ENM 360: Introduction to Data-driven Modeling

Lecture #1: Introduction, motivation and course logistics

Paris Perdikaris
August 29th, 2019

Penn
UNIVERSITY of PENNSYLVANIA

ENM360: INTRODUCTION TO DATA-DRIVEN MODELING

From recognizing voice, text or images to designing more efficient airplane wings and discovering new drugs, machine learning is introducing a transformative set of tools in data analysis with increasing impact across engineering, sciences, and commercial applications. In this course, you will learn about principles and algorithms for extracting patterns from data and and making effective automated predictions. We will cover concepts such as regression, classification, density estimation, feature extraction, sampling, and probabilistic modeling, and provide a formal understanding of how, why, and when these methods work in the context of analyzing physical, biological, and engineering systems.

Roadmap to Trillion Sensors

Accelerometer
Gyro
Pendulum Resistive Tilt Sensors
Piezo Bend Sensor
Metal Detector
Gas Sensor
Gieger-Muller Radiation Sensor
Pyroelectric Detector
UV Detector
Resistive Bend Sensors
Digital Infrared Ranging
CDS Cell Resistive Light Sensor
Limit Switch
Mechanical Tilt Sensors
Touch Switch
Pressure Switch
Miniature Polaroid Sensor
IR Pin Diode
IR Sensor w/lens
Thyristor
Magnetic Sensor
Polaroid Sensor Board
IR Reflection Sensor
IR Amplifier Sensor
IRDA Transceiver
Magnetic Reed Switch
Hall Effect Magnetic Field Sensors
Lite-On IR Remote Receiver
Radio Shack Remote Receiver
IR Modulator Receiver
Solar Cell
Compass
Compass
Piezo Ultrasonic Transducers

Sensors are expected to generate a bronto-bytes (1000 trillion trillion) of data by 2025!

Recent success of machine learning

Working with high-dimensional data

Karras, T., Aila, T., Laine, S., & Lehtinen, J. (2017). Progressive growing of GANs for improved quality, stability, and variation. arXiv preprint.

Working with high-dimensional data

Chan, C., Ginosar, S., Zhou, T., & Efros, A.A. (2018). Everybody Dance Now. arXiv preprint arXiv:1808.07371.

Working with high-dimensional data

Motivation and open challenges

*image credit: S. Sherwin (Imperial)

$$\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} = -\frac{1}{\rho}\nabla p + \nu \nabla^2 \boldsymbol{u}$$

$$\nabla \cdot \boldsymbol{u} = 0$$

Computational science & engineering

One exaflop is a thousand petaflops or a quintillion (10^18) floating point operations per second!

All models are wrong

but some are useful

George E.P. Box

Predictive science

Oden, T., Moser, R., & Ghattas, O. (2010). Computer predictions with quantified uncertainty, part I. SIAM News, 43(9), 1-3.

CSE for ML

Leverage domain-specific prior information to construct more structured, interpretable, and accurate and data-efficient learning algorithms.

ML for CSE

Leverage recent developments in machine learning to construct robust data-driven solvers, discover new models, and address longstanding challenges in scientific computing (e.g. high-dimensional problems, UQ, etc.)

Correct PDE	$u_t + (uu_x + vu_y) = -p_x + 0.01(u_{xx} + u_{yy})$ $v_t + (uv_x + vv_y) = -p_y + 0.01(v_{xx} + v_{yy})$
Identified PDE (clean data)	$u_t + 0.999(uu_x + vu_y) = -p_x + 0.01047(u_{xx} + u_{yy})$ $v_t + 0.999(uv_x + vv_y) = -p_y + 0.01047(v_{xx} + v_{yy})$
Identified PDE (1% noise)	$u_t + 0.998(uu_x + vu_y) = -p_x + 0.01057(u_{xx} + u_{yy})$ $v_t + 0.998(uv_x + vv_y) = -p_y + 0.01057(v_{xx} + v_{yy})$

$$f : \mathcal{X} \to \mathcal{Y}$$

Supervised learning

$$f : \mathcal{X} \rightarrow \mathcal{Y}$$

$$\mathcal{D} = \{\boldsymbol{x}, \boldsymbol{y}\},\ \boldsymbol{x} \in \mathcal{X},\ \boldsymbol{y} \in \mathcal{Y}$$

$$\boldsymbol{y} = f(\boldsymbol{x}) + \epsilon$$

$$p(f(\boldsymbol{x}^*)|\boldsymbol{x}^*, \mathcal{D})$$

Regression

Classification

Classification

$$\mathcal{D} = \{\boldsymbol{x}, \boldsymbol{y}\},\ \boldsymbol{x} \in \mathcal{X},\ \boldsymbol{y} \in \mathcal{Y}$$

$$\boldsymbol{y} = f(\boldsymbol{x}) + \epsilon$$

Unsupervised learning

$$\{\boldsymbol{y}\},\ \boldsymbol{y} \in \mathcal{Y}$$

$$\boldsymbol{y} = f(\boldsymbol{z}) + \epsilon$$

$$p(\boldsymbol{y}),\ \boldsymbol{z} \sim p(\boldsymbol{z})$$

Density estimation

Clustering

Dimensionality reduction

Generative modeling

Generative modeling

Gómez-Bombarelli, R., el. al. (2016). Automatic chemical design using a data-driven continuous representation of molecules. arXiv preprint arXiv:1610.02415.

Figure 8: Molecules decoded from randomly-sampled points in the latent space of the ZINC VAE.

