Enter Neural Networks

SICSS Paris '25

Julien Boelaert

CERAPS, Université de Lille

24/06/2025

Outline

1. Introduction: supervized learning
2. Multilayer perceptrons
3. Model quality, overlearning
4. Advanced neural networks: deep, convolutional, recurrent

1/ Introduction: supervized learning

Supervised learning = **predictive algorithms**

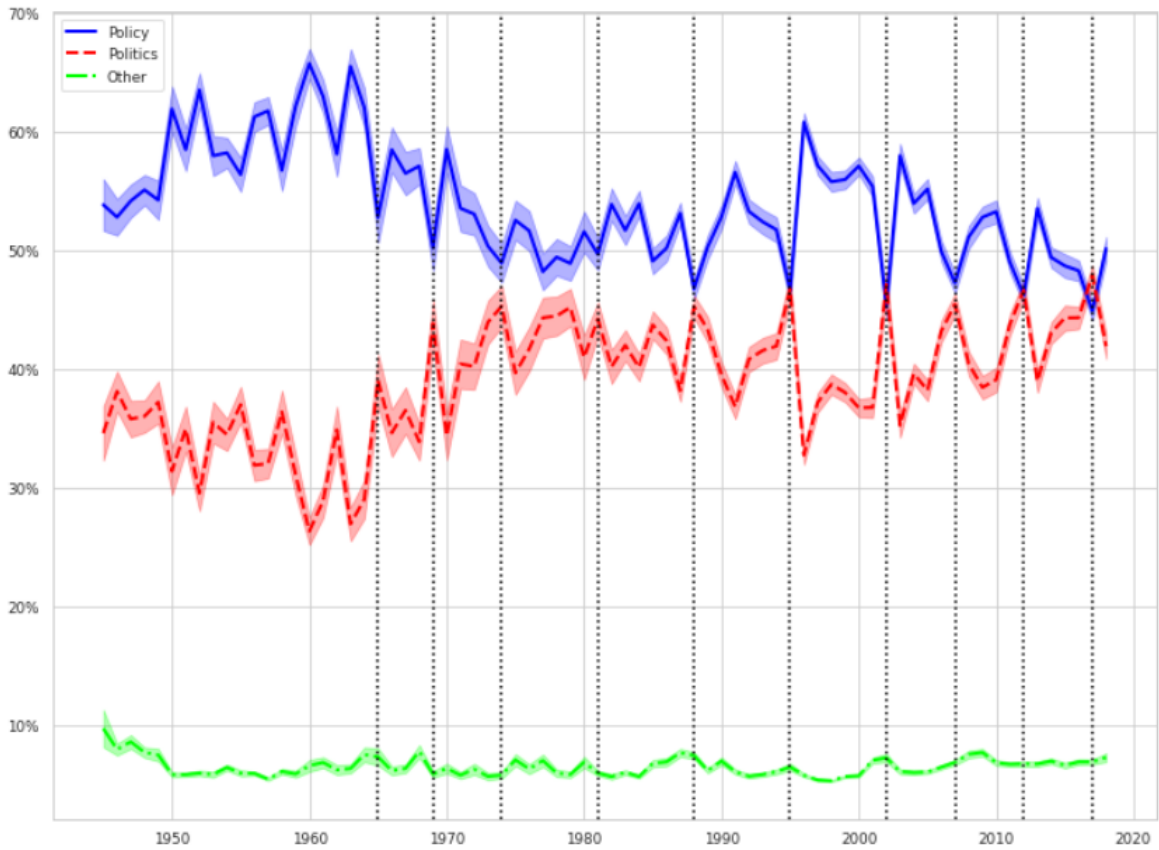
- based on a set of predictors (X), predict an outcome (Y)
- "supervised": we give the model examples of X and Y, and it must "learn" the relationship between them
  (vs. "unsupervised": explore the inner structure of X, eg. GDA, clustering)
- hypothesis: there exists a function f so that $Y = f(X)$

1/ Introduction: supervized learning

Applications of supervised learning in NLP

- Classic example: spam detection
- NLP pre-treatment: part-of-speech, named entities recognition
- sentiment analysis / stance detection: typically on social media posts, eg. detect sentiment about political candidates
- textual data augmentation: make huge corpora exploitable, by extrapolating to the whole corpus a judgment made on a small sample

1/ Introduction: supervized learning



Evolution of Politics and Policy in *Le Monde* political articles, since 1945

In 2000, out of all the articles written by political reporters in *Le Monde*, 57% evoked policy measures, 37% dealt with politics, 6% with other aspects.

Do, S., Ollion, É., & Shen, R. (2024). The Augmented Social Scientist: Using Sequential Transfer Learning to Annotate Millions of Texts with Human-Level Accuracy. *Sociological Methods & Research*, 53(3), 1167-1200. https://doi.org/10.1177/00491241221134526

1/ Introduction: supervized learning

Supervised learning in NLP: how?

- **Models:** simple (linear), intermediate (standard machine learning: SVM, Random Forests...), advanced (large language models: BERT, GPT)
- **Features:** (ie predictors)
  - hand-crafted features: text length, presence of some terms (dictionary methods), presence of all-caps words ...
  - word counts: document-term matrix, tf-idf
  - word embeddings, sentence embeddings
  - raw text, processed by transformer models

2/ Multilayer perceptrons

2/ Multilayer perceptrons

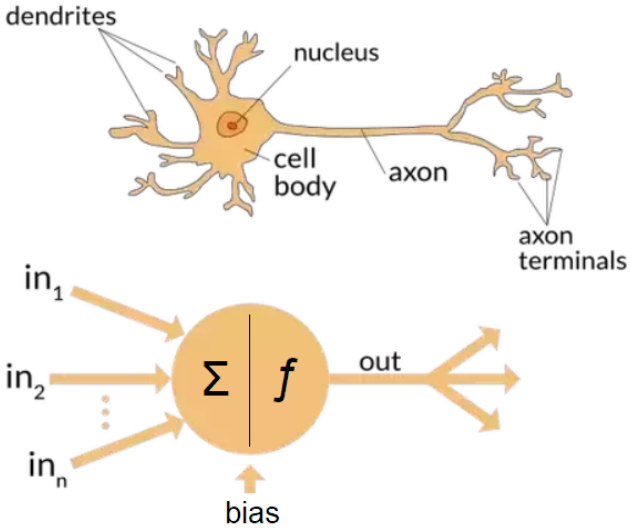
Short history of neural networks

- 1950s: the Perceptron (Rosenblatt 1957, refuted by Minsky and Papert 1969)
- 1970s: the first "AI winter"
- 1980s: Multilayer perceptron (Rumelhart, Hinton), Convolutional neural nets (Fukushima, LeCun)
- 1990s-2000s: replaced by more efficient ML algorithms (SVM, Random forests, Gradient boosting machines, Lasso)
- 2010s: Deep neural networks
- 2020s: Transformers

2/ Multilayer perceptrons

2/ Multilayer perceptrons

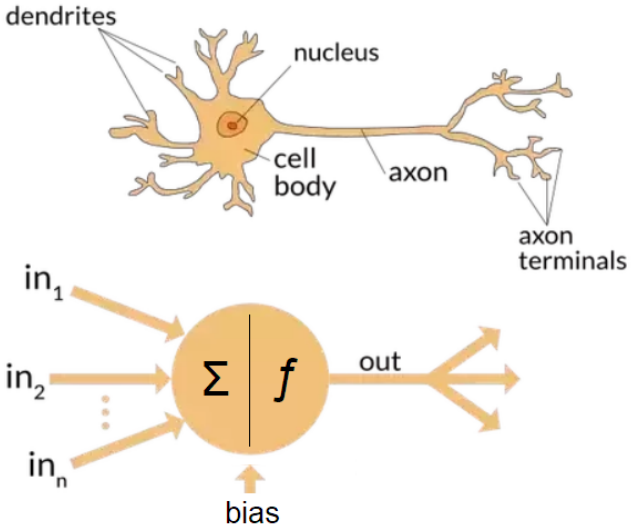
The basic building block of all NNs is the **artificial neuron**



