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	$\{\boldsymbol{x}_n, \boldsymbol{y}_n\}_{n=1}^{N}$	$\{\boldsymbol{x}_n\}_{n=1}^{N}$
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	$E_0 = \sum_{n=1}^{N} \{\|\boldsymbol{y}_n - f(\boldsymbol{x}_n; \boldsymbol{\theta})\|\}^2$
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	$p(\boldsymbol{\theta} \vert \{\boldsymbol{x}_n, \boldsymbol{y}_n\}_{n=1}^N)$

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	$E_0 = \sum_{n=1}^{N} \{\lVert \boldsymbol{y}_n - f(\boldsymbol{x}_n; \boldsymbol{\theta}) \rVert\}^2$
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	$p(\boldsymbol{\theta} \vert \{\boldsymbol{x}_n, \boldsymbol{y}_n\}_{n=1}^{N})$
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	$p(\boldsymbol{\theta} \vert \{\boldsymbol{x}_n, \boldsymbol{y}_n\}_{n=1}^N)$
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:
  $\boldsymbol{y}_t = f(\boldsymbol{y}_{t-1}, \boldsymbol{x}_t, \boldsymbol{\theta}) = f(f(\boldsymbol{y}_{t-2}, \boldsymbol{x}_{t-1}, \boldsymbol{\theta}), \boldsymbol{\theta}) = \ldots$

What is Deep Learning?

Deep learning is primarily characterised by function compositions:

- Feedforward networks: $\boldsymbol{y} = f(g(\boldsymbol{x}, \boldsymbol{\theta}_g), \boldsymbol{\theta}_{\boldsymbol{f}})$
  - 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.

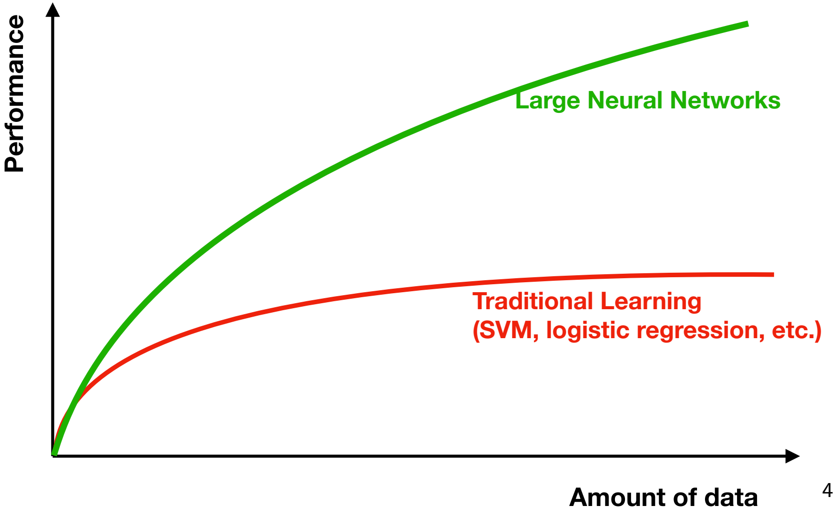
[3] including at least myself, my PhD students and Geoff Hinton!

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!

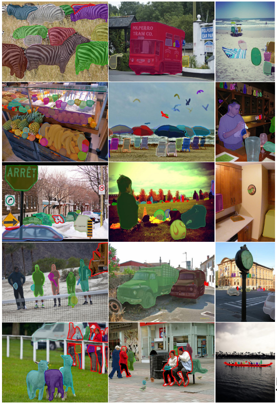
Why should we care about this?



[4]

[4] Reference: Andrew Ng

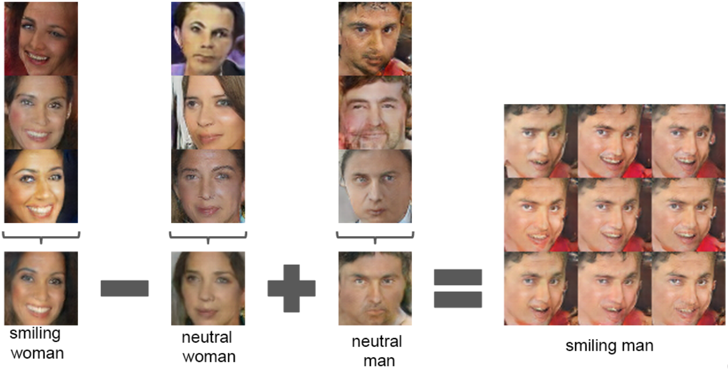
Success stories - Object detection and segmentation



[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).