COMP6248 Differentiable Programming

(and some Deep Learning)

Jonathon Hare

Vision, Learning and Control
University of Southampton

Machine Learning - A Recap

All credit for this slide goes to Niranjan

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

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	$E_0 = \sum_{n=1}^N \{\lVert \boldsymbol{y}_n - f(\boldsymbol{x}_n; \boldsymbol{\theta}) \rVert\}^2$
Prediction	$\hat{\boldsymbol{y}}_{N+1} = f(\boldsymbol{x}_{N+1}, \hat{\boldsymbol{\theta}})$
Regularisation	$E_1 = \sum_{n=1}^N \{\lVert \boldsymbol{y}_n - f(\boldsymbol{x}_n; \boldsymbol{\theta}) \rVert\}^2 + r(\lVert \boldsymbol{\theta} \rVert)$

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	$p(\boldsymbol{\theta} \mid \{\boldsymbol{x}_n, \boldsymbol{y}_n\}_{n=1}^N)$

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	$E_0 = \sum_{n=1}^N \{\lVert \boldsymbol{y}_n - f(\boldsymbol{x}_n; \boldsymbol{\theta}) \rVert\}^2$
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	$p(\boldsymbol{\theta} \mid \{\boldsymbol{x}_n, \boldsymbol{y}_n\}_{n=1}^N)$
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	$E_0 = \sum_{n=1}^N \{\lVert \boldsymbol{y}_n - f(\boldsymbol{x}_n; \boldsymbol{\theta}) \rVert\}^2$
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	$p(\boldsymbol{\theta} \mid \{\boldsymbol{x}_n, \boldsymbol{y}_n\}_{n=1}^N)$
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: $\boldsymbol{y} = f(g(\boldsymbol{x}, \boldsymbol{\theta}_g), \boldsymbol{\theta}_{\boldsymbol{f}})$
  - Often with relatively simple functions (e.g. $f(\boldsymbol{x}, \boldsymbol{\theta}_f) = \sigma(\boldsymbol{x}^\top \boldsymbol{\theta}_f)$)
- Recurrent networks:
  $\boldsymbol{y}_t = f(\boldsymbol{y}_{t-1}, \boldsymbol{x}_t, \boldsymbol{\theta}) = f(f(\boldsymbol{y}_{t-2}, \boldsymbol{x}_{t-1}, \boldsymbol{\theta}), \boldsymbol{x}_t, \boldsymbol{\theta}) = \ldots$

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:
  $\boldsymbol{y}_t = f(\boldsymbol{y}_{t-1}, \boldsymbol{x}_t, \boldsymbol{\theta}) = f(f(\boldsymbol{y}_{t-2}, \boldsymbol{x}_{t-1}, \boldsymbol{\theta}), \boldsymbol{x}_t, \boldsymbol{\theta}) = \dots$

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 and NeurIPS 2019 paper: https://arxiv.org/abs/1906.06565

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 and NeurIPS 2019 paper: https://arxiv.org/abs/1906.06565

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 and NeurIPS 2019 paper: https://arxiv.org/abs/1906.06565

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 and NeurIPS 2019 paper: https://arxiv.org/abs/1906.06565

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.

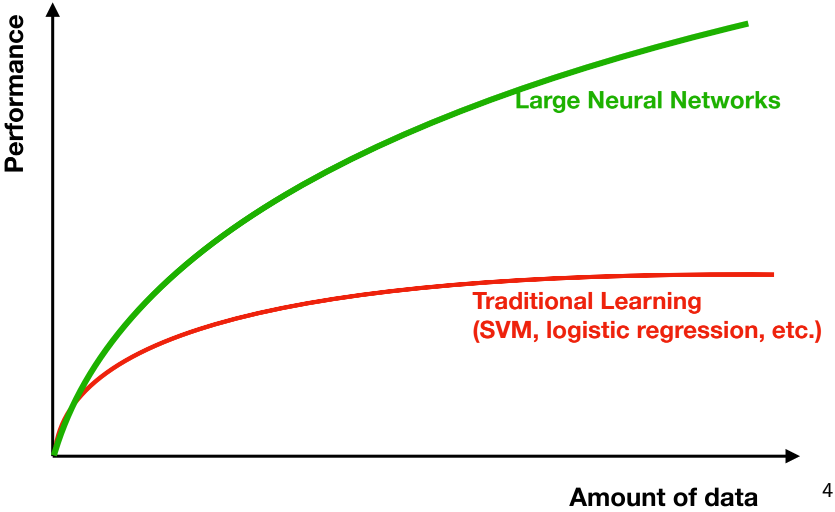
[3] including at least myself, my PhD students and Geoff Hinton!

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.
  - This course will primarily focus on differentiable methods, but we'll look at how relaxations can be made to make non-differentiable operators learnable with gradient-based optimisers.

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

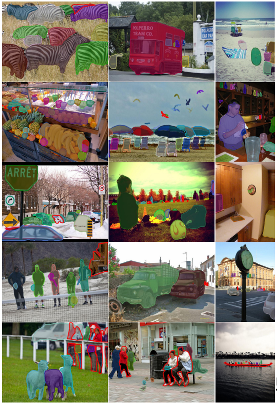
Why should we care about this?



