EE-559

Deep Learning

What's on today?

- Function approximation: learning complex functions from data
- Values and principles: guiding ethical, responsible AI design
- Shallow vs deep learning: layer count distinguishes learning depth
- Building a deep network: your first 'deep' network
- Network diagram: to visualize nodes, layers and parameters
- Activation function: the non-linearities in the composition
- Loss function: training signal to optimize the parameters
- Exercises: hands-on practice to solidify the above concepts

Function approximation

$$y = f(x)$$

$$y = f(x; \Theta)$$

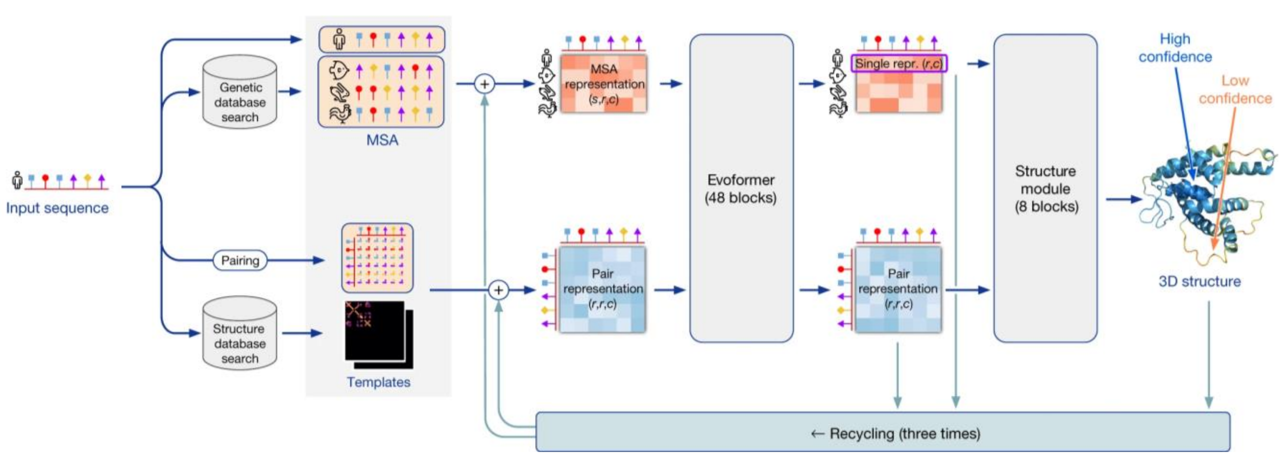
1,000,000,000,000

$$\Theta^* = \arg\min_{\Theta} L(\Theta)$$

$$\{x_i, y_i\}_{i=1}^{N}$$

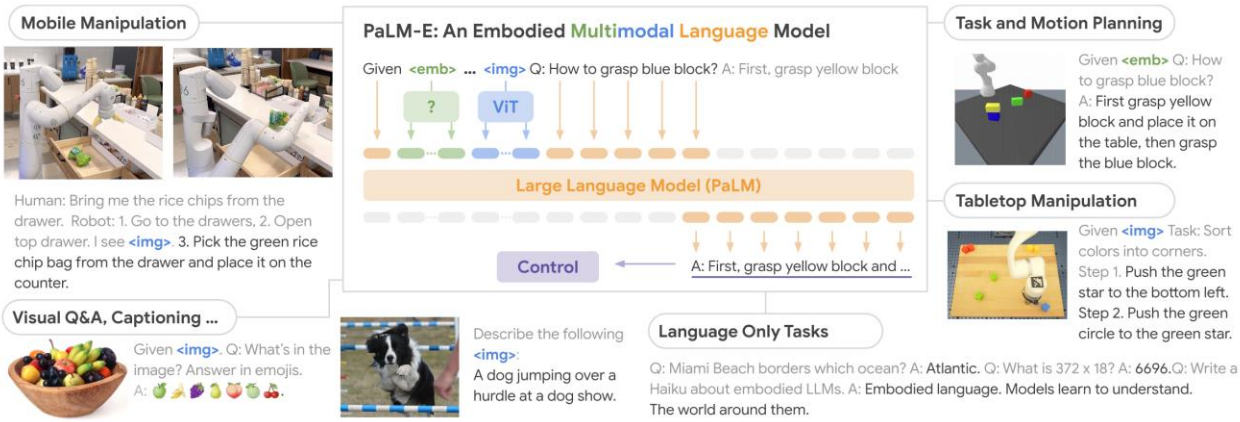
$$y = f(x)$$

Protein structure prediction



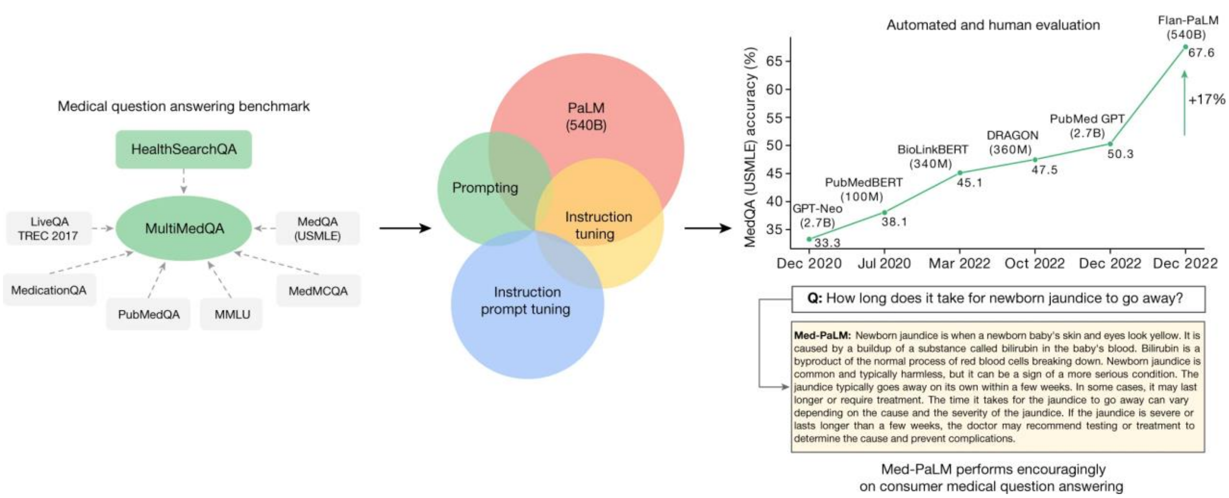
Nature **volume 596**, pages 583–589 (2021)

