An introduction to artificial neural networks and how to use them

Christopher Albert

Max-Planck-Institut für Plasmaphysik, 85748 Garching

NMPP Seminar, 25.09.2018

e-mail: calbert@ipp.mpg.de

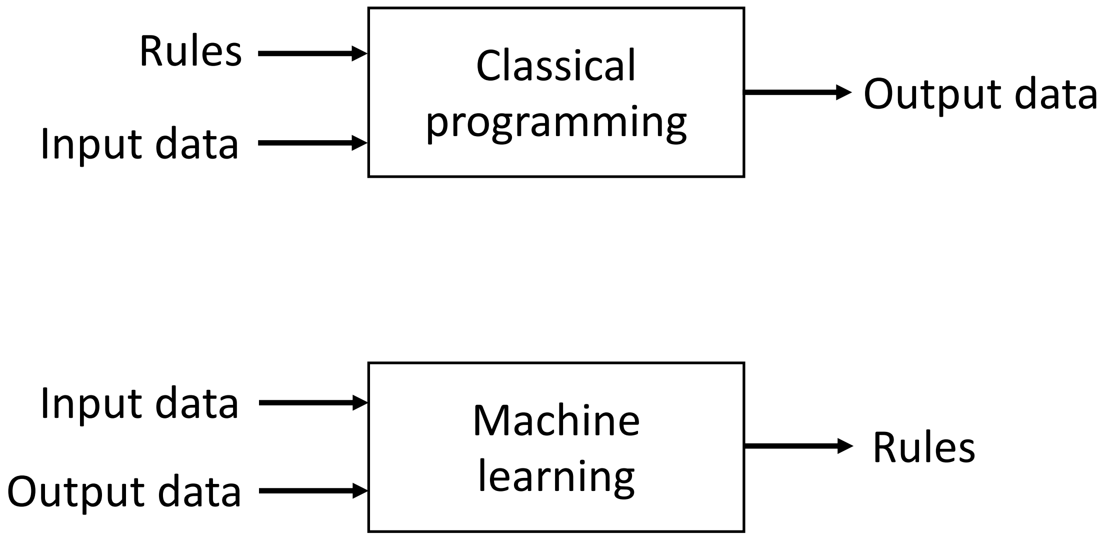
Main sources

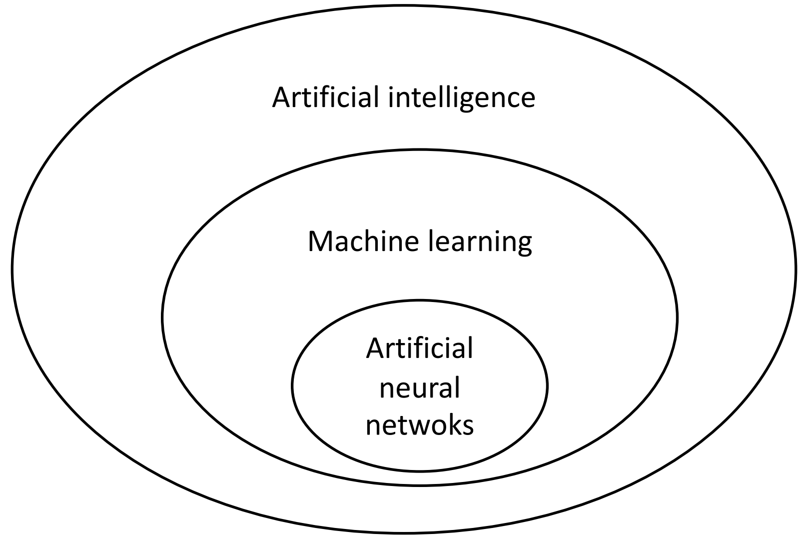
[1] François Chollet. *Deep learning with Python* (Pearson, 2017) (developer of Keras)

- *Machine learning for artists* https://ml4a.github.io/
- Keras blog https://blog.keras.io/
- Kaggle (datasets, algorithms, challenges) https://www.kaggle.com/

What is machine learning?

