Artificial neural networks

Olga Razuvaeva[1,2], Sergey Korpachev[3,4] and Stepan Zakharov[5]

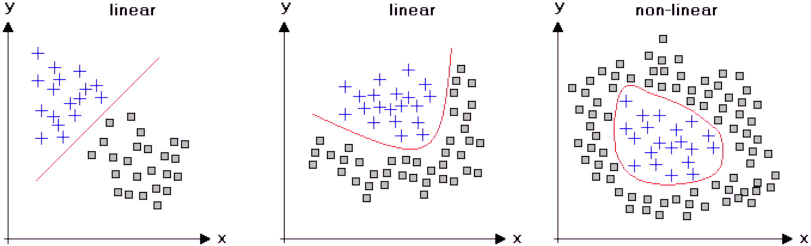
[1]Institute for Theoretical and Experimental Physics
[2]National Research Nuclear University MEPhI (Moscow Engineering Physics Institute)
[3]Moscow Institute of Physics and Technology
[4]Lebedev Physical Institute of the Russian Academy of Sciences
[5]Novosibirsk State University

?????Outline?????

- Non-linearity in data
- Feature extraction
- ANN structure
- How to train?
- Deep Learning
- The Achilles heel of the DL models
- Evolution of GD
- Reaching stability
- Network architectures

Modelling nonlinearities

Modelling nonlinearities — they exist



Do you know how to describe the non-linear data in the right picture?
Linear model, seriously?
!!!!!Here goes picture of linear vs non-linear data!!!!!

Modelling nonlinearities

linear models

- The linear model does not describe complex nonlinear data
- A combination of linear models is also a linear model

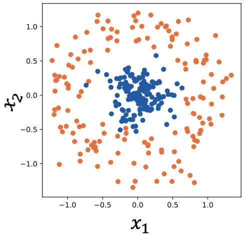
!!!!!Explanation why linear models don't fit them out-of-the box!!!!!

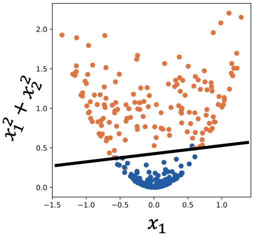
Modelling nonlinearities

- Trees were designed to approximate nonlinearities
- They do a pretty good job in it
- They are fast and interpretable
- But they are just "brute-force" algorithms
- They don't infer symmetries in data by design
- ad-hoc, cut-based and piecewise approximations of data at hand
- Hence, not differentiable and smooth

Modelling nonlinearities

feature engineering





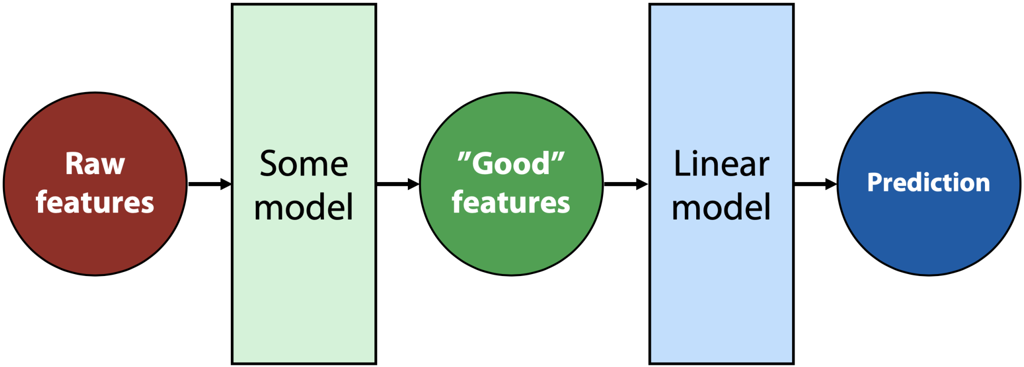
- But sometimes we know *a priori* that there are transformations simplifying the problem for our model
- So that even linear models can do the job
- But this **feature engineering** is non-trivial, requires domain knowledge and is time-consuming

What if we design a model which could automatically feature-engineer itself?

