^T\Sigma_p\Phi + \sigma_n^2 I\right) && (6.48)\\
&= \frac{1}{\sigma_n^2}\Phi\Phi^T\Sigma_p\Phi + \Phi I && (6.49)\\
&= \left(\frac{1}{\sigma_n^2}\Phi\Phi^T\right)\Sigma_p\Phi + \left(\Phi I\Phi^{-1}\Sigma_p^{-1}\right)\Sigma_p\Phi && (6.50)\\
&= \left(\frac{1}{\sigma_n^2}\Phi\Phi^T + \Sigma_p^{-1}\right)\Sigma_p\Phi && (6.51)\\
&= A\Sigma_p\Phi && (6.52)
\end{aligned}$$

From there we see that

$$A^{-1}\frac{1}{\sigma_n^2}\Phi\left(K+\sigma_n^2 I\right)=\Sigma_p\Phi \tag{6.53}$$

$$\Rightarrow \frac{1}{\sigma_n^2}A^{-1}\Phi=\Sigma_p\Phi\left(K+\sigma_n^2 I\right)^{-1} \tag{6.54}$$

$$\Rightarrow \frac{1}{\sigma_n^2}\phi_*^T A^{-1}\Phi=\phi_*^T\Sigma_p\Phi\left(K+\sigma_n^2 I\right)^{-1} \tag{6.55}$$

For the covariance, we utilize the matrix inversion lemma which states

$$(Z+UWV^T)^{-1}=Z^{-1}-Z^{-1}U(W^{-1}+V^TZ^{-1}U)^{-1}V^TZ^{-1} \tag{6.56}$$

With the identification

$$Z^{-1}\to\Sigma_p \tag{6.57}$$

$$W^{-1}\to\sigma_n^2 I \tag{6.58}$$

$$V\to\Phi \tag{6.59}$$

$$U\to\Phi \tag{6.60}$$

we find

$$\Sigma_p-\Sigma_p\Phi\left(\Sigma_p+\Phi^T\Sigma_p\Phi\right)^{-1}\Phi^T\Sigma_p=\left(\Sigma_p^{-1}+\Phi\frac{1}{\sigma_n^2}I\Phi^T\right)^{-1} \tag{6.61}$$

$$=\left(\frac{1}{\sigma_n^2}\Phi\Phi^T+\Sigma_p^{-1}\right)^{-1} \tag{6.62}$$

$$=A^{-1} \tag{6.63}$$

Thus, we have the equivalent form for our predictive distribution:

$$\boxed{p(y_*|\mathbf{x}_*,X,\mathbf{y})=\mathcal{N}\left(\phi_*^T\Sigma_p\Phi(K+\sigma_n^2 I)^{-1}\mathbf{y},\ \phi_*^T\Sigma_p\phi_*-\phi_*^T\Sigma_p\Phi(K+\sigma_n^2 I)^{-1}\Phi^T\Sigma_p\phi_*\right)} \tag{6.64}$$

where the pesky $N\times N$ term has been replaced by the $n\times n$ matrix $\Phi^T\Sigma_p\Phi$.

6.4.4.2 Kernelization

We now make the the *key* observation that the only matrices that appear in the above expression are

$$\Phi^T \Sigma_p \Phi, \qquad \phi_*^T \Sigma_p \phi_* \tag{6.65}$$

$$\phi_*^T \Sigma_p \Phi, \qquad \Phi^T \Sigma_p \phi_* \tag{6.66}$$

whose matrix elements we can write abstractly as

$$\phi(\mathbf{x})^T \Sigma_p \phi(\mathbf{x}') \tag{6.67}$$

To fit our model, we must determine appropriate values for the symmetric, positive semi-definite covariance matrix Σ_p (and σ_n too, technically). Instead, we observe that this matrix product is a quadratic form which we can think of as representing an inner product on our transformed vectors:

$$K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle \tag{6.68}$$

We call the function $k(\mathbf{x}, \mathbf{x}')$ the **kernel function** or the *covariance function*.

All we need to perform the above calculations are the matrix elements of K on our data $\mathcal{D}$ and any test points $\mathbf{x}_*$ we wish to apply our model to. In effect, this means we are free to use feature vectors **of any dimension, including** ∞.[2]

There are many choices for the kernel function. One of the most popular is the RBF (radial basis function) kernel, also known as the *squared exponential kernel*:

$$k_{\text{rbf}}(\mathbf{x}, \mathbf{x}') := \sigma_f^2 \exp(-\frac{1}{2\ell^2} |\mathbf{x} - \mathbf{x}'|^2) \tag{6.69}$$

[2]The idea here is that ther kernel vunction represents an inner product over *some* vector space. As it turns out, the RBF kernel corresponds to a an infinite dimensional feature vector.

where σ_f^2 is the *signal variance* and ℓ denotes the similarity length scale.

For notational convenience, let's define

$$K := k(X, X) \tag{6.70}$$

$$K_{**} := k(X_*, X_*) \tag{6.71}$$

$$K_* := k(X, X_*) \tag{6.72}$$

then, our predictive distribution takes the final, *clean* form

$$\boxed{p(\mathbf{y}_*|X_*, X, \mathbf{y}) = \mathcal{N}\left(K_*^T(K + \sigma_n^2 I)^{-1}\mathbf{y},\ K_{**} - K_*^T(K + \sigma_n^2 I)^{-1}K_*\right)} \tag{6.73}$$

This is the *end-result* of Gaussian Process Regression acheived via the *weight-space view*.

6.4.5 The Function-space View

So far our approach has been to generalize the standard linear regression model to allow for fitting over a (possibly infinite) basis of features *with* consideration for measurement and model uncertainty (our Bayesian priors). In essence, the idea was to fit *the distribution of all possible weights conditioned on the available training data*, $p(\mathbf{w}|X, \mathbf{y})$. A second *equivalent* approach is to instead consider the distribution of all possible model function $f(\mathbf{x})$. By constructing a Bayesian prior of this space, we constrain the space of possible model functions and fit a *distribution over all allowed model functions*, $p(f|X, \mathbf{y})$. To do so we will need to develop the abstract machinery of distributions over function spaces. When these distributions are Gaussian in nature, the result is called a *Gaussian process*.

6.4.5.1 Gaussian Processes

By this point, we are all familiar with the Gaussian distribution, aka the Normal distribtion $\mathcal{N}(\mu, \sigma^2)$. This distribution is defined by a mean value μ and a variance σ^2. It's *big brother* is the **Multivariate Normal Distribution**, $\mathcal{N}(\ ,\Sigma)$, described be a vector of means   and a covariance matrix Σ. A natural question, then, is can we generalize the concept of the Gaussian distribution from N dimensions to being defined over a continuous field? This question leads naturally to the definition of a **Gaussian Process**

Definition: A *Gaussian Process*, $\mathcal{GP}$, is a collection of random variables for which any finite subset are described by a joint Gaussian distribution.

To see where this comes from, recall that in our previous derivation, we already made the assumption that all our our data points $\mathcal{D}$ are i.i.d. Gaussian distributed. A gaussian process is the natural extension of this and makes the assumption that the continuous set from which are data are sampled are **so Guassian** that any finite sample will be jointly Gaussian distributed. The term *process* is used to distinguish between finite collections of random variables (distributions) and their continuous counterparts described here.

Because each finite subset of this continuous collection is jointly gaussian, we can completely specify a Gaussian Process with two functions: the mean function $m(x)$ and the covariance function $k(\mathbf{x}, \mathbf{x}')$. To denote this, we typically write

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')) \tag{6.74}$$

6.4.5.2 Bayesian Regression is a Gaussian Process

To see this in action, recall our Bayesian regression model

$$f(\mathbf{x} = \phi(\mathbf{x})^T \mathbf{w} \qquad \mathbf{w} \sim \mathcal{N}(\mathbf{0}, \Sigma_p) \tag{6.75}$$

where we have set the prior on w to have zero mean.

The mean function is given by the expectation value of our model:

$$\mathbb{E}[f(\mathbf{x})] = \phi(\mathbf{x})^T \mathbb{E}[\mathbf{w}] = 0 \tag{6.76}$$

and the covariance function is given by

$$\mathbb{E}[f(\mathbf{x})f(\mathbf{x}')] = \phi(\mathbf{x})^T \mathbb{E}[\mathbf{w}\mathbf{w}^T]\phi(\mathbf{x}') = \phi(\mathbf{x})^T \Sigma_p \phi(\mathbf{x}') \tag{6.77}$$

6.4.5.3 Prediction with Noise-free Observations

To repeat the point, the key feature of Gaussian processes is that finite subsets are jointly Gaussian distributed. Thus we can we can split our data into the testpoints $\mathcal{D} = (X, \mathbf{y})$ and testpoints X_* t and treat each collection as joint distributions with the following priors:

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X,X) & K(X,X_*) \\ K(X_*,X) & K(X_*,X_*) \end{bmatrix}\right) \tag{6.78}$$

where $\mathbf{f} := f(X)$ and $\mathbf{f}_* = f(X_*)$.

6.4.5.4 Conditioning the Joint Distribution

To obtain our predictive distribution, $p(\mathbf{f}_*|X_*, X, \mathbf{y})$, we *condition the joint prior distribution* on the observations. To see how this works, consider a general joint gaussian distribution given by

$$\begin{bmatrix} x \\ y \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu_x \\ \mu_y \end{bmatrix}, \begin{bmatrix} \Sigma_{xx} & \Sigma_{xy} \\ \Sigma_{yx} & \Sigma_{yy} \end{bmatrix}\right) \tag{6.79}$$

define the centered values $\tilde{x} := x - \mu_x$ and $\tilde{y} := x - \mu_y$. Define the intermediate variable

$$z := \tilde{x} - A\tilde{y} \tag{6.80}$$

Note that since we've subtracted out the mean we have $\mathbb{E}[\tilde{x}] = \mathbb{E}[\tilde{y}] = \mathbb{E}[z] = 0$

Let's now find A...

$$\mathbb{E}[z\tilde{y}^T] = \mathbb{E}[(\tilde{x} - A\tilde{y})\tilde{y}^T] \tag{6.81}$$

$$= \mathbb{E}[\tilde{x}\tilde{y}^T - A\tilde{y}\tilde{y}] \tag{6.82}$$

$$= \mathbb{E}[\tilde{x}\tilde{y}^T] - \mathbb{E}[A\tilde{y}\tilde{y}^T] \tag{6.83}$$

$$= \Sigma_{xy} - A\mathbb{E}[\tilde{y}\tilde{y}^T] \tag{6.84}$$

$$= \Sigma_{xy} - A\Sigma_{yy} \tag{6.85}$$

Therefore if we choose A so that z and $\tilde{y}$ are independent and uncorrelated, then $\Sigma_{zy} = \mathbb{E}[z\tilde{y}^T] = 0$. Using this assumption, we find

$$0 = \mathbb{E}[z\tilde{y}^T] = \Sigma_{xy} - A\Sigma_{yy} \Rightarrow \boxed{A = \Sigma_{xy}\Sigma_{yy}^{-1}} \tag{6.86}$$

If we now condition $\tilde{x}$ on $\tilde{y}$ (i.e. look at $\tilde{x}$ when $\tilde{y}$ is constant), we find

$$\mathbb{E}[\tilde{x}|\tilde{y}] = A\tilde{y} + \mathbb{E}[z] \tag{6.87}$$

$$= A\tilde{y} + 0 \tag{6.88}$$

$$= \Sigma_{xy}\Sigma_{yy}^{-1} \tag{6.89}$$

$$\tag{6.90}$$

By manipulating this expression, we can now derive $\mathbb{E}[x|y]$ as follows:

$$\mathbb{E}[x|\tilde{y}] = \mathbb{E}[\tilde{x}|\tilde{y}] + \mu_x \tag{6.91}$$

$$= \mu_x + \Sigma_{xy}\Sigma_{yy}^{-1}\tilde{y} \tag{6.92}$$

$$\tag{6.93}$$

$$\boxed{\mathbb{E}[x|y] = \mu_x + \Sigma_{xy}\Sigma_{yy}^{-1}(y - \mu_y)} \tag{6.94}$$

