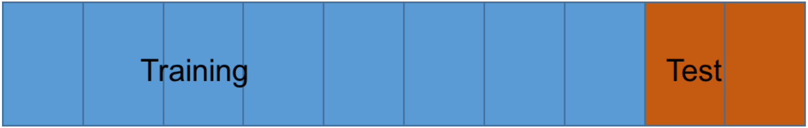
LING439/539 - Statistical NLP
Chapter 6. Statistical inference: n-gram models over sparse data (continued)

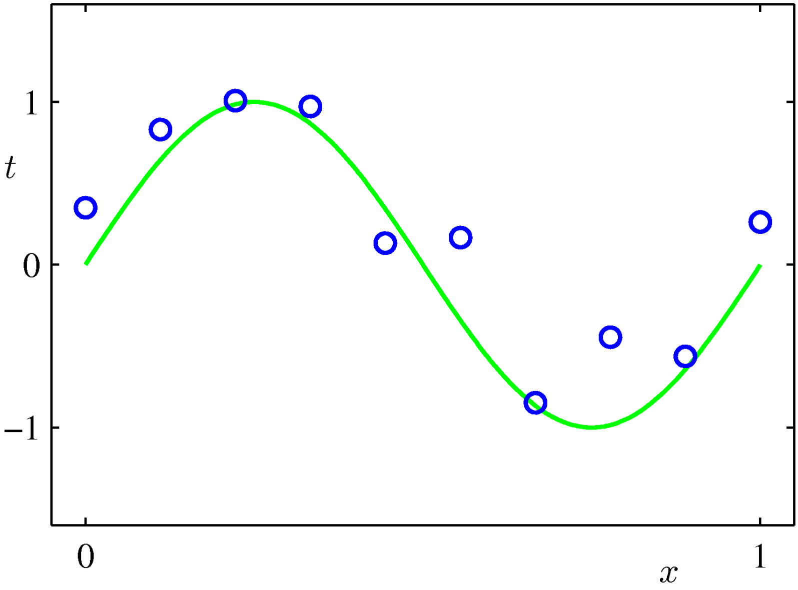
Thursday, September 8 2016

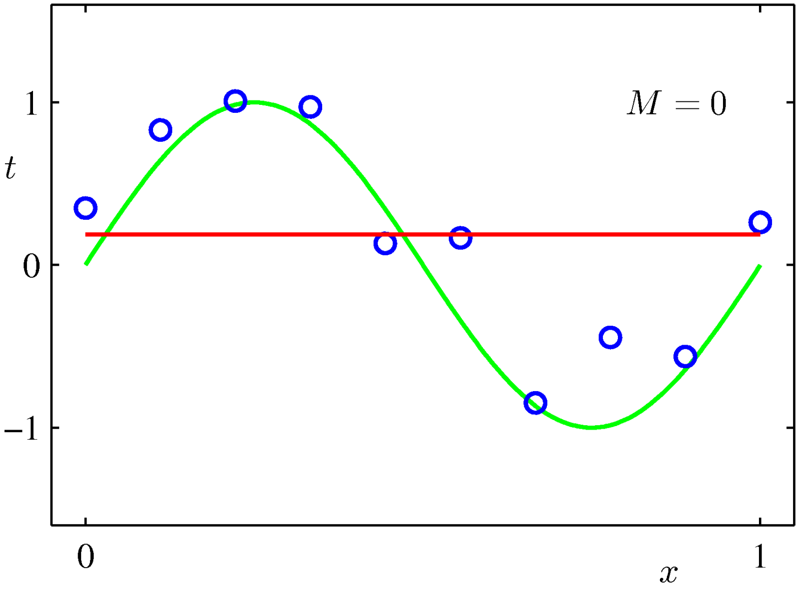
Training and test sets

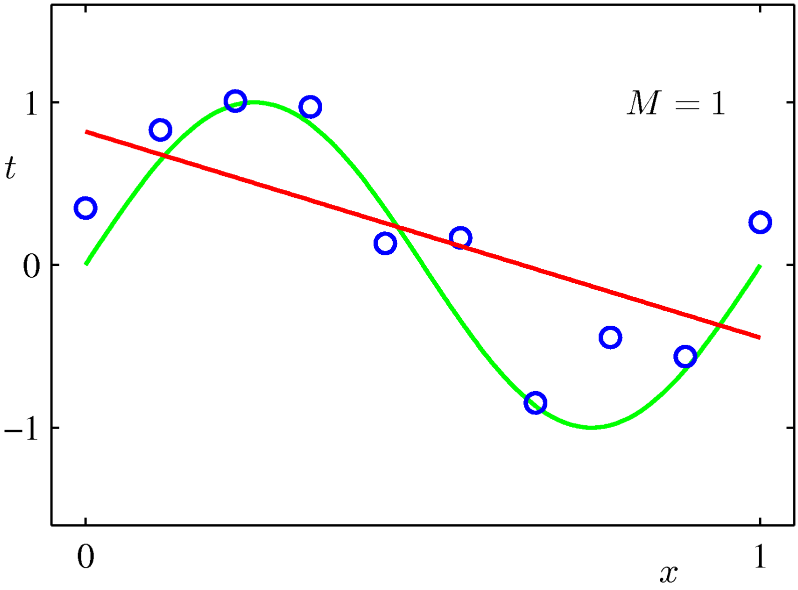


Training and test sets

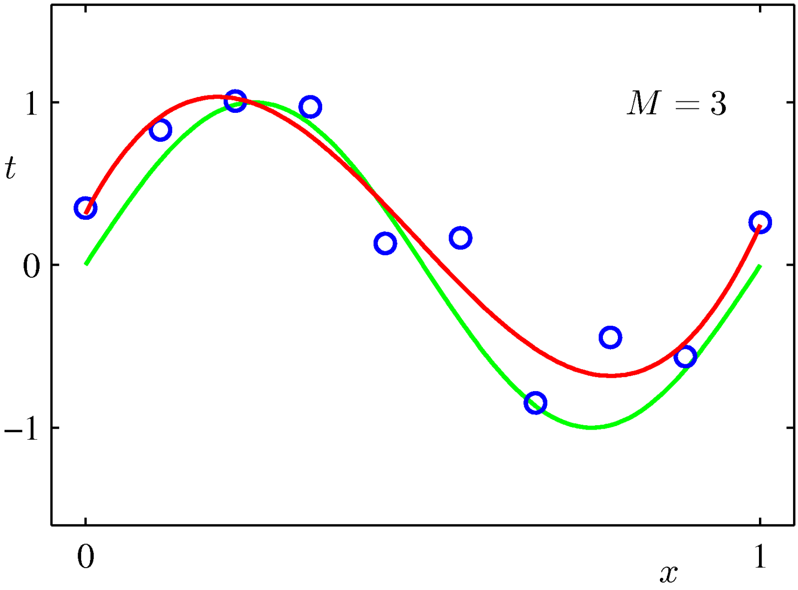
Training



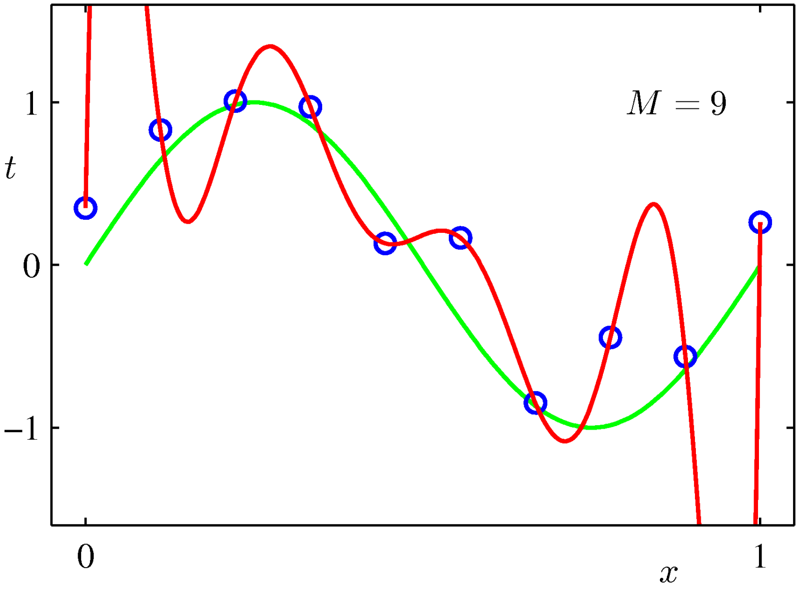

$M = 0$
t
1
0
−1
0
1
x



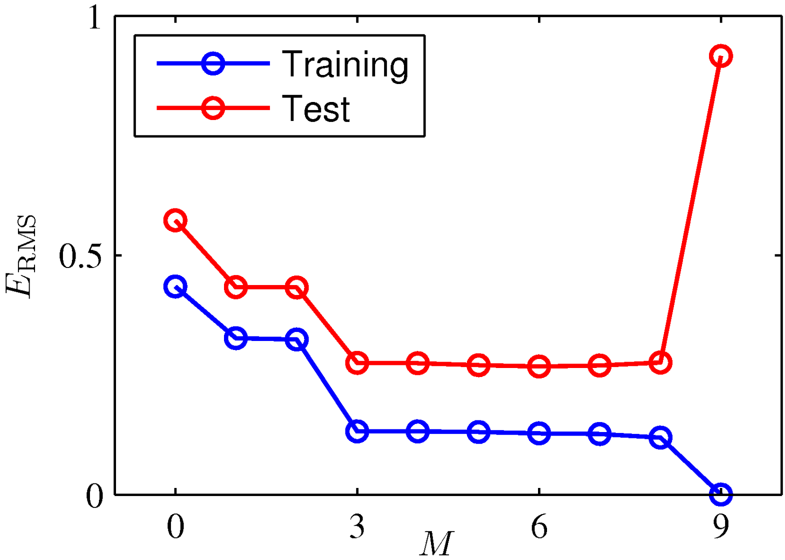
$M = 1$
t
1
0
−1
0
x
1

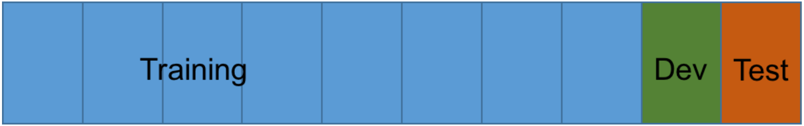


$M = 3$
t
1
0
−1
0
x
1


M = 9
t
1
0
−1
0
x
1

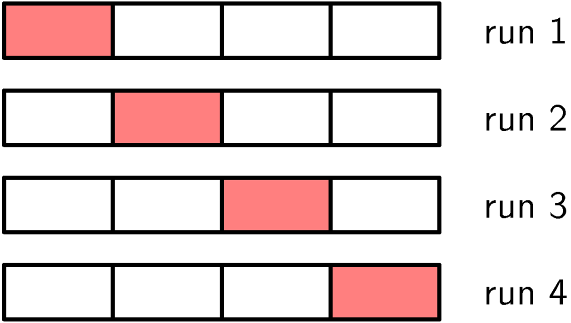
Overfitting





Training, development and test sets

