Special Topics in Natural Language Processing

CS698O

Ashutosh Modi
CSE Department, IIT Kanpur

Lecture 8: Language Models Pre-Finale
Jan 21, 2020

Information Theory Crash Course: Entropy

- Entropy: Degree of randomness in a probability distribution

Information Theory Crash Course: Entropy

- Entropy: Degree of randomness in a probability distribution

$$H(p) = -\sum_{i=1}^{N} p(X_i) \log_2 p(X_i) = -\mathbb{E}_p[\log_2 p(X)] \quad \textbf{(8.1)}$$

Information Theory Crash Course: Entropy

- Entropy: Degree of randomness in a probability distribution

$$H(p) = -\sum_{i=1}^{N} p(X_i) \log_2 p(X_i) = -\mathbb{E}_p[\log_2 p(X)]$$

Images: Walmart.com

Information Theory Crash Course: Entropy

- Entropy: Degree of randomness in a probability distribution

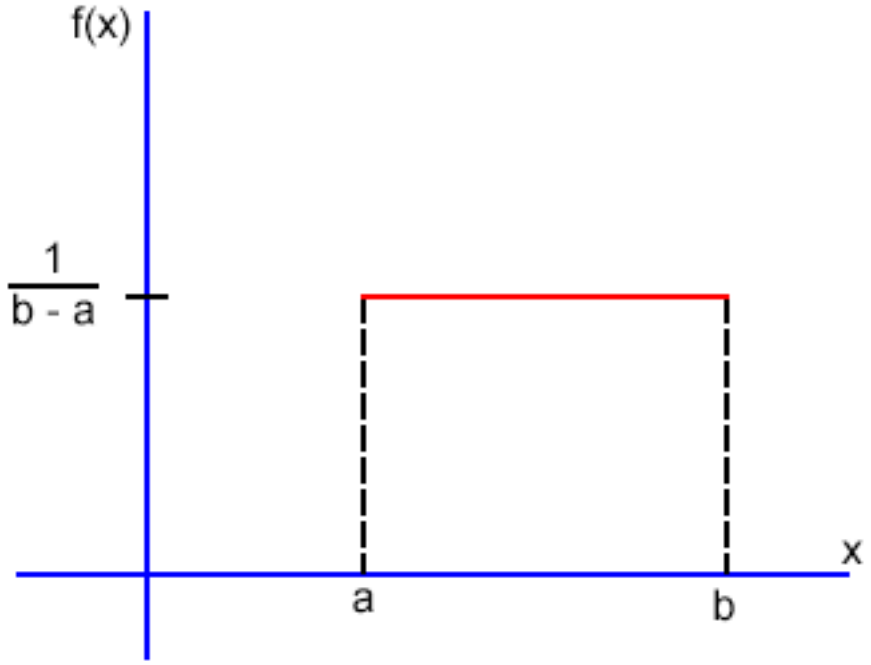
$$H(p) = -\sum_{i=1}^{N} p(X_i) \log_2 p(X_i) = -\mathbb{E}_p[\log_2 p(X)]$$

Images: Walmart.com

Information Theory Crash Course: Entropy

- Entropy: Degree of randomness in a probability distribution

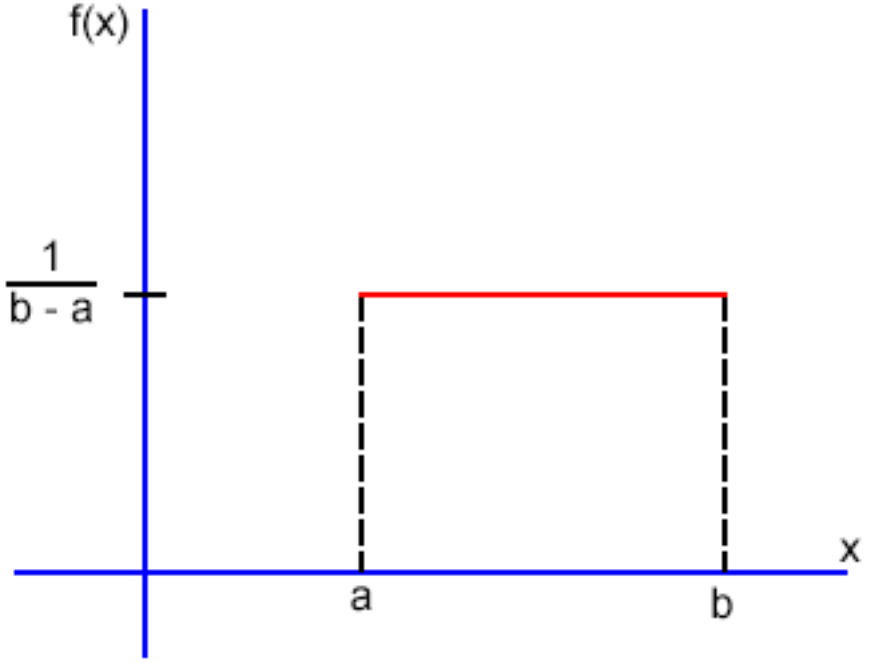
$$H(p) = -\sum_{i=1}^{N} p(X_i) \log_2 p(X_i) = -\mathbb{E}_p[\log_2 p(X)]$$

