自然语言处理导论

#L4

语言模型的评价

袁彩霞

yuancx@bupt.edu.cn

智能科学与技术中心

语言模型

- n-gram语言模型
  - 模型参数
  - 数据稀疏及平滑
- 神经网络语言模型
- 模型质量评价

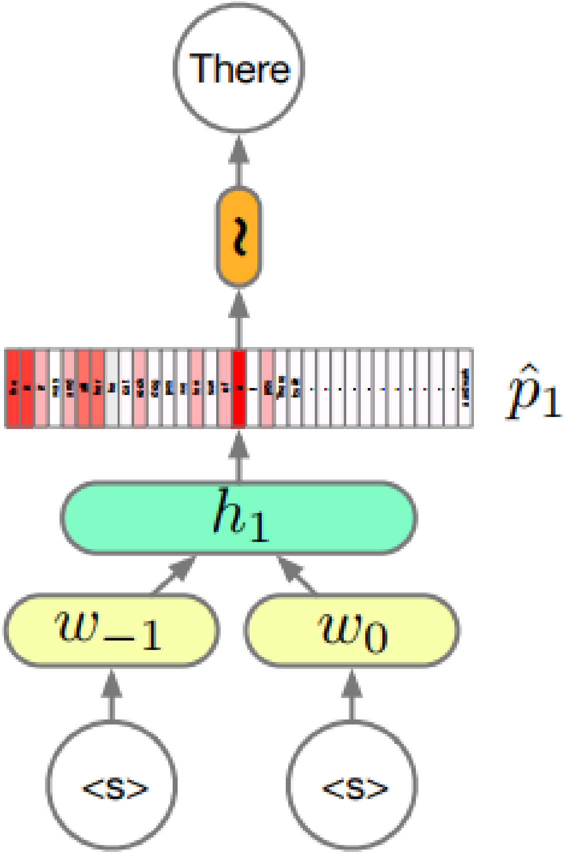
Core Problems Revisited

- N-gram Language Model
  - Often simplified to trigrams:

$$P(w_1, w_2, w_3, \ldots, w_n) = \prod_i P(w_i | w_{i-2}, w_{i-1})$$

- Neural Network based Language Model

$$
\begin{aligned}
h_n &= g(V[w_{n-1}; w_{n-2}] + c) \\
\hat{p}_n &= \text{softmax}(Wh_n + b) \\
\text{softmax}(u)_i &= \frac{\exp u_i}{\sum_j \exp u_j}
\end{aligned}
$$



How the NNLM works - example

- Consider the training corpus having the following sentences:
  - “the dog saw a cat”
  - “the dog chased the cat”
  - “the cat climbed a tree”
- Vocabulary: eight words sorted alphabetically
  - “a cat chased climber dog saw the tree”
- Network architecture: eight input neurons and eight output neurons, let us assume that we decide to use three neurons in the hidden layer
- Parameters:
  - *WI* and *WO* will be 8×3 and 3×8 matrices
  - Before training begins, these matrices are initialized to small random values as is usual in neural network training, for example,

How the NNLM works - example

WI =

-0.094491	-0.443977	0.313917
-0.490796	-0.229903	0.065460
0.072921	0.172246	-0.357751
0.104514	-0.463000	0.079367
-0.226080	-0.154659	-0.038422
0.406115	-0.192794	-0.441992
0.181755	0.088268	0.277574
-0.055334	0.491792	0.263102

WO =

0.023074	0.479901	0.432148	0.375480	-0.364732	-0.119840	0.266070	-0.351000
-0.368008	0.424778	-0.257104	-0.148817	0.033922	0.353874	-0.144942	0.130904
0.422434	0.364503	0.467865	-0.020302	-0.423890	-0.438777	0.268529	-0.446787

How the NNLM works - example

- Suppose we want the network to learn relationship between the words "cat" and "climbed".
- The input vector X: $[0\ 1\ 0\ 0\ 0\ 0\ 0\ 0]^T$
- The target vector: $[0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ ]^T$
- The output at the hidden layer neurons can be computed as:

$$H^T = X^T\,WI = [-0.490796\ -0.229903\ 0.065460]$$

- The activation vector for output layer neurons can be computed as

$$H^T WO = [0.100934\ \ -0.309331\ \ -0.122361\ \ -0.151399\ \ 0.143463\ \ -0.051262\ \ -0.079686\ \ 0.112928]$$

How the NNLM works - example

- Since the goal is to produce probabilities for words in the output layer, $P(word_k|word_{context})$ for $k = 1,\ldots, V$
  - i.e., $\Sigma_k P(word_k|word_{context}) = 1$
- This can be achieved by converting activation values of output layer neurons to probabilities using the softmax function

$$y_k = \Pr(word_k \mid word_{context}) = \frac{\exp(activation(k))}{\sum_{n=1}^{V} \exp(activation(n))}$$

