CS626: Speech, NLP and the Web

Shallow Parsing with Maxmimum Entropy Markov Models and Conditional Random Fields

Pushpak Bhattacharyya

Computer Science and Engineering Department

IIT Bombay

Week of 7th September, 2020

Agenda for the week (1/2)

- Work out the mathematics of MEMM and CRF
- Apply to shallow parsing
- Elaborate discussion on **FEATURE ENGG**
- Discuss Gradient Descent with Feed Forward Network and BackPropagation
- Introduce Deep Parsing
- Make reference to Neural Parsing

Algorithmics and Mathematics of Chunking

Noisy Channel Model

W *Noisy Channel* **T**

$(w_n, w_{n-1}, \ldots, w_1)$ $(t_m, t_{m-1}, \ldots, t_1)$

Sequence *W* is transformed into sequence *T*

$$T^* = \underset{T}{\operatorname{argmax}}(P(T|W))$$

$$W^* = \underset{W}{\operatorname{argmax}}(P(W|T))$$

Sequence to Sequence Labelling: Chunk w/o chunk type

माणसाने	उडण्याचा	प्रयत्न	केला
NN	**VG**	**NN**	**VBD**
B	**B**	**B**	**I**

Chunking vs. POS Tagging

- Much simpler task than POS tagging!
- Only 2 tags in the simplest form: ‘***B***’ and ‘***I***’
- Makes use of POS and MORPH information
- Slightly more complex when the “TYPE” of chunk also is required

Chunk with chunk type

[NP He] [VP reckons] [NP the current account deficit] [VP will narrow] [PP to] [NP only # 1.8 billion] [PP in] [NP September] .

He	PRP	B-NP
reckons	VBZ	B-VP
the	DT	B-NP
current	JJ	I-NP
account	NN	I-NP
deficit	NN	I-NP
will	MD	B-VP
narrow	VB	I-VP

to	TO	B-PP
only	RB	B-NP
#	#	I-NP
1.8	CD	I-NP
billion	CD	I-NP
in	IN	B-PP
September	NNP	B-NP
.	.	

Noisy Channel

Sequence *W* is transformed into sequence *T*

$$T^* = \underset{T}{\operatorname{argmax}}(P(T|W))$$

$$W^* = \underset{W}{\operatorname{argmax}}(P(W|T))$$

Maximum Entropy Markov Model (1/2)

$$P(t_1, t_2, t_3 \ldots, t_n | w_1, w_2, w_3, \ldots, w_n) = \prod_{i=1}^{n} P(t_i | h_i)$$

h_i is called the **history** at position *i*. This captures a lot of information REQUIRED for putting the label at position i.

Digression

Principle behind MEMM

- Choose the probability distribution *p* that has the highest entropy out of those distributions that satisfy a certain set of constraints.
- The PRINCIPLE OF MAXIMIZING ENTROPY
- Competitor to MAXIMUM LIKELIHOOD

Comparing Maximum Likelihood and Maximum Entropy

- MLE maximizes probability of observations: chooses parameters accordingly
- ME maximizes entropy, satisfying given constraints: takes a stand of minimum bias

Illustration for Maximum Entropy Principle

- Take a coin with parameter *p*=probability of head
- The coin is NOT tossed, so there is no observation!
- What is the value of p?
- Intuitively 0.5: WHY?
- Uniform distribution; equal probability for head and tail; no bias
- That is, maximum entropy

Entropy in case of coin with NO toss, and deriving parameter

- $E = -p\log p - (1-p)\log(1-p)$
- E is a function of p
- Maximize entropy, $\frac{dE}{dp}=0$

$-\log p - 1 + \log(1-p) + 1 = 0$

$p = 1-p$, i.e. $p = 0.5$, QED

Case of coin with N tosses and K heads: apply MLE

- $L = p^K(1-p)^{(N-K)}$

$$\frac{dL}{dp} = 0 \text{ gives } p = \frac{K}{N}$$

- Shown before

Back to MEMM for seq2seq labeling

MEMM Idea (1/2)

- Choose the probability distribution *p* that has the highest entropy out of those distributions that satisfy a certain set of constraints

- The constraints restrict the model to behave in accordance with a set of statistics collected from the training data

MEMM Idea (2/2)

