

Machine Learning for Collinear QCD

Lecture 2

Rabah Abdul Khalek
Jefferson Lab

CFNS 2022
Tuesday 07.19.2022

Outline

- A Recap on Machine Learning
- B Intro to Neural Networks
- C Feed-forward Neural Networks
- D Monte-Carlo Inference
- E Tutorial 1 — Minimisation

Machine Learning

What do you want the machine learning system to do?

I want to see if there are natural clusters or dimensions in the data I have about different situations.

I want to learn what actions to take in different situations.

Do you want the ML system to be active or passive?

ACTIVE

The system's own actions will affect the situations it sees in the future.

PASSIVE

The system will learn from data I give it.

Do you have access to data that describes a lot of examples of situations and appropriate actions for each situation?

Will the system be able to gather a lot of data by trying sequences of actions in many different situations and seeing the results?

No

Could a knowledgeable human decide what actions to take based on the data you have about the situation?

Yes

No

Yes

Could there be patterns in these situations that humans haven't recognized before?

No

Yes

UNSUPERVISED LEARNING MAY BE APPROPRIATE

*clustering
anomaly detection*

SUPERVISED LEARNING MAY BE APPROPRIATE

*neural nets
support vector machines
regression
recommender systems*

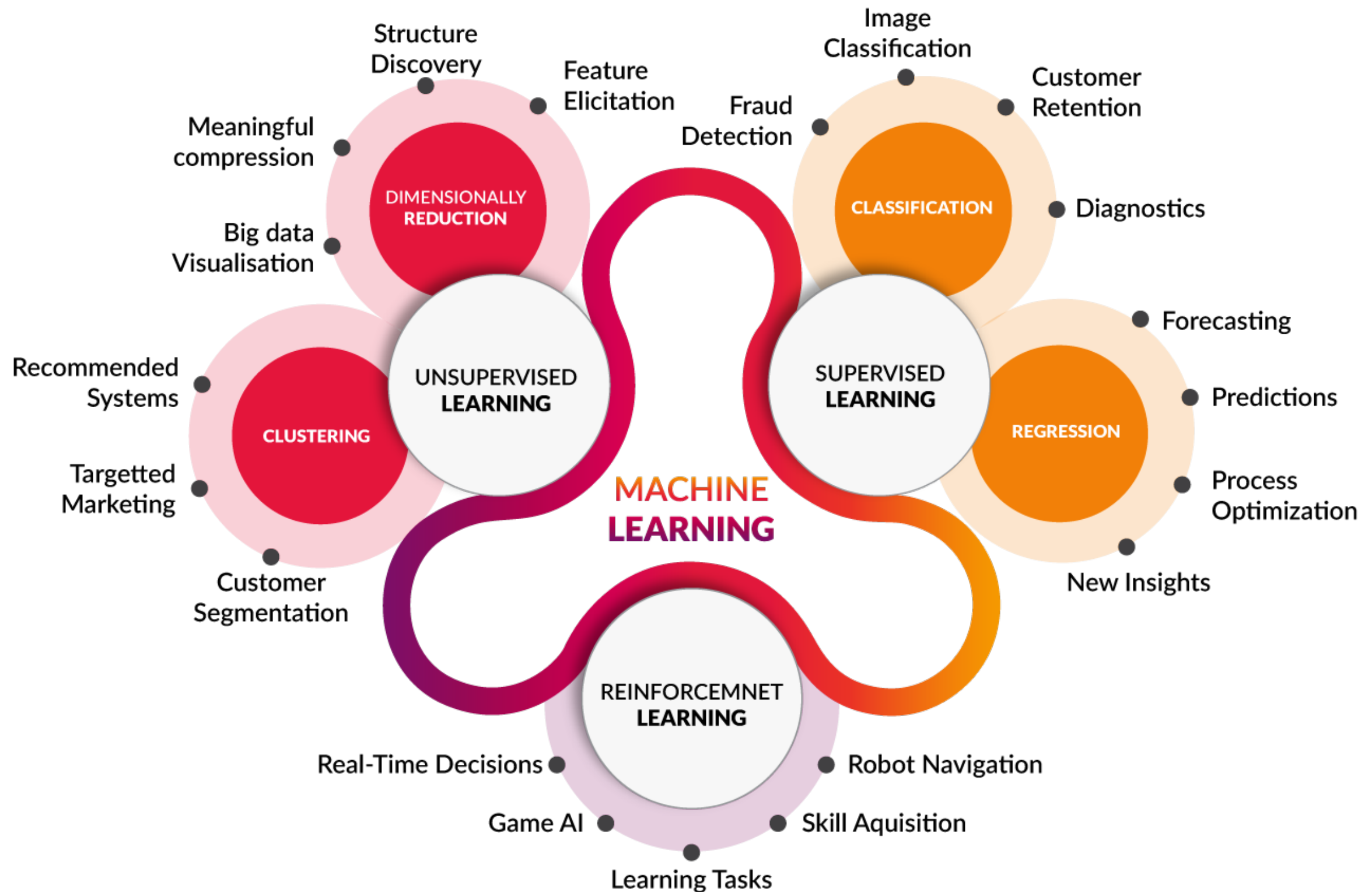
MACHINE LEARNING IS NOT USEFUL

REINFORCEMENT LEARNING MAY BE APPROPRIATE

Credit: Thomas Malone, MIT Sloan | Design: Laura Wentzel

Source: Thomas Malone | MIT Sloan. See: <https://bit.ly/3gvRho2>, Figure 2.

Machine Learning



Source: <https://medium.com/analytics-vidhya/an-introduction-to-machine-learning-574bafa6fc66>

A mostly complete chart of Neural Networks

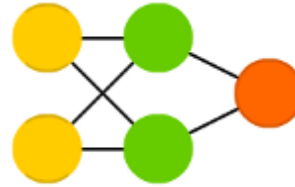
©2016 Fjodor van Veen - asimovinstitute.org

-  Backfed Input Cell
-  Input Cell
-  Noisy Input Cell
-  Hidden Cell
-  Probabilistic Hidden Cell
-  Spiking Hidden Cell
-  Output Cell
-  Match Input Output Cell
-  Recurrent Cell
-  Memory Cell
-  Different Memory Cell
-  Kernel
-  Convolution or Pool

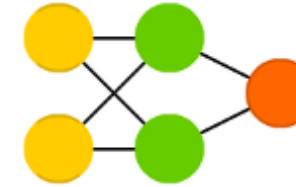
Perceptron (P)



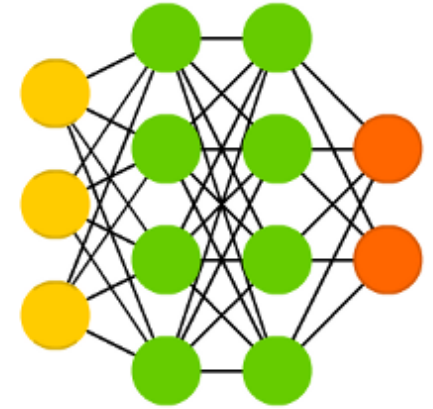
Feed Forward (FF)



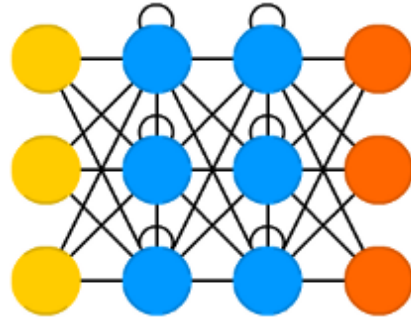
Radial Basis Network (RBF)



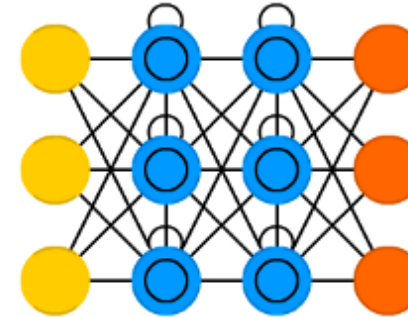
Deep Feed Forward (DFF)



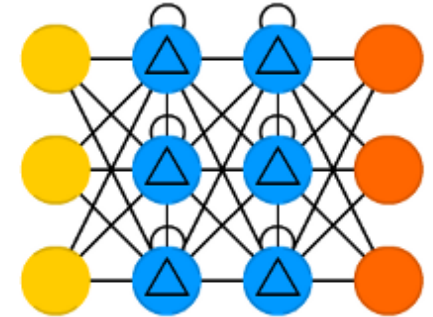
Recurrent Neural Network (RNN)