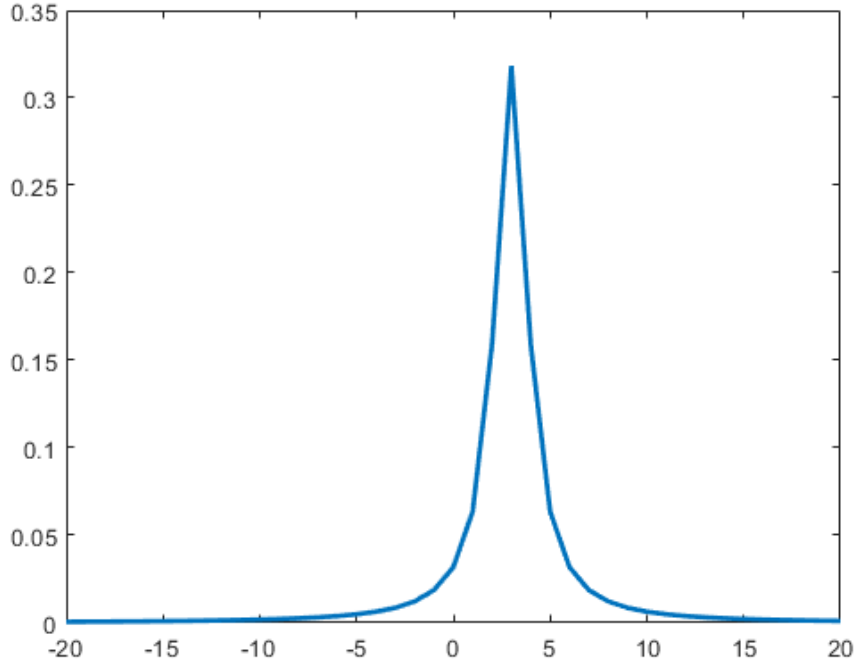


Information Theory Crash Course: Entropy

- Entropy: Degree of randomness in a probability distribution

$$H(p) = -\sum_{i=1}^{N} p(X_i) \log_2 p(X_i) = -\mathbb{E}_p[\log_2 p(X)]$$



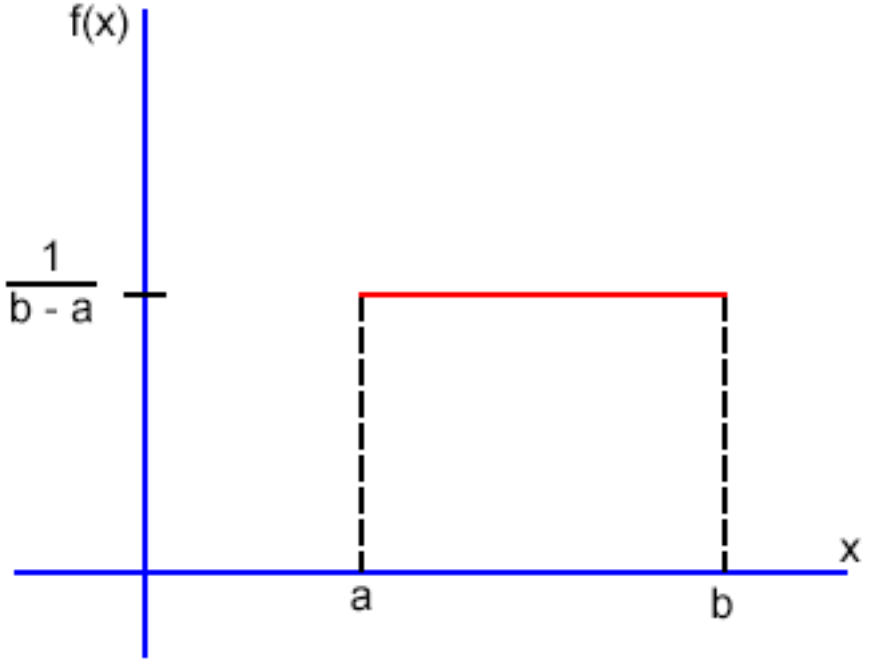


Information Theory Crash Course: Entropy

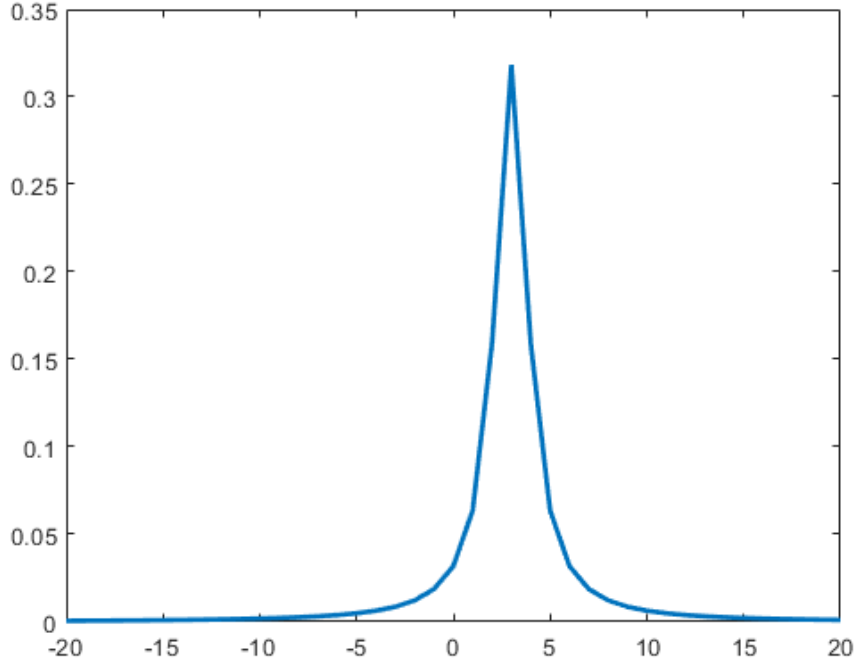
- Entropy: Degree of randomness in a probability distribution

$$H(p) = -\sum_{i=1}^{N} p(X_i) \log_2 p(X_i) = -\mathbb{E}_p[\log_2 p(X)]$$

High Entropy



Low Entropy



Information Theory Crash Course: Entropy

- Entropy: Degree of randomness in a probability distribution

$$H(p) = -\sum_{i=1}^{N} p(X_i) \log_2 p(X_i) = -\mathbb{E}_p[\log_2 p(X)]$$

- Entropy is measure of information (in *bits*) and is indicative of how much information is there in a given model
- Minimum number of bits required to encode information using optimal coding scheme

Information Theory Crash Course: Entropy

- Language Model Entropy

$$H(L) = -\sum_{w_1^M \in \mathbb{L}} p(w_1, w_2, \ldots, w_M) \log_2 \; p(w_1, w_2, \ldots, w_M)$$

- Smaller the entropy of a Language Model the better

Information Theory Crash Course: Cross-Entropy

- Cross-Entropy: Indicative of distance between two distributions

$$H(p, q) = -\sum_{i=1}^{N} p(X_i) \log_2 q(X_i) \quad \textbf{(8.2)}$$

Information Theory Crash Course: Cross-Entropy

- Cross-Entropy:

$$H(p,q) = -\sum_{i=1}^{N} p(X_i)\log_2 q(X_i) \quad \textbf{(8.2)}$$

$$H(p) \leq H(p,q) \quad \textbf{(8.3)}$$

- Cross Entropy helps to estimate true entropy of a sequence of words drawn according to probability *p*

Information Theory Crash Course: Cross-Entropy

- Cross-Entropy of a Language Model:

$$H(W) = -\sum_{w_1^M \in \mathbb{L}} \frac{1}{M} \log_2 p(w_1, \ldots, w_M) \quad \textbf{(8.4)}$$

Information Theory Crash Course: Cross-Entropy

- Cross-Entropy of a Language Model:

$$H(W) = -\sum_{w_1^M \in \mathbb{L}} \frac{1}{M} \log_2 p(w_1, \ldots, w_M) \quad \textbf{(8.4)}$$

- Is it related to Perplexity?

Information Theory Crash Course: Cross-Entropy

- Cross-Entropy of a Language Model:

$$H(W) = -\sum_{w_1^M \in \mathbb{L}} \frac{1}{M} \log_2 p(w_1, \ldots, w_M)$$

- Is it related to Perplexity?

$$l = \frac{1}{M} \sum_{i=1}^{m} log_2\ p(S_i)$$

$$\text{Perplexity} := 2^{-l}$$

Information Theory Crash Course: Cross-Entropy

- Cross-Entropy of a Language Model:

$$H(W) = -\sum_{w_1^M \in \mathbb{L}} \frac{1}{M} \log_2 p(w_1, \ldots, w_M)$$

- Is it related to Perplexity?

$$l = \frac{1}{M} \sum_{i=1}^{m} log_2\ p(S_i)$$

$$\text{Perplexity} := 2^{-l}$$

$$\text{Perplexity} = 2^{H(W)} \quad \textbf{(8.5)}$$

Cross Entropy Loss

$$\hat{p}(y \mid x; w) = \frac{exp(w^T \phi(x, y))}{\sum_{y' \in \mathcal{Y}} exp(w^T \phi(x, y'))}$$

Cross Entropy Loss

$$\hat{p}(y \mid x; w) = \frac{exp(w^T \phi(x, y))}{\sum_{y' \in \mathcal{Y}} exp(w^T \phi(x, y'))}$$

$$\mathcal{CE}_{\mathcal{L}oss}(w) = -\frac{1}{N} \sum_{i=1}^{N} \sum_{l=1}^{|\mathcal{V}|} \delta_{y^{(i)}=l} \log \left(\hat{p}(y^{(i)} = l \mid x^{(i)}; w) \right)$$

(8.6)

Binary Cross Entropy Loss

$$\hat{p}(y \mid x; w) = \frac{exp(w^T \phi(x, y))}{\sum_{y' \in \mathcal{Y}} exp(w^T \phi(x, y'))}$$

$$\mathcal{CE}_{\mathcal{L}oss}(w) = -\frac{1}{N} \sum_{i=1}^{N} \sum_{l=1}^{|\mathcal{V}|} \delta_{y^{(i)}=l} \log \left(\hat{p}(y^{(i)} = l \mid x^{(i)}; w) \right)$$

$$\text{Binary}-\mathcal{CE}_{\mathcal{L}oss} = -\frac{1}{N} \sum_{i=1}^{N} y \log \left(\hat{p}(y = 1 \mid x; w) \right) + (1-y) \left(1 - \log \left(\hat{p}(y = 1 \mid x; w) \right) \right)$$

(8.7)

Log-Linear Model Parameter Estimation

$$\hat{p}(y \mid x; w) = \frac{exp(w^T \phi(x, y))}{\sum_{y' \in \mathcal{Y}} exp(w^T \phi(x, y'))}$$

$$\mathcal{CE}_{\mathcal{L}oss}(w) = -\frac{1}{N} \sum_{i=1}^{N} \sum_{l=1}^{|\mathcal{V}|} \delta_{y^{(i)}=l} \log \left(\hat{p}(y^{(i)} = l \mid x^{(i)}; w) \right)$$

How do you estimate the parameters of the model?

