Syllabification and Accent using Dynamic Computational Networks

Brandon Rhodes

June 11, 2019

Question: How can we account for syllabification, accent and their interaction (quantity sensitivity) in one integrated system?

Deep(er) question: How much more is less?

First steps to answering this question

1. Start with a system — Dynamic Computational Networks (DCN)
2. Define and investigate Dynamic Computational Networks (DCN) in initial form — Goldsmith and Larson 1990s
3. Make reasonable modifications to DCN for quantity-sensitivity and investigate new properties
4. Evaluate model and its difficulties
5. Modify current model and explore new ones

Overview

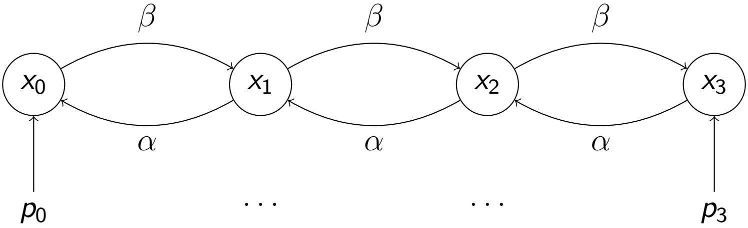
1. Introduction
2. DCNs and learning
3. Extending DCNs for quantity-sensitivity
4. Current work
5. Conclusions and future directions

Basics of Dynamic Computational Networks (DCN)

Dynamic Computational Networks — Goldsmith and Larson 1990s

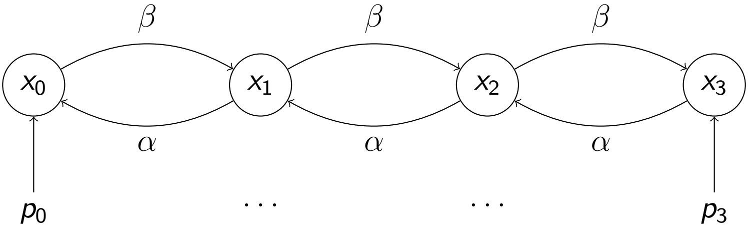
- Dynamic system to account for syllabification and stress
- Neural network with local excitatory and inhibitory connections between nodes in network
- In quantity-insensitive situation, network consists of
  (i) Single layer of nodes
  (ii) Trainable (learnable from data) parameters that govern relationship between nodes

Diagram of basic architecture



- Nodes: circles
- Parameters: $\alpha, \beta, p_i, (x_i)$

Interpretation of basic architecture: syllabification



- Nodes: phonological segment
- Parameters:
  - x_i inherent sonority
  - α effect on left neighbor's sonority
  - β effect on right neighbor's sonority
  - p_i positional bias for node i

Prediction / determination of syllable structure

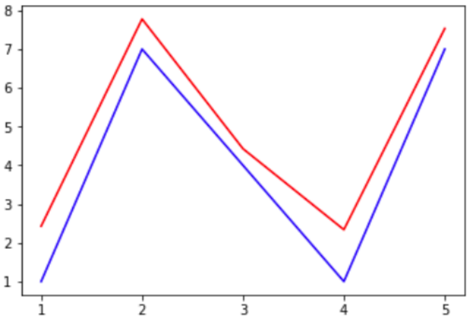
1. Input at each node i an inherent sonority x_i, make activation value v_i^0 of node $i = x_i$
2. Compute a new activation value for node i at step t as

$$v_i^t = x_i + p_i + \alpha \cdot v_{i+1}^{t-1} + \beta \cdot v_{i-1}^{t-1}$$

3. Once a v_i^t changes too little from previous v_i^{t-1}, stop computation; call v_i^t derived sonority
4. Label local peaks of derived sonority syllable nuclei; local troughs as syllable onsets; codas as those to right of nucleus and left of onset; remaining are onset

```
Phon string:
B,AA1,NG,G,OW2

Inherent sonority (blue):
[1. 7. 4. 1. 7.]

Derived sonority (red):
[2.42927291 7.77661163 4.42792293 2.33806806 7.53227872]

(alpha, beta):
(0.184,0.229)
```

```
Out[91]: array([1., 7., 4., 1., 7.])
```



