The Noisy Channel Model and Markov Models

Mark Johnson

September 3, 2014
The big ideas

• The story so far:
  ▶ machine learning classifiers learn a function that maps a data item $X$ to a label $Y$
  ▶ handle large item spaces $\mathcal{X}$ by decomposing each data item $X$ into a collection of features $F$
  ▶ but the label spaces $\mathcal{Y}$ have to be small to avoid sparse data problems

• Where we’re going from here:
  ▶ many important problems involve large label spaces $\mathcal{Y}$
    – Part-Of-Speech (POS) tagging, where $Y$ is a vector of POS tags
    – Syntactic parsing, where $Y$ is a syntactic parse tree
  ▶ basic approach: decompose $Y$ into parts or features $G$
  ▶ if each feature $G_j$ is independent, just learn a separate classifier for each $G_j$
  ▶ but often there are important dependencies between the $G_j$
    – in POS tagging, adjectives typically precede nouns
  $\Rightarrow$ more sophisticated models are required to capture these dependencies
Outline

The noisy channel model

Bigram language models

Learning bigram models from text

Markov Chains and higher-order $n$-grams

$n$-gram language models as Bayes nets

Summary
Many problems involve predicting a complex label $Y$ from data $X$

- in *automatic speech recognition*, $X$ is acoustic waveform, $Y$ is transcript
- in *machine translation*, $X$ is source language text, $Y$ is target language translation
- in *spelling correction*, $X$ is a source text with spelling mistakes, and $Y$ is a target text without spelling mistakes
- in *automatic summarisation*, $X$ is a document, $Y$ is a summary of that document

Suppose we can estimate $P(Y \mid X)$ somehow. Then we should compute:

$$
\hat{y}(x) = \arg\max_{y \in \mathcal{Y}} P(Y=y \mid X=x)
$$

Problems we have to solve:

- $|\mathcal{Y}|$ is astronomical $\Rightarrow$ computing $\arg\max$ may be intractable
- $|\mathcal{Y}|$ is astronomical $\Rightarrow$ *how can we estimate* $P(Y \mid X)$?
The noisy channel model

- The noisy channel model uses **Bayes rule** to invert $P(Y \mid X)$

$$P(Y \mid X) = \frac{P(X \mid Y) P(Y)}{P(X)}$$

- We can ignore $P(X)$ if our goal is to compute

$$\hat{y}(x) = \arg\max_{y \in Y} P(X=x \mid Y=y) P(Y=y)$$

because $P(X=x)$ is a constant

- $P(X \mid Y)$ is called the **channel model** or the **distortion model**
- $P(Y)$ is called the **source model** or the **language model**
  - can often be learnt from cheap, readily available data
The noisy channel model in spelling correction, speech recognition and machine translation

\[
\hat{y}(x) = \arg\max_{y \in Y} P(X=x \mid Y=y) \cdot P(Y=y)
\]

- The channel models are task-specific
  - for spelling correction, \( P(X \mid Y) \) maps words to their likely mis-spellings
  - for speech recognition, \( P(X \mid Y) \) maps words or phonemes to acoustic waveforms
  - for machine translation, \( P(X \mid Y) \) maps words or phrases to their translations
- The source model \( P(Y) \), which is also called a language model, is the same
  - \( P(Y) \) is the *probability of a sentence* \( Y \)
  - can be learned from readily available text corpora
Outline

The noisy channel model

Bigram language models

Learning bigram models from text

Markov Chains and higher-order $n$-grams

$n$-gram language models as Bayes nets

Summary
Language models

- A *language model* calculates the probability of a sequence of words or phonemes.
- In many NLP applications the *source model* $P(Y)$ in a noisy channel model is a language model.

$$
\hat{y}(x) = \arg\max_{y \in Y} P(X=x \mid Y=y) \cdot P(Y=y)
$$

- In such applications, $Y$ is a *sequence of words or phonemes*.
- The language model is used to calculate the probability of possible sequences $Y$ of words or phonemes.
- The job of the language model is to distinguish likely sequences of words or phonemes from unlikely ones.
- It can be learnt from cheaply-available text collections.
The bigram assumption for language models

- A language model estimates the probability $P(Y)$ of a sentence $Y = (Y_1, \ldots, Y_n)$, where $Y_i$ is the $i$th word in the sentence.
- Recall the relationship between joint and conditional probabilities:

  $$P(U, V) = P(V) P(U | V)$$

- We can use this to rewrite $P(Y) = P(Y_1, \ldots, Y_n)$:

  $$P(Y_1, \ldots, Y_n) = P(Y_1) P(Y_2, \ldots, Y_n | Y_1)$$
  $$= P(Y_1) P(Y_2 | Y_1) P(Y_3 | Y_1, Y_2)$$
  $$\ldots P(Y_n | Y_1, \ldots, Y_{n-1})$$