- The statistics are expressed as the expected values of appropriate functions defined on the contexts *h* and tags *t*

- In particular, the constraints demand that the expectations of the features for the model match the empirical expectations of the features over the training data

Constraint

- A reasonable assumption
- The expectations of features according to the joint distribution p are equal to the expectations of the features in the empirical (training data) distribution $\tilde{p}$

Mathematically, the constraint is expressed as

$$E_{p(t_i,hi)} f_j(t_i, hi) = E_{p\sim(ti,hi)} f_j(t_i, hi)$$

From this,

$$p(t_i|h_i) = [\prod_{j=1}^{K} e^{\lambda_j f_j(h_i,t_i)}] / [\sum_{t_i'} \prod_{j=1}^{K} e^{\lambda_j f_j(h_i,t_i')}]$$

In sum form

$$p(t_i|h_i) = \frac{e^{\sum_{j=1}^{K} \lambda_j f_j(t_i, h_i)}}{Z}$$

where, $Z = \sum_{t_i'} e^{\sum_{j=1}^{K} \lambda_j f_j(t_i', h_i)}$

The crux of the matter is estimation of λ_js

Use Gradient Descent

- $log(p(t_i|h_i) = \sum_{j=1}^{K} \lambda_j f_j(t_i, hi)$ - $logZ$

- $Z = \sum_{t_i'} e^{\sum_{j=1}^{K} \lambda_j f_j(t_i', hi)}$

Iteration

- $X_n = log(p(t_i|h_i) = \sum_{j=1}^{K} \lambda_j(n) f_j(t_i, hi)$ - $logZ_n$

- Where, *n* refers to the iteration no.

- X_n is the value of $log(p(t_i|h_i)$ at the n^{th} iteration, which is a function of $\lambda_j(n)$ only

Gradient Descent based training

- Goal is to find λ_j
- Closed form expression not possible
- Find iteratively

$$\lambda_j(n+1) = \lambda_j(n) - \eta \frac{\delta X(n)}{\delta \lambda_j}$$

- Where, derivative of X at n^{th} iteration with respect to λ_j is taken
- Start with some initial value of λ_j

How will the gradient descent run?

[NP He] [VP reckons] [NP the current account deficit] [VP will narrow] [PP to] [NP only # 1.8 billion] [PP in] [NP September] .

He	PRP	B-NP
reckons	VBZ	B-VP
the	DT	B-NP
current	JJ	I-NP
account	NN	I-NP
deficit	NN	I-NP
will	MD	B-VP
narrow	VB	I-VP

to	TO	B-PP
only	RB	B-NP
#	#	I-NP
1.8	CD	I-NP
billion	CD	I-NP
in	IN	B-PP
September	NNP	B-NP
.	.	

How does the training go?

- Have training corpus
- Tagged with B-I labels
- POS tags
- Features extracted
- Then compute iteratively

