
Deep Learning and its application in tokamak magnetic equilibrium reconstruction

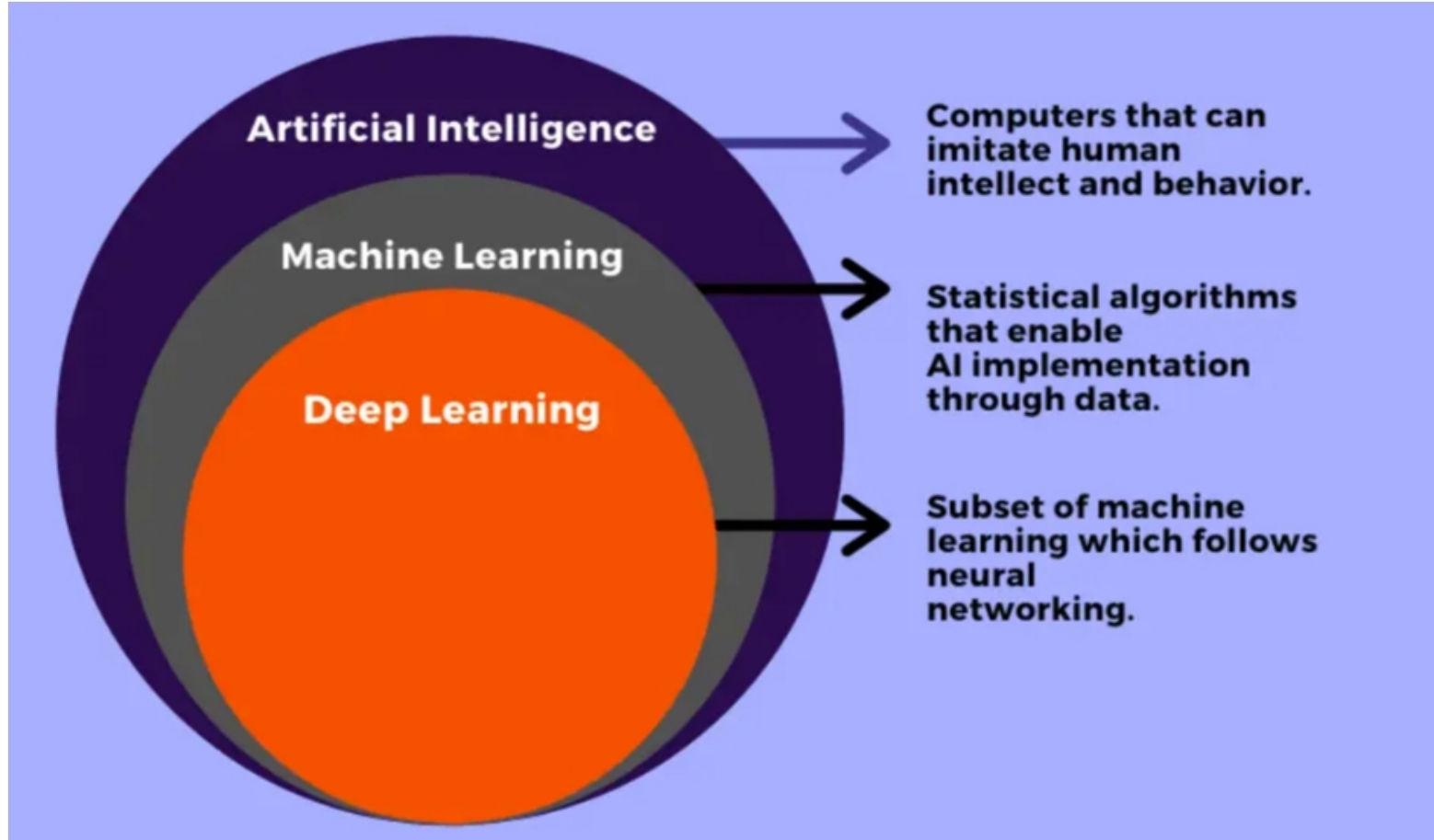
Youjun Hu (胡友俊)

Institute of Plasma Physics, Chinese Academy of Sciences
2022-05-27, seminar@ASIPP

Outline

- **Artificial intelligence (AI) and Machine Learning (ML)**
- **Deep Learning**
 - **Neural Network**
- **Reconstruct EAST magnetic equilibrium using deep learning**
- **How to use Tensorflow (deep learning library by google)**

Artificial intelligence (AI) and Machine Learning (ML)



A short history of AI

- Founded as an academic discipline in 1956 (**LISP vs. Fortran**)
- Waves of optimism, followed by disappointment => loss of funding (known as "AI winter"), followed by new approaches => success and renewed funding
- AI research has tried and discarded many different approaches since its founding
- Traditional AI: formal logic, explicit rule-based (**If-then**)
- Modern AI: based on numerical optimization, statistical machine learning, using data to infer rules
- Deep learning: due to discover of an efficient way of computing differential in multiple-layer neural network (backpropation algorithm)

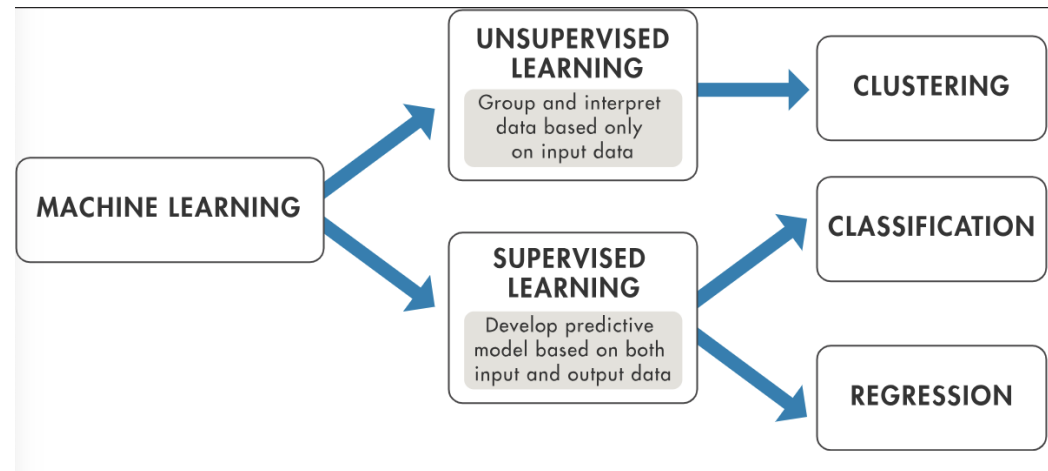
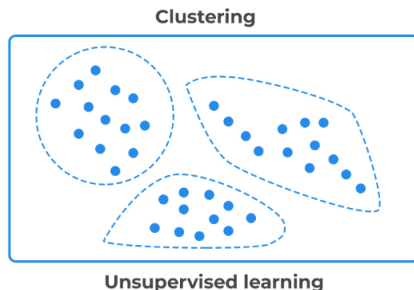
Informal definition of ML

