Proceedings of the 2007 IEEE Symposium on
Computational Intelligence and Data Mining (CIDM 2007)

Query-sensitive Feature Selection for Lazy Learners

Xin Tong and Mingyang Gu
Department of Computer and Information Science,
Norwegian University of Science and Technology,
7030, Trondheim, Norway
{tongxin, mingyang}@idi.ntnu.no

Abstract

Feature selection contributes to increasing many learners' accuracy by identifying and removing irrelevant features in multidimensional datasets. Conventional feature selection methods determine the optimal feature subset independently from and prior to the introduction of a new query. In general, some features will be relevant only in certain tasks. We argue that a query, as an indicator of the attention focus and current task, is a major part of the context and should be involved in the determination of the final feature subset. In this paper we attempt to propose a query-sensitive feature selection model, present two algorithms for applying such a feature selection method, and test their effectiveness by comparing their performances to those of the conventional selection algorithms. Our experiments are executed under a nearest neighbor classification environment and the results show a consistent improvement in the classification performance when a query-sensitive feature subset is selected and used for measuring similarities between the query and other instances. The results suggest that the performance of a lazy learner has the potential to increase through query-sensitive feature selection.

1 Introduction

The presence of irrelevant features confuses many data mining methods and may result in severe degradation of accuracy. This problem can be solved by identifying the irrelevant features and removing them from future processing. Feature selection is to select a feature subset from the full set of features based on some optimization criterion, through which the irrelevant, redundant, or least useful features are discarded. With the primary objective to reduce the number of features used to characterize a dataset so as to improve a system's performance on a given task[1], most feature selection methods would take a tradeoff between a high system performance and a low cardinality of the finally chosen feature subset. That is, they attempt to draw a more compact feature subset with as little performance degradation as possible.

Different feature selection methods have been proposed since 70's and surveyed in [6] and [10]. Feature selection algorithms can be grouped into 'Filter' or 'Wrapper' approaches. A 'Wrapper' approach [9], such as FSS [4], is to evaluate and select the feature subset by using the actual accuracy measure of the corresponding learning machine. A 'Filter' approach, such as FOCUS [2], Relief [7], attempts to select the feature subset independent of the concrete learner. The examples of feature evaluating measures are distance measures, dependence measures, distance measures, information theoretic measures.

One important property of the forementioned feature selection methods is that they are static because the features have been selected before a query is presented and the chosen feature subset would not change any more across different queries. In other words, features would be chosen only based on known dataset. However, some features will be relevant only in certain tasks. That is, the relevance of features may change given the different queries which indicate different problems. Studies in psychology, cognitive science and computer science [15] [11] [13] [14] share the similar views suggesting that feature relevance is a highly dynamic concept that is highly influenced by purpose. From this view, a query, as an indicator of the attention focus and current task, is a major part of the context and should be involved in the determination of final feature subset. The static feature selection methods cannot catch such potentially important information.

In this paper we suggest that query information should play a positive role in adjusting selected feature subset by favoring those dimensions that are more significant for accomplishing the particular task. Query-sensitivity means that determination of the relevance of features is not done in isolation from the new query in advance, but partially decided by the query itself.

