A very brief overview of deep learning

Maarten Grachten

Austrian Research Institute for Artificial Intelligence
`http://www.ofai.at/research/impml`

O F A I

Learning
to Create

SEVENTH FRAMEWORK PROGRAMME

Co-funded by the FP7 Programme of the EUROPEAN UNION

Table of contents

Deep learning

There is no single **definition** of deep learning, but most definitions emphasize:

- Branch of **machine learning**
- Models are graph structures (**networks**) with **multiple layers** (**deep**)
- Models are typically **non-linear**
- Both **supervised** and **unsupervised** methods are used for fitting models to data

Deep learning

There is no single **definition** of deep learning, but most definitions emphasize:

- Branch of **machine learning**
- Models are graph structures (**networks**) with **multiple layers** (**deep**)
- Models are typically **non-linear**
- Both **supervised** and **unsupervised** methods are used for fitting models to data

Deep learning

There is no single **definition** of deep learning, but most definitions emphasize:

- Branch of **machine learning**
- Models are graph structures (**networks**) with **multiple layers** (**deep**)
- Models are typically **non-linear**
- Both **supervised** and **unsupervised** methods are used for fitting models to data

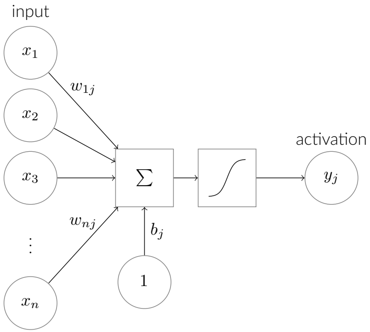
Deep learning

There is no single **definition** of deep learning, but most definitions emphasize:

- Branch of **machine learning**
- Models are graph structures (**networks**) with **multiple layers** (**deep**)
- Models are typically **non-linear**
- Both **supervised** and **unsupervised** methods are used for fitting models to data

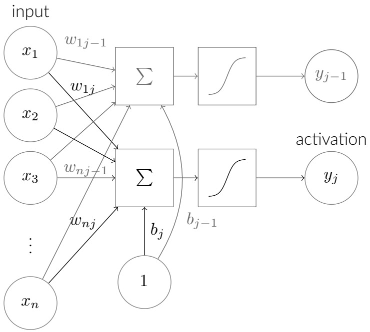
An example of deep models: Deep neural networks

- A ***neuron*** is a **non-linear transformation** of a **linear sum** of inputs: $y = f(\mathbf{w}^T\mathbf{x} + b)$
- An **array of neurons** taking the same input $\mathbf{x}$ form a new ***layer*** on top of the input in a neural network: $\mathbf{y} = f(\mathbf{W}^T\mathbf{x} + \mathbf{b})$
- Third layer: $\mathbf{y_2} = f(\mathbf{W}_2^T f(\mathbf{W}_1^T\mathbf{x} + \mathbf{b}_1) + \mathbf{b}_2)$



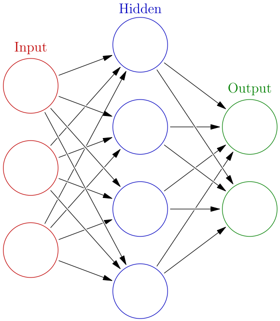
An example of deep models: Deep neural networks

- A ***neuron*** is a **non-linear transformation** of a **linear sum** of inputs: $y = f(\mathbf{w}^T\mathbf{x} + b)$
- An **array of neurons** taking the same input $\mathbf{x}$ form a new ***layer*** on top of the input in a neural network: $\mathbf{y} = f(\mathbf{W}^T\mathbf{x} + \mathbf{b})$
- Third layer: $\mathbf{y}_2 = f(\mathbf{W}_2^T f(\mathbf{W}_1^T\mathbf{x} + \mathbf{b}_1) + \mathbf{b}_2)$



An example of deep models: Deep neural networks

- A ***neuron*** is a **non-linear transformation** of a **linear sum** of inputs: $y = f(\mathbf{w}^T\mathbf{x} + b)$
- An **array of neurons** taking the same input $\mathbf{x}$ form a new ***layer*** on top of the input in a neural network: $\mathbf{y} = f(\mathbf{W}^T\mathbf{x} + \mathbf{b})$
- Third layer: $\mathbf{y}_2 = f(\mathbf{W}_2^T f(\mathbf{W}_1^T\mathbf{x} + \mathbf{b}_1) + \mathbf{b}_2)$



Relation to other machine learning approaches

How is deep learning different from 80's NN research?