Log-Linear Model Parameter Estimation

$$\hat{p}(y \mid x; w) = \frac{exp(w^T \phi(x, y))}{\sum_{y' \in \mathcal{Y}} exp(w^T \phi(x, y'))}$$

$$\mathcal{CE}_{\mathcal{L}oss}(w) = -\frac{1}{N} \sum_{i=1}^{N} \sum_{l=1}^{|\mathcal{V}|} \delta_{y^{(i)}=l} \log \left(\hat{p}(y^{(i)} = l \mid x^{(i)}; w) \right)$$

$$w^* = \underset{w \in \mathbb{R}^d}{\operatorname{argmin}} \ \mathcal{CE}_{\mathcal{L}oss}(w) \quad \textbf{(8.8)}$$

Parameters can be found by minimizing the Cross Entropy loss

Log-Linear Model Parameter Estimation

$$\hat{p}(y \mid x; w) = \frac{exp(w^T \phi(x, y))}{\sum_{y' \in \mathcal{Y}} exp(w^T \phi(x, y'))}$$

$$\mathcal{L}(w) = log \left(\prod_{i=1}^{n} \hat{p}(y^{(i)} \mid x^{(i)}; w) \right)$$

Log-Linear Model Parameter Estimation

$$\hat{p}(y \mid x; w) = \frac{exp(w^T \phi(x, y))}{\sum_{y' \in \mathcal{Y}} exp(w^T \phi(x, y'))}$$

$$\mathcal{L}(w) = log \left(\prod_{i=1}^{n} \hat{p}(y^{(i)} \mid x^{(i)}; w) \right)$$

$$w^* = \underset{w \in \mathbb{R}^d}{\text{argmax}} \, \mathcal{L}(w) = \underset{w \in \mathbb{R}^d}{\text{argmin}} \, -\mathcal{L}(w)$$

Log-Linear Model Parameter Estimation

$$\hat{p}(y \mid x; w) = \frac{exp(w^T \phi(x, y))}{\sum_{y' \in \mathcal{Y}} exp(w^T \phi(x, y'))}$$

$$\mathcal{L}(w) = log \left(\prod_{i=1}^{n} \hat{p}(y^{(i)} \mid x^{(i)}; w) \right)$$

$$w^* = \underset{w \in \mathbb{R}^d}{\text{argmax}}\, \mathcal{L}(w) = \underset{w \in \mathbb{R}^d}{\text{argmin}}\, \underbrace{-\mathcal{L}(w)}_{\text{Negative Log Liklihood: } \mathcal{NLL}}$$

Log-Linear Model Parameter Estimation

$$\hat{p}(y \mid x; w) = \frac{exp(w^T \phi(x, y))}{\sum_{y' \in \mathcal{Y}} exp(w^T \phi(x, y'))}$$

$$\mathcal{CE}_{\mathcal{L}oss}(w) = -\frac{1}{N} \sum_{i=1}^{N} \sum_{l=1}^{|\mathcal{V}|} \delta_{y^{(i)}=l} \log \left(\hat{p}(y^{(i)} = l \mid x^{(i)}; w) \right)$$

$$\mathcal{NLL}(w) = -\frac{1}{N} log \left(\prod_{i=1}^{n} \hat{p}(y^{(i)} \mid x^{(i)}; w) \right)$$

What is the relationship between cross entropy loss and negative log likelihood?

Log-Linear Model Parameter Estimation

$$\hat{p}(y \mid x; w) = \frac{exp(w^T \phi(x, y))}{\sum_{y' \in \mathcal{Y}} exp(w^T \phi(x, y'))}$$

$$\mathcal{CE}_{\mathcal{L}oss}(w) = -\frac{1}{N} \sum_{i=1}^{N} \sum_{l=1}^{|\mathcal{V}|} \delta_{y^{(i)}=l} \log \left(\hat{p}(y^{(i)} = l \mid x^{(i)}; w) \right)$$

$$\mathcal{NLL}(w) = -\frac{1}{N} log \left(\prod_{i=1}^{n} \hat{p}(y^{(i)} \mid x^{(i)}; w) \right)$$

What is the relationship between cross entropy loss and negative log likelihood?

$$\mathcal{CE}_{\mathcal{L}oss}(w) = \mathcal{NLL}(w) \quad \textbf{(8.9)}$$

MaxEnt Models

- Log-Linear model are also called by other names:
  - *Maximum Entropy (MaxEnt) Model* (Mainly in NLP community). But Why?
  - Multinomial Logistic Regression (Softmax activation in NN community)

MaxEnt Models

- Log-Linear model are also called by other names:
  - *Maximum Entropy (MaxEnt) Model* (Mainly in NLP community). But Why?
  - Multinomial Logistic Regression (Softmax activation in NN community)

- MaxEnt Model Intuition: The probabilistic model should follow the constraints imposed by the data but beyond these constraints, the model should make the fewest possible assumptions (Occam's Razor) i.e. maximize the entropy.

MaxEnt Models

- MaxEnt Model Intuition: The probabilistic model should follow the constraints imposed by the data but beyond these constraints, the model should make the fewest possible assumptions (Occam's Razor) i.e. maximize the entropy.

- The model builds probability distribution by continuously adding features.

MaxEnt Models

- MaxEnt Model Intuition: The probabilistic model should follow the constraints imposed by the data but beyond these constraints, the model should make the fewest possible assumptions (Occam's Razor) i.e. maximize the entropy.

- The model builds probability distribution by continuously adding features.

- Each feature (indicator function) picks a subset of training data and in turn imposes a constraint on the distribution.

MaxEnt Models

- MaxEnt Model Intuition: The probabilistic model should follow the constraints imposed by the data but beyond these constraints, the model should make the fewest possible assumptions (Occam's Razor) i.e. maximize the entropy.
- The model builds probability distribution by continuously adding features.
- Each feature (indicator function) picks a subset of training data and in turn imposes a constraint on the distribution.
- The distribution for this subset should match the empirical distribution observed in the training data.

