4. Logistic Regression, Neural Networks

NLP for CogSci Research

Marlene Staib

September 21, 2018

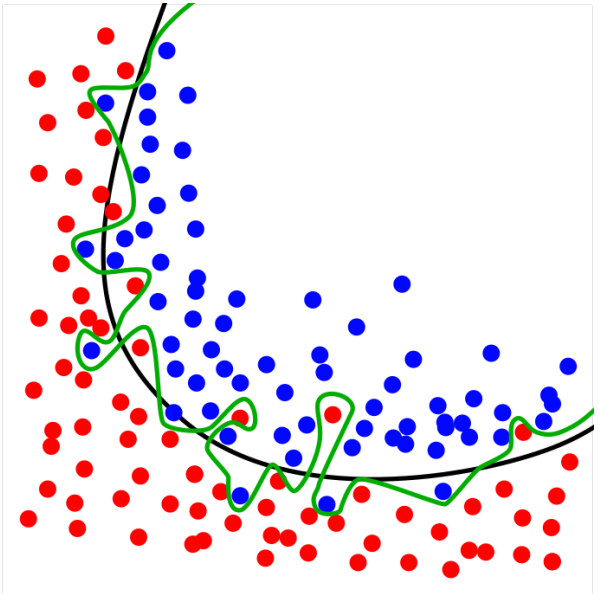
Recap: Modelling in NLP

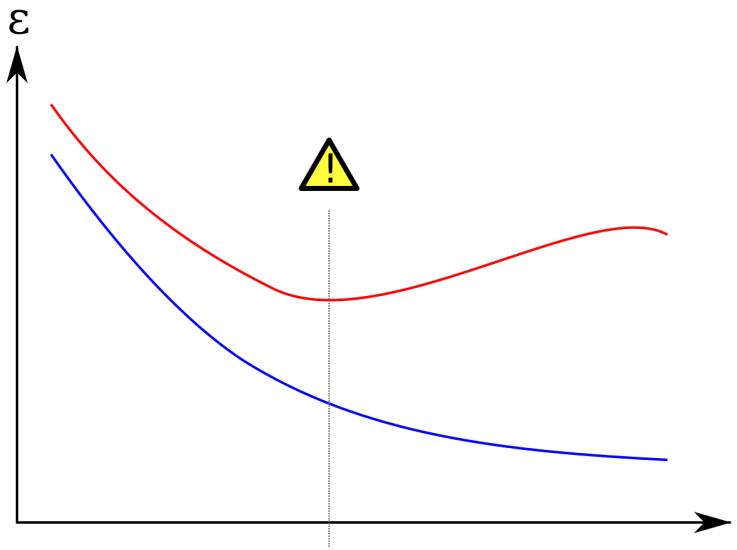
- Generative vs. discriminative models
- Supervised vs. unsupervised models
- Classification vs. regression
- Parameter optimization: MLE
- Iterative methods

Recap: Modelling in NLP

	Supervised	Unsupervised
Generative	Naive Bayes	GMMs
Discriminative	Logistic Regression Neural Nets	

Overfitting and Underfitting





Hyperparameters ($\neq$ parameters)

- Smoothing parameter α in Naive Bayes
- k in k-means
- m in GMMs
- Dimensionality of dense word embeddings
- Regularization weight: λ in Logistic Regression
- Not learned in training; tuned on eval set

Background: Logistic Regression

Recap: Logistic Regression

1. Intuition/interpretation
2. Example
3. Formula

3. Formula

Regression:

$$\hat{y} = w_1 x_1 + w_2 x_2 + \ldots + w_n x_n + b$$

$$\hat{y} = \mathbf{w}\mathbf{x} + b$$

Logistic regression:

- Put the regression output through logistic "squishing function", sigmoid:

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

- To get the probability that y is class 1, given **x**:

$$P(y = 1|\mathbf{x}) = \sigma(\mathbf{w}\mathbf{x} + b)$$

Logistic Regression: Learning the model

Gradient Descent Algorithm:

0. Initialize the weights and bias randomly

Iterate until **convergence**[1]:

1. Compute the prediction error on the training data
2. Get the derivative of the error with respect to the weights
3. Move in the direction of steepest descent

[1] **convergence**: no/little change in error

Logistic Regression: Learning the model

The likelihood of training data under the model is calculated as:

$$P(\mathbf{X}|\mathbf{w}) = \prod_i P(y=1|\mathbf{x_i},\mathbf{w})^{y_i} P(y=0|\mathbf{x_i},\mathbf{w})^{1-y_i}$$

$$P(\mathbf{X}|\mathbf{w}) = \prod_i P(y=1|\mathbf{x_i},\mathbf{w})^{y_i} (1 - P(y=1|\mathbf{x_i},\mathbf{w}))^{1-y_i}$$

We use the **negative log likelihhod** as our error function:

minimize negative log-likelihood = maximize log-likelihood

$$L(w) = -\sum_i y_i \ log(\sigma(\mathbf{wx}+b)) + (1-y_i) \ log(\sigma(\mathbf{wx}+b)$$

