KNN算法确实具有惰性学习的性质。惰性学习，也称为延迟学习或基于实例的学习，是一种机器学习策略，其核心思想是在训练阶段不生成一个泛化的模型（如决策树或神经网络），而是将训练数据保存下来，并在收到新的查询样本时，基于与训练样本的相似性（通常是距离度量）来进行预测。

A *k*-Nearest Neighbor Based Algorithm for Multi-label Classification

Lazy Learning
的思想体现在，对于一个新的未知样本，算法会首先找到其在训练集中的K个最近邻，然后基于这些近邻的信息来进行预测或分类。这个过程是在预测阶段进行的，而不是在训练阶段

Min-Ling Zhang and Zhi-Hua Zhou
National Laboratory for Novel Software Technology,
Nanjing University, Nanjing 210093, China
Email: {zhangml, zhouzh}@lamda.nju.edu.cn

***Abstract*— In multi-label learning, each instance in the training set is associated with a set of labels, and the task is to output a label set whose size is unknown a priori for each unseen instance. In this paper, a multi-label lazy learning approach named ML-*k*NN is presented, which is derived from the traditional *k*-nearest neighbor (*k*NN) algorithm. In detail, for each new instance, its *k* nearest neighbors are firstly identified. After that, according to the label sets of these neighboring instances, maximum a posteriori (MAP) principle is utilized to determine the label set for the new instance. Experiments on a real-world multi-label bioinformatic data show that ML-*k*NN is highly comparable to existing multi-label learning algorithms.**

I. Introduction

Multi-label classification tasks are ubiquitous in real-world problems. For example, in text categorization, each document may belong to several predefined topics; in bioinformatics, one protein may have many effects on a cell when predicting its functional classes. In either case, instances in the training set are each associated with a set of labels, and the task is to output the label set for the unseen instance whose set size is not available *a priori*.

Traditional two-class and multi-class problems can both be cast into multi-label ones by restricting each instance to have only one label. However, the generality of multi-label problem makes it more difficult to learn. An intuitive approach to solve multi-label problem is to decompose it into multiple independent binary classification problems (one per category). But this kind of method does not consider the correlations between the different labels of each instance. Fortunately, several approaches specially designed for multi-label classification have been proposed, such as multi-label text categorization algorithms [1], [2], [3], multi-label decision trees [4], [5] and multi-label kernel method [6], etc. However, multi-label lazy learning approach is still not available. In this paper, this problem is addressed by a multi-label classification algorithm named ML-*k*NN, i.e. Multi-Label *k*-Nearest Neighbor, which is derived from the popular *k*-nearest neighbor (*k*NN) algorithm [7]. ML-*k*NN first identifies the *k* nearest neighbors of the test instance where the label sets of its neighboring instances are obtained. After that, maximum a posteriori (MAP) principle is employed to predict the set of labels of the test instance.

The rest of this paper is organized as follows. Section 2 reviews previous works on multi-label learning and summarizes different evaluation criteria used in this area. Section 3 presents the ML-*k*NN algorithm. Section 4 reports experimental results on a real-world multi-label bioinformatic data. Finally, Section 5 concludes and indicates several issues for future work.

II. Multi-label Learning

Research of multi-label learning was initially motivated by the difficulty of concept ambiguity encountered in text categorization, where each document may belong to several topics (labels) simultaneously. One famous approach to solving this problem is BoosTexter proposed by Schapire and Singer [2] , which is in fact extended from the popular ensemble learning method AdaBoost [8]. In the training phase, BoosTexter maintains a set of weights over both training examples and their labels, where training examples and their corresponding labels that are hard (easy) to predict correctly get incrementally higher (lower) weights. Following the work of BoosTexter, multi-label learning has attracted many attentions from machine learning researchers.