Learning parameters α, β via simulated annealing – intuition

- Setup: Have some temperature τ, a measure for system's accuracy
- Goal: lower τ — 'freeze' the system
- How?
  1. Give the network a lexical item to syllabify
  2. If system prediction is correct, decrease τ; if wrong, change α, β by some small random value and increase τ
  3. Once system 'freezes' ($\tau < T$, for some T), stop

Extending DCNs for quantity-sensitivity

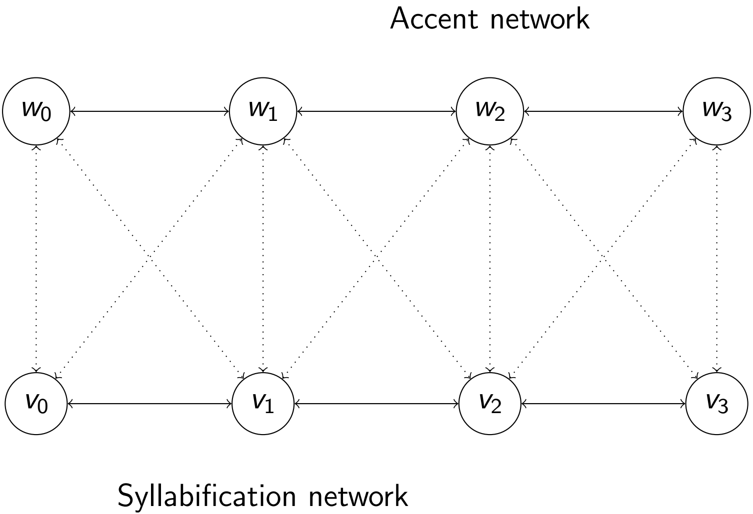
Quantity-sensitivity from DCN perspective

- Have:
  - a single layer which computes derived sonorities to predict syllabification
  - a single layer which computes derived accent to predict accent contour
- Want: a two-layer system which predicts both syllabification and accent contour
- Questions:
  - (i) Do we want to jointly (simultaneously) predict syllabification and accent?
  - (ii) How do we connect the two layers?
  - (iii) How does information flow between the two layers?

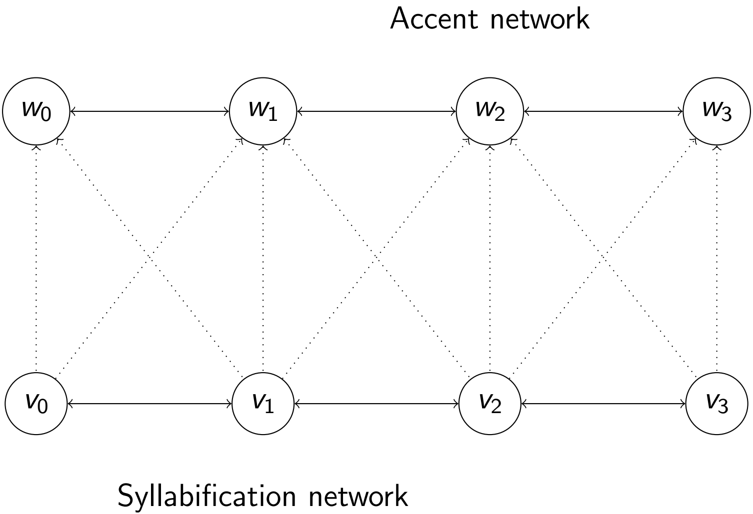
Salient theoretical options

1. Layers are independent and have bidirectional connections
2. Layers are independent and have unidirectional connections
3. Syllabification layer determines structure of accent layer, unidirectional connections