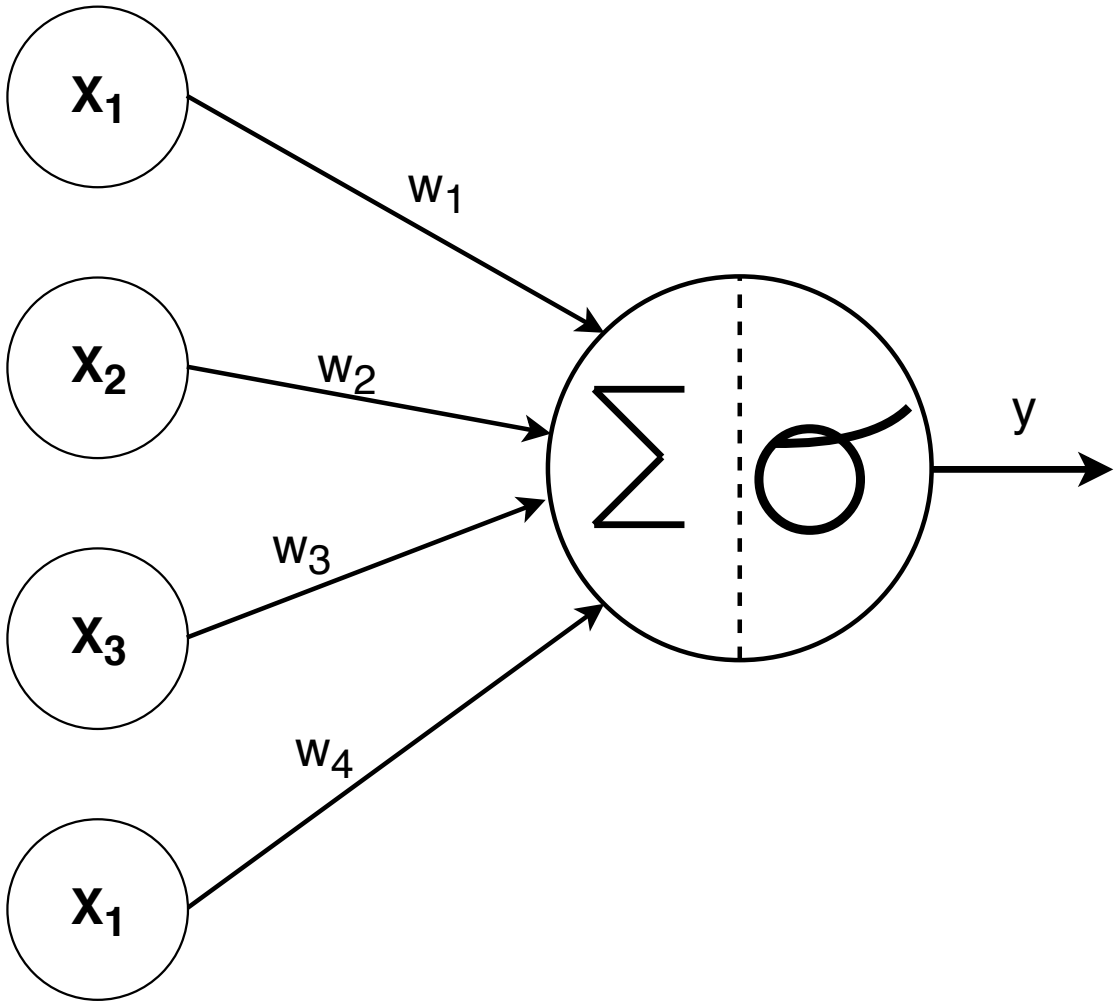
Regularisation: Avoid overfitting

Add a "regularisation term" to the error function to penalize large weights:

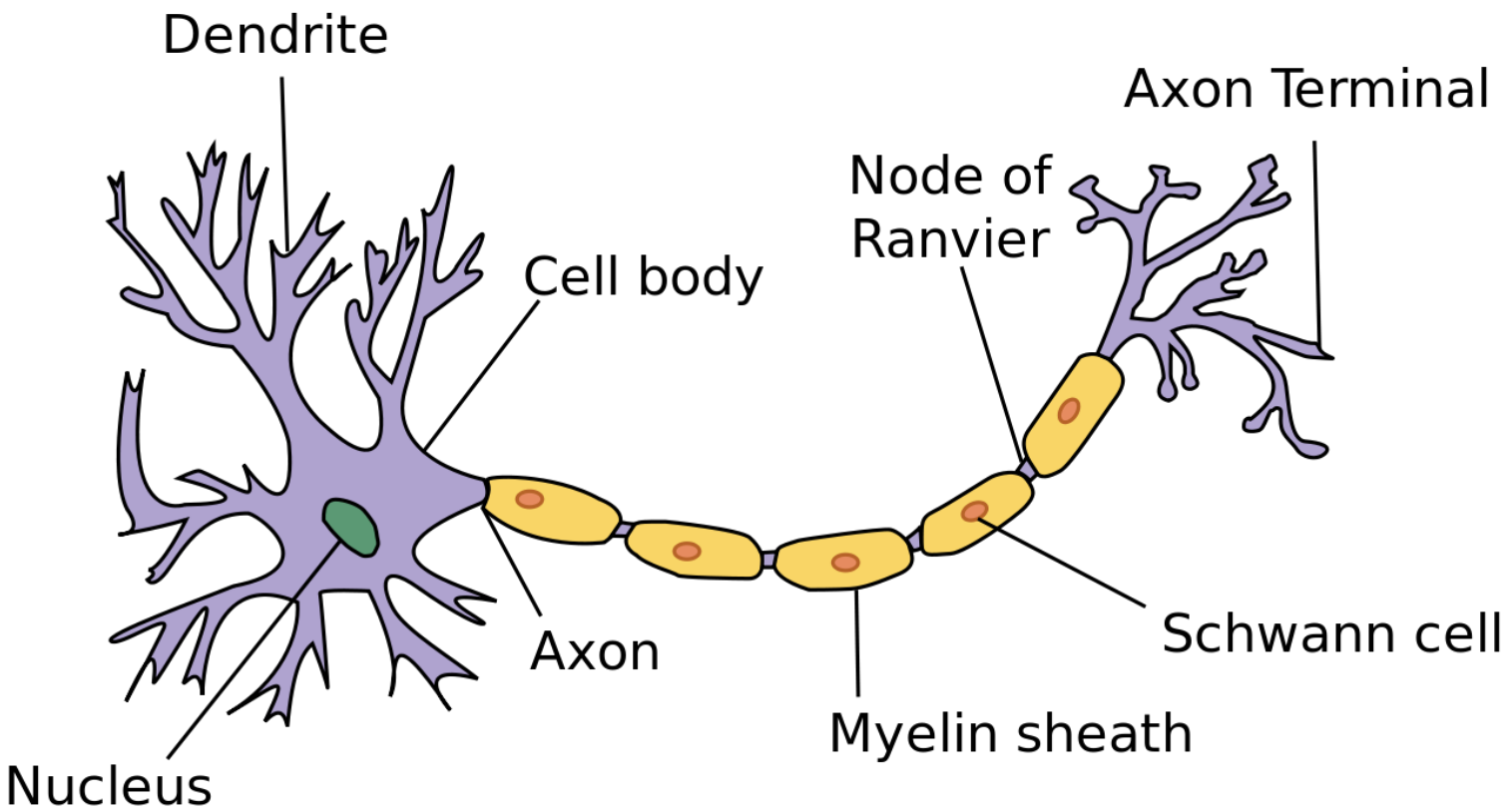
$$L_{reg}(w) = L(w) + \lambda |w|_2^2$$

λ is a tunable hyperparameter of the model.

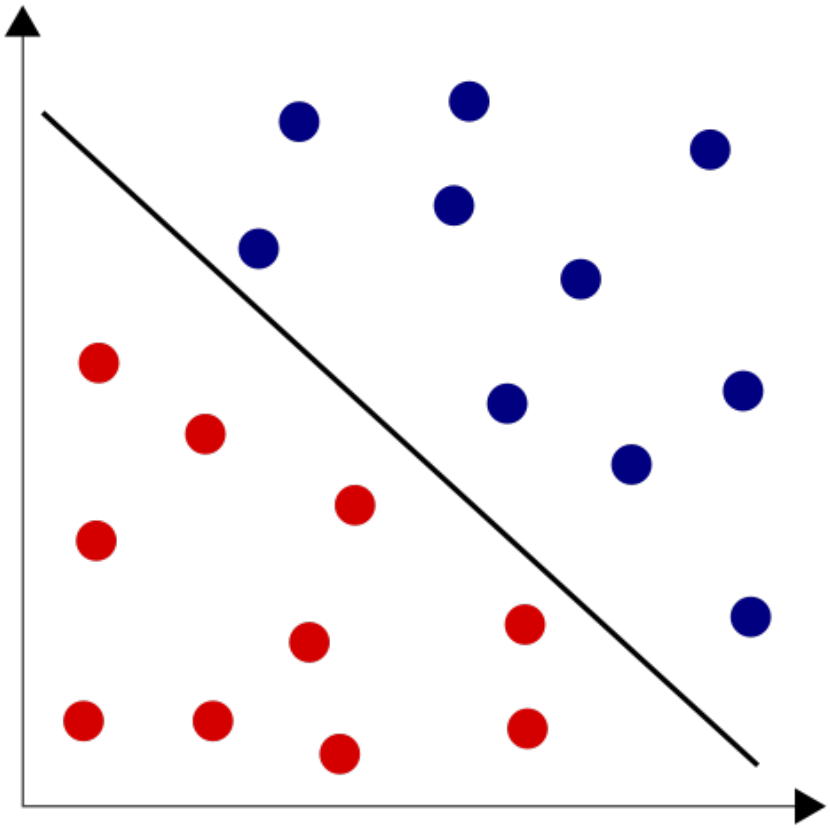
Logistic Regression as a "Neuron"