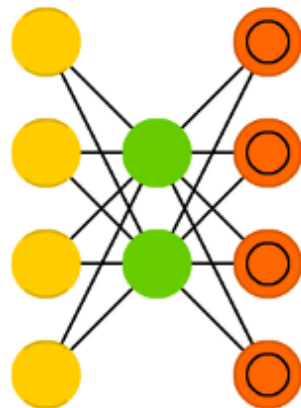
Long / Short Term Memory (LSTM)



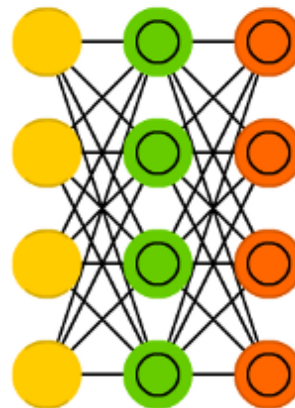
Gated Recurrent Unit (GRU)



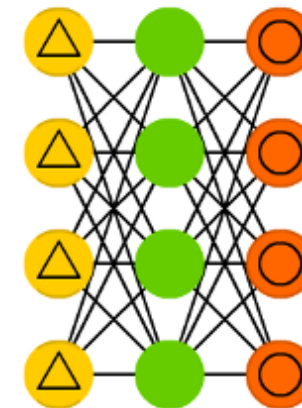
Auto Encoder (AE)



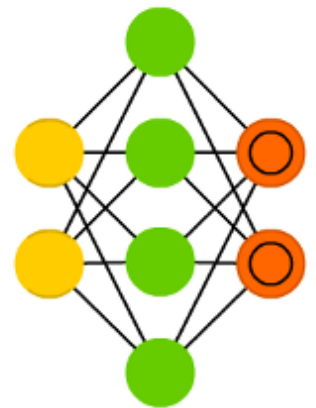
Variational AE (VAE)



Denoising AE (DAE)



Sparse AE (SAE)



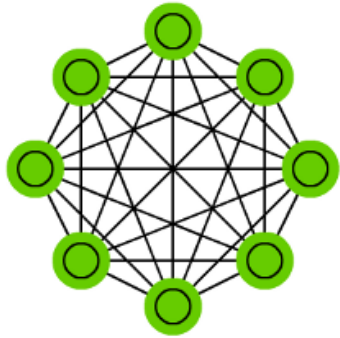
Source: <https://towardsdatascience.com/the-mostly-complete-chart-of-neural-networks-explained-3fb6f2367464>

A mostly complete chart of

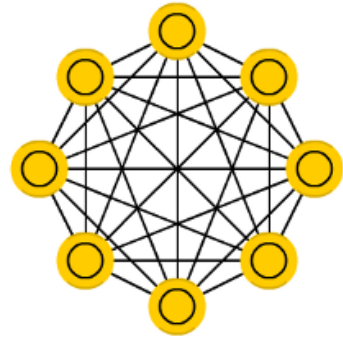
Neural Networks

©2016 Fjodor van Veen - asimovinstitute.org

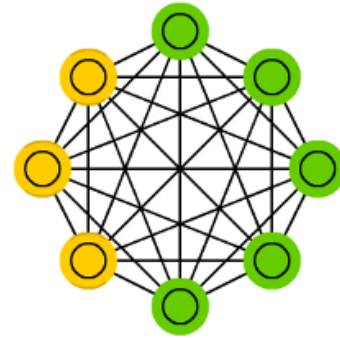
Markov Chain (MC)



Hopfield Network (HN)



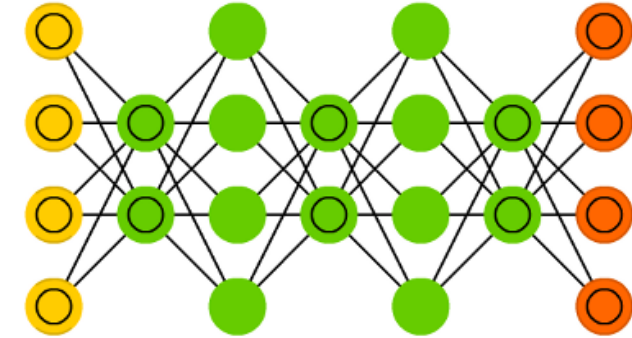
Boltzmann Machine (BM)



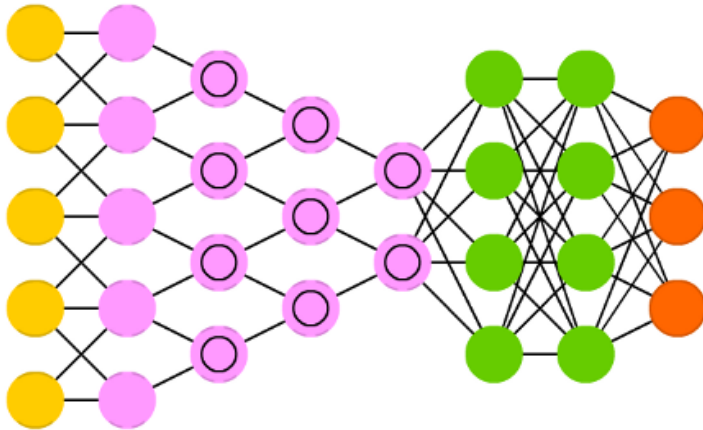
Restricted BM (RBM)



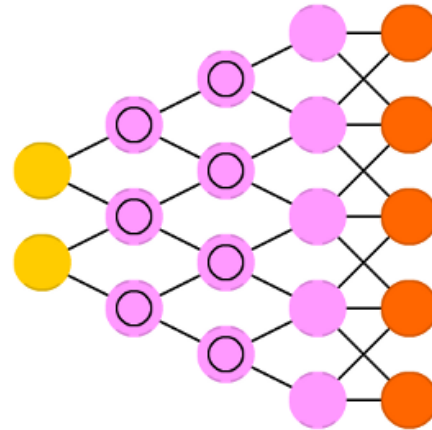
Deep Belief Network (DBN)



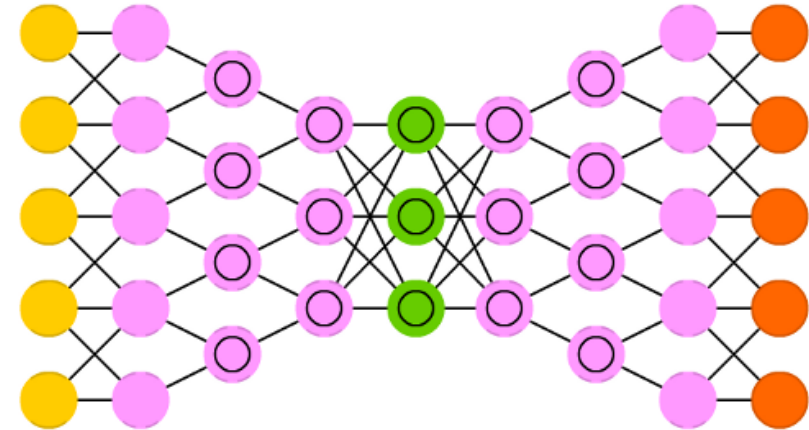
Deep Convolutional Network (DCN)



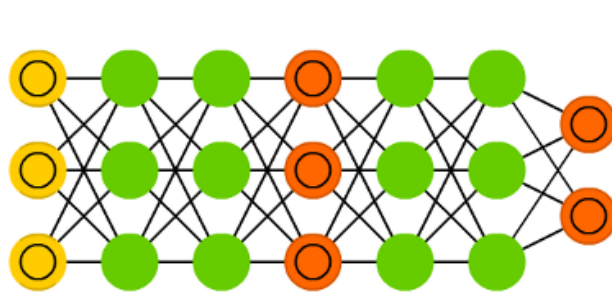
Deconvolutional Network (DN)



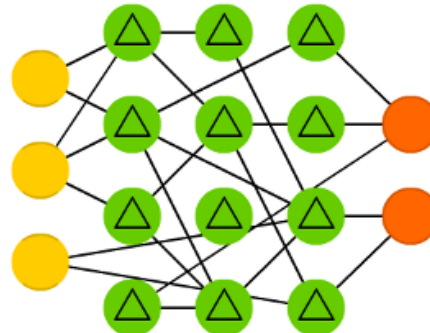
Deep Convolutional Inverse Graphics Network (DCIGN)



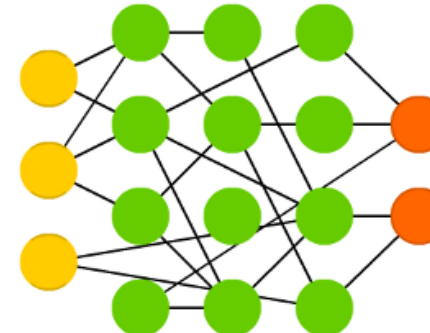
Generative Adversarial Network (GAN)



