Introduction to Informatics for Students of Other Subjects (TUM-BWL)
IN8005

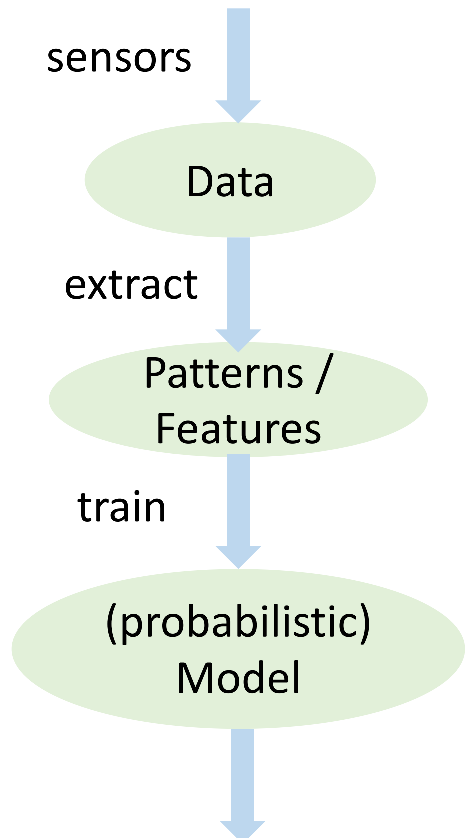
Prof Dr. Georg Groh

Social Computing Research Group

Fakultät für Informatik

Machine Learning / Data-Mining

Goal: Train parametric mapping from data-space D to some desired solution space S (e.g. nominal Ys (classification), continuous Ys (regression) or partitions of X (clustering))



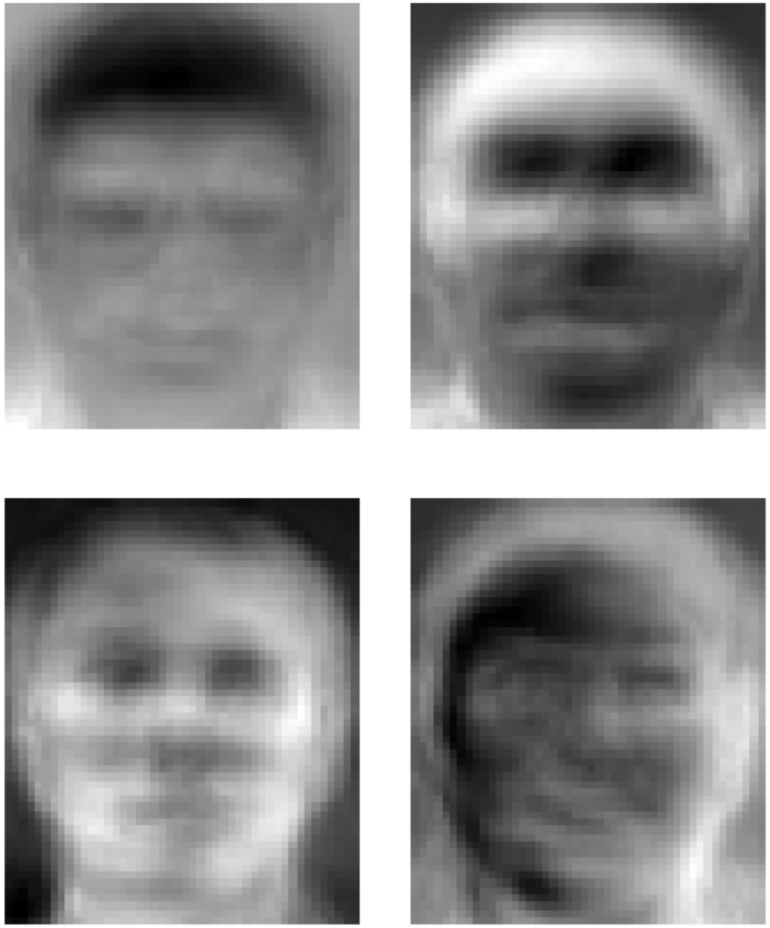
Find clusters, predict values, classify

Data: Feature-/Pattern-Extraction Example

- sensor: camera → images
- feature extraction → Eigenfaces

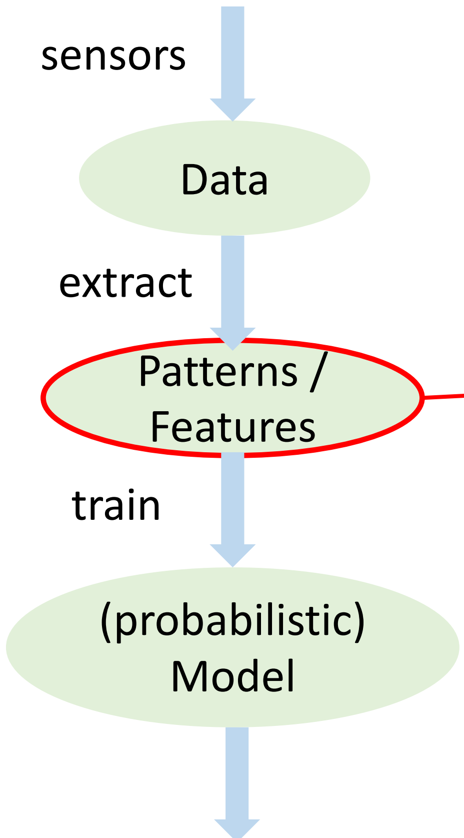
other example:

- sound → 30ms frames → FFT, filtering → MFCCs



[Wikipedia 2016]

Training Data

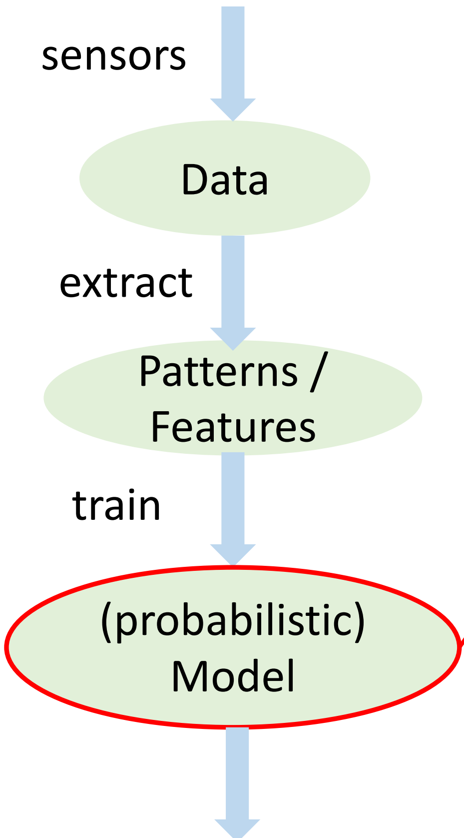


cases:

- $\{x_i\}_{i=1}^{N}$ (unsupervised learning)
- $\{(x_i, y_i)\}_{i=1}^{N}$ (regression, supervised learning)
- $\{(x_i, y_i)\}_{i=1}^{N}$ (classification, supervised learning)

Find clusters, predict values, classify

Training Data



Find clusters, predict values, classify

cases:

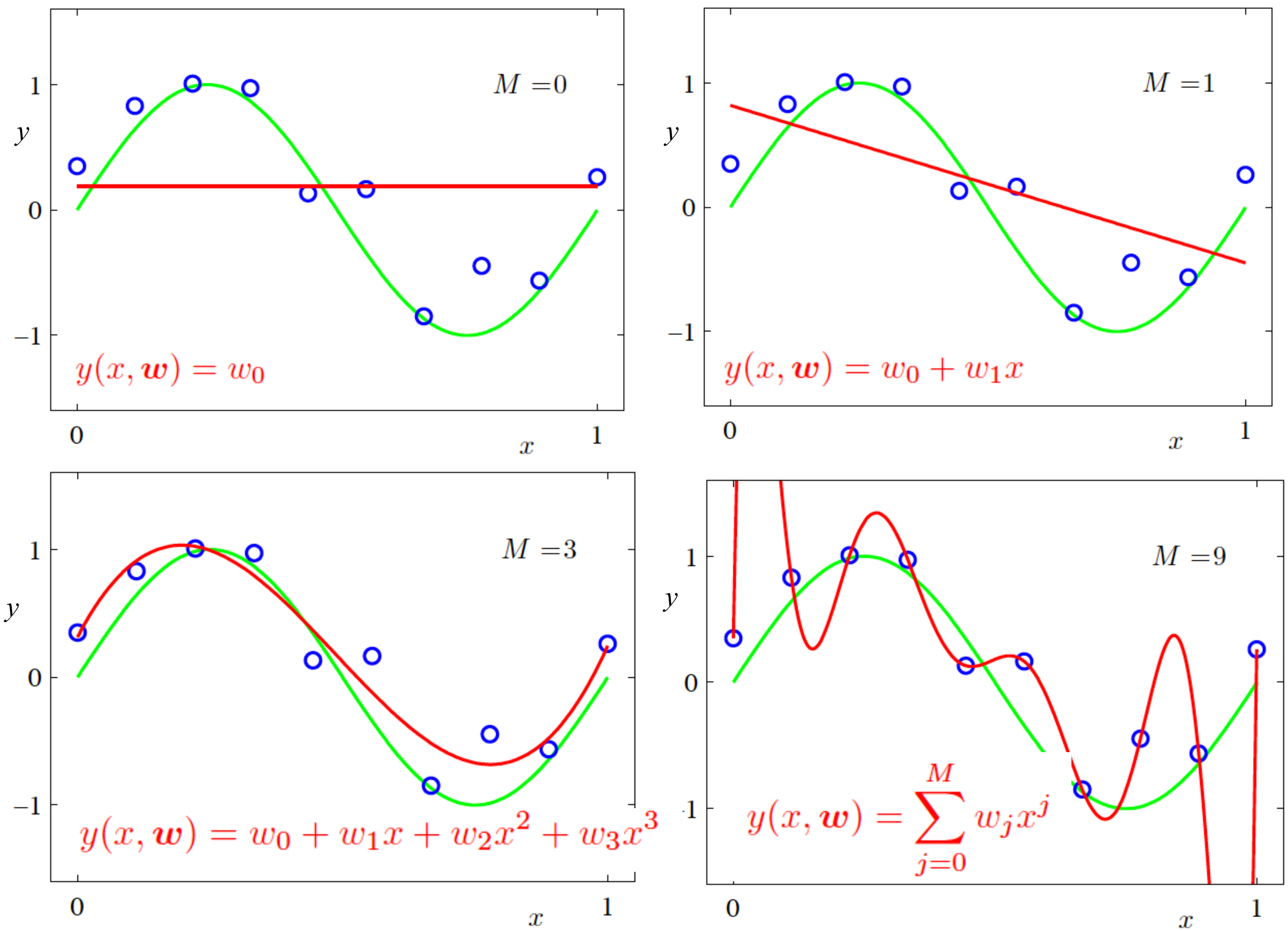
- parametric models (GMMs, Random Forests, Linear Regression, SVMs, Neuronal Networks etc.) vs non-parametric models (KNN, DBScan etc.)
- probabilistic vs. non-probabilistic
- generative vs. discriminative models
- etc.

Bayesian Thinking

$$p(A|B) = \frac{p(B|A)p(A)}{p(B)}$$

$$\begin{aligned} p(\theta_H|n_H, n_T) &\propto \; p(n_H, n_T|\theta_H) \;\; p(\theta_H) \\ &= p(n_H, n_T|\theta_H) \;\; p(\theta_H|\tilde{n}_H, \tilde{n}_T) \\ &\rightarrow p(\theta_H|n_H + \tilde{n}_H, n_T + \tilde{n}_T) \end{aligned}$$

