Selection of Import Vectors via Binary Particle Swarm Optimization and Cross-Validation for Kernel Logistic Regression

Kenji TANAKA, Takio KURITA, Tohru KAWABE

Abstract—Kernel logistic regression (KLR) is a powerful discriminative algorithm. It has similar loss function and algorithmic structure to the kernel support vector machine (SVM). Recently, Zhu and Hastie proposed the import vector machine (IVM) in which a subset of the input vectors of KLR are selected by minimizing the regularized negative log-likelihood to improve the generalization performance and to reduce computation cost. In this paper, two modifications of the original IVM are proposed. The cross-validation based criterion is used to select import vectors instead of the likelihood based criterion. Also binary particle swarm optimization is used to select good subset instead of the greedy stepwise algorithm of the original IVM. Through the comparison experiment, the improvement of the generalization performance of the proposed algorithm was confirmed.

I. INTRODUCTION

Logistic regression (LR) is a well-known statistical model for binary classification. It is known as one of the generalized linear models. It directly estimates the posterior probability. The parameters of LR can be trained by the iteratively reweighted least squares algorithm. It can naturally be generalized to the multiclass case through multi-logit regression. Its extension to non-linear by using kernel trick is kernel logistic regression (KLR) (with regularized function).

The criterion to determine the parameters of LR or KLR is similar to that of support vector machine (SVM) [1]. But they do not have sparseness. Although the support vectors of SVM are automatically obtained by the optimization of the criterion, the parameters of LR or KLR are nonzero because of the small difference between the criteria. It means that they do not have the advantage of less storage and efficient computation like SVM. Therefore, Zhu and Hastie proposed a novel algorithm called the import vector machine (IVM) [2] in which a good subset of the input vectors of KLR are automatically selected. The authors showed that IVM could give a comparable recognition performance to the kernel SVM with fewer import vectors than the support vectors of the kernel SVM.

In this paper we propose two modifications of the original IVM. In the original IVM, the regularized negative log-likelihood is used as the criterion to select the import vectors. Through experiments we found that this criterion often selected larger number of import vectors than the subset with minimum generalization error. To prevent this tendency of overfitting, cross-validation based criterion is used instead of the likelihood based criterion of the original KLR. The import vectors of the original IVM are selected one by one by the greedy stepwise optimization algorithm. In that algorithm, computational cost increase as number of samples increases. It does not guaranty the global optimal. To search a good subset of import vectors, we use the binary particle swarm optimization (BPSO) [3]. Particle Swarm Optimization (PSO) is a method for optimization of continuous nonlinear functions, proposed by Kennedy and Eberhart [4] in 1995. The algorithm was inspired by social behavior of flocks of birds when they are searching for food. BPSO is a modification of the PSO algorithm for solving problems with binary values, such as combinational optimization problems. It was also developed by Kennedy and Eberhart [3].

II. KERNEL LOGISTIC REGRESSION AND IMPORT VECTOR MACHINE

A. Kernel Logistic Regression

Logistic regression can be applied to both real and binary responses, and its output can be processed as posterior probabilities. Let \( D = \{ < x_1, y_1 >, \ldots, < x_n, y_n > \} \) be a given training set of samples, with \( y \in \{ 0, 1 \} \) the class label for data \( x^T = (x_1, x_2, \ldots, x_m) \in \mathcal{X} \), and probabilities are shown as follows:

\[
p(y|x) = \frac{1}{1 + \exp(-f(x))},
\]

\[
f(x) = \sum_{j=1}^{m} w_j x_j + b.
\]

Here, \( w^T = (w_1, w_2, \ldots, w_m) \) and \( b \) are the weight vector and bias of the decision hyperplane, respectively. The logistic regression is trained by minimizing negative log-liklyhood (NLL) as

\[
w = \arg \min_w \frac{1}{n} \sum_{i=1}^{n} \{ \log(1 + \exp(f(x_i))) - y_i f(x_i) \}.
\]