Similarly for the covariance, we have

$$\begin{aligned}
\mathrm{Cov}(x|y) &= \mathrm{Cov}(\tilde{x} + \mu_x|\tilde{y}) && (6.95)\\
&= \mathrm{Cov}(\tilde{x} + \mu_x|\tilde{y} + \mu_y) && (6.96)\\
&= \mathrm{Cov}(\tilde{x}|(\tilde{y} + \mu_y)) && (6.97)\\
&= \mathrm{Cov}(\tilde{x}|\tilde{y}) && (6.98)\\
&= \mathrm{Cov}((z + A\tilde{y})|\tilde{y}) && (6.99)\\
&= \mathrm{Cov}(z) + A\mathrm{Cov}(\tilde{y}) && (6.100)\\
&= \mathrm{Cov}(z) + 0 && (6.101)\\
&= \mathbb{E}[zz^T] && (6.102)\\
&= \mathbb{E}[(\tilde{x} - A\tilde{y})(\tilde{x} - A\tilde{y})^T] && (6.103)\\
&= \mathbb{E}[\tilde{x}\tilde{x}^T - A\tilde{y}\tilde{x}^T - x(A\tilde{y})^T + A\tilde{y}\tilde{y}^T A^T] && (6.104)\\
&= \Sigma_{xx} - A\Sigma_{yx} - \Sigma_{xy}A^T + A\Sigma_{yy}A^T && (6.105)\\
&= \Sigma_{xx} - (\Sigma_{xy}\Sigma_{yy}^{-1})\Sigma_{yx} - \Sigma_{xy}(\Sigma_{yy}^{-1})^T\Sigma_{xy}^T + \Sigma_{xy}\Sigma_{y}^{-1}\Sigma_{y}(\Sigma_{y}^{-1})^T\Sigma_{xy}^T && (6.106)\\
&= \Sigma_{xx} - \Sigma_{xy}\Sigma yy^{-1}\Sigma_{xy}^T - \Sigma_{xy}(\Sigma_{yy}^{-1})^T\Sigma_{xy}^T + \Sigma_{xy}(\Sigma_{yy}^{-1})^T\Sigma_{xy}^T && (6.107)\\
&= \Sigma_{xx} - \Sigma_{xy}\left[\Sigma_{yy}^{-1} - (\Sigma_{yy}^{-1})^T + (\Sigma_{yy}^{-1})^T\right]\Sigma_{xy}^T && (6.108)\\
&= \Sigma_{xx} - \Sigma_{xy}\Sigma_{yy}^{-1}\Sigma_{yx} && (6.109)
\end{aligned}$$

$$\boxed{\mathrm{Cov}(x|y) = \Sigma_{xx} - \Sigma_{xy}\Sigma_{yy}^{-1}\Sigma_{yx}} \quad (6.110)$$

armed with this identity for joint Guassian distributions, we are ready to derive the predictive distribution for Gaussian Process Regression

6.4.5.5 Prediction with Gaussian Processes

Applying these results for our gaussian process, we find

$$p(\mathbf{f}_*|X_*, X, \mathbf{y} = \mathcal{N}\left(K_*^T K^{-1}\mathbf{f},\ K_{**} - K_*^T K^{-1} K_*\right) \tag{6.111}$$

To account for noisy observations, we can augment our correlation function to include a noise offset. The joint distrubtion then becomes:

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X,X) - \sigma_n^2 I & K(X,X_*) \\ K(X_*,X) & K(X_*,X_*) \end{bmatrix}\right) \tag{6.112}$$

which leads to the predictive distribution

$$\boxed{p(\mathbf{f}_*|X_*, X, \mathbf{y}) = \mathcal{N}\left(K_*^T \left[K + \sigma_n^2 I\right]^{-1}\mathbf{f},\ K_{**} - K_*^T \left[K + \sigma_n^2 I\right]^{-1} K_*\right)} \tag{6.113}$$

6.4.6 Doing it in Julia

`KernelFunctions.jl` provides a clean interface to create various kernelfunctions and apply them to data to create our matrices `K`.

Due to the fact that kernel functions obey composition laws, we can easily build up complicated Kernels from basic pieces via function composition with $\circ$

Unsurprisingly, there is a lot of activation on the diagonal as for a single datapoint $\mathbf{x}$, we have

$$k(\mathbf{x},\mathbf{x}) = \exp\left(-\frac{0}{2\ell^2}\right) = 1.0 \tag{6.114}$$

Now that we have our Kernel function, let's construct our Gaussian Process.

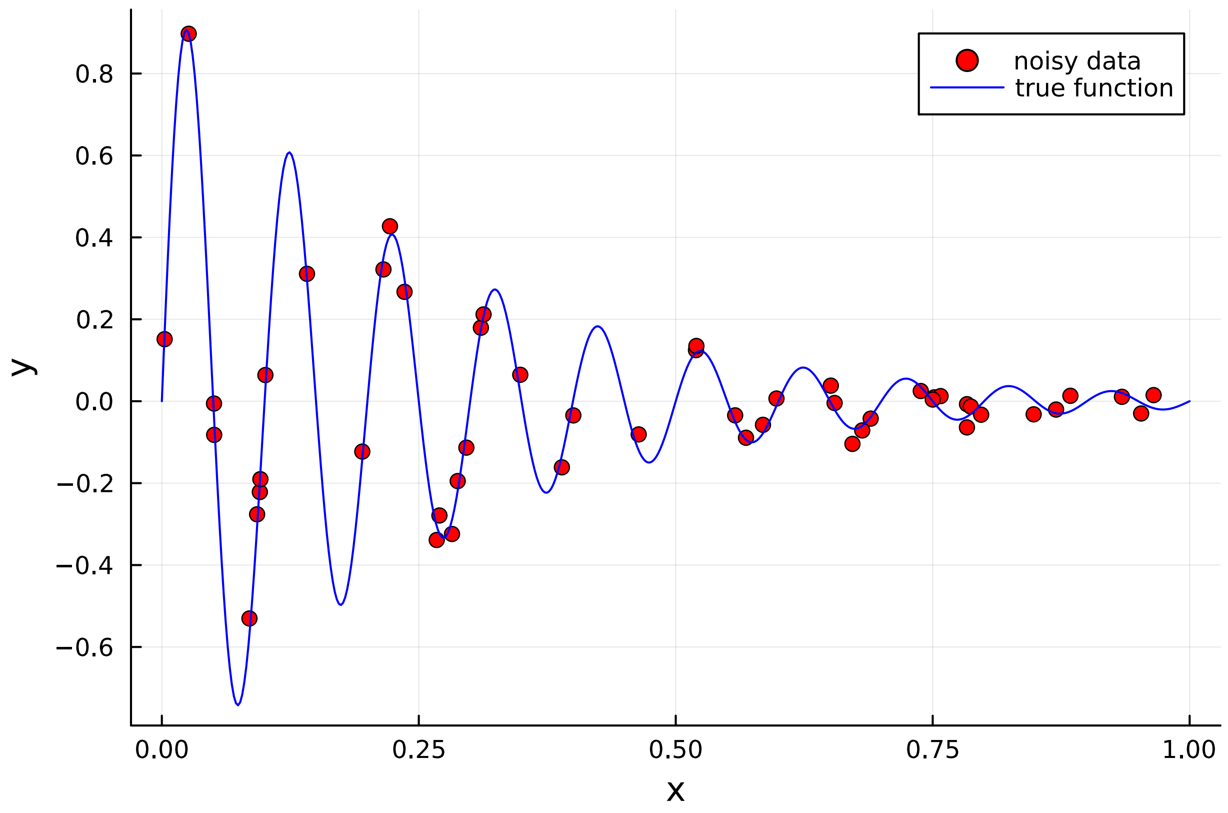


Figure 6.3: Training set

`AbstractGPs.jl` provides an excellent way to define Gaussian Processes by supplying mean and kernel functions. We can then sample from our GPs with a simple interface designed to extend the basic functions from `Statistics.jl`. From an `AbstractGP` we can construct a `FiniteGP` by *indexing* into our datasets.

First we construct $f \sim \mathcal{GP}(0, k(\cdot, \cdot))$

From this `AbstractGP`, we can now construct a *FiniteGP*, i.e. a multivariate normal distribution by applying GP to our training data. We include a measurement variance of $\sigma^2 = 0.1$ to account for noisy observations

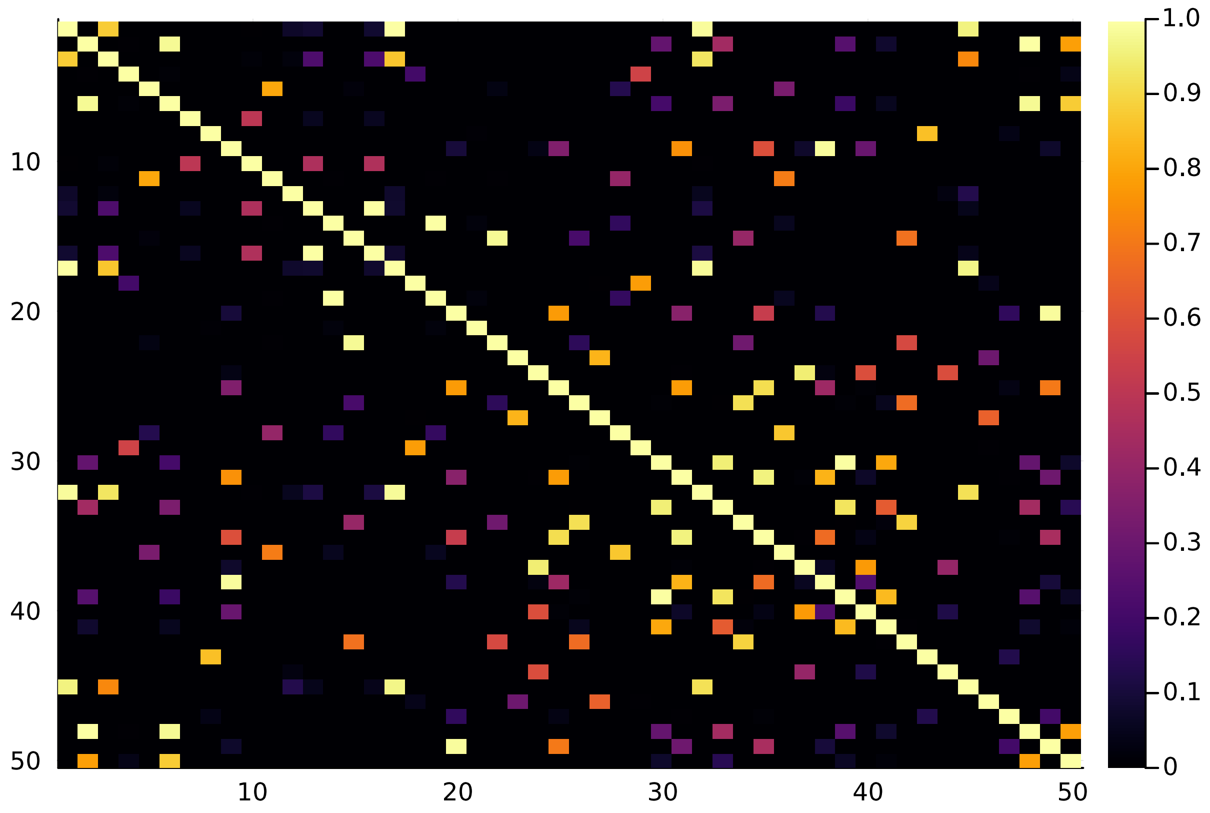


Figure 6.4: Kernel matrix visualization

Now that we have our Gaussian Process, we can compute the log-marginal likelihood $p(\mathbf{y}|X, \theta)$, i.e. the probabity of obtaining the targets given the features, hyperparameters, etc...

Next, we demonstrate how to compute the posterior Gaussian process (for us that would be f_*). First we create the finite gaussian process (a function) which we will use to compute the posterior distribution

$$p(\mathbf{f}_*|X_*, X, \mathbf{y}) \tag{6.115}$$

Now that we have the distribution, we can form our predictions... This can be done a few different ways:

Alternatively, if we instead want a distribution for each datapoint we can compute

$$p(\mathbf{y}_x|\mathbf{x}_*, X, y) \tag{6.116}$$

When treated as a collection, we can think about each of these representing a marginalized distribution over the test points $\mathbf{x}_*$ and hence, we call `marginals()`

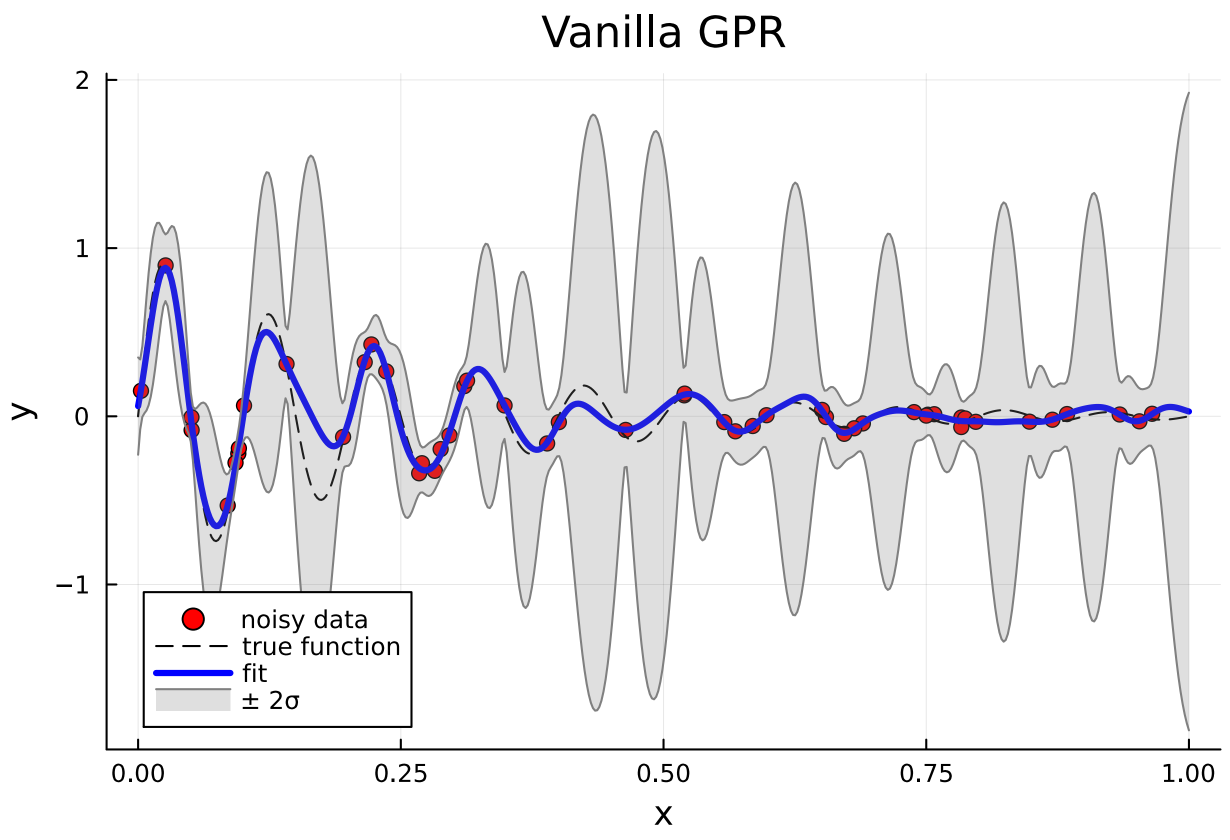


Figure 6.5: Gaussian process fit with vanilla hyperparameters

6.4.6.1 Summary:

So far we have shown how to:

1. Build a kernel function $k(\cdot, \cdot)$ via composition using `KernelFunctions.jl`
2. Construct an a Gaussian Process $f \sim \mathcal{GP}$ abstractly using `AbstractGPs.jl`

3. Construct a finite representation of our GP, f_x, over training data
4. Construct a posterior Gaussian Process from f_x and our training targets $\mathbf{y}$.
5. Construct a finite representation of the posterior GP applied to our prediction data (here `Xtrue`).
6. Sample this final distribution to obatin a prediction via `mean()` and variances via `var()`. Alternatively, we can obtain a multivariate normal distribution for each point by calling `marginals()`.

6.4.7 Fitting the Gaussian Process

You may think we have *already* fit the Guassian process however, we were forced to choose values for both ℓ and σ^2. How can we optimally select the ideal hyperparameters for our Gaussian Process? This leads us into the realm of Bayesian Model Selection

6.4.8 Bayesian Model Selection

There are several levels of parameters in machine learning. At the lowest level, we have the model weights $\mathbf{w}$. Above that, we have model hyperparameters, θ. At the top we have model structure $\mathcal{H}$. In our Bayesian framework, we can consider distributions defined at each of these levels. At the bottom, we have

$$p(\mathbf{w}|X, \mathbf{y}, \theta, \mathcal{H}_i) = \frac{p(\mathbf{y}|X, \mathbf{w}, \theta, \mathcal{H}_i)p(\mathbf{w}|\theta, \mathcal{H}_i)}{p(\mathbf{y}|X, \theta, \mathcal{H}_i)} \tag{6.117}$$

Note

If this looks confusing, consider Bayes rule for 3 events R, H, S. We have:

$$P(R|H,S) = \frac{P(R,H,S)}{P(H,S)} \tag{6.118}$$

$$= \frac{P(H|R,S)P(R,S)}{P(H,S)} \tag{6.119}$$

$$= \frac{P(H|R,S)P(R|S)P(S)}{P(H|S)P(S)} \tag{6.120}$$

$$= \frac{P(H|R,S)P(R|S)}{P(H|S)} \tag{6.121}$$

To get the result, just think of θ and $\mathcal{H}_i$ as a single *event* and translate the above to distribution functions.

stack exchange link

The prior $p(\mathbf{w}|\theta,\mathcal{H}_i)$ encodes any knowledge we have about the parameters prior to seeing the data. The denominator is the *marginal likelihood* and is given by

$$p(\mathbf{y}|X,\theta,\mathcal{H}_i) = \int d\mathbf{w}\; p(\mathbf{y}|X,\mathbf{w},\theta,\mathcal{H}_i)p(\mathbf{w}|\theta,\mathcal{H}_i) \tag{6.122}$$

The next level up is to express the distribution of hyper-parameters θ:

$$p(\theta|X,\mathbf{y},\mathcal{H}_i) = \frac{p(\mathbf{y}|X,\theta,\mathcal{H}_i)p(\theta|\mathcal{H}_i)}{p(\mathbf{y}|X,\mathcal{H}_i)} \tag{6.123}$$

Here $p(\theta|\mathcal{H}_i)$ is called the *hyper-prior*. Similarly, the normalization constant is given by

$$p(\mathbf{y}|X,\mathcal{H}_i) = \int d\theta\; p(\mathbf{y}|X,\theta,\mathcal{H}_i)p(\theta|\mathcal{H}_i) \tag{6.124}$$

Finally, at the top level we have the set of possible model structures $\{\mathcal{H}_i\}$. This leads to

$$p(\mathcal{H}_i|X,\mathbf{y}) = \frac{p(\mathbf{y}|X,\mathcal{H}_i)p(\mathcal{H}_i)}{p(\mathbf{y}|X)} \tag{6.125}$$

with normlization constant

$$p(\mathbf{y}|X) = \sum_i p(\mathbf{y}|X, \mathcal{H}_i)p(\mathcal{H}_i) \tag{6.126}$$

Depending on the model details, these integrals may be intractible to approximations or Monte Carlo methods. Since we rarely have sufficient knowledge to form a hyperparameter prior, one often attempts to maximize the marginal likelihood $p(\mathbf{y}|X, \theta, \mathcal{H}_i)$ with respect to the hyperparameters θ instead. This is known as Type II Maximium Likelihood Estimation.

In the case of Gaussian Process Regression, we are once again saved by the fact that every piece has a convenient functional from resulting in analytically tractible integrals for the marginal likelihood function. We find

$$\ln p(\mathbf{y}|X, \theta) = -\frac{1}{2}\mathbf{y}^T(K_f + \sigma_n^2 I)^{-1}\mathbf{y} - \frac{1}{2}\ln|K_f + \sigma_n^2 I| - \frac{n}{2}\ln(2\pi) \tag{6.127}$$

Note

We should add a derivation of this when possible. It's just a big 'ol nasty integral.

Let's try this out in code! See this section from the `AbstractGPs.jl` docs…

We want to maximize the log-marginal-likelihood and therefore want to minimize minus that quantitty:

Excellent! Now that we have the hang of it, let's try another fit for a function of two variables.

we need a way to intelligently initialize the outputs. See this post for ideas

"

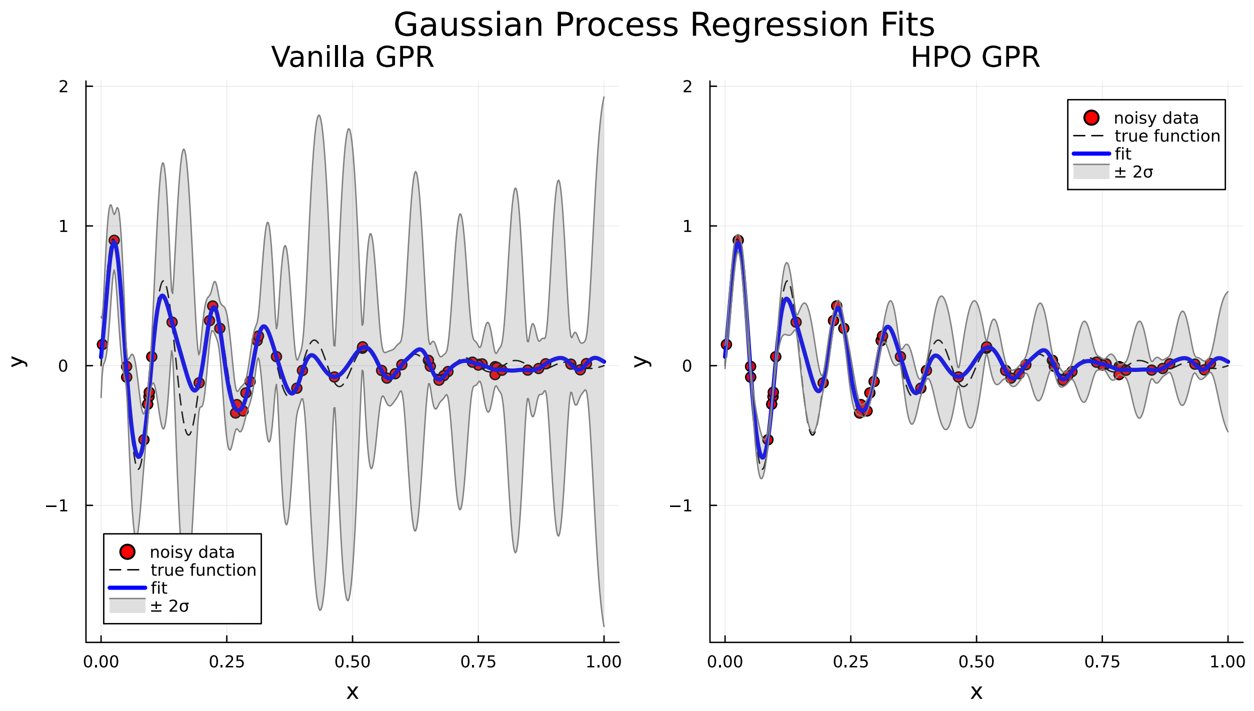


Figure 6.6: Hyperparameter optimized GPR fit

6.5 Model Ensembling

Boosting vs Bagging

XGBoost vs RandomForest (and other implementations)

MLJ and SciKitLearn documentation sites will have good references for this, I think.

6.6 Super Learners

Use the example of model stacking from MLJ documentation to describe our approach.

6.7 Self Organizing Maps

Self organizing maps (SOMs) are an unsupervised machine learning technique developed by Kohonen (see Kohonen 1982) based on the simple biological principle that *neurons near each*

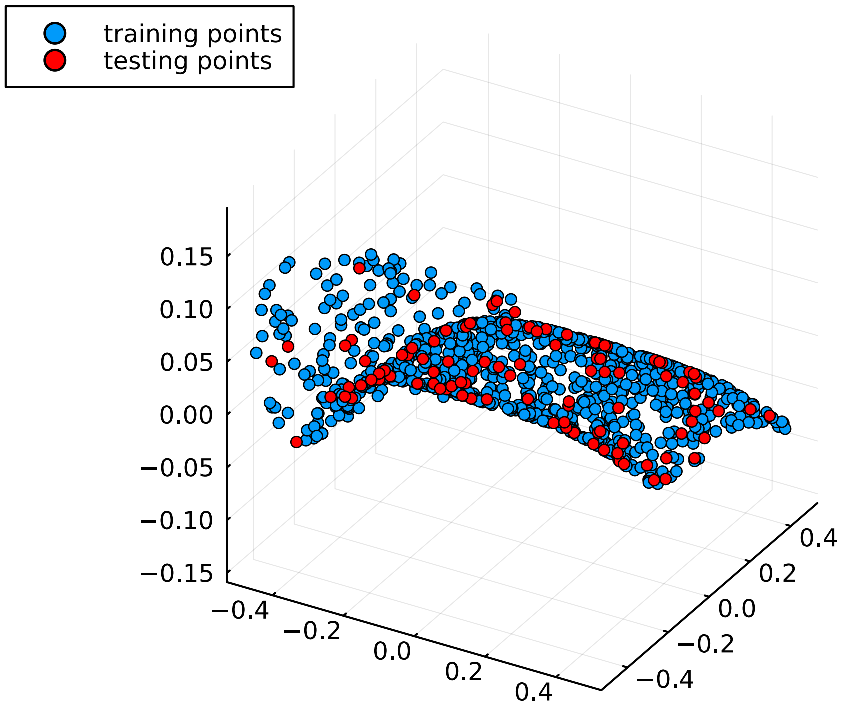


Figure 6.7: Nonlinear function fit example

other fire together. This observation that the toplogical *closeness* of similar computational units is a critical feature of intelligent systems leads to a natural reinterpretation of the familiar perceptron model into a new form amenable for a variety of clustering and dimensionality reduction tasks. In particular, the SOM enables a rapid unsupervised classification of multidimensional data into a (typically) one or two dimensional *simplicial complex*, the discrete realization of a topological manifold, whose vertices correspond to representative points in the original data space $\mathcal{D}$. While a tad esoteric compared to other popular unsupervised methods like KMeans clustering or DBSCAN, the SOM distinguishes itself with the added benefit that it's training procedure guarantees nodes (i.e. classes) which are topologically close in the SOM simplex share similar weights. This additional structure makes the SOM particularly attractive when an interpretation of the discovered clusters *as well as* the relationships between them is desired.

Note

We should add some more context (and references) for how the SOM has been used here. Perhaps we can ask David about his diatom identification from the Saharan dust sources.

