

# Computational Linguistics

CSC 485/2501  
Fall 2023

# 2A

## 2A. Dependency Grammar

Gerald Penn  
Department of Computer Science, University of Toronto

Based on slides by Roger Levy, Yuji Matsumoto, Dragomir Radev, Dan Roth, David Smith and Jason Eisner

Copyright © 2019  
Gerald Penn. All  
rights reserved.

# Word Dependency Parsing

## Raw sentence

He reckons the current account deficit will narrow to only 1.8 billion in September.



Part-of-speech tagging

## POS-tagged sentence

He reckons the current account deficit will narrow to only 1.8 billion in September.

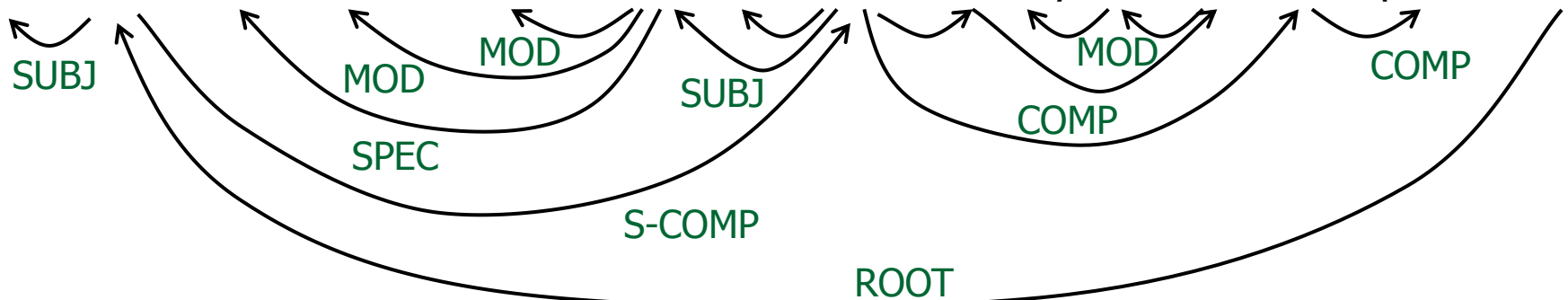
PRP VBZ DT JJ NN NN MD VB TO RB CD CD IN NNP .



Word dependency parsing

## Word dependency parsed sentence

He reckons the current account deficit will narrow to only 1.8 billion in September .



# Dependency Graphs

- ▶ A dependency structure can be defined as a directed graph  $G$ , consisting of
  - ▶ a set  $V$  of nodes,
  - ▶ a set  $E$  of arcs (edges),
  - ▶ a linear precedence order  $<$  on  $V$ .
- ▶ Labeled graphs:
  - ▶ Nodes in  $V$  are labeled with word forms (and annotation).
  - ▶ Arcs in  $E$  are labeled with dependency types.
- ▶ Notational conventions ( $i, j \in V$ ):
  - ▶  $i \rightarrow j \equiv (i, j) \in E$
  - ▶  $i \rightarrow^* j \equiv i = j \vee \exists k : i \rightarrow k, k \rightarrow^* j$