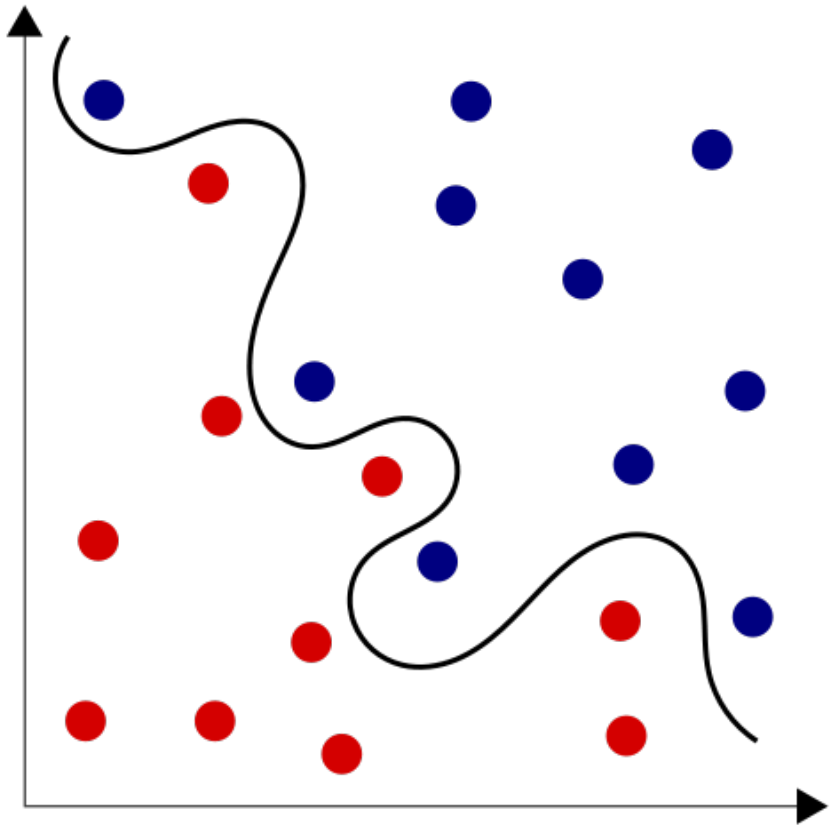


Logistic Regression as a "Neuron"?!

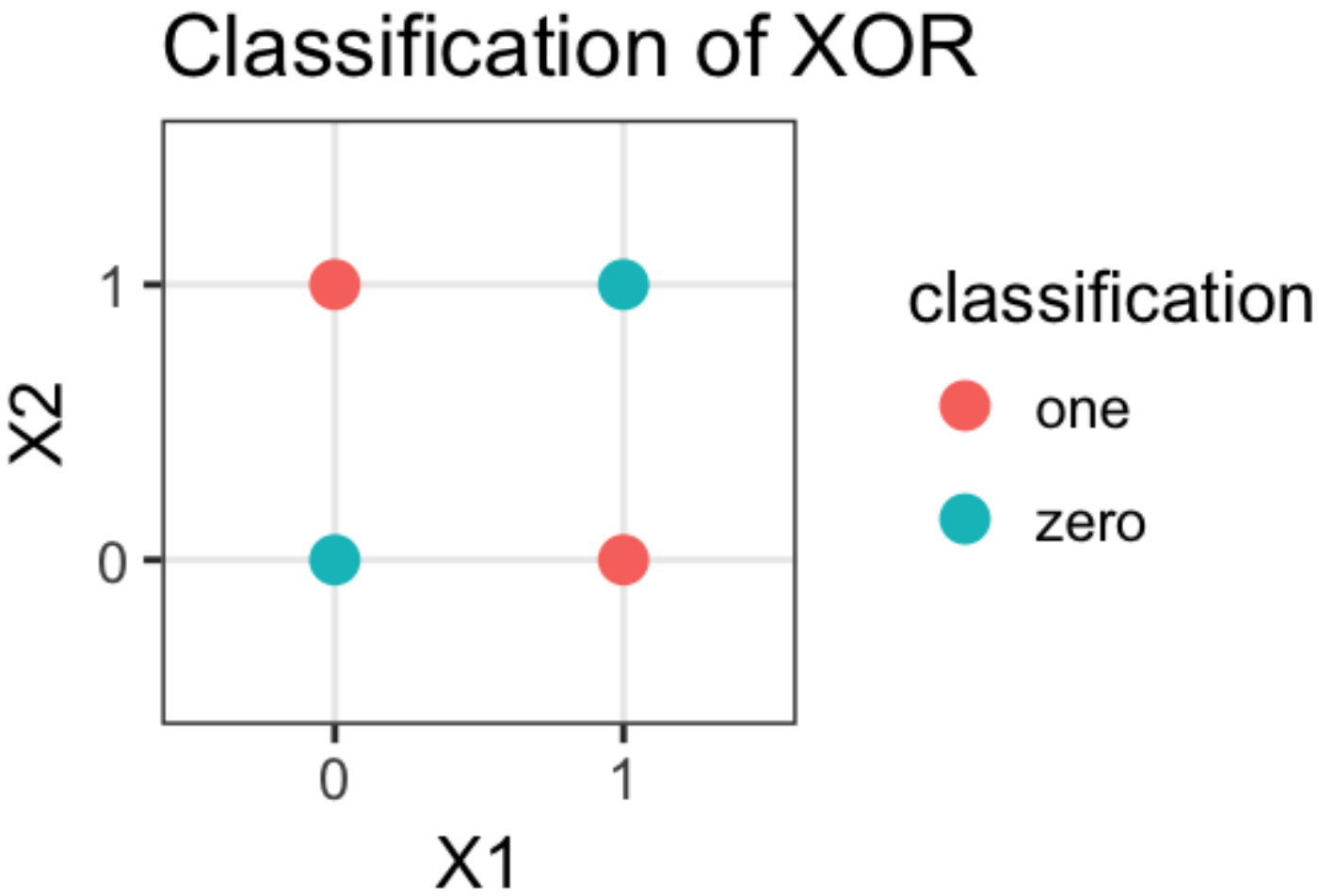


Logistic regression – linear separability





Linear separability - XOR



Multi-class regression: Softmax

A more general case of sigmoid is softmax:

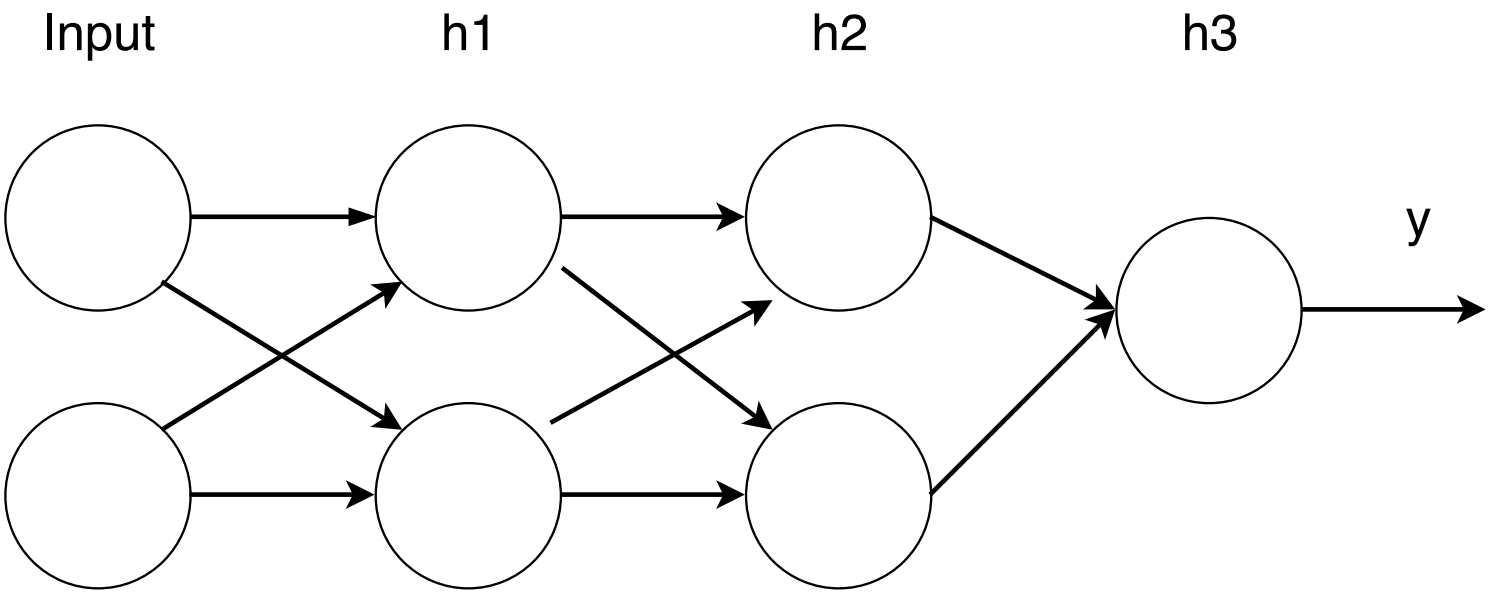
$$P(y = k|\mathbf{x}) = \frac{exp(\mathbf{w_k x})}{\sum_j exp(\mathbf{w_j x})}$$

This ensures that $\sum_j P(y = j|\mathbf{x}) = 1$ and $P(y = j|\mathbf{x}) > 0$ for all j.

Neural Networks: The Basics

Multi-layer perceptron (MLP)

A neural network: Connecting multiple regression models



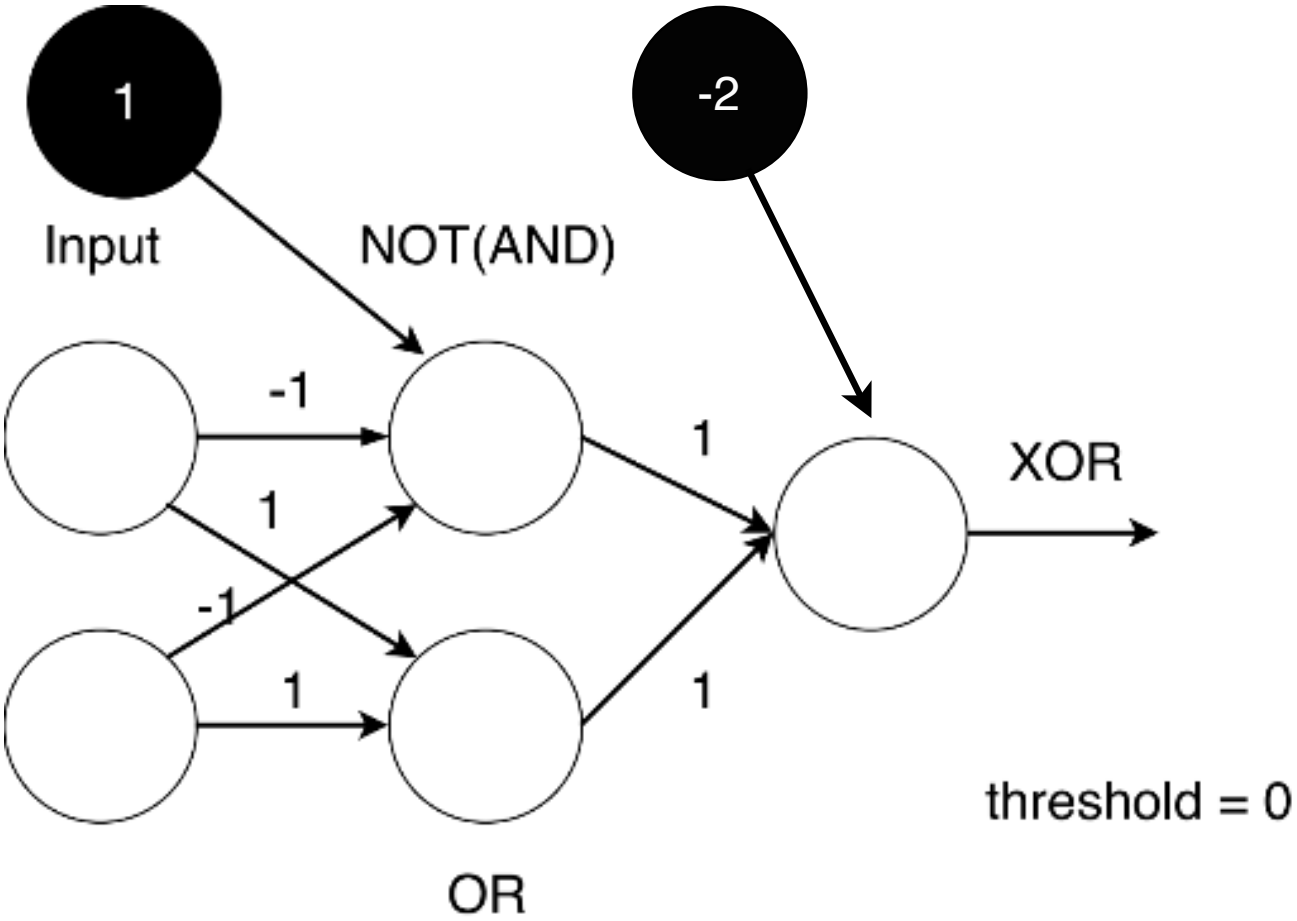
Why layers???

Why layers???