Cross-validation



The technique of S-fold cross-validation, illustrated here for the case of S = 4,

Out of vocabulary: OOV

No OOV in the closed vocabulary
In an open vocabulary system, we add a pseudo-word `<unk>`.

1. Choose a vocabulary (word list) that is fixed in advance.
2. Covert in the training set any word that is not in this list to `<unk>`.
3. Estimate the probabilities for `<unk>` from its counts.

or

1. Covert in the training set any word that **occurs once** to `<unk>` (*hapax legomenon*).
2. Estimate the probabilities for `<unk>` from its counts.

Evaluating N-grams

How to evaluate the performance of a language model:

extrinsic evaluation	**intrinsic evaluation**
* in vivo evaluation * embed the language model in an application and measure the total performance of the application	* perplexity
expensive	

Note that an improvement in perplexity **does not guarantee** an extrinsic improvement such as speech recognition performance.

Perplexity

$$W = w_1 w_2 ... w_N$$

$$\begin{aligned}
P(W) &= P(w_1 w_2 ... w_N)^{-\frac{1}{N}} \\
&= \prod_{i=1}^{N} P(w_i | w_1 w_2 ... w_{i-1})^{-\frac{1}{N}} && \text{\# chain rule} \\
&= \prod_{i=1}^{N} P(w_i | w_{i-1})^{-\frac{1}{N}} && \text{\# bigram language model}
\end{aligned}$$

Berkeley Restaurant Project (Recap)

A dialogue system that answered questions about a database of restaurants in Berkeley, California. It contains 9,332 sentences.