- loose biological inspiration
- transforms inputs to outputs

2/ Multilayer perceptrons

The basic building block of all NNs is the **artificial neuron**

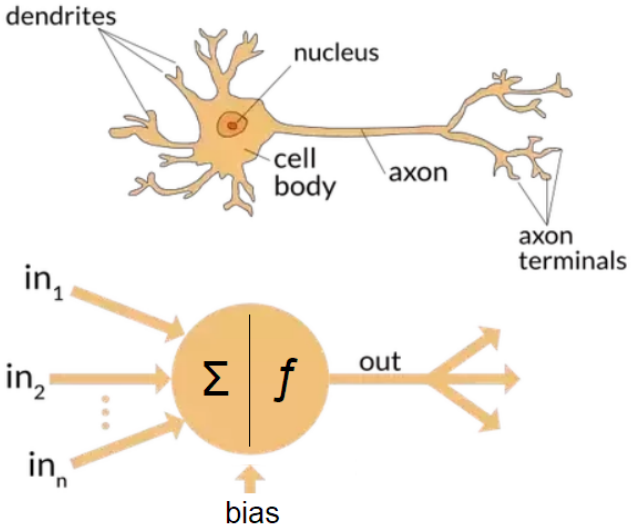


- loose biological inspiration
- transforms inputs to outputs, by
  1. **weighted sum** of the inputs
  2. and **activation function** f

$$\mathbf{out} = f\left(\alpha_0 + \sum_i \alpha_i \mathbf{in_i}\right)$$

2/ Multilayer perceptrons

The basic building block of all NNs is the **artificial neuron**



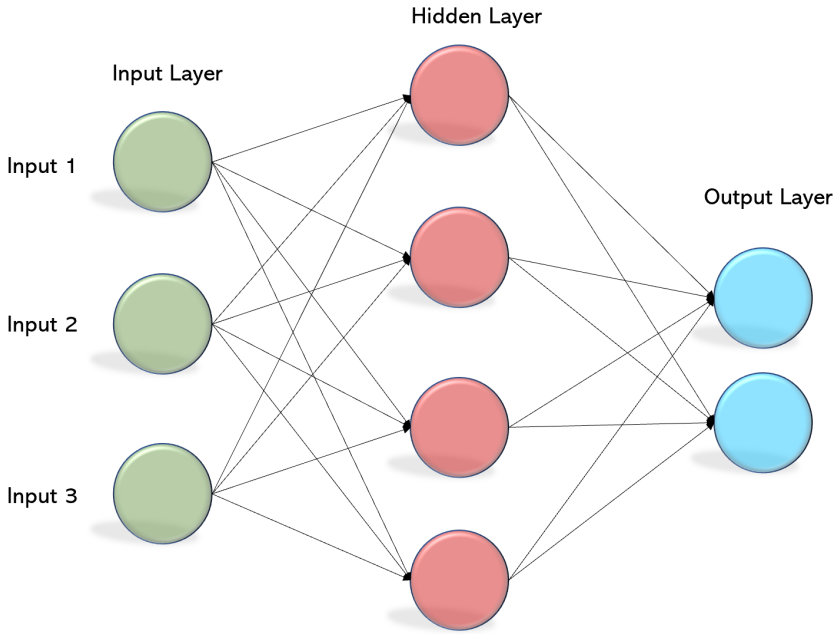
- loose biological inspiration
- transforms inputs to outputs, by
  1. **weighted sum** of the inputs
  2. and **activation function** f

  $\mathbf{out} = f\left(\alpha_0 + \sum_i \alpha_i \mathbf{in_i}\right)$
- With f = identity, linear regression (or perceptron, Rosenblatt 1958)
- With $f(x) = \frac{1}{1+exp(-x)}$, logistic regression

Fitting (*training*): find the best weights α that minimize a cost function over the training data (MSE / cross-entropy).

2/ Multilayer perceptrons

A **multilayer perceptron** is a composition of such neurons (known since the 1960s, popularized 1985 by Rumelhart, Hinton and Williams with the **backpropagation** algorithm)



- each arrow (connection) has a weight
- layers: input, hidden, output
- information flows left to right: **feedforward** network

$$\mathbf{out_k} = g\left(\beta_{0k} + \sum_j \beta_{jk} f\left(\alpha_{0j} + \sum_i \alpha_{ij} \mathbf{in_i}\right)\right)$$

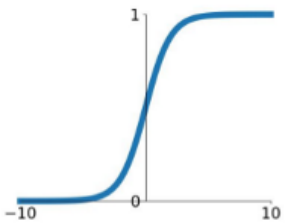
2/ Multilayer perceptrons

Activation functions for hidden layer:

- if f is linear, an MLP reduces to a linear model
- common choices of nonlinear activations:
  sigmoid or tanh (1980s), ReLU and variants (2000s)

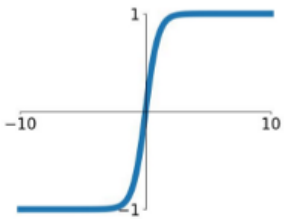
Sigmoid

$\sigma(x) = \frac{1}{1+e^{-x}}$



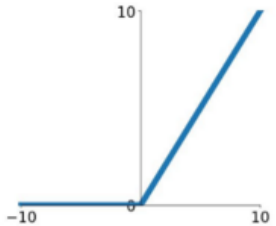
tanh

$\tanh(x)$



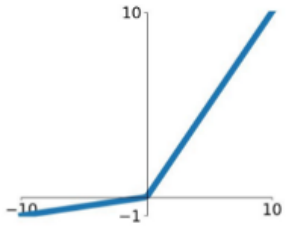
ReLU

$\max(0, x)$



Leaky ReLU

$\max(0.1x, x)$

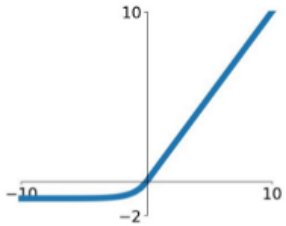


Maxout

$\max(w_1^T x + b_1, w_2^T x + b_2)$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



2/ Multilayer perceptrons

Common choices of **output layer activations**:

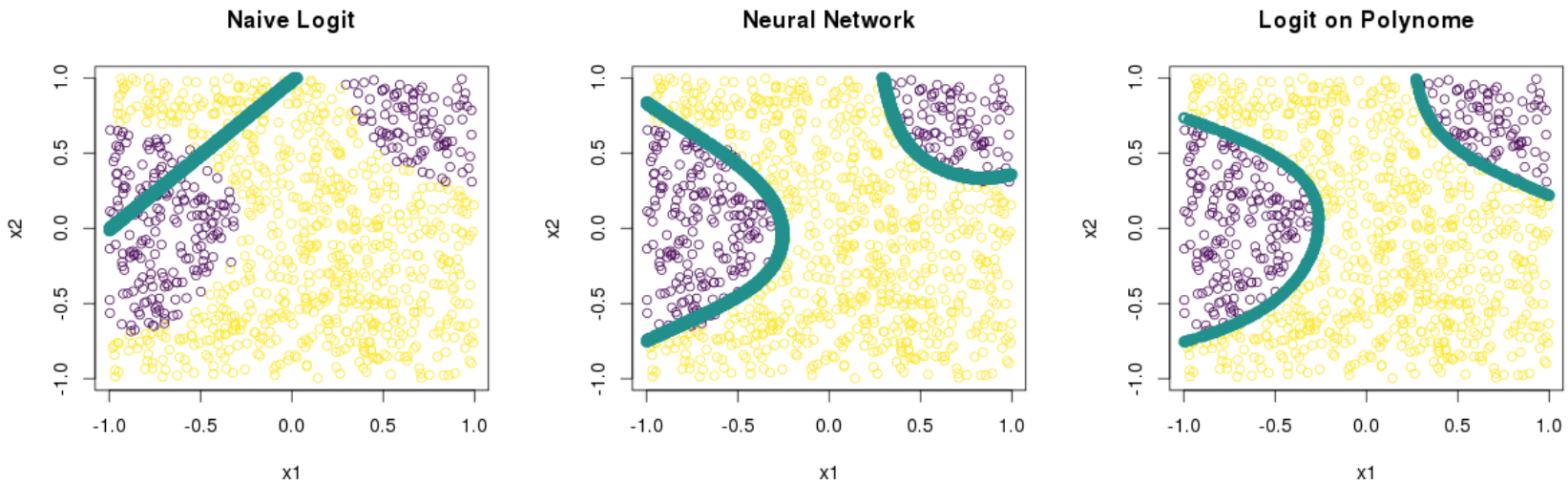
- regression: identity, or tanh (where outputs are first normalized to (-1,1))
- binary classification: sigmoid (as in logistic regression)
- multiple classification: softmax

$$softmax(out_k) = \frac{exp(out_k)}{\sum_l exp(out_l)}$$

2/ Multilayer perceptrons

Why use MLPs / ANNs?

- **universal approximation**: MLP theoretically capable of approximating *any* function, if enough neurons in hidden layer
- theoretically removes the need for hand-crafted **features** to feed the model: automatically constructed during training.

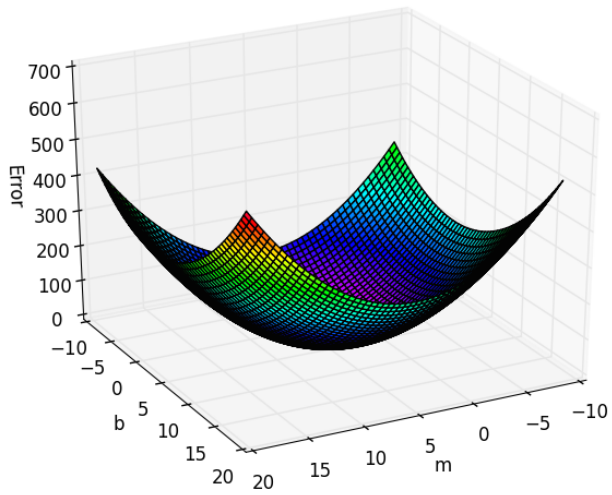


Example: binary classification on two predictors

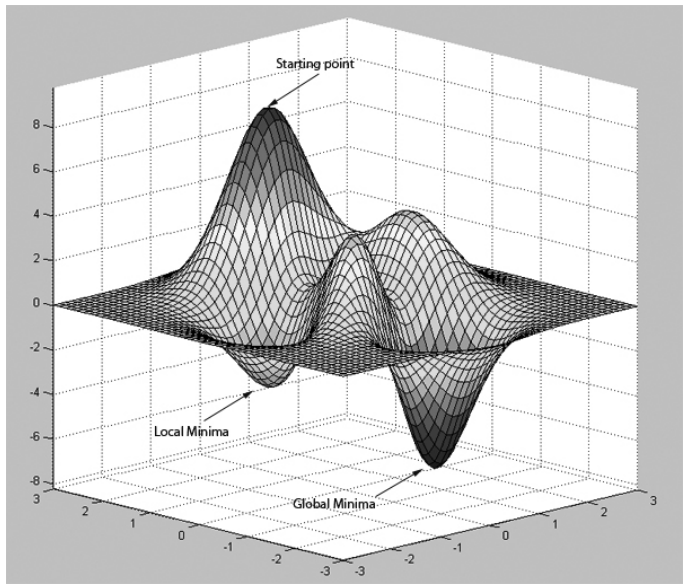
2/ Multilayer perceptrons

Training: adjusting the weights so that the predicted outputs correspond to the observed outputs