1. Intuition:
   - Each "neuron" learns to pick out certain features/qualities
   - Early/lower layers represent lower-level features
   - Building up representations

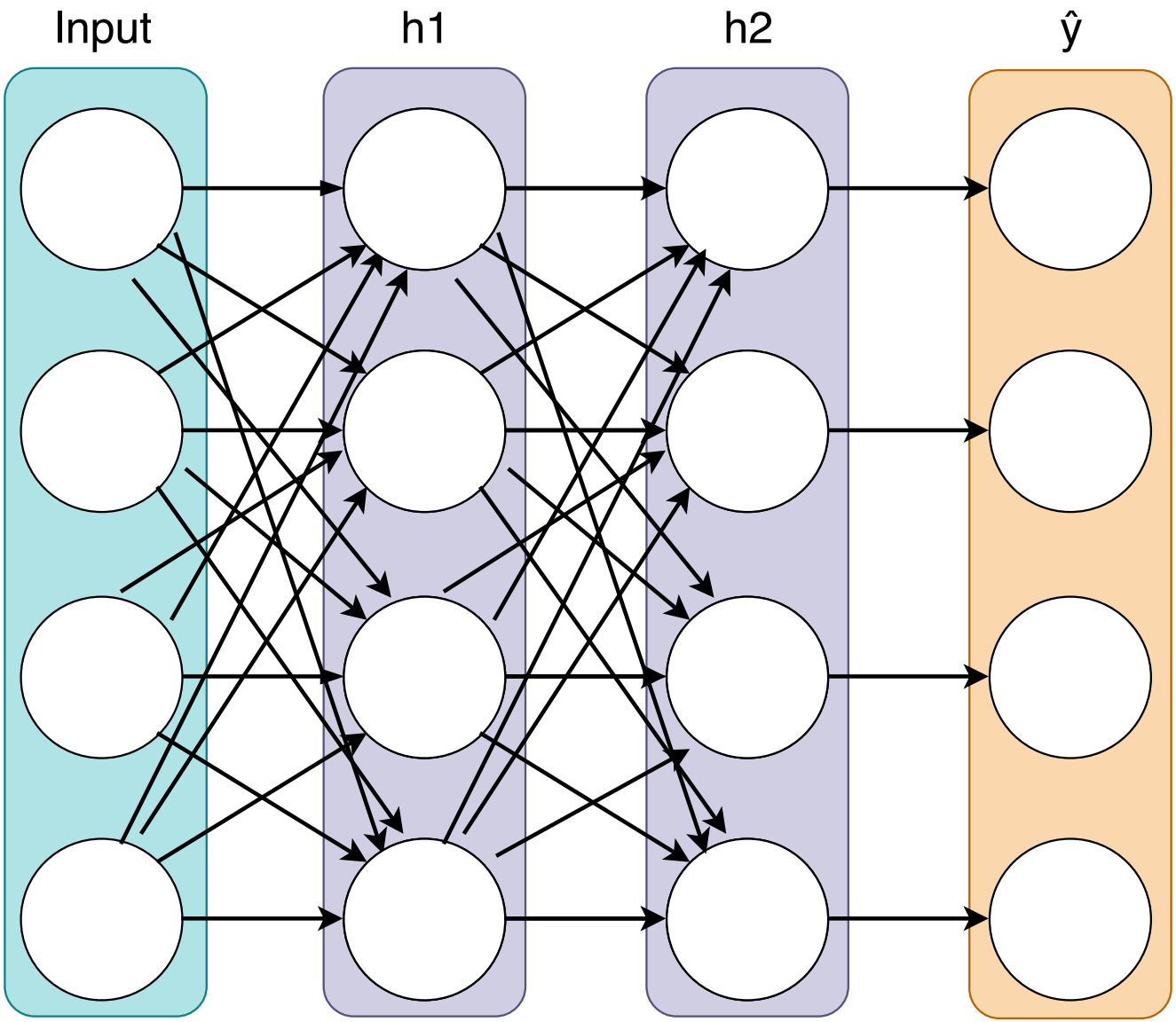
Why layers???

2. Non-linearity:



https://goo.gl/GUWdj2

Prediction: Forward pass



Prediction: Forward pass

Use matrix multiplications:

$$h_1 = \sigma(\mathbf{W_1 x} + b_1)$$

$$h_2 = \sigma(\mathbf{W_2 h_1} + b_2)$$

$$y = softmax(h2)$$

Learning: backpropagation

- Apply the gradient descent algorithm to a DNN
  - (Only slightly more complicated - use chain rule from Calculus)
  - Automatic Differentiation
- Choose a suitable loss function, such as negative log likelihood, cross-entropy, MSE, ...

NLP with Neural Nets

NLP with Neural Nets: Skip-gram

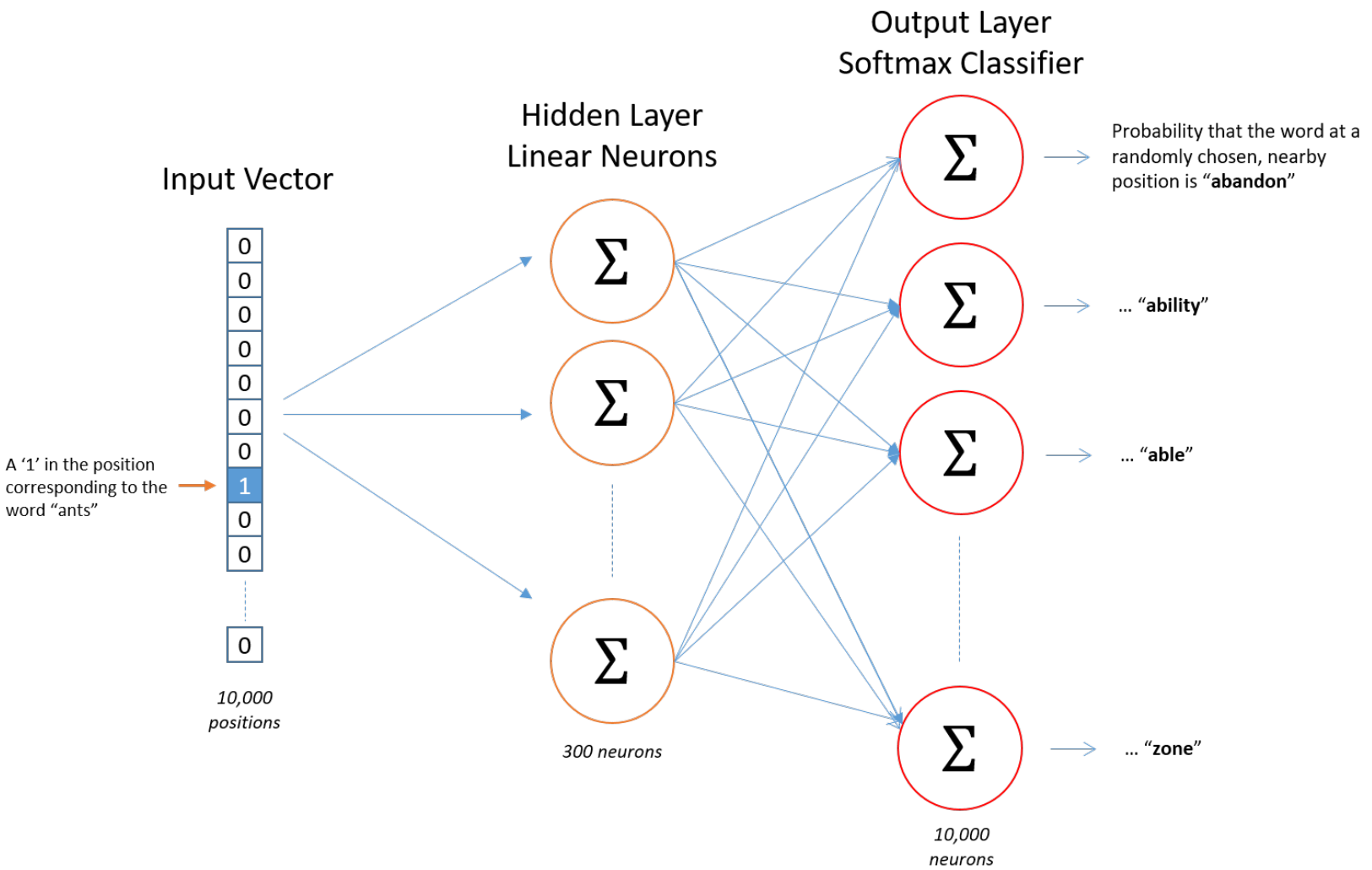


image from: https://towardsdatascience.com/word2vec-skip-gram-model-part-1-intuition-78614e4d6e0b

