Special Topics in Natural Language Processing

CS698O

Ashutosh Modi
CSE Department, IIT Kanpur

Lecture 7: Language Models 4
Jan 17, 2020

Language Models: Extending Context

- Larger the context the better

Language Models: Extending Context

- Larger the context the better
- In principle we could have larger window in n-gram, but this is computationally problematic

Language Models: Extending Context

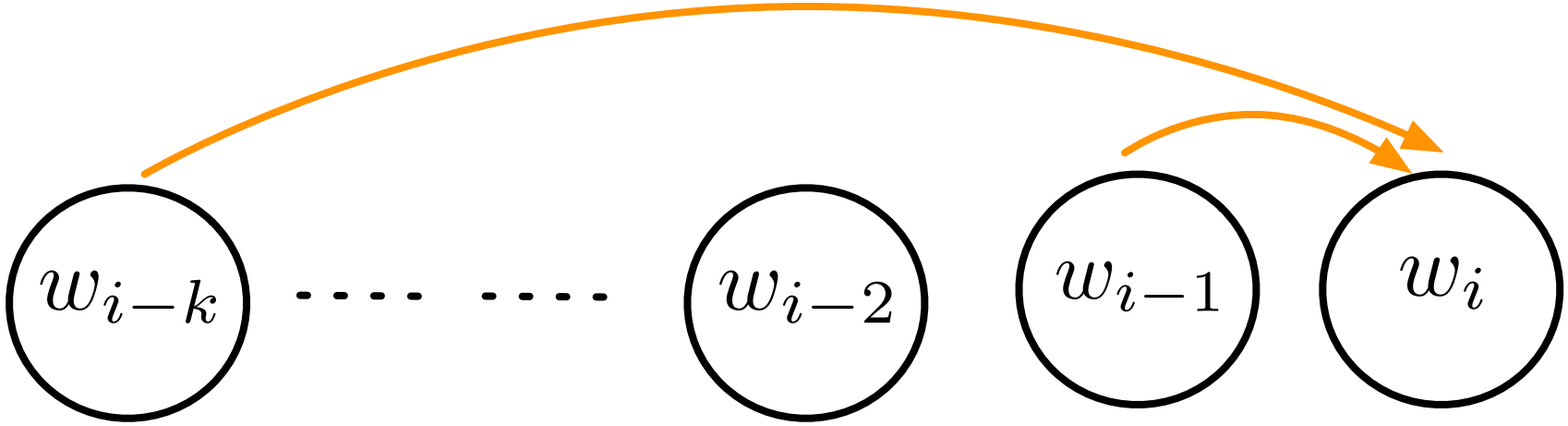
- Larger the context the better
- In principle we could have larger window in n-gram, but this is computationally problematic
- One alternative: **Skip-k-Grams Language Model**

$$P(w_i|w_{i-1}, w_{i-k})$$

Language Models: Extending Context

- **Skip-k-Grams LM**

$$P(w_i|w_{i-1}, w_{i-k})$$



Language Models: Extending Context

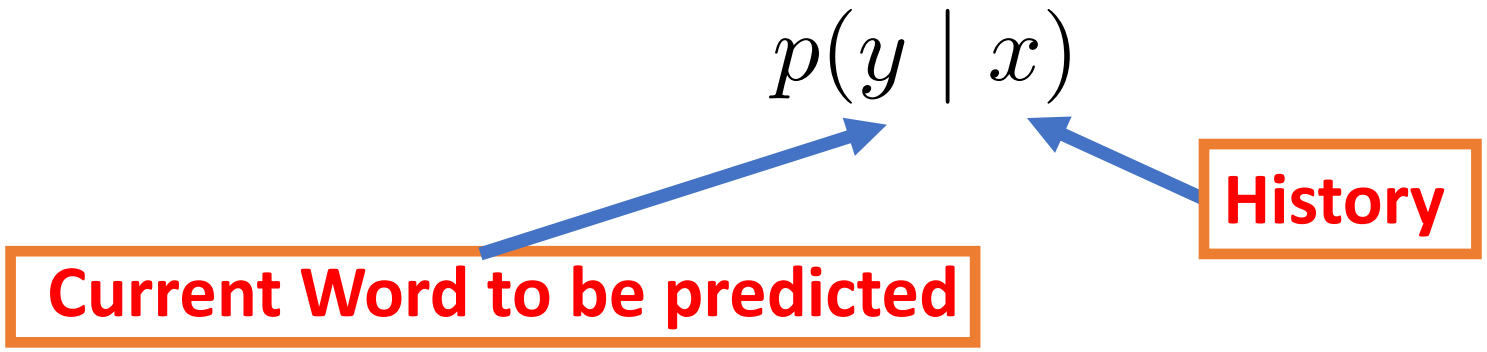
- Larger the context the better
- In principle we could have larger window in n-gram, but this is computationally problematic
- Other than n-grams is there any other way of extending context and making Language model richer and explicitly more expressive?

Language Models: Extending Context

- What about select features from the entire history and then make prediction?

Language Models: Extending Context

- What about select features from the entire history and then make prediction?



