COMP6248 Differentiable Programming
(and some Deep Learning)

Kate Farrahi and Jonathon Hare

Vision, Learning and Control
University of Southampton

Machine Learning - A Recap

All credit for this slide goes to Niranjan

Data $\quad \{\boldsymbol{x}_n, \boldsymbol{y}_n\}_{n=1}^{N} \quad \{\boldsymbol{x}_n\}_{n=1}^{N}$

Machine Learning - A Recap

All credit for this slide goes to Niranjan

Data	$\{\boldsymbol{x}_n, \boldsymbol{y}_n\}_{n=1}^{N}$	$\{\boldsymbol{x}_n\}_{n=1}^{N}$
Function Approximator	$\boldsymbol{y} = f(\boldsymbol{x}, \boldsymbol{\theta}) + \nu$	

Machine Learning - A Recap

All credit for this slide goes to Niranjan

Data	$\{\boldsymbol{x}_n, \boldsymbol{y}_n\}_{n=1}^N \qquad \{\boldsymbol{x}_n\}_{n=1}^N$
Function Approximator	$\boldsymbol{y} = f(\boldsymbol{x}, \boldsymbol{\theta}) + \nu$
Parameter Estimation	$E_0 = \sum_{n=1}^N \{\lVert \boldsymbol{y}_n - f(\boldsymbol{x}_n; \boldsymbol{\theta}) \rVert\}^2$

Machine Learning - A Recap

All credit for this slide goes to Niranjan

Data	$\{\boldsymbol{x}_n, \boldsymbol{y}_n\}_{n=1}^N \qquad \{\boldsymbol{x}_n\}_{n=1}^N$
Function Approximator	$\boldsymbol{y} = f(\boldsymbol{x}, \boldsymbol{\theta}) + \nu$
Parameter Estimation	$E_0 = \sum_{n=1}^N \{\lVert \boldsymbol{y}_n - f(\boldsymbol{x}_n; \boldsymbol{\theta}) \rVert\}^2$
Prediction	$\hat{\boldsymbol{y}}_{N+1} = f(\boldsymbol{x}_{N+1}, \hat{\boldsymbol{\theta}})$

Machine Learning - A Recap

All credit for this slide goes to Niranjan

Data	$\{\boldsymbol{x}_n, \boldsymbol{y}_n\}_{n=1}^N \qquad \{\boldsymbol{x}_n\}_{n=1}^N$
Function Approximator	$\boldsymbol{y} = f(\boldsymbol{x}, \boldsymbol{\theta}) + \nu$
Parameter Estimation	$E_0 = \sum_{n=1}^N \{\|\boldsymbol{y}_n - f(\boldsymbol{x}_n; \boldsymbol{\theta})\|\}^2$
Prediction	$\hat{\boldsymbol{y}}_{N+1} = f(\boldsymbol{x}_{N+1}, \hat{\boldsymbol{\theta}})$
Regularisation	$E_1 = \sum_{n=1}^N \{\|\boldsymbol{y}_n - f(\boldsymbol{x}_n; \boldsymbol{\theta})\|\}^2 + r(\|\boldsymbol{\theta}\|)$

Machine Learning - A Recap

All credit for this slide goes to Niranjan

Data	$\{\boldsymbol{x}_n, \boldsymbol{y}_n\}_{n=1}^N \qquad \{\boldsymbol{x}_n\}_{n=1}^N$
Function Approximator	$\boldsymbol{y} = f(\boldsymbol{x}, \boldsymbol{\theta}) + \nu$
Parameter Estimation	$E_0 = \sum_{n=1}^N \{\lVert \boldsymbol{y}_n - f(\boldsymbol{x}_n; \boldsymbol{\theta}) \rVert\}^2$
Prediction	$\hat{\boldsymbol{y}}_{N+1} = f(\boldsymbol{x}_{N+1}, \hat{\boldsymbol{\theta}})$
Regularisation	$E_1 = \sum_{n=1}^N \{\lVert \boldsymbol{y}_n - f(\boldsymbol{x}_n; \boldsymbol{\theta}) \rVert\}^2 + r(\lVert \boldsymbol{\theta} \rVert)$
Modelling Uncertainty	$p(\boldsymbol{\theta} \vert \{\boldsymbol{x}_n, \boldsymbol{y}_n\}_{n=1}^N)$