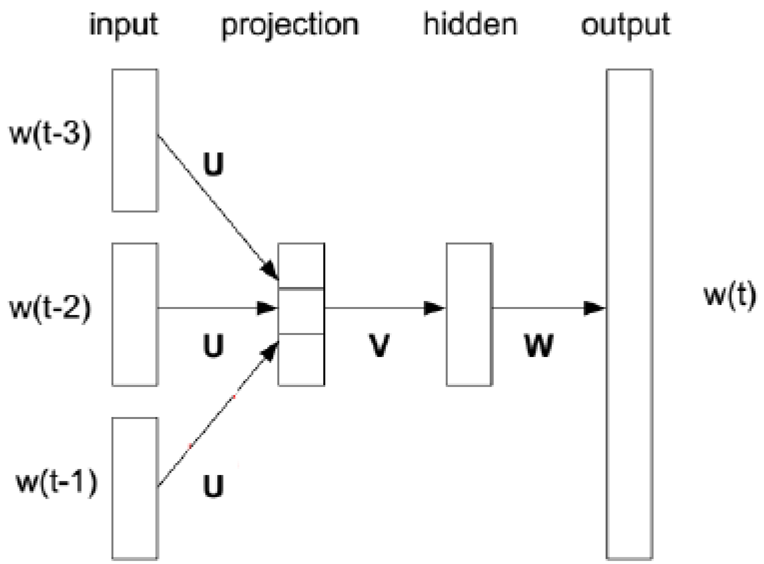
- Thus, the probabilities for eight words in the corpus are: [0.143073 0.094925 0.114441 **0.111166** 0.149289 0 .122874 0.119431 0.144800]

How the NNLM works - example

- Given the target vector $[0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ ]^T$, the error vector for the output layer is easily computed by subtracting the probability vector from the target vector.
- Once the error is known, the weights in the matrices *WO* and *WI* can be updated using backpropagation.
- In essence, this is how NNLM learns relationships between words and in the process develops vector representations for words in the corpus.

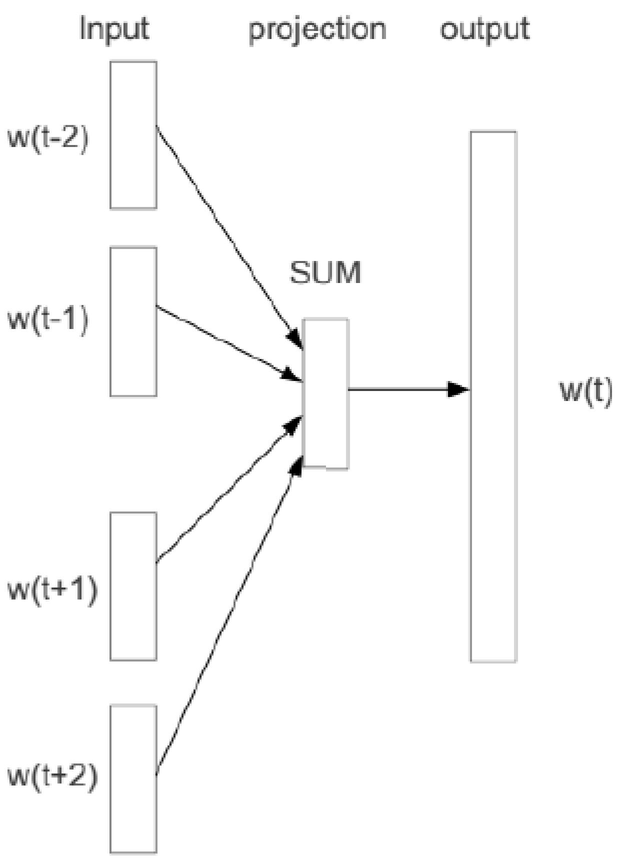
NNLM: a big picture

- Neural net based word vectors were traditionally trained as part of neural network language model (Bengio, et al, 2003)
- This models consists of input layer, projection layer, hidden layer and output layer