33_1_0001	okay let's see i want to go to a thai restaurant . [uh] with less than ten dollars per person
33_1_0002	<i> <like> <to> <eat> [uh] i like to eat at lunch time . so that would be eleven a_m to one p_m
33_1_0003	i don't want to walk for more than five minutes
33_1_0004	tell me more about the [uh] na- nakapan [uh] restaurant on martin luther king
33_1_0005	i like to go to a hamburger restaurant
33_1_0006	let's start again
33_1_0007	i like to get a hamburger at an american restaurant
33_1_0008	i'd like to eat dinner . and i don't mind walking [uh] . for half an hour

	i	want	to	eat	chinese	foot	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

Bigram counts for eight of the words (out of $V = 1446$) in the Berkeley Restaurant Project corpus of 9332 sentences.

(unigram)	**i**	**want**	**to**	**eat**	**chinese**	**foot**	**lunch**	**spend**
	2533	927	2417	746	158	1093	341	278

	i	want	to	eat	chinese	foot	lunch	spend
i	.002	.33	0	.0036	0	0	0	.00079
want	.0022	0	.66	0.0011	.0065	.0065	.0054	.0011
to	.00083	0	.0017	.28	.00083	0	.0025	.087
eat	0	0	.0027	0	.021	.0027	.0056	0
chinese	.0063	0	0	0	0	.52	.0063	0
food	.014	0	.014	0	.00092	.0037	0	0
lunch	.0059	0	0	0	0	.0029	0	0
spend	.0036	0	.0036	0	0	0	0	0

Bigram probabilities for eight of the words in the Berkeley Restaurant Project corpus of 9332 sentences.

The probability of the sentence *I want English food*:

$P(\texttt{<s> I want English food </s>})$

$= P(\texttt{i}|\texttt{<s>})\ P(\texttt{want}|\texttt{i})\ P(\texttt{english}|\texttt{want})\ P(\texttt{food}|\texttt{english})\ P(\texttt{</s>}|\texttt{food})$

Smoothing

Sparse data because any corpus is limited
$\rightarrow$ never observed in training
$\rightarrow$ zero probability N-grams

Furthermore, the MLE method also produces poor estimates when the counts are non-zero but still small
$\rightarrow$ modify the MLE for computing N-gram probabilities.

Smoothing

Shaving a little bit of probability mass from the higher counts and piling it instead on the zero counts.

Laplace smoothing

Unigram probabilities:

$$P(w_i) = \frac{C(w_i)}{N} \qquad \Rightarrow \qquad P_{\text{Laplace}}(w_i) = \frac{C(w_i) + 1}{N + V}$$

- N: the total number of word tokens
- V: # of types

Bigram probabilities:

$$P(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)}{C(w_{n-1})}$$

$$\Rightarrow \quad P_{\text{Laplace}}(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n) + 1}{C(w_{n-1}) + V}$$

- N: the total number of word tokens
- V: # of types

	i	**want**	**to**	**eat**	**chinese**	**foot**	**lunch**	**spend**
i	6	828	*1*	10	*1*	*1*	*1*	3
want	3	1	609	2	7	7	6	2
to	3	1	5	687	3	1	7	212
eat	1	1	3	1	17	3	43	1
chinese	2	1	1	1	1	83	2	1
food	16	1	16	1	2	5	1	1
lunch	3	1	1	1	1	2	1	1
spend	2	1	2	1	1	1	1	1

Add-one smoothed bigram counts for eight of the words (out of $V = 1446$) in the Berkeley Restaurant Project corpus of 9332 sentences.

	i	**want**	**to**	**eat**	**chinese**	**foot**	**lunch**	**spend**
i	.0015	.21	*.00025*	.0025	*.00025*	*.00025*	*.00025*	.00075
want	.0013	.00042	.26	.00084	.0029	.0029	.0025	.00084
to	.00078	.00026	.0013	.18	.00078	.00026	.0018	.055
eat	.00046	.00046	.0014	.00046	.0078	.0014	.02	.00046
chinese	.0012	.00062	.00062	.00062	.00062	.052	.0012	.00062
food	.0063	.00039	.0063	.00039	.00079	.002	.00039	.00039
lunch	.0017	.00056	.00056	.00056	.00056	.0011	.00056	.00056
spend	.0012	.00058	2	.00058	.00058	.00058	.00058	.00058

Add-one smoothed bigram probabilities for eight of the words (out of $V = 1446$) in the Berkeley Restaurant Project corpus of 9332 sentences.