“Machine learning is essentially a form of **applied statistics** with increased emphasis on the use of computers to **statistically estimate complicated functions** and a **decreased emphasis on proving confidence intervals** around these functions.”

--- Ian Goodfellow et al., Deep Learning, MIT Press, 2016

Supervised learning vs. Unsupervised learning

- **Supervised** learning utilizes **labeled** datasets to make predictions.
 - The predication has two types:
 - Classification (predicting **discrete** values)
 - Regression (predicting **continuous** values).
- **Unsupervised** learning: doesn't require labeled datasets,
- it detects patterns in the data



More abstract definition of (un-)supervised learning

- Unsupervised learning involves observing several examples of a random vector \mathbf{x} ,
 - then attempting to learn the probability distribution $\mathbf{p}(\mathbf{x})$
- Supervised learning involves observing several examples of a random vector \mathbf{x} and an associated value or vector \mathbf{y} ,
 - then learning to predict \mathbf{y} from \mathbf{x} , usually by estimating $\mathbf{p}(\mathbf{y} | \mathbf{x})$.
- In both unsupervised learning and supervised learning, the training dataset is **fixed** during the training process.
- In the **reinforcement learning**, new training data are produced by the algorithm itself.
- **Deep learning** can be applied to all the above algorithms.

[Published: 27 January 2016](#)

Mastering the game of Go with deep neural networks and tree search

[David Silver](#) , [Aja Huang](#), ... [Demis Hassabis](#) 

+ Show authors

[Nature](#) **529**, 484–489 (2016) | [Cite this article](#)

415k Accesses | **6010** Citations | **3058** Altmetric | [Metrics](#)

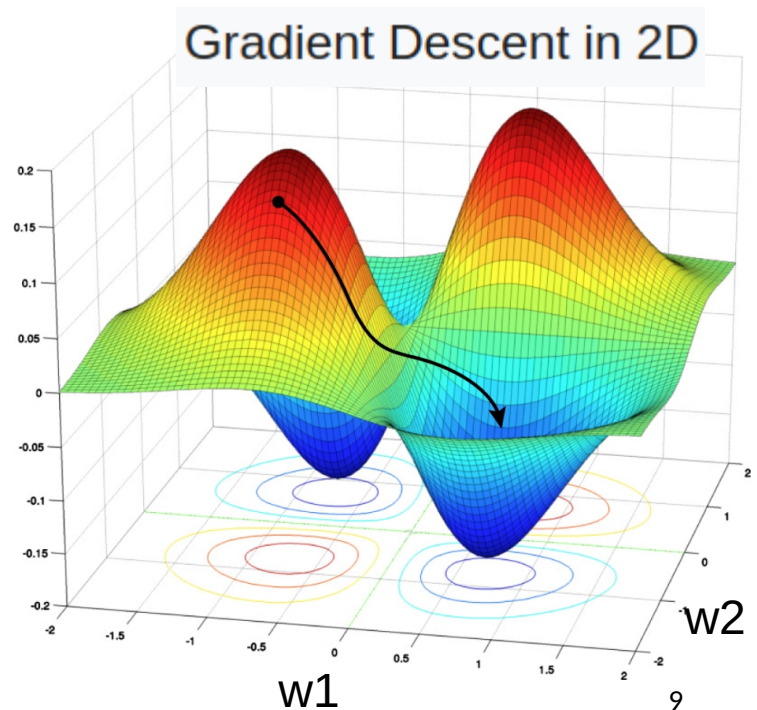
The deep neural networks are trained by a combination of **supervised learning** (from human expert games) and **reinforcement learning** (from games of self-play).

But what is “learning”?

- Learning/Traning ==> a gradual process of improving
- Supervised ML==> a gradual process of **adjusting parameters** to get **better** answers
- Q: How to judge if an answer is better?
 - A: Define a loss/cost function
- Q: How to adjust parameters?
 - A: Gradient descent
- ==>Reach Local minimal
- and hopefully global minimal of the loss function