Other NNLMs

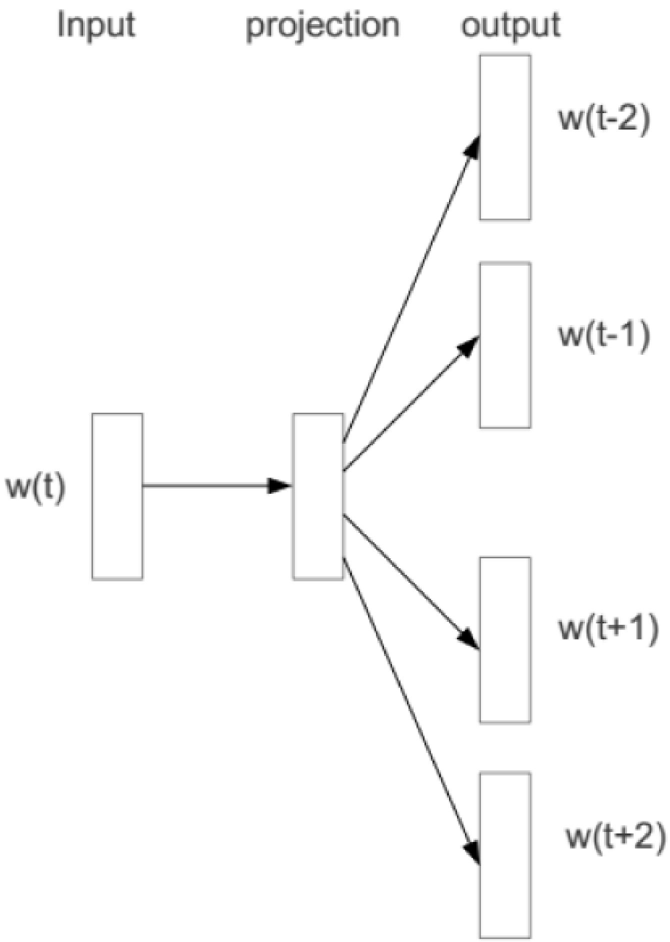
- The ‘continuous bag-of-words model’ (CBOW) adds inputs from words within short window to predict the current word
- The weights for different positions are shared
- Computationally much more efficient than normal NNLM
- The hidden layer is just linear



Mikolov et al. 2013. Distributed Representations of Words and Phrases and their Compositionality.

Other NNLMs

- We can reformulate the CBOW model by predicting surrounding words using the current word
- This architectures is called 'skip-gram NNLM'
- If both are trained for sufficient number of epochs, their performance is similar



Mikolov et al. 2013. Distributed Representations of Words and Phrases and their Compositionality.

Word Analogies

- Test for linear relationships, examined by Mikolov et al. (2014)

+king	[0.30 0.70]
-man	[0.20 0.20]
+woman	[0.60 0.30]
=queen	[0.70 0.80]

RNNLM toolkit

- Available at rNNLM.org
- Allows training of RNN and RNNME models
- Extensions are actively developed, for example multi-threaded version with hierarchical softmax: http://svn.code.sf.net/p/kaldi/code/trunk/tools/rNNLM-hs-0.1b/

Feedforward NNLM toolkit

- Continuous Space Language Model toolkit: http://www-lium.univ-lemans.fr/cslm/
- Implementation of feedforward neural network language model by Holger Schwenk

Word2vec

- Available at https://code.google.com/p/word2vec/
- Tool for training the word vectors using CBOW and skip-gram architectures, supports both negative sampling and hierarchical softmax
- Optimized for very large datasets (>billions of training words)
- Includes links to models pre-trained on large datasets (100B words)

Summary: NNLM

- NNLMs are currently the state-of-the-art in language modeling
- RNN outperforms FNN on language modeling tasks, both are better than n-grams, in many NLP tasks like ASR, MT
- Significant ongoing efforts to scale training to very large datasets
- The question "are neural nets better than n-grams" is incomplete: the best solution is to use both