- Machine learning algorithms find rules relating input and output data

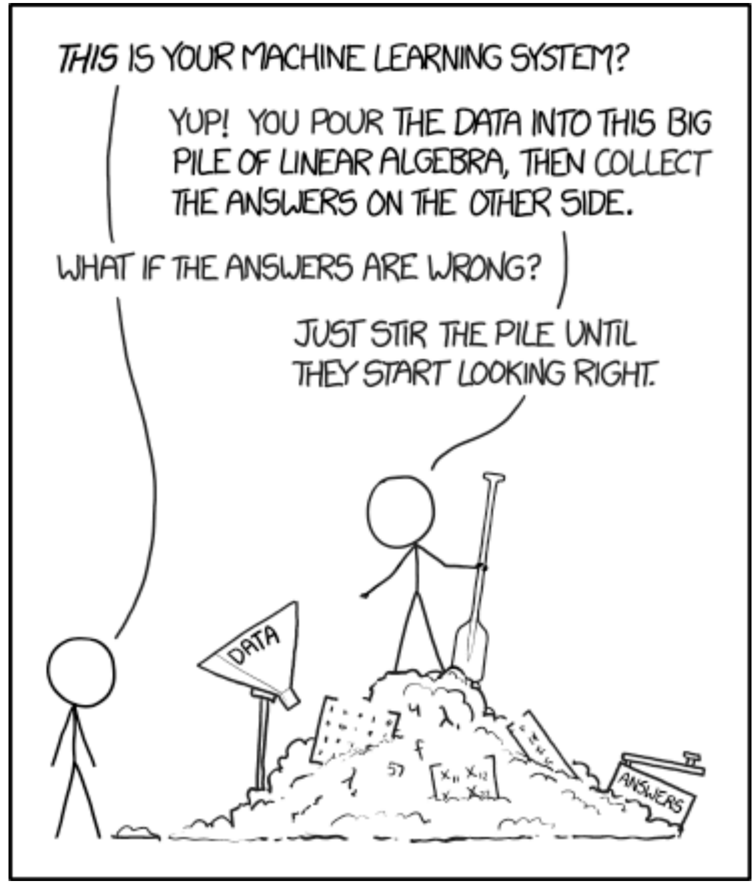




Relation of Machine learning to programming, AI and artificial neural networks (after [1])

Types of machine learning

- Supervised learning
  - Training with correct input/output pairs
  - Classification
  - Regression / interpolation
- Unsupervised learning:
  machine is “on its own“
  - Correct output and rules unknown
  - Clustering
- Mixed forms exist



source: https://xkcd.com/1838/, license: https://xkcd.com/license.html

Some machine learning techniques

- Classical interpolation and regression
  - Linear regression, splines
- Stochastic methods including uncertainty
  - Polynomial chaos expansion, Gaussian processes
- Decision trees
  - Random forest, gradient boosting machine
- Clustering methods
  - k-Means, EM algorithm
- Artificial neural networks
  - Convolutional networks (recognition), LTSM (time series prediction), autoencoder (dimensionality reduction), and many more

Rule of thumb: the more flexible a method, the more difficult it is to interpret its parameters.

Classification and regression

- Take map F of *input* $u \in U$ to *output* $v \in V$

$$F: U \to V$$
$$u \mapsto v = F(u)$$

- Examples:
  - Simulation with input parameters $u \mapsto$ result v
  - Pixels $u \in \{1 \dots 16\}^{L \cdot W}$ of an image $\mapsto$ digits $v \in \{0 \dots 9\}$
  - Microphone signal $u(t) \mapsto$ words $v \in$ dictionary

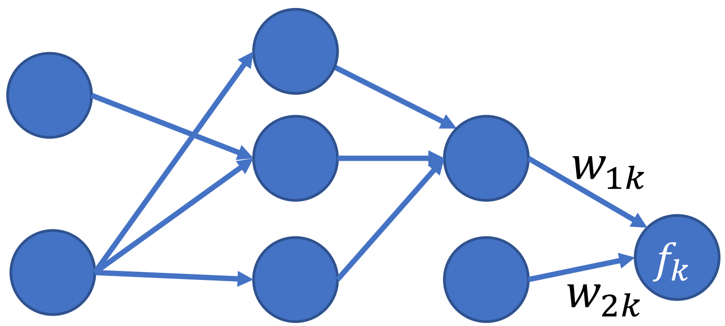
- Machine learning: find an approximate map $\tilde{F} \approx F$