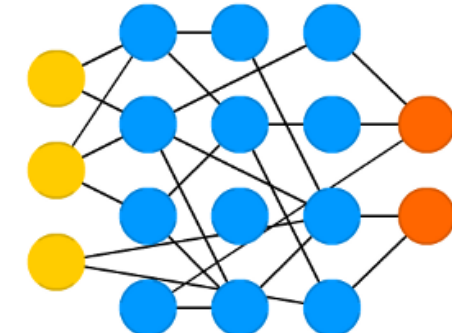
Liquid State Machine (LSM)



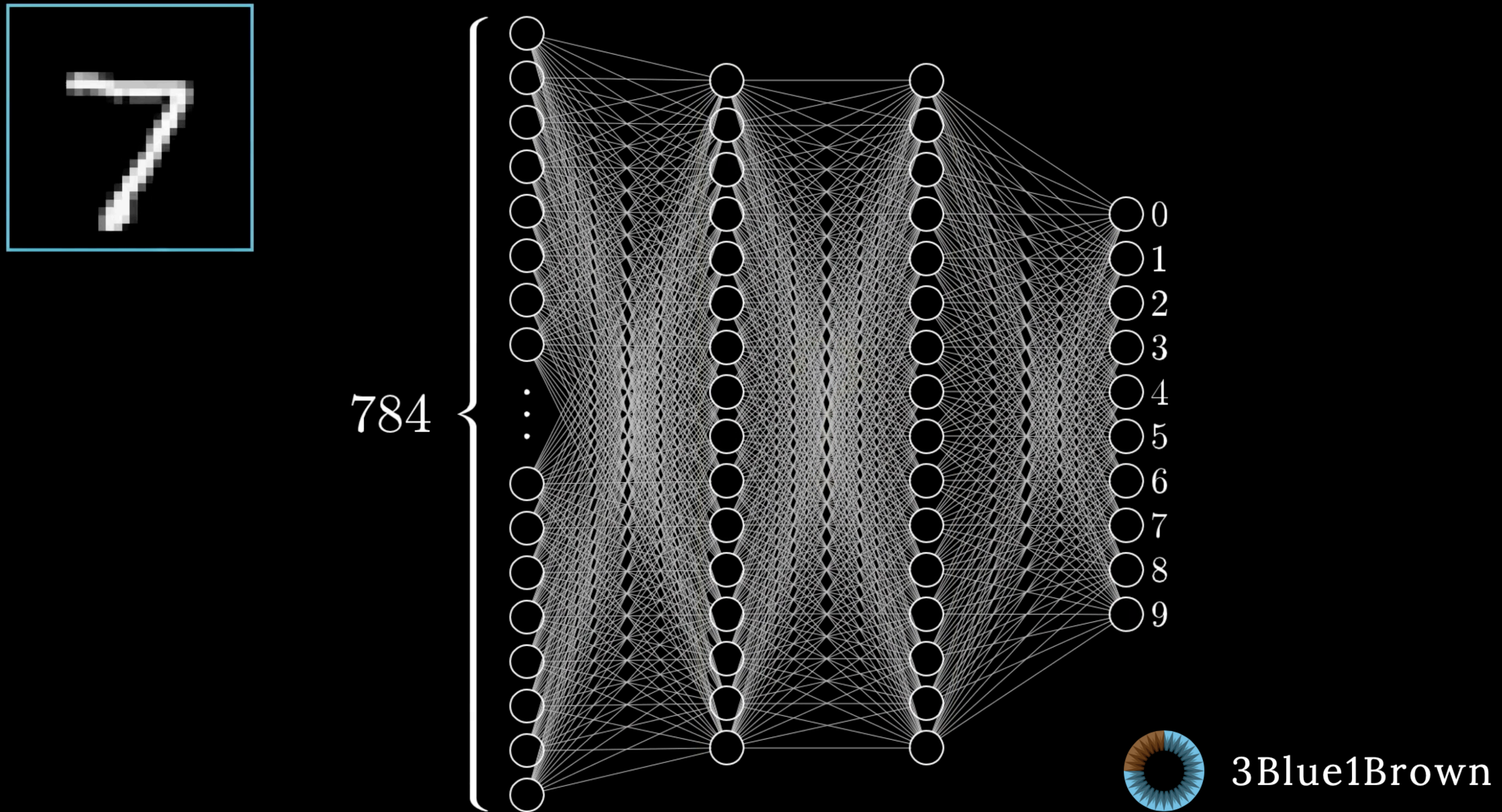
Extreme Learning Machine (ELM)



Echo State Network (ESN)