- Training methods derived from **probabilistic interpretation** of networks as **generative models**
- Greedy **layer-wise** training
- More powerful **optimization methods**
- More **computing power**, larger **data sets**

Feature design vs. feature learning

- The success of most machine learning approaches critically depends on **appropriately designed** features
- Deep learning reduces need for manual feature design:
  - Models **learn features** as non-linear transformations of data
  - Deep models learn **hierarchies of features**
  - Unsupervised (pre) training **prevents overfitting**

Relation to other machine learning approaches

How is deep learning different from 80's NN research?

- Training methods derived from **probabilistic interpretation** of networks as **generative models**
- Greedy **layer-wise** training
- More powerful **optimization methods**
- More **computing power**, larger **data sets**

Feature design vs. feature learning

- The success of most machine learning approaches critically depends on **appropriately designed** features
- Deep learning reduces need for manual feature design:
  - Models **learn features** as non-linear transformations of data
  - Deep models learn **hierarchies of features**
  - Unsupervised (pre) training **prevents overfitting**

What are deep models used for?

Tasks

- **Prediction** classification, regression problems
  - Prediction as part of model (output layer, input layer)
  - Use model to obtain feature vectors for data, use any classifier for prediction (WEKA)
- **Generation** e.g. facial expressions, gait, music
  - Denoising of data
  - Reconstruction/completion of partial data
  - Generation of new data by sampling

Successful application domains

- **Image**: object recognition, optical character recognition
- **Audio**: speech recognition, music retrieval, transcription
- **Text**: parsing, sentiment analysis, machine translation

What are deep models used for?

Tasks

- **Prediction** classification, regression problems
  - Prediction as part of model (output layer, input layer)
  - Use model to obtain feature vectors for data, use any classifier for prediction (WEKA)
- **Generation** e.g. facial expressions, gait, music
  - Denoising of data
  - Reconstruction/completion of partial data
  - Generation of new data by sampling

Successful application domains

- **Image**: object recognition, optical character recognition
- **Audio**: speech recognition, music retrieval, transcription
- **Text**: parsing, sentiment analysis, machine translation

What are deep models used for?

Tasks

- **Prediction** classification, regression problems
  - Prediction as part of model (output layer, input layer)
  - Use model to obtain feature vectors for data, use any classifier for prediction (WEKA)
- **Generation** e.g. facial expressions, gait, music
  - Denoising of data
  - Reconstruction/completion of partial data
  - Generation of new data by sampling

Successful application domains

- **Image**: object recognition, optical character recognition
- **Audio**: speech recognition, music retrieval, transcription
- **Text**: parsing, sentiment analysis, machine translation

Traditional learning in neural networks: Backpropagation

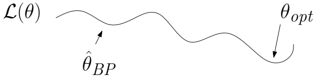
- Given data D, define a **loss function** on targets and actual output: $\mathcal{L}_D(\theta)$, for example:
  - **summed squared error** between output and targets
  - **cross-entropy** between output and targets
- Use **gradient descent** to iteratively find better weights θ
  - Compute the gradient of $\mathcal{L}$ with respect to θ, either:
    - **Batch** gradient: $\nabla \mathcal{L}_D(\theta)$
    - **Stochastic** gradient: $\nabla \mathcal{L}_d(\theta)$ for $d \in D$
  - Update $w \in \theta$ by subtracting $\alpha \frac{\partial \mathcal{L}}{\partial w}$ (α: learning rate)
  - Continue descent until some **stopping criterion**, e.g.:
    - **Convergence** of θ
    - **Early stopping** (stop when error on validation set starts to increase)