Independent layers, bidirectional connections



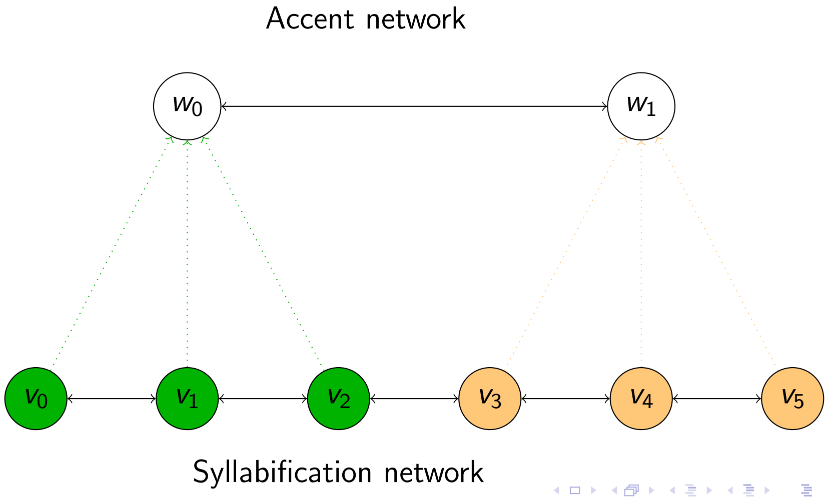
Independent layers, unidirectional connections



Potential theoretical pros / cons

- Potential pros:
  - Most general, and therefore flexible
  - Maintains (most) similarity with traditional DCN
- Potential cons:
  - Interpretation not straightforward
  - Does not (explicitly) claim that stress is a property of syllables
  - Undesirable consequences? Example: coda could have most prominent derived accent

Accent layer dependent on syllabification, unidirectional connections



Potential theoretical pros / cons

- Potential pros:
  - Intuitive interpretation of accent nodes — syllables
  - Explicitly states that stress is property of syllables
  - Intuitive interpretation of input to accent nodes — syllable weight
- Potential cons:
  - Many ways to implement this dependence, so what is best?
  - Slightly more involved implementation-wise, so may not scale as nicely with data

Current work and evaluation methods

Data and processing

- Used data from Carnegie Mellon University (CMU) pronouncing dictionary for English
  - Phonemic transcriptions with ARPAbet
  - Syllable boundaries provided by Bartlett et al. (2009) — structured SVM-HMM
- Used binary labelling scheme described in Larson (1993) $\approx$ onset / rime
  - Example: *happy*
    [hæp.i] $\rightsquigarrow$ ONCN, UDDD
    [hæ.pi] $\rightsquigarrow$ ONON, UDUD

Data and processing

- For inherent sonority, used basic hierarchy

$$\text{vowels} = 7 > \text{semi-vowels} = 6 > \text{liquids} = 5 > \text{nasals} = 4 > \text{fricatives} = 3 > \text{affricates} = 2 > \text{stops} = 1$$

- Assumed zero positional activation (and zero bias)
- For two-layer network, assumed zero inherent accent

Example data entry

raw data: WARRIORS(2), W AO1 R - Y ER0 Z

syllable structure: ONCONC

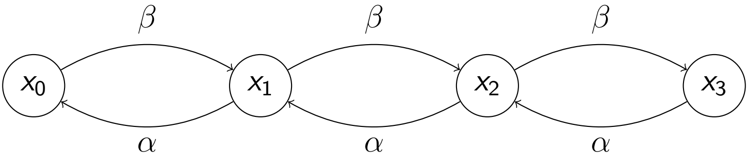
syllable label: UDDUDD $= (0,1,1,0,1,1)$