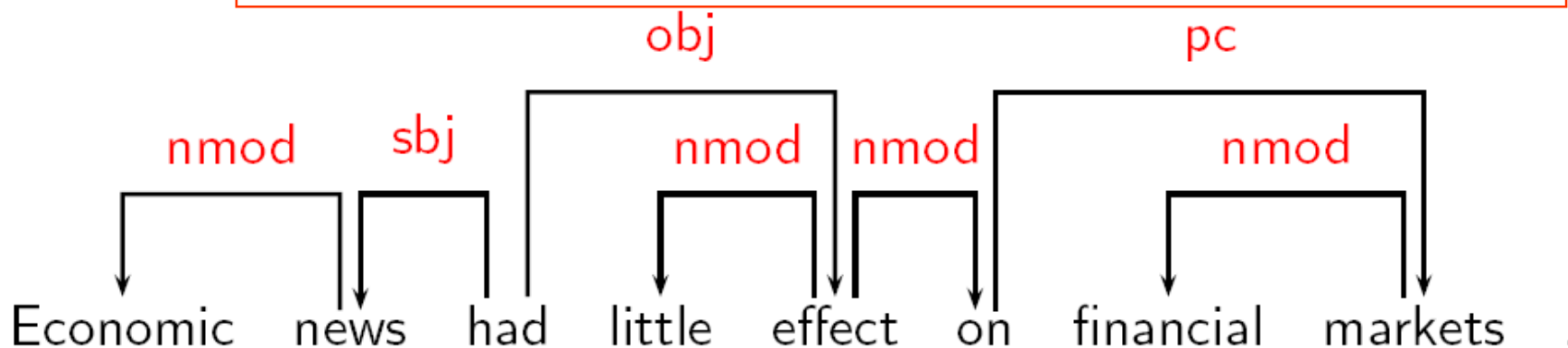
# Formal Conditions on Dependency Graphs

- ▶  $G$  is (weakly) **connected**:
  - ▶ For every node  $i$  there is a node  $j$  such that  $i \rightarrow j$  or  $j \rightarrow i$ .
- ▶  $G$  is **acyclic**:
  - ▶ If  $i \rightarrow j$  then not  $j \rightarrow^* i$ .
- ▶  $G$  obeys the **single-head** constraint:
  - ▶ If  $i \rightarrow j$ , then not  $k \rightarrow j$ , for any  $k \neq i$ .
- ▶  $G$  is **projective**:
  - ▶ If  $i \rightarrow j$  then  $i \rightarrow^* k$ , for any  $k$  such that  $i < k < j$  or  $j < k < i$ .

# Connectedness, Acyclicity and Single-Head

- ▶ Intuitions:
  - ▶ Syntactic structure is complete (**Connectedness**).
  - ▶ Syntactic structure is hierarchical (**Acyclicity**).
  - ▶ Every word has at most one syntactic head (**Single-Head**).
- ▶ Connectedness can be enforced by adding a special root node.

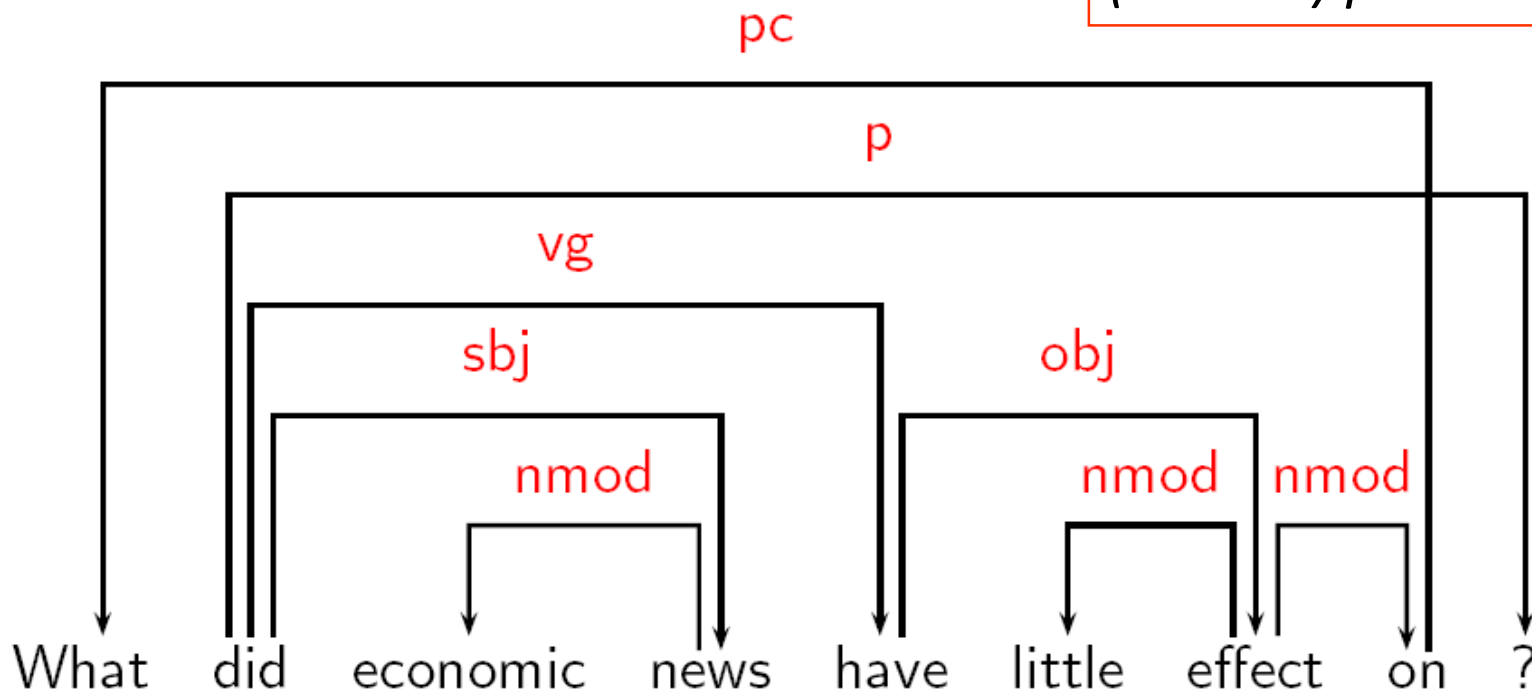
*All these conditions will be violated for semantic dependency graphs we will consider later*