Limitations of backpropagation (BP)

- Does not scale well to deep networks (including recurrent networks): **gradients** further away from the outputs tend to either **vanish** or **explode** [Hochreiter and Schmidhuber, 1997]
- Likely to settle at (poor) **local minima** of the loss function
- Since BP used to be the state-of-the-art training algorithm: **limited success** with **deep neural networks**

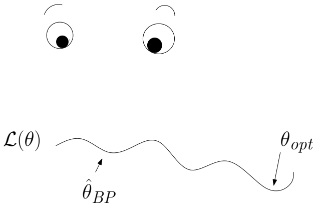
Limitations of backpropagation (BP)

- Does not scale well to deep networks (including recurrent networks): **gradients** further away from the outputs tend to either **vanish** or **explode** [Hochreiter and Schmidhuber, 1997]
- Likely to settle at (poor) **local minima** of the loss function
- Since BP used to be the state-of-the-art training algorithm: **limited success** with **deep neural networks**



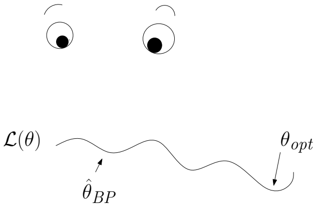
Limitations of backpropagation (BP)

- Does not scale well to deep networks (including recurrent networks): **gradients** further away from the outputs tend to either **vanish** or **explode** [Hochreiter and Schmidhuber, 1997]
- Likely to settle at (poor) **local minima** of the loss function
- Since BP used to be the state-of-the-art training algorithm: **limited success** with **deep neural networks**



Limitations of backpropagation (BP)

- Does not scale well to deep networks (including recurrent networks): **gradients** further away from the outputs tend to either **vanish** or **explode** [Hochreiter and Schmidhuber, 1997]
- Likely to settle at (poor) **local minima** of the loss function
- Since BP used to be the state-of-the-art training algorithm: **limited success** with **deep neural networks**



Modern approaches to train deep neural networks

- Long short term memory [Hochreiter and Schmidhuber, 1997]
  - Specialized recurrent structure + gradient descent to explicitly **preserve error gradients** over long distances
- Training networks by second order optimization
  - **Hessian-free** training [Martens, 2010]
- Greedy layer-wise training [Hinton et al., 2006]
  - Train layers individually, supervised or unsupervised
  - Higher layers take as input the output from lower layers
  - Layers often trained as **Restricted Boltzmann Machines** or **Autoencoders**
- Data-specific models and training
  - **Convolutional** neural networks [Lecun et al., 1998]
- Dropout [Hinton et al., 2012]
  - Randomly **ignore hidden units** during training
  - Avoids overfitting

Topics covered in following talks

- Recurrent Neural Networks
  - Beat-tracking with LSTM (Sebastian Böck)
  - Hessian-free training (Carlos Cancino)
- (Stacked) Autoencoders
  - Learning binary codes for fast music retrieval (Jan Schlüter)
- (Stacked) Restricted Boltzmann Machines
  - Speech/Music classification (Jan Schlüter)
  - Learning tonal structure from melodies (Carlos Cancino)
- Convolutional Neural Networks and dropout
  - Onset detection / Audio segmentation (Jan Schlüter)
  - High-dimensional aspects of CNNs (Karen Ullrich)

References I

Hinton, G. E., Osindero, S., and Teh, Y. (2006).
A fast learning algorithm for deep belief nets.
Neural Computation, 18:1527–1554.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2012).
Improving neural networks by preventing co-adaptation of feature detectors.
CoRR, abs/1207.0580.

Hochreiter, S. and Schmidhuber, J. (1997).
Long Short-Term Memory.
Neural Computation, 9(8):1735–1780.

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998).
Gradient-based learning applied to document recognition.
In *Proceedings of the IEEE*, pages 2278–2324.

Martens, J. (2010).
Deep learning via hessian-free optimization.
In *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*, pages 735–742.