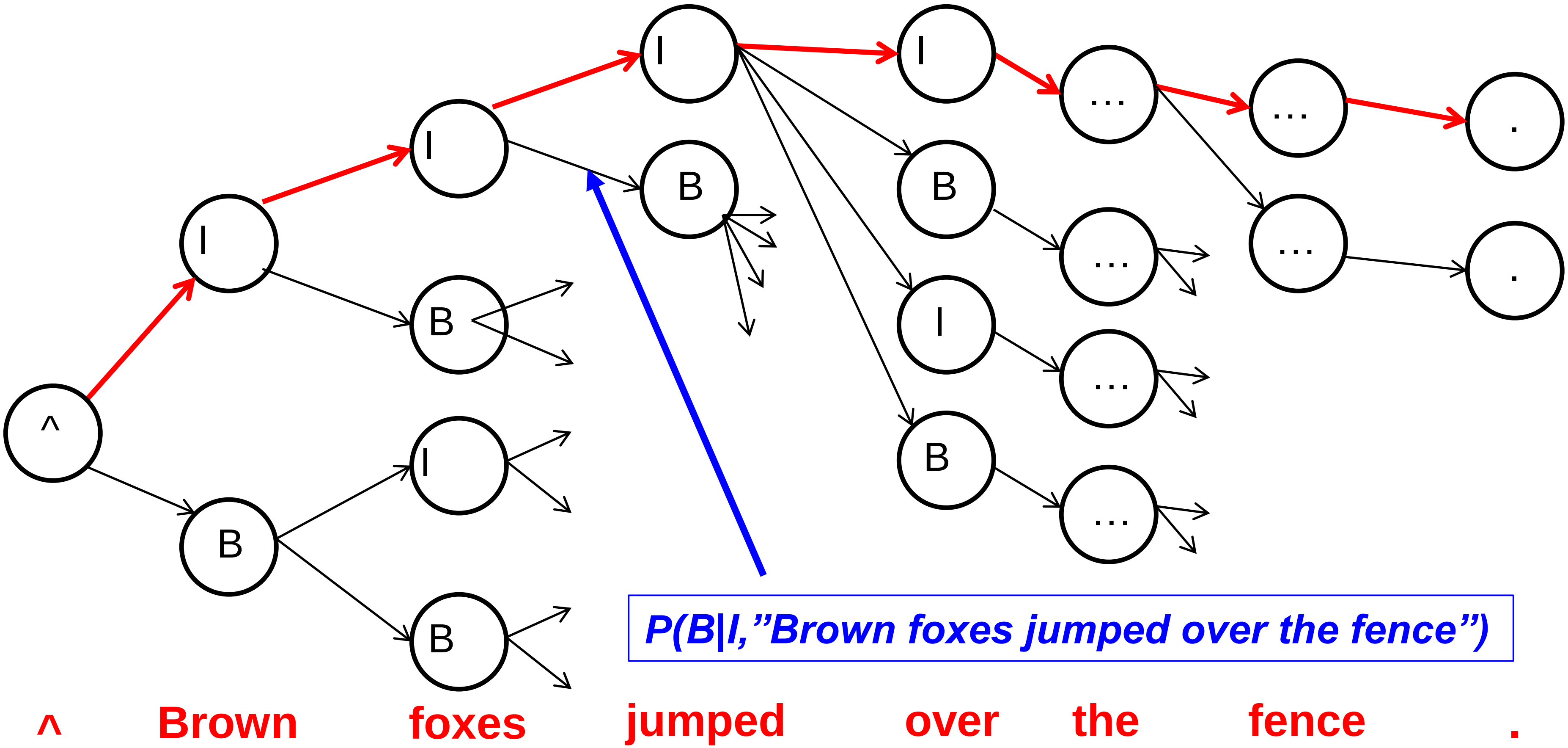
MEMM Decoding

- If we have $f_a(s,o)$ and λ_a values, we can run Viterbi decoding (or beam search) to do the labelling
- The moot question now:

  (a) how to design $f_a(s,o)$, the feature set, and

  (b) and assign the weights λ_a

Illustration with example



Probability of a path (e.g. Top most path) =
Product of $P(Y_i|Y_{i-1}, X)$

Design the feature set: a proposal (out of many) (1/5)

Word based (window size 5)

- f_1= current word (‘foxes’)
- f_2= previous word (‘brown’)
- f_3= prev to prev word (‘^’)
- f_4= following word (‘jumped’)
- f_5= following to following word (‘over’)

^ brown foxes jumped over the fence .

Feature Set Design (2/3)

POS based (window size 5)

- f_6= POS of current word (NNS)
- f_7= POS of previous word (JJ)
- f_8= POS of prev to prev word (^)
- f_9= POS of following word (VBD)
- f_{10}= POS of following to following word (IN)

^ brown foxes jumped over the fence .

Feature Set Design (3/3)

CHUNK based (window size 5)

- f_{11}= B/I of previous word (B)
- f_{12}= B/I of prev to prev word (B)

^ brown foxes jumped over the fence .

Feature Set Design

MORPH based (window size 5)

- f_{12}= does the current word have a particular noun suffix, like ‘s’, ‘es’, ‘ies’, etc. (yes: ‘es’): f_{12} *itself is a feature vector*!
- f_{13}= particular verbal suffix, like ‘d’, ‘ed’, ‘t’, etc. (no): f_{13} *itself is a feature vector*!
- f_{14}= adjective suffix, like ‘ly’, ‘ment’, ‘tion’, etc. (no): f_{13} *itself is a feature vector*!
- f_{14}= adverb suffix, like ‘ly’, ‘ment’, ‘tion’, etc. (no): f_{13} *itself is a feature vector*!