The original treatment of the SOM by Kohonen was made in terms of processing units, sensory signals, and relaying networks (Kohonen 1982), however, in the modern era of deep learning, a more palatable derivation can be obtained by re-interpreting the weights of a simple perceptron model to provide the foundation for a clustering approach.

6.7.1 Reinterpreting the Perceptron

As described in Section 6.2, a perceptron is a function of the form

$$\mathbf{y} = \sigma.(W\mathbf{x}) \tag{6.128}$$

where $W \in \mathbb{R}^{n\times m}$ is a matrix of weights which transform the input $\mathbf{x} \in \mathbb{R}^m$ into $\mathbb{R}^n$ and σ is a nonlinear *activation function* applied element-wise to the outputs of the matrix multiplication (indicated by the . syntax). If we instead think of the weight matrix as an ordered collection of vectors $\{\mathbf{w}_i\}_{i=1}^n$, then this formula can be further decomposed into

$$\mathbf{y} = \sum_{i=1}^n \sigma(\mathbf{w}_i^T\mathbf{x}) = \sum_{i=1}^n \sigma(\langle \mathbf{w}_i, \mathbf{x}\rangle) \tag{6.129}$$

The function of the perceptron is now clear: given an input vector $\mathbf{x}$ and a collection of n-many weight vectors $\mathbf{w}_i$, compute the inner product of $\mathbf{x}$ with each weight vector, apply the nonlinear activation function σ, and concatenate the results. If we allow ourselves to imagine the weight vectors $\mathbf{w}_i$ as members of the same space as the inputs $\mathbf{x}$, a reasonable question to ask is: *how similar is the input* $\mathbf{x}$ *to each* $\mathbf{w}_i$. Further, the application of the inner product $\langle \cdot, \cdot \rangle$ suggests we may answer this question in terms of the distance

$$\langle \mathbf{w}_i - \mathbf{x}, \mathbf{w}_i - \mathbf{x} \rangle = d(\mathbf{w}_i, \mathbf{x})^2. \tag{6.130}$$

In other words, given a set of weight vectors $\mathbf{w}_i$ which we may now think of as the clusters for our unsupervised model, we can measure the similarity between a given datum $\mathbf{x}_j$ and each cluster by computing the distance

$$d_{ij} = d\left(\mathbf{w}_i, \mathbf{x}_j\right) \tag{6.131}$$

6.7.2 The Training Process

6.7.3 Common SOM topologies

- Square
- Cylindrical
- Toroidal
- Spherical

6.7.4 A simple example: partitioning color spaces

6.7.5 Drawbacks of the SOM model

6.8 Generative Topographic Maps

6.9 Data Assimilation

NOTE: It would be nice to provide additional derivations (where possible) in a Bayesian framework... We should also liberally cite Dr. Lary's original papers on the chemical 4d-var implementation.

6.9.1 Overview

The proper application of scientific models to make real-world predictions requires that we commit ourselves to a full accounting of all possible sources of uncertainty when reporting results. Further, the explosion of *big data* across scientific fields provieds a plethora observational data that our models are typically unequipped to incorporate when making predictions. The field of **Data Assimilation** addresses this problem by providing a family of techniques engineered to combine model output together with observational data whilst enabling a complete accounting the sources of uncertainty. For chaotic systems in particular, data assimilation enables integration on long time scales that would be impossible via models alone.

In this overview, we will follow the examples from this nice paper.

6.9.2 Framing the Problem

Data assimilation can be understood most generally in terms of dyscrete dynamical systems. This enables us to apply the methods to most mathematical models from gridded PDE solvers to systems of ordinary differential equations. Our goal is to find the best prediction for the system state vector u that combines our model predictions, also known as forecasts, with observational data. Model predictions are summarized via the discrete update equation:

$$u_{k+1} = \mathcal{M}(u_k; \theta) \tag{6.132}$$

For ODE systems, $\mathcal{M}$ represents the time integration scheme for a system of ODEs like

$$\frac{du}{dt} = f(u, t; \theta) \tag{6.133}$$

To measure the performance of our assimilation scheme, we denote the *true* value of the state vector as $u^{(t)}$. The output of our model is denoted $u^{(b)}$ (b subscript for *background*). The discrepancy between the true value and our forecast is denoted $\xi^{(b)} = u^{(t)} - u^{(b)}$ characterizing the extent to which our model prediction is imperfect.

The observations of our system are denoted by $w_k = w(t_k)$. These observations do not necessarily need to be components of the state vector u, but rather, are related to it via the *observation function* h. For example, one may attempt to predict sea surface temperature using data assimilation with data from satellite observations. The function h would then be the Stefan-Boltzmann law. However, real world data is noisy, which we must take into account. We write

$$w_k = h(u_k) + \xi_k^{(m)} \tag{6.134}$$

where $\xi_k^{(m)}$ denotes this measurement noise.

Given our model predictions $u_k^{(b)}$ and observations w_k, we seek to obtain the *optimal* or best-possible prediction called the **analysis**, $u^{(a)}$. This analysis will still not be perfect, so we further specify the analysis error via

$$\xi^{(a)} = u^{(t)} - u^{(a)} \tag{6.135}$$

6.9.3 Summary

$u_k^{(t)} \in \mathbb{R}^n$	the true state vector	(6.136)
$u_k^{(b)} \in \mathbb{R}^n$	the kth model forecast	(6.137)
$u_k^{(a)} \in \mathbb{R}^n$	the analysis	(6.138)
$w_k \in \mathbb{R}^m$	the kth observation vector	(6.139)
$\xi^{(b)} \in \mathbb{R}^n$	the model forecast error	(6.140)
$\xi^{(m)} \in \mathbb{R}^m$	the observation noise vector	(6.141)
$\xi^{(a)} \in \mathbb{R}^n$	the analysis error	(6.142)
$\xi^{(p)} \in \mathbb{R}^n$	the process noise if we used our model on the true state	(6.143)
$\mathcal{M} : \mathbb{R}^n \to \mathbb{R}^n$	the model update function	(6.144)
$f : \mathbb{R}^n \to \mathbb{R}^n$	differential equation model	(6.145)
$h : \mathbb{R}^n \to \mathbb{R}^m$	observation function	(6.146)

6.9.4 Assumptions

To make possible the derivation of a *unique* analysis $u^{(a)}$, the following assumptions are in order.

$$\mathbb{E}[\xi_k^{(b)}] = 0 \qquad \mathbb{E}[\xi_k^{(b)}(\xi_j^{(b)})^T] = 0 \text{ for } k \neq j \tag{6.147}$$

$$\mathbb{E}[\xi_k^{(m)}] = 0 \qquad \mathbb{E}[\xi_k^{(m)}(\xi_j^{(m)})^T] = 0 \text{ for } k \neq j \tag{6.148}$$

$$\mathbb{E}[\xi_k^{(b)}(u_0)^T] = 0 \qquad \mathbb{E}[\xi_k^{(m)}(u_0)^T] = 0 \tag{6.149}$$

$$\mathbb{E}[\xi_k^{(b)}\xi_j^{(m)}] = 0 \tag{6.150}$$

$$\mathbb{E}[u_k^{(t)}] = u_k^{(b)} \tag{6.151}$$

We also define the error covariance matrices

$$Q_k := \mathbb{E}[\xi_k^{(p)}(\xi_k^{(p)})^T] \tag{6.152}$$

$$R_k := \mathbb{E}[\xi_k^{(m)}(\xi_k^{(m)})^T] \tag{6.153}$$

$$B_k := \mathbb{E}[\xi_k^{(b)}(\xi_k^{(b)})^T] \tag{6.154}$$

which we will use in our consideration of the final error of our analysis.

6.9.5 Kalman Filtering

Given some model for the error covariance matrices Q_k and R_k, we would like a method that propagates *both* our model **and** the errors forward. This way we may guarantee that the accuracy of our analysis doesn't come at the cost of higher uncertainty.

The original implementation of the Kalman filter was for strictly linear systems. We will first develop the analysis for this simplified case adn then will generalize to the **Extended Kalman Filter** (EKF) that can handle fully nonlinear situations.

In the linear case, our system may be written as

$$u_{k+1}^{(t)} = M_k u_k^{(t)} + \xi_{k+1}^{(p)} \tag{6.155}$$

$$w_k = H_k u_k^{(t)} + \xi_k^{(m)} \tag{6.156}$$

where M_k and H_k are now matrices defining the linear problem.

The goal of the Kalman filter is to derive the analysis $u^{(a)}$ which optimizes the trace of the analysis error covariance matrix (i.e. sum of squared errors):

$$\mathrm{Tr}\,(P_k) := \mathbb{E}[(u_k^{(t)} - u_k^{(a)})^T (u_k^{(t)} - u_k^{(a)})] \tag{6.157}$$

Finding the analysis consists of two steps: the forecast step and the assimilation step.

6.9.5.1 Forecast Step

Assume we have the analysis at time t_k denoted $u_k^{(a)}$. Then the forecast for time t_{k+1} is

$$u_{k+1}^{(b)} = M_k u_k^{(a)} \tag{6.158}$$

The background error is therefore

$$\begin{aligned}
\xi_{k+1}^{(b)} &= u_{k+1}^{(t)} - u_{k+1}^{(b)} &(6.159)\\
&= M_k u_k^{(t)} + \xi_{k+1}^{(p)} - M_k u_k^{(a)} &(6.160)\\
&= M_k \left(u_k^{(t)} - u_k^{(a)}\right) + \xi_{k+1}^{(p)} &(6.161)\\
&= M_k \xi_k^{(a)} + \xi_{k+1}^{(p)} &(6.162)
\end{aligned}$$

We may now evaluate the covariance matrix of our background estimate as:

$$B_{k+1} = \mathbb{E}[\xi^{(b)}_{k+1}(\xi^{(b)}_{k+1})^T] \tag{6.163}$$
$$= \mathbb{E}\left[\left(M_k\xi^{(a)}_k + \xi^p_{k+1}\right)\left(M_k\xi^{(a)}_k + \xi^p_{k+1}\right)^T\right] \tag{6.164}$$
$$\tag{6.165}$$

If we presume that $\mathbb{E}[\xi^{(b)}_k(\xi^{(p)}_{k+1})^T] = 0$, then the cross terms vanish and we are left with

$$\boxed{B_{k+1} = M_kP_kM_k^T + Q_{k+1}} \tag{6.166}$$

Thus we now have the background (i.e forecast) estimate of the state at t_{k+1} and its covariance matrix. Given a measurement w_{k+1} at the same time with covariance matrix R_{k+1}, then we may now perform the assimilation step where we fuse the two sources of information to obtain $u^{(a)}_{k+1}$ and P_{k+1}.

6.9.5.2 Data Assimilation Step

Let's suppose that the analysis has the form

$$u^{(a)}_{k+1} = \nu + K_{k+1}w_{k+1} \tag{6.167}$$

for some vector $\nu \in \mathbb{R}^n$ and matrix $K_{k+1} \in \mathbb{R}^{m\times n}$. In a perfect world, we would have $\mathbb{E}[u^{(t)}_k - u^{(a)}_k] = 0$. Therefore,

$$0 = \mathbb{E}[u_k^{(t)} - u_k^{(a)}] \tag{6.168}$$

$$= \mathbb{E}[(u_k^{(b)} + \xi_k^{(b)}) - (\nu + K_k w_k)] \tag{6.169}$$

$$= \mathbb{E}[(u_k^{(b)} + \xi_k^{(b)}) - (\nu + K_k H_k u_k^{(t)} + K_k \xi_k^{(m)})] \tag{6.170}$$

$$= \mathbb{E}[u_k^{(b)}] + \mathbb{E}[\xi_k^{(b)}] - \mathbb{E}[\nu] - K_k H_k \mathbb{E}[u_k^{(t)}] - K_k \mathbb{E}[\xi_k^{(m)}] \tag{6.171}$$

$$= u_k^{(b)} + 0 - \nu - K_k H_k u_k^{(b)} - 0 \tag{6.172}$$

$$= u_k^{(b)} - \nu - K_k H_k u_k^{(b)} \tag{6.173}$$

$$\Rightarrow \nu = u_k^{(b)} - K_k H_k u_k^{(b)} \tag{6.174}$$

which we now substitute to obtain

$$\boxed{u_k^{(a)} = u_k^{(b)} + K_k(w_k - H_k u_k^{(b)})} \tag{6.175}$$

Now that we know the form for the analysis we may derive the optimal matrix K_k by optimization of P_k. We have

$$\xi_k^{(a)} = u_k^{(t)} - u_k^{(a)} \tag{6.176}$$

$$= M_{k-1}u_{k-1}^{(t)} + \xi_k^{(p)} - u_k^{(b)} - K_k\left(w_k - H_k u_k^{(b)}\right) \tag{6.177}$$

$$= M_{k-1}u_{k-1}^{(t)} + \xi_k^{(p)} - M_{k-1}u_{k-1}^{(a)} - K_k\left(H_k u_k^{(t)} + \xi_k^{(m)} - H_k u_k^{(b)}\right) \tag{6.178}$$

$$= M_{k-1}u_{k-1}^{(t)} + \xi_k^{(p)} - M_{k-1}u_{k-1}^{(a)} - K_k H_k u_k^{(t)} - K_k\xi_k^{(m)} + K_k H_k u_k^{(b)} \tag{6.179}$$