Machine Learning - A Recap

All credit for this slide goes to Niranjan

Data	$\{\boldsymbol{x}_n, \boldsymbol{y}_n\}_{n=1}^N \qquad \{\boldsymbol{x}_n\}_{n=1}^N$
Function Approximator	$\boldsymbol{y} = f(\boldsymbol{x}, \boldsymbol{\theta}) + \nu$
Parameter Estimation	$E_0 = \sum_{n=1}^N \{\|\boldsymbol{y}_n - f(\boldsymbol{x}_n; \boldsymbol{\theta})\|\}^2$
Prediction	$\hat{\boldsymbol{y}}_{N+1} = f(\boldsymbol{x}_{N+1}, \hat{\boldsymbol{\theta}})$
Regularisation	$E_1 = \sum_{n=1}^N \{\|\boldsymbol{y}_n - f(\boldsymbol{x}_n; \boldsymbol{\theta})\|\}^2 + r(\|\boldsymbol{\theta}\|)$
Modelling Uncertainty	$p(\boldsymbol{\theta} \| \{\boldsymbol{x}_n, \boldsymbol{y}_n\}_{n=1}^N)$
Probabilistic Inference	$\mathbb{E}[g(\boldsymbol{\theta})] = \int g(\boldsymbol{\theta}) p(\boldsymbol{\theta}) d\boldsymbol{\theta} = \frac{1}{N_s} \sum_{n=1}^{N_s} g(\boldsymbol{\theta}^{(n)})$

Machine Learning - A Recap

All credit for this slide goes to Niranjan

Data	$\{\boldsymbol{x}_n, \boldsymbol{y}_n\}_{n=1}^N \qquad \{\boldsymbol{x}_n\}_{n=1}^N$
Function Approximator	$\boldsymbol{y} = f(\boldsymbol{x}, \boldsymbol{\theta}) + \nu$
Parameter Estimation	$E_0 = \sum_{n=1}^N \{\|\boldsymbol{y}_n - f(\boldsymbol{x}_n; \boldsymbol{\theta})\|\}^2$
Prediction	$\hat{\boldsymbol{y}}_{N+1} = f(\boldsymbol{x}_{N+1}, \hat{\boldsymbol{\theta}})$
Regularisation	$E_1 = \sum_{n=1}^N \{\|\boldsymbol{y}_n - f(\boldsymbol{x}_n; \boldsymbol{\theta})\|\}^2 + r(\|\boldsymbol{\theta}\|)$
Modelling Uncertainty	$p(\boldsymbol{\theta} \| \{\boldsymbol{x}_n, \boldsymbol{y}_n\}_{n=1}^N)$
Probabilistic Inference	$\mathbb{E}[g(\boldsymbol{\theta})] = \int g(\boldsymbol{\theta}) p(\boldsymbol{\theta}) d\boldsymbol{\theta} = \frac{1}{N_s} \sum_{n=1}^{N_s} g(\boldsymbol{\theta}^{(n)})$
Sequence Modelling	$\boldsymbol{x}_n = f(\boldsymbol{x}_{n-1}, \boldsymbol{\theta})$

What is Deep Learning?

Deep learning is primarily characterised by function compositions:

What is Deep Learning?

Deep learning is primarily characterised by function compositions:

- Feedforward networks: $\boldsymbol{y} = f(g(\boldsymbol{x}, \boldsymbol{\theta}_g), \boldsymbol{\theta_f})$
  - Often with relatively simple functions (e.g. $f(\boldsymbol{x}, \boldsymbol{\theta}_f) = \sigma(\boldsymbol{x}^\top \boldsymbol{\theta}_f)$)

What is Deep Learning?