Linear Regression

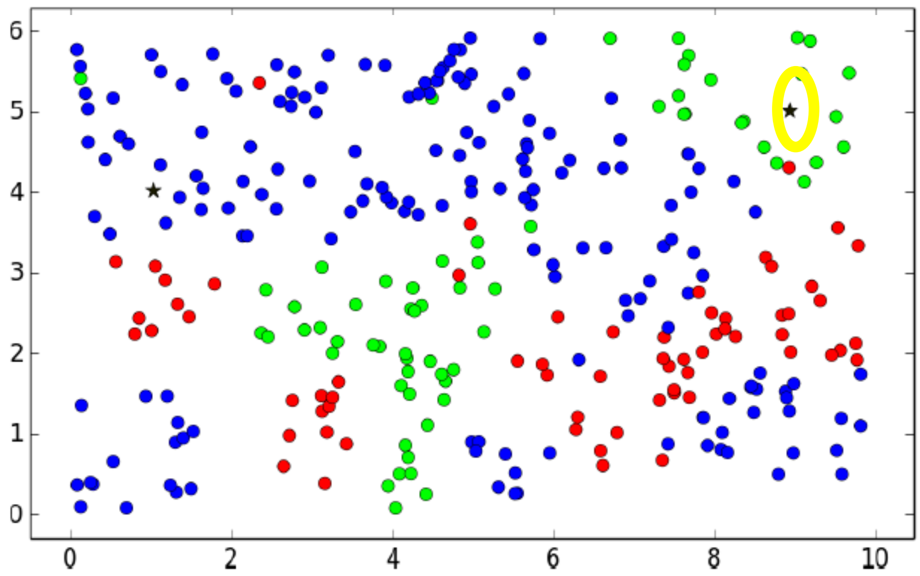


Aspects:
Overfitting,
Regularization

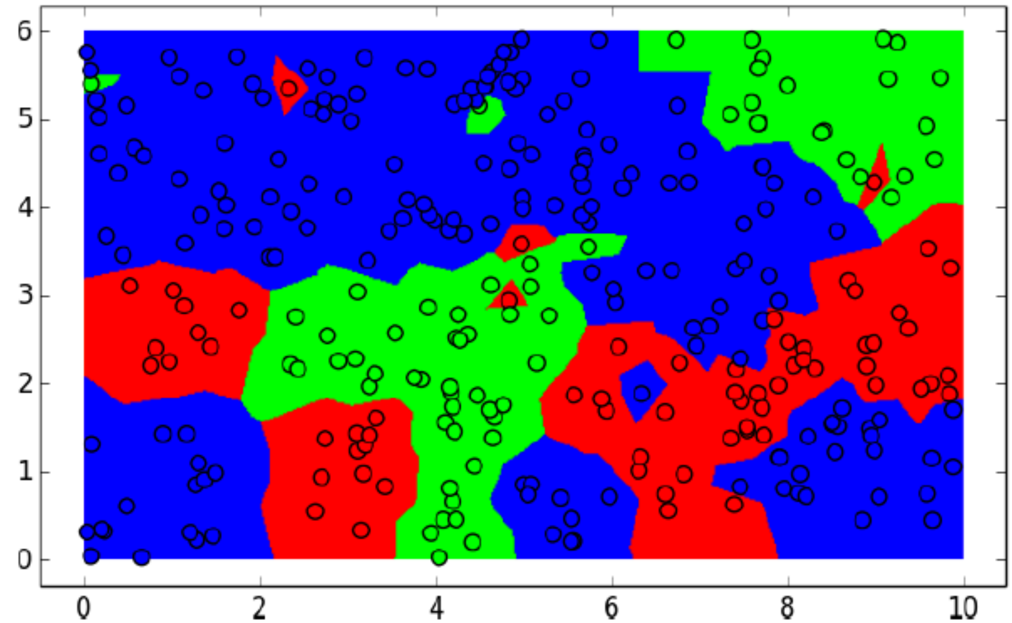
General Model: $y(\boldsymbol{x}, \boldsymbol{w}) = w_0 + \sum_{j=1}^{M-1} w_j \phi_j(\boldsymbol{x}) = \boldsymbol{w}^{\mathsf{T}} \boldsymbol{\phi}(\boldsymbol{x})$

[Bishop, 2005]

Classification with KNN



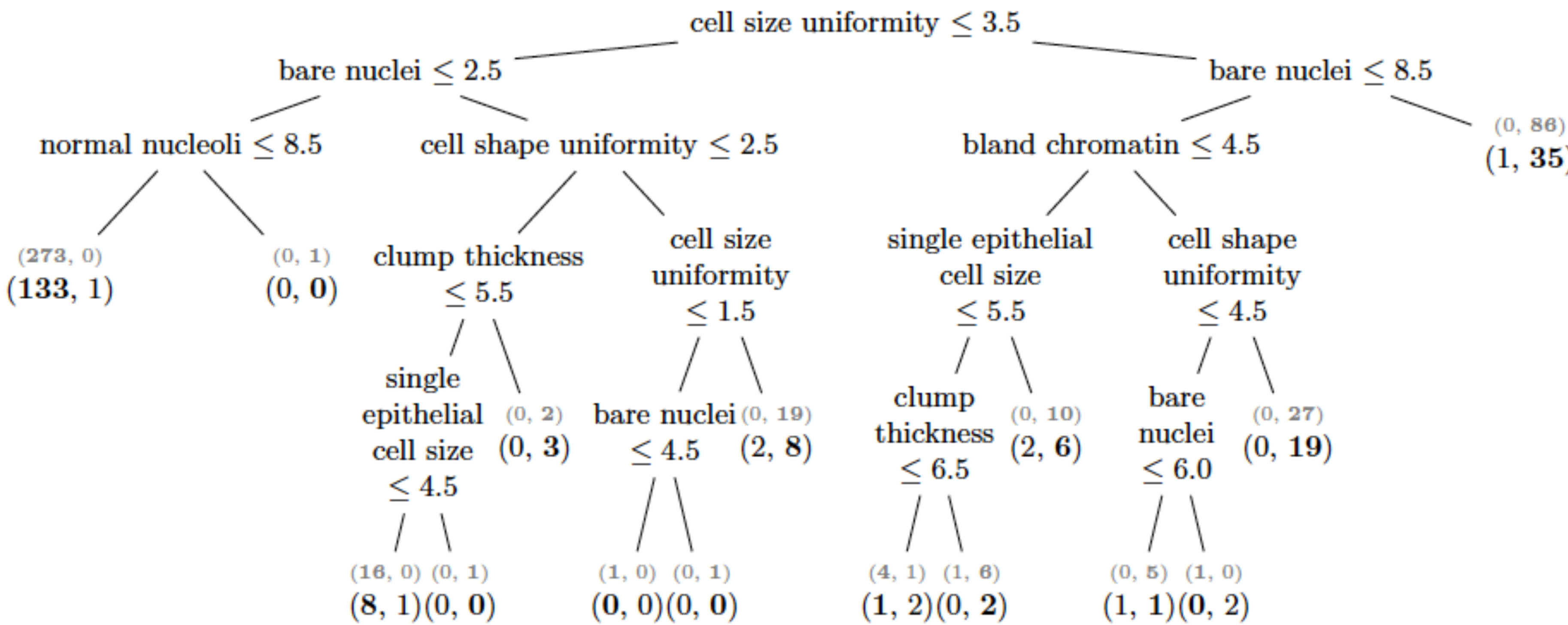
Aspects: Curse of Dimensionality



[Bishop, 2005]

Classification: Decision Trees

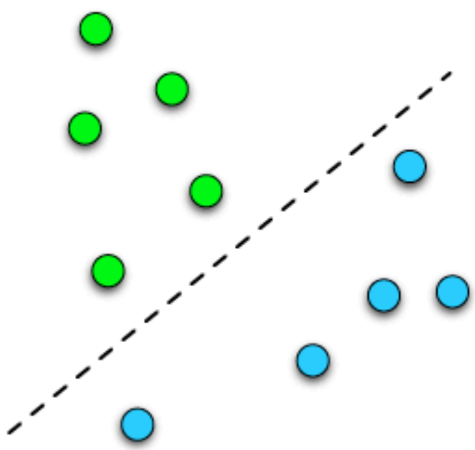
Breast Cancer Decision Tree



Aspects:
Overfitting,
Regularization

[Bishop, 2005]

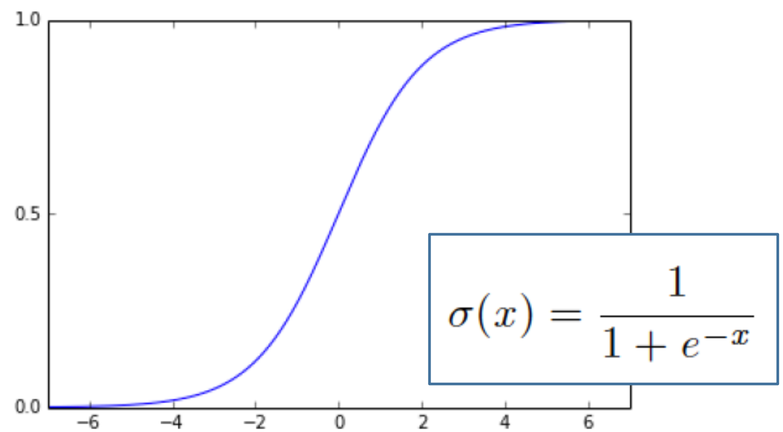
Classification: Logistic Regression



Let a plane be defined by its normal vector $\boldsymbol{w}$ and an offset b.