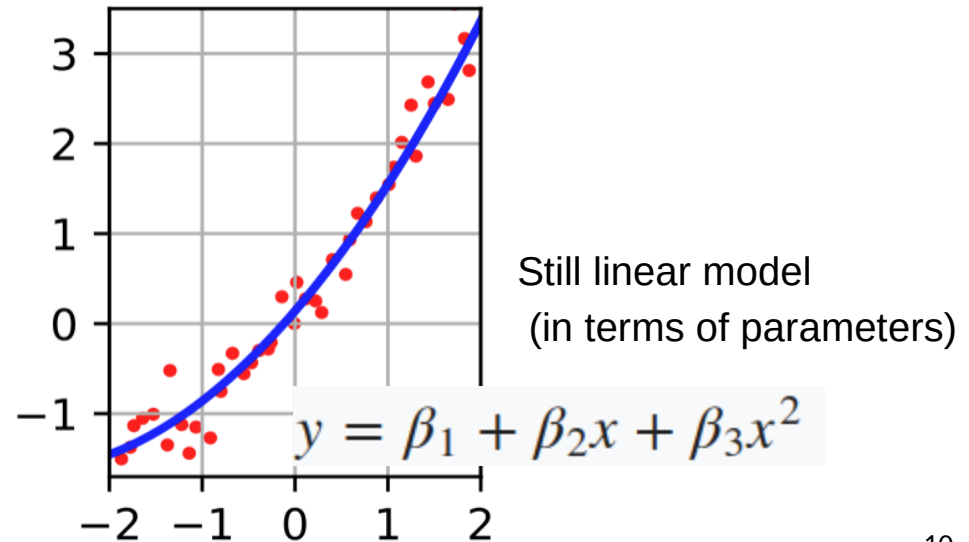
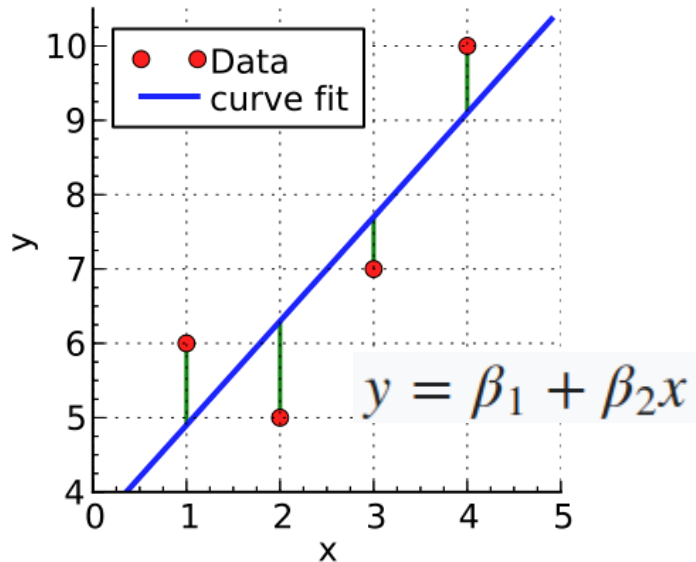
$$\mathbf{w}^{(n+1)} = \mathbf{w}^{(n)} - \gamma \nabla C(\mathbf{w}^{(n)})$$

learning rate $\gamma \in \mathbb{R}_+$



Linear least squares is a machine learning algorithm?

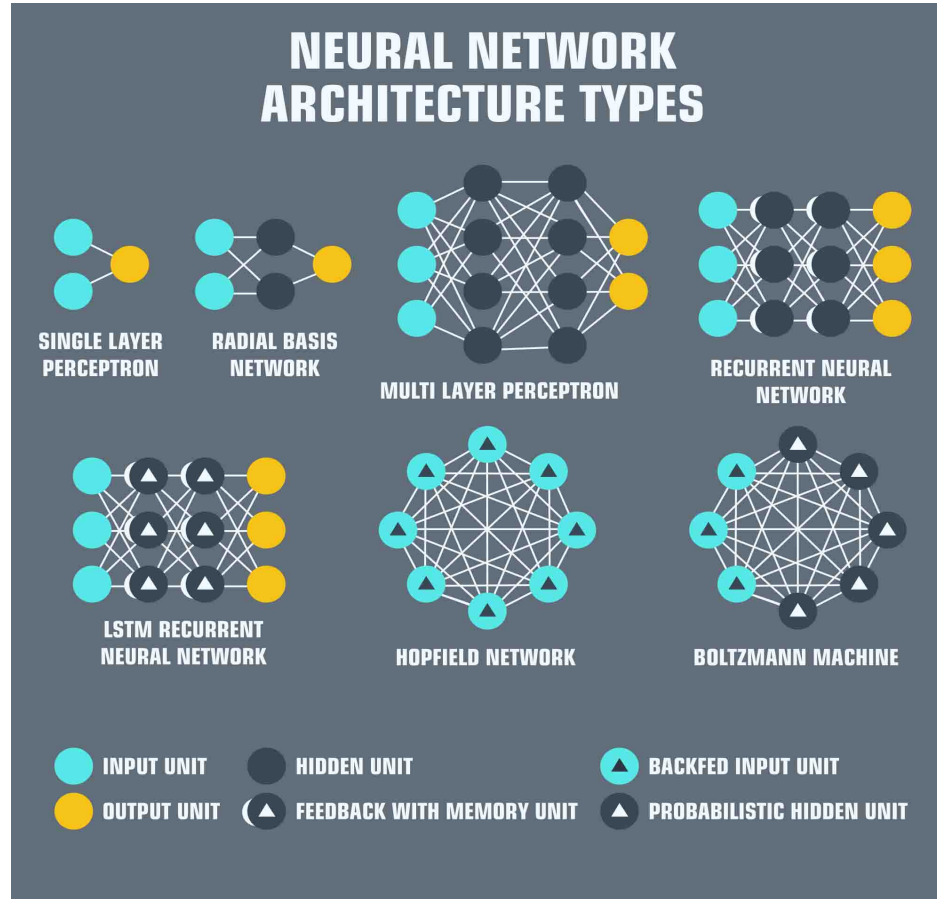
- Yes, it is.
- Q: Where is the training process?
 - A: There exist closed expressions for the fitting parameters.
 - We do not need to use **iterative** methods to **learn** the parameters.