accent label: $(0,1,0,0,0,0)$

inherent sonority: $(6,7,5,6,7,3)$

inherent accent : $(0,0,0,0,0,0)$

Syllabification, single layer — performance and parameter estimates



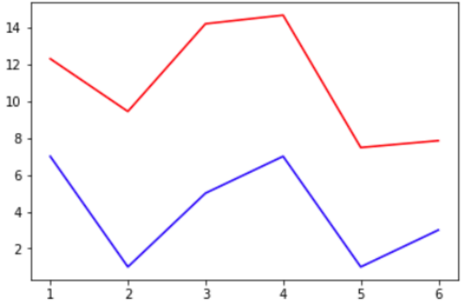
- 5-fold cross-validation, 3000 randomly selected examples
  - α range $\approx [0.11, 0.23]$
  - β range $\approx [0.11, 0.24]$
  - Average test error 30%

Syllabification, single layer — common mistakes

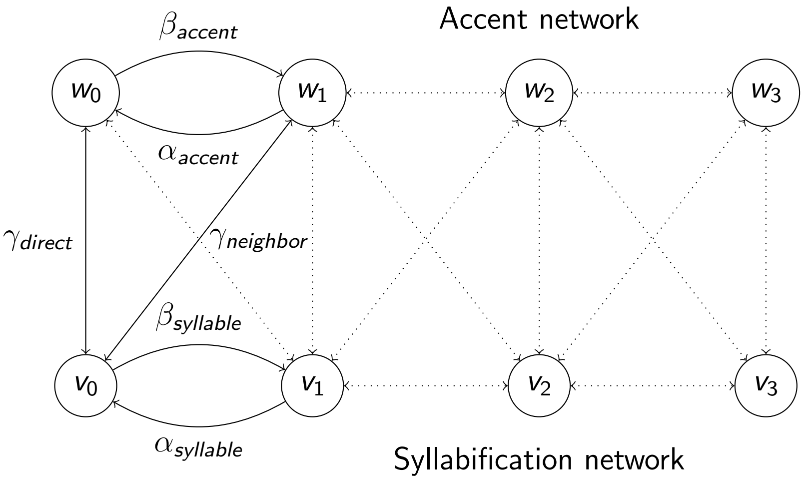
- Sonority sequencing
  (i) Complex onset and codas (fricative-stop, stop-fricative)
  (ii) Word internal syllable boundaries (sonority)
- See example for *outlooks*

```
Phon string:
AW1,T,L,UH2,K,S

Syllabification:
NCONCC

Actual syllable:
[1. 1. 0. 1. 1. 1.]

Predicted syllable:
[1. 0. 0. 1. 0. 1.]

(alpha, beta):
(0.103,0.222)
```

```
Out[148]: array([7., 1., 5., 7., 1., 3.])
```



Syllabification and stress jointly



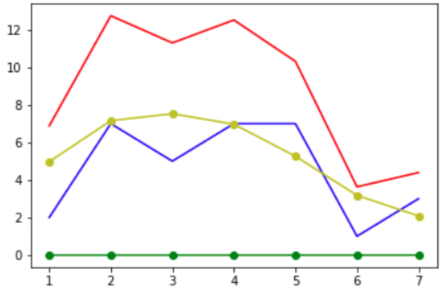
Syllabification and stress, two layer — performance and parameter estimates

- Trained new γ parameters via simulated annealing
- 5-fold cross-validation, 3000 randomly selected examples
  - $\alpha_{syllable}$ range [0.07, 0.23]
  - $\beta_{syllable}$ range [-0.04, 0.19]
  - α_{accent} range [0.03, 0.24]
  - β_{accent} range [0.02, 0.23]
  - γ_{direct} range [0.14, 0.24]
  - $\gamma_{neighbor}$ range [0.07, 0.25]
- Average test error 80%

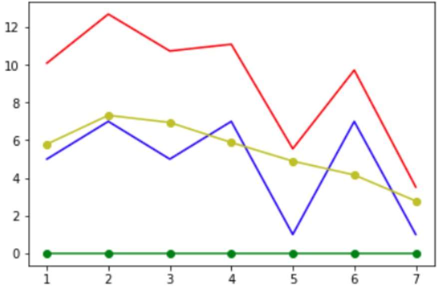
Syllabification and stress, two layer — common mistakes