MaxEnt Models

- Constrained Optimization Problem

$$\underset{p \in \mathcal{P}}{\text{argmax}} \; H(p(Y \mid X))$$

$$\text{s.t.} \qquad \mathbb{E}_{p(Y|X)}[\phi_j(x, y)] = \frac{1}{N} \sum_{i=1}^{N} \phi_j(x^{(i)}, y^{(i)}) \qquad \forall j = 1, \ldots, d$$

MaxEnt Models

- Constrained Optimization Problem

$$\underset{p \in \mathcal{P}}{\text{argmax}} \; H(p(Y \mid X))$$

$$\text{s.t.} \qquad \mathbb{E}_{p(Y|X)}[\phi_j(x, y)] = \frac{1}{N} \sum_{i=1}^{N} \phi_j(x^{(i)}, y^{(i)}) \qquad \forall j = 1, \ldots, d$$

SOLUTION:

$$p(y \mid x; w) = \frac{exp(w^T \phi(x, y))}{\sum_{y' \in \mathcal{Y}} exp(w^T \phi(x, y'))}$$

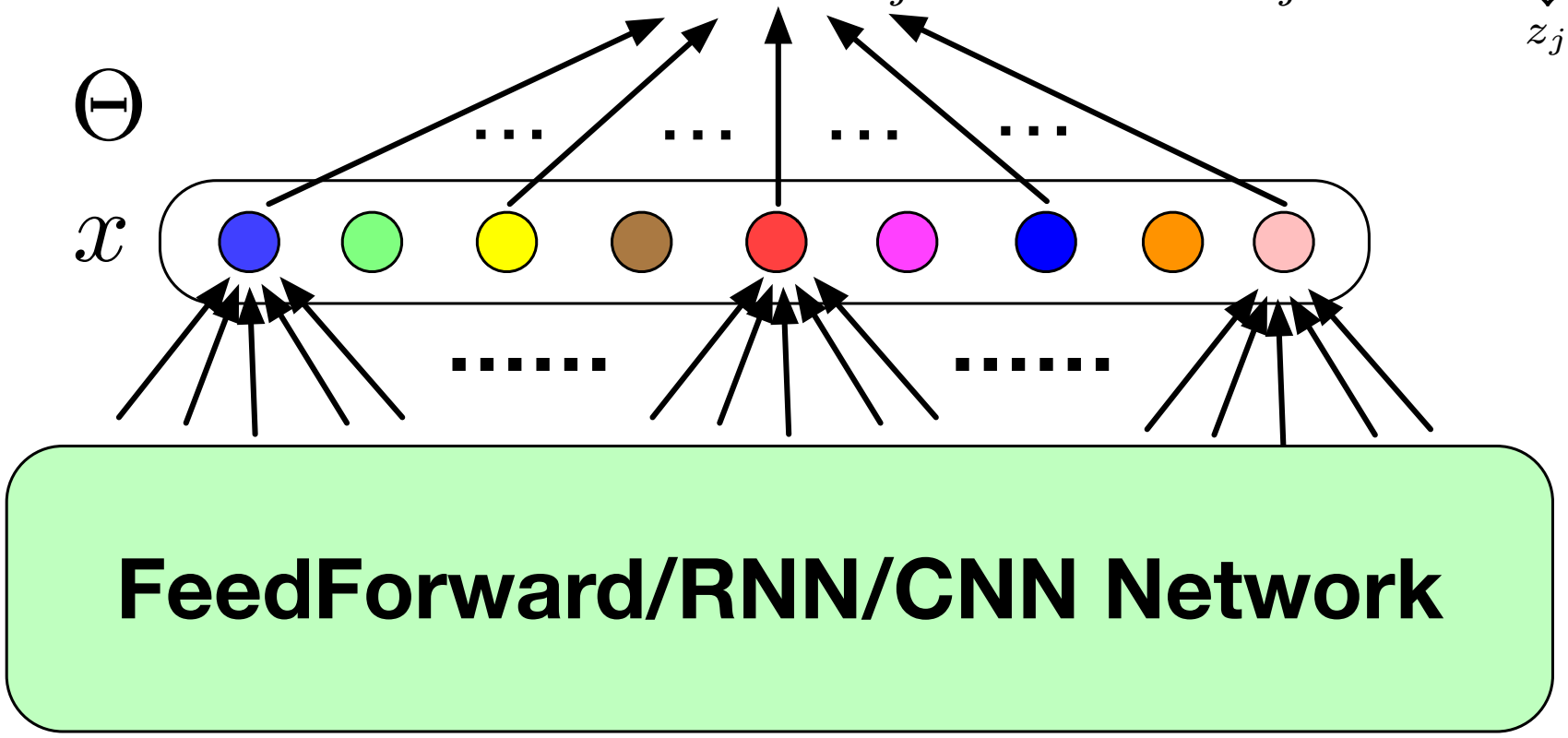
Refer [2]

MaxEnt Models

- Log-Linear model are also called by other names:
  - *Maximum Entropy (MaxEnt) Model* (Mainly in NLP community). But Why?
  - Multinomial Logistic Regression (Softmax activation in NN community)

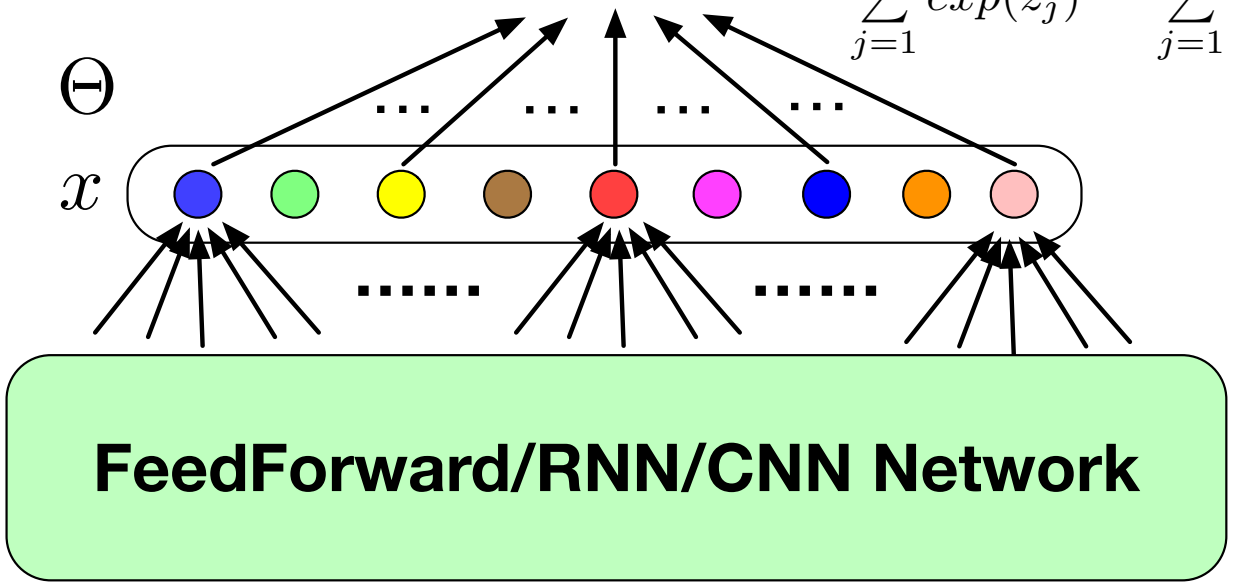
Classification in NN: Softmax Function

$$\text{Softmax output} = p(y = l|x) = \sigma(\mathbf{z})_l = \frac{exp(z_l)}{\sum_{j=1}^{L} exp(z_j)} = \frac{exp(\overbrace{\theta_l^T x}^{z_l})}{\sum_{j=1}^{L} exp(\underbrace{\theta_j^T x}_{z_j})}$$