$$\boldsymbol{x}^T\boldsymbol{w} + b \begin{cases} = 0 & \text{if } \boldsymbol{x} \text{ on the plane} \\ > 0 & \text{if } \boldsymbol{x} \text{ on normal's side of plane} \\ < 0 & \text{else} \end{cases}$$

$$p(z = 1 \mid \boldsymbol{x}) = \sigma(b + \boldsymbol{x}^T\boldsymbol{w}),$$
$$p(z = 0 \mid \boldsymbol{x}) = 1 - \sigma(b + \boldsymbol{x}^T\boldsymbol{w}),$$



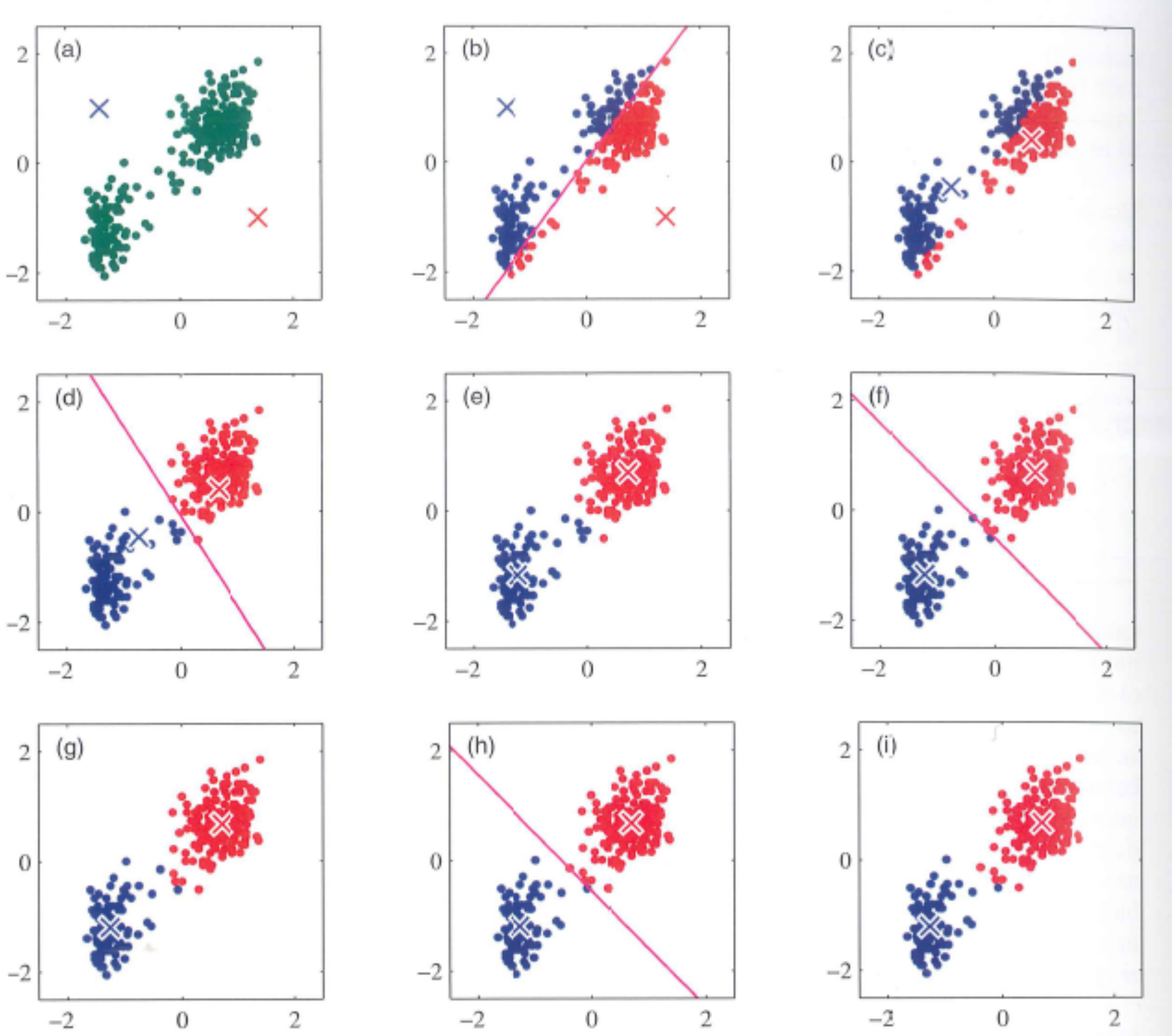
$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

[Bishop, 2005]

Classification: Naive Bayes

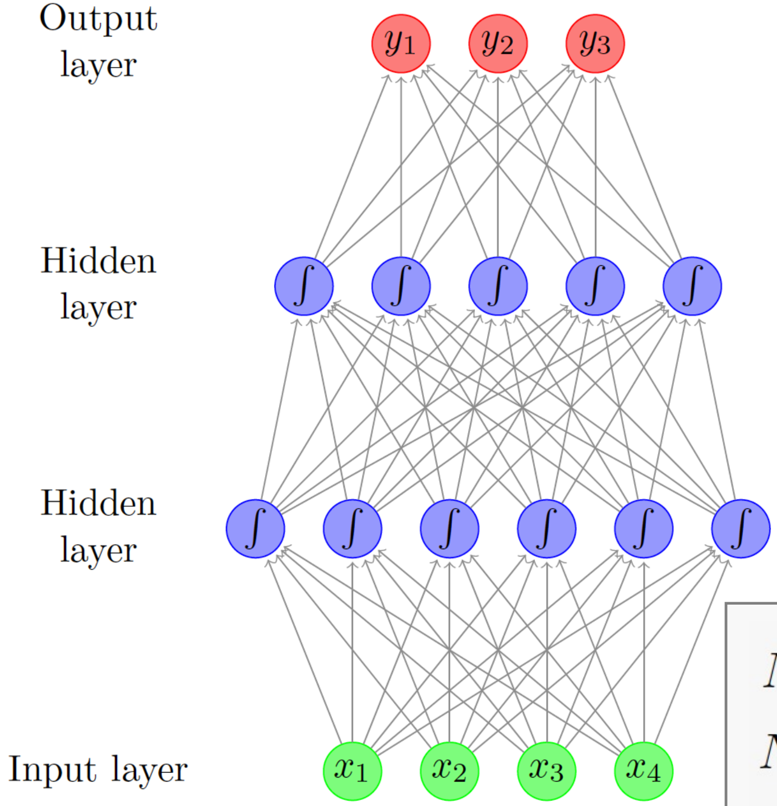
$$\boxed{p(x, y|\theta) = p(x|y, \theta)\, p(y|\theta) = \prod_{v=1}^{V} p(x_v|y, \theta)\, p(y|\theta)}$$

Clustering: K-Means, GMM, EM



[Bishop, 2005]

(Deep) Neural Networks: Supervised Learning



$$NN_{Perceptron}(\mathbf{x}) = \mathbf{x}\mathbf{W} + \mathbf{b}$$
$$\mathbf{x} \in \mathbb{R}^{d_{in}}, \quad \mathbf{W} \in \mathbb{R}^{d_{in} \times d_{out}}, \quad \mathbf{b} \in \mathbb{R}^{d_{out}}$$