Language Models: Extending Context

- What about select features from the entire history and then make prediction?

$$p(y \mid x) = \mathcal{F}(x, y)$$

Language Models: Extending Context

- What about select features from the entire history and then make prediction?

$$p(y \mid x) = \mathcal{F}(x, y)$$

$$\mathcal{F} : \ (x, y) \rightarrow [0, 1]$$

Language Models: Extending Context

- What about select features from the entire history and then make prediction?

$$p(y \mid x) = \mathcal{F}(x, y)$$

$$\mathcal{F} : \ (x, y) \rightarrow [0, 1]$$

$$\sum_{y'} \mathcal{F}(x, y') = 1$$

Language Models: Extending Context

$$p(y \mid x) = \mathcal{F}(x, y)$$

$$\mathcal{F} : \ (x, y) \rightarrow [0, 1]$$

$$\sum_{y'} \mathcal{F}(x, y') = 1$$

$$\phi(x, y)$$

Feature Vector

Language Models: Extending Context

$$p(y \mid x) = \mathcal{F}(x, y)$$

$$\mathcal{F} : \ (x, y) \rightarrow [0, 1]$$

$$\sum_{y'} \mathcal{F}(x, y') = 1$$

$$\phi(x, y) \in \mathbb{R}^d$$

Feature Vector

Language Models: Extending Context

$$p(y \mid x) = \mathcal{F}(x, y)$$

$$\mathcal{F}: \ (x, y) \rightarrow [0, 1]$$

$$\sum_{y'} \mathcal{F}(x, y') = 1$$

$$\phi(x, y) \in \mathbb{R}^d$$

$$\phi() : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}^d$$

Language Models: Extending Context

$$\phi(x, y) \in \mathbb{R}^d$$

$$\phi() : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}^d$$

$$p(y \mid x) = \mathcal{F}(\phi(x, y))$$

$$\mathcal{F} : \ \phi(x, y) \rightarrow [0, 1]$$

$$\sum_{y'} \mathcal{F}(\phi(x, y^{'})) = 1$$

Log-Linear Models

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

Log-Linear Models

$$p(y \mid x; w) = \frac{exp(w^T \phi(x, y))}{\sum_{y' \in \mathcal{Y}} exp(w^T \phi(x, y'))} \quad \textbf{(7.1)}$$

Probability of y conditioned on x parametrized by weights w

Log-Linear Models

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

(Input) $\in \mathcal{X}$

$\in \mathcal{V}$ **(Vocabulary)**

Log-Linear Models

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

(Input) $\in \mathcal{X}$

(Set of possible inputs)

$\in \mathcal{V}$ **(Vocabulary)**

Log-Linear Models

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

(Input) $\in \mathcal{X}$

(Set of possible inputs)

$\in \mathcal{V}$ **(Vocabulary)**

$\phi() : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}^d$ **(Feature Vector)**

Log-Linear Models

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

(Weight\ Parameter Vector) $\in \mathbb{R}^d$

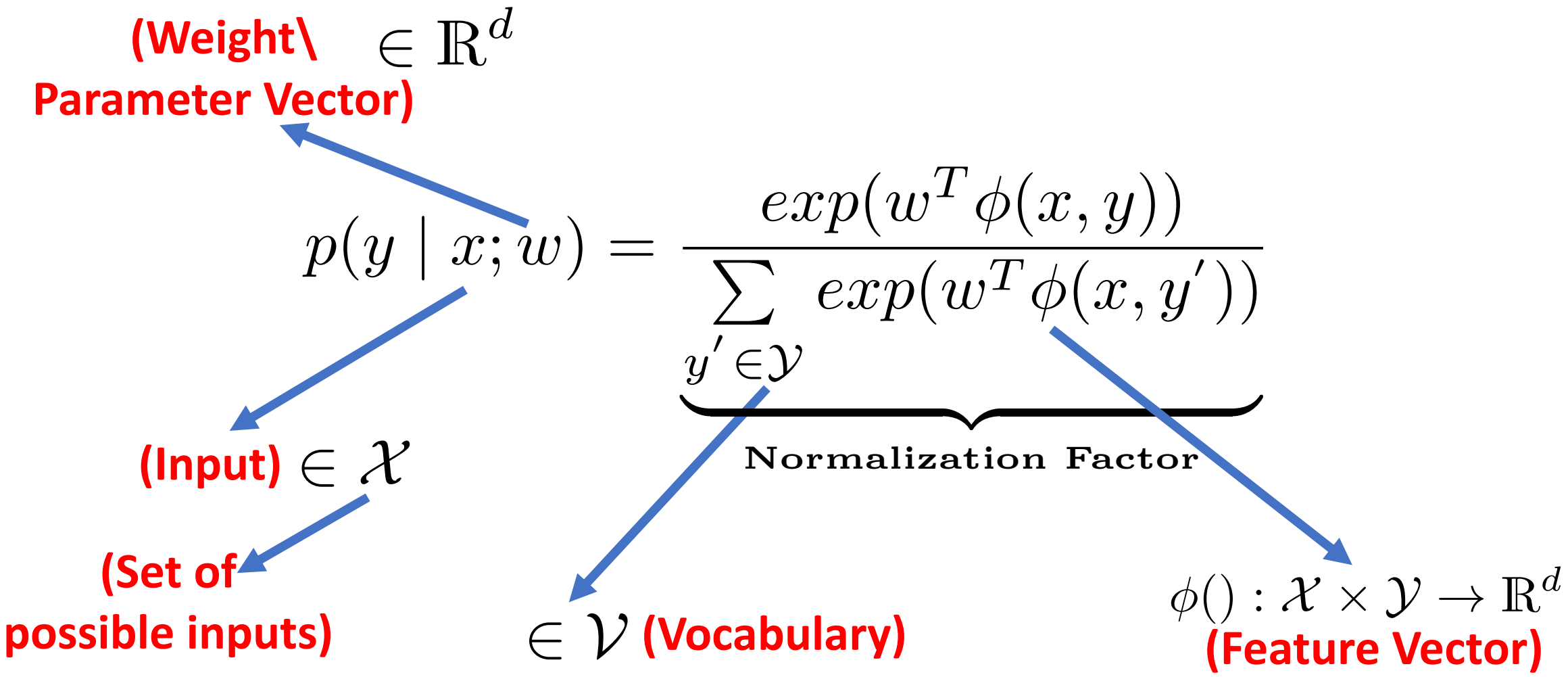
(Input) $\in \mathcal{X}$

(Set of possible inputs)