Classification in NN: Softmax Function

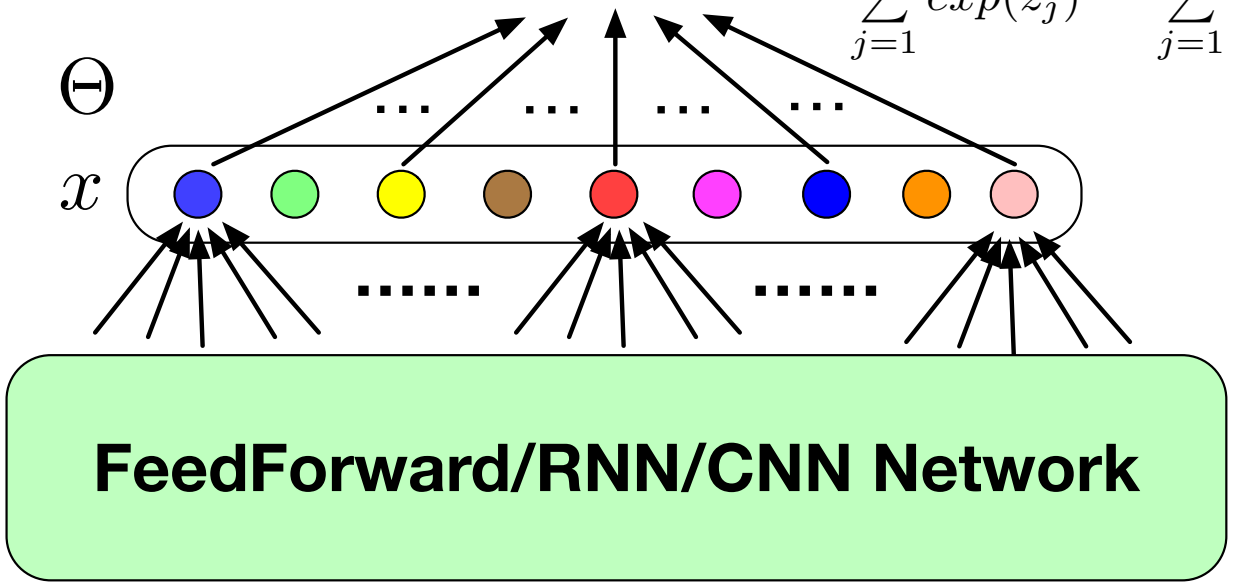
$$\text{Softmax output} = p(y = l|x) = \sigma(\mathbf{z})_l = \frac{exp(z_l)}{\sum_{j=1}^{L} exp(z_j)} = \frac{exp(\overbrace{\theta_l^T x}^{z_l})}{\sum_{j=1}^{L} exp(\underbrace{\theta_j^T x}_{z_j})}$$



$$\Theta = \begin{bmatrix} — & \theta_1^T & — \\ & \vdots & \\ — & \theta_l^T & — \\ & \vdots & \\ — & \theta_L^T & — \end{bmatrix} \quad x = \begin{bmatrix} x_1 \\ \vdots \\ x_k \\ \vdots \\ x_d \end{bmatrix}$$

Classification in NN: Softmax Function

$$\text{Softmax output} = p(y = l|x) = \sigma(\mathbf{z})_l = \frac{exp(z_l)}{\sum_{j=1}^{L} exp(z_j)} = \frac{exp(\overbrace{\theta_l^T x}^{z_l})}{\sum_{j=1}^{L} exp(\underbrace{\theta_j^T x}_{z_j})}$$



$$\Theta = \begin{bmatrix} — & \theta_1^T & — \\ & \vdots & \\ — & \theta_l^T & — \\ & \vdots & \\ — & \theta_L^T & — \end{bmatrix} \quad x = \begin{bmatrix} x_1 \\ \vdots \\ x_k \\ \vdots \\ x_d \end{bmatrix}$$

$$\sigma(\mathbf{z}) = \frac{1}{Z} exp \left(\begin{bmatrix} — & \theta_1^T & — \\ & \vdots & \\ — & \theta_l^T & — \\ & \vdots & \\ — & \theta_L^T & — \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_k \\ \vdots \\ x_d \end{bmatrix} \right)$$

Softmax Function Properties

$$\text{Softmax output} = p(y = l|x) = \sigma(\mathbf{z})_l = \frac{exp(z_l)}{\sum_{j=1}^{L} exp(z_j)} = \frac{exp(\overbrace{\theta_l^T x}^{z_l})}{\sum_{j=1}^{L} exp(\underbrace{\theta_j^T x}_{z_j})} \quad \textbf{(8.10)}$$

Softmax Function Properties

$$\text{Softmax output} = p(y = l|x) = \sigma(\mathbf{z})_l = \frac{exp(z_l)}{\sum_{j=1}^{L} exp(z_j)} = \frac{exp(\overbrace{\theta_l^T x}^{z_l})}{\sum_{j=1}^{L} exp(\underbrace{\theta_j^T x}_{z_j})} \quad \textbf{(8.10)}$$

- Monotonic function

Softmax Function Properties

$$\text{Softmax output} = p(y = l|x) = \sigma(\mathbf{z})_l = \frac{exp(z_l)}{\sum_{j=1}^{L} exp(z_j)} = \frac{exp(\overbrace{\theta_l^T x}^{z_l})}{\sum_{j=1}^{L} exp(\underbrace{\theta_j^T x}_{z_j})} \quad \textbf{(8.10)}$$

- Over-parameterization

$$\frac{exp((\theta_l - \psi)^T x)}{\sum_{j=1}^{L} exp((\theta_l - \psi)^T x)} = ?$$

Softmax Function Properties

$$\text{Softmax output} = p(y = l|x) = \sigma(\mathbf{z})_l = \frac{exp(z_l)}{\sum_{j=1}^{L} exp(z_j)} = \frac{exp(\overbrace{\theta_l^T x}^{z_l})}{\sum_{j=1}^{L} exp(\underbrace{\theta_j^T x}_{z_j})}$$

- Over-parameterization

$$\frac{exp((\theta_l - \psi)^T x)}{\sum_{j=1}^{L} exp((\theta_l - \psi)^T x)} = \sigma(\mathbf{z})_l$$

Softmax Function Properties

$$\text{Softmax output} = p(y = l|x) = \sigma(\mathbf{z})_l = \frac{exp(z_l)}{\sum_{j=1}^{L} exp(z_j)} = \frac{exp(\overbrace{\theta_l^T x}^{z_l})}{\sum_{j=1}^{L} exp(\underbrace{\theta_j^T x}_{z_j})}$$

- Over-parameterization

$$\frac{exp((\theta_l - \psi)^T x)}{\sum_{j=1}^{L} exp((\theta_l - \psi)^T x)} = \sigma(\mathbf{z})_l$$

Minimizing Parameters are not unique.
Multiple parameter settings can lead to same minima

Softmax Function Properties

$$\text{Softmax output} = p(y = l|x) = \sigma(\mathbf{z})_l = \frac{exp(z_l)}{\sum_{j=1}^{L} exp(z_j)} = \frac{exp(\overbrace{\theta_l^T x}^{z_l})}{\sum_{j=1}^{L} exp(\underbrace{\theta_j^T x}_{z_j})}$$

- Overparameterization is useful for Numerical Stability

$$\sigma(\mathbf{z})_l = \frac{exp(z_l)}{\sum_{j=1}^{L} exp(z_j)} = \frac{exp(z_l - z_{max})}{\sum_{j=1}^{L} exp(z_j - z_{max})}$$

$$z_{max} = max([z_1, z_2, \cdots, z_L])$$

Softmax Function Properties

$$\text{Softmax output} = p(y = l|x) = \sigma(\mathbf{z})_l = \frac{exp(z_l)}{\sum_{j=1}^{L} exp(z_j)} = \frac{exp(\overbrace{\theta_l^T x}^{z_l})}{\sum_{j=1}^{L} exp(\underbrace{\theta_j^T x}_{z_j})}$$

- Invariance under Translation but not Scaling

$$\sigma(\mathbf{z} + \mathbf{c}) = \sigma(\mathbf{z})$$

$$\sigma(\mathbf{c} * \mathbf{z}) \neq \sigma(\mathbf{z})$$

Softmax Function Properties

$$\text{Softmax output} = p(y = l|x) = \sigma(\mathbf{z})_l = \frac{exp(z_l)}{\sum_{j=1}^{L} exp(z_j)} = \frac{exp(\overbrace{\theta_l^T x}^{z_l})}{\sum_{j=1}^{L} exp(\underbrace{\theta_j^T x}_{z_j})}$$