# Projectivity

- ▶ Most theoretical frameworks do **not** assume projectivity.
- ▶ Non-projective structures are needed to account for
  - ▶ long-distance dependencies,
  - ▶ free word order.

*You can think of it as  
(related) planarity*



# Underspecifications of simple typed dependencies

- ▶ Flat bracketings

- ▶ Non-projective dependency

*A woman arrived who was wearing a hat*

- ▶ Complex word-word dependency constructions:

- ▶ Predicative adjectives

*I ate the fish naked/raw*

- ▶ Coordination

*Pat and Terry sat and laughed*

- ▶ More generally, semantic roles:

*The door opened*

*Erin opened the door*

*The door opened a crack*

- ▶ Quantifier scoping, temporal interpretation and so forth

# Shift-Reduce Type Algorithms

- ▶ Data structures:
  - ▶ Stack  $[\dots, w_i]_S$  of partially processed tokens
  - ▶ Queue  $[w_j, \dots]_Q$  of remaining input tokens
- ▶ Parsing actions built from atomic actions:
  - ▶ Adding arcs  $(w_i \rightarrow w_j, w_i \leftarrow w_j)$
  - ▶ Stack and queue operations
- ▶ Left-to-right parsing in  $O(n)$  time
- ▶ Restricted to **projective** dependency graphs



# Yamada's Algorithm

- ▶ Three parsing actions:

$$\text{Shift} \quad \frac{[\dots]_S \quad [w_i, \dots]_Q}{[\dots, w_i]_S \quad [\dots]_Q}$$

$$\text{Left} \quad \frac{[\dots, w_i, w_j]_S \quad [\dots]_Q}{[\dots, w_i]_S \quad [\dots]_Q} \quad w_i \rightarrow w_j$$

$$\text{Right} \quad \frac{[\dots, w_i, w_j]_S \quad [\dots]_Q}{[\dots, w_j]_S \quad [\dots]_Q} \quad w_i \leftarrow w_j$$

- ▶ Algorithm variants:
  - ▶ Originally developed for Japanese (strictly head-final) with only the **Shift** and **Right** actions [Kudo and Matsumoto 2002]
  - ▶ Adapted for English (with mixed headedness) by adding the **Left** action [Yamada and Matsumoto 2003]
  - ▶ Multiple passes over the input give time complexity  $O(n^2)$