语言模型

- n-gram语言模型
  - 模型参数
  - 数据稀疏及平滑
- 神经网络语言模型
- 模型质量评价

模型质量评价

- 语言模型的建模质量如何?
- 简单说来：一个好的语言模型应该给实际使用的句子打较高的概率

- 构建语言模型的两个阶段：
  - 模型训练：从训练数据（training data）中学习得到语言模型，即从训练样例中统计n-grams的参数，并估计其n-gram的条件概率；
  - 模型测试：在给定的测试数据（test data）中评价学习得到的语言模型

实例：unigram

- 训练数据：

there is a big house
i buy a house
they buy the new house
…

- 模型：

p(there) = 0.0714, p(is) = 0.0714, p(a) = 0.1429
p(big) = 0.0714, p(house) = 0.2143, p(i) = 0.0714
p(buy) = 0.1429, p(they) = 0.0714, p(the) = 0.0714
p(new) = 0.0714, …

- 测试数据：S=they buy a big house
  - p(S) = 0.0714(they) ×0.1429(buy) ×0.0714(a) ×0.1429(big) ×0.2143(house)= 0.0000231

实例：bigram

- 训练数据：

there is a big house
i buy a house
they buy the new house
…

- 模型：

$p(big|a) = 0.5$, $p(is|there) = 1$, $p(buy|they) = 1$
$p(house|a) = 0.5$, $p(buy|i) = 1$, $p(a|buy) = 0.5$
$p(new|the) = 1$, $p(house|big) = 1$, $p(the|buy) = 0.5$
$p(a|is) = 1$, $p(house|new) = 1$, $p(they|<s>) = .333$
…

- 测试数据：S= they buy a big house
  - $p(S) = 0.333\ (they) \times 1\ (buy|they) \times 0.5\ (a|buy) \times 0.5\ (big|a) \times 1\ (house|big) = 0.0833$

模型质量评价

- 外部评价（Extrinsic measure）：
  - 一般用于比较两个模型A和B的质量
  - 将模型 A 和 B 应用于同一个任务（数据）
    - 拼写检查、机器翻译、语音识别等
  - 执行该任务（在同一个数据集中运行），分别得到模型A和B的表现
    - 有多少个错误的词被正确检测出来?
    - 有多少个词被正确翻译?
    - 有多少个音节被正确识别?
  - 通过比较准确率来比较模型A和B的质量
- 内部评价（Intrinsic measure）：
  - 对模型进行直接评价

模型质量评价

- Perplexity(困惑度，PP)
  - PP：P(W) 的几何平均值的倒数
  - 对于测试集W上的n-gram模型：

$$PP(W) = P(W)^{-\frac{1}{N}} = \left[\prod_{i=1}^{N} P(w_i \mid w_{i-n}, \ldots, w_{i-1})\right]^{-\frac{1}{N}}$$

- 对于给定的测试语料W
  - PP值越小，所用的LM越好
- 为什么？

模型质量评价

- 解释方式1:
  - 若W中都是正确的句子，则P(W)应该具有较高的概率（较小的PP值）

模型质量评价

- 解释方式2：
  - The Shannon Game:
    - 能正确预测下一个词的能力有多强？
      - When I eat pizza, I wipe off the ____
      - Many children are allergic to ____
      - I saw a ____
  - 直观上来讲：PP指在每次预测时可能的候选词的平均个数
  - PP越大，则表明该语言的不可预测性越高

困惑度与信息熵

- 一个词串$W=w_1,w_2,\ldots w_N$的信息熵：

$$H(W) = -\sum q(w_1 \ldots w_N)\log_2 q(w_1 \ldots w_N)$$

- 每个词w的平均信息熵：

$$H(w) = -\frac{1}{N}\sum q(w_1 \ldots w_N)\log_2 q(w_1 \ldots w_N)$$

- 语言L的信息熵：

$$H(L) = -\lim_{N\to\infty}\frac{1}{N}\sum q(w_1 \ldots w_N)\log_2 q(w_1 \ldots w_N)$$