Adjusted count c^*

unigram

$$c_i^* = (c_i + 1)\frac{N}{N + V}$$

bigram

$$c^*(w_{n-1}w_n) = (C(w_{n-1}w_n) + 1)\frac{C(w_{n-1})}{C(w_{n-1}) + V}$$

	i	**want**	**to**	**eat**	**chinese**	**foot**	**lunch**	**spend**
i	5	827	0	9	0	0	0	2
	→3.8	→527	→.64	→6.4	→.64	→.64	→.64	→1.9
want	2	0	608	1	6	6	5	1
	→1.2	→.39	→238	→.78	→2.7	→2.7	→2.3	→.78
to	2	0	4	686	2	0	6	211
	→1.9	→.63	→3.1	→430	→1.9	→.63	→4.4	→133
eat	0	0	2	0	16	2	42	0
	→.34	→.34	→1	→.34	→5.8	→1	→15	→.34
chinese	1	0	0	0	0	82	1	0
	→.2	→.098	→.098	→.098	→.098	→8.2	→.2	→.098
food	15	0	15	0	1	4	0	0
	→6.9	→.43	→6.9	→.43	→.86	→2.2	→.43	→.43
lunch	2	0	0	0	0	1	0	0
	→.57	→.19	→.19	→.19	→.19	→.38	→.19	→.19
spend	1	0	1	0	0	0	0	0
	→.32	→.16	→.32	→.16	→.16	→.16	→.16	→.16

Adjusted bigram counts for eight of the words (out of $V = 1446$) in the Berkeley Restaurant Project corpus of 9332 sentences.

Other discounting algorithms

A basic idea of other discounting algorithms such as

- Good-Turing
- Witten-Bell
- Kneser-Ney

is to use the count of things you've seen *once* (a singleton or a hapax legomenon) to help estimate the count of things you've never seen.

For Good-Turing smoothing, we use the frequency of hapax as a re-estimate of the frequency of zero-count bigrams.

Good-Turing discounting

N_c = the number of N-grams that occur c times
$\rightarrow$ frequency of frequency c.

- N_0: the number of bigrams with count 0.
- N_1: the number of bigrams with count 1 (hapax).
- ...

The Good-Turing intuition is to estimate the probability of things that occur c times in the training corpus by the MLE probability of things that occur $c+1$ times in the corpus.

Smoothed (or adjusted) count $\quad c^* = (c+1)\dfrac{N_{c+1}}{N_c}$

—

The probability estimate in Good-Turing estimation is of the form

$$P_{GT} = \frac{c^*}{N} \quad (c > 0) \qquad \text{or} \qquad P_{GT} = \frac{N_1}{N} \quad (c = 0)$$

- N: the total number of word tokens

- $N_1 = 3$
- $N_2 = 1$
- $N = 18$

If $C(w_1 ... w_n) = 1$:

c	1
MLE	$P = \frac{1}{N} = \frac{1}{18}$
c^*	$c^*(w_1 ... w_n) = \ ?$
GT	$P_{GT}(w_1 ... w_n) = \ ?$

- $N_1 = 3$
- $N_2 = 1$
- $N = 18$

If $C(w_1 ... w_n) = 1$:

c	1
MLE	$P = \frac{1}{N} = \frac{1}{18}$
c^*	$c^*(w_1 ... w_n) = (c+1) \times \frac{N_2}{N_1} = (1+1) \times \frac{1}{3} = \frac{2}{3}$
GT	$P_{GT}(w_1 ... w_n) = ?$

- $N_1 = 3$
- $N_2 = 1$
- $N = 18$

If $C(w_1 ... w_n) = 1$:

c	1
MLE	$P = \frac{1}{N} = \frac{1}{18}$
c^*	$c^*(w_1 ... w_n) = (c+1) \times \frac{N_2}{N_1} = (1+1) \times \frac{1}{3} = \frac{2}{3}$
GT	$P_{GT}(w_1 ... w_n) = \frac{c^*}{N} = \frac{\frac{2}{3}}{18} = \frac{1}{27}$

- $N_1 = 3$
- $N_2 = 1$
- $N = 18$

If $C(w_1...w_n) = 0$:

c	0
MLE	$P = \frac{0}{N} = 0$
c^*	-
GT	$P_{GT}(w_1...w_n) = \frac{N_1}{N} = \frac{3}{18}$