- in linear regression, exact closed-form solution
- in logistic regression, algorithmic solution through convex optimization (solution is existing and unique)
- in MLPs, the **error surface** is not convex $\rightarrow$ minimum can only be approached algorithmically, with no guarantee to find the global minimum



Convex error surface

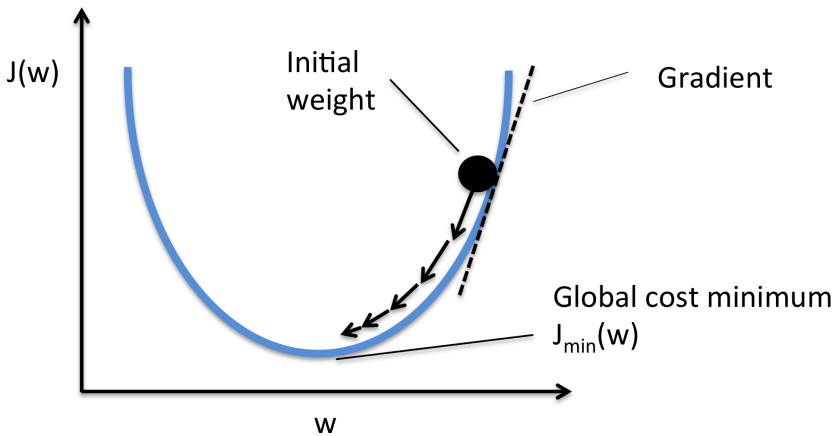


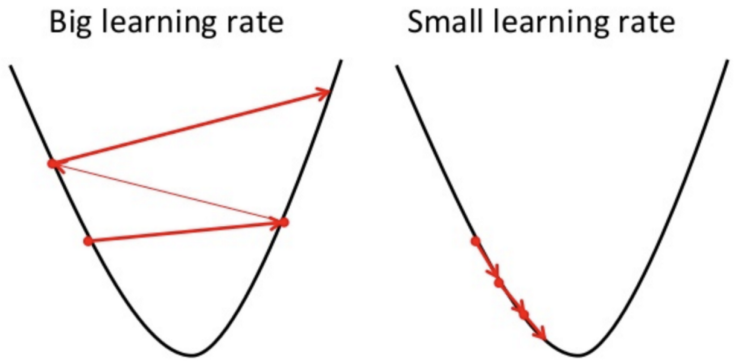
Nonconvex error surface

2/ Multilayer perceptrons

Training: **(stochastic) gradient descent**, aka **backpropagation**.

- weights are initialized randomly
- for each observation (several **epochs**, *ie* runs over the complete training set):
  1. compute the output error
  2. compute the gradient of the error wrt weights
  3. move the weights in the opposite direction of the gradient (with a small **step size** / **learning rate**)
- possible because error function differentiable wrt weights





- today, the *de facto* standard optimizer is Adam (Kingma and Ba, 2015), adaptively adjusts the step size for each individual weight.

2/ Multilayer perceptrons

MLP (and machine learning) for social science?

$\rightarrow$ marginal place compared to linear regression

- ML = focus on prediction, but hard to interpret

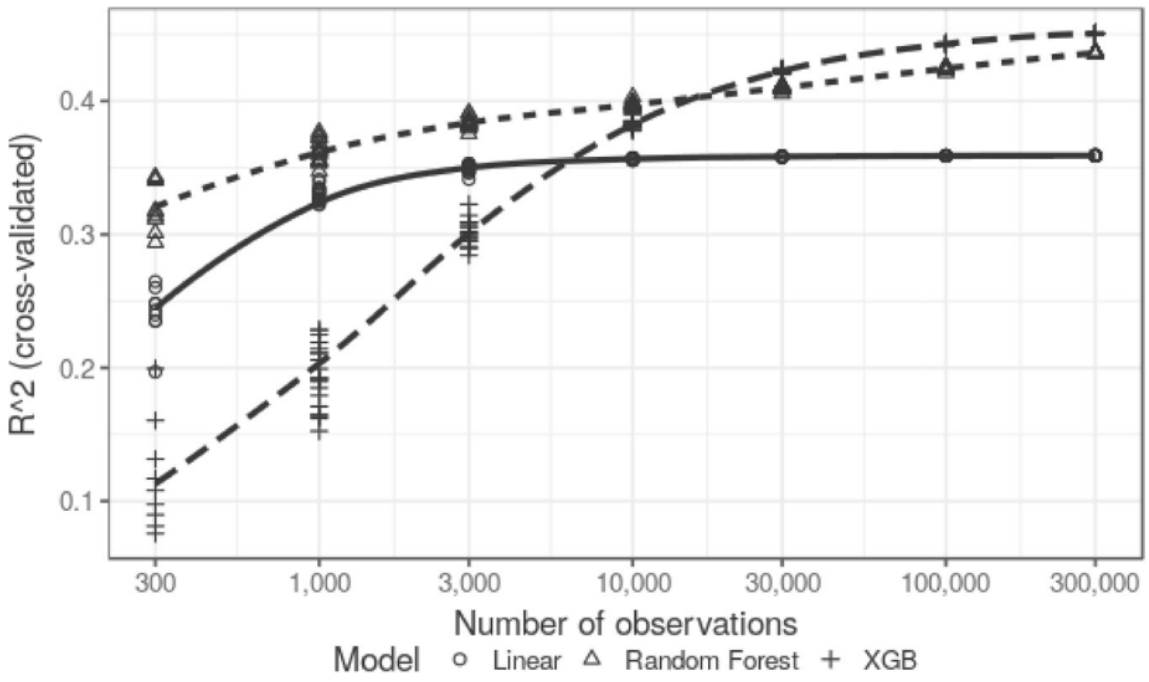
2/ Multilayer perceptrons

MLP (and machine learning) for social science?

$\rightarrow$ marginal place compared to linear regression

- ML = focus on prediction, but hard to interpret
- On well-studied questions (eg. wage equations), the prediction advantage of ML models only occurs with *a lot* of data

FIGURE 4. – ***R^2 for three different predictive models, on random subsamples of varying sizes taken from the Swedish wages dataset***



2/ Multilayer perceptrons

MLP for text analysis?

- using text as predictors in MLPs is possible but awkward: fixed text length, and each weight applies only to a certain position in the sequence
- Bengio et al. (2003) used an MLP for *language modeling* (predicting the next word, based on the k previous ones): starting point of word embeddings in the NN literature, and more generally of neural language modeling.

3/ Model quality, overlearning

3/ Model quality, overlearning

Quality of regression models: Mean squared error

With classification models, several options:

- **accuracy**: how often is the classifier correct?
- **precision**: when the classifier outputs "A", how often is it correct?
- **recall**: of all the true "A" cases, how many are detected by the classifier?
- **F1 score**: harmonic mean of precision and recall

3/ Model quality

Accuracy: how often is the classifier correct?

$$\text{Accuracy} = \frac{TP + TN}{N}$$

obs	1	2	3	4	5	6	7	8	9	10
truth	**A**	A	A	**A**	B	**B**	B	**A**	B	B
prediction	**A**	B	B	**A**	A	**B**	A	**A**	A	A

3/ Model quality

Accuracy: how often is the classifier correct?

$$\text{Accuracy} = \frac{TP + TN}{N}$$

obs	1	2	3	4	5	6	7	8	9	10
truth	**A**	A	A	**A**	B	**B**	B	**A**	B	B
prediction	**A**	B	B	**A**	A	**B**	A	**A**	A	A

$$\text{Accuracy} = 4 \ / \ 10 = 40 \ \%$$

Problem for unbalanced data: if 99% of true values are "A", then always predicting "A" gives 99% accuracy...