$$= M_{k-1}u_{k-1}^{(t)} + \xi_k^{(p)} - M_{k-1}u_{k-1}^{(a)} - K_k H_k u_k^{(t)} - K_k\xi_k^{(m)} + K_k H_k u_k^{(b)} \tag{6.180}$$

$$= \left\{M_{k-1}(\xi_{k-1}^{(a)} + u_{k-1}^{(a)}) + \xi_k^{(p)} - M_{k-1}u_{k-1}^{(a)}\right\} - K_k H_k u_k^{(t)} - K_k\xi_k^{(m)} + K_k H_k u_k^{(b)} \tag{6.181}$$

$$= \left\{M_{k-1}\xi_{k-1}^{(a)} + \xi_k^{(p)}\right\} - K_k H_k u_k^{(t)} + K_k H_k u_k^{(b)} - K_k\xi_k^{(m)} \tag{6.182}$$

$$= M_{k-1}\xi_{k-1}^{(a)} + \xi_k^{(p)} - K_k H_k(M_{k-1}u_{k-1}^{(t)} + \xi_k^{(b)}) + K_k H_k u_k^{(b)} - K_k\xi_k^{(m)} \tag{6.183}$$

$$= M_{k-1}\xi_{k-1}^{(a)} + \xi_k^{(p)} - K_k H_k M_{k-1}(\xi_{k-1}^{(a)} + u_{k-1}^{a}) - K_k H_k\xi_k^{(b)} + K_k H_k u_k^{(b)} - K_k\xi_k^{(m)} \tag{6.184}$$

$$= M_{k-1}\xi_{k-1}^{(a)} + \xi_k^{(p)} - K_k H_k M_{k-1}(\xi_{k-1}^{(a)} + u_{k-1}^{a}) - K_k H_k\xi_k^{(b)} + K_k H_k M_{k-1}u_{k-1}^{(a)} - K_k\xi_k^{(m)} \tag{6.185}$$

$$= M_{k-1}\xi_{k-1}^{(a)} + \xi_k^{(p)} - K_k H_k M_{k-1}\xi_{k-1}^{(a)} - K_k H_k\xi_k^{(b)} - K_k\xi_k^{(m)} \tag{6.186}$$

$$= (I - K_k H_k)(M_{k-1}\xi_{k-1}^{(a)} - \xi_k^{p}) - K_k\xi_k^{(m)} \tag{6.187}$$

$$\tag{6.188}$$

and therefore the covariance matrix is

$$P_k = \mathbb{E}[\xi_k^{(a)}(\xi_k^{(a)})^T] \tag{6.189}$$

$$= \mathbb{E}\Big[\Big((I - K_k H_k)(M_{k-1}\xi_{k-1}^{(a)} - \xi_k^p) - K_k \xi_k^{(m)}\Big) \Big((I - K_k H_k)(M_{k-1}\xi_{k-1}^{(a)} - \xi_k^p) - K_k \xi_k^{(m)}\Big)^T \Big] \tag{6.190}$$

$$= (I - K_k H_k) M_{k-1} \mathbb{E}[\xi_{k-1}^{(a)}(\xi_{k-1}^{(a)})^T] M_{k-1}^T (I - K_k H_k)^T + (I - K_k H_k)\mathbb{E}[\xi_k^{(p)}(\xi_k^{(p)})^T](I - K_k H_k)^T \tag{6.191}$$

$$- K_k \mathbb{E}[\xi_k^{(m)}(\xi_k^{(m)})^T] K_k^T \tag{6.192}$$

$$= (I - K_k H_k) B_k (I - K_k H_k)^T - K_k R_k K_k^T \tag{6.193}$$

6.9.5.3 Deriving K_k

The Kalman filter is defined at that K_k which which minimizes the sum of squared analysis errors, i.e. the trace of the analysis error covariance matrix. The following identies will be useful:

$$\underset{A}{\nabla} \operatorname{tr}(AB) = B^T \tag{6.194}$$

$$\underset{A}{\nabla} \operatorname{tr}(BA^T) = B \tag{6.195}$$

$$\underset{A}{\nabla} \operatorname{tr}(ABA^T) = AB^T + AB \tag{6.196}$$

$$\tag{6.197}$$

from which we obtain

$$
\begin{aligned}
0 &= \underset{K_k}{\nabla} \operatorname{tr}(P_k) && (6.198)\\
&= \underset{K_k}{\nabla} \left\{ B_k - B_k H_k^T K_k^T - K_k H_k B_k + K_k H_k B_k H_k^T B_k^T - K_k R_k K_k^T \right\} && (6.199)\\
&= -B_k H_k^T - (H_k B_k)^T + K_k \left[H_k B_k H_k^T + (H_k B_k H_k^T)^T - R_k + R_k^T \right] && (6.200)\\
&= -2 B_k H_k^T + 2 K_k \left(H_k B_k H_k^2 - R_k \right) && (6.201)\\
\Rightarrow K_k &= B_k H_k^T \Big[H_k B_k H_k^T - R_k \Big]^{-1} && (6.202)
\end{aligned}
$$

we now substitute this result to obtain a simplified form for P_k.

$$
\begin{aligned}
P_k &= (I - K_k H_k) B_k (I - K_k H_k)^T + K_k R_k K_k^T && (6.203)\\
&= (I - K_k H_k) B_k - (I - K_k H_k) B_k (K_k H_k)^T + K_k R_k K_k^T && (6.204)\\
&= (I - K_k H_k) B_k - \Big\{ (I - K_k H_k) B_k (K_k H_k)^T + K_k R_k K_k^T \Big\} && (6.205)\\
&= (I - K_k H_k) B_k - \left\{ (I - K_k H_k) B_k H_k^T K_k^T + K_k R_k K_k^T \right\} && (6.206)\\
&= (I - K_k H_k) B_k - \left\{ (I - K_k H_k) B_k H_k^T + K_k R_k \right\} K_k^T && (6.207)\\
&= (I - K_k H_k) B_k - \left\{ B_k H_k^T - K_k \left(H_k B_k H_k^T + R_k \right) \right\} K_k^T && (6.208)\\
&= (I - K_k H_k) B_k - \left\{ B_k H_k^T - B_k H_k^T \right\} K_k^T && (6.209)\\
&= (I - K_k H_k) B_k && (6.210)
\end{aligned}
$$

NOTE: we have used the fact that covariance matrices are symmetric.

6.9.5.4 Summary

Let's summarize the whole process. We have

1. Initialization We must set the system to some initial condition. This means we must define u_0^a and P_0. We must also come up with a model for the process noise covariance Q_k and measurement error covariance R_k.

2. Forecast Step

$$u_k^{(b)} = M_{k-1} u_{k-1}^{(a)} \tag{6.211}$$

$$B_k = M_{k-1} P_{k-1} M_{k-1}^T + Q_k \tag{6.212}$$

3. Assimilation Step

$$K_k = B_k H_k^T \left[H_k B_k H_k^T - R_k \right]^{-1} \tag{6.213}$$

$$u_k^{(a)} = u_k^{(b)} + K_k (w_k - H_k u_k^{(b)}) \tag{6.214}$$

$$P_k = (I - K_k H_k) B_k \tag{6.215}$$

6.9.6 Extended Kalman Filter

Given the nonlinear nature of many scientific models it is desirable to extend the **Kalman Filter** to be able to handle nonlinear models $f(\cdot)$ (and by extension, their update function $\mathcal{M}(\cdot)$), and nonlinear observation functions $h(\cdot)$. This can be accomplished so long as these functions are sufficiently smooth (C^1 to be precise) so as to admit valid Taylor approximations to first order. That is,

$$\mathcal{M}(u_k) \approx \mathcal{M}(u_k^{(a)}) + D_M(u_k^{(a)}) \xi_k^{(a)} \qquad h(u_k) \approx h(u_k^{(b)}) + D_h(u_k^{(b)}) \xi_k^{(b)} \tag{6.216}$$

$$D_M := \left[\frac{\partial \mathcal{M}_i}{\partial u_j} \right] \qquad D_h := \left[\frac{\partial h_i}{\partial u_j} \right] \tag{6.217}$$

where $\mathcal{M}_i$ and h_i denote the ith component functions of $\mathcal{M}$ and h.

Using these substitutions for the previously linear functions M_k and H_k, we may follow the same derivation to obtain the following procedure.

1. Initialization To begin we must choose values for $u_0^{(a)}$ and P_0. We must also provide models for Q_k and R_k.

2. Forecast Step

$$u_k^{(b)} = \mathcal{M}(u_{k-1}^{(a)}) \tag{6.218}$$

$$B_k = D_M(u_{k-1}^{(a)})P_{k-1}D_M^T(u_{k-1}^{(a)}) + Q_k \tag{6.219}$$

3. Assimilation Step

$$K_k = B_k D_M^T(u_k^{(b)}) \left[D_h(u_k^{(b)})B_k D_h^T(u_k^{(b)}) + R_k\right]^{-1} \tag{6.220}$$

$$u_k^{(a)} = u_k^{(b)} + K_k(w_k - h(u_k^{(b)})) \tag{6.221}$$

$$P_k = \left(I - K_k D_h(u_k^{(b)})\right) B_k \tag{6.222}$$

6.9.7 3D-Var

For the **Kalman Filter** and the **EKF**, we derived the optimal way to combine observation with simulation so as to minimize the trace of the analysis error covariance matrix, P_k. An alternative approach is to recast the problem as a pure optimzation problem where rather than finding a filter K_k that will add an innovation to $u_k^{(b)}$ to obtain the analysis $u_k^{(a)}$, we obtain the analysis by optimizing the following cost function

$$J(u) = \frac{1}{2}\left(w - h(u)\right)^T R^{-1}\left(w - h(u)\right) + \frac{1}{2}\left(u - u^{(b)}\right)^T B^{-1}\frac{1}{2}\left(u - u^{(b)}\right) \tag{6.223}$$

which we can justify as coming from the joint probability distribution assuming Gaussian errors

$$\mathcal{P}(u|w) = C\exp\left(-\frac{1}{2}\left(u - u^{(b)}\right)^T B^{-1}\frac{1}{2}\left(u - u^{(b)}\right)\right)\cdot\exp\left(-\frac{1}{2}\left(w - h(u)\right)^T R^{-1}\left(w - h(u)\right)\right) \tag{6.224}$$

with model error covariance B and measurement error covariance R as before. This is clearly a *very strong assumption.*

To optimize $J(u)$, we begin by taking it's gradient.

$$\nabla_u J(u) = -D_h^T R^{-1}(w - h(u)) + B^{-1}(u - u^{(b)}) \tag{6.225}$$

Thus, finding the analysis $u^{(a)}$ ammounts to solving the system

$$a - D_h^T R^{-1}(w - h(u^{(a)})) + B^{-1}(u^{(a)} - u^{(b)}) = 0 \tag{6.226}$$

As for Kalman filtering, let's begin with the assumption that our model and observation function are linear.

6.9.7.1 Linear Case

Suppose that we have $h(u) = Hu$ so that $D_h(u) = H$. Then, we have

$$D_h^T R^{-1}(w - Hu^{(a)}) = B^{-1}(u^{(a)} - u^{(b)}) \tag{6.227}$$

$$D_h^T R^{-1} w - D_h^T R^{-1} H u^{(a)} = B^{-1} u^{(a)} - B^{-1} u^{(b)} \tag{6.228}$$

$$\left(D_h^T R^{-1} H + B^{-1}\right) u^{(a)} = D_h^T R^{-1} + B^{-1} u^{(b)} \tag{6.229}$$

$$\left(H^T R^{-1} H + B^{-1}\right) u^{(a)} = H^T R^{-1} + B^{-1} u^{(b)} \tag{6.230}$$

Thus we see that the analysis is given by

$$u^{(a)} = u^{(b)} + BH^T \left(R + HB^T H\right)^{-1} (w - Hu^{(b)}) \tag{6.231}$$

which agrees with what we found for the Linear Kalman Filter.

6.9.7.2 Nonlinear Case

To deal with the nonlinearity, we can expand h about an initial guess $u^{(c)}$ which we will later choose to be $u^{(b)}$ for convenience.

$$h(u^{(a)}) \approx h(u^{(c)}) + D_h(u^{(c)})\Delta u \tag{6.232}$$

Using this, we have

$$D_h^T(u^{(a)}) R^{-1}(w - h(u^{(a)})) = B^{-1}(u^{(a)} - u^{(b)}) \tag{6.233}$$

$$D_h^T(u^{(a)}) R^{-1}(w - h(u^{(c)}) - D_h(u^{(c)})\Delta u) \approx B^{-1}(u^{(c)} + \Delta u - u^{(b)}) \tag{6.234}$$

$$D_h^T(u^{(c)}) R^{-1}(w - h(u^{(c)}) - D_h(u^{(c)})\Delta u) \approx B^{-1}(u^{(c)} + \Delta u - u^{(b)}) \tag{6.235}$$

which we now solve for the update Δu to obtain the linear system

$$\left(B^{-1} + D_h^T(u^{(c)})R^{-1}D_h(u^{(c)})\right)\Delta u = B^{-1}(u^{(b)} - u^{(c)}) + D_h^T(u^{(c)})R^{-1}(w - h(u^{(c)})) \tag{6.236}$$

Thus we have the following prescription 1. To begin, take $u^{(c)} == u^{(b)}$. 2. Solve the system

$$\left(B^{-1} + D_h^T(u^{(c)})R^{-1}D_h(u^{(c)})\right)\Delta u = B^{-1}(u^{(b)} - u^{(c)}) + D_h^T(u^{(c)})R^{-1}(w - h(u^{(c)})) \tag{6.237}$$

to obtain Δu

3. Update your guess using your favorite optimization algorithm. For example, in steppest descent, choose a learning rate η and set

$$u^{(c)}_{\text{new}} = u^{(c)}_{\text{prev}} + \eta\Delta u \tag{6.238}$$

4. Repeat the procedure until $|u^{(c)}_{\text{new}} - u^{(c)}_{\text{prev}}|$ converges to a desired tolerance.

In both the linear and nonlinear case, it should be noted that we have not added time indices to our state vectors. This is an indication that the 3d-var procedure is performed **at every time where you have observation data**.