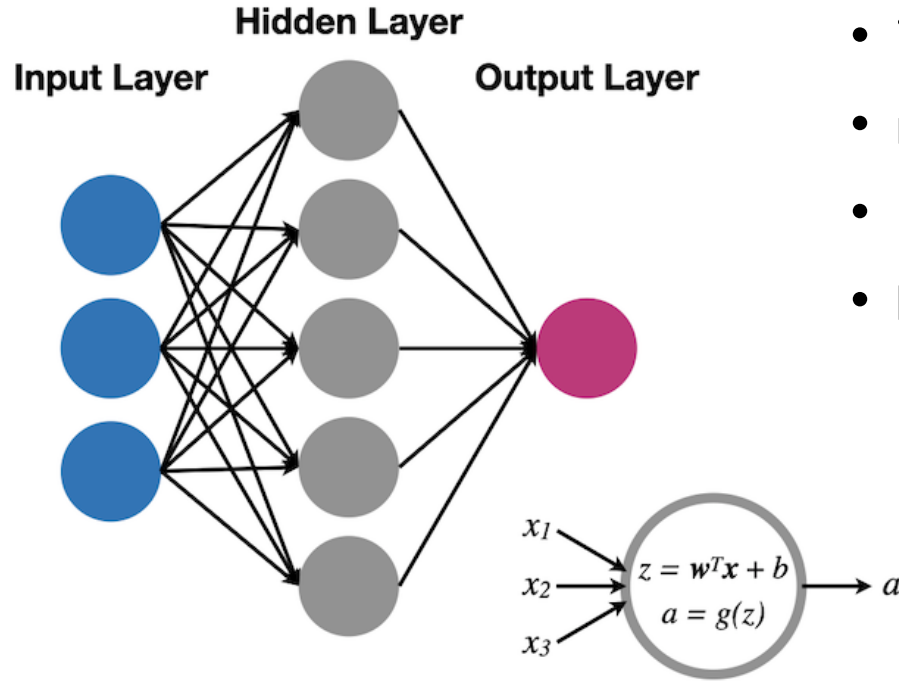
Deep learning models

Deep feedforward networks, also called multilayer perceptrons, are the quintessential deep learning models.

Many kinds of networks



Feedforward neural networks



- information moves in only one direction
- no cycles or loops
- No intra-layer connections
- Fully connected

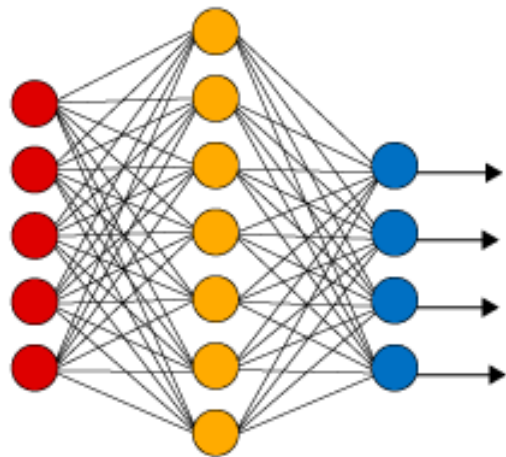
Information propagation correspond to function composition:

$$f(\mathbf{x}) = f^{(3)}(f^{(2)}(f^{(1)}(\mathbf{x})))$$

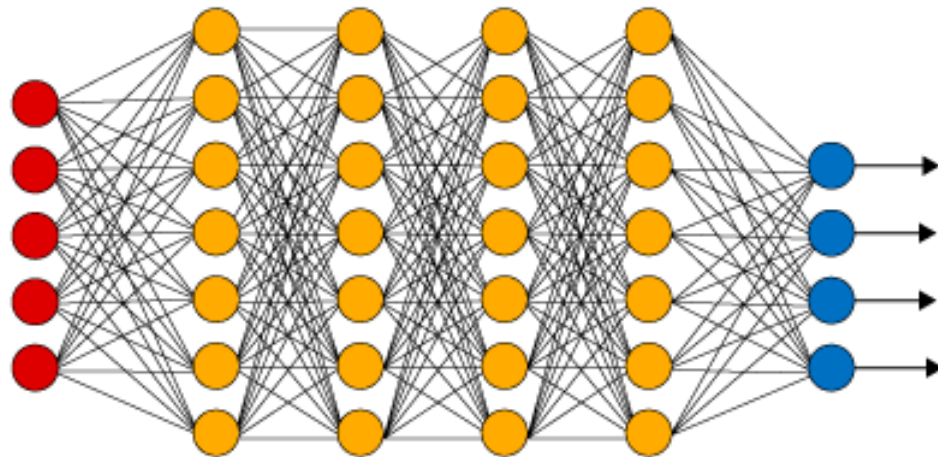
Q: “Deep” in deep learning?

A: The adjective "deep" in deep learning refers to the use of multiple layers in the neural network