Simple Perceptron

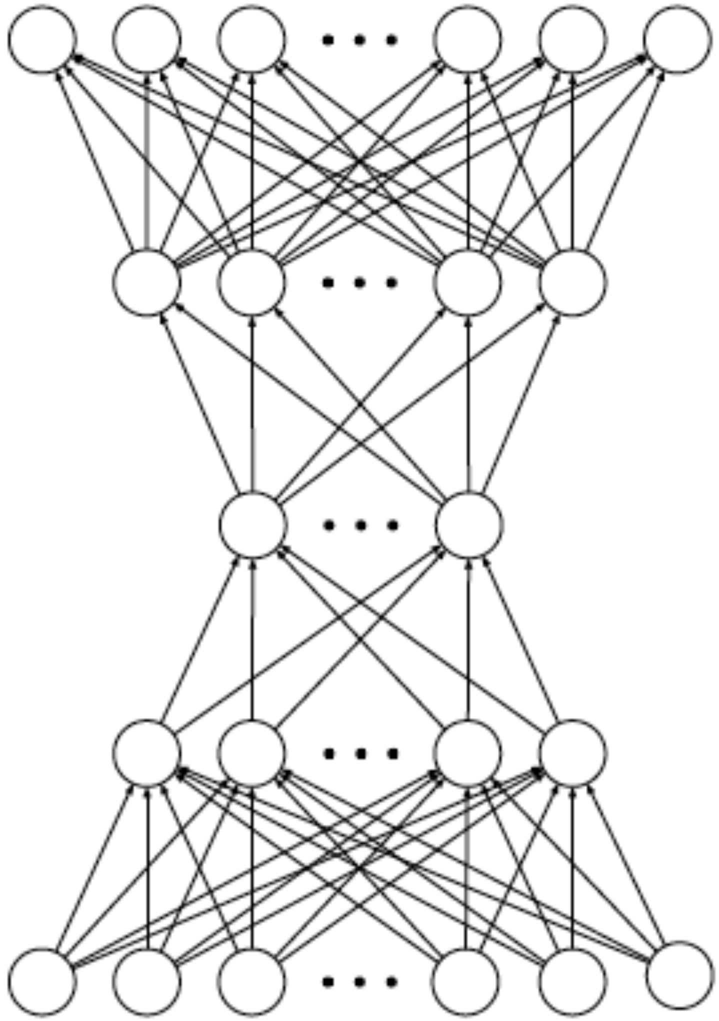
$$NN_{MLP1}(\mathbf{x}) = g(\mathbf{x}\mathbf{W^1} + \mathbf{b^1})\mathbf{W^2} + \mathbf{b^2}$$
$$\mathbf{x} \in \mathbb{R}^{d_{in}}, \quad \mathbf{W^1} \in \mathbb{R}^{d_{in} \times d_1}, \quad \mathbf{b^1} \in \mathbb{R}^{d_1},$$
$$\mathbf{W^2} \in \mathbb{R}^{d_1 \times d_2}, \quad \mathbf{b^2} \in \mathbb{R}^{d_2}$$

One hidden layer FFNN, no output transformation

$$NN_{MLP2}(\mathbf{x}) = (g^2(g^1(\mathbf{x}\mathbf{W^1} + \mathbf{b^1})\mathbf{W^2} + \mathbf{b^2}))\mathbf{W^3}$$
$$\begin{aligned} NN_{MLP2}(\mathbf{x}) =& \mathbf{y} \\ \mathbf{h^1} =& g^1(\mathbf{x}\mathbf{W^1} + \mathbf{b^1}) \\ \mathbf{h^2} =& g^2(\mathbf{h^1}\mathbf{W^2} + \mathbf{b^2}) \\ \mathbf{y} =& \mathbf{h^2}\mathbf{W^3} \end{aligned}$$

Two hidden layers FFNN, no output transformation

(Deep) Neural Networks: Unsupervised



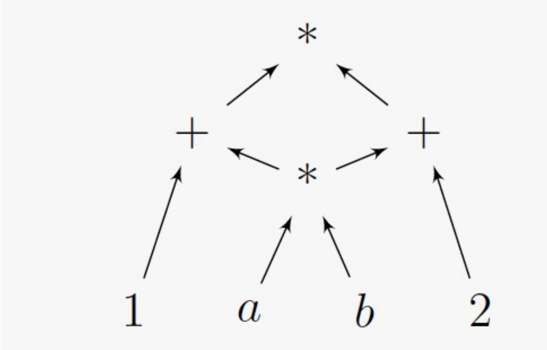
Auto-Encoder Network

[ML1, 2016]

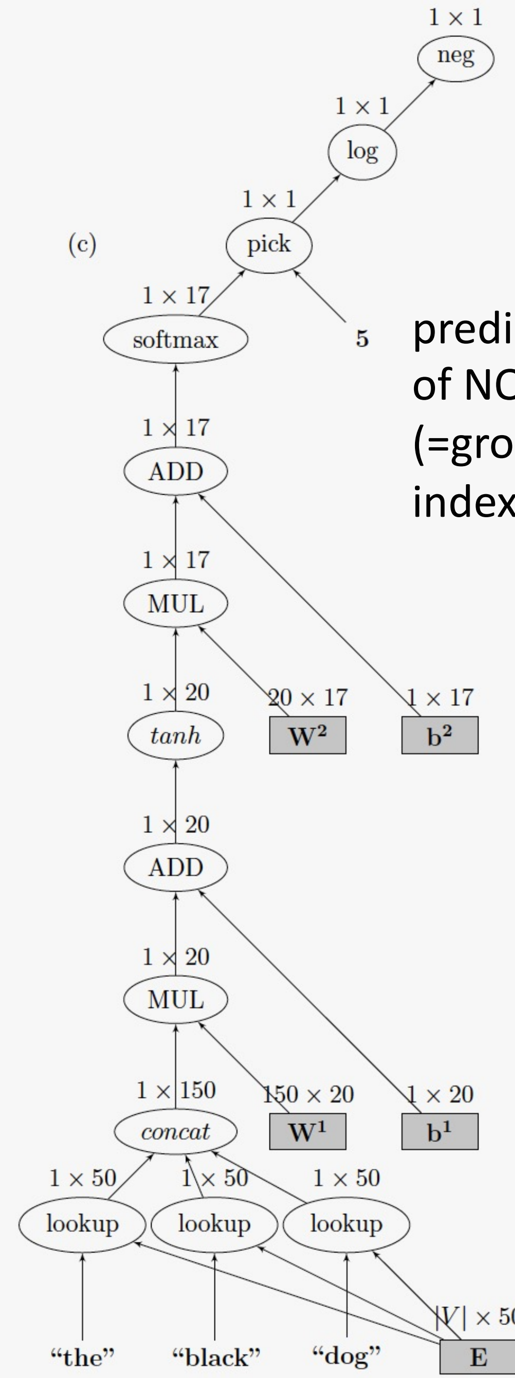
Computation Graph Abstraction

abstract the computations
(e.g. forward pass) in a DAG

$(a * b + 1) * (a * b + 2)$:



simple computation



predicted output
of NOUN
(=ground truth,
index=5)