Embodied multimodal language model



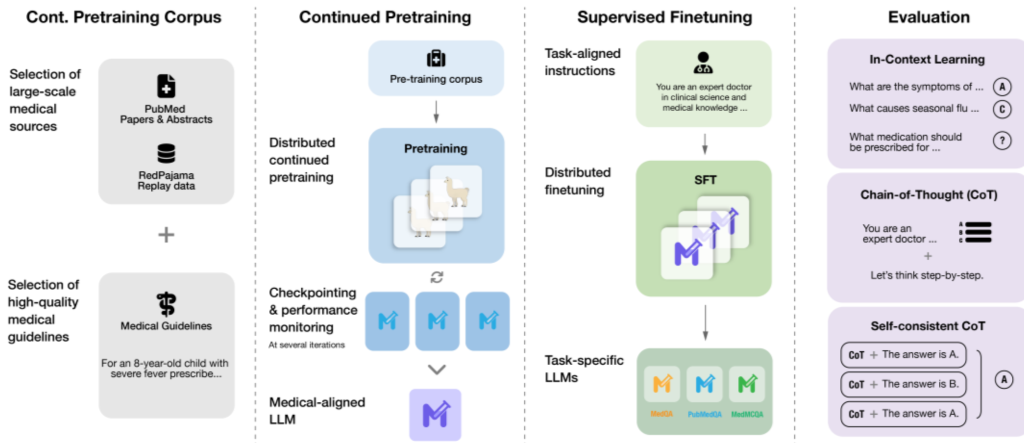
arXiv:2303.03378

Encoding clinical knowledge



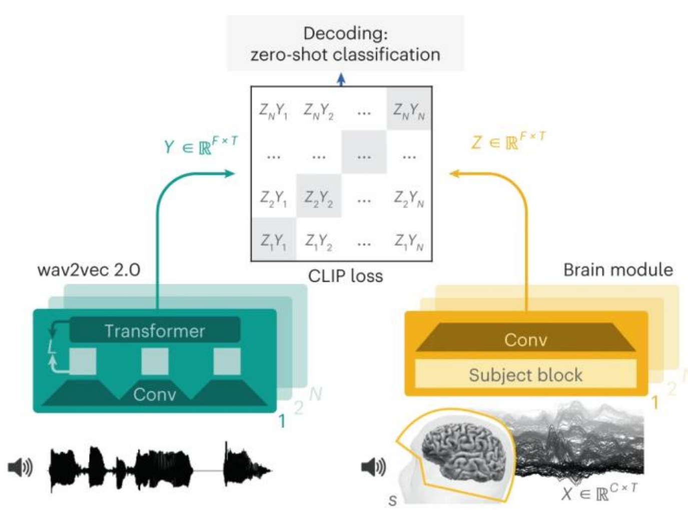
Nature volume 620, pages 172–180 (2023)

Meditron: medical Large Language Models



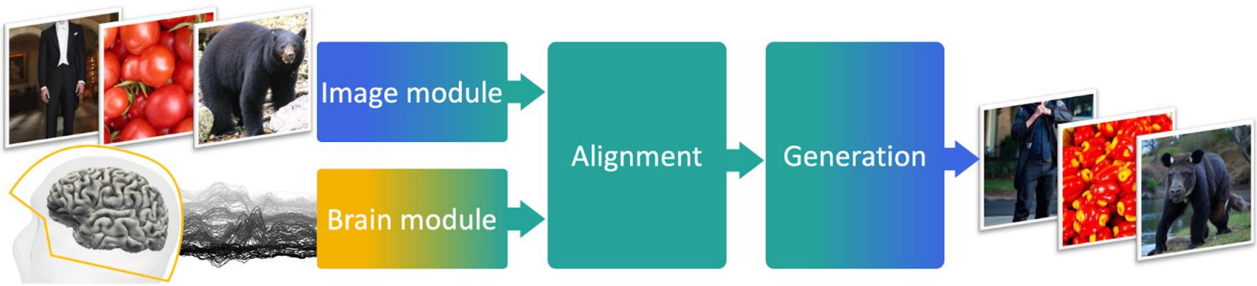
arXiv:2311.16079

Decoding speech from brain activity



Nature Machine Intelligence **volume 5**, pages 1097–1107 (2023)

Decoding images from brain activity



arXiv:2310.19812

Decoding images from brain activity

arXiv:2310.19812

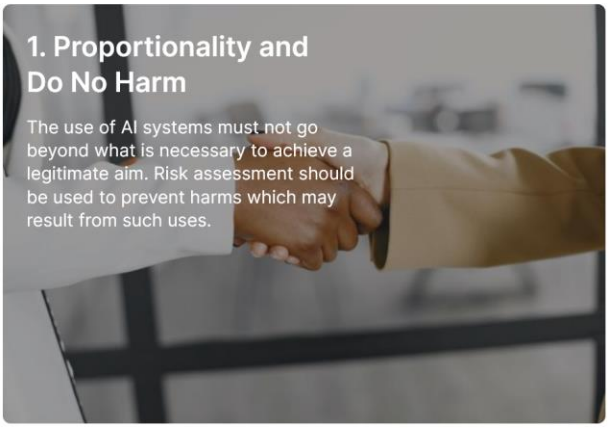
How do we know if an AI model is good for society?

Core values

- Human rights & human dignity
- Ensuring diversity & inclusiveness
- Environment & ecosystem flourishing
- Living in peaceful, just, interconnected societies

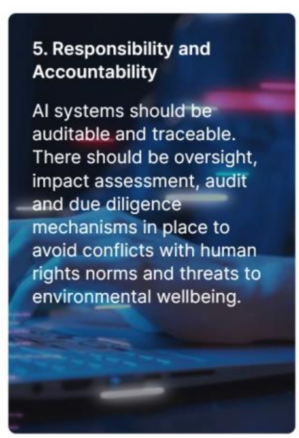
https://www.unesco.org/en/artificial-intelligence/recommendation-ethics

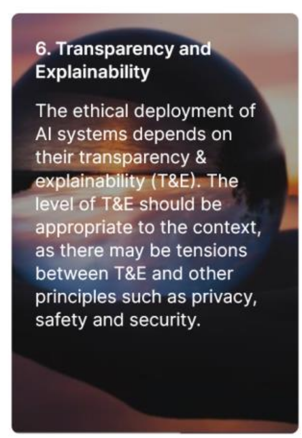
10 core principles



https://www.unesco.org/en/artificial-intelligence/recommendation-ethics

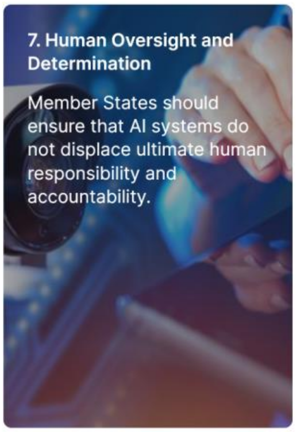
10 core principles





https://www.unesco.org/en/artificial-intelligence/recommendation-ethics

10 core principles



https://www.unesco.org/en/artificial-intelligence/recommendation-ethics

slido

How would you incorporate values and principles in a deep learning model?