A non-linear form of logistic regression is known as kernel logistic regression (KLR). It can be obtained by using features in a high-dimensional feature space instead of that in \( \mathcal{X} \) on a conventional logistic regression model. That high-dimensional feature space is generated by the kernel function \( K(\cdot) \) that evaluates the inner product between the images of input vectors in the feature space \( \mathcal{X} \), i.e., \( K(x, x') = \phi(x) \cdot \phi(x') \).
\( \phi(x') \); the feature space produced by the kernel function is defined as the reproducing kernel Hilbert space (RKHS) \( \mathcal{H}_k \). The kernel function used here is the radial basis function (RBF),

\[
K(x, x') = \exp\left(-\frac{||x - x'||^2}{2\sigma^2}\right). \tag{4}
\]

Given \( n \) samples as input vectors, a conventional logistic regression model is then constructed in the feature space \( \mathcal{H}_k \), such that

\[
p(y|x) = \frac{1}{1 + \exp(-f(x))}, \tag{5}
\]

\[
f(x) = \sum_{j=1}^{n} a_j K(x_j, x) + b. \tag{6}
\]

Here, \( \{a_1, a_2, ..., a_n\} \) are the coeffients of the kernel expansion that satisfy the representer theorem [5], [6],

\[
\sum_{j=1}^{n} w_j \cdot \phi(x_j) = \sum_{j=1}^{n} a_j \phi(x_i) \cdot \phi(x) = \sum_{j=1}^{n} a_j K(x_j, x) \tag{7}
\]

To simplify notation, let \( \mathbf{a}^T = (a_1, a_2, ..., a_n, b) \), \( \mathbf{k}^T = (K(x_1, x), ..., K(x_n, x), 1) \) including a bias term \( b \). The optimal model parameter \( \mathbf{a} \) is found by minimizing a cost function represented by the regularised NLL of the data,

\[
\min_{\mathbf{a}} \frac{1}{n} \sum_{i=1}^{n} \left\{ \log(1 + \exp(f(x_i))) - y_i f(x_i) \right\} + \lambda \|f\|^2_{\mathcal{H}_k}. \tag{8}
\]

Here, \( \lambda \) is a regularisation parameter controlling the bias-variance trade-off. Furthermore, that Eq. 8 represents a convex optimisation problem is straightforward, and so there is only a single, global minimum. The optimal model parameter \( \mathbf{a} \) can be found using Newton-Raphson method or equivalently an iteratively re-weighted least-squares (IRWLS) procedure. With a slight modification to accommodate the regularization term, kernel logistic regression models can also be trained using IRWLS. The coefficients \( \mathbf{a} \) in each iteration are given by the solution of a weighted least squares problem,

\[
\mathbf{a}^{new} = \frac{1}{n} (\mathbf{K}^T \mathbf{W} \mathbf{K} + \lambda \mathbf{K})^{-1} \mathbf{K}^T \mathbf{W} \mathbf{z}, \tag{9}
\]

\[
\mathbf{z} = \frac{1}{n} [\mathbf{K} \mathbf{a}^{old} + W^{-1}(y - \mathbf{p})], \tag{10}
\]

here

\[
\mathbf{K} = (\mathbf{k}_1, \mathbf{k}_2, ..., \mathbf{k}_n)^T, \tag{11}
\]

the so-called gram matrix, and

\[
W = \text{diag}\{p_1(1-p_1), p_2(1-p_2), ..., p_n(1-p_n)\}. \tag{12}
\]

Then Eq. 9 is repeated until convergence.

The relationship between the support vector machine (SVM) and KLR in RKHS has been noted in many studies [2], [6], [7], [8]. The SVM is well known for its good generalization ability in two-class classification and it builds a sparse classification model with support vectors by maximizing the margin. Fitting the SVM is equivalent to

\[
\min_{\mathbf{a}} \frac{1}{n} \sum_{i=1}^{n} \left[\min\left\{1 - t_i f(x_i), 0\right\}\right] + \frac{\lambda}{2} ||f||^2_{\mathcal{H}_k}, \tag{13}
\]

where \( t = \{-1, 1\} \) is class label, and

\[
|h|_+ = \begin{cases} h & \text{if } h > 0 \\ 0 & \text{otherwise}. \end{cases} \tag{14}
\]

The SVM minimizes the loss function with a regularized function. Furthermore, the loss function \( 1 - t_i f(x_i) \) has a similar shape to that of KRL in Eq. 8 (Fig.1). Because of the similarity between the two loss functions, the fitted function of KLR performs similarly to that of the SVM for classification [2].