6.9.8 4D-Var

The **3D-Var** algorithm attempts to optimize a cost function to obtain the ideal analysis *for each point where we have observation data.* This can become computationally expensive as we require model evaluations *and* an optimization routine for every observation point.

An alternative approach is to simultaneously optimize accross all observations in order to obtain the ideal *initial condition* that acheive the best model fit. This approach is similar to sensitivity analysis which seeks to fit a model's parameters to data.

To begin, we construct the 4d-var cost function

$$
\begin{aligned}
J(u_0) &= \frac{1}{2}\left(u_0 - u_0^{(b)}\right)^T B^{-1}\left(u_0 - u_0^{(b)}\right) + \frac{1}{2}\sum_k \left(w_k - h(u_k)\right)^T R_k^{-1}\left(w_k - h(u_k)\right) && (6.239)\\
&= J_b(u_0) + J_m(u_0) && (6.240)
\end{aligned}
$$

The first term is usefull if we already have an initial guess $u_0^{(b)}$ for the inital condition in mind. If we do not have one, we may ommit this term.

As before, we now want to optimize this cost function. To do so, we first observe that

$$
u_k = \mathcal{M}^{(k)}(u_0;\theta) \tag{6.241}
$$

It is easy to obtain the gradient of J_0 so we shall focus on the second term. We find that

$$
\begin{aligned}
\nabla_{u_0} J_m &= \nabla_{u_0}\Big\{\sum_k \frac{1}{2}\left(w_k - h(u_k)\right)^T R_k^{-1}\left(w_k - h(u_k)\right)\Big\} && (6.242)\\
&= -\sum_k \left[\frac{\partial}{\partial u_0} h\left(\mathcal{M}^{(k-1)}(u_0)\right)\right]^T R_k^{-1}\left(w_k - h(u_k)\right) && (6.243)\\
&= -\sum_k \left[D_h(u_k) D_M(u_{k-1}) D_M(u_{k-2}) \cdots D_M(u_0)\right]^T R_k^{-1}\left(w_k - h(u_k)\right) && (6.244)\\
&= -\sum_k \left[D_M^T(u_0) D_M^T(u_1) \cdots D_M^T(u_{k-1}) D_h^T(u_k)\right] R_k^{-1}\left(w_k - h(u_k)\right) && (6.245)\\
& && (6.246)
\end{aligned}
$$

Given that we can now obtain the gradient of the cost function, the procedure is nearly identical to 3d-var:

1. Integrate your model forward to obtain $\{u_k\}$
2. Evaluate each of the $D_M^T(u_{k-1:0})$ and $D_h(u_k)$.
3. Using these values, compute $\nabla J_m(u)$
4. Set $u_0^{(new)} = u_0^{(prev)} - \eta \nabla J(u_0^{(prev)})$
5. Stop when $|u_0^{(new)} - u_0^{(prev)}|$ converges to your desired tolerance.

You can of course substitute another optimzation scheme after step 3.

6.9.9 Sensitivity Analysis for Differential Equations

Provided some model for a physical system in the form of a set of differential equations, a natural question is: How can we select the parameters for our model in order to get the best possible fit to some experimental data. Similarly, one may wonder what would happen to the prediction of your model if you were to slightly change the values of some parameters. In other words, how *sensitive* is the output of our model to your choice of parameter values?

In the most general sense, we may frame the problem as follows. Suppose we have a model of the form

$$\frac{du}{dt} = f(u, t, \theta) \tag{6.247}$$

Given this model, our goal is to optimize a cost function

$$J(u; \theta) := \int_0^T g(u; \theta) dt \tag{6.248}$$

where $g(u;\theta)$ is usually taken to be some *quadratic form.*

As an example, we might consider $g(u\ \theta) = (u(t) - w(t))^T(u(t) - w(t))$ where $w(t)$ denotes the vector of observations at time t.

Our goal then is to find out how J depends on the parameters θ, in other words, to find $\partial J/\partial\theta$. To do this, we will use the method of Lagrange multipliers to generate a so called *adjoint equation* that enables us to find this derivative in a way that minimizes computational cost. As always, this method begins by adding a term that evaluates to 0 into our cost function:

$$\mathcal{L} := \int_0^T \left[g(u;\theta) + \lambda^T(t)\left(f - \frac{du}{dt}\right)\right] dt \tag{6.249}$$

From this, we find

$$\frac{\partial\mathcal{L}}{\partial\theta} := \int_0^T \left[\frac{\partial g}{\partial\theta} + \frac{\partial g}{u}\frac{\partial u}{\partial\theta} + \lambda^T(t)\left(\frac{\partial f}{\partial\theta} + \frac{\partial f}{\partial u}\frac{\partial u}{\partial\theta} - \frac{d}{dt}\frac{\partial u}{\partial\theta}\right)\right] dt \tag{6.250}$$

$$= \int_0^T \left[\frac{\partial g}{\partial\theta} + \lambda^T(t)\frac{\partial f}{\partial\theta} + \left(\frac{\partial g}{\partial u} + \lambda^T(t)\frac{\partial f}{\partial u} - \lambda^T(t)\frac{d}{dt}\right)\frac{\partial u}{\partial\theta}\right] dt \tag{6.251}$$

This reorganization is nice because the term $\partial u/\partial\theta$ is the one thats *hard* to compute. Therefore, if we can make the terms in the paretheses evaluate to 0, we will be able to remove this pesky term. Let's use integration by parts to further rearrange by moving the d/dt.

$$\int_0^T -\lambda^T(t)\frac{d}{dt}\frac{\partial u}{\partial\theta}dt = \left[-\lambda^T(t)\frac{\partial u}{\partial\theta}\right]_0^T + \int_0^T \frac{d\lambda^T(t)}{dt}\frac{\partial u}{\partial\theta}dt \tag{6.252}$$

$$= \lambda^T(0)\frac{\partial u_0}{\partial\theta} - \lambda^T(T)\frac{\partial u(T)}{\partial\theta} + \int_0^T \left[\frac{d\lambda}{dt}\right]^T \frac{\partial u}{\partial\theta}dt \tag{6.253}$$

so that plugging this back into our expression for $\partial\mathcal{L}//\partial\theta$, we obtain

$$\frac{\partial\mathcal{L}}{\partial\theta} = \int_0^T \left[\frac{\partial g}{\partial\theta} + \lambda^T\frac{\partial f}{\partial\theta} + \left(\frac{\partial g}{\partial u} + \lambda^T\frac{\partial f}{\partial u} + \left[\frac{d\lambda}{dt}\right]^T\right)\frac{\partial u}{\partial\theta}\right] dt + \lambda^T(0)\frac{\partial u_0}{\partial\theta} - \lambda^T(T)\frac{\partial u(T)}{\partial\theta} \tag{6.254}$$

Thus, forcing the nasty terms to dissappear is equivalent find the $\lambda(t)$ subject to the differential equations

$$\frac{\partial g}{\partial u} + \lambda^T(t)\frac{\partial f}{\partial u} + \frac{d\lambda^T(t)}{dt} = 0 \tag{6.255}$$

$$\lambda^T(T) = 0 \tag{6.256}$$

or by taking the transpose:

$$\frac{d}{dt}\lambda = -\left[\frac{\partial g}{\partial u}\right]^T - \left[\frac{\partial f}{\partial u}\right]^T\lambda \tag{6.257}$$

$$\lambda(T) = 0 \tag{6.258}$$

6.9.9.1 Summary

To find the sensitivities $\partial J/\partial\theta$, we perform the following:

1. Integrate the model $du/dt = f(u, t, \theta)$ forward to obtain $u(t)$.
2. Integrate the adjoint model $d\lambda/dt = -(\partial f/\partial u)^T\lambda - (\partial g/partialu)^T$ backwards in time from T to 0 to obtain $\lambda(t)$.
3. Evaluate $\partial J/\partial\theta = \int_0^T (\partial g/\partial\theta + \lambda^T\partial f/\partial\theta)\, dt + \lambda^T(0)\partial u_0/\partial\theta$

6.10 Conformal Prediction

The focus of this section can be on the concept of uncertainty quantification. For many physics theories that are nicely linearized, uncertainty analysis can be easily accomplished at the level of first order sensitivites. That is, we can look at the Jacobian of our model to infer the behavior of small deviations about initial conditions. This approach does not easily extend to more complicated domains where the nonlinear effects dominate. Further, we also often want to establish ways to think about the fundamental instrument uncertainty for a measuring device. This can require meticulous calibrations which often assume a linear or polynomial fit... We can do better. Why not let the data tell us what the measurement uncertainty really is?

A good motivating example for the discussion of instrumental uncertainty is the use of a thermistor to measure temperature. One must assume a reasonable range of temperatures to establish the linear relationship between temperature and resistivity that is used determine the temperature. However, the material characteristics of the thermistor that introduce nonlinearities at extreme temperatures don't necessarily mean we should have to throw out measurements that do not fall within this well-behaved range. Rather, we can preform a more sophisticated *calibration* to learn a model mapping resistivity to temperature that can account for these effects.

This is the bread-and-butter of the MINTS sensing efforts. Often low-cost sensing solutions provide decent measurements within a limit domain. With quality data from superior (but often prohibitively expensive) reference instruments, we can improve the default calibration to improve the reliability of data (by reducing uncertainty) and extend it's domain of usefulness.

6.11 Generative Methods

6.12 Topological Data Analysis

6.13 Auto Encoders

This is a good place to talk about dimensionality reduction in general, e.g. PCA and other linear methods...

6.14 Physics Informed Neural Networks

6.15 Universal Differential Equations

It may also be nice to add a section on model evaluation criteria. Similarly, we can have a section on *Feature selection and dimensionality reduction*

CHAPTER 7

ROBOT TEAM

7.1 Georectification

7.2 georectification

7.3 supervised ML for concentration

7.4 super resolution if we have time (and open data cube)

7.5 solar geometry

7.6 reflectance/radiance

7.7 unsupervised methods

7.8 synthetic data generation

7.9 RobotTeam Papers

7.9.1 Robot Team II: Electric Boogaloo (title w.i.p.)

- Discuss real time georectification, generation of reflectance data, etc...
- Combine Multiple days of observations
- Discuss need for both viewing geometry and solar geometry
  - make reference to highly nonuniform reflectance as a function of incident angle
  - make reference to Beer's law as justification for direct determination of concentration of concentration from spectra
  - in depth discussion of fluorometers (maybe save this for the dissertation)

7.9.2 Unsupervised Classification of Hyperspectral Imagery for Rapid Characterization of Novel Environments with Autonomous Robotic Teams

7.9.2.1 K-means / Fuzzy K-means

7.9.2.2 Self Organizing Maps

- fit an SOM model to the data
- for each pixel in entire map, assign best matching unit (use distinguishable colors for a 10×10 or 25×25 SOM grid)
- For class, investigate the learned "spectrum" representation and compare against known chemical spectra.
  - is there a database we could try to use to look up possible species in the reflectance spectra?

7.9.2.3 Generative Topographic Mapping

7.9.2.4 analysis ideas

- The nice feature of both the SOM and the GTM is we can reinterpret them to be spectral-unmixing models. Each SOM node has a feature vector of identical length to the input vector. Similarly, the mean projection $y(x;W)$ in GTM represents the center of a gaussian in *data space*. Thus, we can reinterpret these feature vectors and gaussian centers as representative endmember spectra for the dataset. Further, the topographic properties of these methods ensure we have similarity between classes (at least in the latent space). For the GTM we are guarenteed that the data space projections of our GTM nodes will be similar. **We should see if this holds for SOM**. We can also interpret the cluster means from K-means as our endmembers.
- For SOM and GTM once we have fit the models, we can look at the map of "winning nodes" (BMU for SOM and Mean for GTM) and perform a secondary clustering (via

K-means or DBSCAN). These new clusters can define our endmember-bundles for a spectral-unmixing model.