^ *brown foxes jumped over the fence .*

Noun Suffixes https://examples.yourdictionary.com/list-of-suffixes-and-suffix-examples.html

- **-eer**
  Meaning: engaged in something, associated with something
  Examples: auctioneer, volunteer, engineer, profiteer
- **-er**
  Meaning: someone who performs an action
  Examples: helper, teacher, preacher, dancer
- **-ion**
  Meaning: the action or process of
  Examples: celebration, opinion, decision, revision
- **-ity**
  Meaning: the state or condition of
  Examples: probability, equality, abnormality, civility
- **-ment**
  Meaning: the action or result of
  Examples: movement, retirement, abandonment, establishment
- **-ness**
  Meaning: a state or quality
  Examples: fondness, awareness, kindness, darkness
- **-or**
  Meaning: a person who is something
  Examples: distributor, investigator, translator, conductor
- **-sion**
  Meaning: state or being
  Examples: depression, confusion, tension, compulsion
- **-ship**
  Meaning: position held
  Examples: worship, ownership, courtship, internship
- **-th**
  Meaning: state or quality
  Examples: strength, labyrinth, depth, warmth

Adjective Suffixes

- **-able, -ible:**
  Meaning: capable of being
  Examples: preventable, adaptable, predictable, credible
- **-al**
  Meaning: pertaining to
  Examples: theatrical, natural, criminal, seasonal
- **-ant**
  Meaning: inclined to or tending to
  Examples: vigilant, defiant, brilliant, reliant
- **-ary**
  Meaning: of or relating to
  Examples: budgetary, planetary, military, honorary
- **-ful:** Meaning: full of or notable of
  Examples: grateful, beautiful,
- **-ic**
  Meaning: relating to
  Examples: iconic, organic, heroic, poetic
- **-ious, -ous**
  Meaning: having qualities of
  Examples: gracious, cautious, humorous, fabulous
- **-ive**
  Meaning: quality or nature of
  Examples: creative, expensive, expressive, pensive
- **-less**
  Meaning: without something
  Examples: hopeless, faultless, fearless, restless
- **-y:** Meaning: made up of or characterized by

Verb Suffixes

- **-ed**
  Meaning: past-tense version of a verb
  Examples: laughed, climbed, called, missed
- **-en**
  Meaning: become
  Examples: soften, fasten, lengthen, strengthen
- **-er**
  Meaning: action or process, making an adjective comparative
  Examples: faster, bigger, fuller, longer
- **-ing**
  Meaning: verb form/present participle of an action
  Examples: laughing, swimming, driving, writing
- **-ize, -ise**
  Meaning: to cause or to become
  Examples: memorialize, authorize, commercialize, advertise

Adverb suffixes

- **-ly**
  Meaning: in what manner something is being done
  Examples: bravely, simply, honestly, gladly
- **-ward**
  Meaning: in a certain direction
  Examples: backward, wayward, awkward, afterward
- **-wise**
  Meaning: in relation to
  Examples: clockwise, edgewise, lengthwise, otherwise

Similarly for prefixes

- **PREFIX MEANING EXAMPLES**
- a-, an- without, not anesthetic, atheist
- ab- away, from abject, abscess
- ad-, a-, ac-, as- to, toward access, admit, assist
- ante before antecedent, anterior
- anti- against antibiotics, antioxidant
- auto- self autoimmune, autonomous
- ben- good benefit, benign
- bi- two, both bifocals, bipolar
- circum- around circumference, circumscribe
- co-, com-, con- with, together companion, concurrent*
- contra-, counter- against contradict, counteract
- de- down, undo, not degenerate, depress
- di-, dis- lack of, not, apart disadvantage, displacement

Remarks for morphology features

- Needs morphology analysis
- Shallow MA: affix separation
- Deep MA: features (like GNPTAM for verbs)
- Statistical stemmers go by frequency of substrings
- E.g., BPE, Morfessor, Porter etc.
- For a given word, **mostly the features will be 0!**

Feature engineering: word form based

- <u>Word property based (syntactic, window size 5)</u>
- Capitalization? (‘no’)
- Length (5)
- #Orthographic syllables (3: fo, xe, s)
- #BPEs
- #syllables (2: fox, es)
- ***^ brown foxes jumped over the fence .***