**B. Import Vector Machine**

Two advantages are provided by substituting the regularized NLL loss function for the SVM loss function: first, KLR gives a natural estimate of the probability \( p(x) = 1/[1+\exp(f(x))] \), while the SVM only estimates class labels by \( \text{sign}(f(x)) \); second, KLR can naturally be generalized to the multiclass case through kernel multi-logit regression. However, because KLR differs from the SVM in the loss function, it no longer has model sparseness; all the \( a_i \) in Eq. 6 are nonzero. KLR does not allow for data compression and does not have the advantages of less storage and quicker evaluation.

Therefore, Zhu and Hastie proposed a novel method the import vector machine (IVM) that finds a submodel to approximate the full model given by KLR [2]. The submodel has the form:

\[
f(x) = \sum_{j \in \mathcal{S}} a_j K(x_j, x) + b. \tag{15}
\]

Here \( \mathcal{S} \) is a subset of the training data \( \{x_1, x_2, ..., x_n\} \), and Zhu and Hastie call data in \( \mathcal{S} \) import points. Import points
are selected using greedy forward strategy: add data into subset \( S \) one by one, evaluate each model, and select the one minimizing regularized NLL most as the import point. The advantage of this submodel is that the computational cost for evaluation can be reduced, especially for large training datasets, while not weakening classification [2]. We show a basic IVM algorithm in Algorithm 1. The stopping rule at line 10 is defined as that improve rate \( \frac{\|H_k - H_k - \Delta k\|}{\|H_k\|} < \epsilon \), where \( \Delta k \) and \( \epsilon \) are parameters chosen by the user (e.g. \( \Delta k = 1, \epsilon = 0.001 \)).

**Algorithm 1 Import Vector Machine**

1: Let \( S = \emptyset, \mathcal{L} = \{x_1, ..., x_n\}, k = 1. \\
2: repeat \\
3: for all \( x_l \in \mathcal{L} \) do \\
4: \[ f_i(x) = \sum_{x_i \in S \cup \{x_l\}} a_i K(x_i, x) + b \]
5: minimize 
\[ H(x_l) = \frac{1}{n} \{ 1^T \log[1 + \exp(K(x_l^1, a))] - y^T K(x_l^1, a) \} + \frac{1}{2} a^T K(x_l^2, a) \]
where 
\[ K_l^1 = (K(x_i, x'_i))_{n \times k}^1, \]
\[ x_i \in \{x_1, ..., x_n\} \text{, } x'_i \in S \cup \{x_l\} \]
and 
\[ K_l^2 = (K(x_i, x'_i))_{k \times k}^2, \text{ } x_i, x'_i \in S \cup \{x_l\} \]
6: end for \\
7: \( x_l^* = \arg \min \{H(\mathcal{L})\} \)
8: \( S = S \cup \{x_l^*\}, \mathcal{L} = \mathcal{L} \setminus \{x_l^*\}, H_k = H(x_l^*) \)
9: \( k = k + 1 \)
10: until \( H_k \) has converged

### III. Selection of Import Vectors by the Cross-Validational Metric

The \( k \)-fold cross-validation [9] is a practical and well-known method for evaluating the generalization performance of a classifier. It evaluates models more directly than other theoretical methods based on likelihood (AIC, MDL, etc). We use \( k \)-fold cross-validation as a criterion to select import points in IVM instead of regularized NLL. It is expected not only to approximate the full model but also to construct a classifier with improved generalization performance. The cross validation method can be summarized as follows:

1) The training samples are arbitrarily divided into \( k \) subsets.
2) The parameters of the classifier are trained using the samples in the \( k - 1 \) subsets, and the left subset is tested. Since there are \( k \) possibilities for a way to leave out the subset for evaluation, all \( k \) subsets are tested.
3) The generalization performance is estimated as the average of the performance for the \( k \) subsets.

Cross-validation thus makes good use of the available training sample, and it is especially useful when the amount of available data is not sufficient.