- Now make the bigram assumption: $P(Y_j | Y_1, \ldots, Y_{j-1}) \approx P(Y_j | Y_{j-1})$, i.e., word $Y_j$ only depends on $Y_{j-1}$.
- Then $P(Y)$ simplifies to:

  $$P(Y_1, \ldots, Y_n) = P(Y_1) P(Y_2 | Y_1) \ldots P(Y_n | Y_{n-1})$$
Homogeneity assumption in language models

- Using bigram assumption we simplified

\[
P(Y_1, \ldots, Y_n) = P(Y_1) P(Y_2 | Y_1) \ldots P(Y_n | Y_{n-1})
\]

\[
= P(Y_1) \prod_{i=2}^{n} P(Y_i | Y_{i-1})
\]

- Homogeneity assumption: conditional probabilities don’t change with \(i\), i.e., there is a matrix \(s\) such that

\[
P(Y_i = y | Y_{i-1} = y') = s_{y,y'}
\]

- Then the bigram model probability \(P(Y=y)\) of a sentence \(y = (y_1, \ldots, y_n)\) is:

\[
P(Y=y) = P(Y_1 = y_1) s_{y_2,y_1} s_{y_3,y_2} \cdots s_{y_n,y_{n-1}}
\]

\[
= P(Y_1 = y_1) \prod_{i=2}^{n} s_{y_i,y_{i-1}}
\]
Using end-markers to handle initial and final conditions

• We simplify the model by assuming the string $y$ is padded with end-markers $\$$.  

\[ y = (\$, y_1, y_2, \ldots, y_n, \$) \]

I.e., $Y_0 = \$$. and $Y_{n+1} = \$$.  

• Then the bigram language model probability $P(Y=y)$ is:

\[
P(Y=y) = s_{y_1,\$} \, s_{y_2,y_1} \, \ldots \, s_{y_n,y_{n-1}} \, s_{\$,y_n} \\
= \prod_{i=1}^{n+1} s_{y_i,y_{i-1}}
\]
Bigram language model example

\[ s = \begin{array}{c|cccc} y_i \backslash y_{i-1} & $ & bow & wow & woof \\ \hline $ & 0 & 0 & 0.1 & 0.2 \\ bow & 0.5 & 0 & 0.7 & 0.4 \\ wow & 0 & 1.0 & 0 & 0 \\ woof & 0.5 & 0 & 0.2 & 0.4 \\ \end{array} \]

\[
P($,\text{bow,} \text{wow,} \text{woof,} \text{woof,}$)
\]

\[ = s_{\text{bow},$} \cdot s_{\text{wow,bow}} \cdot s_{\text{woof,wow}} \cdot s_{\text{woof,woof}} \cdot s_{\$\text{,woof}} \]

\[ = 0.5 \cdot 1.0 \cdot 0.2 \cdot 0.4 \cdot 0.2 \]

\[ = 0.008 \]
Outline

The noisy channel model

Bigram language models

Learning bigram models from text

Markov Chains and higher-order $n$-grams

$n$-gram language models as Bayes nets

Summary
Estimating bigram models from text

- We can estimate the bigram model $s$ from a **text corpus**
- Collect a **vector of unigram counts** $m$ and a **matrix of bigram counts** $n$

$$
m_y = \text{number of times } y \text{ is followed by anything in corpus}
$$

$$
n_{y',y} = \text{number of times } y' \text{ follows } y \text{ in corpus}
$$

- make sure you count the beginning and end of sentence markers!

- Maximum likelihood estimates:

$$
\hat{s}_{y',y} = \frac{n_{y',y}}{m_y}
$$

- Add-1 smoothed estimates (a good idea!):

$$
\hat{s}_{y',y} = \frac{n_{y',y} + 1}{m_y + |\mathcal{V}|}
$$

where $\mathcal{V}$ is the **vocabulary** (set of words) of the corpus
Estimating a bigram model example

\[
\text{corpus} = \begin{pmatrix}
\$ \text{bow} \ \text{wow} \ \text{woof} \ \text{woof} \\
\text{woof} \ \text{bow} \ \text{wow} \ \text{bow} \ \text{wow} \ \text{woof} \\
\text{bow} \ \text{wow} \\
\end{pmatrix}
\]

\[V = \{\$, \text{bow}, \text{wow}, \text{woof}\}\]

\[m = \begin{pmatrix}
\text{bow} & \text{wow} & \text{woof} \\
3 & 4 & 4 & 4 \\
\end{pmatrix}
\]

\[
\begin{array}{c|ccc}
\text{y}_i \ \text{y}_{i-1} & \text{bow} & \text{wow} & \text{woof} \\
\hline
\text{y}_i \ \text{y}_{i-1} & 0 & 0 & 1 & 2 \\
\text{m} & 2 & 0 & 1 & 1 \\
\text{n} & 0 & 4 & 0 & 0 \\
\text{woof} & 1 & 0 & 2 & 1 \\
\end{array}
\]

\[
\hat{s} = \begin{pmatrix}
\$ & \text{bow} & \text{wow} & \text{woof} \\
1/7 & 1/8 & 2/8 & 3/8 \\
\text{bow} & 3/7 & 1/8 & 2/8 & 2/8 \\
\text{wow} & 1/7 & 5/8 & 1/8 & 1/8 \\
\text{woof} & 2/7 & 1/8 & 3/8 & 2/8 \\
\end{pmatrix}
\]
Outline