Feature engineering: word meaning based

- <u>Word property based (semantic, window size 5)</u>
- Place/Organization/Person (‘no’)
- Animate (‘yes’)
- Carnivorous (‘yes’)
- ***^ brown foxes jumped over the fence .***
- Needs Knowledge Graph
- Not really needed at the Chunk Level

NLP inherently has cyclicity

- Semantic features need semantic analysis
- Semantic analysis needs lower level analysis: morph, pos, chunk, parse
- E.g., “Bay of Bengal”
- Should chunk the whole thing
- How to know that all these 3 words form a single unit?
- We need probability: MLE/Bayesian/Max Ent

Feature Vector (considering our situation) (1/4)

- Compartment-1: current word vector (can be word embedding) *(size-300)*
- Comp-2: prev word vector *(size-300)*
- Comp-3: prev to prev word vector *(size-300)*
- Comp-4: following word's vector (300)
- Comp-5: following to following word's vector (300)

Compartment-1	Compartment-1	------------	Compartment-n

Feature Vector (considering our situation) (2/4)

- Comp-6: current word's POS vector *(size-12;* there are 12 universal pos tags*)*
- Comp-7: prev word's POS vector *(size-12)*
- Comp-8: prev to prev word's POS vector (12)
- Comp-9: following POS vector (12)
- Comp-10: following to following POS vector (12)

Compartment-1	Compartment-1	------------	Compartment-n

Feature Vector (considering our situation) (3/4)

- Comp-11: prev word's chunk vector *(size-1;* two chunk labels, so, 1/0*)*
- Comp-12: prev to prev word's chunk *(size-1)*

Compartment-1	Compartment-1	------------	Compartment-n

Feature Vector (considering our situation) (3/4)

- Comp-13: current word's suffix vector *(size-100;* assuming 100 suffixes possible*)*
- Comp-14: current word's prefix vector *(size-100;* assuming 100 prefixes possible*)*
- Comp-15: current word's OTHER properties vector, capitalization,length, #syllables, animacy, etc. (*size-50;* assuming 50 such other properties)

Compartment-1	Compartment-1	------------	Compartment-n

Total size of feature vector

- Word based: 300 dimensions X 5 words window (current+2 prev+2 foll)
- POS based: 12 POSes X 5
- Chunk based: 1 X 2 prev words
- Suffix: 100
- Prefix: 100
- OTHER: 50
- Most of the components will be 0!
- Very sparse feature vector!

TOTAL feature vector size

= 1500+60+2+350=1912

How to weight the features

- MEMM: General Iterative Scaling
- Gradient Descent ** (will do)
- Limited Memory Broyden–Fletcher–Goldfarb–Shanno algorithm (L-BFGS)

Conditional Random Field

CRF Formulation

$$\hat{\boldsymbol{y}} = \arg\max_{\boldsymbol{y}} p_{\boldsymbol{\lambda}}(\boldsymbol{y}|\boldsymbol{x}) = \arg\max_{\boldsymbol{y}} \boldsymbol{\lambda} \cdot \boldsymbol{F}(\boldsymbol{y}, \boldsymbol{x})$$

$$p_{\boldsymbol{\lambda}}(\boldsymbol{Y}|\boldsymbol{X}) = \frac{\exp \boldsymbol{\lambda} \cdot \boldsymbol{F}(\boldsymbol{Y}, \boldsymbol{X})}{Z_{\boldsymbol{\lambda}}(\boldsymbol{X})} \qquad (1)$$

where

$$Z_{\boldsymbol{\lambda}}(\boldsymbol{x}) = \sum_{\boldsymbol{y}} \exp \boldsymbol{\lambda} \cdot \boldsymbol{F}(\boldsymbol{y}, \boldsymbol{x})$$

