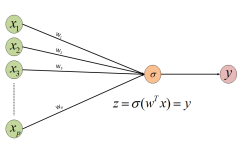
NEURAL NETWORKS – METHOD SUMMARY

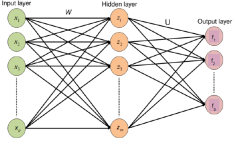
REGRESSION | CLASSIFICATION | (NON)PARAMETRIC | BLACK-BOX

General idea

- Learn **composite function** through series of nonlinear feature transformations, represented as **neurons**, organized hierarchically in **layers**
  - Basic neuron operation: 1) affine **transformation** ϕ (weighted sum of inputs), 2) nonlinear **activation** σ
  - Combinations of simple building blocks to create a complex model
- Optimize via **mini-batch stochastic gradient descent (SGD)** variants:
  - Gradient of each weight can be infered from the **computational graph** of the network $\rightarrow$ **Automatic Differentiation** (AutoDiff)
  - Algorithm to compute weight updates based on the loss is called **Backpropagation**

Hypothesis space $\mathcal{H} = \left\{ f(\mathbf{x}) : f(\mathbf{x}) = \tau \circ \phi \circ \sigma^{(h)} \circ \phi^{(h)} \circ \sigma^{(h-1)} \circ \phi^{(h-1)} \circ \ldots \circ \sigma^{(1)} \circ \phi^{(1)}(\mathbf{x}) \right\}$





NEURAL NETWORKS – METHOD SUMMARY

Architecture

- Input layer: original features **x**
- Hidden layers: nonlinear transformation of previous layer $\phi^{(h)} = \sigma^{(h-1)}(\phi^{(h-1)})$
- Output layer: number of output neurons and activation depends on problem $\tau(\phi)$
  - Regression: one output neuron, $\tau =$ identity
  - Binary classification: one output neuron, $\tau = \frac{1}{1+\exp(-\boldsymbol{\theta}^\top \mathbf{x})}$ (logistic sigmoid)
  - Multiclass Classification: g output neurons, $\tau_j = \frac{\exp(f_j)}{\sum_{j=1}^{g} \exp(f_j)}$ (softmax)

Empirical risk In general, compatible with any differentiable loss

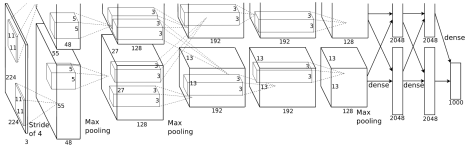
Optimization

- Variety of different optimizers, mostly based on some form of **stochastic gradient descent (SGD)**
- Improvements:
  - **(1)** Accumulation of previous gradients $\rightarrow$ **Momentum**
  - **(2)** Weight specific scaling based on previous squared gradients $\rightarrow$ **RMSProb**
    $\Rightarrow$ **ADAM** combines (1) and (2)
  - **(3)** Learning rate schedules, e.g., decaying or cyclical learning rates
- Training progress is measured in full passes over the full training data, called **epochs**
- **Batch size** is a hyperparameter and limited by input data dimension

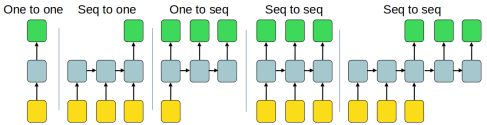
NEURAL NETWORKS – METHOD SUMMARY

Network types Large variety of architectures for different data modelities

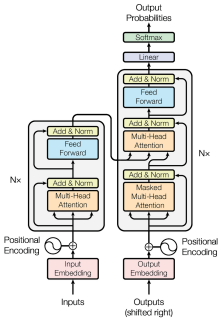
- **Feedforward NNs / multi-layer perceptrons (MLPs):** sequence of **fully-connected** layers ⇒ tabular data
- **Convolutional NNs (CNNs):** sequence of feature map extractors with spatial awareness ⇒ images, time series
- **Recurrent NNs (RNNs):** handling of sequential, variable-length information ⇒ times series, text, audio
- **Transformers:** Learning invariances from data, handling multiple/any data modalities



Convolutional network architecture



Recurrent network architecture



Transformer network architecture

NEURAL NETWORKS – METHOD SUMMARY

Hyperparameters

- **Architecture**:
  - Lots of design choices ⇒ tuning problem of its own.
  - Typically: hierachical optimization of components (cells) and macro structure of network
    → **Neural Architecture Search (NAS)**
  - Many predifined (well working) architectures exist for standard tasks
- **Training**:
  - Initial learning rate and various regularization parameters
  - Number of epochs is determined by **early-stopping**
  - **Data-augmentation**, e.g., applying random rotations to input images

Disadvantages

- Typically, high computational **cost**
- High demand for **training data**
- Strong tendency to **overfit**
- Requiring lots of **tuning expertise**
- **Black-box** model – hard to interpret

Advantages

+ Applicable to **complex, nonlinear** problems
+ Very **versatile** w.r.t. architectures
+ Strong **performance** if done right
+ Built-in **feature extraction**, obtained by intermediate representations
+ Easy handling of **high-dimensional** data
+ **Parallelizable** training

Foundation models

- **Enormous** models trained on vast amounts of (general) data, e.g., all of wikipedia, in **self-supervised** fashion
- Used as starting point (**pre-trained**) and fine-tuned via **transfer** or **few-shot** learning for other tasks

General hints

- Instead of NAS, use a standard architecture and tune training hyperparameters
- Training pipeline (data-augmentation, training schedules, ...) is more crucial than the specific architecture
- While NNets are state-of-the-art for **computer vision (CV)** and **natural language processing (NLP)**, we recommend not to use them for tabular data because alternatives perform better