In 1999, McCallum [1] proposed a Bayesian approach to multi-label document classification, where a mixture probabilistic model is assumed to generate each document and EM [9] algorithm is utilized to learn the mixture weights and the word distributions in each mixture component. In 2001, through defining a special cost function based on *Ranking Loss* (as shown in Eq.(5)) and the corresponding margin for multi-label models, Elisseeff and Weston [6] proposed a kernel method for multi-label classification. In the same year, Clare and King [4] adapted C4.5 decision tree [10] to handle multi-label data through modifying the definition of entropy. One year later, using independent word-based *Bag-of-Words* representation [11], Ueda and Saito [3] presented two types of probabilistic generative models for multi-label text called parametric mixture models (PMM1, PMM2), where the basic assumption under PMMs is that multi-label text has a mixture of characteristic words appearing in single-label text that belong to each category of the multi-categories. In the same year, Comité et al. [5] extended alternating decision tree [12] to handle multi-label data, where the AdaBoost.MH algorithm proposed by Schapire and Singer [13] is employed to train the multi-label alternating decision tree. In 2004, Boutell et al. [14] applied multi-label learning techniques to scene classification. They decomposed the multi-label learning problem into multiple independent binary classification problems (one per category), where each example associated with label set Y will be regarded as positive example when building classifier for class $y \in Y$ while regarded as negative example when building classifier for class $y \notin Y$.

It is worth noting that in multi-label learning paradigm, various evaluation criteria have been proposed to measure the performance of a multi-label learning system. Let $\mathcal{X} = \mathcal{R}^d$ be the d-dimensional instance domain and let $\mathcal{Y} = \{1, 2, ..., Q\}$ be a set of labels or classes. Given a learning set $S =< (x_1, Y_1), ..., (x_m, Y_m) >\in (\mathcal{X} \times 2^{\mathcal{Y}})^m$ i.i.d. drawn from an unknown distribution D, the goal of the learning system is to output a multi-label classifier $h : \mathcal{X} \to 2^{\mathcal{Y}}$ which optimizes some specific evaluation criterion. However, in most cases, the learning system will produce a *ranking* function of the form $f : \mathcal{X} \times \mathcal{Y} \to \mathcal{R}$ with the interpretation that, for a given instance x, the labels in $\mathcal{Y}$ should be ordered according to $f(x, \cdot)$. That is, a label l_1 is considered to be ranked higher than l_2 if $f(x, l_1) > f(x, l_2)$. If Y is the associated label set for x, then a successful learning system will tend to rank labels in Y higher than those not in Y. Note that the corresponding multi-label classifier $h(\cdot)$ can be conveniently derived from the ranking function $f(\cdot, \cdot)$:

$$h(x) = \{l | f(x, l) > t(x), l \in \mathcal{Y}\} \tag{1}$$

where $t(x)$ is the threshold function which is usually set to be the zero constant function.

Based on the above notations, several evaluation criteria can be defined in multi-label learning as shown in [2]. Given a set of multi-label instances $S = \{(x_1, Y_1), ..., (x_m, Y_m)\}$, a learned ranking function $f(\cdot, \cdot)$ and the corresponding multi-label classifier $h(\cdot)$, the first evaluation criterion to be introduced is the so-called *Hamming Loss* defined as:

$$HL_S(h) = \frac{1}{m} \sum_{i=1}^{m} \frac{1}{Q} |h(x_i) \Delta Y_i| \tag{2}$$

where Δ stands for the symmetric difference between two sets. The smaller the value of $HL_S(h)$, the better the classifier's performance. When $|Y_i| = 1$ for all instances, a multi-label system is in fact a multi-class single-label one and the Hamming Loss is $\frac{2}{Q}$ times the loss of the usual classification error.

While Hamming Loss is based on the multi-label classifier $h(\cdot)$, the following measurements will be defined based on the ranking function $f(\cdot, \cdot)$. The first ranking-based measurement to be considered is *One-error*:

$$One\text{-}err_S(f) = \frac{1}{m} \sum_{i=1}^{m} H(x_i), \text{ where}$$
$$H(x_i) = \begin{cases} 0, & \text{if } \arg\max_{k \in \mathcal{Y}} f(x_i, k) \in Y_i \\ 1, & \text{otherwise} \end{cases} \tag{3}$$

The smaller the value of $One{-}err_S(f)$, the better the performance. Note that, for single-label classification problems, the One-error is identical to ordinary classification error.

The second ranking-based measurement to be introduced is *Coverage* defined as:

$$Coverage_S(f) = \frac{1}{m} \sum_{i=1}^{m} |C(x_i)| - 1, \text{where}$$
$$C(x_i) = \{l | f(x_i, l) \geq f(x_i, l_i'), l \in \mathcal{Y}\} \quad \text{and}$$
$$l_i' = \arg\min_{k \in Y_i} f(x_i, k) \tag{4}$$

It measures how far we need, on the average, to go down the list of labels in order to cover all the possible labels assigned to an instance. The smaller the value of $Coverage_S(f)$, the better the performance.