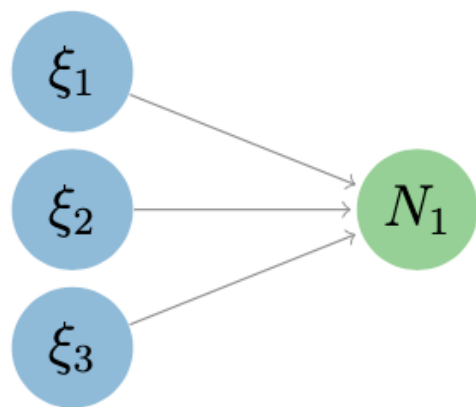
Feed-forward Neural Networks



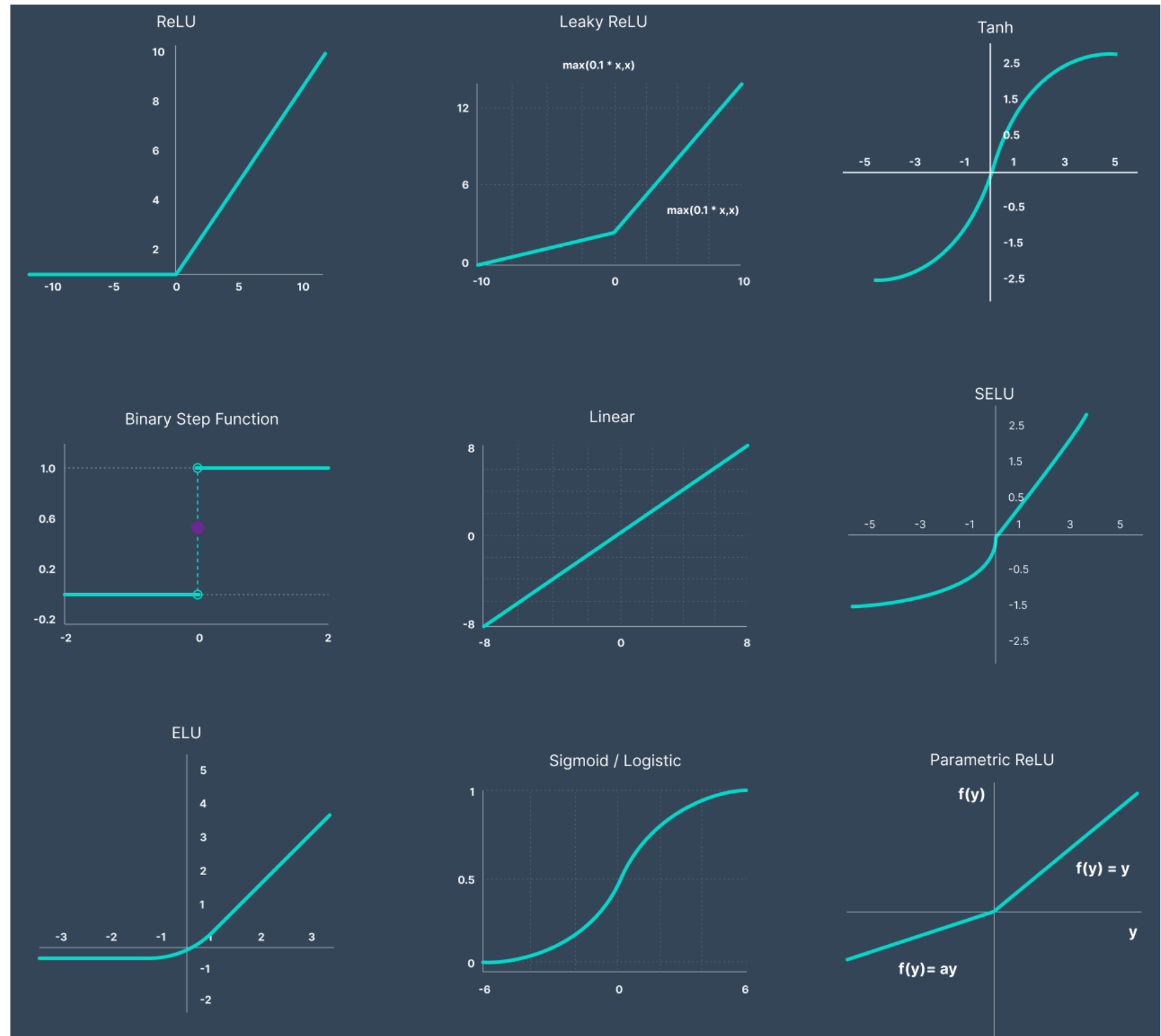
Feed-forward Neural Networks

Elementary Neural Network

Inputs Neuron

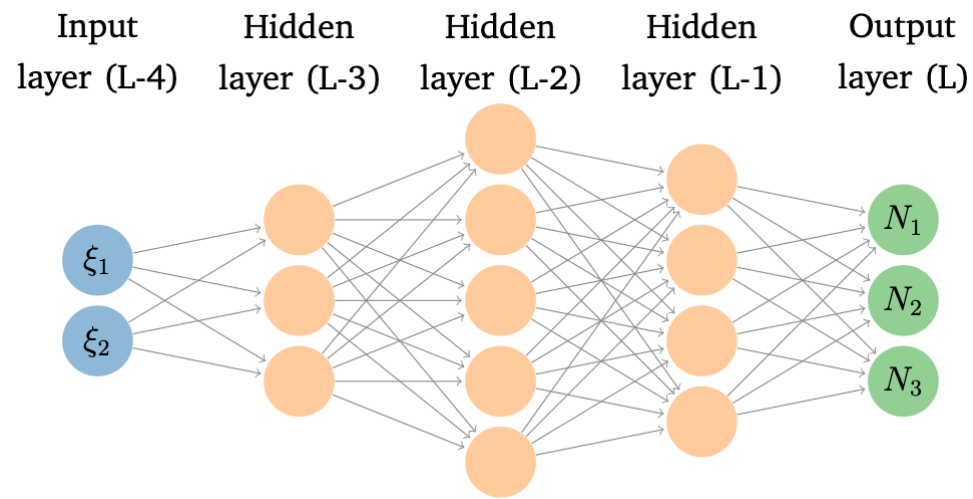


$$N_1(\xi; \{w_{1i}, \theta_1\}) = \phi \left(\sum_i^3 w_{1i} \xi_i + \theta_1 \right)$$



How does f/b-propagation work?

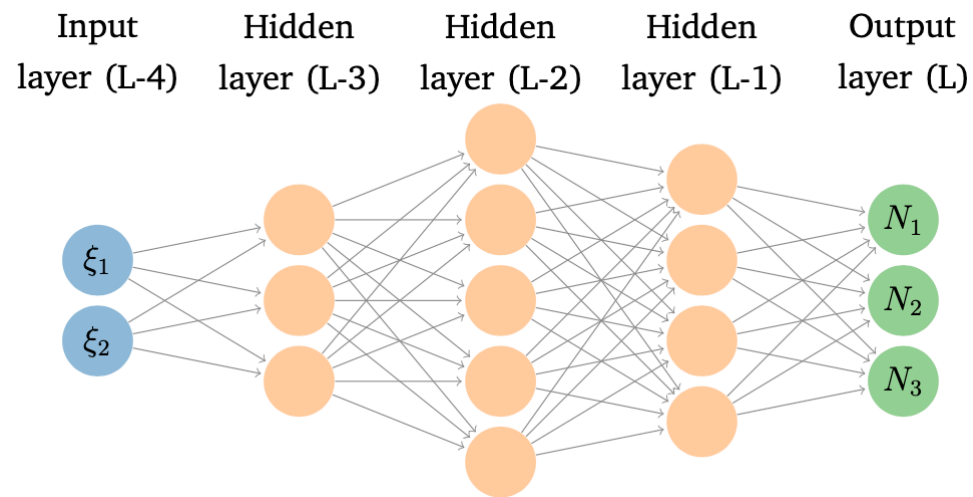
Forward-propagation



$$\begin{aligned}
 N_k(\boldsymbol{\xi}; \{\omega_{ij}^{(\ell)}, \theta_i^{(\ell)}\}) &= \phi_L \left(\sum_{j^{(1)}=1}^{N_{L-1}} \omega_{kj^{(1)}}^{(L)} y_{j^{(1)}}^{(L-1)} + \theta_k^{(L)} \right) \\
 &= \phi_L \left(\sum_{j^{(1)}=1}^{N_{L-1}} \omega_{kj^{(1)}}^{(L)} \phi_{L-1} \left(\sum_{j^{(2)}=1}^{N_{L-2}} \omega_{j^{(1)}j^{(2)}}^{(L)} y_{j^{(2)}}^{(L-2)} + \theta_{j^{(1)}}^{(L-1)} \right) + \theta_k^{(L)} \right) \\
 &= \dots
 \end{aligned}$$

How does f/b-propagation work?

Forward-propagation



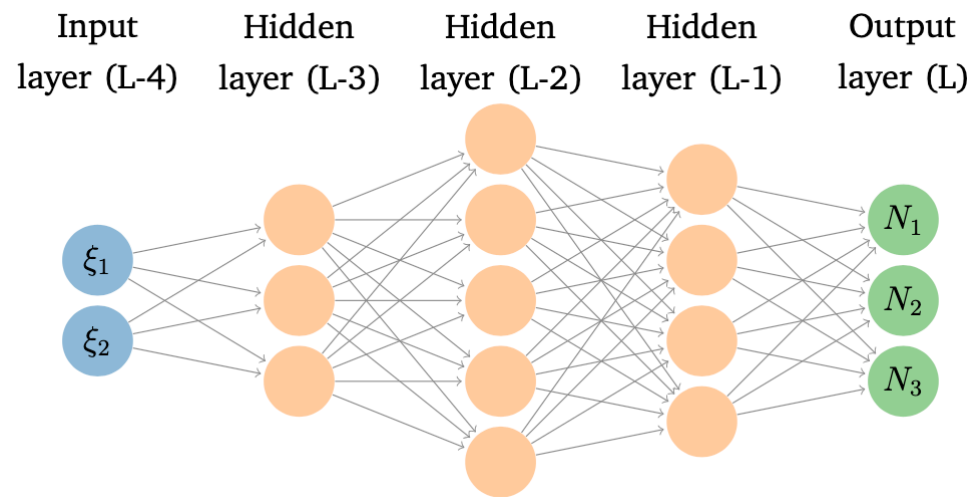
$$\begin{aligned}
 N_k(\xi; \{\omega_{ij}^{(\ell)}, \theta_i^{(\ell)}\}) &= \phi_L \left(\sum_{j^{(1)}}^{N_{L-1}} \omega_{kj^{(1)}}^{(L)} y_{j^{(1)}}^{(L-1)} + \theta_k^{(L)} \right) \\
 &= \phi_L \left(\sum_{j^{(1)=1}}^{N_{L-1}} \omega_{kj^{(1)}}^{(L)} \phi_{L-1} \left(\sum_{j^{(2)=1}}^{N_{L-2}} \omega_{j^{(1)}j^{(2)}}^{(L)} y_{j^{(2)}}^{(L-2)} + \theta_{j^{(1)}}^{(L-1)} \right) + \theta_k^{(L)} \right) \\
 &= \dots
 \end{aligned}$$

backward-propagation

$$\begin{aligned}
 \chi^2[\{\omega_{ij}^{(\ell)}, \theta_i^{(\ell)}\}] &= \sum_{k=1}^n \left(\frac{N_k(\xi; \{\omega_{ij}^{(\ell)}, \theta_i^{(\ell)}\}) - \sigma_k}{s_k} \right)^2 \\
 \frac{\partial \chi^2}{\partial \omega_{ij}^{(\ell)}} &= 2 \sum_{k=1}^n \left(\frac{N_k(\xi; \{\omega_{ij}^{(\ell)}, \theta_i^{(\ell)}\}) - \sigma_k}{s_k^2} \right) \frac{\partial N_k}{\partial \omega_{ij}^{(\ell)}}
 \end{aligned}$$

How does f/b-propagation work?