# Nivre's Algorithm

- ▶ Four parsing actions:

$$\text{Shift} \quad \frac{[\dots]_S \quad [w_i, \dots]_Q}{[\dots, w_i]_S \quad [\dots]_Q}$$

$$\text{Reduce} \quad \frac{[\dots, w_i]_S \quad [\dots]_Q \quad \exists w_k : w_k \rightarrow w_i}{[\dots]_S \quad [\dots]_Q}$$

$$\text{Left-Arc}_r \quad \frac{[\dots, w_i]_S \quad [w_j, \dots]_Q \quad \neg \exists w_k : w_k \rightarrow w_i}{[\dots]_S \quad [w_j, \dots]_Q \quad w_i \stackrel{r}{\leftarrow} w_j}$$

$$\text{Right-Arc}_r \quad \frac{[\dots, w_i]_S \quad [w_j, \dots]_Q \quad \neg \exists w_k : w_k \rightarrow w_j}{[\dots, w_i, w_j]_S \quad [\dots]_Q \quad w_i \stackrel{r}{\rightarrow} w_j}$$

- ▶ Characteristics:

- ▶ Integrated labeled dependency parsing
- ▶ Arc-eager processing of right-dependents
- ▶ Two passes over the input gives time complexity  $O(n)$

# Example

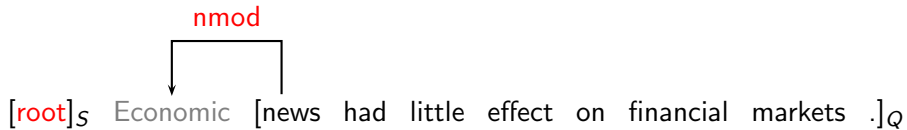
[root]<sub>S</sub> [Economic news had little effect on financial markets .]<sub>Q</sub>

# Example

[root Economic]<sub>S</sub> [news had little effect on financial markets .]<sub>Q</sub>