- Once we have these maps, we should color the point by which day the data came from to see if there are any interesting groups or if the data are distributd across observation days
- We should make sure we apply the method for unsupervised classification to the dye-released images (see if the maps mirror the diffusion of dye). Pick a day where we have >1 dye flight. Make a map for both cases to illustrate how rapidly fitting an unsupervised model on the fly can enable near real time tracking of plume evolution (good for defense, oil spills, etc...). –> can we then construct a vector field / flow field from the difference and predict the plume evolution? Maybe this would be a good excuse to try lagrangian particle tracking...
- Provided our GTM/SOM fits, we should try a secondary
- From nodes/node clusters, can we identify spectral endmembers that represent chemicals we measure (e.g. chlorophyll)?
- From node activations (SOM) or responsabilities (GTM) can we fit a good model that competes with predictions from full spectra? (dimensionality reduction demonstration)
- cluster viewing geometry separate from refelctances and use resulting viewing geo classes to color GTM/SOM map of reflectance data

7.9.3 Spectral Indices for Rapid HSI Surveys: Unsupervised and Supervised Methods via SciML

- Apply to PROSPECT database as a simple test case
- Apply to our own Data
  - Mutliple Endmember Spectral Mixture Analysis

  - generalize **Spectral Unmixing Models** unmixing models… with gaussian process we could think of an infinite basis of gaussians describing "peaks" in the spectrum". Can we try to kernelize this procedure?

7.9.4 Synthetic Data Generation for Hyperspectral Imaging with Autonomous Robotic Teams

- Variational Autoencoders
- Group transformations, e.g. rotations, reflections, translations, cropping, etc… (do these make sense if boat data is point observation)
- Advanced sampling methods for regions with

7.9.5 Uncertainty Quantification via ∂P.

- Categorize Methods into two categories:
  - quantifying uncertainty in collected data
  - quantifying model uncertainty
- Conformal Prediction (we have a NN code for doing this in flux. Just need to apply it)
- Representativeness Uncertainty i.e. when georectifying HSI images and reducing spatial extend via `ilat` and `ilon` settings, also compute the stdev for each grainy pixel
- Measurements.jl *forward mode* once we have the representativeness uncertainty.
- Need a way to quantify uncertainty from Boat sensors…
- Sensativity Analysis with w/ automatic differentiation

the stuff regarding the full robot team that appeared in our 2020 robot team paper that Dr. Lary was the first author on.

7.10 Supervised Methods

stuff from paper # 1

7.11 Unsupervised Methods

stuff from paper # 2

CHAPTER 8

SUPER RESOLUTION

8.1 Cloud & Shadow Mask for Sentinel-2

8.1.1 ML Type

- Supervised Classification

8.1.2 ML Methods

- Single Pixel w/ Tree Based Methods
- Deep NN with Convolutional Layers

8.1.3 Features

- Sentinel 2 Multi-band Imagery
- Land Type
- Viewing Geometry
- Solar Geometry

8.1.4 Targets

- Sentinel 2 Cloud Mask + Cloud Shadow Mask

8.2 Cloud & Shadow Fill

8.2.1 ML Type

- supervised regression

8.2.2 ML Methods

- pixel based (Would it make sense to do something else here?)

8.2.3 Features

- Sentinel 2 Multi-band Imagery
- Sentinel 2 Cloud Mask & Cloud Shadow Mask
- Sentinel 1 SAR Variables (GRD or SLC or both?)
- 10 m Digital Elevation Map
- Viewing Geometry
- Solar Geometry
- Land Type

8.2.4 Targets

- *Cloudless & Shadowless* Sentinel 2 Multi-band imagery

8.3 Sentinel 2 RGB *Spatial* Super Resolution

8.3.1 ML Type

- Supervised Regression

8.3.2 ML Methods

- This has to be a Deep NN method using convolution to get the upsampling. I don't think we can do this with pixel based models (using tree methods)
- Probably should use a GAN

8.3.3 Features

- High (spatial) Resolution NAIP RGB Image
- Sentinel 2 Multi-band Imagery
- Sentinel 1 SAR Variables (GRD or SLC or both?)
- 10 m Digital Elevation Map
- Viewing Geometry
- Solar Geometry
- Land Type

8.3.4 Targets

- Sentinel RGB Bands @ NAIP Resolution

8.3.5 Loss Function Terms

8.3.6 Notes

We could make a model that uses all 3 bands (RGB) simultaneously, or we can make a model for a single band that we validate against R, G, and B bands individually. This has the added perc of increasing the training samples. This will be much easier to then apply to *all* bands of the sentinel imagery (and perhaps Sentinel 1, etc…) independently. We could try:

- Red, Green, Blue bands separately
- Black and White converted RGB image
- Data Augmentation via Scaling / Rotation / Reflection

8.4 Sentinel 2 Multiband *Spatial* Super Resolution

8.4.1 ML Type

- Supervised Regression

8.4.2 ML Methods

- This has to be a Deep NN method using convolution to get the upsampling. I don't think we can do this with pixel based models (using tree methods)
- Probably should use a GAN

8.4.3 Features

- High (spatial) Resolution NAIP RGB Image
- Sentinel 2 Multi-band Imagery
- Sentinel 1 SAR Variables (GRD or SLC or both?)
- 10 m Digital Elevation Map
- Viewing Geometry
- Solar Geometry
- Land Type

8.4.4 Targets

- Sentinel RGB Bands @ NAIP Resolution

8.4.5 Loss Function Terms

8.5 Sentinel 2 Multiband *Spectral* Super Resolution

8.5.1 ML Type

- Supervised Regression

8.5.2 ML Methods

- This can be pixel based

8.5.3 Features

- Sentinel 2 Multi-band Imagery
- Sentinel 1 SAR Variables (GRD or SLC or both?)
- 10 m Digital Elevation Map
- Viewing Geometry
- Solar Geometry
- UAV Hyperspectral Image

8.5.4 Targets

- Sentinel *Hyperspectral* Imagery (i.e. Sentinel at all HSI Bands)

8.5.5 Loss Function Terms

if we get to it… probably with a focus on Scotty's Ranch

CHAPTER 9

CHEMICAL DATA ASSIMILATION

9.1 Photolysis

9.2 HEART Chamber

CHAPTER 10

MASTER CHEMICAL MECHANISM

10.1 sensor fusion

10.2 photolysis

10.3 docker ingestion framework

10.4 master chemical mechanism

10.5 data assimilation

10.6 Chemical Data Assimilation & ActivePure Work

10.6.1 ActivePure Research Lab

- Overview of all sensor in sensor matrix
- Overview of measurement capabilities (list of species, uncertainty levels, etc…)
- Overview of containerized data acquisition pipeline
  - NodeRed
  - InfluxDB
  - Grafana
  - Quarto
  - Automatic Alerts
  - Automatic Reports

10.6.2 Kinetics and Chemical Data Assimilation

- MCM Implementation in Julia
- Direct computation of Photolysis rates
- Combination with Dr. Lary's AutoChem
- Addition of Ion Chemistry from MIT Lightning disseration
- Visualization of chemical cycles
- SciML methods to infer below detection limits

10.7 CRI Mechanism

10.8 AutoChem

10.9 Ion Chemistry

10.10 Evaluation Indoor Air Quality via Chemical Data Assimilation

This is where we put our non-ActivePure results

Specific testing done (e.g. in fourth floor lab space) sans-AcitvePure technology

10.11 Photocatalytic Ionization

This is where we put our ActivePure-specific results

Testing results done with ActivePure technology

CHAPTER 11

TIME SERIES METHODS

11.1 Uncertainty Estimation Via Time Series Sampling

11.1.1 Types Of Uncertainty

11.1.2 Instrument Uncertainty

11.1.3 Representativeness Uncertainty

11.1.4 Variograms

11.1.5 Mutual Information

11.1.6 Auto-correlation

11.2 Time Series Chaos

11.2.1 What is Chaos?

11.2.2 Lyapunov Exponents

11.2.3 Fractal Dimension

11.3 Koopman Operator Theory

11.4 Time Series Modeling Methods

11.4.1 Token-Hankel Delay Embeddings

11.4.2 Embedding Theorems (are magic)

11.4.3 Determination of Optimal Lag

11.4.4 Determination of Intrinsic Dimension (kind of unnecessary given *large enough* embedding)

11.4.5 DMD and HAVOK

11.4.6 Hamiltonian Neural Networks

11.5 Time Series Classification

11.15 Sensor Network + SciML

11.15.1 Evaluation of local chaos in SharedAirDFWNetwork

- This gives me an excuse to work with the sensor data
- Train GTM, SOM, and Variational Autoencoder to produce lower dimensional representation of all data from a central node, e.g. in $\mathbb{R}^2$.
  - For VAE, test a range of dimensions from the number of sensors down to 2 (better for visualization)
- Analyze the variety of methods from DataDrivenDiffEq.jl to infer dynamics in the low dimensional space
- Can we infer some kind of Hamiltonian from the data and do a HamiltonianNN approach?
  - Start of with a standard kinetic-energy style Hamiltonian e.g. $\sum_i \frac{1}{2}\dot{x}_i^2$ where x_i is the
  - use DataDrivenDiffEq approach to learn the associated potential energy term
  - alternatively, attempt to capture diurnal cycle (or other relevant time scales) by *learning* coordinate representation that forces dynamics to be uncoupled harmonic oscillators a la Hamilton-Jacobi theory.
  - Test if this hamiltonian NN model can then be transfered to another central node with an appropriate shift in the “total energy”
- Attempt to analyze the 2d data to infer Koopman operator. We should treat the original sensor values as observables on which the learned koopman operator acts. This should be doable if the NN is just a function.

- use DMD appraoch to identify a "forcing" coordinate that can identify when we switch nodes as in Chaos as an intermittently forced linear system
- video on Physics Informed DMD
- Deep Learning to Discover Coordinates for Dynamics: Autoencoders & Physics Informed Machine Learning
- NOTE: we may need to impute missing values. We shoud do so with either my GPR code or with other ML methods + ConformalPrediction. Provided uncertainty estimates, we should then think about how to propagate errors through our analysis via Measurements.jl, IntervalArithmetic,

CHAPTER 12

DISCUSSION

12.1 Limitations

12.2 Future Work

CHAPTER 13

CONCLUSIONS

CHAPTER 14

TECHNICAL NOTES

14.1 Real Time Georectification of Drone Based Imagery

- Georectification of pushbroom HSI
- Georectifcation of square visible + thermal FLIR imagery

14.2 Self Organizing Maps

14.3 Bayesian Optimization with Gaussian Process Regression

14.4 Gaussian Process Regression / Classification in MLJ

j ## Solar Geometry? (probably not necessary)

CHAPTER 15

OTHER TOPICS

- Sparse Nonlinear Models for Fluid Dynamics with Machine Learning and Optimization
- Residual Dynamic Mode Decomposition: A very easy way to get error bounds for your DMD computations
- Deep Learning of Dynamics and Coordinates with SINDy Autoencoders
- Identifying Dominant Balance Physics from Data

REFERENCES

Al-Fuqaha, Ala, Mohsen Guizani, Mehdi Mohammadi, Mohammed Aledhari, and Mutlag Ayyash. 2015. “Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications.” *IEEE Communications Surveys & Tutorials* 17 (4): 2347–76. https://doi.org/10.1109/COMST.2015.2444095.

Atzori, Luigi, Antonio Iera, and Giacomo Morabito. 2010. “The Internet of Things: A Survey.” *Computer Networks* 54 (15): 2787–2805. https://doi.org/10.1016/j.comnet.2010.05.010.

Bezanson, Jeff, Alan Edelman, Stefan Karpinski, and Viral B. Shah. 2017. “Julia: A Fresh Approach to Numerical Computing.” *SIAM Review* 59 (1): 65–98. https://doi.org/10.1137/141000671.

Brook, Robert D., Sanjay Rajagopalan, C. Arden Pope III, Jeffrey R. Brook, Aruni Bhatnagar, Ana V. Diez-Roux, Fernando Holguin, et al. 2010. “Particulate Matter Air Pollution and Cardiovascular Disease.” *Circulation* 121 (21): 2331–78. https://doi.org/10.1161/CIRCULATIONAHA.109.893472.

Buyantuyev, Alexander, and Jiquan Wu. 2017. “Remote Sensing Applications for Land Cover and Land-Use Transformations in Semiarid and Arid Environments.” *Journal of Arid Environments* 140: 1–5. https://doi.org/10.1016/j.jaridenv.2017.01.008.

Carleo, Giuseppe, Kenny Choo, Johannes Hofmann, Edward Huang, Chris Hughes, Michael Hush, Raban Iten, et al. 2019. “Machine Learning and the Physical Sciences.” *Reviews of Modern Physics* 91 (4): 045002.

Centre for Ecology and Hydrology. 2017. “Ecological Sensing: A Revolution in Biodiversity Monitoring.” https://www.ceh.ac.uk/news-and-media/blogs/ecological-sensing-revolution-biodiversity-monitoring.

Chen, Jiawei, Xiaoming Shi, Xinyi Li, Mingjie Wang, Wei Shen, and Yanzhao Liu. 2019. "A Review of Air Quality Modeling: From Gas-Phase to Particulate Matter." *Advances in Atmospheric Sciences* 36 (10): 921–47. https://doi.org/10.1007/s00376-019-9047-1.