NLP with Neural Nets: Skip-gram

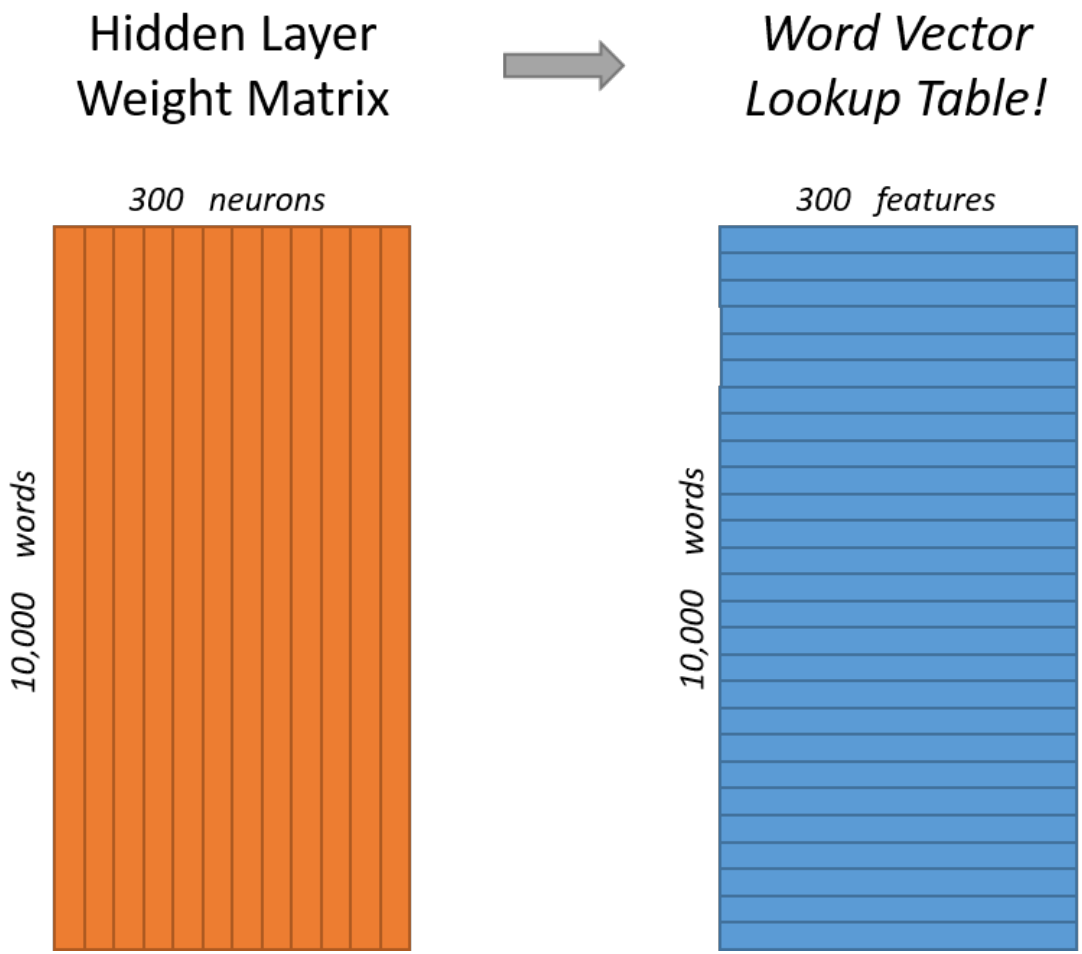


image from: https://towardsdatascience.com/word2vec-skip-gram-model-part-1-intuition-78614e4d6e0b

NLP with Neural Nets: Skip-gram

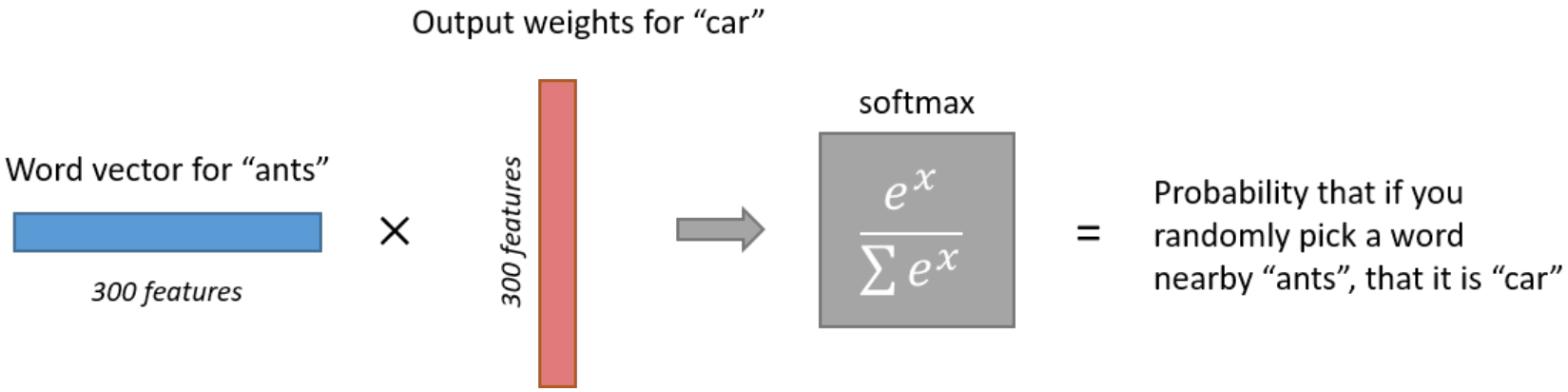
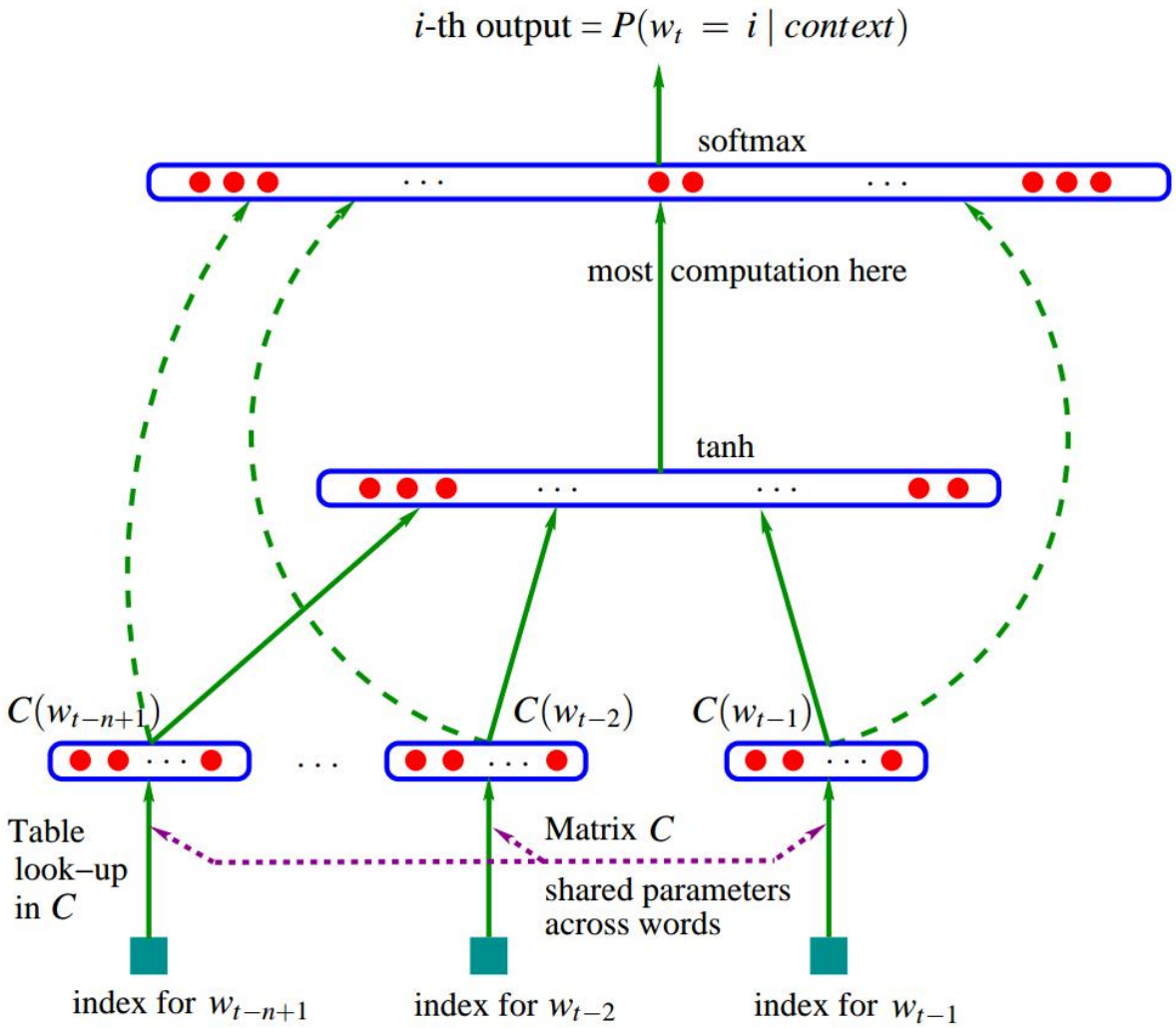