Forward-propagation



$$N_k(\xi; \{\omega_{ij}^{(\ell)}, \theta_i^{(\ell)}\}) = \phi_L \left(\sum_{j^{(1)}}^{N_{L-1}} \omega_{kj^{(1)}}^{(L)} y_{j^{(1)}}^{(L-1)} + \theta_k^{(L)} \right)$$

$$= \phi_L \left(\sum_{j^{(1)=1}}^{N_{L-1}} \omega_{kj^{(1)}}^{(L)} \phi_{L-1} \left(\sum_{j^{(2)=1}}^{N_{L-2}} \omega_{j^{(1)}j^{(2)}}^{(L)} y_{j^{(2)}}^{(L-2)} + \theta_{j^{(1)}}^{(L-1)} \right) + \theta_k^{(L)} \right)$$

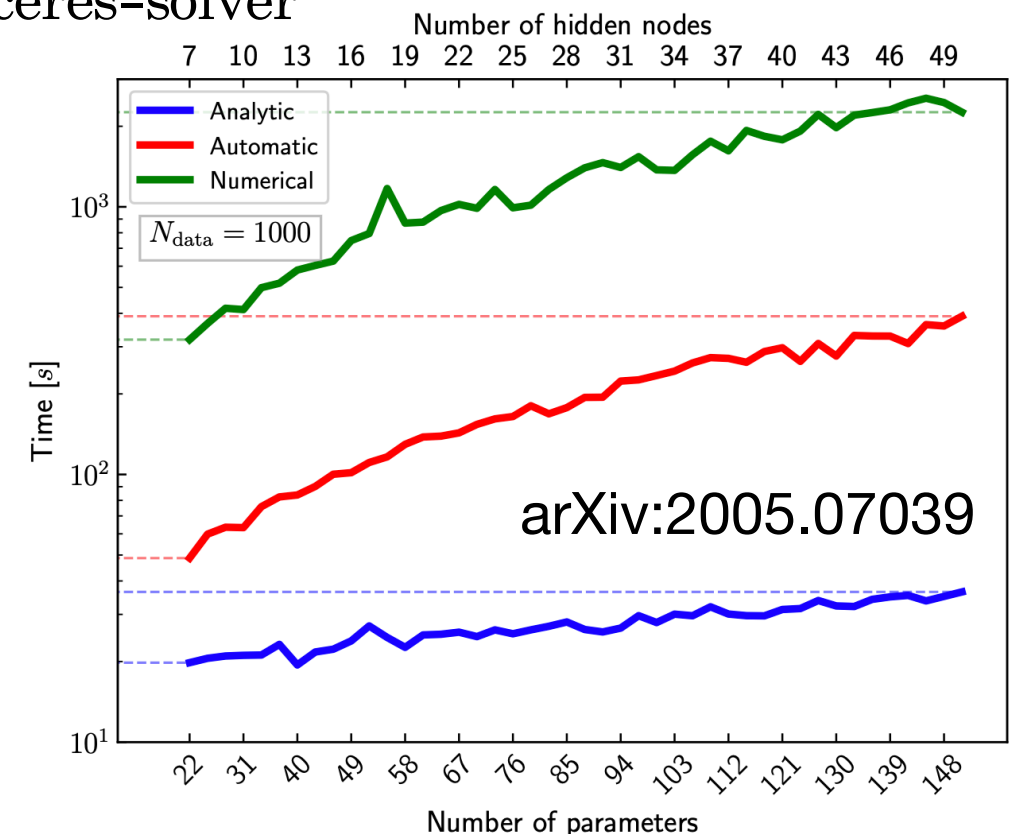
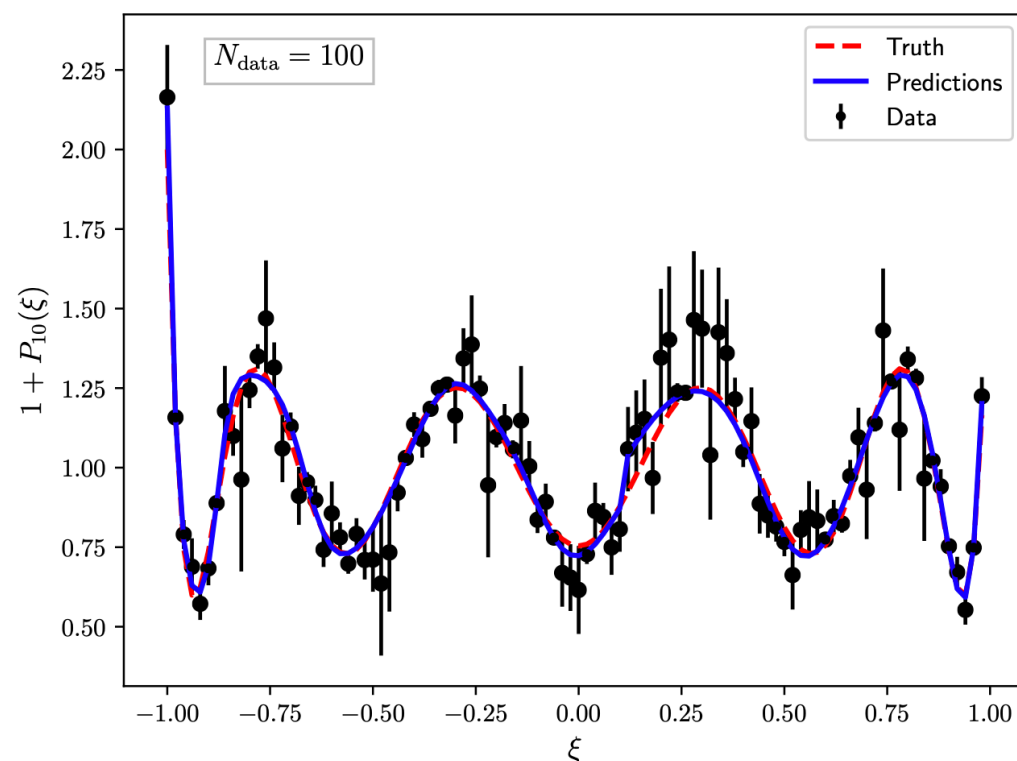
$$= \dots$$

backward-propagation

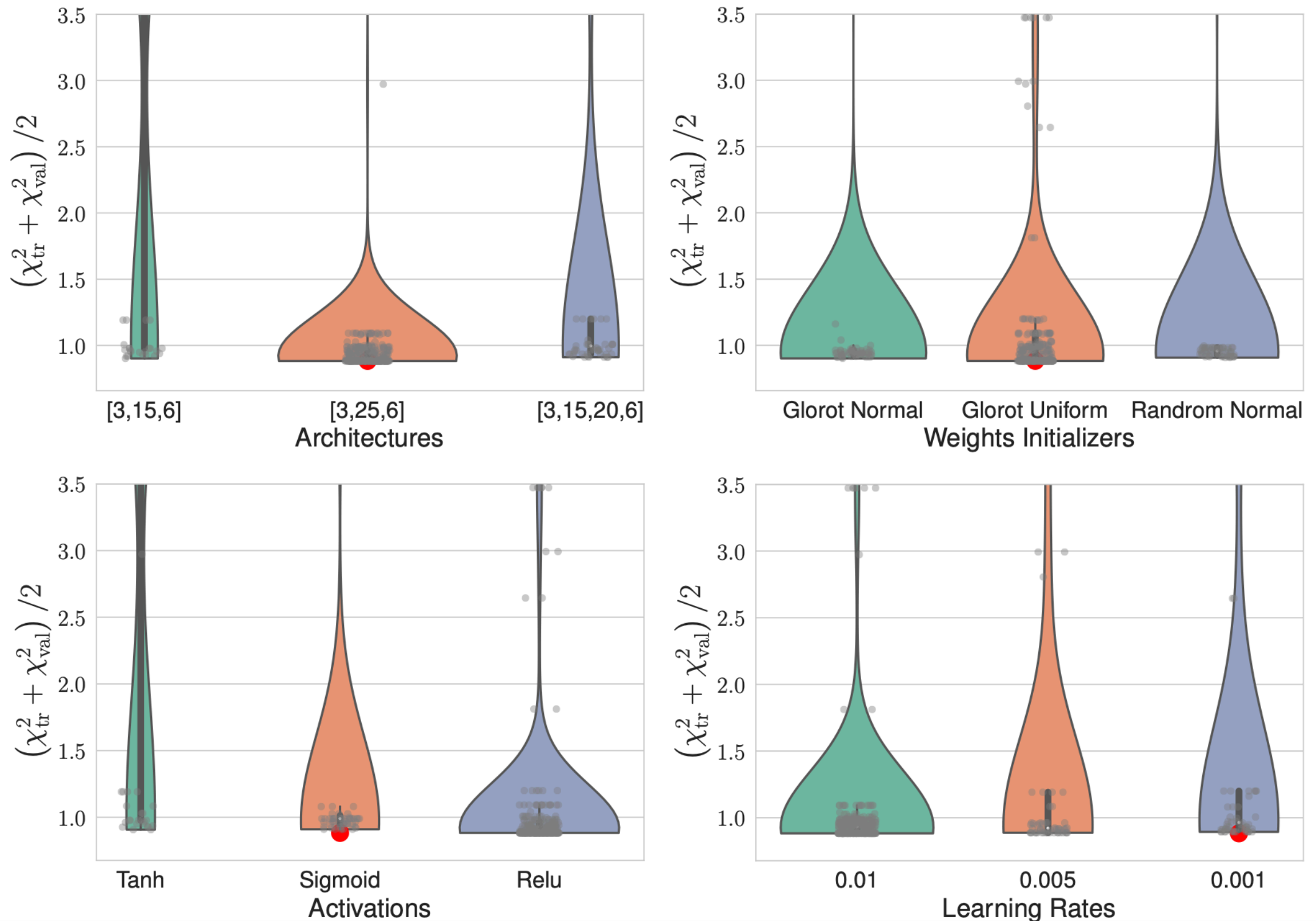
$$\chi^2[\{\omega_{ij}^{(\ell)}, \theta_i^{(\ell)}\}] = \sum_{k=1}^n \left(\frac{N_k(\xi; \{\omega_{ij}^{(\ell)}, \theta_i^{(\ell)}\}) - \sigma_k}{s_k} \right)^2$$