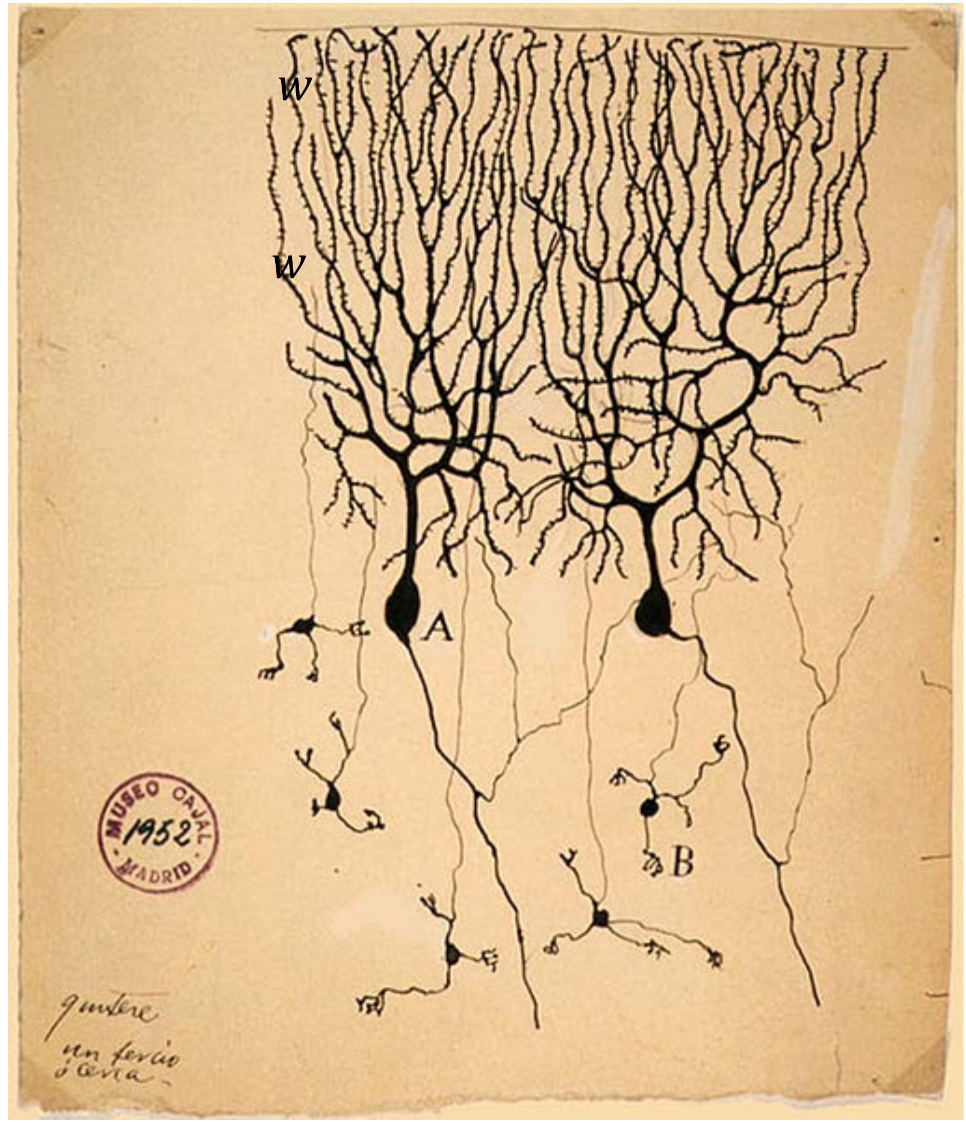
Classification and regression

- Machine learning: find an approximate map $\tilde{F}(u; w) \approx F(u)$
  - Depends on parameters w (e.g. spline coefficients)
  - Should be *fast* to evaluate
  - Should provide *insight* into features
- Training (parameter estimation / fit)
  - Based on given *training data* $\{(u_T, F(u_T))\}$
  - *Minimize* $||g(\tilde{F}(u_T, w) - F(u_T), w)||$ w.r.t. w for some norm $||\cdot||$
  - *Loss function* g may include weight-based regularisation to avoid overfitting
- Validation
  - Check correctness on *test data* $u_V \neq u_T$
  - Measures of goodness: $||g||$ or classification accuracy in %