Deep learning is primarily characterised by function compositions:

- Feedforward networks: $\boldsymbol{y} = f(g(\boldsymbol{x}, \boldsymbol{\theta}_g), \boldsymbol{\theta_f})$
  - Often with relatively simple functions (e.g. $f(\boldsymbol{x}, \boldsymbol{\theta}_f) = \sigma(\boldsymbol{x}^\top \boldsymbol{\theta}_f)$)
- Recurrent networks:
  $\boldsymbol{y}_t = f(\boldsymbol{y}_{t-1}, \boldsymbol{x}_t, \boldsymbol{\theta}) = f(f(\boldsymbol{y}_{t-2}, \boldsymbol{x}_{t-1}, \boldsymbol{\theta}), \boldsymbol{\theta}) = \dots$

What is Deep Learning?

Deep learning is primarily characterised by function compositions:

- Feedforward networks: $\boldsymbol{y} = f(g(\boldsymbol{x}, \boldsymbol{\theta}_g), \boldsymbol{\theta}_{\boldsymbol{f}})$
    - Often with relatively simple functions (e.g. $f(\boldsymbol{x}, \boldsymbol{\theta}_f) = \sigma(\boldsymbol{x}^\top \boldsymbol{\theta}_f)$)
- Recurrent networks:
  $\boldsymbol{y}_t = f(\boldsymbol{y}_{t-1}, \boldsymbol{x}_t, \boldsymbol{\theta}) = f(f(\boldsymbol{y}_{t-2}, \boldsymbol{x}_{t-1}, \boldsymbol{\theta}), \boldsymbol{\theta}) = \dots$

In the early days the focus of deep learning was on learning functions for classification. Nowadays the functions are much more general in their inputs and outputs.

What is Differentiable Programming?

- Differentiable programming is a term coined by Yann Lecun[1] to describe a superset of Deep Learning.

[1] https://www.facebook.com/yann.lecun/posts/10155003011462143
[2] See our ICLR 2019 paper: https://arxiv.org/abs/1812.03928

What is Differentiable Programming?

- Differentiable programming is a term coined by Yann Lecun[1] to describe a superset of Deep Learning.
- Captures the idea that computer programs can be constructed of parameterised functional blocks in which the parameters are learned using some form of gradient-based optimisation.

[1] https://www.facebook.com/yann.lecun/posts/10155003011462143
[2] See our ICLR 2019 paper: https://arxiv.org/abs/1812.03928

What is Differentiable Programming?

- Differentiable programming is a term coined by Yann Lecun[1] to describe a superset of Deep Learning.
- Captures the idea that computer programs can be constructed of parameterised functional blocks in which the parameters are learned using some form of gradient-based optimisation.
  - The implication is that we need to be able to compute gradients with respect to the parameters of these functional blocks. We'll start explore this in detail next week...

[1] https://www.facebook.com/yann.lecun/posts/10155003011462143

[2] See our ICLR 2019 paper: https://arxiv.org/abs/1812.03928

What is Differentiable Programming?

- Differentiable programming is a term coined by Yann Lecun[1] to describe a superset of Deep Learning.
- Captures the idea that computer programs can be constructed of parameterised functional blocks in which the parameters are learned using some form of gradient-based optimisation.
  - The implication is that we need to be able to compute gradients with respect to the parameters of these functional blocks. We'll start explore this in detail next week...
  - The idea of Differentiable Programming also opens up interesting possibilities:
    - The functional blocks don't need to be direct functions in a mathematical sense; more generally they can be *algorithms*.
    - What if the functional block we're learning parameters for is itself an algorithm that optimises the parameters of an internal algorithm using a gradient based optimiser?![2]

[1] https://www.facebook.com/yann.lecun/posts/10155003011462143

[2] See our ICLR 2019 paper: https://arxiv.org/abs/1812.03928

Is all Deep Learning Differentiable Programming?

- Not necessarily!
  - Most deep learning systems are trained using first order gradient-based optimisers, but there is an active body of research on gradient-free methods.