$$= -\lim_{N\to\infty}\frac{1}{N}\log_2 q(w_1 \ldots w_N)$$

困惑度与信息熵

- 但q(.) 未知，在n-gram模型中，采用p作为q的估计，如果N足够大，则：

$$H(W) = -\frac{1}{N}\log_2 p(w_1 \ldots w_N)$$

- 因此：

$$PP(W) = 2^{H(W)}$$

- H越小，则PP越小，反之亦然

困惑度：实例

prediction	p_{LM}	-$\log_2 p_{\mathrm{LM}}$
$p_{\mathrm{LM}}(i \mid </s><s>)$	0.109043	3.197
$p_{\mathrm{LM}}(would \mid <s> i)$	0.144482	2.791
$p_{\mathrm{LM}}(like \mid i\ would)$	0.489247	1.031
$p_{\mathrm{LM}}(to \mid would\ like)$	0.904727	0.144
$p_{\mathrm{LM}}(commend \mid like\ to)$	0.002253	8.794
$p_{\mathrm{LM}}(the \mid to\ commend)$	0.471831	1.084
$p_{\mathrm{LM}}(rapporteur \mid commend\ the)$	0.147923	2.763
$p_{\mathrm{LM}}(on \mid the\ rapporteur)$	0.056315	4.150
$p_{\mathrm{LM}}(his \mid rapporteur\ on)$	0.193806	2.367
$p_{\mathrm{LM}}(work \mid on\ his)$	0.088528	3.498
$p_{\mathrm{LM}}(. \mid his\ work)$	0.290257	1.785
$p_{\mathrm{LM}}(</s> \mid work\ .)$	0.999990	0.000
	average	2.633671

困惑度：实例

- 不同阶的n-gram模型困惑度对比：

word	unigram	bigram	trigram	4-gram
i	6.684	3.197	3.197	3.197
would	8.342	2.884	2.791	2.791
like	9.129	2.026	1.031	1.290
to	5.081	0.402	0.144	0.113
commend	15.487	12.335	8.794	8.633
the	3.885	1.402	1.084	0.880
rapporteur	10.840	7.319	2.763	2.350
on	6.765	4.140	4.150	1.862
his	10.678	7.316	2.367	1.978
work	9.993	4.816	3.498	2.394
.	4.896	3.020	1.785	1.510
</s>	4.828	0.005	0.000	0.000
average	8.051	4.072	2.634	2.251
perplexity	265.136	16.817	6.206	4.758

困惑度：实例

- 不同的平滑算法困惑度对比：

Smoothing method	**bigram**	**trigram**	**4-gram**
Good-Turing	96.2	62.9	59.9
Witten-Bell	97.1	63.8	60.4
Modified Kneser-Ney	95.4	61.6	58.6
Interpolated Modified Kneser-Ney	94.5	59.3	54.0

- 实验语料

	Brown语料	**AP新闻**
训练语料规模	1,181,041词的前800,000词	13,994,528词
发展语料规模(模型选择、权重衰减、early stopping)	随后的200,000词	963,138词
测试语料规模	其余181,041词	963,071词
语料实际含的不同词	47,578(含标点、大小写不同、分割段落与文本的标记符)	148,721词
使用的词，即|V|	16,383去除频率小于等于3的	17,964词(进行一些合并)
学习率	初始$\varepsilon_0=10^{-3}$，之后衰减，按$\varepsilon_t=\varepsilon_0/(1+rt)$	
权重衰减惩罚	10^{-4}	10^{-5}
Early stopping	采用	没有
收敛	10-20epochs后	5epochs后

- 对比模型
  - Benchmark n-gram models
    - interpolated or smoothed trigram model (Jelinek and Mercer, 1980)
  - State-of-the-art n-gram models
    - back-off n-gram models with the Modified Kneser-Ney algorithm (Kneser and Ney, 1995, Chen and Goodman., 1999)
    - class-based n-gram models (Brown et al., 1992, Ney and Kneser, 1993, Niesler et al., 1998).

- Brown语料结果

	模型阶数	词类数	隐单元数	词表示维数	输入到输出的连接	是否与插值trigram混合 (权值均为0.5)		PP	
	n	c	h	m	direct	mix	train.	valid.	test.
MLP1	5		50	60	yes	no	182	284	268
MLP2	5		50	60	yes	yes		275	257
MLP3	5		0	60	yes	no	201	327	310
MLP4	5		0	60	yes	yes		286	272
MLP5	5		50	30	yes	no	209	296	279
MLP6	5		50	30	yes	yes		273	259
MLP7	3		50	30	yes	no	210	309	293
MLP8	3		50	30	yes	yes		284	270
MLP9	5		100	30	no	no	175	280	276
MLP10	5		100	30	no	yes		265	**252**
Del. Int.	3						31	352	336
Kneser-Ney back-off	3							334	323
Kneser-Ney back-off	4							332	321
Kneser-Ney back-off	5							332	321
class-based back-off	3	150						348	334
class-based back-off	3	200						354	340
class-based back-off	3	500						326	**312**
class-based back-off	3	1000						335	319
class-based back-off	3	2000						343	326
class-based back-off	4	500						327	312
class-based back-off	5	500						327	312