Artificial neural network

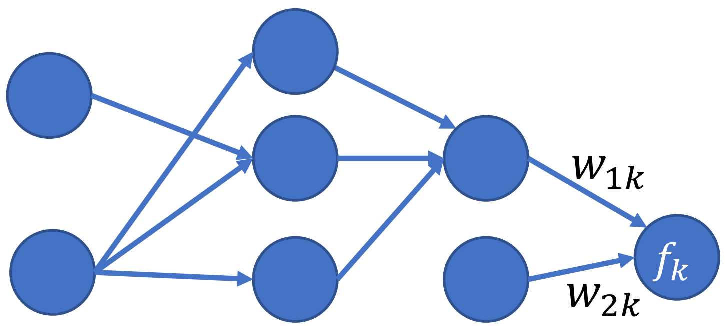
- Neural network = graph
  - Neurons = nodes
  - Dendrites = edges towards nodes
  - Activation function f
  - Weights w

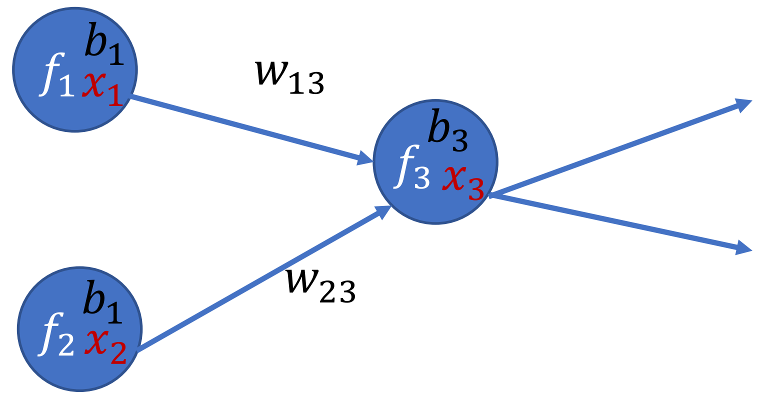




Artificial neural network

- Neural network = directed acyclic graph with summation rules
  - Neurons = nodes that sum input
  - Dendrites = edges carrying weights for node input summation
  - Activation function f_k = transformation to add nonlinearity (prescribed)
  - Weights w_{kl} and bias b_l = (statistical) model parameters

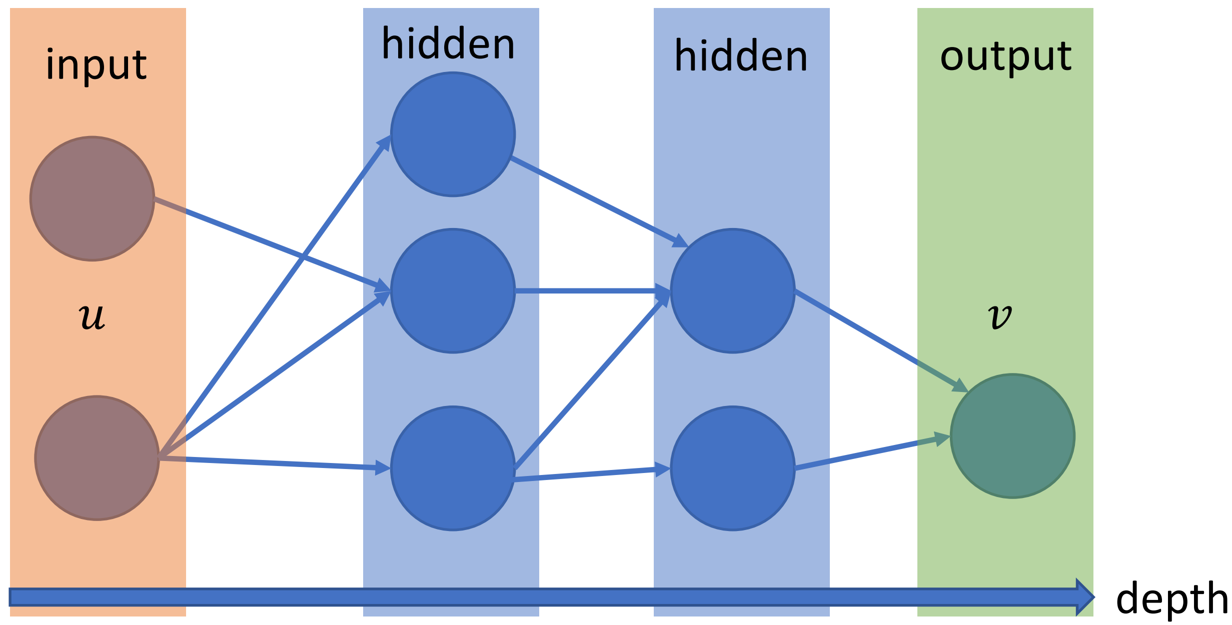




Summation at each node, e.g. $x_3 = f_3(b_3 + x_1 w_{13} + x_2 w_{23})$

Layers and overall map

Layers:



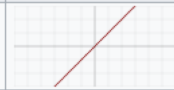
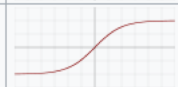
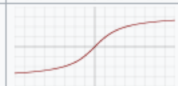
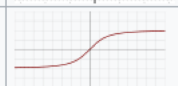
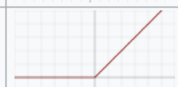
- Map generated by the network, including bias $\bar{f}_i(x) = f_i(b_i + x)$:

$$v_k = \tilde{F}_k(u; w) = \bar{f}_k \circ w_{kj} \bar{f}_j \circ \cdots \circ w_{ab} \bar{f}_b \circ w_{ba} u_a$$

Example

- Let’s have a look on https://ml4a.github.io/ml4a/neural_networks/#regression

Different activation functions

Name	Plot	Equation	Derivative (with respect to x)	Range
Identity		$f(x) = x$	$f'(x) = 1$	$(-\infty, \infty)$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$	$\{0, 1\}$
Logistic (a.k.a. Sigmoid or Soft step)		$f(x) = \sigma(x) = \frac{1}{1 + e^{-x}}$ [1]	$f'(x) = f(x)(1 - f(x))$	$(0, 1)$
TanH		$f(x) = \tanh(x) = \frac{(e^x - e^{-x})}{(e^x + e^{-x})}$	$f'(x) = 1 - f(x)^2$	$(-1, 1)$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2 + 1}$	$\left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$
Softsign[7][8]		$f(x) = \frac{x}{1 + \|x\|}$	$f'(x) = \frac{1}{(1 + \|x\|)^2}$	$(-1, 1)$
Inverse square root unit (ISRU)[9]		$f(x) = \frac{x}{\sqrt{1 + \alpha x^2}}$	$f'(x) = \left(\frac{1}{\sqrt{1 + \alpha x^2}}\right)^3$	$\left(-\frac{1}{\sqrt{\alpha}}, \frac{1}{\sqrt{\alpha}}\right)$
Rectified linear unit (ReLU)[10]		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$[0, \infty)$
Leaky rectified linear unit (Leaky ReLU)[11]		$f(x) = \begin{cases} 0.01x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0.01 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$(-\infty, \infty)$
Parameteric rectified linear unit (PReLU)[12]		$f(\alpha, x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(\alpha, x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$(-\infty, \infty)$[2]
Randomized leaky rectified linear unit (RReLU)[13]		$f(\alpha, x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$ [3]	$f'(\alpha, x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$(-\infty, \infty)$
Exponential linear unit (ELU)[14]		$f(\alpha, x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(\alpha, x) = \begin{cases} f(\alpha, x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$(-\alpha, \infty)$

Are artificial neural networks intelligent?

- Artificial neural networks are *not* intelligent per se.

- Shared features with the brain:
  - Neurons and dendrites
  - Emergence: complex behavior based on simple constituents
  - Convolutional networks model the way we think the brain filters information

- How the brain is different:
  - Bigger. Human brain: 10s of billions of neurons with 1000s of inputs for each.
  - More complex structure. Brain has feedback loops, ANNs usually feed-forward.

Where to use artificial neural networks?

- Why Google, Facebook, Apple (Big Data) love neural networks
  - Have huge amounts of unstructured data for free
  - Need fast evaluation (mobile devices, real-time search, etc.)
  - Can do training in big datacenters