- Effect of Temperature (*T*)

$$\sigma(\beta\mathbf{z})_l = \frac{exp(\beta z_l)}{\sum_{j=1}^{L} exp(\beta z_j)}; \qquad \beta = \frac{1}{T}$$

Softmax Function Properties

$$\text{Softmax output} = p(y = l|x) = \sigma(\mathbf{z})_l = \frac{exp(z_l)}{\sum_{j=1}^{L} exp(z_j)} = \frac{exp(\overbrace{\theta_l^T x}^{z_l})}{\sum_{j=1}^{L} exp(\underbrace{\theta_j^T x}_{z_j})}$$

- Effect of Temperature (*T*)

$$\sigma(\beta\mathbf{z})_l = \frac{exp(\beta z_l)}{\sum_{j=1}^{L} exp(\beta z_j)}; \qquad \beta = \frac{1}{T}$$

$$\text{What happens when } \beta \to 0 \text{ (or } T \to \infty)?$$

Softmax Function Properties

$$\text{Softmax output} = p(y = l|x) = \sigma(\mathbf{z})_l = \frac{exp(z_l)}{\sum_{j=1}^{L} exp(z_j)} = \frac{exp(\overbrace{\theta_l^T x}^{z_l})}{\sum_{j=1}^{L} exp(\underbrace{\theta_j^T x}_{z_j})}$$

- Effect of Temperature (*T*)

$$\sigma(\beta\mathbf{z})_l = \frac{exp(\beta z_l)}{\sum_{j=1}^{L} exp(\beta z_j)}; \qquad \beta = \frac{1}{T}$$

$$\text{As, } \beta \to 0 \text{ (or } T \to \infty), \sigma(\mathbf{z}) \text{ spreads out}$$

Softmax Function Properties

$$\text{Softmax output} = p(y = l|x) = \sigma(\mathbf{z})_l = \frac{exp(z_l)}{\sum_{j=1}^{L} exp(z_j)} = \frac{exp(\overbrace{\theta_l^T x}^{z_l})}{\sum_{j=1}^{L} exp(\underbrace{\theta_j^T x}_{z_j})}$$

- Effect of Temperature (*T*)

$$\sigma(\beta\mathbf{z})_l = \frac{exp(\beta z_l)}{\sum_{j=1}^{L} exp(\beta z_j)}; \qquad \beta = \frac{1}{T}$$

$$\text{As, } \beta \to 0 \text{ (or } T \to \infty), \sigma(\mathbf{z}) \text{ spreads out}$$

$$\text{As, } \beta \to \infty \text{ (or } T \to 0), \sigma(\mathbf{z}) \text{ becomes peaky}$$

Softmax Function Properties

$$\text{Softmax output} = p(y = l|x) = \sigma(\mathbf{z})_l = \frac{exp(z_l)}{\sum_{j=1}^{L} exp(z_j)} = \frac{exp(\overbrace{\theta_l^T x}^{z_l})}{\sum_{j=1}^{L} exp(\underbrace{\theta_j^T x}_{z_j})}$$

- Approximation of Argmax Function

$$\lim_{\beta \to \infty} \sigma(\beta \mathbf{z}) = ?$$

Softmax Function Properties

$$\text{Softmax output} = p(y = l|x) = \sigma(\mathbf{z})_l = \frac{exp(z_l)}{\sum_{j=1}^{L} exp(z_j)} = \frac{exp(\overbrace{\theta_l^T x}^{z_l})}{\sum_{j=1}^{L} exp(\underbrace{\theta_j^T x}_{z_j})}$$

- Approximation of Argmax Function

$$\lim_{\beta \to \infty} \sigma(\beta \mathbf{z}) = \text{argmax}\left([z_1, z_2, \cdots, z_L]\right)$$

$$= [0, 0, \cdots, \underbrace{1}_{\substack{\text{index of} \\ \text{max value}}}, 0, \cdots, 0]$$

Softmax Function Properties

$$\text{Softmax output} = p(y = l|x) = \sigma(\mathbf{z})_l = \frac{exp(z_l)}{\sum_{j=1}^{L} exp(z_j)} = \frac{exp(\overbrace{\theta_l^T x}^{z_l})}{\sum_{j=1}^{L} exp(\underbrace{\theta_j^T x}_{z_j})}$$

- Approximation of Argmax Function

$$\lim_{\beta \to \infty} \sigma(\beta \mathbf{z}) = \text{argmax}\left([z_1, z_2, \cdots, z_L]\right)$$

$$= [0, 0, \cdots, \underbrace{1}_{\substack{\text{index of} \\ \text{max value}}}, 0, \cdots, 0]$$

Hence, the name Softmax (though misleading, rather should be Softargmax)

Softmax Function Properties

$$\text{Softmax output} = p(y = l|x) = \sigma(\mathbf{z})_l = \frac{exp(z_l)}{\sum_{j=1}^{L} exp(z_j)} = \frac{exp(\overbrace{\theta_l^T x}^{z_l})}{\sum_{j=1}^{L} exp(\underbrace{\theta_j^T x}_{z_j})}$$

- Decision Boundary Of Softmax

Softmax Function Properties

$$\text{Softmax output} = p(y = l|x) = \sigma(\mathbf{z})_l = \frac{exp(z_l)}{\sum_{j=1}^{L} exp(z_j)} = \frac{exp(\overbrace{\theta_l^T x}^{z_l})}{\sum_{j=1}^{L} exp(\underbrace{\theta_j^T x}_{z_j})}$$

- Decision Boundary Of Softmax
  - Linear Decision Boundary

Softmax Function Properties

$$\text{Softmax output} = p(y = l|x) = \sigma(\mathbf{z})_l = \frac{exp(z_l)}{\sum_{j=1}^{L} exp(z_j)} = \frac{exp(\overbrace{\theta_l^T x}^{z_l})}{\sum_{j=1}^{L} exp(\underbrace{\theta_j^T x}_{z_j})}$$

- Decision Boundary Of Softmax
  - Linear Decision Boundary
  - Creates Voronoi partitions of the space

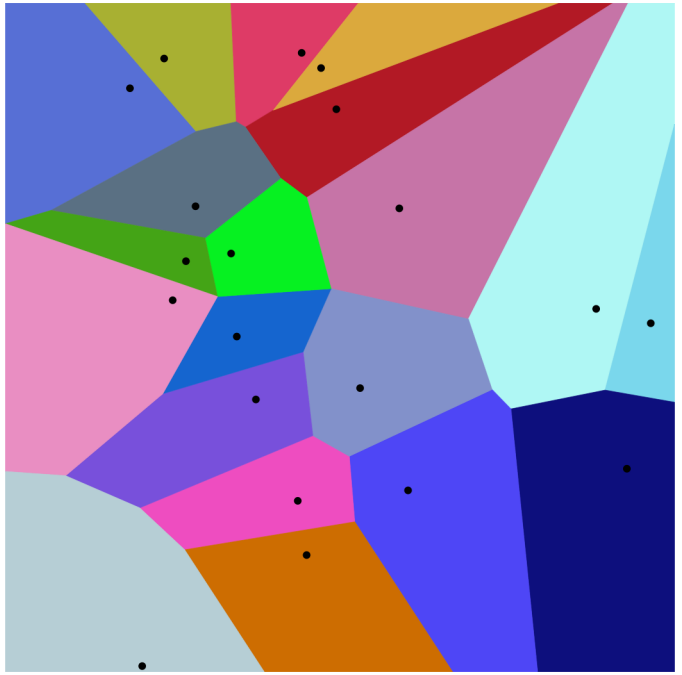


Image: Wikipedia

Softmax Function Properties

$$\text{Softmax output} = p(y = l|x) = \sigma(\mathbf{z})_l = \frac{exp(z_l)}{\sum_{j=1}^{L} exp(z_j)} = \frac{exp(\overbrace{\theta_l^T x}^{z_l})}{\sum_{j=1}^{L} exp(\underbrace{\theta_j^T x}_{z_j})}$$