ⓘ Start presenting to display the poll results on this slide.

Building a deep network

Back to the mapping

$y = f(x; \Theta)$ family of possible relationships between x and y defined by the parameters Θ

$\{x_i, y_i\}_{i=1}^{N}$ training dataset (supervised learning, N pairs)

$$L(\Theta) = \sum_{i=1}^{N} (f(x_i; \Theta) - y_i)^2$$ loss or cost function

$$\Theta^* = \arg\min_{\Theta} L(\Theta)$$

Concepts:
Inference, training / learning, loss, optimization, supervised learning

Training & testing

- **Training, learning, model fitting**:
  the process of finding the parameters Θ^* *that minimize the loss* $L(\Theta)$
- **Expressiveness of a model** f **:**
  its ability to capture the relationship between input x *and output* y
- **Testing**:
  computing the loss on separate test data to determine how well the learned model $y = f(x; \Theta^*)$ *generalizes to unseen data*

Concepts:
Generalization, under-fitting, over-fitting

Linear regression

$$y = f(x; \Theta)$$
$$= \Theta_0 + \Theta_1 x$$

$$\Theta^* = \arg\min_{\Theta} L(\Theta)$$
$$= \arg\min_{\Theta} \sum_{i=1}^{N} (f(x_i; \Theta) - y_i)^2$$
$$= \arg\min_{\Theta} \sum_{i=1}^{N} (\Theta_0 + \Theta_1 x_i - y_i)^2$$

Concept: 1D linear regression represents the input-to-output relationship as a line

Neural network

$$y = f(x;\Theta)$$
$$= \Theta_0 + \Theta_1 a[\underbrace{\Theta_{10} + \Theta_{11}x}_{\text{linear function}}] + \Theta_2 a[\underbrace{\Theta_{20} + \Theta_{21}x}_{\text{linear function}}] + \Theta_3 a[\underbrace{\Theta_{30} + \Theta_{31}x}_{\text{linear function}}]$$

activation function

$$a[z] = \begin{cases} 0 & z < 0 \\ z & z \geq 0 \end{cases}$$

Rectified linear unit (ReLU)

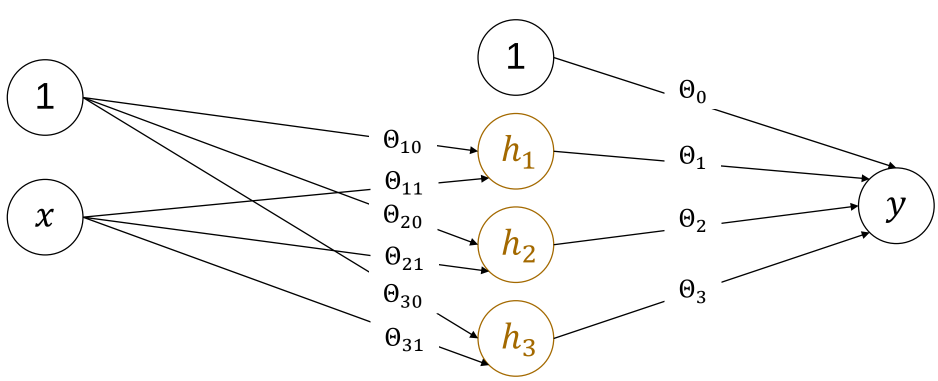
$$\Theta^* = \arg\min_{\Theta} L(\Theta)$$

Optimization to determine the 10 parameters

Concept:
Family of continuous piecewise linear functions with up to four linear regions

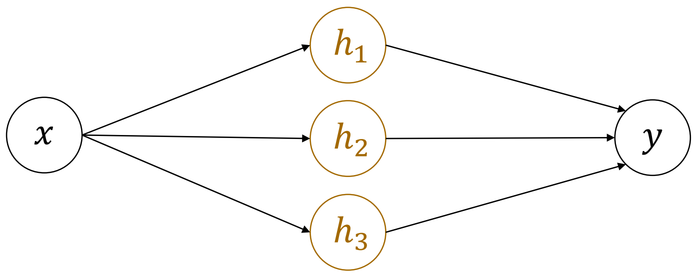
Neural network

$$y = f(x;\Theta)$$
$$= \Theta_0 + \Theta_1 \overbrace{a[\Theta_{10} + \Theta_{11}x]}^{h_1} + \Theta_2 \overbrace{a[\Theta_{20} + \Theta_{21}x]}^{h_2} + \Theta_3 \overbrace{a[\Theta_{30} + \Theta_{31}x]}^{h_3}$$



Neural network: simplified representation

$$y = f(x; \Theta)$$
$$= \Theta_0 + \Theta_1 \overbrace{a[\Theta_{10} + \Theta_{11}x]}^{h_1} + \Theta_2 \overbrace{a[\Theta_{20} + \Theta_{21}x]}^{h_2} + \Theta_3 \overbrace{a[\Theta_{30} + \Theta_{31}x]}^{h_3}$$



Concepts: Shallow neural network, hidden units, feed-forward architecture

Shallow network with capacity D

$$h_d = a[\Theta_{d0} + \Theta_{d1}x]$$

hidden unit

$$y = \Theta_0 + \sum_{d=1}^{D} \Theta_d h_d$$

$D + 1$: linear regions

the larger D,
the higher the descriptive power of the network

Universal approximation theorem
With enough suitably chosen hidden units (basis functions, linear regions), we can describe (approximate) any (non-linear) continuous function on a compact region to any arbitrary accuracy

Concepts:
Network capacity, linear regions, basis functions, approximation of any continuous function

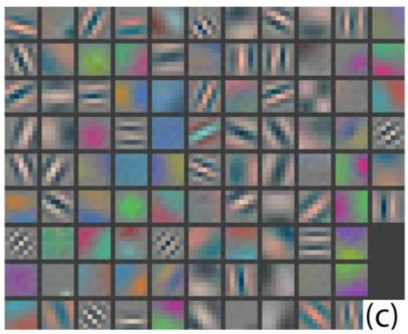
So, why isn't this course called Wide Learning?