- 初步结论：
  - 更多上下文时(高阶语言模型)神经模型性能改善，而原有LM没有太多受益
  - NN模型的隐单元(有无以及数量变化)是有影响的
  - NN模型与原有LM的混合是有帮助的
  - 从输入到输出的直接连接是否有用(图中看不出，但是作者有如下讨论)：
    - 小语料时提供更好的泛化能力(很有限)，大语料时直接连接提供更快的收敛速度(2倍)

- AP语料结果

	n	h	m	direct	mix	train.	valid.	test.
MLP10	6	60	100	yes	yes		104	**109**
Del. Int.	3						126	132
Back-off KN	3						121	127
Back-off KN	4						113	119
Back-off KN	5						112	**117**

  – 由于语料规模大，所以只执行了5epochs迭代，结论和前面相似。

其它的语言模型

- Syntactic language models: using parse trees
- Class-based N-gram Model
- Topic-based N-gram Model
- Skip n-gram models: back-off to $p(w_n|w_{n-2})$

- 补充知识：
  - 似然函数及最大似然估计
  - 几种不同的实验设置

Likelihood

- In statistics, a **likelihood function** (often simply the **likelihood**) is a function of the parameters of a statistical model given data.

- The *likelihood* of a set of parameter values, θ, given outcomes x, is equal to the *probability* of those observed outcomes given those parameter values, that is

$$L(\theta|x)=P(x\,|\,\theta)$$

- The likelihood function is defined differently for discrete and continuous probability distributions.

Likelihood

- **Discrete probability distribution**
- Let X be a random variable with a discrete probability distribution p depending on a parameter θ. Then the function

$$L(\theta|x)=P(x=X\,|\,\theta)$$

- considered as a function of θ, is called the likelihood function (of θ, given the outcome x of the random variable X).

Maximum likelihood estimation

- 最大似然估计提供了一种给定观察数据来评估模型参数的方法
  - 即："模型已定，参数未知"
- 最大似然估计的假设：所有采样独立同分布
- 例：$x_1, \ldots, x_n$为独立同分布的采样，θ为模型参数，*f*为我们所使用的模型
- 则参数为θ的模型*f*可表示为：

$$f(x_1, x_2, \ldots, x_n \mid \theta) = f(x_1 \mid \theta) \times f(x_2 \mid \theta) \ldots, f(x_n \mid \theta)$$

- 似然函数定义为：

$$L(\theta \mid x_1, \ldots, x_n) = f(x_1, \ldots, x_n \mid \theta) = \prod_{i=1}^{n} f(x_i \mid \theta)$$

Maximum likelihood estimation

- 实际应用中常对两边取对数，得到：

$$\ln L(\theta \mid x_1, \ldots, x_n) = \sum_{i=1}^{n} \ln f(x_i \mid \theta) \qquad \hat{\ell} = \frac{1}{n} \ln L$$

- 其中$\ln L(\theta | x_1, \ldots, x_n)$称为对数似然，而$\hat{\ell}$称为平均对数似然
- 最大对数平均似然，得到：

$$\hat{\theta}_{mle} = \arg\max_{\theta \in \Theta} \hat{\ell}(\theta \mid x_1, \ldots, x_n)$$

为什么是MLE？

- 将一枚硬币抛n次，观测到的结果可以表示为概率p（p未知）的函数：

$$P(DATA \mid p) = P(X_1, \cdots, X_n \mid p) = \binom{n}{k} \prod_{i=1}^{n} P(X_i \mid p)$$