- Stress tends to accumulate in the middle
  - (i) Off-by-one
  - (ii) Missing multiple stress
- Sonority sequencing (as before)

```
Word:
CHARIOTS

Phon string:
CH,EH1,R,IY0,AH0,T,S

Actual accent maxima:
[0. 1. 0. 0. 0. 0. 0.]

Predicted accent maxima:
[0. 0. 1. 0. 0. 0. 0.]

(gamma_direct, gamma_neighbors):
(0.246,0.121)

Out[167]: array([2., 7., 5., 7., 7., 1., 3.])
```



```
Word:
RELEGATE

Phon string:
R,EH1,L,AH0,G,EY2,T

Actual accent maxima:
[0. 1. 0. 0. 0. 1. 0.]

Predicted accent maxima:
[0. 1. 0. 0. 0. 0. 0.]

(gamma_direct, gamma_neighbors):
(0.246,0.121)

Out[169]: array([5., 7., 5., 7., 1., 7., 1.])
```



Conclusions and future directions

Conclusions after initial work

- DCNs in most basic form do reasonably well in English syllabification, but must make adjustments
    - Predictions are slightly better when adding negative positional activation on edges
    - Predictions are more dependent upon inherent sonorities than initially expected
- DCNs do not do well when modeling English syllabification and stress jointly; however, many mistakes are not egregious and harsh evaluation

Future directions in DCNs

- Learn values for parameters for positional activation p_i, phonological segment bias b_i and even inherent sonority x_i
  - Assuming α, β known, the dynamic equations are linear in p_i, b_i and x_i
  - Response is $0, 1$ for both syllabification ($\approx$ onset v. rime) and accent (stress v. not stressed), so could do logistic regression
- Try out different neural architectures
- Try out different loss function for learning

Future directions outside of DCNs

- Assign a probability distribution to the data (Hidden Markov Model, Markov Random Field)
    - Would allow us to learn parameters in a more targeted way
- Work towards a continuous, wave-based theory

Future methodological tasks and considerations

- Craft good datasets — accurate and large (enough)
  - Neural networks (DCNs) need a lot of data (could easily be on scale of 50k), other models not so much
  - Need languages where phenomena are clear; that way we can establish gold standard
- With accent, eventually extend the scope of the question beyond the level of the word

Thank you for your time

Simulated annealing

Learning parameters α, β via simulated annealing – implementation

For every lexical item w in training data:

1. Present network with sequence of phonological segments w to syllabify and the correct label
2. Check if predicted syllabification is correct
3. - If correct, $\tau_{new} = \Delta\tau_{old}$.
   - Else:

   $$\begin{aligned} \alpha_{new} &= \alpha_{old} + \varepsilon \\ \beta_{new} &= \beta_{old} + \varepsilon' \\ \tau_{new} &= \tau_{old} + \sqrt{(\alpha_{old} - \alpha_{new})^2 + (\beta_{old} - \beta_{new})^2} \end{aligned}$$

   where $\varepsilon, \varepsilon' \sim N(0, \tau_{old}^2) \times c$
4. If $\tau_{new} < T$, stop; else, go back to step 1.