Building the beast

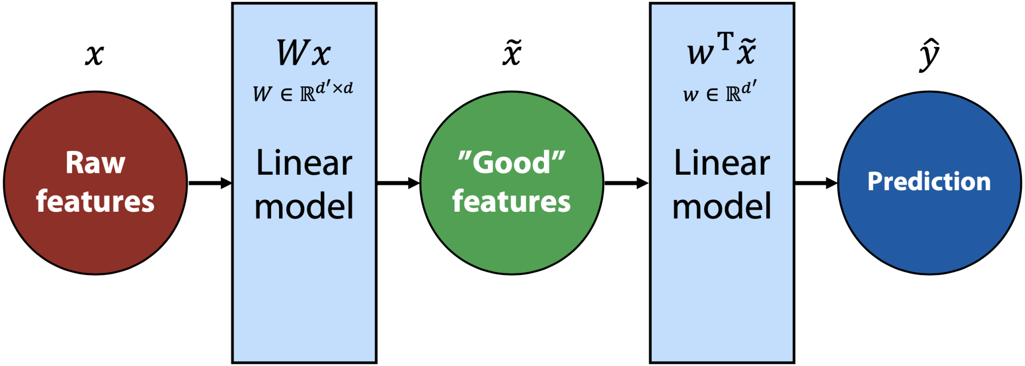
Neural Network

automating FE



Neural Network

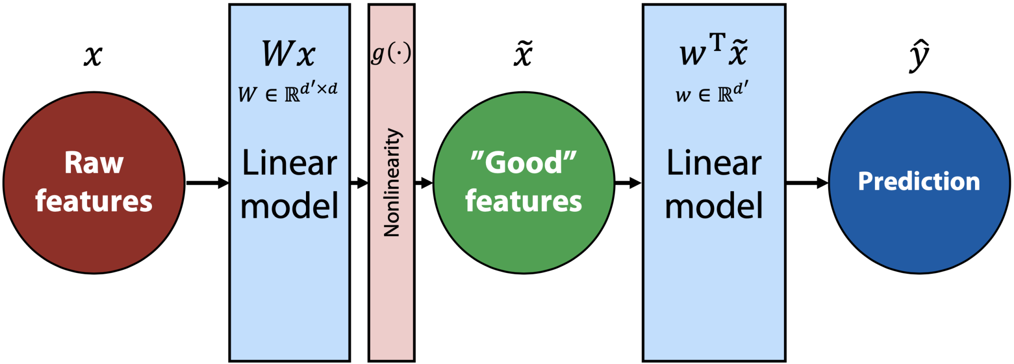
automating FE



$\hat{y} = w^T \cdot \tilde{x} = w^T \cdot (W \cdot x) = (w^T \cdot W) \cdot x = w'^T \cdot x \Rightarrow$ it is still a linear model

Neural Network

automating FE



$$\hat{y} = w^T \cdot \tilde{x} = w^T \cdot g(W \cdot x),$$

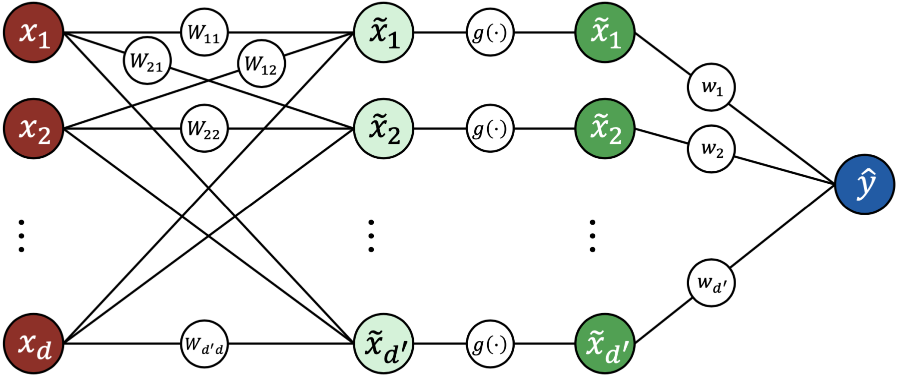
where $g(\cdot)$ some nonlinear scalar function (applied elementwise)