$$= \binom{n}{k} p^k (1-p)^{n-k} \propto \log\ p^k (1-p)^{n-k}$$

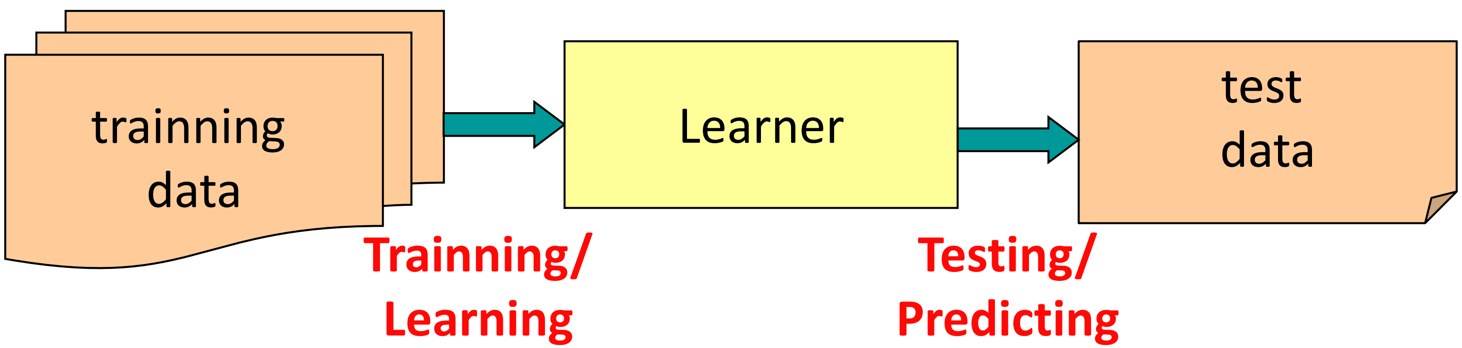
$$= k \log\ p + (n-k) \log(1-p) \equiv f(p)$$

- 令 f 相对于p的一阶导数为零，则：

$$0 = \frac{d}{dp} f(p) = \frac{k}{p} - \frac{n-k}{1-p}, \qquad (1-p)k = (n-k)p, \qquad p = \frac{k}{n}$$

几种不同的实验设置

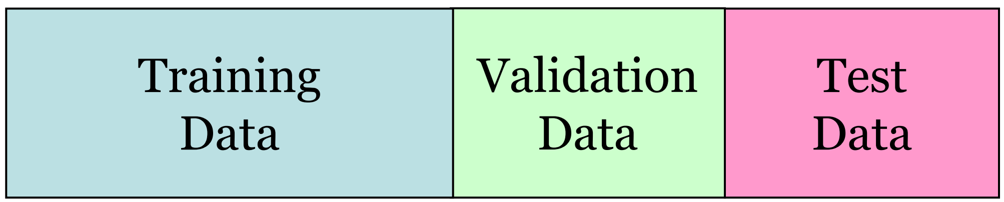
- **Setting 1: training + testing**
  - 将数据集分为两部分：训练集和测试集
  - 在训练集上进行模型学习和训练
  - 在测试集上进行模型评估



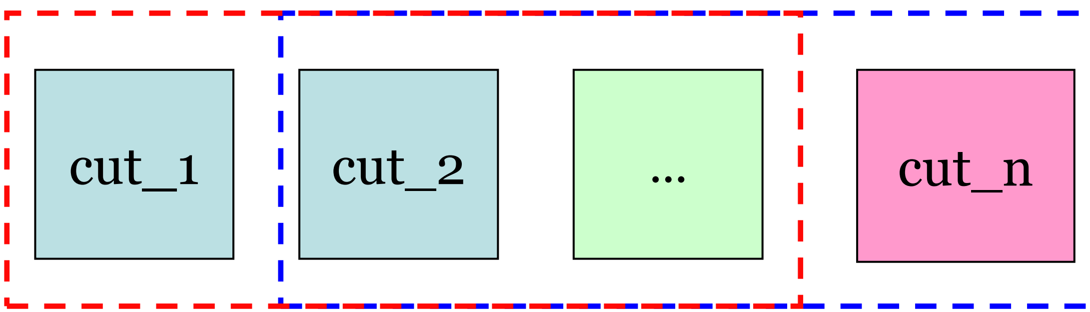
Test errors give an honest assessment of the error for future cases!

几种不同的实验设置

- **Setting 2: training + validation + testing**
  - 将数据集分为三部分：训练集、验证集(发展集)、测试集
  - 在训练集上进行模型训练
  - 在验证集(发展集) 上调整模型(超)参数
  - 在测试集上进行模型评估



- **Setting 3: k倍交叉验证(k-fold Cross-Validation)**



Next lecture

- Part-of-speech
- Hidden Markov Model