Deep vs Wide Learning

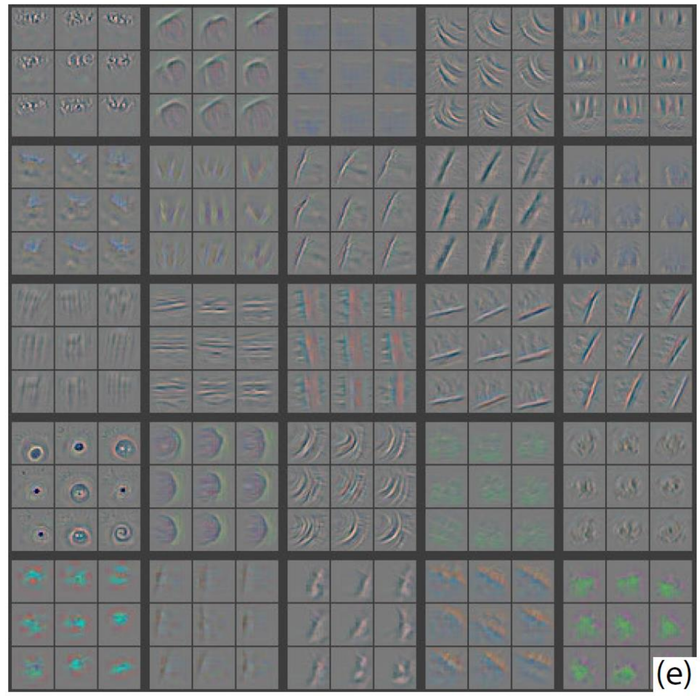
- **Shallow networks** (SNs, "Wide Learning")
  - the number of *basis functions* grows rapidly with the dimensionality of x
  - for some functions, SNs require an impractically large number of *hidden units*
- **Deep networks** (DNs)
  - can produce many more *linear regions* than SNs
  - can describe a broader family of functions
  - multiple layers of learnable parameters
    - hierarchical representation → compositional inductive bias
    - e.g. combination of **low-level features** into **higher level features**

Example

1st layer



2nd layer



arXiv:1311.2901 (fig. 7)

Deep vs shallow networks

For a given number of parameters, the number of regions the input space is divided into is
- **exponential** in the **depth** of the network and
- **polynomial** in the **width** of the hidden layers

SN	$D > 2$ **hidden units**	up to $D + 1$ **linear regions** defined by $3D + 1$ **parameters**
DN	K **layers** of $D > 2$ **hidden units**	up to $(D + 1)^2$ **linear regions** defined by $3D + 1 + (K - 1)D(D + 1)$ **parameters**

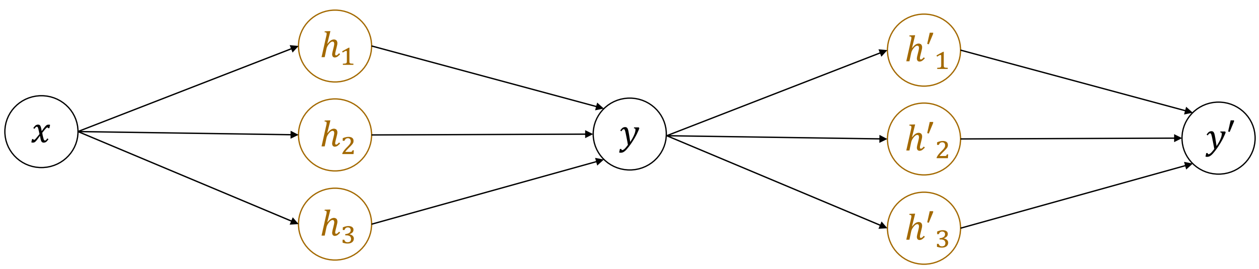
A **SN** needs exponentially more hidden units than a **DN** for an equivalent approximation

(for some functions)

Composition of two shallow networks

$$y = \Theta_0 + \Theta_1 h_1 + \Theta_2 h_2 + \Theta_3 h_3$$

$$y' = \Theta'_0 + \Theta'_1 h'_1 + \Theta'_2 h'_2 + \Theta'_3 h'_3$$ where $\boxed{h'_d = a[\Theta'_{d0} + \Theta'_{d1} y]}$



EE-559

Deep Learning

Break

we will start again at 9.15 am

1,000,000,000,000

My first ‘deep’ network

$$h'_1 = a[\Theta'_{10} + \Theta'_{11}y] \;= a[\Theta'_{10} + \Theta'_{11}(\Theta_0 + \Theta_1 h_1 + \Theta_2 h_2 + \Theta_3 h_3)]$$

$$= a[\Theta'_{10} + \Theta'_{11}\Theta_0 + \Theta'_{11}\Theta_1 h_1 + \Theta'_{11}\Theta_2 h_2 + \Theta'_{11}\Theta_3 h_3]$$

$$h'_2 = a[\Theta'_{20} + \Theta'_{21}y] \;= a[\Theta'_{20} + \Theta'_{21}(\Theta_0 + \Theta_1 h_1 + \Theta_2 h_2 + \Theta_3 h_3)]$$

$$= a[\Theta'_{20} + \Theta'_{21}\Theta_0 + \Theta'_{21}\Theta_1 h_1 + \Theta'_{21}\Theta_2 h_2 + \Theta'_{21}\Theta_3 h_3]$$

$$h'_3 = a[\Theta'_{30} + \Theta'_{31}y] \;= a[\Theta'_{30} + \Theta'_{31}(\Theta_0 + \Theta_1 h_1 + \Theta_2 h_2 + \Theta_3 h_3)]$$

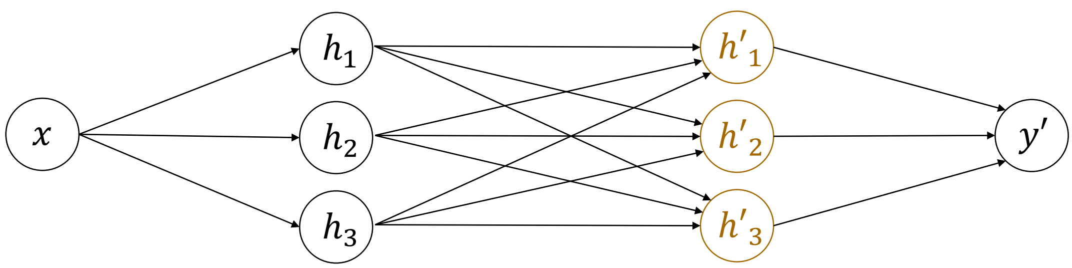
$$= a[\underbrace{\Theta'_{30} + \Theta'_{31}\Theta_0}_{\Psi'_{30}} + \underbrace{\Theta'_{31}\Theta_1}_{\Psi'_{31}} h_1 + \underbrace{\Theta'_{31}\Theta_2}_{\Psi'_{32}} h_2 + \underbrace{\Theta'_{31}\Theta_3}_{\Psi'_{33}} h_3]$$

My first ‘deep’ network

$$h'_1 = a[\Psi'_{10} + \Psi'_{11}h_1 + \Psi'_{12}h_2 + \Psi'_{12}h_3]$$
$$h'_2 = a[\Psi'_{20} + \Psi'_{21}h_1 + \Psi'_{22}h_2 + \Psi'_{22}h_3]$$
$$h'_3 = a[\Psi'_{30} + \Psi'_{31}h_1 + \Psi'_{32}h_2 + \Psi'_{33}h_3]$$

recall that:

$$h_d = a[\Theta_{d0} + \Theta_{d1}x]$$



Concepts: Network depth and width, network capacity, hyperparameters, family of family of functions mapping x to y'