- Decision Boundary Of Softmax
  - Linear Decision Boundary
  - Creates Voronoi partitions of the space
  - If the classes are mutually exclusive use softmax classifier else use *L* binary classifiers

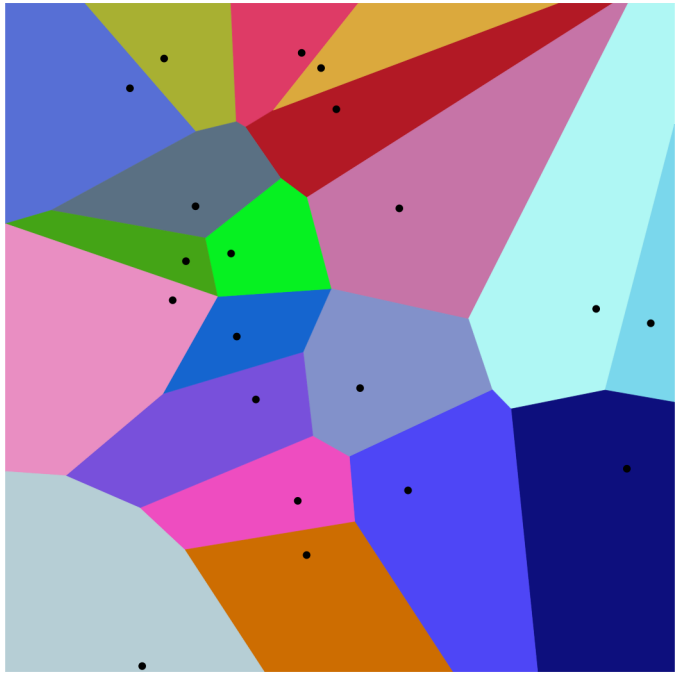


Image: Wikipedia

Softmax Loss Function

$$\text{Softmax output} = p(y = l|x) = \sigma(\mathbf{z})_l = \frac{exp(z_l)}{\sum_{j=1}^{L} exp(z_j)} = \frac{exp(\overbrace{\theta_l^T x}^{z_l})}{\sum_{j=1}^{L} exp(\underbrace{\theta_j^T x}_{z_j})}$$

- Cross Entropy Loss for Softmax

$$\mathcal{CE}_{\mathcal{L}oss}(\Theta) = \mathcal{L}_\Theta = -\sum_{l=1}^{L} \delta_{y=l} \ \log\left(p(y = l \mid x; \Theta)\right)$$

$$= -\sum_{l=1}^{L} \underbrace{y_l}_{\substack{\text{one-hot} \\ \text{vector} \\ \text{element}}} \log\left(p(y = l \mid x; \Theta)\right)$$

Softmax Loss Function

$$\text{Softmax output} = p(y = l|x) = \sigma(\mathbf{z})_l = \frac{exp(z_l)}{\sum_{j=1}^{L} exp(z_j)} = \frac{exp(\overbrace{\theta_l^T x}^{z_l})}{\sum_{j=1}^{L} exp(\underbrace{\theta_j^T x}_{z_j})}$$

- Cross Entropy Loss for Softmax

$$\mathcal{L}_\Theta = -\sum_{l=1}^{L} y_l \ \log\left(p(y = l \mid x; \Theta)\right)$$

$$\frac{d\mathcal{L}_\Theta}{d\theta_l} = (p_l - y_l) * x \qquad \textbf{(8.11)}$$

Large Vocabulary: Class Based Speedup

$$p(y \mid x) = p(y \mid x, c) \times p(c \mid x)$$

- Divide the words into classes
- Each word belongs to unique class

$$\text{We have speedup of } \; O(\sqrt{\mathcal{V}})$$

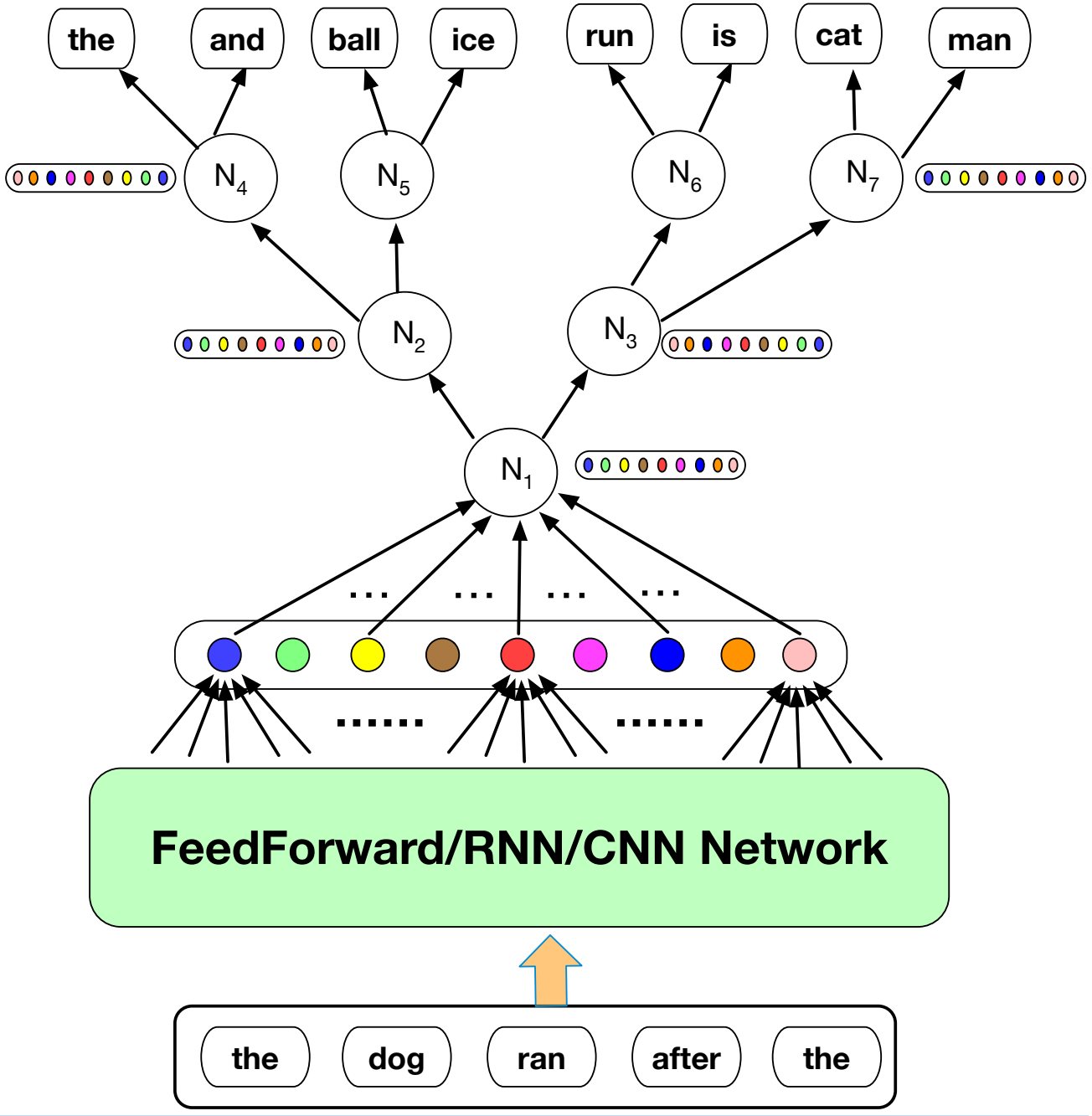
Hierarchical Class Based Speedup

$$p(y \mid x) = p(y \mid c_{l_2}, x) \times p(c_{l_2} \mid c_{l_1}, x) \times p(c_{l_1} \mid x) \quad \textbf{(7.6)}$$

- Instead of single class, one could have hierarchy of classes: each class is further subdivided into more smaller classes and so on.
- This is the key idea behind *Hierarchical Softmax*
- This helps in further speedup.