$$\frac{\partial \chi^2}{\partial \omega_{ij}^{(\ell)}} = 2 \sum_{k=1}^n \left(\frac{N_k(\xi; \{\omega_{ij}^{(\ell)}, \theta_i^{(\ell)}\}) - \sigma_k}{s_k^2} \right) \frac{\partial N_k}{\partial \omega_{ij}^{(\ell)}}$$

NNAD C++ library & ceres-solver



Hyper-optimisation



Monte-Carlo Inference

- ◆ **The Gaussian assumption:**
(data distribution)

$$\mathcal{G}(\mathbf{x}^{(k)}) \propto \exp \left[\left(\mathbf{x}^{(k)} - \boldsymbol{\mu} \right)^T \cdot \mathbf{C}^{-1} \cdot \left(\mathbf{x}^{(k)} - \boldsymbol{\mu} \right) \right]$$

- ◆ **Covariance Matrix:**

$$C_{ij} = \delta_{ij} \sigma_{i, \text{unc}}^2 + \sum_{\beta} \sigma_{i, \text{corr}}^{(\beta)} \sigma_{j, \text{corr}}^{(\beta)}$$

Monte-Carlo Inference

- ◆ **The Gaussian assumption:**
(data distribution)

$$\mathcal{G}(\mathbf{x}^{(k)}) \propto \exp \left[\left(\mathbf{x}^{(k)} - \boldsymbol{\mu} \right)^T \cdot \mathbf{C}^{-1} \cdot \left(\mathbf{x}^{(k)} - \boldsymbol{\mu} \right) \right]$$

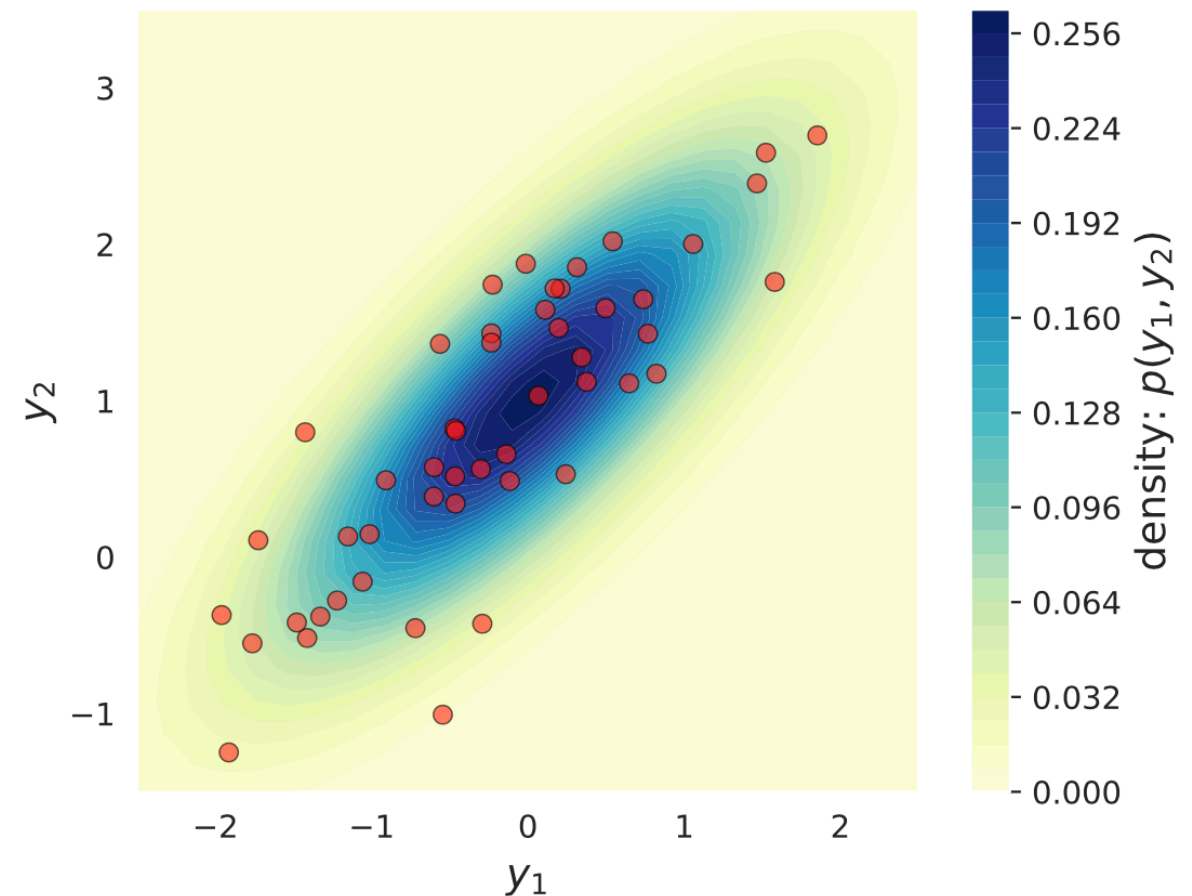
- ◆ **Covariance Matrix:**

$$C_{ij} = \delta_{ij} \sigma_{i, \text{unc}}^2 + \sum_{\beta} \sigma_{i, \text{corr}}^{(\beta)} \sigma_{j, \text{corr}}^{(\beta)}$$

- ◆ **Replica generation:** $\mathbf{x}^{(k)} = \boldsymbol{\mu} + \mathbf{L} \cdot \mathbf{r}^{(k)}$
($\mathbf{C} = \mathbf{L} \cdot \mathbf{L}^T$)

$$\rightarrow \frac{1}{N_{\text{rep}}} \sum_k x_i^{(k)} \simeq \mu_i,$$

$$\rightarrow \frac{1}{N_{\text{rep}}} \sum_k x_i^{(k)} x_j^{(k)} \simeq \mu_i \mu_j + C_{ij}$$



Monte-Carlo Inference

◆ **The Gaussian assumption:** (data distribution) $\mathcal{G}(\mathbf{x}^{(k)}) \propto \exp \left[\left(\mathbf{x}^{(k)} - \boldsymbol{\mu} \right)^T \cdot \mathbf{C}^{-1} \cdot \left(\mathbf{x}^{(k)} - \boldsymbol{\mu} \right) \right]$

◆ **Covariance Matrix:** $C_{ij} = \delta_{ij} \sigma_{i, \text{unc}}^2 + \sum_{\beta} \sigma_{i, \text{corr}}^{(\beta)} \sigma_{j, \text{corr}}^{(\beta)}$

◆ **Replica generation:** $\mathbf{x}^{(k)} = \boldsymbol{\mu} + \mathbf{L} \cdot \mathbf{r}^{(k)}$
 $(\mathbf{C} = \mathbf{L} \cdot \mathbf{L}^T)$

◆ **Maximum-Likelihood estimate:** $\max_{\boldsymbol{\theta}} \ln(\mathcal{L}(\boldsymbol{\theta} | \boldsymbol{\mu})) \rightarrow \min_{\boldsymbol{\theta}} \chi^2(\boldsymbol{\theta}) \rightarrow \hat{\boldsymbol{\theta}}$

$$\max_{\boldsymbol{\theta}^{(1)}} \{ \ln[L(\boldsymbol{\theta}^{(1)} | x^{(1)})] \} \rightarrow \min_{\boldsymbol{\theta}^{(1)}} \chi^2(\boldsymbol{\theta}^{(1)}) \rightarrow \hat{\boldsymbol{\theta}}^{(1)}$$

⋮

$$\max_{\boldsymbol{\theta}^{(k)}} \{ \ln[L(\boldsymbol{\theta}^{(k)} | x^{(k)})] \} \rightarrow \min_{\boldsymbol{\theta}^{(1)}} \chi^2(\boldsymbol{\theta}^{(k)}) \rightarrow \hat{\boldsymbol{\theta}}^{(k)}$$