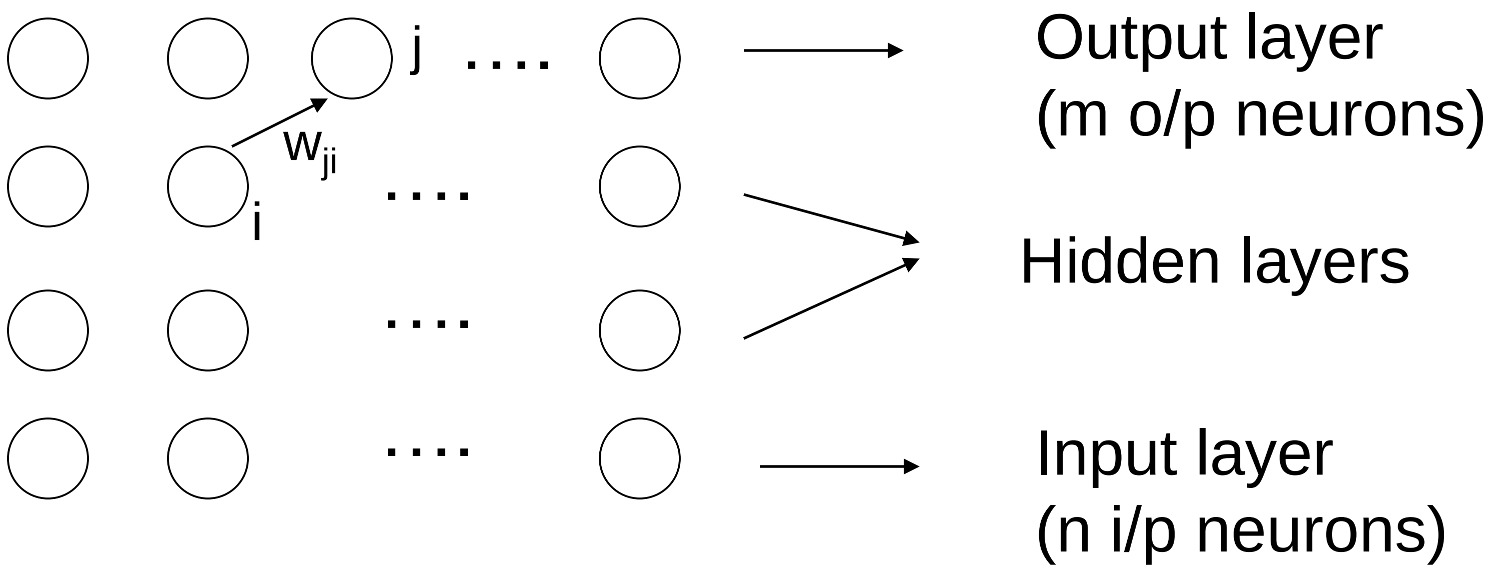
$$\boldsymbol{F}(\boldsymbol{y}, \boldsymbol{x}) = \sum_{i} \boldsymbol{f}(\boldsymbol{y}, \boldsymbol{x}, i)$$

i ranges over the input positions

Gradient Descent

Explaining through Feed Forward Neural Network and Backpropagation

Backpropagation algorithm



- Fully connected feed forward network
- Pure FF network (no jumping of connections over layers)

Gradient Descent Equations

$$\Delta w_{ji} = -\eta \frac{\delta E}{\delta w_{ji}} (\eta = \text{learning rate}, 0 \le \eta \le 1)$$

$$\frac{\delta E}{\delta w_{ji}} = \frac{\delta E}{\delta net_j} \times \frac{\delta net_j}{\delta w_{ji}} (net_j = \text{input at the j}^{th} \text{ layer})$$

$$\frac{\delta E}{\delta net_j} = -\delta j$$

$$\Delta w_{ji} = \eta \delta j \frac{\delta net_j}{\delta w_{ji}} = \eta \delta j o_i$$