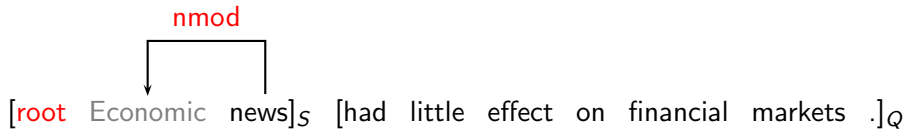
Shift

# Example



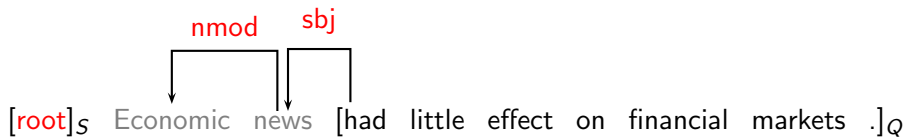
Left-Arc<sub>nmod</sub>

# Example



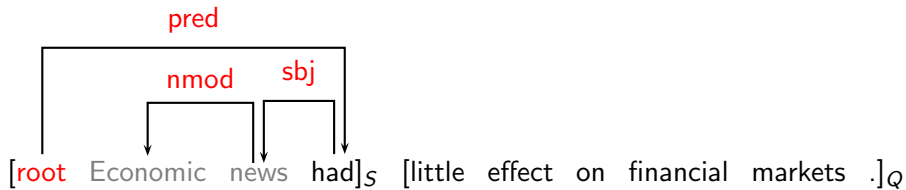
Shift

# Example



Left-Arc<sub>*subj*</sub>

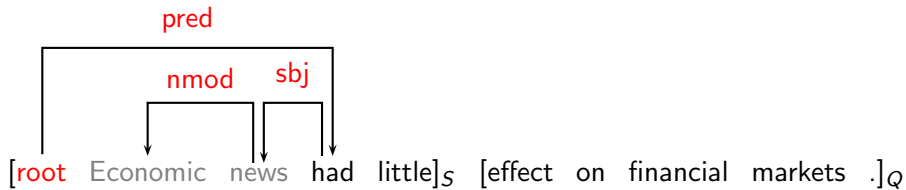
# Example



Right-Arc<sub>pred</sub>

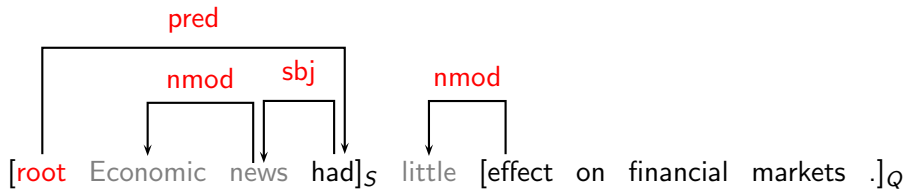


# Example



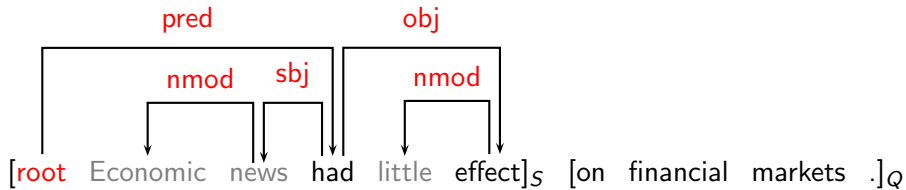
Shift

# Example



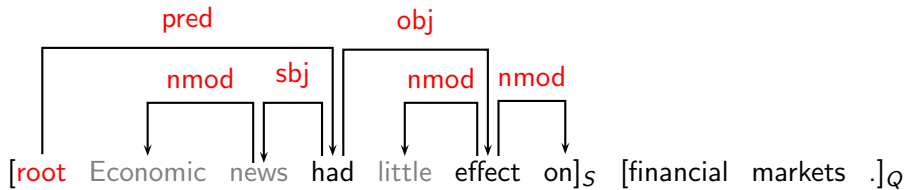
Left-Arc<sub>nmod</sub>

# Example



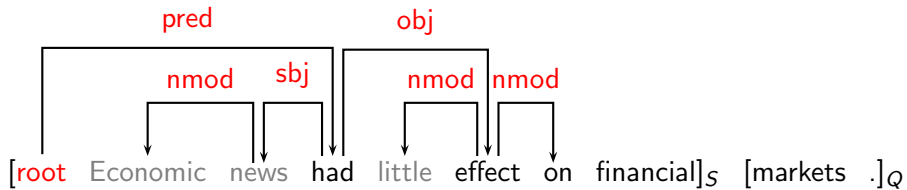
Right-Arc<sub>obj</sub>

# Example



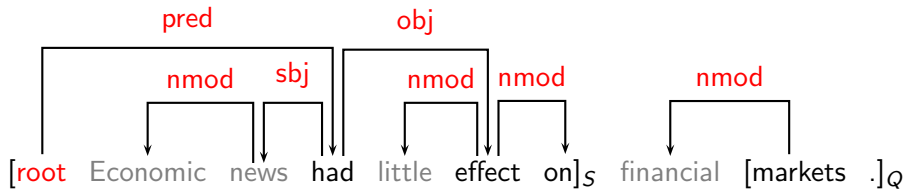
Right-Arc<sub>nmod</sub>

# Example



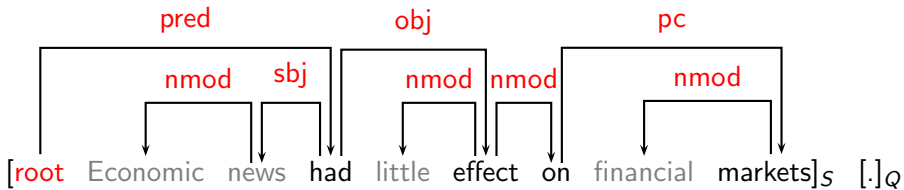
Shift

# Example



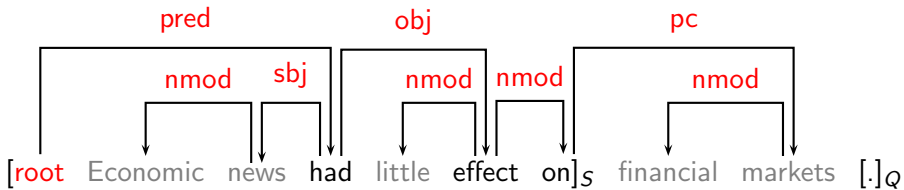
Left-Arc<sub>nmod</sub>

# Example



Right-Arc<sub>pc</sub>