Let $\overline{Y}$ denote the complementary set of Y in $\mathcal{Y}$, another ranking-based measurement named *Ranking Loss* is defined as:

$$RL_S(f) = \frac{1}{m} \sum_{i=1}^{m} \frac{1}{|Y_i||\overline{Y_i}|} |R(x_i)|, \text{ where } R(x_i) =$$
$$\{(l_1, l_0) | f(x_i, l_1) \leq f(x_i, l_0), (l_1, l_0) \in Y_i \times \overline{Y}_i\} \tag{5}$$

It represents the average fraction of pairs that are not correctly ordered. The smaller the value of $RL_S(f)$, the better the performance.

The fourth evaluation criterion for the ranking function is *Average Precision*, which is originally used in information retrieval (IR) systems to evaluate the document ranking performance for query retrieval [15]. Nevertheless, it is used here to measure the effectiveness of the label rankings:

$$Ave\_prec_S(f) = \frac{1}{m} \sum_{i=1}^{m} \frac{1}{|Y_i|} P(x_i), \text{ where}$$
$$P(x_i) = \sum_{k \in Y_i} \frac{|\{l | f(x_i, l) \geq f(x_i, k), l \in Y_i\}|}{|\{l | f(x_i, l) \geq f(x_i, k), l \in \mathcal{Y}\}|} \tag{6}$$

In words, this measurement evaluates the average fraction of labels ranked above a particular label $l \in Y_i$ which actually are in Y_i. Note that when $Ave\_prec_S(f) = 1$, the learning system achieves the perfect performance. The bigger the value of $Ave\_prec_S(f)$, the better the performance.

III. ML-*k*NN

As reviewed in the above Section, although there have been several learning algorithms specially designed for multi-label learning, developing lazy learning approach for multi-label problems is still an unsolved issue. In this section, a novel k-nearest neighbor based method for multi-label classification named ML-*k*NN is presented. To begin, several notations are introduced in addition to those used in Section 2 to simplify the derivation of ML-*k*NN.

Given an instance x and its associated label set $Y_x \subseteq \mathcal{Y}$, suppose k nearest neighbors are considered in the *k*NN method. Let $\vec{y}_x$ be the category vector for x, where its l-th component $\vec{y}_x(l)$ $(l \in \mathcal{Y})$ takes the value of 1 if $l \in Y_x$ and 0 otherwise. In addition, let $N(x)$ denote the index set of the k nearest neighbors of x identified in the training set.

training set S,test instance t,s Laplace smoothing

```
[$\vec{y}_t$, $\vec{r}_t$]=ML-kNN($S$, $k$, $t$, $s$)
%Computing the prior probabilities $P(H_i^l)$
(1)  for $l \in \mathcal{Y}$ do
(2)      $P(H_1^l) = (s + \sum_{i=1}^{m} \vec{y}_{x_i}(l)) / (s \times 2 + m)$ ;
(3)      $P(H_0^l) = 1 - P(H_1^l)$;
%Computing the posterior probabilities $P(E_j^l|H_i^l)$
(4)  Identify $N(x_i),\ i \in \{1, \ldots, m\}$;
(5)  for $l \in \mathcal{Y}$ do
(6)      for $j \in \{0, \ldots, k\}$ do
(7)          $c[j] = 0; c'[j] = 0$;
(8)      for $i \in \{1, \ldots, m\}$ do
(9)          $\delta = \vec{C}_{x_i}(l) = \sum_{a \in N(x_i)} \vec{y}_{x_a}(l)$;
(10)         if $(\vec{y}_{x_i}(l) == 1)$ then $c[\delta] = c[\delta] + 1$;
(11)                              else $c'[\delta] = c'[\delta] + 1$;
(12)     for $j \in \{0, \ldots, k\}$ do
(13)         $P(E_j^l|H_1^l) = \frac{s + c[j]}{s \times (k+1) + \sum_{p=0}^{k} c[p]}$;
(14)         $P(E_j^l|H_0^l) = \frac{s + c'[j]}{s \times (k+1) + \sum_{p=0}^{k} c'[p]}$;
%Computing $\vec{y}_t$ and $\vec{r}_t$
(15) Identify $N(t)$;
(16) for $l \in \mathcal{Y}$ do
(17)     $\vec{C}_t(l) = \sum_{a \in N(t)} \vec{y}_{x_a}(l)$;
(18)     $\vec{y}_t(l) = \arg\max_{b \in \{0,1\}} P(H_b^l) P(E_{\vec{C}_t(l)}^l | H_b^l)$;
(19)     $\vec{r}_t(l) = P(H_1^l | E_{\vec{C}_t(l)}^l)$
             $= P(H_1^l) P(E_{\vec{C}_t(l)}^l | H_1^l) / P(E_{\vec{C}_t(l)}^l)$
             $= \frac{P(H_1^l) P(E_{\vec{C}_t(l)}^l | H_1^l)}{\sum_{b \in \{0,1\}} P(H_b^l) P(E_{\vec{C}_t(l)}^l | H_b^l)}$;
```