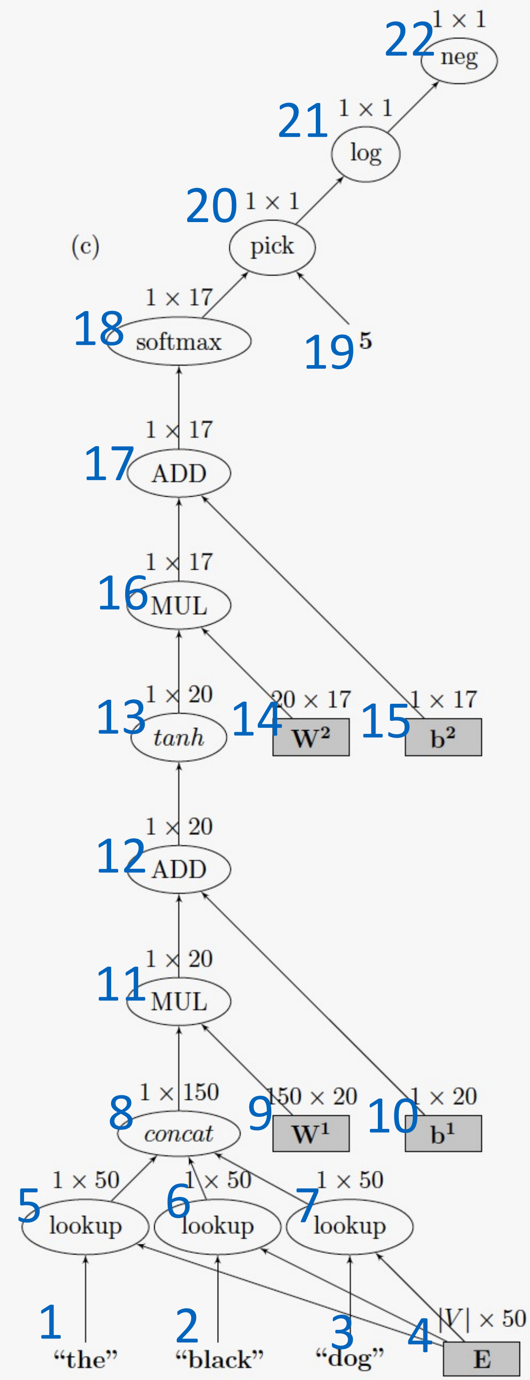
computation graph for a 2 layer FF-NN
predicting probability distribution over 17
possible POS for third word of 3 word sequence

Forward Computation

- compute a topological ordering for DAG (V, E) (an ordering such that $i < j$ if $(i, j) \in E$) (possible in linear time)
- f_i : function computed by node i
- $\pi(i)$: children[1] of node i
- $\pi^{-1}(i)$: parents of node i (outputs of $\pi^{-1}(i)$ are the arguments of f_i)
- $v(i)$: output of node i (applying f_i to the output of the $\pi^{-1}(i)$)
- for constants, variable (parameter) nodes and input nodes: f_i is a constant function and $\pi^{-1}(i)$ is empty

Algorithm 3 Computation Graph Forward Pass

for i = 1 to N **do**
    Let $a_1, \ldots, a_m = \pi^{-1}(i)$
    $v(i) \leftarrow f_i(v(a_1), \ldots, v(a_m))$



regard that in [1] children and parents are defined in reverse order of the edges; we stick to the usual notion of children and parent: j is child of i and i parent of j if $(i, j) \in E$

Backward Computation

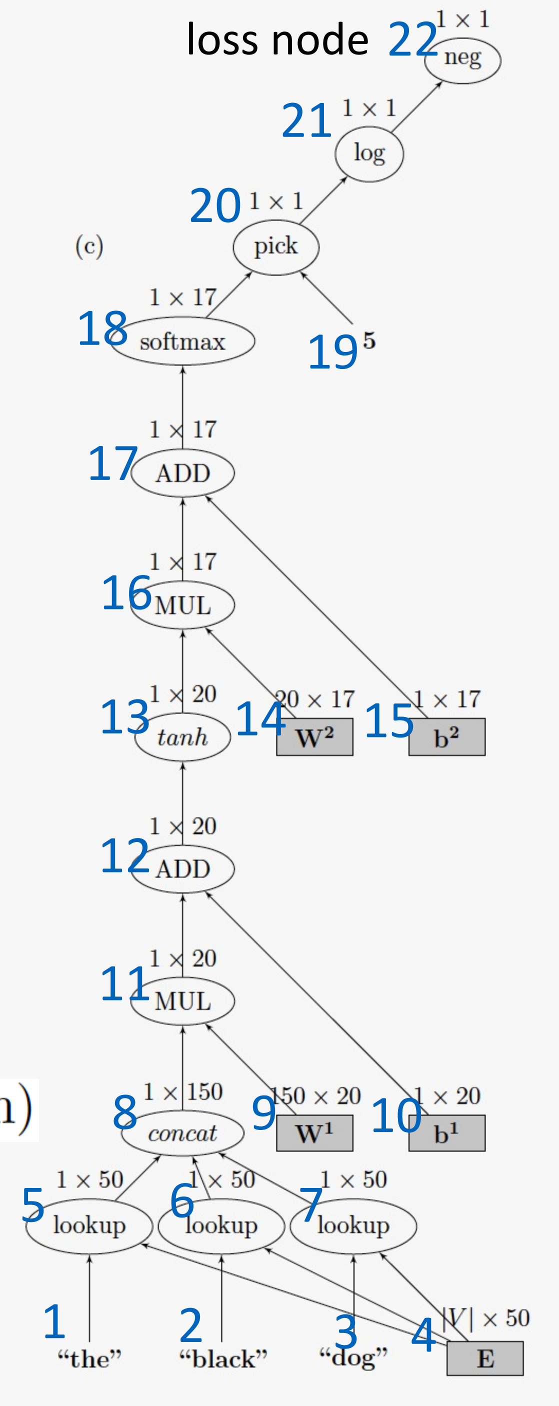
- select the loss node N (usually the last node in the topological ordering)
- $d(i) := \frac{\partial N}{\partial i}$: in the end contains the elements of the gradient w.r.t. the parameter nodes
- $\frac{\partial f_j}{\partial i} = \frac{\partial}{\partial\, i \in \pi^{-1}(j)} f_j(v(\pi^{-1}(j)))$
  where $v(\pi^{-1}(j)) := v(a_1), v(a_2), \ldots, v(a_m)$ are the outputs of $a_1, a_2, \ldots, a_m := \pi^{-1}(j)$
- $\pi(i)$: children of node i

Algorithm 4 Computation Graph Backward Pass (Backpropagation)