However, we cannot run the cross-validation properly in the IVM procedure because IVM selects the \( S \) that is a subset of the entire training data \( D \). Therefore, we regard kernel feature \( k \) as the input vector instead of the original input vector \( x \), and run cross-validation on kernel feature space \( H_k \). In this paper, we call this cross-validation on kernel feature space (CVKFS) to distinguish it from cross-validation on original feature space. Let the regresser matrix \( K_1 \) (size \( n \times |S| \)) be input, we define the \( k \)-fold CVKFS error as follows,

\[ CV_{KFS} = \frac{1}{n} \sum_{i=1}^{k} ||y_i - p_i||^2 \]

Here \( K_{1(-i)} \) is a matrix obtained by leaving out \( i \)th subset from input \( K_1 \), and \( a_i \) is estimated by minimizing regularized NLL according to Eq. 9.

### IV. Selector using Particle Swarm Optimization

#### A. Particle Swarm Optimization

Particle Swarm Optimization (PSO) is a method for optimizing continuous nonlinear functions proposed by Kennedy and Eberhart [4] in 1995. The algorithm was inspired by social behavior of flocks of birds when they are searching for food. A population, also called a swarm, of potential solutions, denoted as particles, flies in the search space exploring for better regions. As in a flock of birds, where the leader exchanges information with the rest of the other birds, in PSO, the particle having the best solution exchanges information with the rest of the particles. Additionally, each particle can search new regions and get better solutions. In PSO, each particle has a current position, denoted \( p_i^t = (p_{i1}, p_{i2}, ..., p_{id}) \), a velocity (rate of position change), denoted \( v_i^t = (v_{i1}, v_{i2}, ..., v_{id}) \), where \( d \) is the dimension of problem, and a personal best solution denoted \( p_{pi} \). This last one is the best personal position for particle \( i \) and contains the best previous \( p_i \) value of particle \( i \). The best \( p_p \) in the group is denoted by \( p_g \) and its information is shared with the rest of the particles.

Let step of iteration be \( t \), number of particle be \( n_p \), then the positions and velocities of each particle are updated as

\[ p_i(t + 1) = p_i(t) + v_i(t + 1), \]
\[ v_i(t + 1) = v_i(t) + c_1 R_p(t)[p_{pi}(t) - p_i(t)] + c_2 R_g(t)[p_g(t) - p_i(t)]. \]

Here, \( R_p, R_g \) are \( d \times d \) diagonal matrix such that elements are random values in \([0,1]\), and coefficients \( c_1, c_2 \) are the acceleration coefficients, generally set to be \( c_1 = c_2 = 2 \). This equation means particle \( i \) flies toward the group best solution \( p_g \) and the individual best solution \( p_{pi} \) when the position \( p_i \) is far from \( p_{pi} \) and \( p_g \); otherwise, particle \( i \) almost remains stationary when \( p_i \) is near to \( p_{pi} \) and \( p_g \). The PSO procedure is summarized in Algorithm 2.
Algorithm 2 PSO procedure

Minimize the function \( g(p) \) to find the optimal solution \( p \).

1. Set \( t_{max}, n_p \), and search range of \( v \)
2. Set \( t = 1 \), initialize \( p \) and \( v \)
3. repeat
   4. for \( i = 1 \ldots n_p \) do
   5. Evaluate \( g(p_i) \).
   6. if \( g(p_i) < g(p_{pi}) \) then
   7. \( p_{pi} = p_i(t) \)
   8. end if
   9. end for
10. \( p_t = \text{arg min}\{g(p_{p1}), \ldots , g(p_{p1})\} \)
11. Update \( p, v \) according to Eq. (18),(19)
12. \( t = t + 1 \)
13. until \( t = t_{max} \)

B. Binary Particle Swarm Optimization

The original PSO algorithm can only optimize problems in which the elements of the solution are continuous real values. A modification of the PSO algorithm for solving problems with binary values, such as combinational optimization problems, was developed by Kennedy and Eberhart [3] who also proposed the original PSO. The modified algorithm is called binary particle swarm optimization (BPSO). One interesting feature of BPSO is that it uses the same velocity as PSO, which means half of elements \( p \) are selected for as many as half of all candidates. The work of Zhu and Hastie [2] mention that the number of import points \( n_p \) is not in proportion to the size of training data; therefore, using half of all training samples as initial import points is not a better way consistently; because the greater the number of import points, the greater the computational costs of the Newton-Raphson method.