[4]

[4] Reference: Andrew Ng

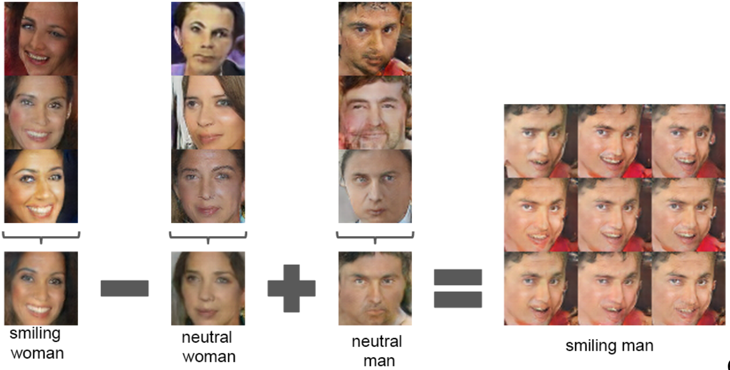
Success stories - Object detection and segmentation



[5]

[5] Pinheiro, Pedro O., et al. "Learning to refine object segments." European Conference on Computer Vision. Springer, Cham, 2016.

Success stories - Image generation



[6]

[6] Radford, Alec, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks." arXiv preprint arXiv:1511.06434 (2015).

Success stories - Translation

- ENGLISH TEXT
- The reason Boeing are doing this is to cram more seats in to make their plane more competitive with our products," said Kevin Keniston, head of passenger comfort at Europe's Airbus.
- TRANSLATED TO FRENCH
- La raison pour laquelle Boeing fait cela est de creer plus de sieges pour rendre son avion plus competitif avec nos produits", a declare Kevin Keniston, chef du confort des passagers chez Airbus. [7]

[7] Wu, Yonghui, et al. "Google's neural machine translation system: Bridging the gap between human and machine translation." arXiv preprint arXiv:1609.08144 (2016).

What is the objective of this module?

A word of warning: This is not a module about how to apply someone else's deep network architecture to a task, or how to train existing models!

You will learn some of that along the way of course, but the real objective is for you to graduate knowing how to understand, critique and implement new and recent research papers on deep learning and associated topics.

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.

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.

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 differentiable programming successful for various applications.

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 differentiable programming successful for various applications.
4. Apply existing deep learning models to real datasets.

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 differentiable programming 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.

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 differentiable programming 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. Lectures
    1. Note: I am refreshing some material from last year, but the website still has links to the old slides.
    2. You need to read the suggested papers/links before the lectures!
2. Labs

Lab session plan

Lab	Date	Topic
Lab 1	05/02/20	Introducing PyTorch
Lab 2	12/02/20	Automatic Differentiation
Lab 3	19/02/20	Optimisation
Lab 4	26/02/20	NNs with PyTorch and Torchbearer
	04/03/20	*NO LAB*
Lab 5	11/03/20	CNNs with PyTorch and Torchbearer
Lab 6	18/03/20	Transfer Learning
Lab 7	18/03/20	RNNs, Sequence Prediction and Embeddings
	Break	
Lab 7	22/04/20	RNNs, Sequence Prediction and Embeddings
Lab 8	29/04/19	Deep Generative Models
	06/05/19	Coursework Help and Advice
	13/05/19	Coursework Help and Advice

What do we expect you already know?

1. COMP3223 or COMP6245 (fundamentals of statistical learning, MLPs, gradient descent, how to train and evaluate learning machines, supervised-vs-unsupervised)

What do we expect you already know?

1. COMP3223 or COMP6245 (fundamentals of statistical learning, MLPs, gradient descent, how to train and evaluate learning machines, supervised-vs-unsupervised)
2. Fundamentals of Matrix Algebra (matrix-matrix, matrix-vector and matrix-scalar operations, inverse, determinant, rank, Eigendecomposition, SVD), Probability & Statistics (1st-order summary statistics, simple continous and discrete probability distributions, expected values, etc) and Multivariable Calculus (partial differentiation, chain-rule)

What do we expect you already know?

1. COMP3223 or COMP6245 (fundamentals of statistical learning, MLPs, gradient descent, how to train and evaluate learning machines, supervised-vs-unsupervised)
2. Fundamentals of Matrix Algebra (matrix-matrix, matrix-vector and matrix-scalar operations, inverse, determinant, rank, Eigendecomposition, SVD), Probability & Statistics (1st-order summary statistics, simple continous and discrete probability distributions, expected values, etc) and Multivariable Calculus (partial differentiation, chain-rule)
3. Programming in Python

What might you already know?

1. How to use a deep learning framework (Keras, Tensorflow, PyTorch)?

What might you already know?

1. How to use a deep learning framework (Keras, Tensorflow, PyTorch)?
2. How to train an existing model architecture using a GPU?

What might you already know?

1. How to use a deep learning framework (Keras, Tensorflow, PyTorch)?
2. How to train an existing model architecture using a GPU?
3. How to perform transfer learning?

What might you already know?

1. How to use a deep learning framework (Keras, Tensorflow, PyTorch)?
2. How to train an existing model architecture using a GPU?
3. How to perform transfer learning?
4. How to perform differentiable sampling of a Multivariate Normal Distribution?

Assessment Structure

1. Lab work 40%
2. Final project 40% - Handin in Week 12 (15th May, 4PM)
3. Online quizzes 20% - Planned for Thursdays in Weeks 6 (5th-Mar) and 10 (30-Apr)

The Main Assignment

The ICLR Reproducibility Challenge

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