3/ Model quality

Precision for "A": when the classifier outputs "A", how often is it correct?

$$\text{Precision} = \frac{TP}{TP + FP}$$

obs	1	2	3	4	5	6	7	8	9	10
truth	**A**	A	A	**A**	B	**B**	B	**A**	B	B
prediction	**A**	B	B	**A**	A	**B**	A	**A**	A	A

3/ Model quality

Precision for "A": when the classifier outputs "A", how often is it correct?

$$\text{Precision} = \frac{TP}{TP + FP}$$

obs	1	2	3	4	5	6	7	8	9	10
truth	**A**	A	A	**A**	B	**B**	B	**A**	B	B
prediction	**A**	B	B	**A**	A	**B**	A	**A**	A	A

Precision for A = 3 / 7 = 42.9 %
Precision for B = 1 / 3 = 33.3 %

3/ Model quality

Recall for "A": of all the true "A" cases, how many are detected by the classifier?

$$\text{Recall} = \frac{TP}{TP + FN}$$

obs	1	2	3	4	5	6	7	8	9	10
truth	**A**	A	A	**A**	B	**B**	B	**A**	B	B
prediction	**A**	B	B	**A**	A	**B**	A	**A**	A	A

3/ Model quality

Recall for "A": of all the true "A" cases, how many are detected by the classifier?

$$\text{Recall} = \frac{TP}{TP + FN}$$

obs	1	2	3	4	5	6	7	8	9	10
truth	**A**	A	A	**A**	B	**B**	B	**A**	B	B
prediction	**A**	B	B	**A**	A	**B**	A	**A**	A	A

Recall for A = 3 / 5 = 60 %
Recall for B = 1 / 5 = 20 %

3/ Model quality

obs	1	2	3	4	5	6	7	8	9	10
truth	**A**	A	A	**A**	B	**B**	B	**A**	B	B
prediction	**A**	B	B	**A**	A	**B**	A	**A**	A	A

Confusion table

	pred A	pred B	Recall (%)
true A	3	2	60.0
true B	4	1	20.0
Precision (%)	42.9	33.3	

3/ Model quality

F1-score : harmonic mean of precision and recall (lower than arithmetic mean)

$$\begin{aligned} \text{F1} &= 2 \times \frac{\textit{precision} \times \textit{recall}}{\textit{precision} + \textit{recall}} \\ &= \frac{TP}{TP + \frac{1}{2}(FP + FN)} \end{aligned}$$

	Precision	Recall	F1
A	42.9	60	50
B	33.3	20	25

3/ Model quality

Global F1-scores: three possibilities

- Macro F1: unweighted average F1
- Weighted F1: weighted average (weights = class sizes)
- Micro F1: based on total True Positives (TP), False Negatives (FN), and False Positives (FP)

$$\text{Micro F1} = \frac{TP}{TP + \frac{1}{2}(FP + FN)}$$

3/ Overlearning

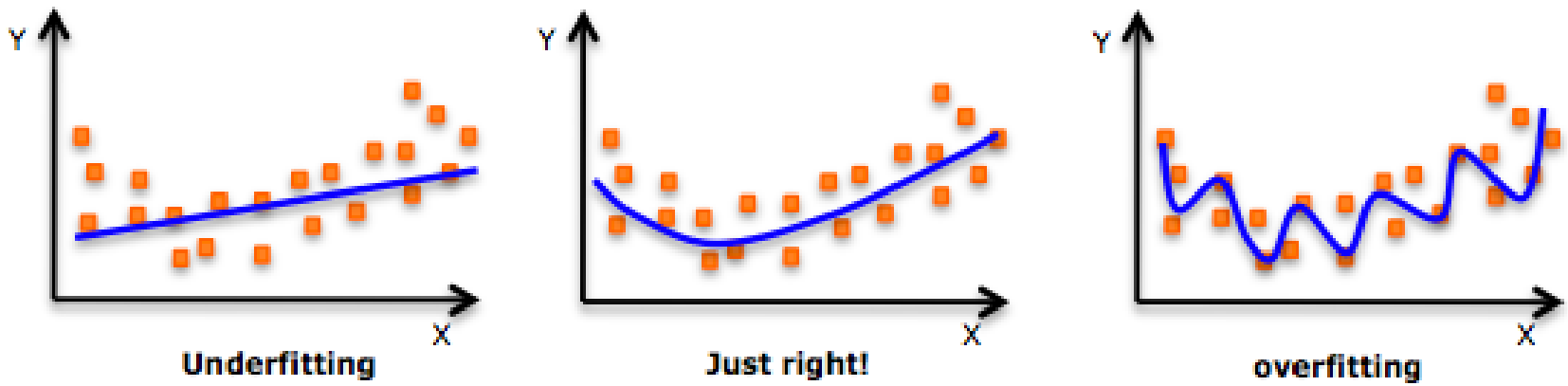
Because MLPs are so flexible, they are prone to **overfitting**

- overfitting = learning noise instead of only signal
  "getting the results right, but the reasoning wrong"
- in theory, a NN can learn any function;
  in practice, it can overlearn any dataset

3/ Overlearning

Because MLPs are so flexible, they are prone to **overfitting**

- overfitting = learning noise instead of only signal
  "getting the results right, but the reasoning wrong"
- in theory, a NN can learn any function;
  in practice, it can overlearn any dataset
- results in perfect fit on training data, but bad **generalization**
  (*ie* prediction on out-of-sample data)



3/ Overlearning

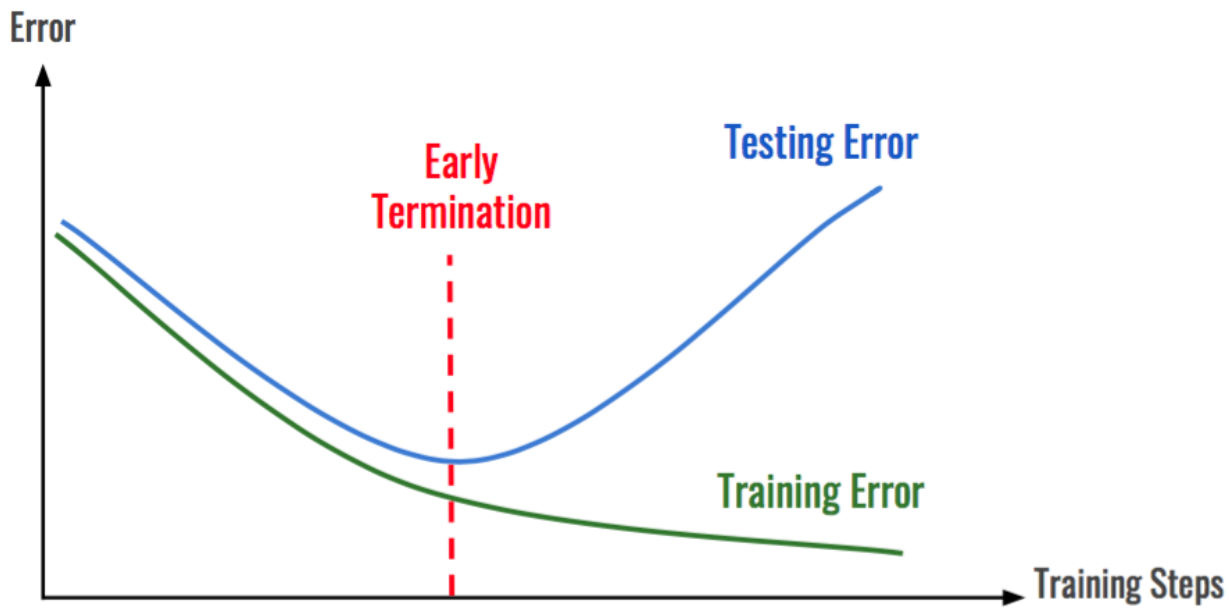
To measure and control overfitting, dataset divided into 3 subsets

- **training** set: observations used for SGD
- **development** / validation set: not used for SGD, but for monitoring during training, and deciding when to stop
- **test** set: used *only* at the end, measure **generalization error**

3/ Overlearning

To measure and control overfitting, dataset divided into 3 subsets

- **training** set: observations used for SGD
- **development** / validation set: not used for SGD, but for monitoring during training, and deciding when to stop
- **test** set: used *only* at the end, measure **generalization error**



3/ Overlearning

To prevent overfitting, use **regularization** techniques:

- add a regularization term to the error function:
  $Err^*(\mathbf{w}) = Err(\mathbf{w}) + \sum_j |w_j|$ (L1-norm)
  $Err^*(\mathbf{w}) = Err(\mathbf{w}) + \sum_j w_j^2$ (L2-norm, "**weight decay**")
  $\rightarrow$ prevents weights from becoming too big, "simpler" model

3/ Overlearning

To prevent overfitting, use **regularization** techniques:

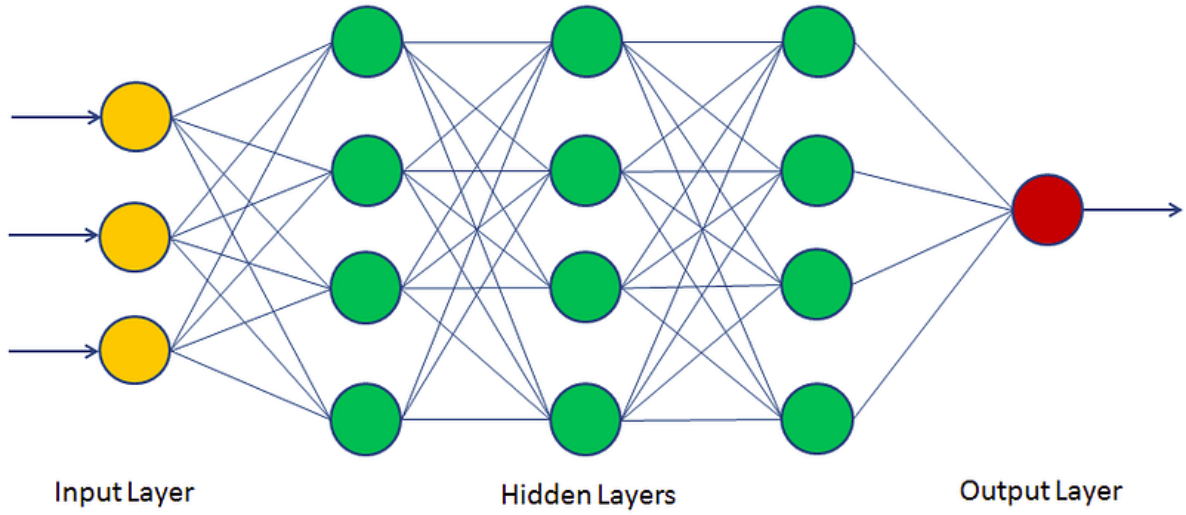
- add a regularization term to the error function:
  $Err^*(\mathbf{w}) = Err(\mathbf{w}) + \sum_j |w_j|$ (L1-norm)
  $Err^*(\mathbf{w}) = Err(\mathbf{w}) + \sum_j w_j^2$ (L2-norm, "**weight decay**")
  $\rightarrow$ prevents weights from becoming too big, "simpler" model
- **dropout**: randomly drop units (along with their connections) from the neural network during training, preventing units from co-adapting too much.
  (N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R., 2014, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting", *JMLR*, 15(56))
- many more: mini batch, gradient clipping, normalization layers such as BatchNorm (2015) or LayerNorm (2016), ...