Simple Neural Network



Deep Learning Neural Network

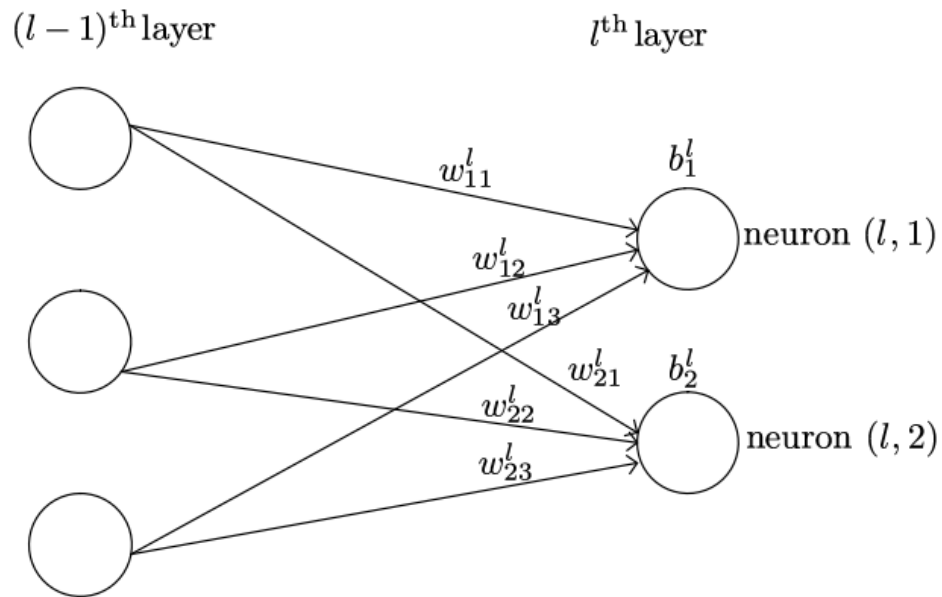


● Input Layer

● Hidden Layer

● Output Layer

Node (Neuron), weight, bias, and activation



$$z_j^l = \sum_k w_{jk}^l a_k^{l-1} + b_j^l,$$

$$a_j^l = \sigma(z_j^l).$$

σ is a function called activation function

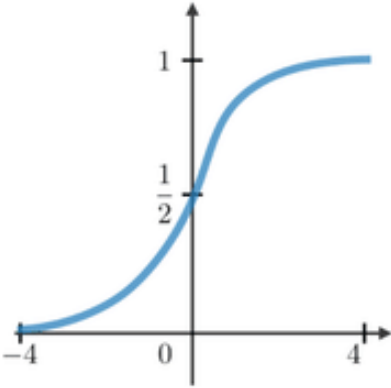
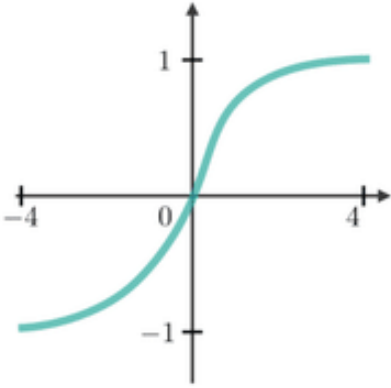
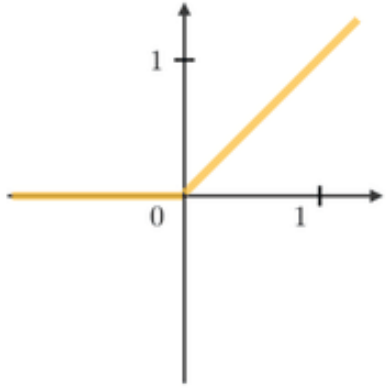
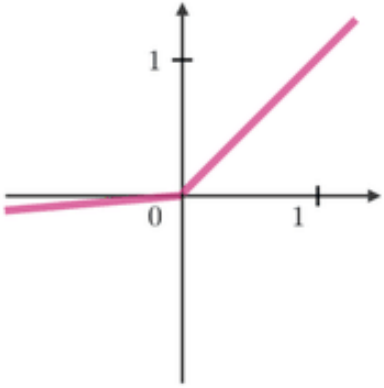
- Trainable parameter: weights, bias
- Millions of trainable parameters in a typical network
- Multilayer network is a universal function approximator

In vector form:

$$z^l = w^l a^{l-1} + b^l$$

$$a^l = \sigma(z^l).$$

Many kinds of activation functions

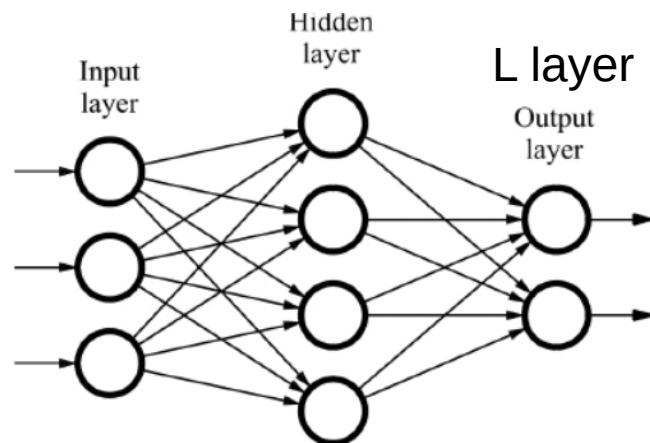
Sigmoid	Tanh	ReLU	Leaky ReLU
$g(z) = \frac{1}{1 + e^{-z}}$	$g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	$g(z) = \max(0, z)$	$g(z) = \max(\epsilon z, z)$ with $\epsilon \ll 1$
			

Before training the network, define cost function to judge answers given by the network

Define a cost (loss, objective) function by

$$C(w, b) \equiv \frac{1}{2n} \sum_x \sum_{j=1}^{N_L} (y_j(x) - a_j^L)^2,$$

where w and b denotes the collection of all weights and biases in the network, n is the total number of training examples x , the summation is over all the training examples, $y(x)$ is the desired output from the network (i.e., correct answer) when x is the input, and a^L is the actual output from the output layer of the network and is a function of w , b , and x . 1



We need the gradient of the cost function when training the network using Gradient descent method

- But how do you compute the partial derivatives of the cost function with respect to millions of variables (weights)?
- ==>computationally expensive
- An efficient method: the backpropagation algorithm

Learning representations by back-propagating errors

[David E. Rumelhart](#), [Geoffrey E. Hinton](#) & [Ronald J. Williams](#)

[Nature](#) **323**, 533–536 (1986) | [Cite this article](#)

87k Accesses | **12489** Citations | **237** Altmetric | [Metrics](#)

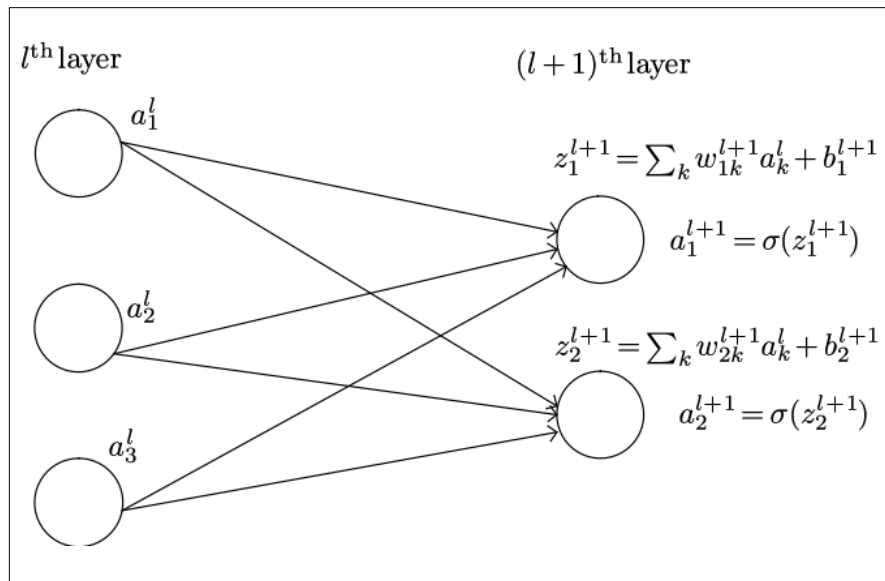
- **Backpropagation algorithm: core of Deep Learning libraries**