Matrix notation

$$h'_1 = a[\Psi'_{10} + \Psi'_{11}h_1 + \Psi'_{12}h_2 + \Psi'_{12}h_3]$$
$$h'_2 = a[\Psi'_{20} + \Psi'_{21}h_1 + \Psi'_{22}h_2 + \Psi'_{22}h_3]$$
$$h'_3 = a[\Psi'_{30} + \Psi'_{31}h_1 + \Psi'_{32}h_2 + \Psi'_{33}h_3]$$

$$\begin{bmatrix} h'_1 \\ h'_2 \\ h'_3 \end{bmatrix} = a\left[\begin{bmatrix} \Psi'_{10} \\ \Psi'_{20} \\ \Psi'_{30} \end{bmatrix} + \begin{bmatrix} \Psi'_{11} & \Psi'_{12} & \Psi'_{13} \\ \Psi'_{21} & \Psi'_{22} & \Psi'_{23} \\ \Psi'_{31} & \Psi'_{32} & \Psi'_{33} \end{bmatrix}\begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix}\right]$$

$$\boldsymbol{h}' = a[\boldsymbol{\Psi}_0 + \boldsymbol{\Psi}\boldsymbol{h}]$$

Matrix notation

$$h_1 = a[\Theta_{10} + \Theta_{11}x]$$
$$h_2 = a[\Theta_{20} + \Theta_{21}x]$$
$$h_3 = a[\Theta_{30} + \Theta_{31}x]$$

$$\begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix} = a\left[\begin{bmatrix} \Theta_{10} \\ \Theta_{20} \\ \Theta_{30} \end{bmatrix} + \begin{bmatrix} \Theta_{11} \\ \Theta_{21} \\ \Theta_{31} \end{bmatrix} x\right]$$

$$\boldsymbol{h} = a[\boldsymbol{\Theta}_0 + \boldsymbol{\Theta}x]$$

$$y' = \Theta'_0 + \Theta'_1 h'_1 + \Theta'_2 h'_2 + \Theta'_3 h'_3$$

$$y' = \Theta'_0 + [\Theta'_1 \Theta'_2 \, \Theta'_3] \begin{bmatrix} h'_1 \\ h'_2 \\ h'_3 \end{bmatrix}$$

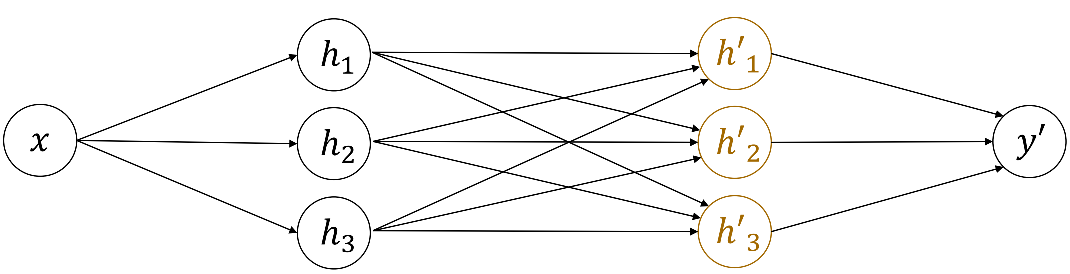
$$y' = \Theta'_0 + \boldsymbol{\Theta}'\boldsymbol{h}'$$

Matrix notation: summary

$$y' = \Theta'_0 + \boldsymbol{\Theta}'\boldsymbol{h}'$$

$$\boldsymbol{h}' = a[\boldsymbol{\Psi}_0 + \boldsymbol{\Psi}\boldsymbol{h}]$$

$$\boldsymbol{h} = a[\boldsymbol{\Theta}_0 + \boldsymbol{\Theta}x]$$



Deep learning: advantages

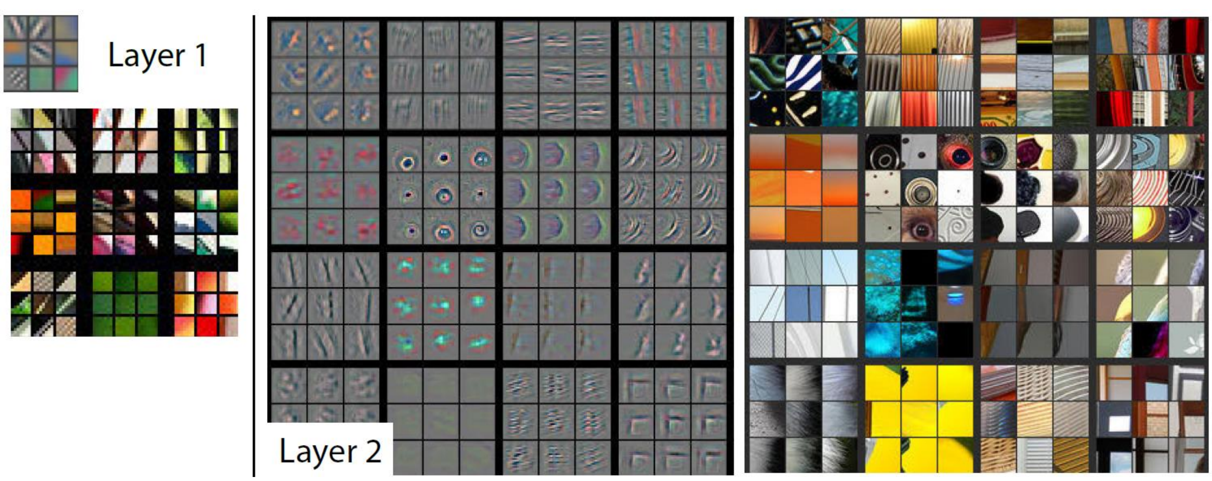
- Transformation of the data across successive layers
- Exponential gain in the number of possibilities with increased depth

- **Hierarchical** representation → compositional inductive bias
- **Distributed** representation → combination of hidden units

- **Representation learning**
  - the representation learned for a task can be useful for another
    - earlier layers: commonality of low-level features across tasks
    - later layers: more specialized to a particular task