In any case, though, neural nets are notoriously **difficult to train**!

4/ Advanced neural networks

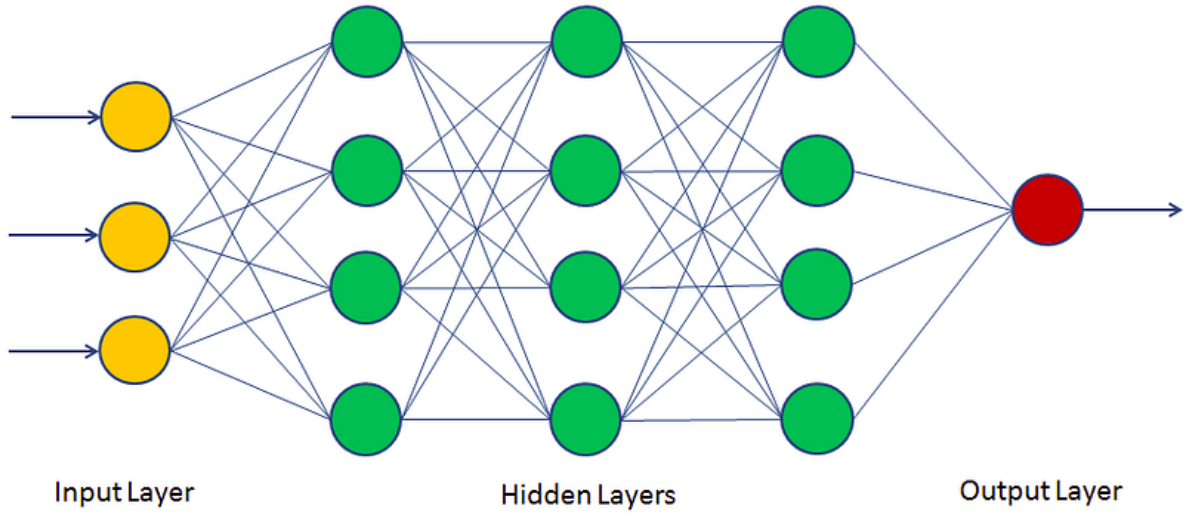
4/ Advanced neural networks

Deep learning started gaining traction in the late 2000s:



4/ Advanced neural networks

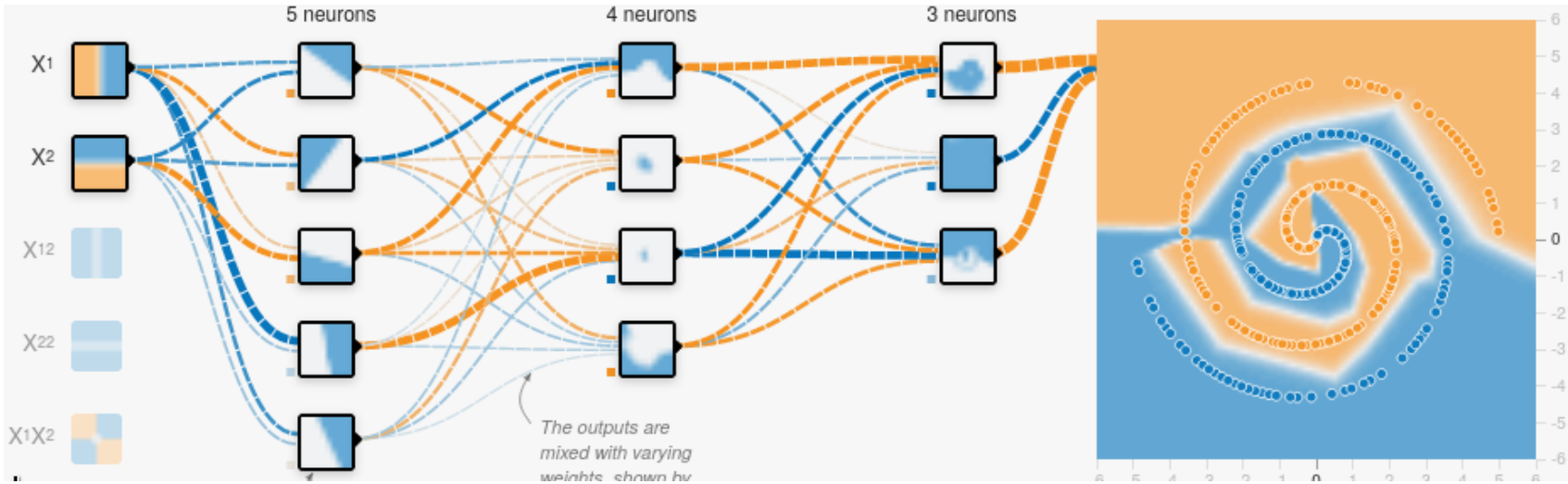
Deep learning started gaining traction in the late 2000s:



- more hidden layers
- bigger training datasets
- a few tricks: ReLU activation, better initialization schemes (*eg.* Xavier Glorot), dropout regularization
- faster computation with GPUs (esp. convnets)

4/ Advanced neural networks

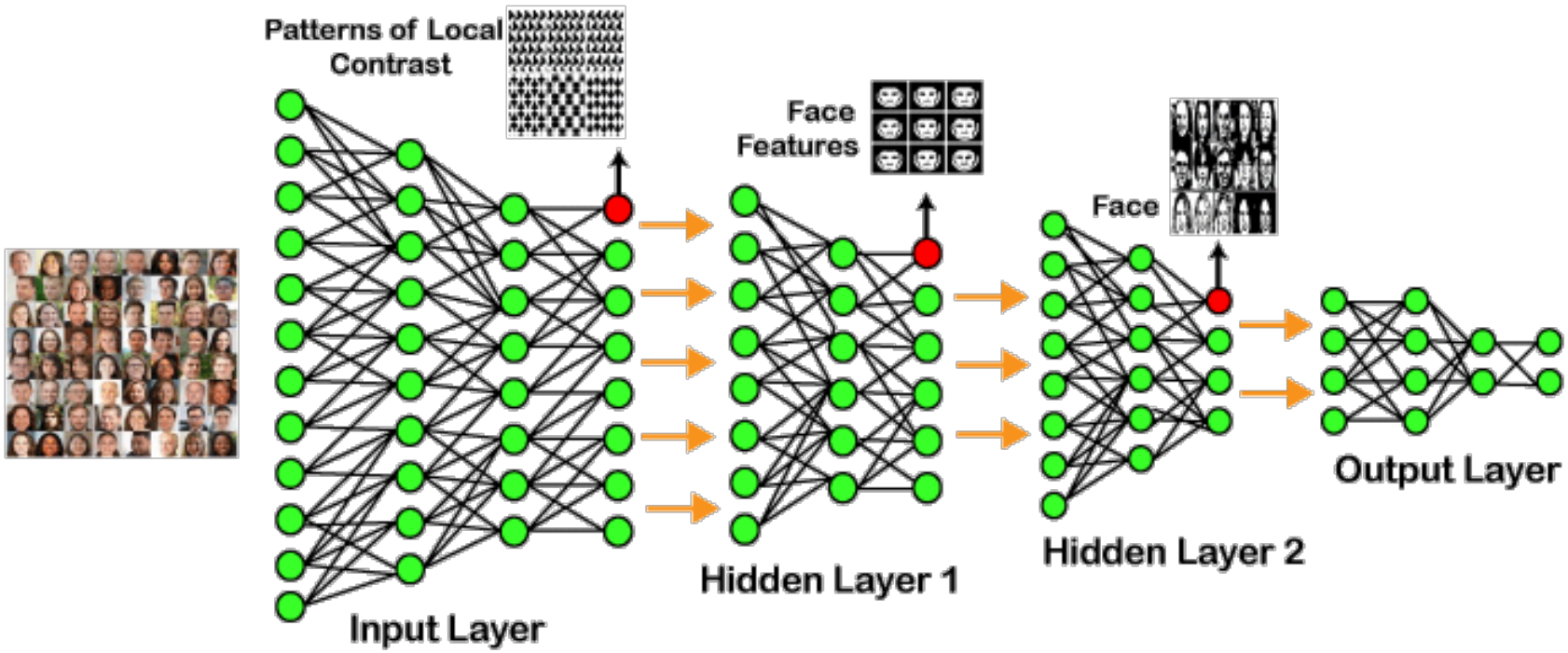
Main intuition: successive layers build **levels of abstraction**, by recombining the features of the previous layers