$$\delta_j^l \equiv \frac{\partial C_x}{\partial z_j^l},$$

$$\frac{\partial C_x}{\partial b_j^l} = \delta_j^l, \quad \frac{\partial C_x}{\partial w_{jk}^l} = \delta_j^l a_k^{l-1}.$$

$$\delta_j^L = (a_j^L - y_j(x)) \sigma'(z_j^L).$$

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l),$$



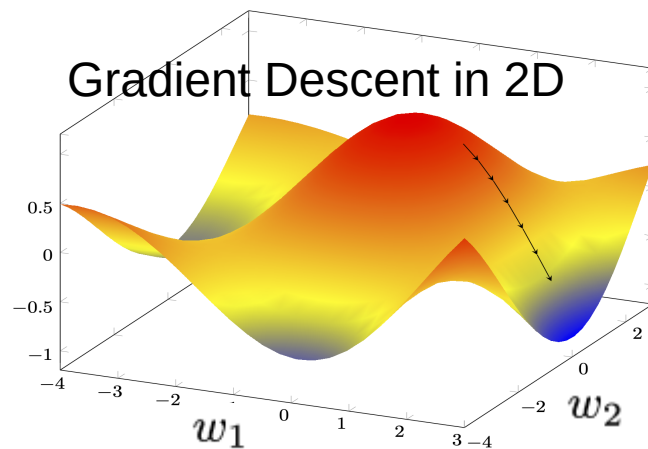
- **Computing starts from the last layer and then moves backward**
- **easy to be derived and implemented, using the chain rule**
- **but it is not obvious why it is more efficient than other methods**
- **This algorithm is in Tensorflow (google) and Pytorch (Facebook)**

- **Stochastic Gradient Descent (SGD)**

Most deep learning algorithms are based on the optimization algorithm SGD

Pseudocode of SGD:

- Choose an initial vector of parameters w and b
- Repeat until an approximate minimum is obtained:
 - Randomly shuffle samples in the training set.
 - For each mini-batch of samples x , do:
 - $w = w - \eta \nabla C_x(w, b)$.

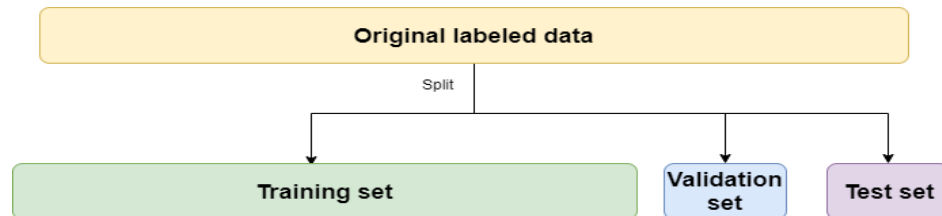
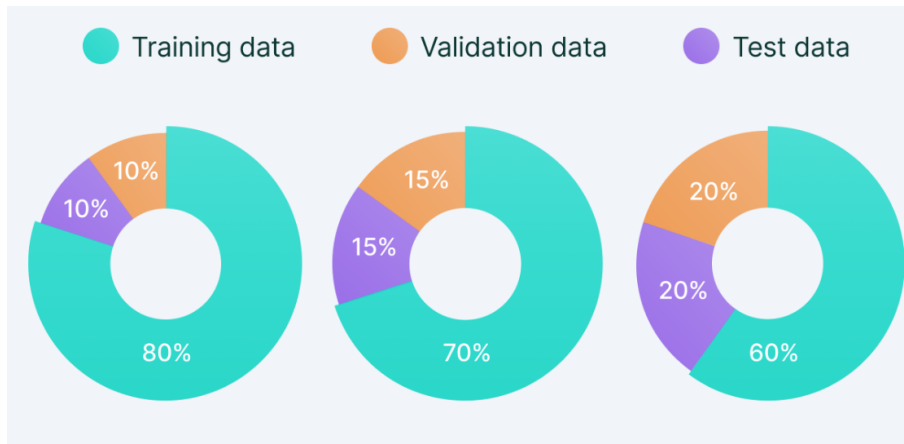


What are hyperparameters?

- Parameters chosen by users (not learned by the algorithm) are called hyperparameters
- are constant, values are set before the learning process begins.
 - Hyperparameters in deep learning:
 - Type of activation functions
 - Number of hidden layers,
 - Number of nodes in a layer
 - Learning rate
 - Batch size in SGD

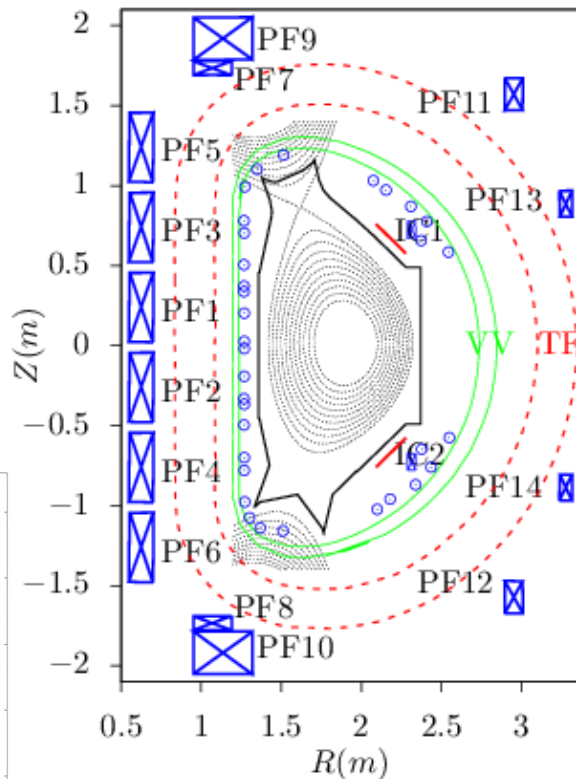
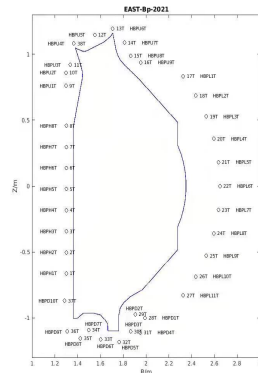
Training, validation, testing dataset