Backpropagation – for outermost layer

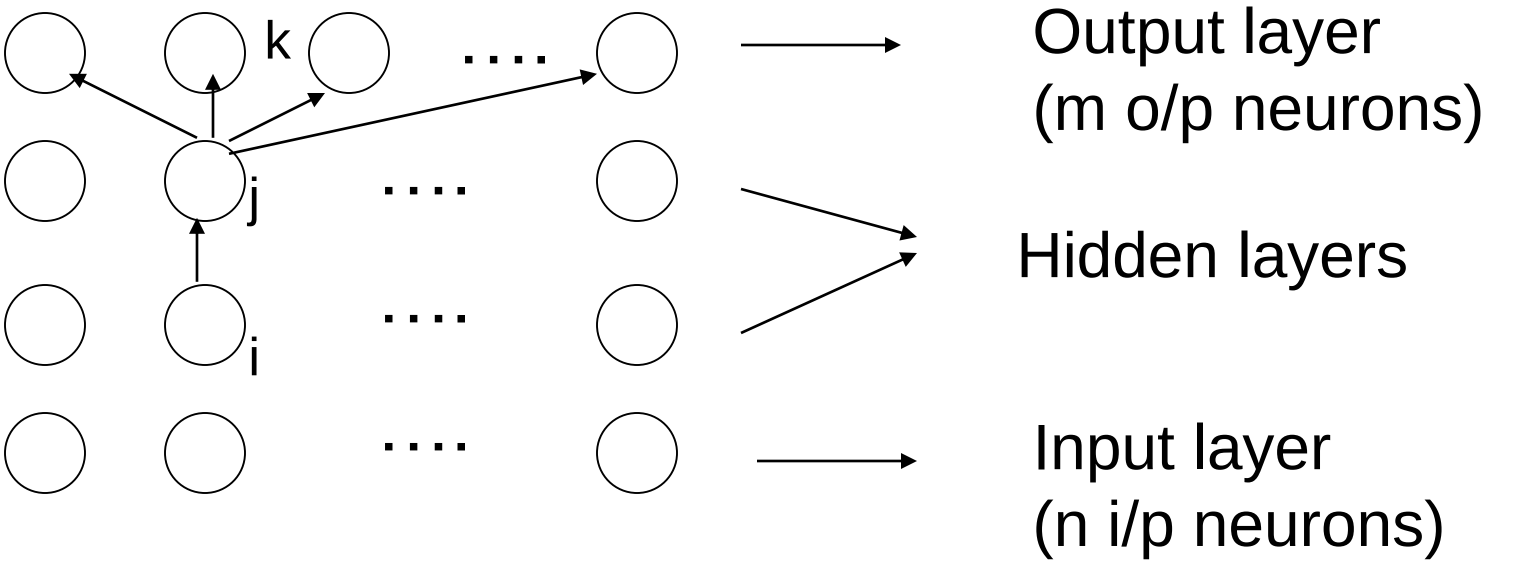
$$\delta j = -\frac{\delta E}{\delta net_j} = -\frac{\delta E}{\delta o_j} \times \frac{\delta o_j}{\delta net_j} \; (net_j = \text{input at the j}^{th} \text{ layer})$$

$$E = \frac{1}{2}\sum_{p=1}^{m}(t_p - o_p)^2$$

$$\text{Hence, } \delta j = -(-(t_j - o_j)o_j(1 - o_j))$$

$$\Delta w_{ji} = \eta(t_j - o_j)o_j(1 - o_j)o_i$$

Backpropagation for hidden layers



δ_k is propagated backwards to find value of δ_j

Backpropagation – for hidden layers

$$\Delta w_{ji} = \eta \delta j o_i$$

$$\delta j = -\frac{\delta E}{\delta net_j} = -\frac{\delta E}{\delta o_j} \times \frac{\delta o_j}{\delta net_j}$$

$$= -\frac{\delta E}{\delta o_j} \times o_j (1 - o_j)$$

$$= -\sum_{k \in \text{next layer}} \left(\frac{\delta E}{\delta net_k} \times \frac{\delta net_k}{\delta o_j}\right) \times o_j (1 - o_j)$$

$$\text{Hence, } \delta_j = -\sum_{k \in \text{next layer}} (-\delta_k \times w_{kj}) \times o_j (1 - o_j)$$

$$= \sum_{k \in \text{next layer}} (w_{kj} \delta_k) o_j (1 - o_j) o_i$$

General Backpropagation Rule

- General weight updating rule:

$$\Delta w_{ji} = \eta \delta j o_i$$

- Where

$$\delta_j = (t_j - o_j) o_j (1 - o_j) \quad \text{for outermost layer}$$

$$= \sum_{k \in \text{next layer}} (w_{kj} \delta_k) o_j (1 - o_j) o_i \quad \text{for hidden layers}$$

How does it work?

- Input propagation forward and error propagation backward (e.g. XOR)