Fig. 1. Pseudo code of ML-*k*NN.

Thus, based on the label sets of these neighbors, a *membership counting* vector can be defined as:

$$\vec{C}_x(l) = \sum_{a \in N(x)} \vec{y}_{x_a}(l),\ l \in \mathcal{Y} \tag{7}$$

where $\vec{C}_x(l)$ counts how many neighbors of x belong to the l-th class.

For each test instance t, ML-kNN first identifies its k nearest neighbors $N(t)$. Let H_1^l be the event that t has label l, while H_0^l be the event that t has not label l. Furthermore, let E_j^l ($j \in \{0, \ldots, k\}$) denote the event that, among the k nearest neighbors of t, there are exactly j instances which have label l. Therefore, based on the membership counting vector $\vec{C}_t$, the category vector $\vec{y}_t$ is determined using the following maximum a posteriori principle:

$$\vec{y}_t(l) = \arg\max_{b \in \{0,1\}} P(H_b^l | E_{\vec{C}_t(l)}^l),\ l \in \mathcal{Y} \tag{8}$$

Using the Bayesian rule, Eq.(8) can be rewritten as:

$$\begin{aligned} \vec{y}_t(l) &= \arg\max_{b \in \{0,1\}} \frac{P(H_b^l) P(E_{\vec{C}_t(l)}^l | H_b^l)}{P(E_{\vec{C}_t(l)}^l)} \\ &= \arg\max_{b \in \{0,1\}} P(H_b^l) P(E_{\vec{C}_t(l)}^l | H_b^l) \end{aligned} \tag{9}$$

Note that the prior probabilities $P(H_b^l)$ ($l \in \mathcal{Y}$, $b \in \{0,1\}$) and the posterior probabilities $P(E_j^l|H_b^l)$ ($j \in \{0, \ldots, k\}$) can all be estimated from the training set S.

Figure 1 illustrates the complete description of ML-kNN. The meanings of the input arguments S, k, t and the output argument $\vec{y}_t$ are the same as described previously. While the input argument s is a smoothing parameter controlling the strength of uniform prior (In this paper, s is set to be 1 which yields the Laplace smoothing). $\vec{r}_t$ is a real-valued vector calculated for ranking labels in $\mathcal{Y}$, where $\vec{r}_t(l)$ corresponds to the posterior probability $P(H_1^l | E_{\vec{C}_t(l)}^l)$. As shown in Figure 1, based on the training instances, steps from (1) to (3) estimate the prior probabilities $P(H_i^l)$. Steps from (4) to (14) estimate the posterior probabilities $P(E_j^l|H_i^l)$, where $c[j]$ used in each iteration of l counts the number of training instances with label l whose k nearest neighbors contain exactly j instances with label l. Correspondingly, $c'[j]$ counts how many training instances without label l whose k nearest neighbors contain exactly j instances with label l. Finally, using the Bayesian rule, steps from (15) to (19) compute the algorithm's outputs based on the estimated probabilities.

IV. Experiments

A real-world Yeast gene functional data which has been studied in the literatures [6], [16] is used for experiments. Each gene is associated with a set of functional classes whose maximum size can be potentially more than 190. In order to make it easier, Elisseeff and Weston [6] preprocessed the data set where only the known structure of the functional classes are used. Actually, the whole set of functional classes is structured into hierarchies up to 4 levels deep (see http://mips.gsf.de/proj/yeast/catalogues/funcat/ for more details). In this paper, as what has been done in the literature [6], only functional classes in the top hierarchy are considered. For fair comparison, the same kind of data set division used in the literature [6] is adopted. In detail, there are 1,500 genes in the training set and 917 in the test set. The input dimension is 103. There are 14 possible class labels and the average number of labels for all genes in the training set is 4.2 ± 1.6.