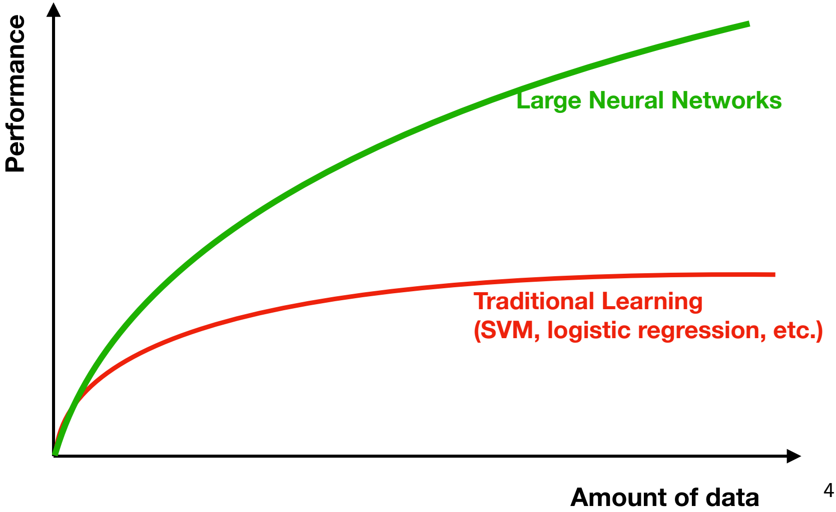
[3] including at least myself, my PhD students and Geoff Hinton!

Is all Deep Learning Differentiable Programming?

- Not necessarily!
  - Most deep learning systems are trained using first order gradient-based optimisers, but there is an active body of research on gradient-free methods.
  - There is an increasing interest in methods that use different styles of learning, such as Hebbian learning, within deep networks. More broadly there are a number of us[3] who are interested in biologically motivated models and learning methods.

[3]including at least myself, my PhD students and Geoff Hinton!

Why should we care about this?



[4]

[4] Reference: Andrew Ng

Where did it all start & what was the motivation?

What is the objective of this module?

1. To gain an in-depth theoretical and practical understanding of modern deep neural networks and their applications.
2. Understand the underlying mathematical and algorithmic principles of deep learning
3. Understand the key factors that have made deep learning successful for various applications
4. Apply existing deep learning models to real datasets
5. Gain facility in working with deep learning libraries in order to create and evaluate network architectures
6. Critically appraise the merits and shortcomings of model architectures on specific problems

What will we cover in the module?

http://comp6248.ecs.soton.ac.uk/

How is this module going to be delivered?

1. Reading material before the lectures

Lab session plan

Lab	Date	Topic
Lab 1	05/02/19	Introducing PyTorch
Lab 2	12/02/19	Automatic Differentiation
Lab 3	19/02/19	Optimisation
Lab 4	26/02/19	NNs with PyTorch and Torchbearer
Lab 5	05/03/19	CNNs with PyTorch and Torchbearer
Lab 6	12/03/19	Transfer Learning
Lab 7	19/03/19	RNNs, Sequence Prediction and Embeddings
Lab 8	26/03/19	Deep Generative Models
	Break	
Lab 9	30/04/19	Coursework Help and Advice
Lab 10	30/04/19	Coursework Help and Advice
Lab 11	30/04/19	Coursework Help and Advice

What do we expect you already know?

1. COMP3206 or COMP3223 or COMP6229 or COMP6245
2. Fundamentals of Linear Algebra, Probability and Statistics, Calculus
3. Programming in Python

Assessment Structure

1. Lab work 40%
2. Final project 40%
3. In-class tests 20%

Assessment Timetable

Assessment	Date	Time
Labs 1-3	22/02/19	16:00
Coursework Team Information	27/02/19	16:00
In - class Test 1	01/03/19	midnight
Labs 4-6	15/03/19	16:00
In - class Test 2	29/03/19	midnight
Labs 7-8	03/05/19	16:00
Final Coursework Submission	15/05/19	16:00

The Main Assignment
The ICLR Reproducibility Challenge

http://comp6248.ecs.soton.ac.uk/coursework.html