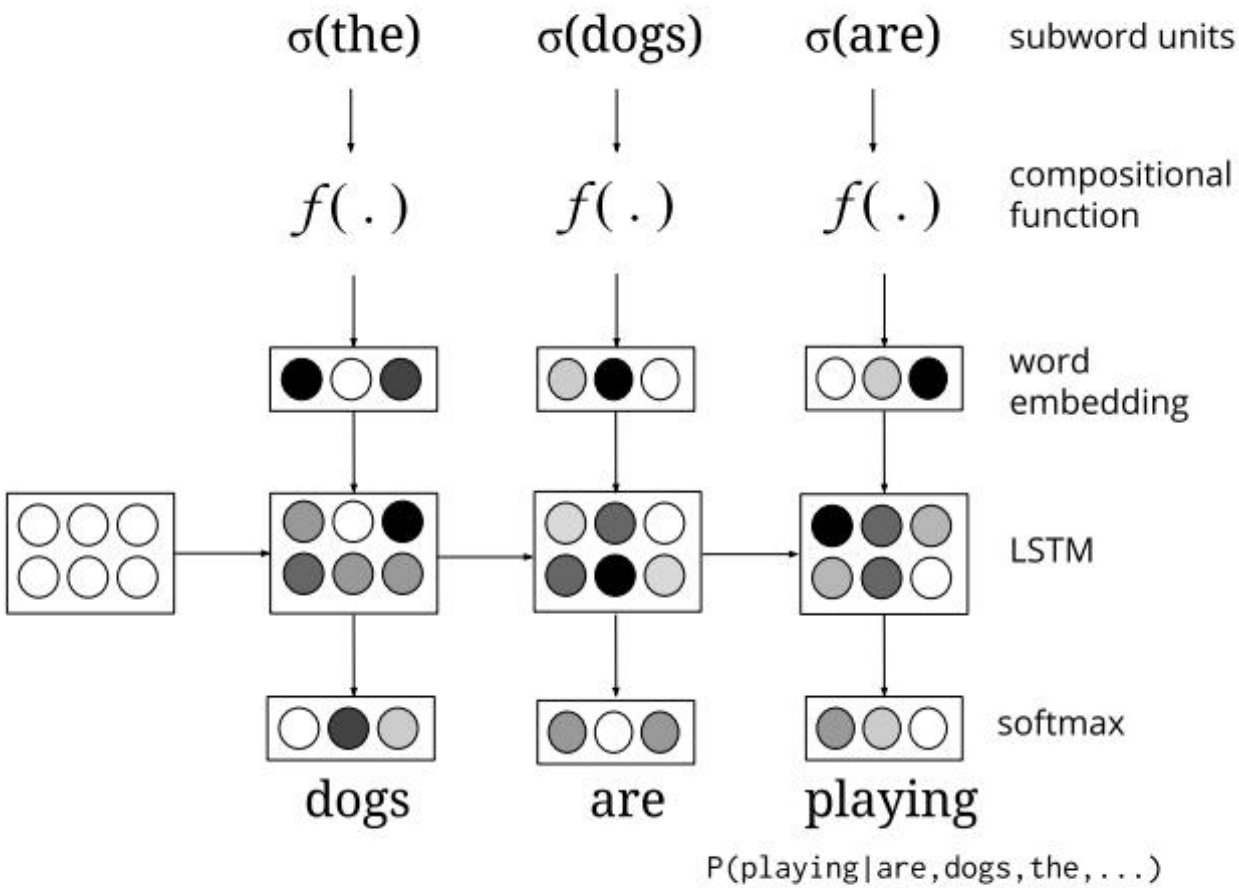
image from: https://towardsdatascience.com/word2vec-skip-gram-model-part-1-intuition-78614e4d6e0b