Here, let initial \( v \) be defined as follows:

\[

v_{init} = rand[-1,1] + \log\left(\frac{n_{IP_{\text{init}}}}{n - n_{IP_{\text{init}}}}\right), \tag{22}
\]

where \( n_{IP_{\text{init}}} \) is arbitrary initial number of import points, the histogram of \( s(v_j) - \gamma \) with the velocities in Eq. 22, shown in Fig. 3; in this case, \( n_{IP_{\text{init}}} = 100 \). See Fig. 3; the amount of number of samples which \( s(v_j) > \gamma \) are about 100; this means about 100 samples are imported into subset \( S \).

Furthermore, we add a term for regularizing the number of import points as follows:

\[

R = \alpha n_{IP} \tag{23}
\]

where \( \alpha \) is a prechosen parameter controlling balance between CVKFS and the number of import points. This term looks like the regularized term in AIC [12]. If we let \( \alpha \) be a modest small number, this helps to select models with less import vectors when the CVKFS value is almost the same, on the other hand, if we let be a somewhat large number, the number of import points are reduced with slight weakening.
of the generalization performance of the classifiers. We define new loss function as,

\[ Q = CV_{KFS} + \alpha n_{IP}, \]

That algorithm is summarized as Algorithm 3.

**Algorithm 3 Selection of Import Vectors via Binary Particle Swarm Optimization and Cross-Validation**

1: Search optimal \( \lambda \) and \( \sigma^2 \) using PSO by minimizing CV error of KLR
2: Set \( t = 1, t_{max}, n_p, n_{IPInit} \) and initial range of \( p, v \) as Eq. (20) and (22)
3: initialize \( v \) and \( p \)
4: repeat
5: \( \text{for } i = 1, ..., n_p \text{ do} \)
6: \( S_i = \emptyset \)
7: \( \text{for } j = 1, ..., n \text{ do} \)
8: \( \text{if } p_{ij} = 1 \text{ then} \)
9: \( S_i = S_i \cup \{x_j\} \) (import \( x_j \) to \( S_i \))
10: \( \text{end if} \)
11: \( \text{end for} \)
12: Evaluate \( Q(S_i) = CV_{KFS}(S_i) + \alpha n_{IP_i} \)
13: \( \text{if } Q(S_i) < Q(S_{pi}) \text{ then} \)
14: \( p_{pi} = p_i(t), S_{pi} = S_i(t) \)
15: \( \text{end if} \)
16: \( \text{end for} \)
17: \( \{S_g, p_g\} = \text{arg min} \{Q(S_1), ..., Q(S_{n_{IP}})\} \)
18: update \( p, v \) according to Eq. (19) and (20)
19: \( t = t + 1 \)
20: \( \text{until } Q_g \text{ has converged or } t = t_{max} \)
21: minimize \( H(S_g) \) to find optimal \( \alpha \)

Regarding the stopping rule in line 20, we finish searching the combinations of import points when \( p_g \) has not been updated in \( \Delta t \) times (e.g., \( \Delta t = 30 \)).

**V. EXPERIMENTAL RESULTS**

In this section, we compare the performance of the basic IVM, the IVM with CVKFS, and the IVM with CVKFS and BPSO on 4 benchmark datasets: Ionosphere, Wisconsin Breast-cancer (shown Breast-c. at the following) [13], Heart, and German [14] (TABLE I is a summary of these datasets). We estimate the optimal regularization parameters \( \lambda \) and kernel parameters \( \sigma^2 \) in IVMs by minimizing the cross-validation error rate of the full-KLR model, and search parameters efficiently with PSO [15]. Let parameter \( n_p = 20 \), \( t_{max} = 200 \), \( n_{IPInit} = 5 \), and \( \alpha = 1.0e^{-4} \) in BPSO and the fold of CVKFS be 5 (these parameters are determined through experiments). We repeat this 3 times from parameter search to evaluation.

The results are in TABLE II, TABLE III and TABLE IV. SVM in the tables is RBF kernel SVM, we used the LIBSVM [16] package, IVM is the original IVM in the work of Zhu and Hastie, IVMCV is the IVM with the CVKFS criteria and selecting import points by greedy forward strategy, and IVMCVBPSO is the IVM with the CVKFS criteria that selects import points using BPSO. IVM, IVMCV, and IVMCVBPSO share the parameters selected by minimizing the KLR cross-validation miss-classification rate, and the SVM parameter is determined by minimizing the SVM cross-validation miss-classification rate, respectively. The number outside each bracket is the average of evaluations, and the number inside each bracket is the standard error.