Chen, Yan, Xun Zhu, Haibo Wang, Wei Ren, and Simon X. Yang. 2017. "Autonomous Robots with Decentralized, Collective Decision-Making." *Proceedings of the IEEE* 105 (2): 321–37. https://doi.org/10.1109/JPROC.2016.2628400.

Clark, Martyn P., W. Neil Adger, Suraje Dessai, Marisa Goulden, David W. Cash, and Richard and Dickson Stern Nicholas and Gonzalez. 2016. "Urbanization, Climate Change and Economic Growth: Challenges and Opportunities for Policy Makers." *Science of the Total Environment* 557-558: 279–91. https://doi.org/10.1016/j.scitotenv.2016.03.022.

Costello, Anthony, Mustafa Abbas, Adriana Allen, Sarah Ball, Sarah Bell, Richard Bellamy, Sharon Friel, et al. 2009. "Managing the Health Effects of Climate Change: Lancet and University College London Institute for Global Health Commission." *The Lancet* 373 (9676): 1693–733. https://www.thelancet.com/journals/lancet/article/PIIS0140-6736(09)60935-1/fulltext.

DeLucia, Evan H., Nuria Gomez-Casanovas, Stephen P. Long, Melanie A. Mayes, Rebecca A. Montgomery, William J. Parton, William J. Sacks, Joshua P. Schimel, Joseph Verfaillie, and Whendee L. Silver. 2021. "The Missing Soil n: Detecting Processes Driving Soil Nitrogen Storage in Complex Ecosystems." *Journal of Ecology* 109 (2): 447–59. https://doi.org/10.1111/1365-2745.13550.

Dunbabin, Matthew, and Lino Marques. 2012. "Robotic Mapping of Environmental Variables for Prediction and Control." *Philosophical Transactions of the Royal Society A* 370 (1962): 298–308. https://doi.org/10.1098/rsta.2011.0243.

Edenhofer, O., R. Pichs-Madruga, Y. Sokona, E. Farahani, S. Kadner, K. Seyboth, A. Adler, et al. 2014. *Climate Change 2014: Mitigation of Climate Change. IPCC Working Group II*. Cambridge University Press. https://doi.org/10.1017/CBO9781107415416.

enVerid. 2022a. "How to Achieve Sustainable Indoor Air Quality: A Roadmap to Simultaneously Improving Indoor Air Quality & Meeting Building Decarbonization and Climate Resiliency Goals."

———. 2022b. "Leaders in Indoor Air Quality and Energy Efficiency Share Framework for Achieving Healthy Indoor Air While Decarbonizing Buildings."

Field, C. B., V. R. Barros, D. J. Dokken, K. J. Mach, M. D. Mastrandrea, T. E. Bilir, M. Chatterjee, et al. 2014. *Climate Change 2014: Impacts, Adaptation, and Vulnerability. Part a: Global and Sectoral Aspects*. Cambridge University Press. https://doi.org/10.1017/CBO9781107415379.

Friedlingstein, Pierre, Matthew W. Jones, Michael O'Sullivan, Robbie M. Andrew, Judith Hauck, Glen P. Peters, Wouter Peters, et al. 2020. "Global Carbon Budget 2020." *Earth System Science Data* 12 (4): 3269–3340. https://doi.org/10.5194/essd-12-3269-2020.

Gamon, John A., K. Fred Huemmrich, Robert S. Stone, and Craig E. Tweedie. 2016. "Spatial and Temporal Variation in Primary Productivity (NDVI) of Coastal Alaskan Tundra: Decreased Vegetation Growth Following Earlier Snowmelt." *Remote Sensing of Environment* 175: 233–42. https://doi.org/10.1016/j.rse.2015.12.051.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. 2016. "Deep Learning." *MIT Press*. https://doi.org/10.1016/j.neunet.2016.10.003.

Gubbi, Jayavardhana, Rajkumar Buyya, Slaven Marusic, and Marimuthu Palaniswami. 2013. "Internet of Things (IoT): A Vision, Architectural Elements, and Future Directions." *Future Generation Computer Systems* 29 (7): 1645–60. https://doi.org/10.1016/j.future.2013.01.010.

Haines, Andrew, R Sari Kovats, Diarmid Campbell-Lendrum, and Carlos Corvalán. 2006. "Climate Change and Human Health: Impacts, Vulnerability and Public Health." *Public Health* 120 (7): 585–96. https://doi.org/10.1016/j.puhe.2006.01.002.

Hantson, Stijn, Almut Arneth, Sandy P. Harrison, Douglas I. Kelley, I. Colin Prentice, Sam S. Rabin, Sally Archibald, et al. 2016. "The Status and Challenge of Global Fire Modelling." *Biogeosciences* 13 (11): 3359–75. https://doi.org/10.5194/bg-13-3359-2016.

Houghton, J. T., Y. Ding, D. J. Griggs, M. Noguer, P. J. van der Linden, X. Dai, K. Maskell, and C. A. Johnson. 2001. *Climate Change 2001: The Scientific Basis*. Cambridge University Press. https://doi.org/10.1017/CBO9780511546013.

Houghton, J. T., G. J. Jenkins, and J. J. Ephraums. 1990. *Climate Change: The IPCC Scientific Assessment*. Cambridge University Press. https://doi.org/10.1017/CBO9780511623521.

Houghton, J. T., L. G. Meira Filho, B. A. Callander, N. Harris, A. Kattenberg, and K. Maskell. 1996. *Climate Change 1995: The Science of Climate Change*. Cambridge University Press. https://doi.org/10.1017/CBO9780511809286.

Huang, Jiaxing, Lejiang Yu, Jianping Guo, Xiaofeng Guo, Wei Wang, Chunyan Liu, and Duoying Ji. 2017. "Assessment of Global Surface Energy Budget Datasets Using Flux Tower Observations." *Journal of Geophysical Research: Atmospheres* 122 (14): 7452–75. https://doi.org/10.1002/2016JD026049.

Jordan, Michael I., and Tom M. Mitchell. 2015. "Machine Learning: Trends, Perspectives, and Prospects." *Science* 349 (6245): 255–60. https://doi.org/10.1126/science.aaa8415.

Kelly, Frank J., and Julian C. Fussell. 2011. "Air Pollution and Public Health: Emerging Hazards and Improved Understanding of Risk." *Environmental Geochemistry and Health* 33 (4): 363–73. https://doi.org/10.1007/s10653-011-9415-1.

Kohonen, Teuvo. 1982. "Self-Organized Formation of Topologically Correct Feature Maps." *Biological Cybernetics* 43 (1): 59–69.

LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. 2015. "Deep Learning." *Nature* 521 (7553): 436–44. https://doi.org/10.1038/nature14539.

Li, Jun, Antonio Plaza, José Bioucas-Dias, Paul Gader, and Jocelyn Chanussot. 2018. "Guest Editorial Deep Learning for Hyperspectral Image Analysis." *IEEE Transactions* *on Geoscience and Remote Sensing* 56 (3): 1362–64. https://doi.org/10.1109/TGRS.2018.2801300.

Masson-Delmotte, V., P. Zhai, H.-O. Pörtner, D. Roberts, J. Skea, P. R. Shukla, A. Pirani, et al. 2018. *Global Warming of 1.5°c. An IPCC Special Report on the Impacts of Global Warming of 1.5°c Above Pre-Industrial Levels and Related Global Greenhouse Gas Emission Pathways, in the Context of Strengthening the Global Response to the Threat of Climate Change, Sustainable Development, and Efforts to Eradicate Poverty. IPCC Special Report*. Intergovernmental Panel on Climate Change. https://www.ipcc.ch/sr15/.

Metz, B., O. R. Davidson, P. R. Bosch, R. Dave, and L. A. Meyer. 2007. *Climate Change 2007: Mitigation of Climate Change*. Cambridge University Press. https://doi.org/10.1017/CBO9780511813573.

National Research Council. 2010. "Verifying Greenhouse Gas Emissions: Methods to Support International Climate Agreements." https://www.nap.edu/catalog/12883/verifying-greenhouse-gas-emissions-methods-to-support-international-climate-agreements.

Oleson, K. W., D. M. Lawrence, G. B. Bonan, and M. G. Flanner. 2013. "Interactions Between Land Use Change and Carbon Cycle Feedbacks." *Global Biogeochemical Cycles* 27 (4): 972–83. https://doi.org/10.1002/gbc.20073.

Parry, M. L., O. F. Canziani, J. P. Palutikof, P. J. van der Linden, and C. E. Hanson. 2007. *Climate Change 2007: Impacts, Adaptation and Vulnerability*. Cambridge University Press. https://doi.org/10.1017/CBO9780511546013.

Pasher, Jon, Bum-Jun Park, Jérôme Théau, Francois Pimont, and Scott Goetz. 2019. "Remote Sensing of Wetlands: An Overview and Practical Guide." *Wetlands Ecology and Management* 27 (2): 129–47. https://doi.org/10.1007/s11273-018-9624-3.

Plaza, Antonio, Jon A. Benediktsson, James W. Boardman, Jason Brazile, Lorenzo Bruzzone, Gustau Camps-Valls, Jocelyn Chanussot, et al. 2009. "Recent Advances in Techniques for Hyperspectral Image Processing." *Remote Sensing of Environment* 113 (S1): S110–22. https://doi.org/10.1016/j.rse.2008.10.008.

Rackauckas, Christopher, David Kelly, Qing Nie, Jesse Li, Cody Warner, Malvika Dhairya, Jie Fang, et al. 2020. "Universal Differential Equations for Scientific Machine Learning." *arXiv Preprint arXiv:2012.09345*.

Raissi, Maziar, and George Em Karniadakis. 2021. "Physics-Informed Neural Networks: A Deep Learning Framework for Solving Forward and Inverse Problems Involving Nonlinear Partial Differential Equations." *Journal of Computational Physics* 378: 686–707.

Raissi, Maziar, Paris Perdikaris, and George Em Karniadakis. 2019. "Physics-Informed Neural Networks: A Deep Learning Framework for Solving Forward and Inverse Problems Involving Nonlinear Partial Differential Equations." *Journal of Computational Physics* 378: 686–707.

Rubenstein, Michael, Alejandro Cornejo, and Radhika Nagpal. 2014. "Programmable Self-Assembly in a Thousand-Robot Swarm." *Science* 345 (6198): 795–99. https://doi.org/10.1126/science.1254295.

Solomon, S., D. Qin, M. Manning, Z. Chen, M. Marquis, K. B. Averyt, M. Tignor, and H. L. Miller Jr. 2007. *Climate Change 2007: The Physical Science Basis*. Cambridge University Press. https://doi.org/10.1017/CBO9780511546013.

Stocker, T. F., D. Qin, G.-K. Plattner, M. Tignor, S. K. Allen, J. Boschung, A. Nauels, Y. Xia, V. Bex, and P. M. Midgley. 2013. *Climate Change 2013: The Physical Science Basis*. Cambridge University Press. https://doi.org/10.1017/CBO9781107415324.

Thenkabail, Prasad S. 2019. "Remote Sensing of Global Croplands for Food Security." *Remote Sensing* 11 (10): 1261. https://doi.org/10.3390/rs11101261.

United Nations. 2015. "Transforming Our World: The 2030 Agenda for Sustainable Development." *UN General Assembly*. https://sdgs.un.org/2030agenda.

United Nations Environment Programme. 2017. "Adaptation Gap Report 2017." https://www.unep.org/resources/adaptation-gap-report-2017.

Wang, Donglian, Donghui Xie, Yichun Xie, and Chen Chen. 2017. "Remote Sensing Applications for Urban Water Resources: A Review." *Remote Sensing* 9 (8): 829. https://doi.org/10.3390/rs9080829.

World Health Organization. 2018. "Climate Change and Health." https://www.who.int/news-room/fact-sheets/detail/climate-change-and-health.

Wu, Jingtao, and Xiaohua Zhang. 2021. "A Review on Physics-Informed Machine Learning: Basic Principles, Recent Developments and Future Directions." *Physics Reports* 903: 1–45.

Xu, Xiaohui, Feng Deng, Xiuwei Guo, Ping Lv, Hua Zhong, Yuantao Hao, Guocheng Hu, et al. 2017. "Association Between Particulate Matter Air Pollution and Hospital Admissions in Patients with Chronic Obstructive Pulmonary Disease in Beijing, China." *Science of the Total Environment* 579: 1616–21. https://doi.org/10.1016/j.scitotenv.2016.11.166.

Zhu, Xiao Xiang, Devis Tuia, Lichao Mou, Gui-Song Xia, Liangpei Zhang, and Feng Xu. 2017. "Deep Learning in Remote Sensing: A Comprehensive Review and List of Resources." *IEEE Geoscience and Remote Sensing Magazine* 5 (4): 8–36. https://doi.org/10.1109/MGRS.2017.2762307.

APPENDIX A

ADDITIONAL STUFF

You might put some computer output here, or maybe additional tables. It is possible to have multiple appendices. Just list them in the appropriate place within `_quarto.yml`.

APPENDIX B

EEG BAND DISCOVERY WITH DECISION TREES

BIOGRAPHICAL SKETCH

Update required!

CURRICULUM VITAE