Neural Network

automating FE

adding nonlinearity
¡pic from slide 12 mlhep¿

Neural Network — architecture

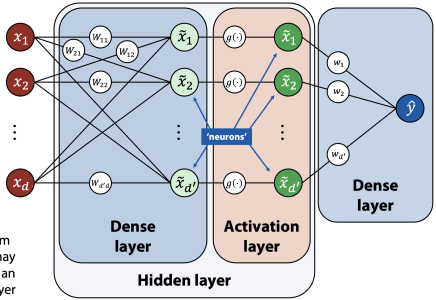


$$\hat{y} = w^T \cdot \tilde{x} = w^T \cdot g(W \cdot x) = \sum_{j=1} \left[w_j \cdot g\left(\sum_i W_{ji} \cdot x_i\right)\right]$$

Neural Network

terminology

Feed-forward network:

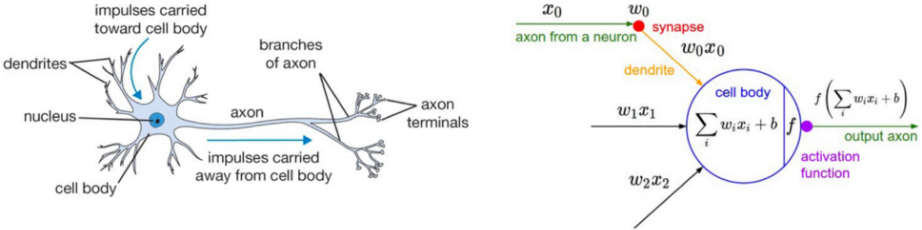


Note: the term "activation" may also stand for an output of a layer

- red nodes (vertices) $x_1, x_2, ..., x_d$ - features from an input layer of the ANN
- green nodes (vertices) - neurons from a hidden layer of the ANN
- blue node (vertex) $\hat{y}$ - neuron from a output layer of the ANN
- Straight lines (edges) correspond to weights (w) between neurons
- $g(\cdot)$ - nonlinear activation function, for example, a sigmoid function
- Each layer (except the output layer) has a bias neuron: $x_{bias} = 1$

Neural Network

It follows the way of human brain processing. ANN modelled how a neuron works in the human brain.



Neurons on human brain consist of a nucleus, dendrites, cell body, and axon. The number of neurons in humans is approximately 140 billion, consist of 100 billion neurons and 40 billion synapses in neurons.

Training the beast

Training

how to train?

- NN is basically a composite linear model (with nonlinearities)
- We can use gradient descent (GD) to train
- GD algorithm:
  $\theta_{i+1} = \theta_i - \eta \cdot \nabla Q(\theta_i)$
- Do you know how to calculate the gradient of the weights?

- But gradients are hard to derive analytically
- Writing down (into NN framework) all the derivatives is tough
- The approach doesn't generalize to architectures
- But NN is just a composite model $\Rightarrow$ can use chain rule for differentiating it

- We know how to find the derivative of a simple function.
- Let's remember the rule for differentiating complex functions.

$$\frac{\partial f(t(x))}{\partial x} = \frac{\partial f(t)}{\partial t(x)} \cdot \frac{\partial t(x)}{\partial x}$$

- What is the derivative of following function with respect to x: $sin(x^2+5)$?
- What is the derivative of the sigmoid function?

Can you convert it to $\sigma'(x) = f(\sigma(x))$?

- Sigmoid function: $\sigma(x) = \dfrac{1}{1+e^{-x}}$

- We know how to find the derivative of a simple function.
- Let's remember the rule for differentiating complex functions.

$$\frac{\partial f\big(t(x)\big)}{\partial x} = \frac{\partial f(t)}{\partial t(x)} \cdot \frac{\partial t(x)}{\partial x}$$

- What is the derivative of following function with respect to x: $sin(x^2 + 5)$?
- What is the derivative of the sigmoid function?

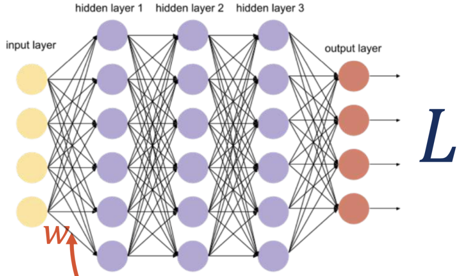
Can you convert it to $\sigma'(x) = f\big(\sigma(x)\big)$?