⋮

$$\max_{\boldsymbol{\theta}^{(N_{rep})}} \{ \ln[L(\boldsymbol{\theta}^{(N_{rep})} | x^{(N_{rep})})] \} \rightarrow \min_{\boldsymbol{\theta}^{(N_{rep})}} \chi^2(\boldsymbol{\theta}^{(1)}) \rightarrow \hat{\boldsymbol{\theta}}^{(N_{rep})}$$

Monte-Carlo Inference

- ◆ The Gaussian assumption:
(data distribution)

$$\mathcal{G}(\mathbf{x}^{(k)}) \propto \exp \left[\left(\mathbf{x}^{(k)} - \boldsymbol{\mu} \right)^T \cdot \mathbf{C}^{-1} \cdot \left(\mathbf{x}^{(k)} - \boldsymbol{\mu} \right) \right]$$

- ◆ Covariance Matrix:

$$C_{ij} = \delta_{ij} \sigma_{i, \text{unc}}^2 + \sum_{\beta} \sigma_{i, \text{corr}}^{(\beta)} \sigma_{j, \text{corr}}^{(\beta)}$$

- ◆ Replica generation: $\mathbf{x}^{(k)} = \boldsymbol{\mu} + \mathbf{L} \cdot \mathbf{r}^{(k)}$
($\mathbf{C} = \mathbf{L} \cdot \mathbf{L}^T$)

- ◆ Maximum-Likelihood estimate: $\max_{\boldsymbol{\theta}} \ln(\mathcal{L}(\boldsymbol{\theta}|\boldsymbol{\mu})) \rightarrow \min_{\boldsymbol{\theta}} \chi^2(\boldsymbol{\theta}) \rightarrow \hat{\boldsymbol{\theta}}$

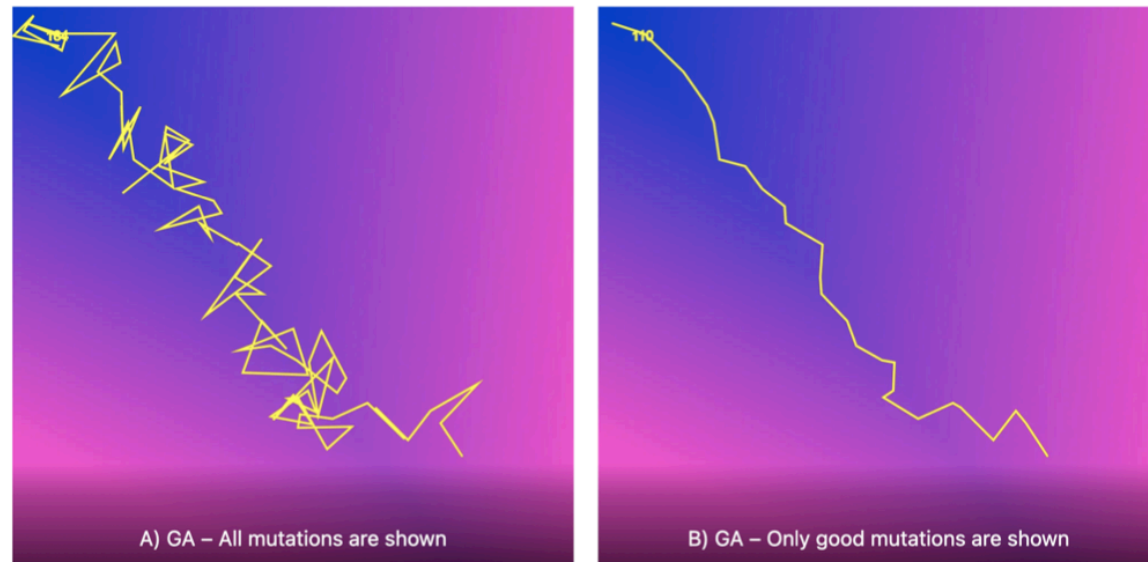
In order to find the maximum of the log-likelihood $\ln \mathcal{L}(\boldsymbol{\theta}|\boldsymbol{\sigma})$ subject to N_{EC} equality constraints $\{(h_1(\boldsymbol{\theta}) = 0), \dots, (h_{N_{\text{EC}}}(\boldsymbol{\theta}) = 0)\}$, the method of Lagrange multipliers state that one should instead find the maximum of the Lagrange function defined as:

$$L(\boldsymbol{\theta}, \boldsymbol{\sigma}, \boldsymbol{\lambda}) = \ln \mathcal{L}(\boldsymbol{\theta}|\boldsymbol{\sigma}) - \sum_i^{N_{\text{EC}}} \lambda_i h_i(\boldsymbol{\theta})$$
$$\max_{\boldsymbol{\theta}} L(\boldsymbol{\theta}, \boldsymbol{\sigma}, \boldsymbol{\lambda}) = \min_{\boldsymbol{\theta}} \left[\chi^2(\boldsymbol{\theta}|\boldsymbol{\sigma}) + \sum_i^{N_{\text{EC}}} \lambda_i h_i(\boldsymbol{\theta}) \right]$$

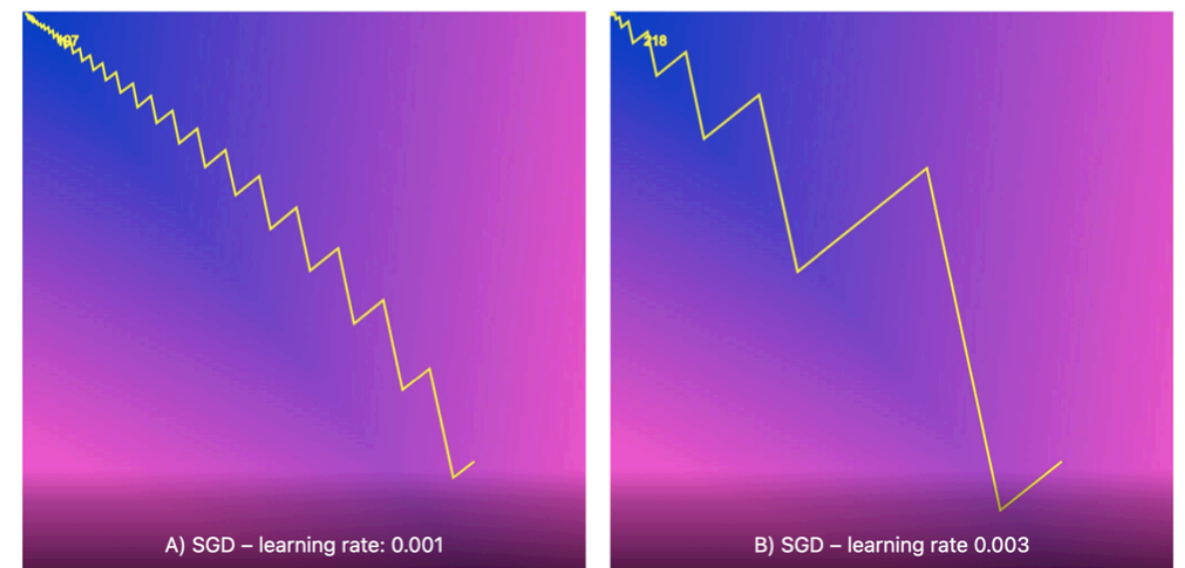
where λ_i are called the lagrange multipliers that can be chosen in a way to determine the size of the constraint contribution during the minimisation.