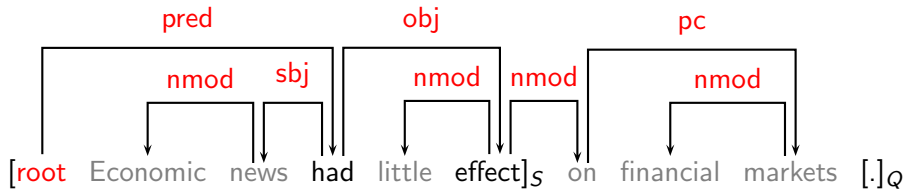
# Example



Reduce

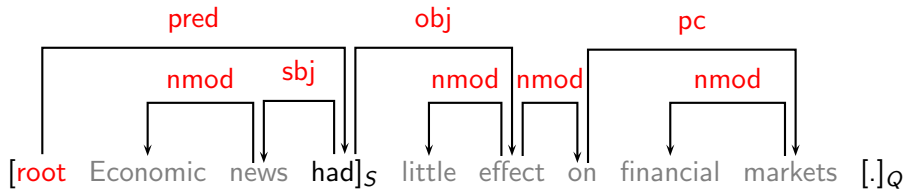


# Example



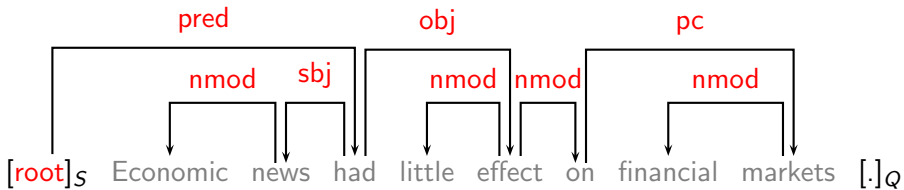
Reduce

# Example



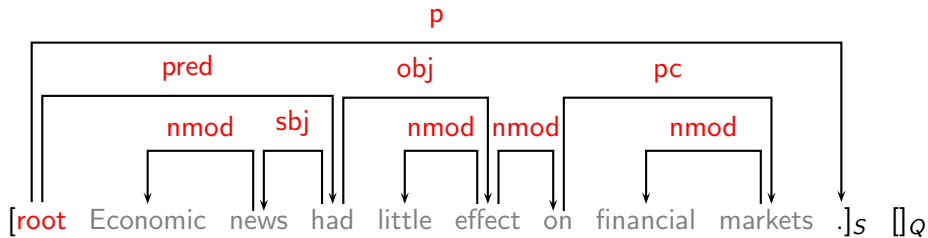
Reduce

# Example



Reduce

# Example



Right-Arc<sub>p</sub>

# Classifier-Based Parsing

- ▶ Data-driven deterministic parsing:
  - ▶ Deterministic parsing requires an **oracle**.
  - ▶ An oracle can be approximated by a **classifier**.
  - ▶ A classifier can be trained using **treebank** data.
- ▶ Learning methods:
  - ▶ Support vector machines (SVM)  
[Kudo and Matsumoto 2002, Yamada and Matsumoto 2003, Isozaki et al. 2004, Cheng et al. 2004, Nivre et al. 2006]
  - ▶ Memory-based learning (MBL)  
[Nivre et al. 2004, Nivre and Scholz 2004]
  - ▶ Maximum entropy modeling (MaxEnt)  
[Cheng et al. 2005]
  - ▶ Neural networks  
[you!]