K-fold cross-validation

K-fold cross-validation — Intuition

1. Split your data set into K chunks
2. Remove one chunk
3. Train using all of remaining chunks
4. Test on the removed chunk, store parameter estimates and error
5. Repeat steps 1–4, except choose different chunk
6. Average test errors to estimate generalization error

K-fold cross-validation — Procedure

K-fold cross-validation

(i) Partition data set X into K parts (called ‘folds’);
$X = X_1 \cup \ldots \cup X_K$

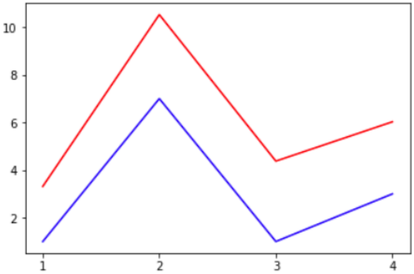
(ii) For k in $1, \ldots, K$:

1. Train model on data $X_{-k} = X - X_k$ to get estimates $\hat{\alpha}_k, \hat{\beta}_k$ at fold k for α and β
2. Test model using estimates $\hat{\alpha}_k, \hat{\beta}_k$ on data X_k
3. Compute the error, call it $error_k$, on this training set
4. Store $error_k$ and parameter estimates $\hat{\alpha}_k, \hat{\beta}_k$

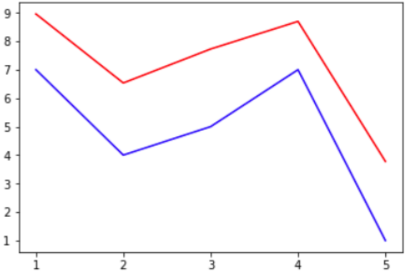
(iii) Compute the average error: $\overline{error} = \frac{1}{K} \sum_{k=1}^{K} error_k$

Syllabification, single layer — example mistakes

```
Phon string:
B,AY1,T,S

Syllabification:
ONCC

Actual syllable:
[0. 1. 1. 1.]

Predicted syllable:
[0. 1. 0. 1.]

(alpha, beta):
(0.184,0.229)

Out[128]: array([1., 7., 1., 3.])
```

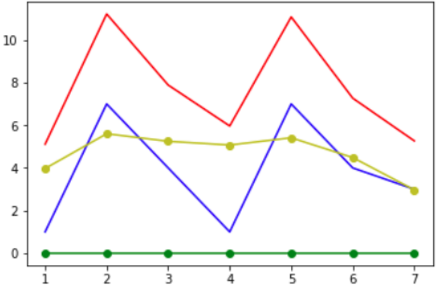


```
Phon string:
AH0,N,R,AE1,P

Syllabification:
NCONC

Actual syllable:
[1. 1. 0. 1. 1.]

Predicted syllable:
[1. 0. 0. 1. 1.]

(alpha, beta):
(0.077,-0.094)

Out[130]: array([7., 4., 5., 7., 1.])
```

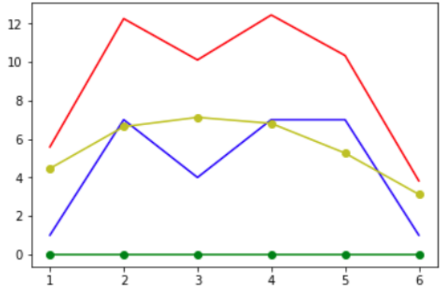


Syllabification and stress, two layer — example mistakes

```
Word:
CONDENSE

Phon string:
K,AH0,N,D,EH1,N,S

Actual accent maxima:
[0. 0. 0. 0. 1. 0. 0.]

Predicted accent maxima:
[0. 1. 0. 0. 1. 0. 0.]

(gamma_direct, gamma_neighbors):
(0.246,0.121)

Out[171]: array([1., 7., 4., 1., 7., 4., 3.])
```



```
Word:
PERMEATE

Phon string:
P,ER1,M,IY0,EY2,T

Actual accent maxima:
[0. 1. 0. 0. 1. 0.]

Predicted accent maxima:
[0. 0. 1. 0. 0. 0.]

(gamma_direct, gamma_neighbors):
(0.246,0.121)

Out[170]: array([1., 7., 4., 7., 7., 1.])
```