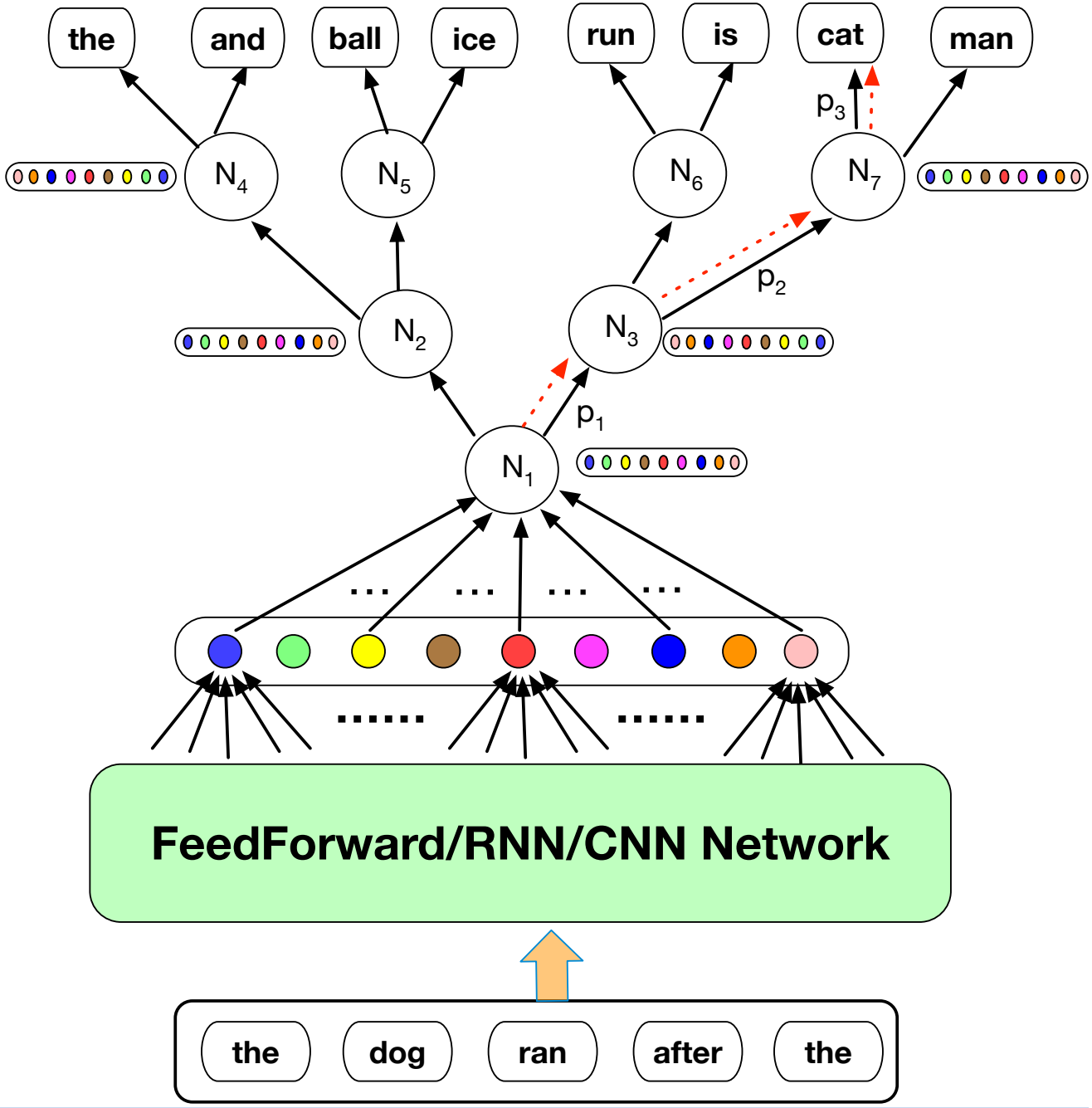
Hierarchical Softmax

- Divide the vocabulary hierarchically e.g. in form of a binary tree
- Each node is associated with a vector representation (embedding)



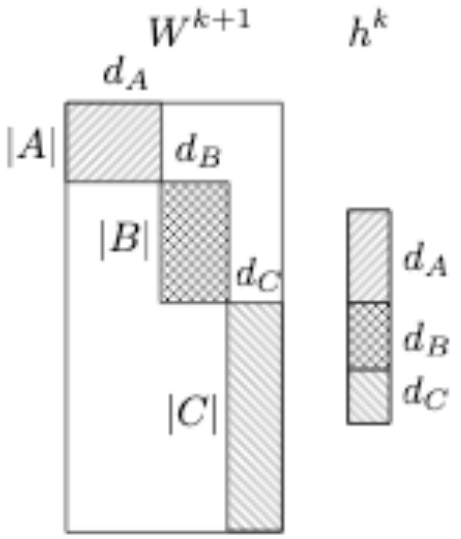
Hierarchical Softmax

- Divide the vocabulary hierarchically e.g. in form of a binary tree
- Each node is associated with a vector representation (embedding)
- At each step make a binary decision (Logistic Regression) about which node to go next



Other Speedup Strategies

- **Differentiated Softmax**:
  - Assign different sized weight vector to each word based on frequency



Strategies for Training Large Vocabulary Neural Language Models : https://www.aclweb.org/anthology/P16-1186.pdf

Other Speedup Strategies

- **Noise Contrastive Estimation (Negative Sampling)**
  - Distinguish true training data from noise

$$p(w \mid x) = \frac{1}{k+1} p_{train}(w \mid x) + \frac{k}{k+1} p_{noise}(w \mid x)$$

Strategies for Training Large Vocabulary Neural Language Models : https://www.aclweb.org/anthology/P16-1186.pdf

Other Speedup Strategies

- **Noise Contrastive Estimation (Negative Sampling)**
  - Distinguish true training data from noise

$$p(w \mid x) = \frac{1}{k+1} p_{train}(w \mid x) + \frac{k}{k+1} p_{noise}(w \mid x)$$

Cross Entropy Loss

$$-y \log \hat{p}(y = 1|w, x) - (1 - y) \log \hat{p}(y = 0 \mid w, x)$$

y is an Indicator Variable

$$\begin{cases} \hat{p}(y = 1|w, x) & = \frac{\hat{p}(w|x)}{\hat{p}(w|x) + k p_{\text{noise}}(w|x)} \\ \hat{p}(y = 0|w, x) & = 1 - \hat{p}(y = 1|w, x) \end{cases}$$

Summary

- Information can be measured in terms of entropy
- Cross-entropy measures the distance between true and predicted distributions
- Log Linear classifiers are Max Entropy Classifiers
- Softmax function is one of the most important activation function in modern NNs
- There are ways to overcome very large vocabulary size limitations

References

1. Chapter 6, Section 6.6, Speech and Language Processing, Dan Jurafsky and James Martin
2. Appendix C, Linguistic Structure Prediction, Noah Smith, Morgan Claypool
3. Strategies for Training Large Vocabulary Neural Language Models : https://www.aclweb.org/anthology/P16-1186.pdf
4. Notes on Backpropagation: https://www.ics.uci.edu/~pjsadows/notes.pdf

- Next class
  - Neural Language Models