Try for yourself at `http://playground.tensorflow.org`

4/ Advanced neural networks

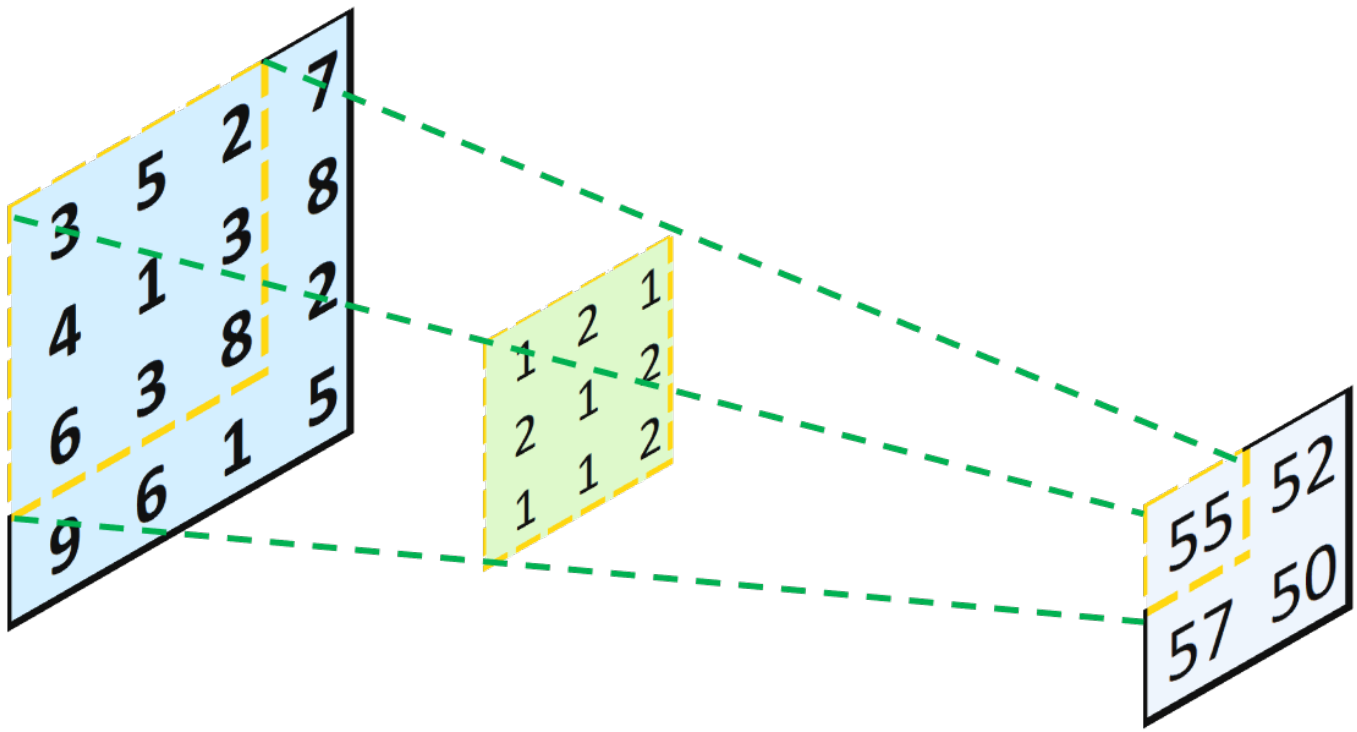
Example: facial features to faces



4/ Advanced neural networks

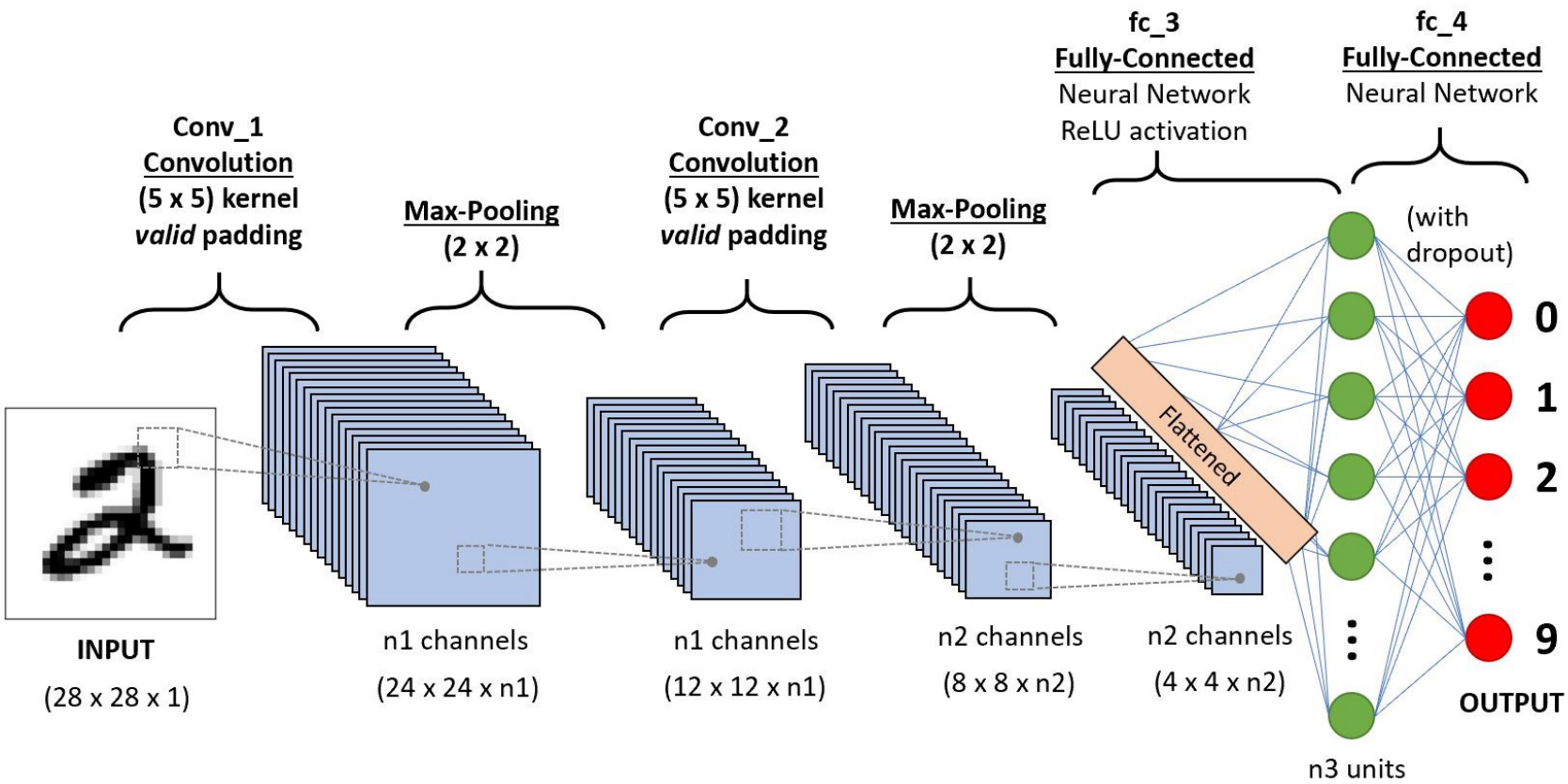
Convolutional neural networks (aka convnets, CNN) are a special architecture, well suited for structured data such as images or text. Invented in 1980s ("Neocognitron" by Fukushima 1980, CNNs by LeCun late 1980s), became huge in the 2000s.

Main building block: a convolution "filter" that slides over the input.



4/ Advanced neural networks

Deep convnets typically alternate convolution and pooling layers, until a final few MLP layers