Table I presents the performance of ML-kNN on the Yeast data when different values of k (number of neighbors) are considered. It can be found that the value of k doesn't

TABLE I

THE PERFORMANCE OF ML-*k*NN ON THE YEAST DATA WITH DIFFERENT VALUES OF k (NUMBER OF NEIGHBORS).

Evaluation	No. of neighbors considered			
Criterion	*k*=6	*k*=7	*k*=8	*k*=9
Hamming Loss	0.197	0.197	0.197	0.197
One-error	0.241	**0.239**	0.248	0.251
Coverage	6.374	**6.302**	6.357	6.424
Ranking Loss	0.170	**0.168**	0.171	0.173
Average Precision	0.758	**0.761**	0.756	0.755

TABLE II

PERFORMANCE ON THE YEAST DATA FOR OTHER MULTI-LABEL LEARNING ALGORITHMS.

Evaluation	Algorithm		
Criterion	Rank-SVM	ADTBoost.MH	BoosTexter
Hamming Loss	0.196	0.213	0.237
One-error	0.225	0.245	0.302
Coverage	6.717	6.502	N/A
Ranking Loss	0.179	N/A	0.298
Average Precision	0.763	0.738	0.717

significantly affect the classifier's Hamming Loss, while ML-*k*NN achieves best performance on the other four ranking-based criteria with $k = 7$.

Table II shows the experimental results on the Yeast data of several other multi-label learning algorithms introduced in Section 2 . It is worth noting that a re-implemented version of Rank-SVM [6] is used in this paper, where polynomial kernels with degree 8 are chosen and the cost parameter C is set to be 1. As for ADTBoost.MH [5], the number of boosting steps is set to 30 considering that the performance of the boosting algorithm rarely changes after 30 iterations. Besides, the results of BoosTexter [2] shown in Table II are those reported in the literature [6].

As shown in Table I and Table II, the performance of ML-*k*NN is comparable to that of Rank-SVM. Moreover, it is obvious that both algorithms perform significantly better than ADTBoost.MH and BoosTexter. One possible reason for the poor results of BoosTexter may be due to the simple decision function realized by this method [6].

V. CONCLUSION

In this paper, the problem of designing multi-label lazy learning approach is addressed, where a *k*-nearest neighbor based method for multi-label classification named ML-*k*NN is proposed. Experiments on a multi-label bioinformatic data show that the proposed algorithm is highly competitive to other existing multi-label learners.

Nevertheless, the experimental results reported in this paper are rather preliminary. Thus, conducting more experiments on other multi-label data sets to fully evaluate the effectiveness of ML-*k*NN will be an important issue to be explored in the near future. On the other hand, adapting other traditional machine learning approaches such as neural networks to handle multi-label data will be another interesting issue to be investigated.

ACKNOWLEDGMENT

Many thanks to A. Elisseeff and J. Weston for providing the authors with the Yeast data and the implementation details of Rank-SVM. This work was supported by the National Natural Science Foundation of China under the Grant No. 60473046.

REFERENCES

[1] A. McCallum, "Multi-label text classification with a mixture model trained by EM," in *Working Notes of the AAAI'99 Workshop on Text Learning*, Orlando, FL, 1999.

[2] R. E. Schapire and Y. Singer, "Boostexter: a boosting-based system for text categorization," *Machine Learning*, vol. 39, no. 2/3, pp. 135–168, 2000.

[3] N. Ueda and K. Saito, "Parametric mixture models for multi-label text," in *Advances in Neural Information Processing Systems 15*, S. Becker, S. Thrun, and K. Obermayer, Eds. Cambridge, MA: MIT Press, 2003, pp. 721–728.

[4] A. Clare and R. D. King, "Knowledge discovery in multi-label phenotype data," in *Lecture Notes in Computer Science 2168*, L. D. Raedt and A. Siebes, Eds. Berlin: Springer, 2001, pp. 42–53.

[5] F. D. Comité, R. Gilleron, and M. Tommasi, "Learning multi-label altenating decision tree from texts and data," in *Lecture Notes in Computer Science 2734*, P. Perner and A. Rosenfeld, Eds. Berlin: Springer, 2003, pp. 35–49.