The noisy channel model

Bigram language models

Learning bigram models from text

Markov Chains and higher-order $n$-grams

$n$-gram language models as Bayes nets

Summary
Markov Models

- A **first-order Markov chain** is a sequence of random variables $Y_1, Y_2, Y_3, \ldots$, where:

  \[ P(Y_i \mid Y_1, \ldots, Y_{i-1}) = P(Y_i \mid Y_{i-1}) \]

  I.e., $Y_i$ is independent of $Y_1, \ldots, Y_{i-2}$ given $Y_i$, or the value $Y_i$ at time $i$ only depends on the value $Y_{i-1}$ at time $i-1$

- The bigram language model is a first-order Markov chain
  - Informally, the order of a Markov chain indicates “how far back in the past” the next state can depend on
Higher-order Markov chains and $n$-gram language models

- An $n$-gram language model uses adjacent $n$-word sequences to predict the probability of a sequence
  - E.g., the *trigrams* in $y = \text{(the, rain, in, spain)}$ are
    - ($$, $$, \text{the})$, ($$, \text{the, rain})$, \text{(the, rain, in)}, \text{(rain, in, spain)}, \text{(in, spain, $$)}, \text{(spain, $$, $$)}$
- In an $m$th order Markov chain, $Y_i$ depends only on $Y_{i-m}, \ldots, Y_{i-1}$
- So an $m + 1$-gram language model is an $m$th order Markov chain
  - E.g., a bigram language model is a first-order Markov chain
  - E.g., a trigram language model is a second-order Markov chain
Outline

The noisy channel model

Bigram language models

Learning bigram models from text

Markov Chains and higher-order $n$-grams

$n$-gram language models as Bayes nets

Summary
**n-gram language models**

- Goal: estimate $P(y)$, where $y = (y_1, \ldots, y_m)$ is a sequence of words
- $n$-gram models decompose $P(y)$ into product of conditional distributions

\[
P(y) = P(y_1)P(y_2 | y_1)P(y_3 | y_1, y_2) \ldots P(y_m | y_1, \ldots, y_{m-1})
\]

E.g., $P(\text{wreck a nice beach}) = P(\text{wreck})P(\text{a} | \text{wreck})P(\text{nice} | \text{wreck a})$

\[
P(\text{beach} | \text{wreck a nice})
\]

- $n$-gram assumption: *no dependencies span more than $n$ words*, i.e.,

\[
P(y_i | y_1, \ldots, y_{i-1}) \approx P(y_i | y_{i-n}, \ldots, y_{i-1})
\]

E.g., A *bigram model* is an $n$-gram model where $n = 2$:

\[
P(\text{wreck a nice beach}) \approx P(\text{wreck})P(\text{a} | \text{wreck})P(\text{nice} | \text{a})P(\text{beach} | \text{nice})
\]
An $n$-gram language model is a Markov model that factorises the distribution over sentences into a product of conditional distributions:

$$P(y) = \prod_{i=1}^{m} P(y_i \mid y_{i-n}, \ldots, y_{i-1})$$

- pad $y$ with end markers, i.e., $y = (\$, y_1, y_2, \ldots, y_m, \$)

- Bigram language model as Bayes net:

- Trigram language model as Bayes net:
The conditional word models in $n$-gram models

- An $n$-gram model factorises $P(y)$ into a product of conditional models, each of the form:
  \[ P(y_n | y_1, \ldots, y_{n-1}) \]

- The performance of an $n$-gram model depends greatly on exactly how these conditional models are defined
  - huge amount of work on this

- *Random forest* and *deep learning* methods for estimating these conditional distributions currently produce state-of-the-art language models
Outline

The noisy channel model

Bigram language models

Learning bigram models from text

Markov Chains and higher-order $n$-grams

$n$-gram language models as Bayes nets

Summary
Summary

- In many NLP applications, the label space $\mathcal{Y}$ is *astronomically large* ⇒ decompose it into components (e.g., elements, features, etc.)
- The *noisy channel model* uses Bayes rule to “invert” a sequence labelling problem
  - enables us to use *language models* trained on large text collections
- $n$-gram language models are *Markov models* that predict the next word based on the preceding $n - 1$ words