NLP with Neural Nets: Neural Language Model

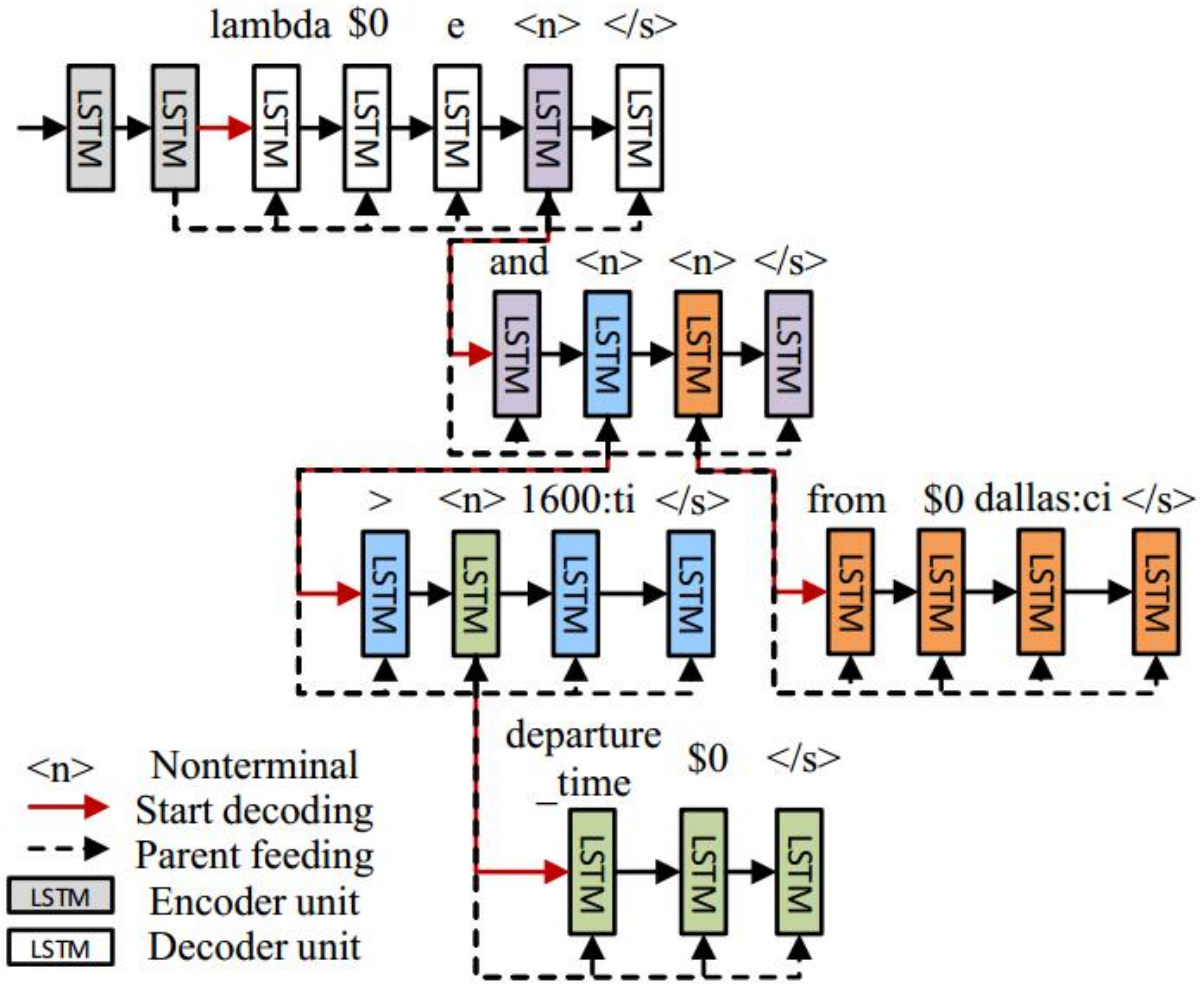


Bengio et al. (2003): A Neural Probabilistic Language Model

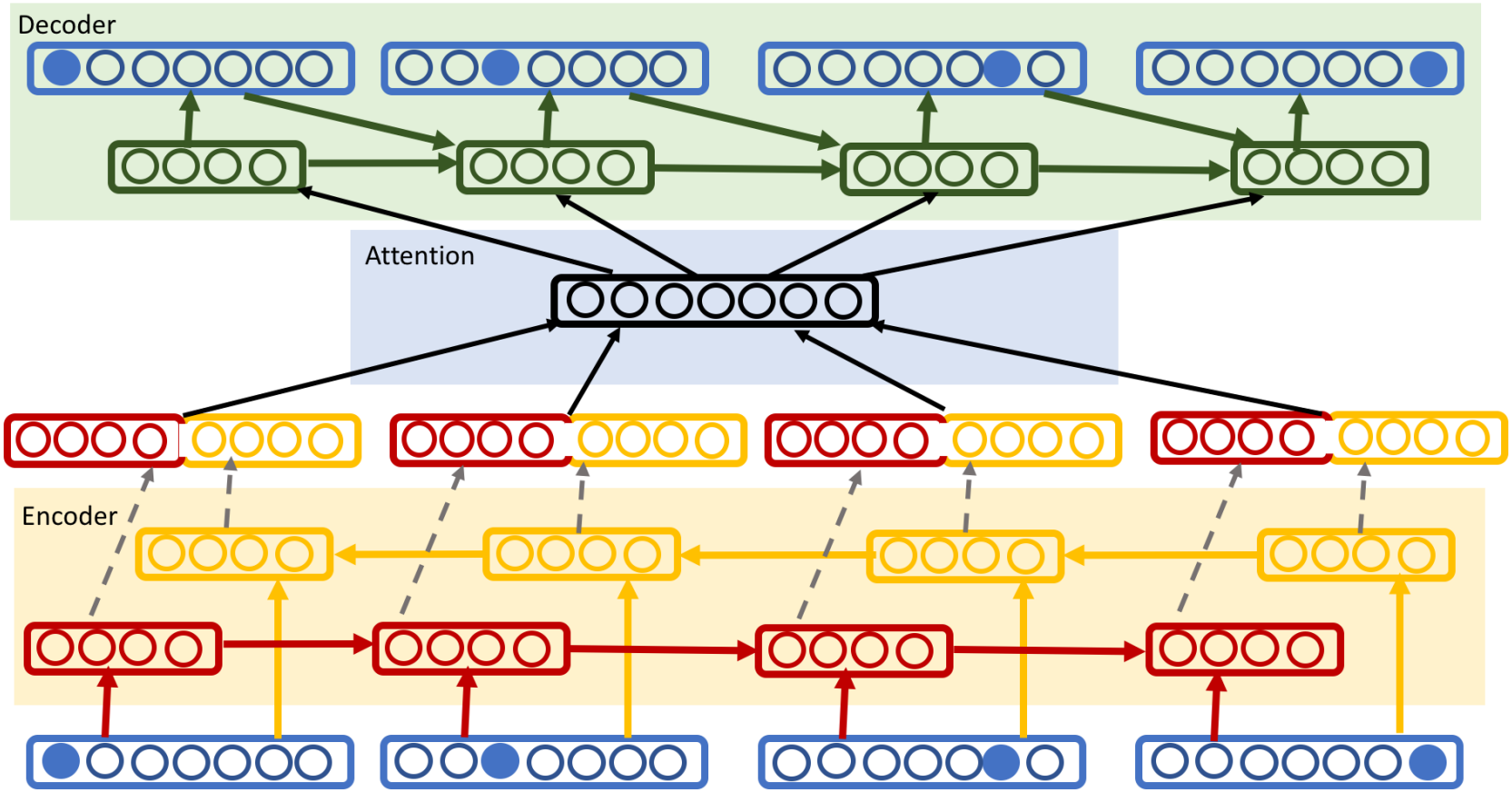
NLP with Neural Nets: Other applications



NLP with Neural Nets: Other applications



NLP with Neural Nets: Other applications



Some success stories...

- Reduction in ASR Word Error Rate from 23-27% to 16-18% in 2012
- WaveNet: https://deepmind.com/blog/wavenet-generative-model-raw-audio/
- Neural MT surpassed phrase-based/statistical MT in 2015/16
- ... (list goes on)

Why Neural Nets are a powerful tool for NLP

- Sequences ✓
- Ambiguity ✓
- Sparsity ✓

Is NLP solved (with Neural Nets)?

Discussion:

- Can you think of situations where Neural Nets would fail/be inadequate?
- Do you think DNNs are just a fashion, or "here to stay"?
- Can you think of examples in language that would be hard to learn with a DNN?

Wrap-up

NLP and Cognitive Science:

1. How can NLP support CogSci?
2. How can CogSci support NLP?