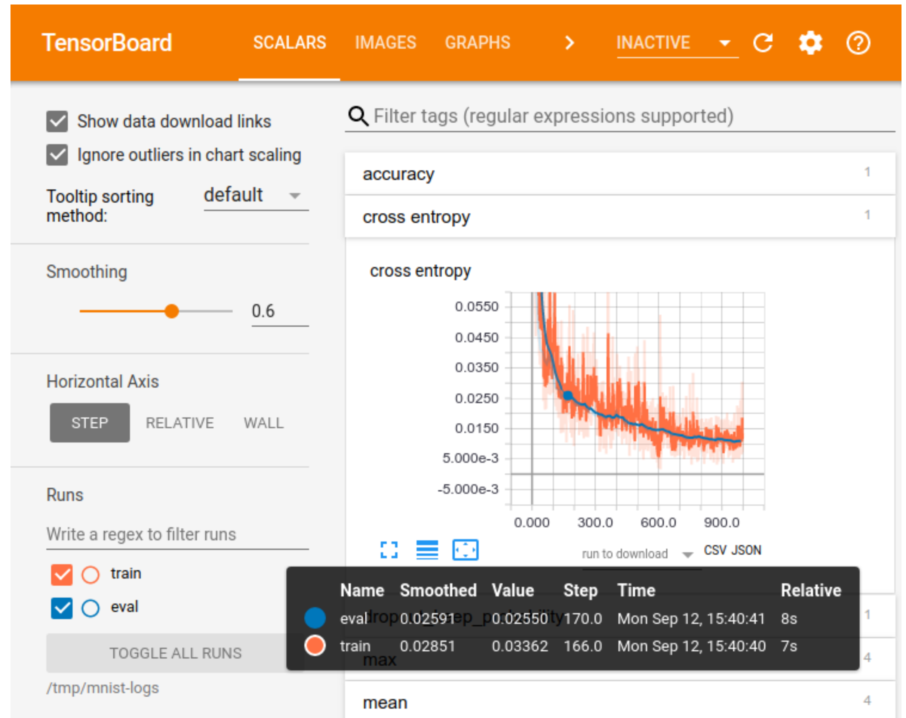
- Experiment and modelling at IPP
  - Data often structured and expensive to produce
  - Fast evaluation e.g. for real-time control, optimisation, parameter studies
  - Can do training in big datacenters

- It depends on the specific problem to solve which tools to use

What is TensorFlow?

- Dataflow programming framework developed by Google (C++, Python)
- High level frontend: Keras (now included)
- Runs fast on parallel GPU/TPU architectures

- Included analysis tool: *TensorBoard*



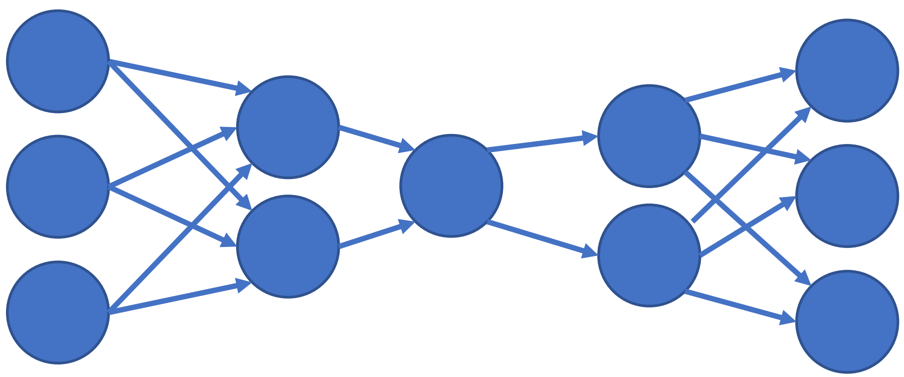
How to build networks yourself

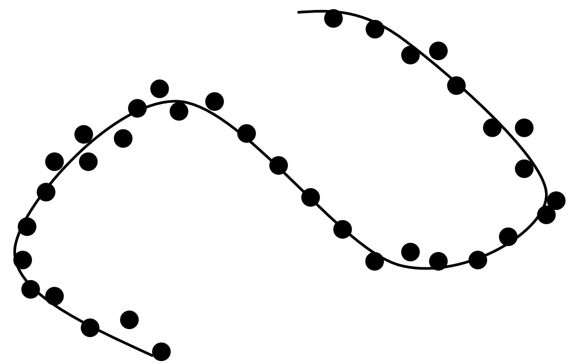
- Install Python environment
  - *NumPy/SciPy/Matplotlib* for Matlab-like functionality
  - *TensorFlow* (Alternative: *Theano*) with *Keras* frontend
    for high performance artificial neural networks
  - *scikit-learn* for different machine-learning methods and tools

- *Anaconda* Python distribution already includes most of it,
  use *conda install -c conda-forge tensorflow*
- *Spyder* is a good editor similar for Matlab
- *Jupyter Notebook* for Mathematica-like notebooks
- Look at examples, e.g. from https://blog.keras.io/