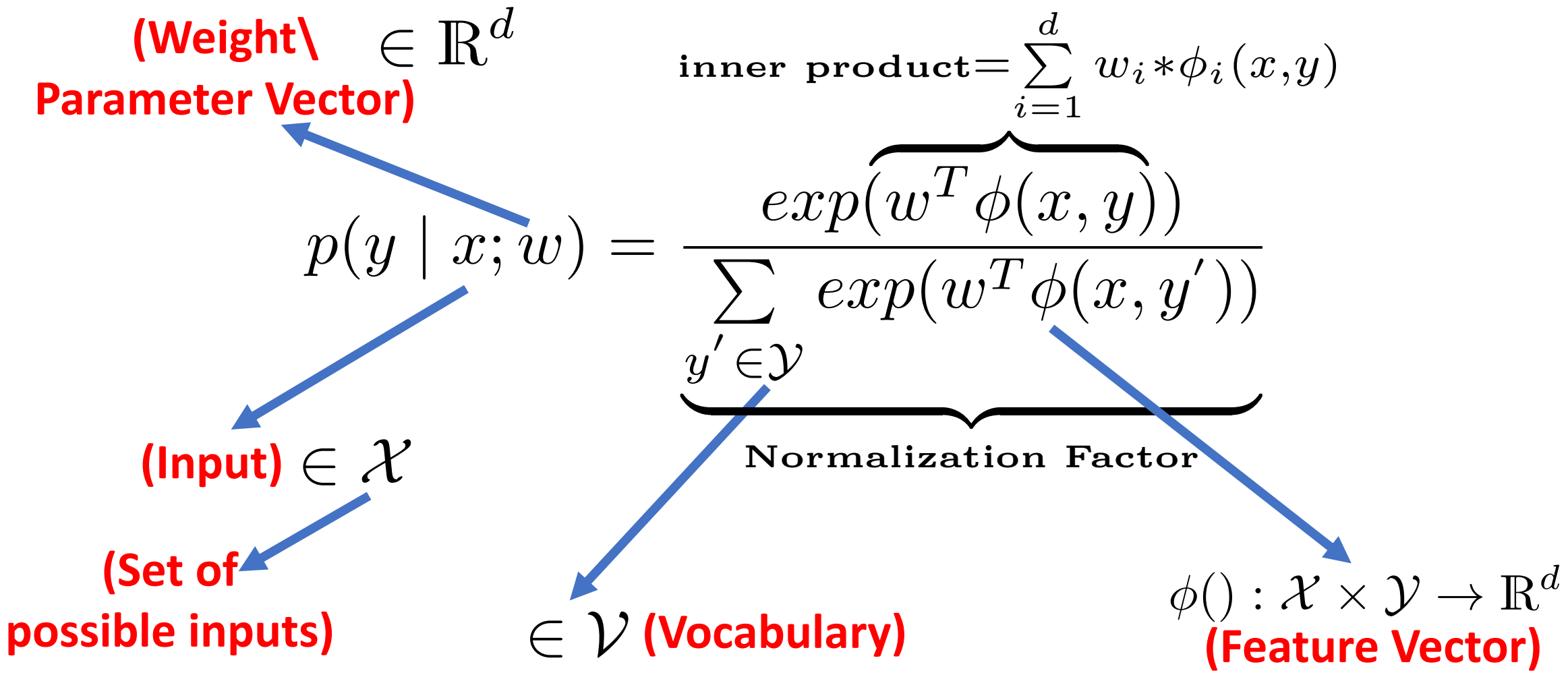
$\in \mathcal{V}$ (Vocabulary)

$\phi() : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}^d$ (Feature Vector)

Log-Linear Models



Log-Linear Models



Log-Linear Models

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

- This probability is calculated for each word in the vocabulary

Log-Linear Models

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

- This probability is calculated for each word in the vocabulary
- This probability depends on the inner product

Log-Linear Models

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

- This probability is calculated for each word in the vocabulary
- This probability depends on the inner product
- The inner product can take any real value

Log-Linear Models

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

- This probability is calculated for each word in the vocabulary
- This probability depends on the inner product
- The inner product can take any real value
- If inner product is high, then that particular word has high probability

Why name Log-Linear Models?

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

Why name Log-Linear Models?

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

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

Why name Log-Linear Models?

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

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

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

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

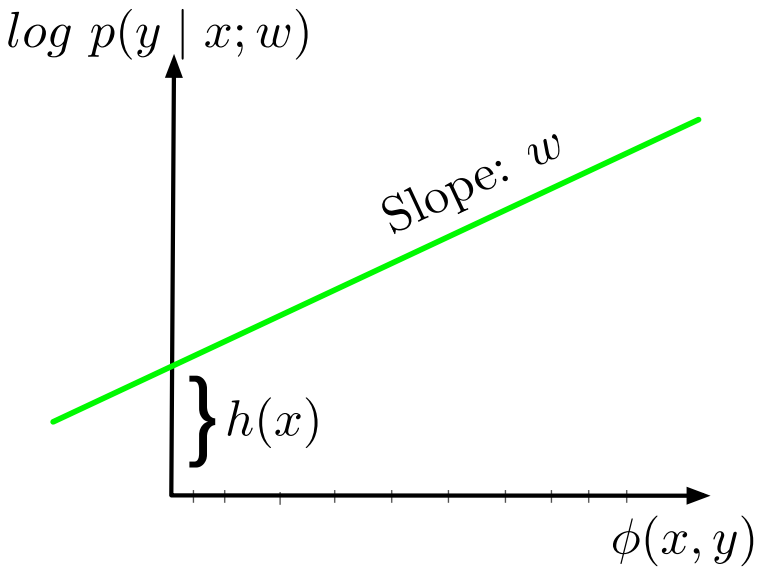
$$log\ p(y \mid x; w) = w^T \phi(x, y) + h(x)$$

Why name Log-Linear Models?