Minimisation

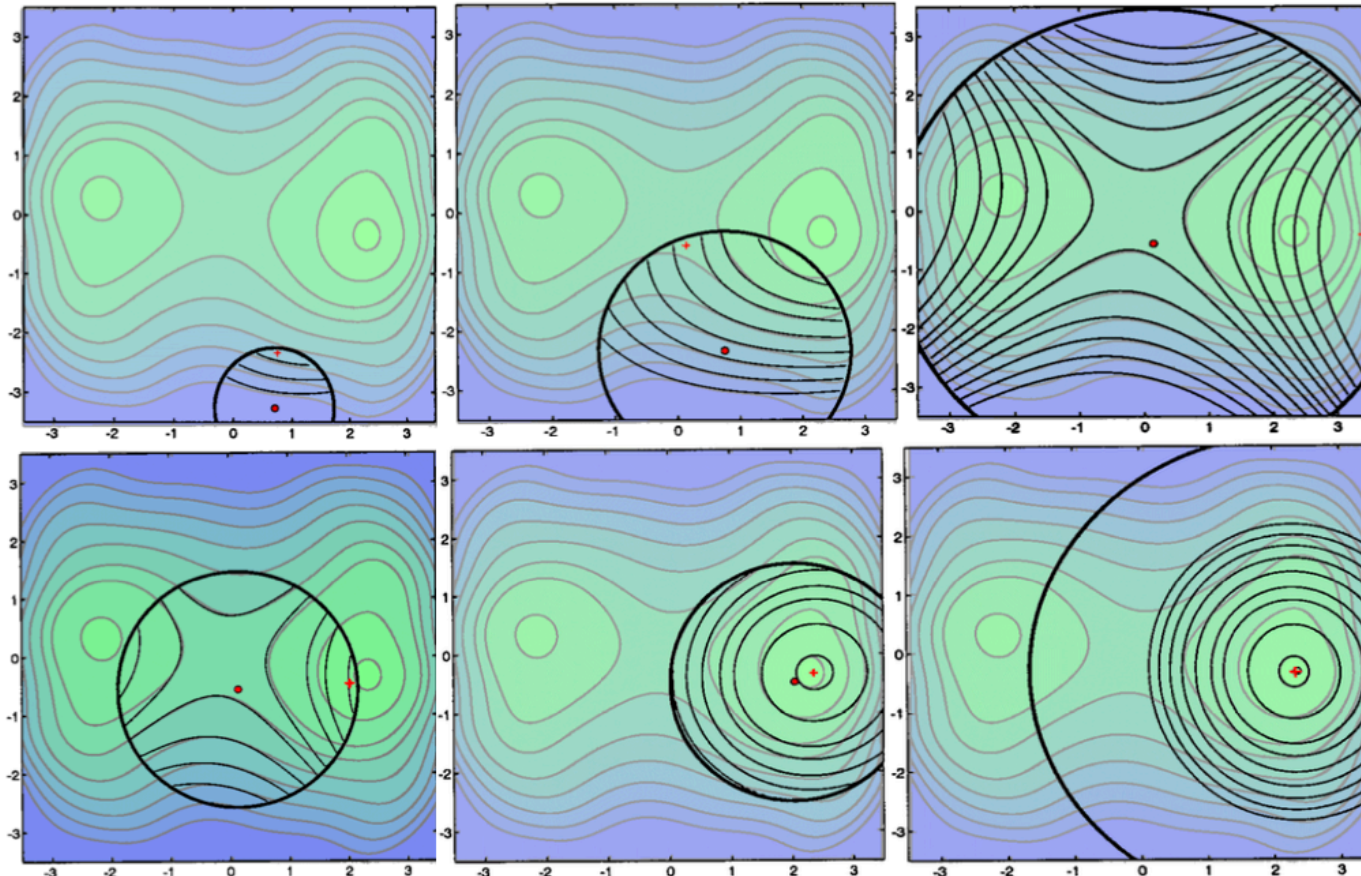
Genetic Algorithms



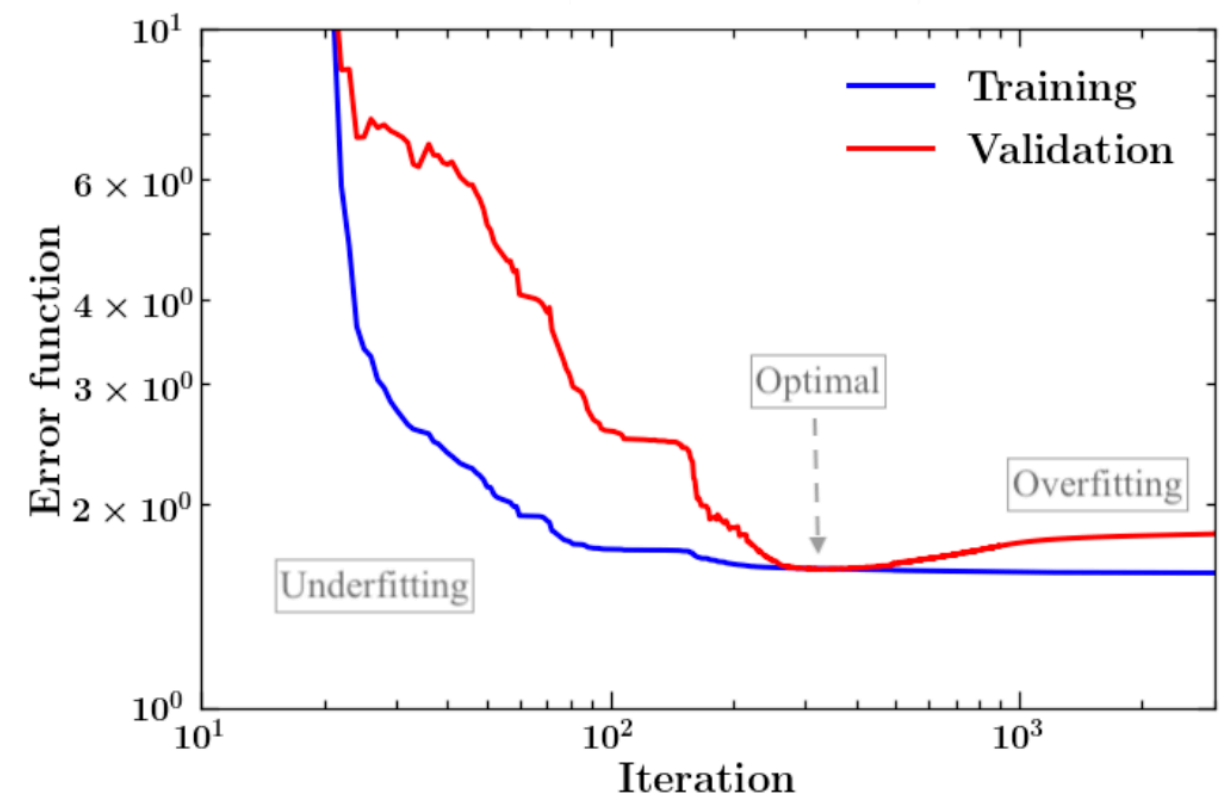
Gradient descent algorithms



Trust-region algorithms



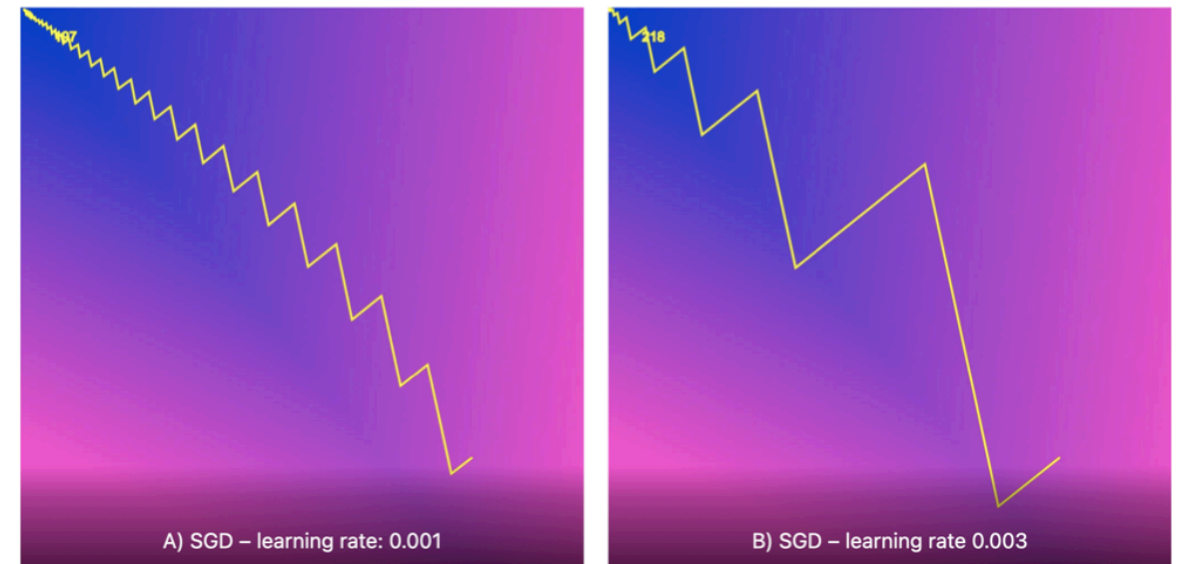
Cross-validation



Tutorial 1 — Minimisation

Gradient descent algorithms

Time for Tutorial 1!



Tutorial 1: Optimisation with Stochastic Gradient Descent in 1D

Author and references

Author: Daniel Newman - <https://github.com/dtnewman>

Edits: Rabah Abdul Khalek - khalek@jlab.org (Postdoc at JLab | Theory division)

Learning Goals

This notebook will serve as an introduction to **Gradient descent**, also known as **steepest descent**, an optimization algorithm for finding the local minimum of a function.