- Sigmoid function: $\sigma(x) = \dfrac{1}{1 + e^{-x}}$

Answer: $\sigma'(x) = \sigma(x) \cdot \big(1 - \sigma(x)\big)$

Training

weights update

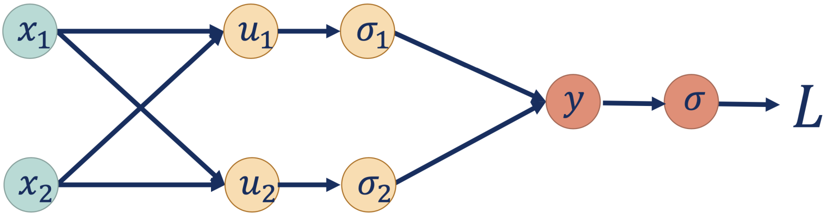


$$w^{(t+1)} = w^{(t)} - \gamma_t \boxed{\frac{\partial L}{\partial w}}$$

!!!!!also mention forward and backward passes!!!!!

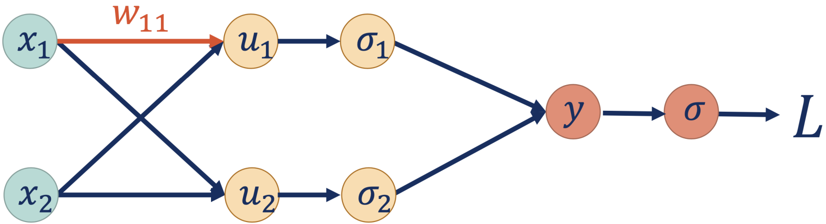
!!!!!maybe it's worth to also put ML in HEP slide first to show the general idea!!!!!

Consider a simpler neural network



Training

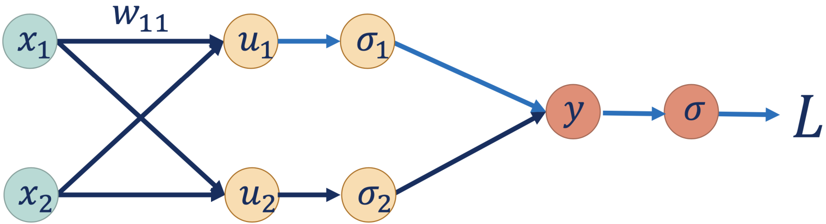
weights update



$$w_{11}^{(t+1)} = w_{11}^{(t)} - \gamma_t \frac{\partial L}{\partial w_{11}}$$

Training

back propagation

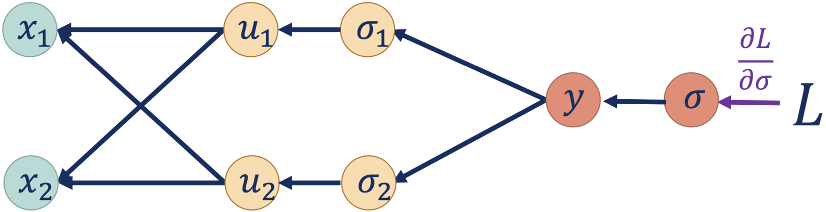


$$w_{11}^{(t+1)} = w_{11}^{(t)} - \gamma_t \frac{\partial L}{\partial w_{11}}$$

$$\boxed{\frac{\partial L}{\partial w_{11}} = \frac{\partial L}{\partial u_1}\frac{\partial u_1}{\partial w_{11}} = \frac{\partial L}{\partial u_1} x_1}$$