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

$$log\ p(y \mid x; w) = w^T \phi(x, y) + h(x) \quad \textbf{(7.2)}$$

$$= -log \left(\sum_{y' \in \mathcal{Y}} exp(w^T \phi(x, y')) \right)$$



Feature Vector

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

$$\phi() : \mathcal{X} \times \mathcal{Y} \rightarrow \{0, 1\}^d$$

$$\phi(x, y) = \begin{bmatrix} \phi_1(x, y) \\ \phi_2(x, y) \\ \vdots \\ \phi_k(x, y) \\ \vdots \\ \phi_d(x, y) \end{bmatrix}$$

Feature Vector Example

$$\phi() : \mathcal{X} \times \mathcal{Y} \rightarrow \{0, 1\}^d$$

Under the guidance of economist John Kenneth Galbraith, IIT Kanpur was the first institute in India to offer Computer science education. The earliest computer courses were started at IIT Kanpur in August 1963 on an IBM _____

Feature Vector Example

$$\phi() : \mathcal{X} \times \mathcal{Y} \rightarrow \{0,1\}^d$$

Under the guidance of economist John Kenneth Galbraith, IIT Kanpur was the first institute in India to offer Computer science education. The earliest computer courses were started at IIT Kanpur in August 1963 on an IBM **system**

Feature Vector Example

Under the guidance of economist John Kenneth Galbraith, IIT Kanpur was the first institute in India to offer Computer science education. The earliest computer courses were started at IIT Kanpur in August 1963 on an IBM **system**

$$\phi_1(x, y) = \begin{cases} 1 & \text{If } y = \texttt{system} \\ 0 & \text{otherwise} \end{cases}$$

Unigram Feature

Feature Vector Example

Under the guidance of economist John Kenneth Galbraith, IIT Kanpur was the first institute in India to offer Computer science education. The earliest computer courses were started at IIT Kanpur in August 1963 on an IBM **system**

$$\phi_1(x, y) = \begin{cases} 1 & \text{If } y = \texttt{system} \\ 0 & \text{otherwise} \end{cases}$$

$$\phi_2(x, y) = \begin{cases} 1 & \text{If } y = \texttt{system and } w_{i-1} = \texttt{IBM} \\ 0 & \text{otherwise} \end{cases}$$

Bigram Feature

Feature Vector Example

Under the guidance of economist John Kenneth Galbraith, IIT Kanpur was the first institute in India to offer Computer science education. The earliest computer courses were started at IIT Kanpur in August 1963 on an IBM **system**

$$\phi_1(x,y) = \begin{cases} 1 & \text{If } y = \texttt{system} \\ 0 & \text{otherwise} \end{cases} \qquad \phi_2(x,y) = \begin{cases} 1 & \text{If } y = \texttt{system} \text{ and } w_{i-1} = \texttt{IBM} \\ 0 & \text{otherwise} \end{cases}$$

$$\phi_3(x,y) = \begin{cases} 1 & \text{If } y = \texttt{system} \text{ and } w_{i-1} = \texttt{IBM}, w_{i-2} = \texttt{an} \\ 0 & \text{otherwise} \end{cases}$$

Trigram Feature

Feature Vector Example

Under the guidance of economist John Kenneth Galbraith, IIT Kanpur was the first institute in India to offer Computer science education. The earliest computer courses were started at IIT Kanpur in August 1963 on an IBM **system**

$$\phi_1(x,y) = \begin{cases} 1 & \text{If } y = \texttt{system} \\ 0 & \text{otherwise} \end{cases} \qquad \phi_2(x,y) = \begin{cases} 1 & \text{If } y = \texttt{system and } w_{i-1} = \texttt{IBM} \\ 0 & \text{otherwise} \end{cases}$$

$$\phi_3(x,y) = \begin{cases} 1 & \text{If } y = \texttt{system and } w_{i-1} = \texttt{IBM}, w_{i-2} = \texttt{an} \\ 0 & \text{otherwise} \end{cases}$$

$$\phi_4(x,y) = \begin{cases} 1 & \text{If } y = \texttt{system and } w_{i-1} \in COMPANY \\ 0 & \text{otherwise} \end{cases}$$

Class Feature

Feature Vector Example

Under the guidance of economist John Kenneth Galbraith, IIT Kanpur was the first institute in India to offer Computer science education. The earliest computer courses were started at IIT Kanpur in August 1963 on an IBM **system**

$$\phi_1(x,y) = \begin{cases} 1 & \text{If } y = \texttt{system} \\ 0 & \text{otherwise} \end{cases} \qquad \phi_2(x,y) = \begin{cases} 1 & \text{If } y = \texttt{system} \text{ and } w_{i-1} = \texttt{IBM} \\ 0 & \text{otherwise} \end{cases}$$

$$\phi_3(x,y) = \begin{cases} 1 & \text{If } y = \texttt{system} \text{ and } w_{i-1} = \texttt{IBM}, w_{i-2} = \texttt{an} \\ 0 & \text{otherwise} \end{cases}$$

$$\phi_4(x,y) = \begin{cases} 1 & \text{If } y = \texttt{system} \text{ and } w_{i-1} \in COMPANY \\ 0 & \text{otherwise} \end{cases}$$

$$\phi_5(x,y) = \begin{cases} 1 & \text{If } y = \texttt{system} \text{ and } POS(w_{i-1}) = NOUN \\ 0 & \text{otherwise} \end{cases}$$