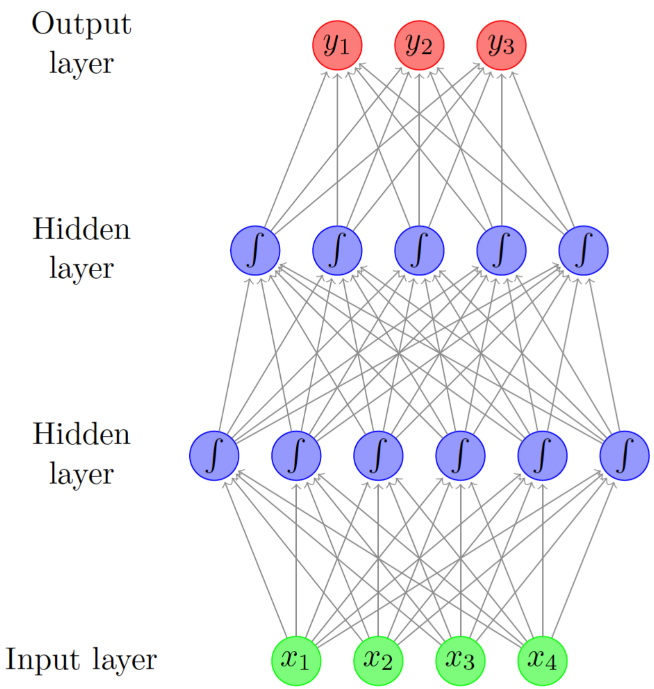
$d(N) \leftarrow 1$

for i = N-1 to 1 **do**

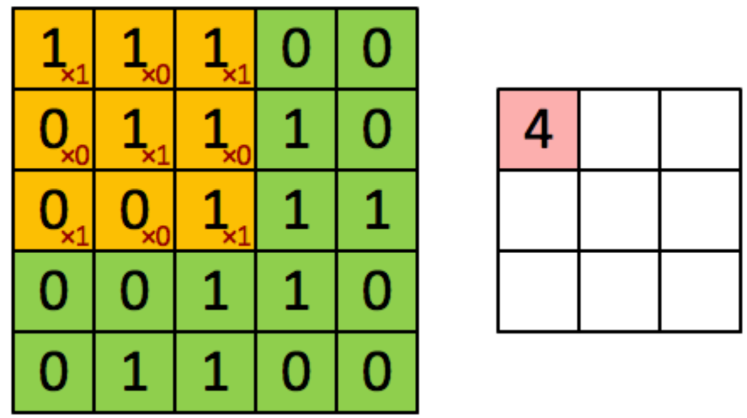
$\quad d(i) \leftarrow \sum_{j \in \pi(i)} d(j) \cdot \frac{\partial f_j}{\partial i}$



Convolutional Neural Networks



2d convolution:



$$\begin{bmatrix} 1 & 2 & 3 \\ 9 & 6 & 5 \\ 2 & 3 & 1 \\ 7 & 8 & 1 \\ 3 & 4 & 1 \end{bmatrix} \xrightarrow{\text{max pooling over columns}} \begin{bmatrix} 9 & 8 & 5 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 & 3 \\ 9 & 6 & 5 \\ 2 & 3 & 1 \\ 7 & 8 & 1 \\ 3 & 4 & 1 \end{bmatrix} \xrightarrow{\text{2-max pooling over columns}} \begin{bmatrix} 9 & 6 & 3 \\ 7 & 8 & 5 \end{bmatrix}$$

[ML1, 2016]

Recurrent Neural Networks

$$RNN(\mathbf{s_0}, \mathbf{x_{1:n}}) = \mathbf{s_{1:n}},\ \mathbf{y_{1:n}}$$

$$\mathbf{s_i} = R(\mathbf{s_{i-1}}, \mathbf{x_i}) \qquad \mathbf{x_i} \in \mathbb{R}^{d_{in}},\ \ \mathbf{y_i} \in \mathbb{R}^{d_{out}},$$

$$\mathbf{y_i} = O(\mathbf{s_i}) \qquad \mathbf{s_i} \in \mathbb{R}^{f(d_{out})}$$

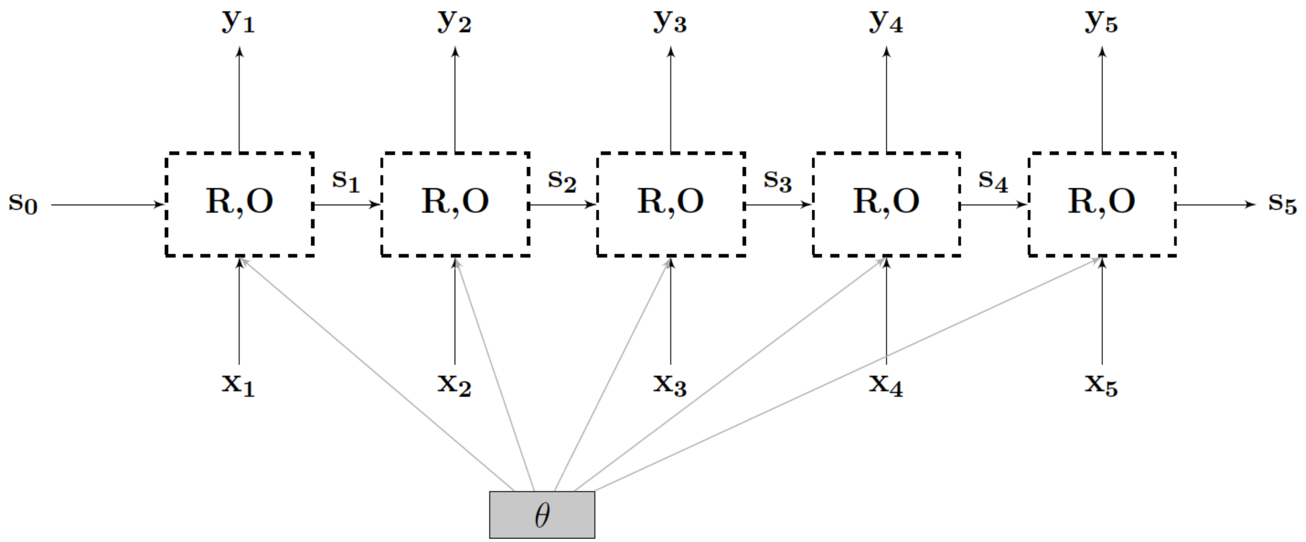
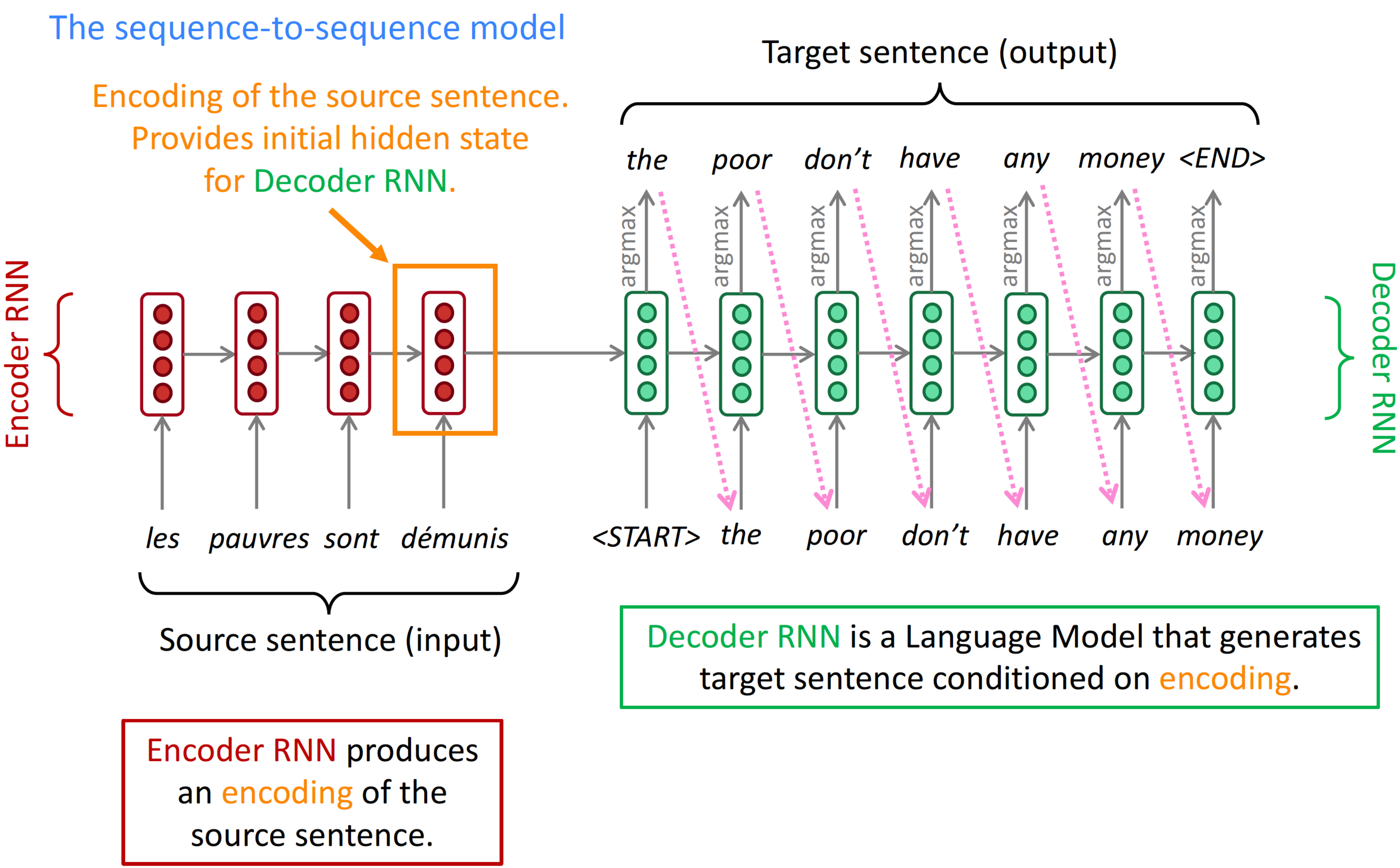