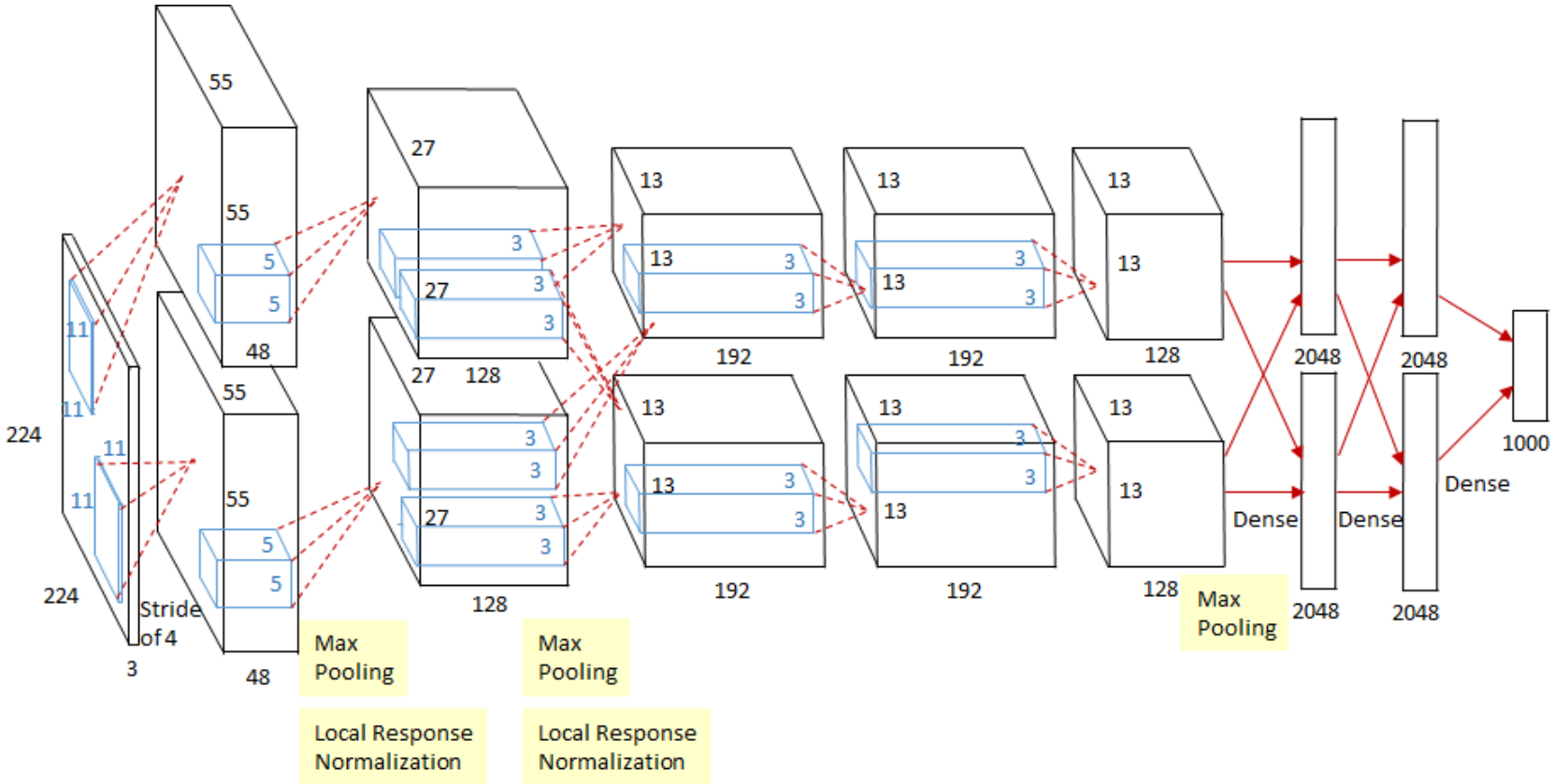
4/ Advanced neural networks

Perks of convolutional networks:

- make explicit use of the structure (images 2D, text 1D)
- every filter is applied to every region of the input data
- deep convnets: higher layers have access to large regions
- efficient computation with GPUs
- trained end-to-end with backpropagation, no need for hand-crafted features.

4/ Advanced neural networks

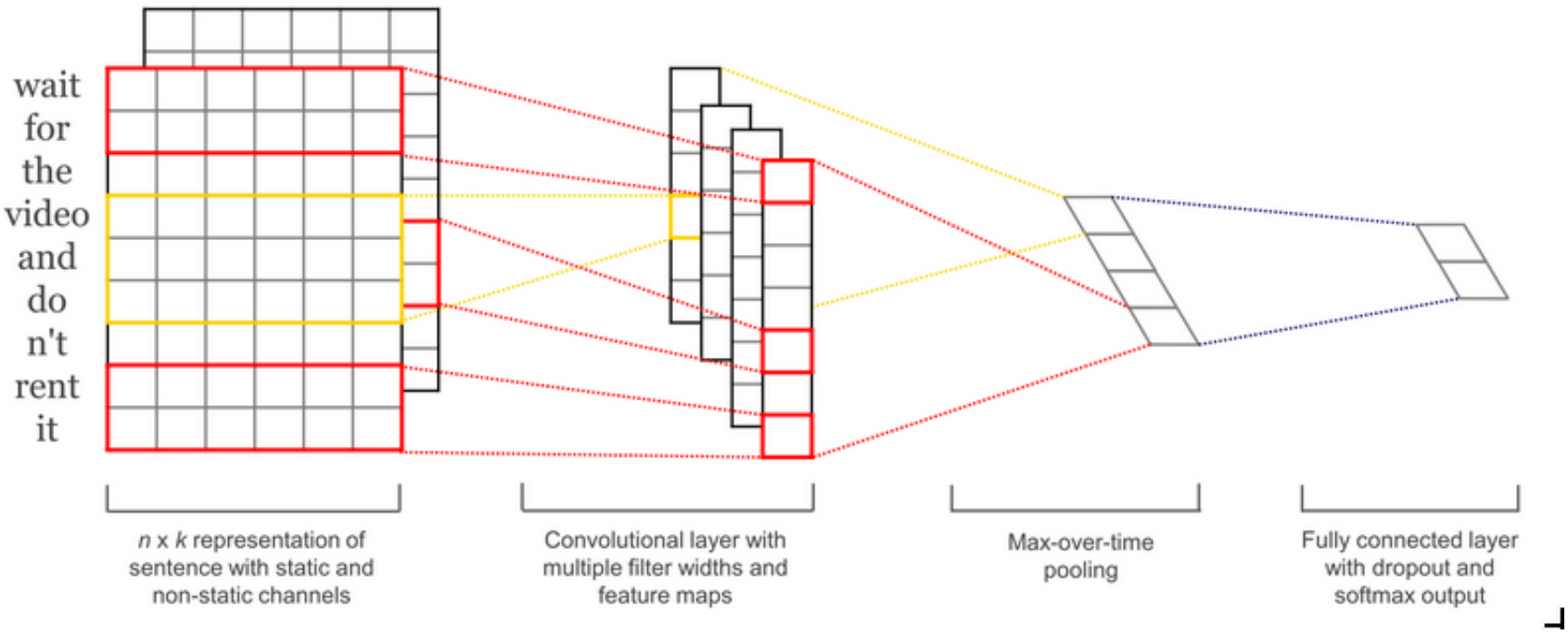
Example convnet: Alexnet (2012) achieved state-of-the-art image recognition, and started the deep learning hype.



A. Krizhevsky, I. Sutskever, G. Hinton, 2017, "ImageNet classification with deep convolutional neural networks", *Communications of the ACM*, 60 (6)

4/ Advanced neural networks

Convnets for text analysis: 1D convolutions and pooling, possibly on pre-trained word embeddings.

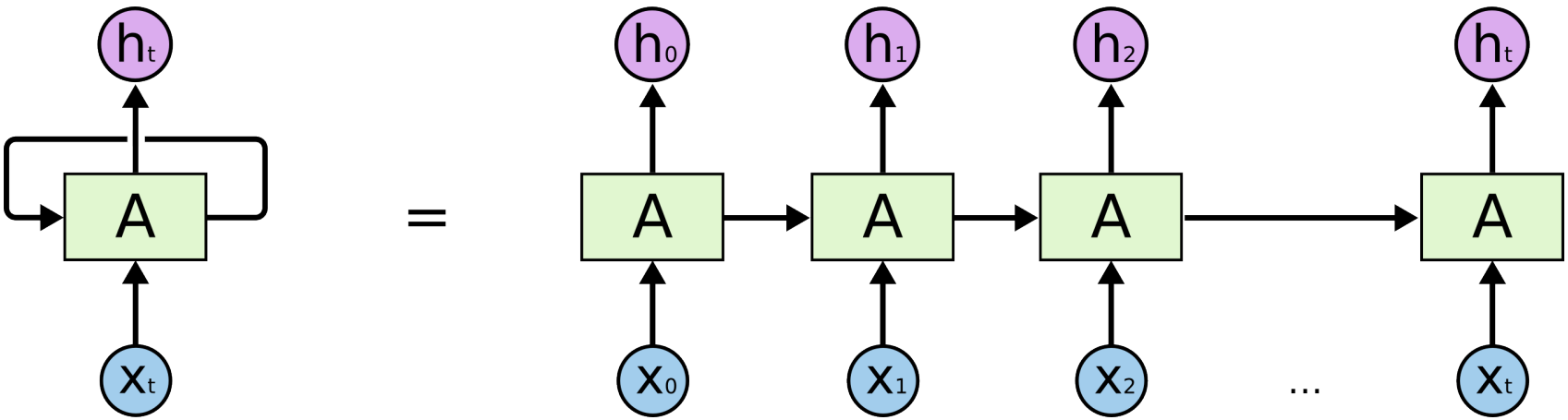


(Y. Kim, 2014, "Convolutional Neural Networks for Sentence Classification")

4/ Advanced neural networks

Recurrent neural networks (RNN):

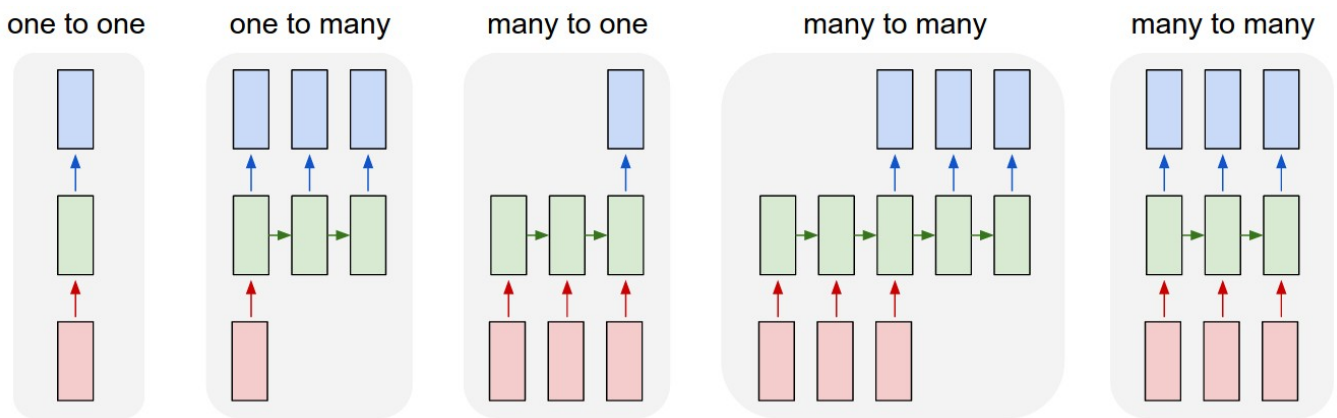
- a single neural network with **memory**
- processes the inputs as a sequence (one after the other)
- at each time step t, the network processes input t and its own previous state $(t-1)$
- trained by *backpropagation through time*