Syntactic Feature

Feature Vector Example

Under the guidance of economist John Kenneth Galbraith, IIT Kanpur was the first institute in India to offer Computer science education. The earliest computer courses were started at IIT Kanpur in August 1963 on an IBM **system**

$$\phi_1(x,y) = \begin{cases} 1 & \text{If } y = \texttt{system} \\ 0 & \text{otherwise} \end{cases} \qquad \phi_2(x,y) = \begin{cases} 1 & \text{If } y = \texttt{system and } w_{i-1} = \texttt{IBM} \\ 0 & \text{otherwise} \end{cases}$$

$$\phi_3(x,y) = \begin{cases} 1 & \text{If } y = \texttt{system and } w_{i-1} = \texttt{IBM}, w_{i-2} = \texttt{an} \\ 0 & \text{otherwise} \end{cases}$$

$$\phi_4(x,y) = \begin{cases} 1 & \text{If } y = \texttt{system and } w_{i-1} \in COMPANY \\ 0 & \text{otherwise} \end{cases}$$

$$\phi_5(x,y) = \begin{cases} 1 & \text{If } y = \texttt{system and } POS(w_{i-1}) = NOUN \\ 0 & \text{otherwise} \end{cases}$$

$$\phi_6(x,y) = \begin{cases} 1 & \text{If } y = \texttt{system, computer} \in \{w_1, \ldots, w_{i-1}\} \\ 0 & \text{otherwise} \end{cases}$$

Document Feature

Feature Vector Example

Under the guidance of economist John Kenneth Galbraith, IIT Kanpur was the first institute in India to offer Computer science education. The earliest computer courses were started at IIT Kanpur in August 1963 on an IBM **system**

$$\phi_1(x,y) = \begin{cases} 1 & \text{If } y = \texttt{system} \\ 0 & \text{otherwise} \end{cases} \qquad \phi_2(x,y) = \begin{cases} 1 & \text{If } y = \texttt{system and } w_{i-1} = \texttt{IBM} \\ 0 & \text{otherwise} \end{cases}$$

$$\phi_3(x,y) = \begin{cases} 1 & \text{If } y = \texttt{system and } w_{i-1} = \texttt{IBM}, w_{i-2} = \texttt{an} \\ 0 & \text{otherwise} \end{cases}$$

$$\phi_4(x,y) = \begin{cases} 1 & \text{If } y = \texttt{system and } w_{i-1} \in COMPANY \\ 0 & \text{otherwise} \end{cases}$$

$$\phi_5(x,y) = \begin{cases} 1 & \text{If } y = \texttt{system and } POS(w_{i-1}) = NOUN \\ 0 & \text{otherwise} \end{cases}$$

$$\phi_6(x,y) = \begin{cases} 1 & \text{If } y = \texttt{system, computer} \in \{w_1, \ldots, w_{i-1}\} \\ 0 & \text{otherwise} \end{cases}$$

$$\phi_7(x,y) = \begin{cases} 1 & \text{If } y = \texttt{system, chemistry} \notin \{w_1, \ldots, w_{i-1}\} \\ 0 & \text{otherwise} \end{cases}$$

Why not Linear Interpolated LM?

$$
\begin{aligned}
p(\texttt{system} \mid w_{i-1}, \ldots, w_1) = \; & \lambda_1 \times p(\texttt{system} \mid w_{i-1} = \texttt{IBM}, \texttt{an}) + \\
& \lambda_2 \times p(\texttt{system} \mid w_{i-1} = \texttt{IBM}) + \\
& \lambda_3 \times p(\texttt{system}) + \\
& \lambda_4 \times p(\texttt{system} \mid w_{i-1} \in COMPANY) + \\
& \lambda_5 \times p(\texttt{system} \mid POS(w_{i-1}) = NOUN) + \\
& \lambda_6 \times p(\texttt{system} \mid \texttt{computer} \in \{w_{i-1}, \ldots, w_1\}) + \\
& \lambda_7 \times p(\texttt{system} \mid \texttt{chemistry} \notin \{w_{i-1}, \ldots, w_1\}) + \\
& \vdots
\end{aligned}
$$

Why not Linear Interpolated LM?

$$
\begin{aligned}
p(\texttt{system} \mid w_{i-1}, \ldots, w_1) = {} & \lambda_1 \times p(\texttt{system} \mid w_{i-1} = \texttt{IBM}, \texttt{an}) + \\
& \lambda_2 \times p(\texttt{system} \mid w_{i-1} = \texttt{IBM}) + \\
& \lambda_3 \times p(\texttt{system}) + \\
& \lambda_4 \times p(\texttt{system} \mid w_{i-1} \in COMPANY) + \\
& \lambda_5 \times p(\texttt{system} \mid POS(w_{i-1}) = NOUN) + \\
& \lambda_6 \times p(\texttt{system} \mid \texttt{computer} \in \{w_{i-1}, \ldots, w_1\}) + \\
& \lambda_7 \times p(\texttt{system} \mid \texttt{chemistry} \notin \{w_{i-1}, \ldots, w_1\}) + \\
& \vdots
\end{aligned}
$$

As number of features increase,
linear interpolation becomes extremely challenging

Feature Templates

$$\phi_1(x, y) = \begin{cases} 1 & \text{If } y = \texttt{system} \\ 0 & \text{otherwise} \end{cases}$$

$$\phi_2(x, y) = \begin{cases} 1 & \text{If } y = \texttt{system} \text{ and } w_{i-1} = \texttt{IBM} \\ 0 & \text{otherwise} \end{cases}$$

$$\phi_3(x, y) = \begin{cases} 1 & \text{If } y = \texttt{system} \text{ and } w_{i-1} = \texttt{IBM}, w_{i-2} = \texttt{an} \\ 0 & \text{otherwise} \end{cases}$$

Can we define generic feature function which work across all unigrams, bigrams, or trigrams ?