[6] A. Elisseeff and J. Weston, "A kernel method for multi-labelled classification," in *Advances in Neural Information Processing Systems 14*, T. G. Dietterich, S. Becker, and Z. Ghahramani, Eds. Cambridge, MA: MIT Press, 2002, pp. 681–687.

[7] D. W. Aha, "Special AI review issue on lazy learning," *Artificial Intelligence Review*, vol. 11, 1997.

[8] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," in *Lecture Notes in Computer Science 904*, P. M. B. Vitányi, Ed. Berlin: Springer, 1995, pp. 23–37.

[9] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistics Society -B*, vol. 39, no. 1, pp. 1–38, 1977.

[10] J. R. Quinlan, *C4.5: Programs for Machine Learning*. San Mateo, California: Morgan Kaufmann, 1993.

[11] S. T. Dumais, J. Platt, D. Heckerman, and M. Sahami, "Inductive learning algorithms and representation for text categorization," in *Proc. of the 7th ACM International Conference on Information and Knowledge Management (CIKM'98)*, Bethesda, MD, 1998, pp. 148–155.

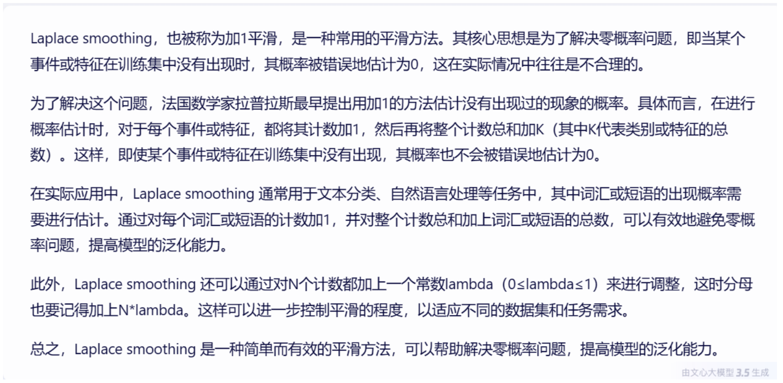
[12] Y. Freund and L. Mason, "The alternating decision tree learning algorithm," in *Proc. of the 16th International Conference on Machine Learning (ICML'99)*, Bled, Slovenia, 1999, pp. 124–133.

[13] R. E. Schapire and Y. Singer, "Improved boosting algorithms using confidence-rated predictions," in *Proc. of the 11th Annual Conference on Computational Learning Theory (COLT'98)*, New York, 1998, pp. 80–91.

[14] M. R. Boutell, J. Luo, X. Shen, and C. M. Brown, "Learning multi-label scene classification," *Pattern Recognition*, vol. 37, no. 9, pp. 1757–1771, 2004.

[15] G. Salton, "Developments in automatic text retrieval," *Science*, vol. 253, pp. 974–980, 1991.

[16] P. Pavlidis, J. Weston, J. Cai, and W. N. Grundy, "Combining microarray expression data and phylogenetic profiles to learn functional categories using support vector machines," in *Proceedings of the 5th Annual International Conference on Computational Biology*, Montréal, Canada, 2001, pp. 242–248.



Laplace smoothing，也被称为加1平滑，是一种常用的平滑方法。其核心思想是为了解决零概率问题，即当某个事件或特征在训练集中没有出现时，其概率被错误地估计为0，这在实际情况中往往是不合理的。

为了解决这个问题，法国数学家拉普拉斯最早提出用加1的方法估计没有出现过的现象的概率。具体而言，在进行概率估计时，对于每个事件或特征，都将其计数加1，然后再将整个计数总和加K（其中K代表类别或特征的总数）。这样，即使某个事件或特征在训练集中没有出现，其概率也不会被错误地估计为0。

在实际应用中，Laplace smoothing 通常用于文本分类、自然语言处理等任务中，其中词汇或短语的出现概率需要进行估计。通过对每个词汇或短语的计数加1，并对整个计数总和加上词汇或短语的总数，可以有效地避免零概率问题，提高模型的泛化能力。

此外，Laplace smoothing 还可以通过对N个计数都加上一个常数lambda（0≤lambda≤1）来进行调整，这时分母也要记得加上N*lambda。这样可以进一步控制平滑的程度，以适应不同的数据集和任务需求。

总之，Laplace smoothing 是一种简单而有效的平滑方法，可以帮助解决零概率问题，提高模型的泛化能力。