# Feature Models

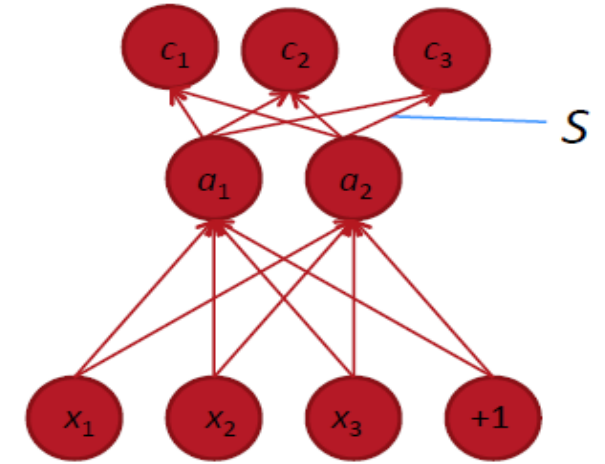
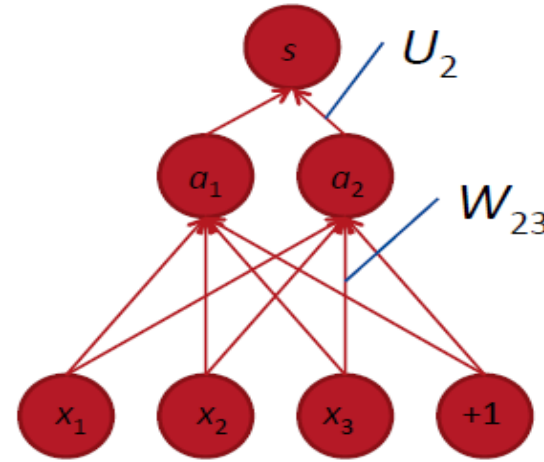
- ▶ Learning problem:
  - ▶ Approximate a function from **parser states**, represented by feature vectors to **parser actions**, given a training set of gold standard derivations.
- ▶ Typical features:
  - ▶ Tokens:
    - ▶ Target tokens
    - ▶ Linear context (neighbors in  $S$  and  $Q$ )
    - ▶ Structural context (parents, children, siblings in  $G$ )
  - ▶ Attributes:
    - ▶ Word form (and lemma)
    - ▶ Part-of-speech (and morpho-syntactic features)
    - ▶ Dependency type (if labeled)
    - ▶ Distance (between target tokens)

# Neural Networks

(Fig: courtesy R Socher)

Neural Networks can be built for different input, output types.

- Outputs can be:
  - Linear, single output (Linear)
  - Linear, multiple outputs (Linear)
  - Single output binary (Logistic)
  - Multi output binary (Logistic)
  - 1 of k ~~Multinomial~~ output (Softmax) categorical
- Inputs can be:
  - A scalar number
  - Vector of Real numbers
  - Vector of Binary



**Goal of training:** Given the training data (inputs, targets) and the architecture, determine the model parameters.

Model Parameters for a 3 layer network:

- Weight matrix from input layer to the hidden ( $W_{jk}$ )
- Weight matrix from hidden layer to the output ( $W_{kj}$ )
- Bias terms for hidden layer
- Bias terms for output layer

**Our strategy** will be:

- Compute the error at the output
- Determine the contribution of each parameter to the error by taking the differential of error wrt the parameter
- Update the parameter commensurate with the error it contributed.

# Design Choices

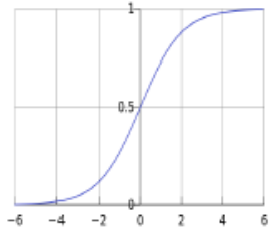
- When building a neural network, the designer would choose the following hyper parameters and non linearities based on the application characteristics:
  - Number of hidden layers
  - Number of hidden units in each layer
  - Learning rate
  - Regularization coefft
  - Number of outputs
  - Type of output (linear, logistic, softmax)
  - Choice of Non linearity at the output layer and hidden layer (See next slide)
  - Input representation and dimensionality



# Commonly used non linearities (fig: courtesy Socher)

logistic ("sigmoid")