4/ Advanced neural networks

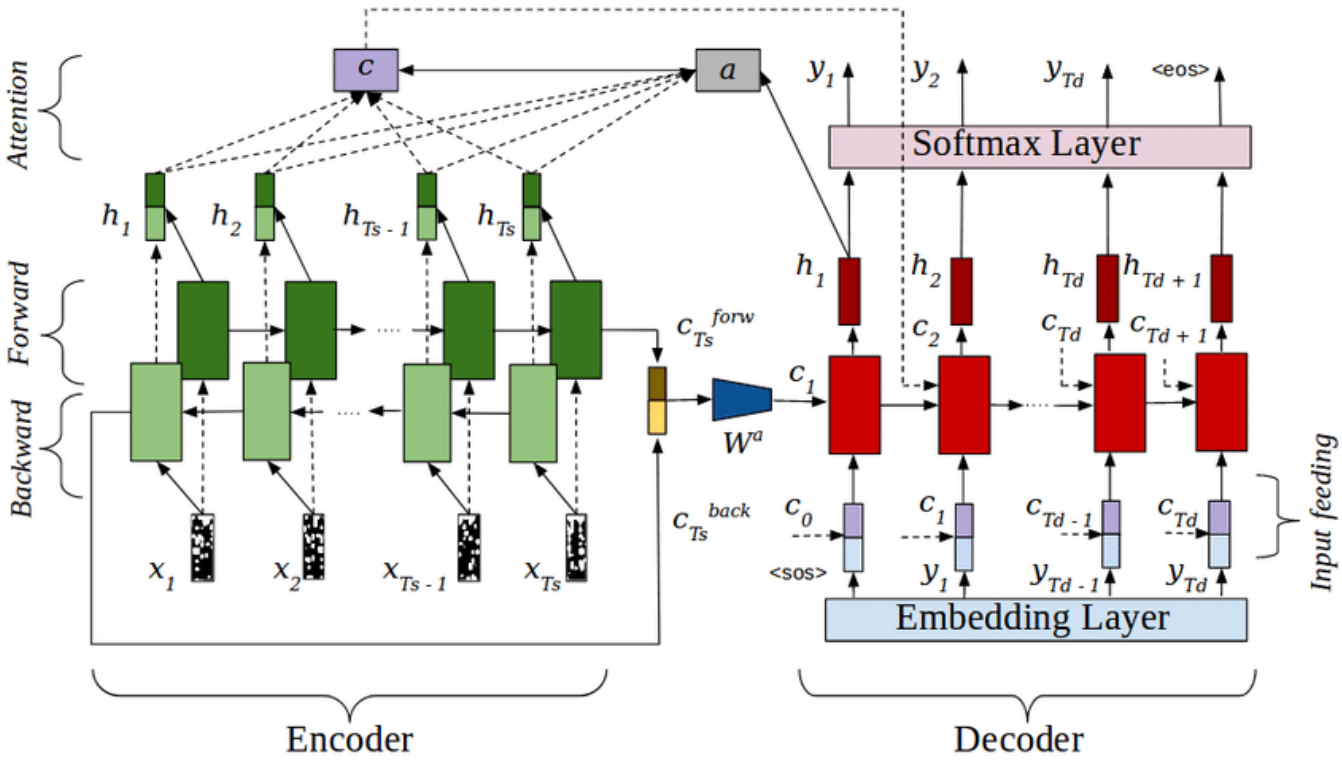
RNNs:

- **Turing-complete** (universal approximation over algorithms!)
- Even more difficult to train than feedforward neural nets
- Improvements: **LSTM** (Hochreiter and Schmidhuber 1997) and **GRU** (Cho *et al.* 2014) use forget-gates to "decide" when to "forget" some information of the past $\rightarrow$ easier to train



4/ Advanced neural networks

RNNs with **attention** have direct access to all previous states, and learn to "decide" which ones are important for the current time step (in encoder-decoder models).



4/ Advanced neural networks

RNNs for NLP?

- Google Translate's 2016 switch to neural translation: LSTMs
- Just as convnets, no longer BOW, and usable with pre-trained word embeddings
- Bidirectional RNN: process text start-to-end and end-to-start simultaneously
- Example: ELMo (Embeddings from Language Models, Peters *et al.* 2018) used bidirectional LSTMs to produce context-based word embeddings
- State of the art NLP before Transformers

NN - Concluding remarks

MLP, deep networks, CNNs, RNNs :

- extremely potent (MLP $<$ CNN $<$ RNN $<$ RNN + attention)
- hard to train! easy to get lost, hard to keep track
- much room for creativity
- user-friendly library (python and R): keras, with tensorflow or torch python backend
- in current NLP, mostly superseded by Transformer models

5/ Character-level language models

Language modeling:

- LM = predicting the probability of a sentence (*ie* is it likely to occur in the corpus the model was trained on?)
- mathematically equivalent to predicting the next word: $P(W_1 \bigcap W_2 \bigcap W_3) = P(W_1) \times P_{W_1}(W_2) \times P_{W_1 \bigcap W_2}(W_3)$
- central task in NLP: machine translation, summarization, speech recognition, grammar correction, dialogue, ...
- traditionally done with trigram models
- RNNs are a prime candidate for this task

5/ Character-level language models

Andrej Karpathy, 2015, "The Unreasonable Effectiveness of Recurrent Neural Networks"

`http://karpathy.github.io/2015/05/21/rnn-effectiveness/`

- LSTMs to predict the next character in a corpus
- Predict character: small dictionary, but hard task
- Generative model: once trained, it can produce new text

5/ Character-level language models

Example of generated text on complete Shakespeare corpus (4.4MB, 3-layer LSTM with 512 hidden nodes each):

```
PANDARUS:
Alas, I think he shall be come approached and the day
When little srain would be attain'd into being never fed,
And who is but a chain and subjects of his death,
I should not sleep.

Second Senator:
They are away this miseries, produced upon my soul,
Breaking and strongly should be buried, when I perish
The earth and thoughts of many states.

DUKE VINCENTIO:
Well, your wit is in the care of side and that.
```

5/ Character-level language models

Continues with several examples: wikipedia (markdown), a book on algebraic geometry (producing LaTeX that almost compiles), Linux source code... all producing very random but somewhat credible text.

I tried it myself on the complete Johnny Hallyday corpus (around 1K songs, 1.1MB), on his death in 2017:

Je veux te voir aimer
Pardon
J'ai vu ce que je t'ai du les chansons
Que tout ce que je t'aime
Que tu sais que tu mens ce que je crois
J'ai la paix de mes mots de toi
Je suis né dans la nuit
Mais je n'ai pas défilé
Je n'ai pas de coeur dans le jour

Je suis mort de la vie
Mais j'ai tout comme un seul pour toi
Le coeur pour partir
Pour dire que je la prend pourtant la vie
Quand je vois que je vois
Et je sais au soleil
C'est plus fort
Pour moi ce **que je t'aime**

5/ Character-level language models

Also on a corpus of French *éditorialistes* (3.4 MB of Christophe Barbier, Bernard-Henri Lévy, Jacques Attali, Alain Duhamel...)

"Au contraire, le président de la démocratie s'applique de l'obsession de la guerre, c'est la construction de la sécurité tout pour autant répondre à la croissance de la réalité et de l'union et de la mondialisation et le commerce de la confiance de la désinvolture de sa terreur sur la mondialisation et au franc pour les solutions communes et la foi des finances de la compétitivité. Cette stratégie ne peut pas donner des riches et de relances d'autorité et de santé et de les autres. Le déficit de la crise est un aide. Le manque de confiance est en constat de la construction des réformes politiques qui ont considéré les déficits pour les confiances politiques et personnelles, des réformes de la construction de la relance des problèmes et des constructions des économistes et des plans de territoire, et celles des politiques politiques et de salariés et de l'avenir et le problème. Mais les confiances sont au pouvoir pour la promesse de la crise et de la recherche. Le changement de promesse sont devenues proposées de nouveaux libre-échanges et des banques sociales. Le régime de la société publique est profonde. Les communistes seront plus forts, à cette politique internationale. Le vide de la défiance est toujours plus autoritaire dans le

Further reading

- Dan Jurafsky and James H. Martin, 2023, *Speech and Language Processing* (3rd ed. draft), chapters I.7, I.9-12, `https://web.stanford.edu/~jurafsky/slp3`
- R. Rojas, 1996, *Neural Networks - A Systematic Introduction*, Springer-Verlag
- I. Goodfellow, Y. Bengio, A. Courville, 2016, *Deep Learning*, MIT Press

Tutorials:

- Tensorflow playground: `https://playground.tensorflow.org`

YouTube:

- 3Blue1Brown "But what is a neural network?" (and following) `https://www.youtube.com/watch?v=aircAruvnKk`

Questions and comments: `julien.boelaert@univ-lille.fr`