Trigram Feature Template

$$\phi_{\mathcal{N}(q,r,s)}(x,y) = \begin{cases} 1 & \text{If } y = \texttt{q},\ w_{i-1} = \texttt{r},\ w_{i-2} = \texttt{s} \\ 0 & \text{otherwise} \end{cases}$$

$$\mathcal{N}() : \underbrace{(q,r,s)}_{trigram} \rightarrow \underbrace{\mathbb{N}}_{Integers}$$

- Template generates features only for trigrams seen during training

Trigram Feature Template

$$\phi_{\mathcal{N}(q,r,s)}(x,y) = \begin{cases} 1 & \text{If } y = \mathtt{q},\ w_{i-1} = \mathtt{r},\ w_{i-2} = \mathtt{s} \\ 0 & \text{otherwise} \end{cases}$$

$$\mathcal{N}() : \underbrace{(q,r,s)}_{trigram} \rightarrow \underbrace{\mathbb{N}}_{Integers}$$

$$\text{For, } q \neq w, v \neq r, s \neq u$$

$$\mathcal{N}(q,r,s) \neq \mathcal{N}(w,v,u)$$

Bigram Feature Template

$$\phi_{\mathcal{N}(q,r,s)}(x,y) = \begin{cases} 1 & \text{If } y = \mathtt{q},\ w_{i-1} = \mathtt{r},\ w_{i-2} = \mathtt{s} \\ 0 & \text{otherwise} \end{cases}$$

$$\phi_{\mathcal{N}(q,r)}(x,y) = \begin{cases} 1 & \text{If } y = \mathtt{q},\ \ w_{i-1} = \mathtt{r} \\ 0 & \text{otherwise} \end{cases}$$

Bigram Feature Template

$$\phi_{\mathcal{N}(q,r,s)}(x,y) = \begin{cases} 1 & \text{If } y = \texttt{q},\ w_{i-1} = \texttt{r},\ w_{i-2} = \texttt{s} \\ 0 & \text{otherwise} \end{cases}$$

$$\phi_{\mathcal{N}(q,r)}(x,y) = \begin{cases} 1 & \text{If } y = \texttt{q},\ \ w_{i-1} = \texttt{r} \\ 0 & \text{otherwise} \end{cases}$$

$$\phi_{\mathcal{N}(q)}(x,y) = \begin{cases} 1 & \text{If } y = \texttt{q} \\ 0 & \text{otherwise} \end{cases}$$

Other Feature Templates

$$\phi_{\mathcal{N}(q,r,s)}(x,y) = \begin{cases} 1 & \text{If } y = \texttt{q},\ w_{i-1} = \texttt{r},\ w_{i-2} = \texttt{s} \\ 0 & \text{otherwise} \end{cases}$$

$$\phi_{\mathcal{N}(q,r)}(x,y) = \begin{cases} 1 & \text{If } y = \texttt{q},\ \ w_{i-1} = \texttt{r} \\ 0 & \text{otherwise} \end{cases}$$

$$\phi_{\mathcal{N}(q)}(x,y) = \begin{cases} 1 & \text{If } y = \texttt{q} \\ 0 & \text{otherwise} \end{cases}$$

$$\phi_{\mathcal{N}(q,POS=DET)}(x,y) = \begin{cases} 1 & \text{If } y = \texttt{q},\ \ POS(w_{i-1}) = DET \\ 0 & \text{otherwise} \end{cases}$$

Feature Sparsity

- Feature vector is bit string of 1s and 0s

$$10101010000000001000001$$

Feature Sparsity

- Feature vector is bit string of 1s and 0s

$$10101010000000001000001$$

- Number of features is large

$$d \sim \text{100s of thousands or millions}$$

Feature Sparsity

- Feature vector is bit string of 1s and 0s

$$10101010000000001000001$$

- Number of features is large

$$d \sim 100\text{s of thousands or millions}$$

- Feature vector is very sparse bit-spring.

Feature Sparsity

- Feature vector is bit string of 1s and 0s

$$10101010000000001000001$$

- Number of features is large

$$d \sim 100\text{s of thousands or millions}$$

- Feature vector is very sparse bit-spring.

- However, this is beneficial for efficient computation.

Feature Sparsity

- Feature sparsity is beneficial for efficient computation.
- We can represent feature vector by a smaller sized vector and manipulate that.

$$Z(x, y) = \{k : \phi_k(x, y) = 1\}$$

$$| Z(x, y) | << d$$

Feature Sparsity

$$Z(x, y) = \{k : \phi_k(x, y) = 1\}$$

$$w^T \phi(x, y) = \sum_{k=1}^{d} w_k \phi_k(x, y)$$

Feature Sparsity

$$Z(x,y) = \{k : \phi_k(x,y) = 1\}$$

$$w^T \phi(x,y) = \sum_{k=1}^{d} w_k \phi_k(x,y)$$

$$= \sum_{k \in Z(x,y)} w_k$$

Parameter Estimation

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

Given the training set, $(x^{(i)}, y^{(i)})\ \forall i = \{1, 2, \ldots, n\}$,
we want to estimate the parameter vector w

Parameter Estimation

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

Given the training set, $(x^{(i)}, y^{(i)})\ \forall i = \{1, 2, \ldots, n\}$, we want to estimate the parameter vector w

MLE

Parameter Estimation

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

Given the training set, $(x^{(i)}, y^{(i)})\ \forall i = \{1, 2, \ldots, n\}$, we want to estimate the parameter vector w

MLE

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

Parameter Estimation

MLE

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

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

$$w^* = \arg\max_{w \in \mathbb{R}^d} \mathcal{L}(w)$$

But MLE can lead to problems

- Features that are very infrequent would lead to very large corresponding weight
- Suppose, the trigram *(an, IBM, system)* is seen only once during training

$$\text{When, } p(\texttt{system} \mid \texttt{IBM, an}) \rightarrow 1$$

$$w_{\phi_{\mathcal{N}}(\texttt{system, IBM, an})} \rightarrow \infty$$

Regularization

$$\mathcal{L}(w) = \sum_{i=1}^{n} log\left(p(y^{(i)} \mid x^{(i)}; w)\right) - \frac{\lambda}{2} \mid\mid w \mid\mid^2 \quad \textbf{(7.3)}$$