Example: Autoencoder

- We would like to build a tool for dimensionality reduction
- Map from N-D to N-D data, but only $M < N$ dimensions relevant
- Introduce „bottleneck“ layer of order M in neural network





- Let‘s start our own environment on http://localhost:8888/tree/Dropbox/ipp/neuralnet/jupyter
- Diagnostics (TensorBoard) on http://localhost:6006/

Examples for machine learning and neural networks at IPP

- Projects at TOK
  - Daniel Schäfer: MSc thesis (Zohm) with TU Eindhoven,
    Fast neural network surrogate model for QuaLiKiz (turbulence)
  - [AI-at-IPP], meetings organised by Lennart Bock
- Expertise at TUM
  - Tobias Neckel: Uncertainty quantification
  - Nils Thuerey: Neural networks, computer graphics
- Projects at NMPP
  - Helmholtz project: reduced complexity models (Albert, Tyranowski, v.Toussaint)
  - TODO: Coster, Hoenen, Luk, Preuss, v.Toussaint
  - Dirk Nille: PhD thesis, high-res/smooth divertor IR thermography via Bayes
- Tell me if you know more!

Initial ideas on symplectic neural networks

- A *linear* transformation stays *linear* with the right network…
  - Criterion: linearity of layers via linear activation functions
- … (non-linear) *symplectic* transform stays *symplectic* if done correctly!
  - Criterion: symplecticity of (combination of) layers
  - Interpolate mechanical system. Input: initial conditions, Output: final conditions
- Idea based on symplectic Euler integrator for canonical $z = (q, p)$

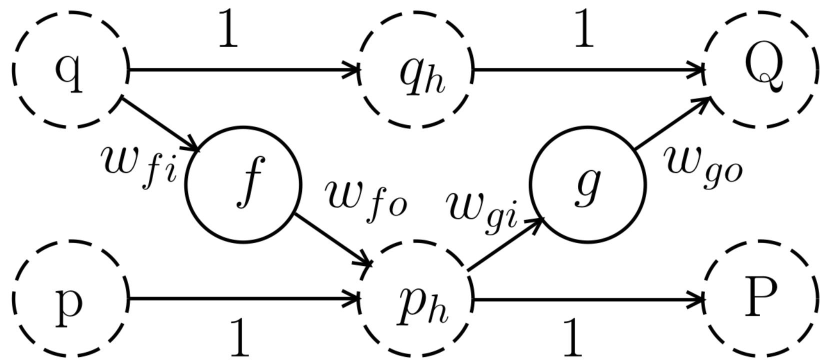
$$z^{(n+1)} = z^{(n)} + h \begin{cases} J\nabla H(q^{(n)}, p^{(n+1)}) & \text{(Variant 1)} \\ J\nabla H(q^{(n+1)}, p^{(n)}) & \text{(Variant 2)} \end{cases}$$

- Must be explicit (separable Hamiltonian H) for forward network

see Deco&Brauer, Neural Networks 8, 525 (1995) with focus on entropy

Initial ideas on symplectic neural networks

- Approximate real Hamiltonian map by many simple ones



- Activation function due to Hamiltonian H_n in each node,

$$f = -\frac{\partial H_n}{\partial q}, \qquad g = -\frac{\partial H_n}{\partial p}, \qquad \text{(Euler: } w_{.i} = 1, w_{.o} = h\text{)}$$

  - First guess: harmonic oscillator, but yields linear map
  - Second guess: pendulum, “half-harmonic” oscillator, or “quendulum”

$$H_n = \frac{p^2}{2} - \cos q\,, \qquad H_n = \left(\frac{p^2}{2} + \frac{q^2}{2}\right)\Theta(q), \qquad H_n = -\cos p - \cos q$$

Example: image recognition with convolutional net

https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html

("very little data" means 1000s of images artificially blown up to 10s of 1000s of training pairs)

Keras offers convolutional layers to detect features on pictures in a translationally invariant way

Also videos: Methods exist to recognize if someone is packing or un-packing the trunk of a car

Conclusion

- Choice of machine-learning method is highly problem-specific
- Current boom has made a lot of user-friendly tools available
  - Extending methods still requires low-level work

- Artificial neural networks for classification and regression
  - Working on unstructured data
  - Involve hard optimisation problem while training
  - Once trained, extremely fast to evaluate (in parallel)

- The machine is only clever if you are

Thank you for your attention!

Talk available on https://itp.tugraz.at/~ert/teaching