Uncertainty quantification

Course goals

- Learning how to analyze and synthesize data towards enhancing their understanding and ability to model physical, biological, and engineering systems.
- Hands-on skills on contemporary machine learning tools enabling them to construct prediction models, extract patterns and characterize the statistical properties of data.
- Applications of these tools spanning a diverse set of engineering disciplines, including fluid dynamics, heat transfer, mechanical design, and biomedical engineering.

<u>Key motifs:</u>

- Representation
- Approximation
- Optimization
- Control

Course logistics

- Duration: 14 Weeks
- Time: Tuesdays and Thursdays, 3:00 pm to 4:30 pm
- Dates: August 27 to December 19, 2019. No classes on Thursday October 10.
- Location: Towne Building, Room 303
- Office Hours: Thursdays 4:30 pm to 6:30 pm at Office 527A (3401 Walnut Street).
- Weekly/bi-weekly homework assignments, mid-term exam, final project.

For more details visit the course website:
https://www.seas.upenn.edu/~enm360/

Slides and code can be found on Github:
https://github.com/PredictiveIntelligenceLab/ENM360

Course outline

- Week 1: Course introduction, Python tutorial.
- Week 2: Primer on scientific computing: Linear systems, curve fitting, numerical differentiation and integration.
- Week 3: Primer on probability: random variables, distributions and moments, independence, Bayes rule.
- Week 4: Linear regression.
- Week 5: Logistic regression.
- Week 6: Density estimation.
- Week 7: Dimensionality reduction: Principal Component Analysis and Dynamic Mode Decomposition.
- Week 8: Sparse regression and compressive sensing.
- Week 9: Kalman filtering and state-space models.
- Week 10: Sampling and Monte Carlo inference.
- Week 11: Neural networks.
- Week 12: Kernel methods.
- Week 13: Bayesian optimization and active learning.
- Week 14: Final project presentations.

Textbooks

Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.

Reading

- The Emergence of Predictive Computational Science:
  - Computer predictions with quantified uncertainty (Oden, Moser, & Ghattas, 2010)
  - Lecture by J.T Oden.
- Review papers on recent advances in machine learning:
  - Probabilistic machine learning and artificial intelligence (Ghahramani, 2015)
  - Deep learning (LeCun, Bengio, & Hinton, 2015)
  - Machine learning: Trends, perspectives, and prospects (Jordan & Mitchell, 2015)
- Scientific computing in Python:
  - Lectures and code by Robert Johansson.

1. Oden, T., Moser, R., & Ghattas, O. (2010). Computer predictions with quantified uncertainty, part I. *SIAM News*, *43*(9), 1–3.
2. Ghahramani, Z. (2015). Probabilistic machine learning and artificial intelligence. *Nature*, *521*(7553), 452–459.
3. LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444.
4. Jordan, M. I., & Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. *Science*, *349*(6245), 255–260.

Software

Consult the Web for instructions and make sure you have the following software properly installed and working on your laptop:

- A Python 3 distrubution configured for scientific computing. The simplest way to set this up is by installing the Anaconda distribution.
- Tensorflow. If you have access to GPU hardware, make sure you install a version with GPU support.
- PyTorch. If you have access to GPU hardware, make sure you install a version with GPU support.
- Autograd. Efficiently computes derivatives of numpy code.
- Jupyter notebook. You will need this in order to follow some of the in-class tutorials.
- Git. You will need this in order to download and stay in sync with the latest code we will develop in class.

Python interpreter

The standard way to use the Python programming language is to use the Python interpreter to run python code. The python interpreter is a program that reads and execute the python code in files passed to it as arguments. At the command prompt, the command `python` is used to invoke the Python interpreter.

For example, to run a file `my-program.py` that contains python code from the command prompt, use::

```
$ python my-program.py
```

We can also start the interpreter by simply typing `python` at the command line, and interactively type python code into the interpreter.

This is often how we want to work when developing scientific applications, or when doing small calculations. But the standard python interpreter is not very convenient for this kind of work, due to a number of limitations.

IPython

IPython is an interactive shell that addresses the limitation of the standard python interpreter, and it is a work-horse for scientific use of python. It provides an interactive prompt to the python interpreter with a greatly improved user-friendliness.

Some of the many useful features of IPython includes:

- Command history, which can be browsed with the up and down arrows on the keyboard.
- Tab auto-completion.
- In-line editing of code.
- Object introspection, and automatic extract of documentation strings from python objects like classes and functions.
- Good interaction with operating system shell.
- Support for multiple parallel back-end processes, that can run on computing clusters or cloud services like Amazon EC2.

jupyter rht-stock-price (autosaved)

Logout

File Edit View Insert Cell Kernel Help

Python 3

Code CellToolbar

In [1]:
```
%matplotlib inline
```

In [2]:
```
import datetime
```

In [3]:
```
import pandas_datareader.data as web
```

In [4]:
```
end = datetime.date.today()
start = end - datetime.timedelta(days=365)
```

In [5]:
```
values = web.get_data_yahoo(["RHT"], start, end)['Adj Close']
```

In [6]:
```
values.plot()
```

Out[6]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5845889780>
```

Spyder

Spyder is a MATLAB-like IDE for scientific computing with python. It has the many advantages of a traditional IDE environment, for example that everything from code editing, execution and debugging is carried out in a single environment, and work on different calculations can be organized as projects in the IDE environment.

Some advantages of Spyder:

- Powerful code editor, with syntax high-lighting, dynamic code introspection and integration with the python debugger.
- Variable explorer, IPython command prompt.
- Integrated documentation and help.