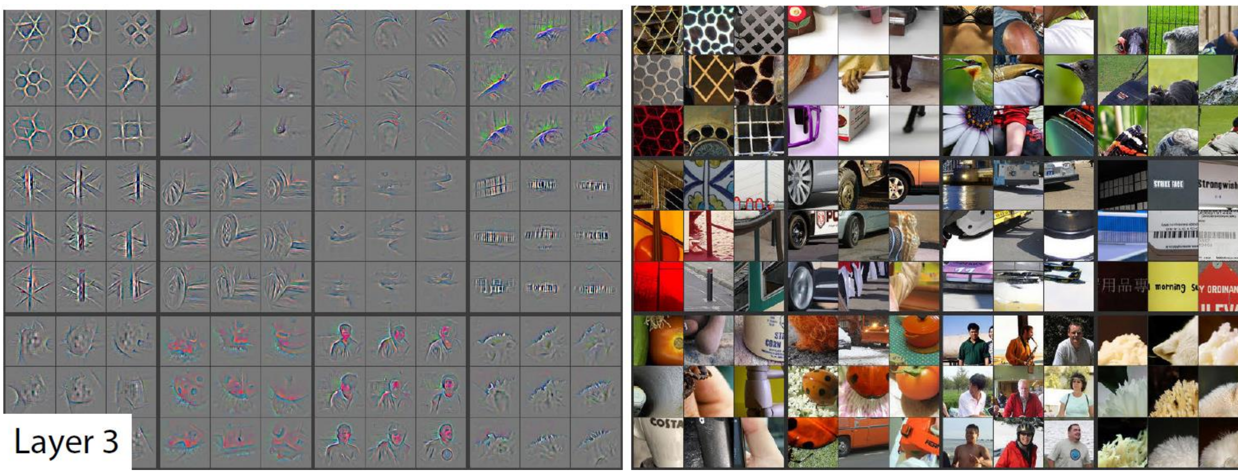
Concepts:
Embedding space, internal representation, transfer learning, pre-training, fine-tuning, multi-task learning

Example

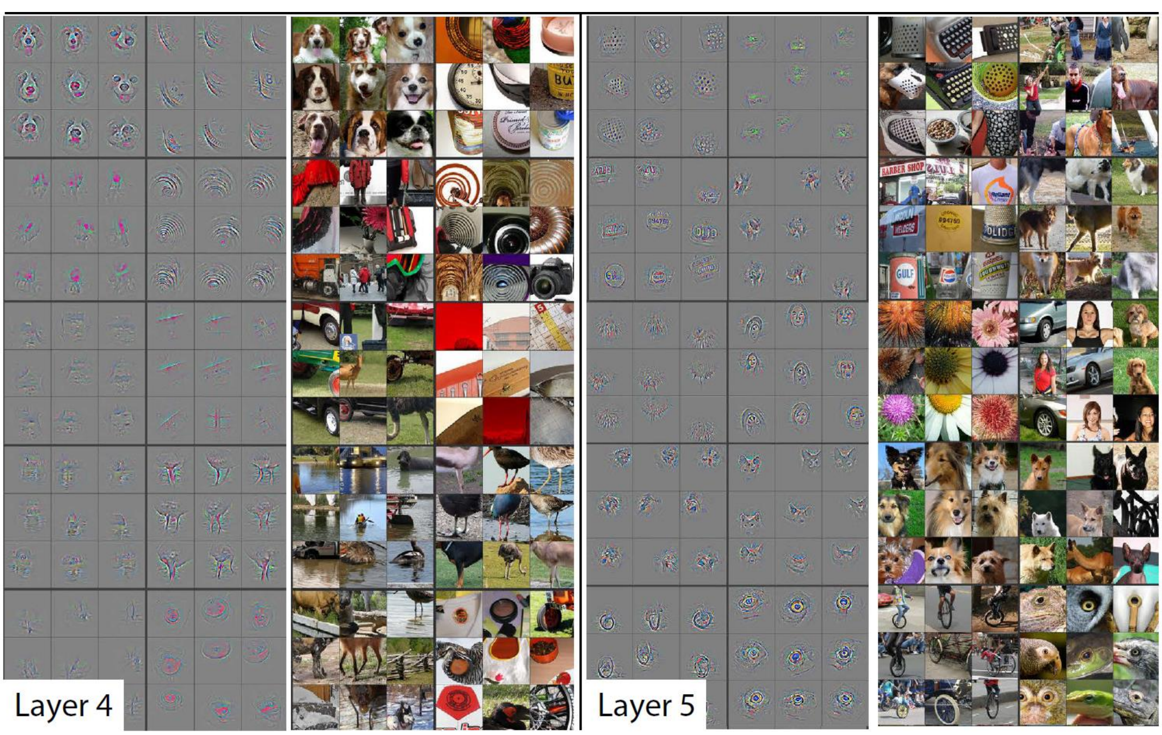


arXiv:1311.2901 (fig. 2)

Example



arXiv:1311.2901 (fig. 2)



arXiv:1311.2901

Basis functions

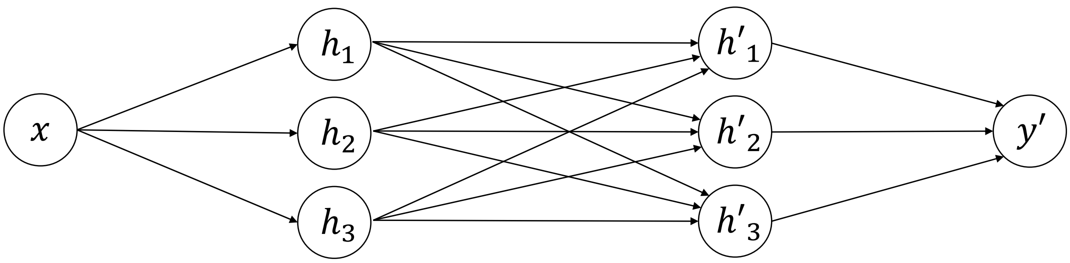
- Hand-crafted
  - Defined through expert (domain) knowledge + trial-and-error

- Data-driven
  - Learned from training data
    - *not learning only from data* → use of biases
  - Domain knowledge used to introduce biases (constraints) when
    - designing the **network architecture**
    - incorporating **invariances** (e.g. translational or scale invariance)
    - defining the **training** process (e.g. regularization, transfer learning, augmentation)

Concepts:
Inductive bias, explicit assumptions, geometric deep learning

Network diagram

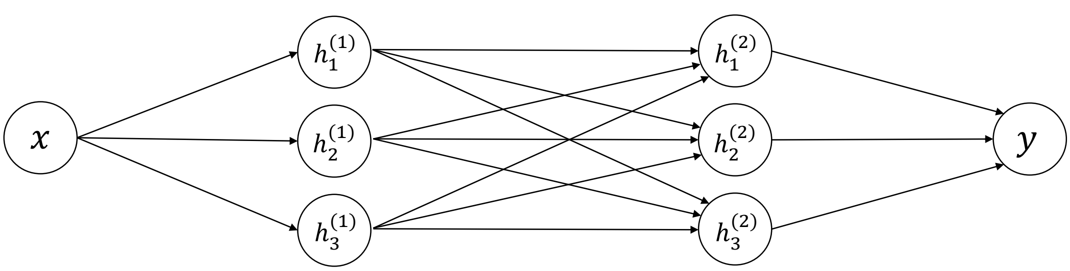
Network diagram & matrix notation



$$y' = \Theta'_0 + \boldsymbol{\Theta}' a\underbrace{\left[\boldsymbol{\Psi}_0 + \boldsymbol{\Psi}\underbrace{a[\boldsymbol{\Theta}_0 + \boldsymbol{\Theta}x]}_{\boldsymbol{h}}\right]}_{\boldsymbol{h}'}$$