Exploiting linearity to learn x_i, p_i and b_i

Expressing computations with matrices and vectors

$$\mathbf{v}^t = W\mathbf{v}^{t-1} + \mathbf{x} + \mathbf{p} + \mathbf{b}$$

$$W = \begin{pmatrix} 0 & \alpha & 0 & \dots & \dots & 0 \\ \beta & 0 & \alpha & 0 & \dots & \vdots \\ \vdots & \beta & \ddots & & & \vdots \\ \vdots & & & & & \alpha \\ 0 & \dots & \dots & \dots & \beta & 0 \end{pmatrix}$$

$$\mathbf{v}^0 = \mathbf{0}$$

Expressing computations with matrices and vectors

$$\mathbf{v}^t = W\mathbf{v}^{t-1} + \mathbf{x} + \mathbf{p} + \mathbf{b}$$

$$\mathbf{v}^t = W(W\mathbf{v}^{t-2} + \mathbf{x} + \mathbf{p} + \mathbf{b}) + \mathbf{x} + \mathbf{p} + \mathbf{b}$$

$$\mathbf{v}^t = W(W(W\mathbf{v}^{t-3} + \mathbf{x} + \mathbf{p} + \mathbf{b}) + \mathbf{x} + \mathbf{p} + \mathbf{b}) + \mathbf{x} + \mathbf{p} + \mathbf{b}$$

$$\vdots$$

$$\mathbf{v}^t = \mathbf{x} + \mathbf{p} + \mathbf{b} + \textstyle\sum_{k=0}^{t-1} W^k(\mathbf{x} + \mathbf{p} + \mathbf{b})$$

$$\mathbf{v}^t = (I + \textstyle\sum_{k=1}^{t-1} W^k)(\mathbf{x} + \mathbf{p} + \mathbf{b})$$

Basic idea

- So, $\mathbf{v}^t = (I + \sum_{k=1}^{t-1} W^k)(\mathbf{x} + \mathbf{p} + \mathbf{b})$

  is a linear system in $\mathbf{x}, \mathbf{p}$ and $\mathbf{b}$!
- The matrix $(I + \sum_{k=1}^{t-1} W^k)$ gives us coefficients (products of α, β) for $\mathbf{x}, \mathbf{p}$ and $\mathbf{b}$
- Treat coefficients as vector of 'data' and $\mathbf{x}, \mathbf{p}$ and $\mathbf{b}$ as unknowns / parameters to be estimated
- Make a vector of labels
  $\mathbf{Y} = (\mathbf{Y_{11}}, \ldots, \mathbf{Y_{1n_1}}, \ldots, \ldots, \mathbf{Y_{w1}}, \ldots, \mathbf{Y_{wn_w}})$ where Y_{ij} = label for node j in word i

Logistic regression

- Treating each individual phonological element independently (obviously not true in reality)

$$\text{Let } \mu = \mathbb{E}[\mathbf{Y}]$$

$$\log \frac{\mu}{1-\mu} = \begin{pmatrix} (I + \sum_{k=1}^{t-1} W^k)(\mathbf{x}_1 + \mathbf{p} + \mathbf{b}) \\ (I + \sum_{k=1}^{t-1} W^k)(\mathbf{x}_2 + \mathbf{p} + \mathbf{b}) \\ \vdots \\ \vdots \\ (I + \sum_{k=1}^{t-1} W^k)(\mathbf{x}_w + \mathbf{p} + \mathbf{b}) \end{pmatrix}$$

Logistic regression — observation level

$$\text{Let } \mu_{ij} = \mathbb{E}[\mathbf{Y}_{ij}]$$

$$\log \frac{\mu_{ij}}{1-\mu_{ij}} = \left[(I + \textstyle\sum_{k=1}^{t-1} W^k)(\mathbf{x}_i + \mathbf{p} + \mathbf{b})\right]_j$$

- Logistic regression with observations = # of phonological elements in the sample, assumed independent (again, not true in reality, but first approximation)
- Maximize the log-likelihood via (stochastic) gradient descent to get estimates for $\mathbf{x}, \mathbf{p}$ and $\mathbf{b}$