Regularization

L2-Norm

$$w^T w = \sum_{i=1}^{d} w_i^2$$

$$\mathcal{L}(w) = \sum_{i=1}^{n} log\left(p(y^{(i)} \mid x^{(i)}; w)\right) - \frac{\lambda}{2} \mid\mid w \mid\mid^2$$

Regularization

L2-Norm

$$w^T w = \sum_{i=1}^{d} w_i^2$$

$$\mathcal{L}(w) = \sum_{i=1}^{n} log\left(p(y^{(i)} \mid x^{(i)}; w)\right) - \frac{\lambda}{2} \mid\mid w \mid\mid^2$$

log-likelihood

regularizer

Regularization

L2-Norm

$$w^T w = \sum_{i=1}^{d} w_i^2$$

$$\mathcal{L}(w) = \sum_{i=1}^{n} log\left(p(y^{(i)} \mid x^{(i)}; w)\right) - \frac{\lambda}{2} \mid\mid w \mid\mid^2$$

log-likelihood

regularizer

- Balance between two terms

Regularization

L2-Norm

$$w^T w = \sum_{i=1}^{d} w_i^2$$

$$\mathcal{L}(w) = \sum_{i=1}^{n} log\left(p(y^{(i)} \mid x^{(i)}; w)\right) - \frac{\lambda}{2} \mid\mid w \mid\mid^2$$

log-likelihood

regularizer

- Balance between two terms
- Log-likelihood terms tries to fit best to the data

Regularization

L2-Norm

$$w^T w = \sum_{i=1}^{d} w_i^2$$

$$\mathcal{L}(w) = \sum_{i=1}^{n} log\left(p(y^{(i)} \mid x^{(i)}; w)\right) - \frac{\lambda}{2} \mid\mid w \mid\mid^2$$

log-likelihood

regularizer

- Balance between two terms
- Log-likelihood terms tries to fit best to the data
- Regularizer penalizes large values

Regularization

L2-Norm

$$w^T w = \sum_{i=1}^{d} w_i^2$$

$$\mathcal{L}(w) = \sum_{i=1}^{n} log\left(p(y^{(i)} \mid x^{(i)}; w)\right) - \frac{\lambda}{2} \mid\mid w \mid\mid^2$$

log-likelihood regularizer

- Balance between two terms
- Log-likelihood terms tries to fit best to the data
- Regularizer penalizes large values
- Lambda maintains the balance between the two terms

Parameter Estimation

$$\arg\max_{w \in \mathbb{R}^d} \mathcal{L}(w)$$

Parameter Estimation

$$\arg\max_{w \in \mathbb{R}^d} \mathcal{L}(w)$$

Gradient Ascent Algorithm (Hill Climbing)

Parameter Estimation

$$\arg\max_{w \in \mathbb{R}^d} \mathcal{L}(w)$$

Gradient Ascent Algorithm (Hill Climbing)

$$\text{Initialization } w = 0$$

Parameter Estimation

$$\arg\max_{w \in \mathbb{R}^d} \mathcal{L}(w)$$

Gradient Ascent Algorithm (Hill Climbing)

$$\text{Initialization } w = 0$$

$$\text{Calculate } \delta = \nabla_w \mathcal{L}$$

Parameter Estimation

$$\arg\max_{w \in \mathbb{R}^d} \mathcal{L}(w)$$

Gradient Ascent Algorithm (Hill Climbing)

$$\text{Initialization } w = 0$$

$$\text{Calculate } \delta = \nabla_w \mathcal{L} \rightarrow \nabla_w \mathcal{L} = \begin{bmatrix} \frac{d\mathcal{L}}{dw_1} \\ \vdots \\ \frac{d\mathcal{L}}{dw_k} \\ \vdots \\ \frac{d\mathcal{L}}{dw_d} \end{bmatrix}$$

Parameter Estimation

$$\arg\max_{w \in \mathbb{R}^d} \mathcal{L}(w)$$

Gradient Ascent Algorithm (Hill Climbing)

$$\text{Initialization } w = 0$$

$$\text{Calculate } \delta = \nabla_w \mathcal{L}$$

$$\text{Perform line search: } \alpha^* = \arg\max_{\alpha \in \mathbb{R}} \mathcal{L}\left(w + \alpha\delta\right)$$

Parameter Estimation

$$\arg\max_{w \in \mathbb{R}^d} \mathcal{L}(w)$$

Gradient Ascent Algorithm (Hill Climbing)

Initialization $w = 0$

Calculate $\delta = \nabla_w \mathcal{L}$

Perform line search: $\alpha^* = \arg\max_{\alpha \in \mathbb{R}} \mathcal{L}\left(w + \alpha\delta\right)$

Update: $w = w + \alpha^* \delta$

Parameter Estimation

$$\arg\max_{w \in \mathbb{R}^d} \mathcal{L}(w)$$

Gradient Ascent Algorithm (Hill Climbing)

Initialization $w = 0$

Repeat until convergence:

Calculate $\delta = \nabla_w \mathcal{L}$

Perform line search: $\alpha^* = \arg\max_{\alpha \in \mathbb{R}} \mathcal{L}\left(w + \alpha\delta\right)$

Update: $w = w + \alpha^* \delta$

Parameter Estimation

$$\arg\max_{w \in \mathbb{R}^d} \mathcal{L}(w)$$

Gradient Ascent Algorithm (Hill Climbing)

Initialization $w = 0$

Repeat until convergence:

Calculate $\delta = \nabla_w \mathcal{L}$

Perform line search: $\alpha^* = \arg\max_{\alpha \in \mathbb{R}} \mathcal{L}(w + \alpha\delta)$

Update: $w = w + \alpha^* \delta$

Parameter Estimation

$$\arg\max_{w \in \mathbb{R}^d} \mathcal{L}(w)$$

- Since objective function is convex, gradient ascent will find the optimal solution
- In practice, gradient ascent can be slow to converge
- More sophisticated algorithms like LBFGs give faster convergence