Concepts: Recursive construction, hierarchical model with multiple layers, basis functions with learnable parameters, pre-activation, biases

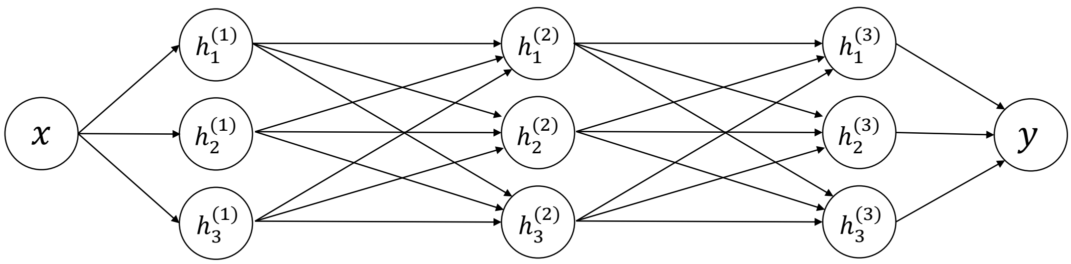
Network diagram & matrix notation



$$y = \Theta_0^{(2)} + \boldsymbol{\Theta}^{(2)} a\underbrace{\left[\boldsymbol{\Theta}_0^{(1)} + \boldsymbol{\Theta}^{(1)}\underbrace{a\left[\boldsymbol{\Theta}_0^{(0)} + \boldsymbol{\Theta}^{(0)}x\right]}_{\boldsymbol{h}^{(1)}}\right]}_{\boldsymbol{h}^{(2)}}$$

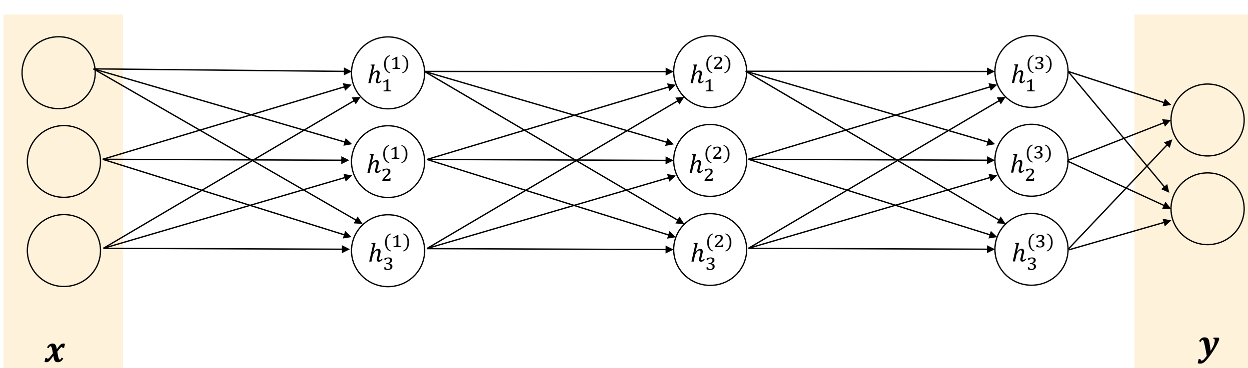
Concepts:
Linear basis, non-linear activation

Network diagram & matrix notation



$$y = \Theta_0^{(3)} + \boldsymbol{\Theta}^{(3)} a\left[\boldsymbol{\Theta}_0^{(2)} + \boldsymbol{\Theta}^{(2)} a\left[\boldsymbol{\Theta}_0^{(1)} + \boldsymbol{\Theta}^{(1)} a\left[\boldsymbol{\Theta}_0^{(0)} + \boldsymbol{\Theta}^{(0)} x\right]\right]\right]$$

Network diagram



Concepts:
Feed-forward topology, fully connected network, bias parameter (omitted in the network diagram)

Activation function

Rectified linear unit (ReLU)

$$y = f(x; \Theta)$$
$$= \Theta_0 + \Theta_1 a[\underbrace{\Theta_{10} + \Theta_{11}x}_{\text{linear function}}] + \Theta_2 a[\underbrace{\Theta_{20} + \Theta_{21}x}_{\text{linear function}}] + \Theta_3 a[\underbrace{\Theta_{30} + \Theta_{31}x}_{\text{linear function}}]$$

$$a[z] = \begin{cases} 0 & z < 0 \\ z & z \geq 0 \end{cases}$$

ReLU ← **activation function**

non-linear function

$$= \max(0, z)$$

enables efficient training
less sensitive to random initialization of the weights
computationally cheap to evaluate

Concepts:
Pre-activation, linear vs non-linear activation function, derivative of the ReLU not defined in 0

Other activation functions

$a[z] = \ln(1 + e^{z})$ — **softplus** (soft ReLU)

if $z \gg 1$ then $a[z] \simeq z$

gradient is non-zero for large, positive pre-activation values

$a[z] = \max(0, z) + \lambda \min(0, z)$ — **leaky ReLU**

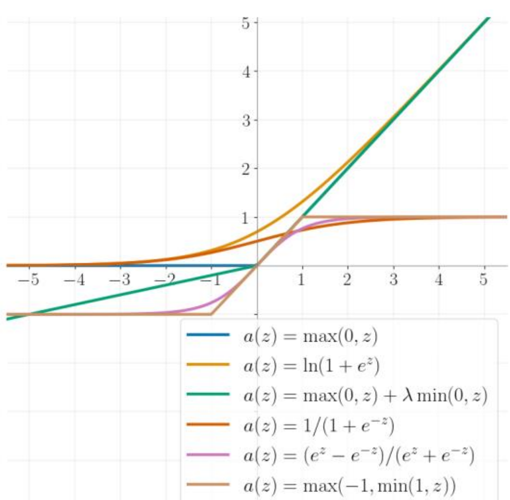
$a[z] = \frac{1}{1+e^{-z}}$ — **logistic sigmoid**