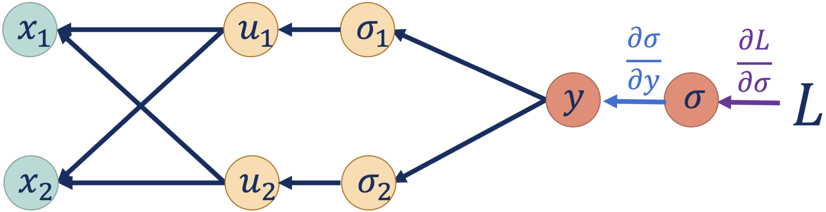
Training

back propagation



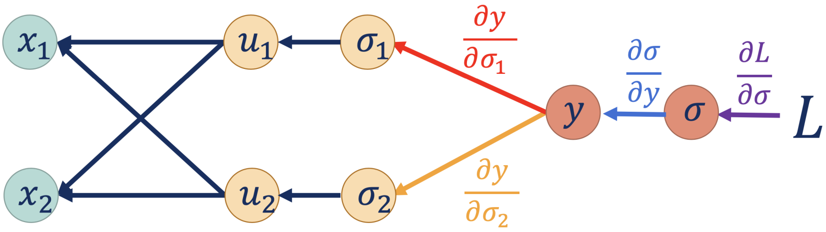
Training

back propagation



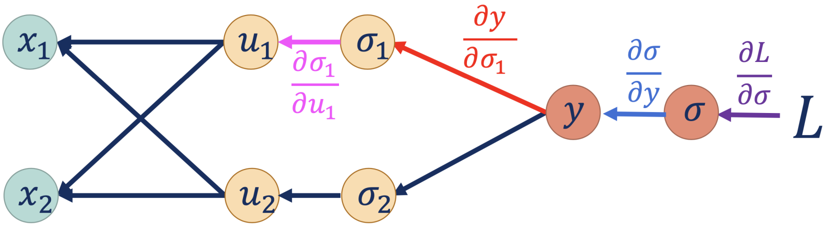
Training

back propagation



Training

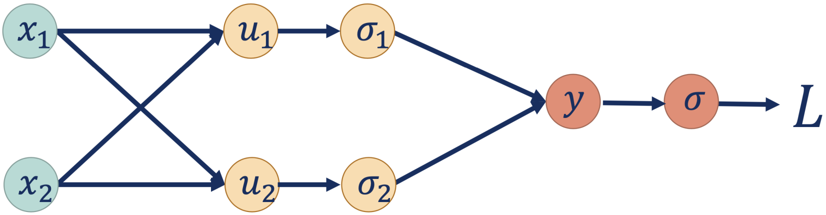
back propagation



$$\frac{\partial L}{\partial u_1} = \frac{\partial L}{\partial \sigma}\frac{\partial \sigma}{\partial y}\frac{\partial y}{\partial \sigma_1}\frac{\partial \sigma_1}{\partial u_1}$$

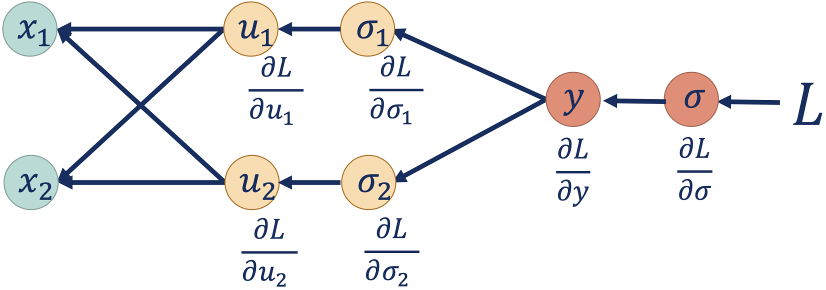
Training

forward pass



Training

backward pass



Break time

Break time

this is big brain time

- Is it necessary to use a bias neuron? Why?
- What values can the weights of neurons take?

Going deeper

Going deeper

Universal approximation theorem

Going deeper

stack more layers

in practise stacking more layers improves performance
also example of Stepan with human brain (slide 18)

slide explaining vanishing gradients

Going deeper — problems

slide explaining exploding gradients

Going deeper — problems

slide explaining vanishing gradients

Activation functions

solution: improve activation function slides for that with examples

Weight init schemas

solutions: proper weight initialization slides for that with examples

Modifying gradient descent

complexity

complicated model -¿ complicated landscape -¿ easy to stuck in local minumum
slide showing loss landscape and emphasizing the problem
then slides with modifications of SGD

Modifying gradient descent

- classical SGD

Modifying gradient descent

Momentum/Nesterov

Modifying gradient descent

- Adagrad/RMSProp/Adam

Tackling overfitting

complexity

highly complex models -¿ prone to overfitting
some pics, maybe loss landscape once again

Tackling overfitting

weight regularisation

already know that, recap slide

Tackling overfitting

- dropout

Tackling overfitting

- batchnorm

Tackling overfitting

summary