Gradient Calculation

$$\mathcal{L}(w) = \sum_{i=1}^{n} log\left(p(y^{(i)} \mid x^{(i)}; w)\right) - \frac{\lambda}{2} \mid\mid w \mid\mid^2$$

$$\nabla_w \mathcal{L} = \begin{bmatrix} \frac{d\mathcal{L}}{dw_1} \\ \vdots \\ \frac{d\mathcal{L}}{dw_k} \\ \vdots \\ \frac{d\mathcal{L}}{dw_d} \end{bmatrix}$$

Gradient Calculation

$$\mathcal{L}(w) = \sum_{i=1}^{n} log\left(p(y^{(i)} \mid x^{(i)}; w)\right) - \frac{\lambda}{2} \mid\mid w \mid\mid^2$$

$$\frac{d\mathcal{L}}{dw_k} = \sum_{i=1}^{n} \phi_k(x^{(i)}, y^{(i)}) - \sum_{i=1}^{n} \sum_{y \in \mathcal{V}} p(y \mid x^{(i)}; w)\phi_k(x^{(i)}, y) - \lambda w_k$$

(7.4)

Challenge With Log-Linear Model

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

- Where is most time spent in calculating the above probability?

Challenge With Log-Linear Model

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

- Where is most time spent in calculating the above probability?
- Normalization dominates the computation time

Challenge With Log-Linear Model

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

- Where is most time spent in calculating the above probability?
- Normalization dominates the computation time
- As the vocabulary size increases the problem becomes pronounced

Large Vocabulary Size

- Suppose words are divided into classes

Large Vocabulary Size

- Suppose words are divided into classes

$$p(y \mid x) = \sum_{c \in \mathbb{C}} p(y \mid x, c) \times p(c \mid x)$$

Large Vocabulary Size

- Suppose words are divided into classes

$$p(y \mid x) = \sum_{c \in \mathbb{C}} p(y \mid x, c) \times p(c \mid x)$$

- Moreover, each word belongs to a unique class

Large Vocabulary Size

- Suppose words are divided into classes

$$p(y \mid x) = \sum_{c \in \mathbb{C}} p(y \mid x, c) \times p(c \mid x)$$

- Moreover, each word belongs to a unique class

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

Large Vocabulary Size

- Suppose words are divided into classes

$$p(y \mid x) = \sum_{c \in \mathbb{C}} p(y \mid x, c) \times p(c \mid x)$$

- Moreover, each word belongs to a unique class

$$p(y \mid x) = p(y \mid x, c) \times p(c \mid x) \quad \textbf{(7.5)}$$

Class Based Speedup

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

- Now, we have two models

Class Based Speedup

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

- Now, we have two models
- The first model is bounded by number of words in a class: $\mid c \mid << \mid \mathcal{V} \mid$

Class Based Speedup

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

- Now, we have two models
- The first model is bounded by number of words in a class: $\mid c \mid << \mid \mathcal{V} \mid$
- Second model is bounded by number of classes: $\mid \mathbb{C} \mid << \mid \mathcal{V} \mid$

Class Based Speedup

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

- Now, we have two models
- The first model is bounded by number of words in a class: $\mid c \mid << \mid \mathcal{V} \mid$
- Second model is bounded by number of classes: $\mid \mathbb{C} \mid << \mid \mathcal{V} \mid$

$$\text{Assuming, } \mid \mathbb{C} \mid \approx \mid c \mid \approx \sqrt{\mid \mathcal{V} \mid}$$

Class Based Speedup

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

- Now, we have two models
- The first model is bounded by number of words in a class: $\mid c \mid << \mid \mathcal{V} \mid$
- Second model is bounded by number of classes: $\mid \mathbb{C} \mid << \mid \mathcal{V} \mid$

$$\text{Assuming, } \mid \mathbb{C} \mid \approx \mid c \mid \approx \sqrt{\mid \mathcal{V} \mid}$$

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

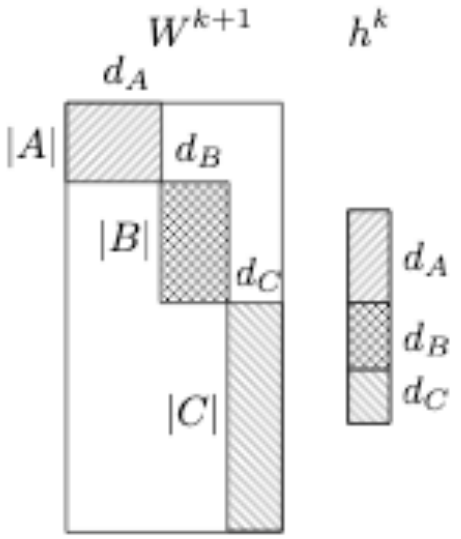
Hierarchical Class Based Speedup

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

- 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.

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 between 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 between 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

- We can incorporate more context by a feature based discriminative model
- One such approach is Log-linear model
- Idea is to create millions of indicator features via few feature templates
- Regularization is important for Log-linear model to work in practice
- There are ways to overcome very large vocabulary size limitations

References

1. Michael Collin's NLP Lecture Notes: http://www.cs.columbia.edu/~mcollins/loglinear.pdf
2. Chapter 4, Speech and Language Processing, Dan Jurafsky and James Martin
3. A Bit of Progress in Language Modeling: https://arxiv.org/pdf/cs/0108005.pdf
4. Classes For Fast Maximum Entropy Training : https://arxiv.org/pdf/cs/0108006.pdf
5. Strategies for Training Large Vocabulary Neural Language Models : https://www.aclweb.org/anthology/P16-1186.pdf

- Next class
  - Max Entropy Models
  - Neural Language Models