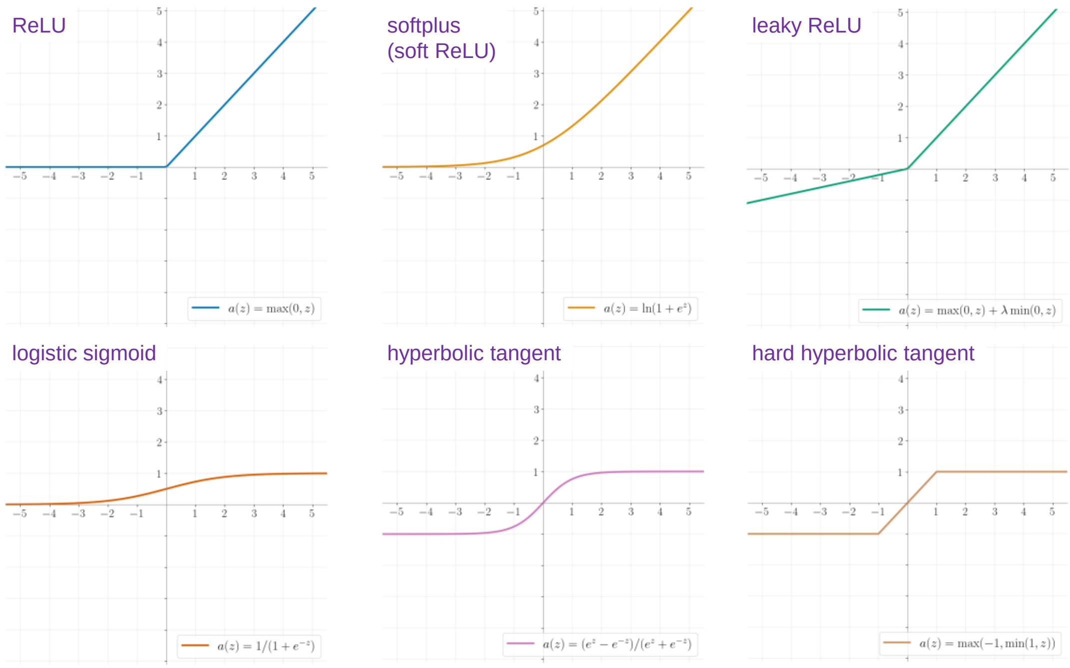
$a[z] = \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}$ — **hyperbolic tangent**

$a[z] = \max(-1, \min(1, z))$ — **hard hyperbolic tangent**

Concepts:
Training error signal, vanishing gradient, initialization scheme for weights and biases

Activation functions





Your feedback, please

What do you think of today's lecture?

ⓘ Start presenting to display the poll results on this slide.

Loss

$L(\Theta)$: training signal

$L(\Theta)$
describes
the mismatch

$f(x_i;\Theta) \longleftrightarrow y_i$

model prediction — ground truth

Recall: $$L(\Theta) = \sum_{i=1}^{N} \left(f(x_i;\Theta) - y_i\right)^2$$ Least square loss (for univariate regression)

Concepts:
Labels, annotation

Model parameters

hyperparameters

parameters

$y = f(x;\Theta)$ — (family of) family of possible relationships between x and y defined by the model parameters Θ

$\{x_i, y_i\}_{i=1}^{N}$ — training dataset of N input-output pairs

$\Theta^* = \arg\min_{\Theta} L(\Theta)$ — minimization of the loss to determine the model parameters Θ

Prediction of y from x

⬇

Computing the parameters Θ of $\mathrm{P}(y|\Theta)$ over the output space

Concept:
Conditional probability distribution

How do we build a loss function?

$y = f(x; \Theta)$ predicts y from x

compute $P(y|x)$ over the output space

encourage $P(y_i|x_i)$ to represent y_i with high probability

select a parametric distribution $P(y|\varphi)$ defined on the output space

use the network $y = f(x; \Theta)$ to determine the parameter(s) φ of the distribution

How do we build a loss function?

$\varphi_i = f(x_i; \Theta)$ parameter of the distribution corresponding to training input x_i

each training output y_i has to have a high probability under $P(y_i|\varphi_i)$

Determining the parameters

$$\Theta^* = \arg\max_{\Theta} \prod_{i=1}^{N} P(y_i|x_i) = \arg\max_{\Theta} \prod_{i=1}^{N} P(y_i|\varphi_i)$$

$$= \arg\max_{\Theta} \prod_{i=1}^{N} P(y_i|f(x_i;\Theta))$$

maximum likelihood criterion

$$= \arg\max_{\Theta} \log\left[\prod_{i=1}^{N} P(y_i|f(x_i;\Theta))\right]$$

$$= \arg\max_{\Theta} \sum_{i=1}^{N} \log[P(y_i|f(x_i;\Theta))]$$

log-likelihood criterion

Determining the parameters

$$\Theta^* = \arg\max_{\Theta} \sum_{i=1}^{N} \log[P(y_i|f(x_i;\Theta))]$$

$$= \arg\min_{\Theta} \left[-\sum_{i=1}^{N} \log[P(y_i|f(x_i;\Theta))]\right]$$

negative log-likelihood criterion

$$= \arg\min_{\Theta} L(\Theta)$$

Univariate regression

$y \in \mathbb{R}$ $\quad y = f(x;\Theta)$ $\quad \Theta^* = \arg\max_{\Theta} \prod_{i=1}^{N} P(y_i | f(x_i;\Theta))$

$$P(y|\varphi) = P(y|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-\mu)^2}{2\sigma^2}}$$

univariate normal distribution

$$P(y|f(x;\Theta),\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-f(x;\Theta))^2}{2\sigma^2}}$$

f to compute μ

$$L(\Theta) = -\sum_{i=1}^{N} \log[P(y_i|f(x_i;\Theta),\sigma^2)] = -\sum_{i=1}^{N} \log\left[\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_i-f(x_i;\Theta))^2}{2\sigma^2}}\right]$$

Loss minimization

$$\Theta^* = \arg\min_{\Theta}\left[-\sum_{i=1}^{N} \log[P(y_i|f(x_i;\Theta))]\right]$$

$$= \arg\min_{\Theta}\left[-\sum_{i=1}^{N} \log\left[\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_i-f(x_i;\Theta))^2}{2\sigma^2}}\right]\right]$$

$$= \arg\min_{\Theta}\left[\boxed{-}\sum_{i=1}^{N} \cancel{\log\left[\frac{1}{\sqrt{2\pi\sigma^2}}\right]} \boxed{-} \frac{(y_i-f(x_i;\Theta))^2}{2\cancel{\sigma}^2}\right]$$

$$= \arg\min_{\Theta}\left[\sum_{i=1}^{N} (y_i-f(x_i;\Theta))^2\right]$$

Least square loss

Inference

$$y \in \mathbb{R} \qquad y = f(x;\Theta)$$

Point estimate from the distribution $P(y_i|f(x_i;\Theta))$

$$\hat{y} = \arg\max_y P(y|f(x;\Theta^*))$$

maximum is determined by μ of the normal distribution

$$\hat{y} = f(x;\Theta^*)$$

What did we learn today?

- Function approximation
- Values and principles
- Shallow vs deep learning
- Building a deep network
- Network diagram
- Activation function
- Loss function
- Exercises will take place in PO 01

EE-559

Deep Learning

andrea.cavallaro@epfl.ch