- Full dataset is separated to 3 subsets:
 - Training----directly used by the training algorithm
 - Validation---- used to guide the selection of hyperparameters
 - Testing---- not seen by the training algorithm.
- Typical ratio between the 3 subsets: 80% : 10% : 10%



Deep learning for magnetic equilibrium reconstruction

- Input: total 82 values per time slice
 - Equilibrium magnetic probe measurements (34 points)
 - Flux loop measurements (35 points)
 - PF coil currents (12 values)
 - Plasma current (1 value)
- Output: Pooidal flux values on 33x33 grids (1089 values)
- Data are read from EAST MdSplus server using Python

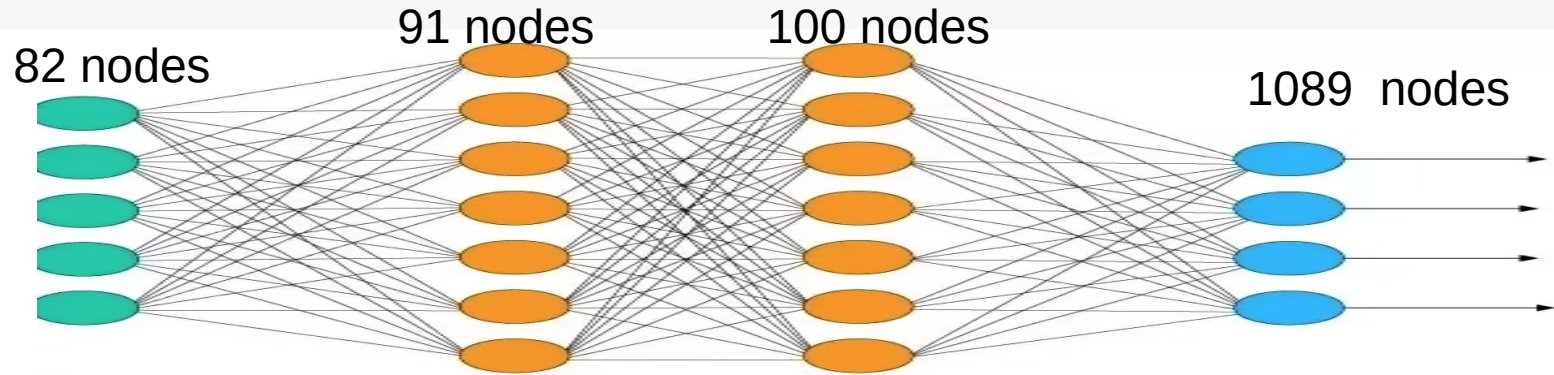


Circles: Flux loop

Magnetic probes

Deep learning network architecture used

```
model = keras.Sequential(  
    [  
        layers.Dense(units=91, activation="relu", kernel_initializer='random_normal',  
            bias_initializer='zeros', input_shape=(82,)),  
  
        layers.Dense(units=100, activation="relu", kernel_initializer='random_normal',  
            bias_initializer='zeros'),  
  
        layers.Dense(units=1089, activation=None, kernel_initializer='random_normal',  
            bias_initializer='zeros'),  
    ]  
)
```




```
In [12]: model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
layer1 (Dense)	(None, 91)	7553
layer2 (Dense)	(None, 100)	9200
layer3 (Dense)	(None, 1089)	109989

Total params: 126,742

Trainable params: 126,742

Non-trainable params: 0

```
In [13]: model.compile(optimizer=tf.optimizers.SGD(learning_rate=0.01),  
                        loss="MAE")
```

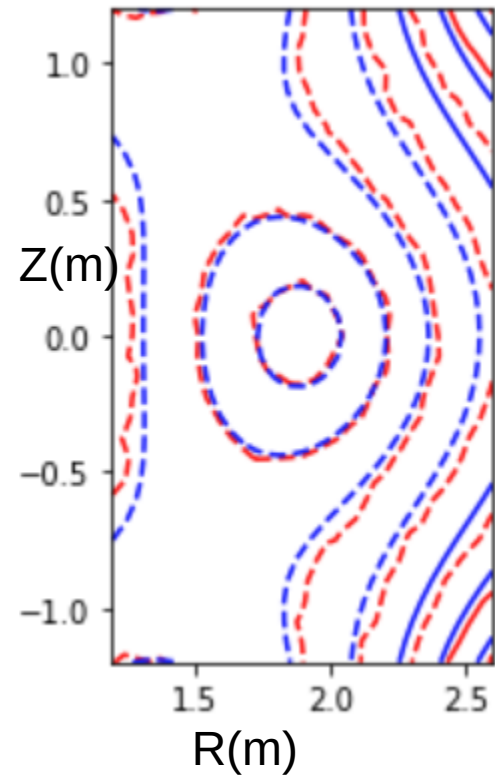
Training the model

```
In [14]: history = model.fit(features, results, epochs=5, batch_size=16)
```

Epoch 1/5
60/60 [=====] - 0s 2ms/step - loss: 4.5860
Epoch 2/5
60/60 [=====] - 0s 3ms/step - loss: 0.2272
Epoch 3/5
60/60 [=====] - 0s 3ms/step - loss: 0.2270
Epoch 4/5
60/60 [=====] - 0s 3ms/step - loss: 0.2269
Epoch 5/5
60/60 [=====] - 0s 3ms/step - loss: 0.2268

Comparison of poloidal magnetic flux between data and ML prediction

Data (blue)
Prediction (red)