Figure 6: Graphical representation of an RNN (unrolled).

[Goldberg 2017]

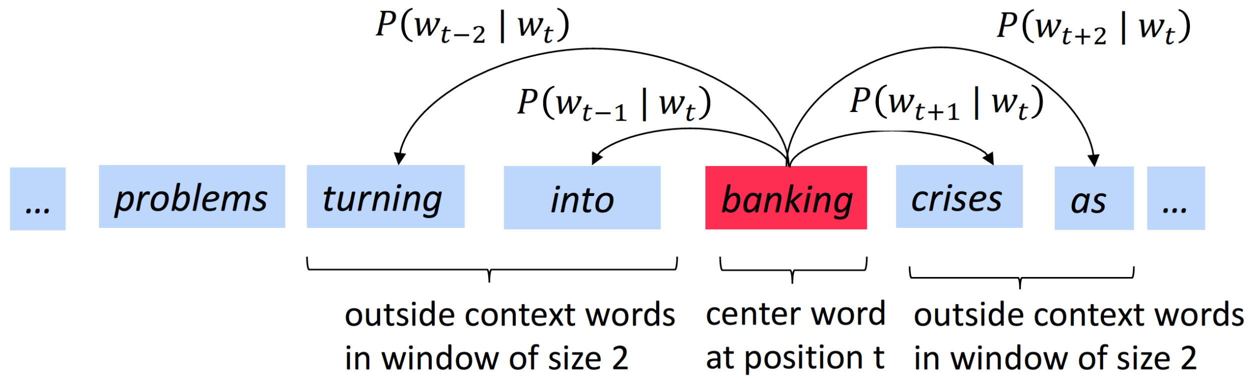
Recurrent Neural Networks (e.g. for Machine Translation)



[Socher et al 2018]

word2Vec

word2vec
basic approach:
skip-gram



$$L(\theta) = \prod_{t=1}^{T} \prod_{\substack{-m \le j \le m \\ j \ne 0}} P\left(w_{t+j} \mid w_t; \theta\right) \qquad J(\theta) = -\frac{1}{T} \sum_{t=1}^{T} \sum_{\substack{-m \le j \le m \\ j \ne 0}} \log P\left(w_{t+j} \mid w_t; \theta\right)$$

Recurrent Neural Networks (e.g. for Machine Translation)

GPT-2 language model (cherry-picked) output

SYSTEM PROMPT (HUMAN-WRITTEN)

In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.

MODEL COMPLETION (MACHINE-WRITTEN, 10 TRIES)

The scientist named the population, after their distinctive horn, Ovid's Unicorn. These four-horned, silver-white unicorns were previously unknown to science.

Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally solved.

Dr. Jorge Pérez, an evolutionary biologist from the University of La Paz, and several companions, were exploring the Andes Mountains when they found a small valley, with no other animals or humans. Pérez noticed that the valley had what appeared to be a natural fountain, surrounded by two peaks of rock and silver snow.

Pérez and the others then ventured further into the valley. …

[Manning et al 2020]

What is Intelligence?

Assigning a Probability to a Sequence of Words

- **Language model**: assignment of (joint) probabilities to sequences of words:

$$P(w_1, w_2, \ldots, w_n) = P(w_1^n) = P(w_1)P(w_2|w_1)P(w_3|w_1^2)\ldots P(w_n|w_1^{n-1})$$
$$= \prod_{k=1}^{n} P(w_k|w_1^{k-1})$$

notation:
$w_n^m = w_{n:m}$

or (equivalently) modelling conditional probabilities
(e.g. for predicting next word)

$$P(w_n|w_1^{n-1})$$

- MLE based probability estimation: counting instances of sequences in corpora: e.g.

$$P(\textit{the}|\textit{its water is so transparent that}) = \frac{C(\textit{its water is so transparent that the})}{C(\textit{its water is so transparent that})}$$

problem: language is creative, combinatorial explosion
→ only very few instances of given sequence → sparsity, poor estimates

N-Gram Models

- Solution: Assumption: restrict chain rule to Markov order N → N-Gram Models:

$$P(w_n|w_1^{n-1}) \approx P(w_n|w_{n-N+1}^{n-1})$$

- N=2 : Bigram model

$$P(w_n|w_1^{n-1}) \approx P(w_n|w_{n-1})$$

$$P(w_1^n) \approx \prod_{k=1}^{n} P(w_k|w_{k-1})$$

- MLE for Bigram model: Count $C(w_{n-1}w_n)$ of bigram $w_{n-1}w_n$ in corpus →

$$P(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)}{\sum_w C(w_{n-1}w)} = \frac{C(w_{n-1}w_n)}{C(w_{n-1})}$$