TABLE II provides a comparison of the miss-classification rate for the test data. The IVMCVBPSO model outperformed the other models on 2 of 4 data sets (Heart and German) and demonstrated similar performance with that of IVMCV on 1 dataset (Ionosphere). This denotes CVKFS criteria and BPSO selector behave well for generalization.

TABLE III shows a comparison of the number of import points. The IVMs use a much smaller subset \( S \) than the SVM as Zhu and Hastie mentioned [2]. Furthermore, IVMCV and IVMCVBPSO constructed more sparse models (this means less import points) than basic IVM on 2 of 4 data sets (Ionosphere and Heart), and a similar one on 1 of 4 data sets (Breast-c.).

TABLE IV shows a comparison of the values of CVKFS on IVMCV and IVMCVBPSO. Obviously IVMCVBPSO outperforms on all data sets. This means that the BPSO can find better solutions to minimize CVKFS than the greedy forward strategy.

Fig. 4 (a) shows values of regularized NLL and missclassification rate for test data on IVM, and Fig. 4 (b) shows values of CVKFS and missclassification rate for test data on IVMCV at each iteration step in dataset Heart. The criterion CVKFS estimates generalization error better than regularized NLL.

**VI. CONCLUSIONS**

In this paper, we discussed a scheme that improves the generalization and computational cost of KLR in actual pattern classification. The proposed algorithm is based on the IVM, the cross-validational metric on kernel feature space, and the BPSO. We could select a small subset of import vectors and improve generalization performance by using the proposed algorithm.

Regarding elements of methods we used, such as Newton-Raphson method and \( k \)-fold cross-validation, they are simple and basic algorithms. LR and KLR are well studied algorithms, and IVMCV and IVMCVBPSO are a lot of efficient methods have been proposed. For example, Cawley and Talbot presented approximate leave-one-out cross-validation for KLR [17], [18]. We can

**TABLE I**

**SUMMARY OF THE BENCHMARK DATASETS**

<table>
<thead>
<tr>
<th>Dataset</th>
<th># of training</th>
<th># of test</th>
<th># of feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ionosphere</td>
<td>200</td>
<td>151</td>
<td>33</td>
</tr>
<tr>
<td>Heart</td>
<td>170</td>
<td>100</td>
<td>13</td>
</tr>
<tr>
<td>German</td>
<td>700</td>
<td>300</td>
<td>24</td>
</tr>
<tr>
<td>Breast-c.</td>
<td>200</td>
<td>483</td>
<td>10</td>
</tr>
</tbody>
</table>
Here, $\Delta k = 1, \epsilon = 0.001$ in IVM. IVM satisfied the stopping rule when 31 import points were selected. The CVKFS criterion was minimized when 15 import points were selected.

![Image](image1)

**Fig. 4.** Comparison of Criterion (Heart)

Here, $\Delta k = 1, \epsilon = 0.001$ in IVM. IVM satisfied the stopping rule when 31 import points were selected. The CVKFS criterion was minimized when 15 import points were selected.

<table>
<thead>
<tr>
<th>Table II: Comparison of Test error</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM</td>
</tr>
<tr>
<td>---</td>
</tr>
<tr>
<td>Ionosphere</td>
</tr>
<tr>
<td>Heart</td>
</tr>
<tr>
<td>German</td>
</tr>
<tr>
<td>Breast-c.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table III: Comparison of number of import points</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM</td>
</tr>
<tr>
<td>---</td>
</tr>
<tr>
<td>Ionosphere</td>
</tr>
<tr>
<td>Heart</td>
</tr>
<tr>
<td>German</td>
</tr>
<tr>
<td>Breast-c.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table IV: Comparison of $CV_{KFS}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>IVMCV</td>
</tr>
<tr>
<td>---</td>
</tr>
<tr>
<td>Ionosphere</td>
</tr>
<tr>
<td>Heart</td>
</tr>
<tr>
<td>German</td>
</tr>
<tr>
<td>Breast-c.</td>
</tr>
</tbody>
</table>

References