```
In [18]: f, r = one_discharge(shot_number=109426)
```

```
In [20]: pred = model.predict(f)
```

- Training data from EAST #109423:
958 time slices of magnetic measurements and Ψ
- Testing data from EAST # 109426
- at an arbitrarily chosen time slice
- To demonstrate that the prototype works,
- not for demonstrating the accuracy

Thank you for your attention!

Interested in ML?

Useful tools:

Python, Jupyter-notebook, tensorflow

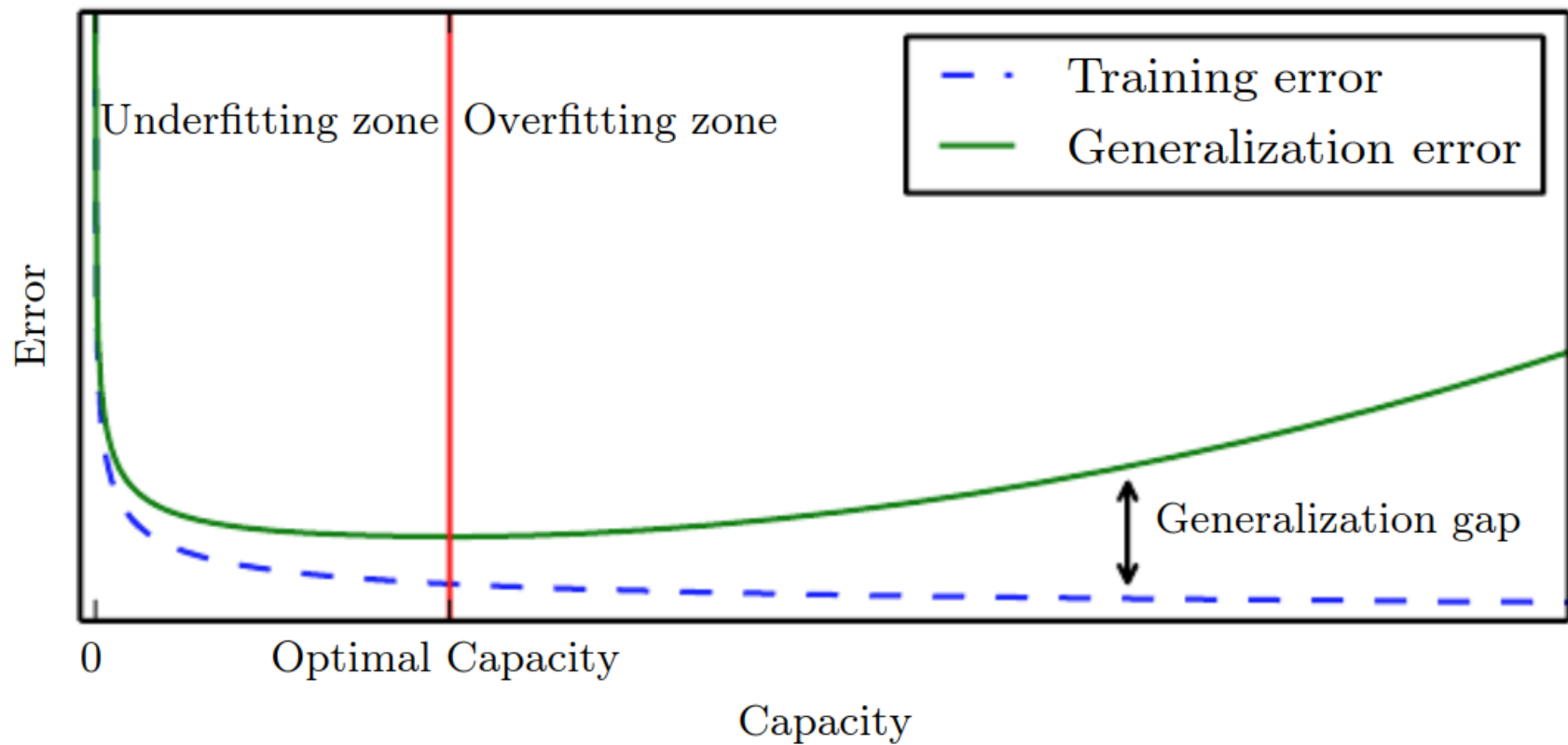
This means that the goal of machine learning research is not to seek a universal learning algorithm or the absolute best learning algorithm. Instead, our goal is to understand what kinds of distributions are relevant to the “real world” that an AI agent experiences, and what kinds of machine learning algorithms perform well on data drawn from the kinds of data-generating distributions we care about.

A feedforward network defines a mapping $y=f(x;w, b)$ and learns the value of the parameters (w,b) that result in the best function approximation.

These models are called feedforward because information flows through the function being evaluated from x , through the intermediate computations used to define f , and finally to the output y . There are no feed back connections in which outputs of the model are fed back into itself. When feedforward neural networks are extended to include feedback connections, they are called **recurrent neural networks**

The central challenge in machine learning is that our algorithm must perform well on new, previously unseen inputs—not just those on which our model was trained. The ability to perform well on previously unobserved inputs is called generalization

Typically, when training a machine learning model, we have access to a training set; we can compute some error measure on the training set, called the training error; and we reduce this training error. So far, what we have described is simply an optimization problem. What separates machine learning from optimization is that we want the generalization error, also called the test error, to be low as well.



The ideal model is an oracle that simply knows the true probability distribution that generates the data.

Even such a model will still incur some error on many problems, because there may still be some noise in the distribution.

In the case of supervised learning, the mapping from x to y may be inherently stochastic, or y may be a deterministic function that involves other variables besides those included in x .

The error incurred by an oracle making predictions from the true distribution $p(x, y)$ is called the Bayes error.

Linear models, such as logistic regression and linear regression, are appealing because they can be fit efficiently and reliably, either in closed form or with convex optimization.

Unsupervised learning and supervised learning are not formally defined terms. The lines between them are often blurred.