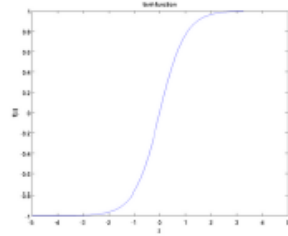
$$f(z) = \frac{1}{1 + \exp(-z)}$$



$$f'(z) = f(z)(1 - f(z))$$

tanh

$$f(z) = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

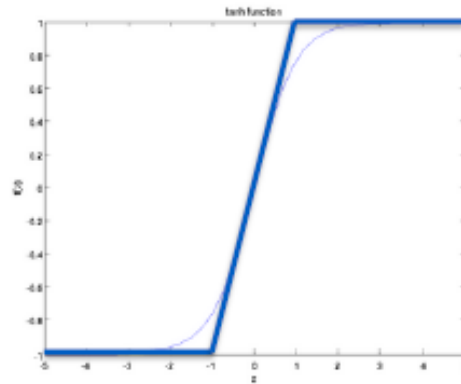


$$f'(z) = 1 - f(z)^2$$

hard tanh

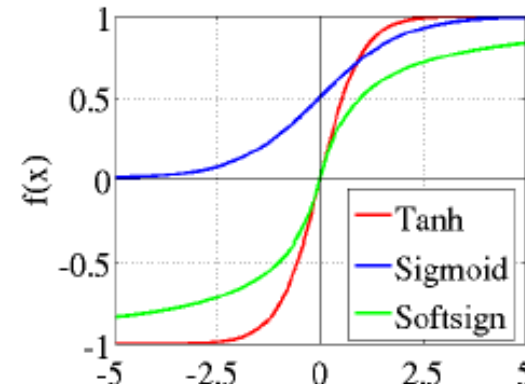
$$\tanh(z) = 2\text{logistic}(2z) - 1$$

$$\text{HardTanh}(x) = \begin{cases} -1 & \text{if } x < -1 \\ x & \text{if } -1 \leq x \leq 1 \\ 1 & \text{if } x > 1 \end{cases}$$



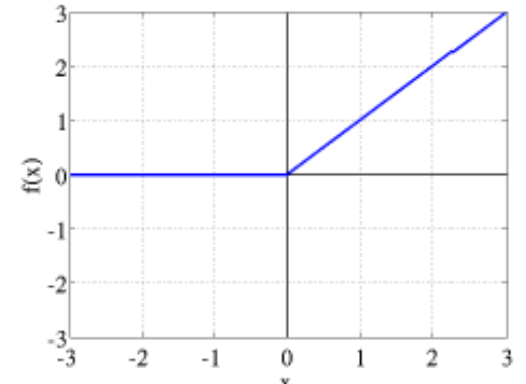
soft sign

$$\text{softsign}(z) = \frac{a}{1 + |a|}$$



**rectified linear (ReLU)**

$$\text{rect}(z) = \max(z, 0)$$



# Objective Functions and gradients

- Linear – Mean squared error

- $E(w) = \frac{1}{2N} \sum_1^N (t_n - y_n)^2$

- Logistic with binary classifications: Cross Entropy Error

- Logistic with k outputs:  $k > 2$ : Cross Entropy Error

- Softmax: 1 of K "multinomial" classification: Cross Entropy Error, minimize NLL

- In all the above cases we can show that the gradient is:  $(y_k - t_k)$  where  $y_k$  is the predicted output for the output unit k and  $t_k$  is the corresponding target

# High Level Backpropagation Algorithm

- Apply the input vector to the network and forward propagate. This will yield the activations for hidden layer(s) and the output layer
  - $net_j = \sum_i w_{ji} z_i$ ,
  - $z_j = h(net_j)$  where  $h$  is your choice of non linearity. Usually it is sigmoid or tanh. Rectified Linear Unit (ReLU) is also used.

- Evaluate the error  $\delta_k$  for all the output units

$\delta_k = o_k - t_k$  where  $o_k$  is the output produced by the model and  $t_k$  is the target provided in the training dataset

- Backpropagate the  $\delta$ 's to obtain  $\delta_j$  for each hidden unit  $j$

$$\delta_j = h'(z_j) \sum_k w_{kj} \delta_k$$

- Evaluate the required derivatives

$$\frac{\partial E}{\partial W_{ji}} = \delta_j z_i$$