In this paper we attempt to introduce the notion of query-

1-4244-0705-2/07/$20.00 ©2007 IEEE

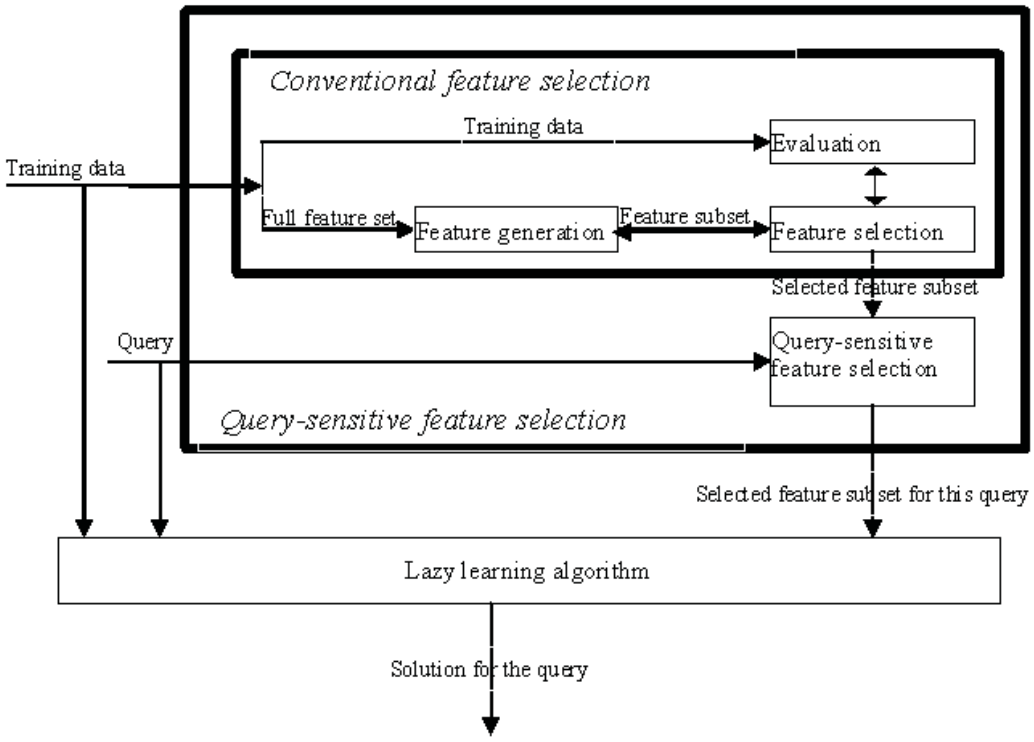

Figure 1. Query-sensitive feature selection model

sensitive feature selection, present two algorithms for applying such a selection, and test the effectiveness of these algorithms by comparing their performances to those of the conventional selection methods. Our experiments are under a nearest neighbor classification environment and our results show a consistent improvement in the classification performance when query-sensitive feature subsets are used for similarity measurement. These results suggest that the effectiveness of a lazy learner has the potential to improve through the use of query-sensitive feature selection.

The paper is structured as follows: in Section 2, we propose the model of query-sensitive feature selection and introduce the selection process. In Section 3, we propose two query-sensitive feature selection algorithms. Section 4 introduces a series of experiments testing the effectiveness of query-sensitive feature selection. The results are analyzed and discussed in detail. At the end we draw a conclusion of our work.

2 A Query-sensitive Feature Selection Model

In this section, we introduce a query-sensitive feature selection model. As showed in Figure 1, this model adds an extra query-sensitive feature selection module into the traditional feature selection process. In this model, a final query-sensitive feature subset is selected through two main steps: step 1 - given the training data, a feature subset is selected through a conventional feature selection process; step 2 - staring from the feature subset from step 1 and the features appearing in a concrete query, a query-sensitive feature selection process is invoked to achieve the final query-sensitive feature subset which is used to solve this query.

The modules in Figure 1 are explained in the following. The feature generation module would generate candidate feature subsets and the feature selection module will determine which subset will be selected based on the returned values of the evaluation module on the candidate subsets. Sometimes, feature subset generation depends on the partial results of feature selection module, so there may be interaction between these two modules. A query-sensitive feature selection module is to decide the final selected feature subset based on selected subset by a traditional method and query information when a new query comes. This module would examine the features specified in the query but missing in the traditional selected feature subset and decide whether they should be selected or not according to certain criterion. Different query-sensitive feature selection algorithms are designed based on different criteria.

The execution process of a lazy learner with a query sensitive feature selection algorithm is as follows. First a subset of features are selected using conventional methods based on the information of training data. Given a new query, the

system invokes the query-sensitive feature selection to decide the final feature subset for this query, calculates the similarity between the query and instances on the query-sensitive subset of features and then finds the most matched instance to solve this query.

The motivation of this two-step process instead of 'only' doing query-time feature selection is to reduce the on-line computation complexity. A query is given during on-line process. Considering both the query and instances information to effectively select the features during the query-time can be very time-consuming. Therefore, we try to improve the effectiveness and efficiency of the total feature selection through using the result of conventional feature selection processes that are mainly completed offline. And then the query-sensitive feature selection process does some adjustment to the output of the conventional feature selection process.

3 Query-sensitive Feature Selection Algorithms

Treating the query as a factor in determining the final feature subset raises a set of new issues , for example, what information within the query should be taken into consideration, and how to quantify such information. In this section we propose two concrete algorithms referred as QSFS1 (Query Sensitive Feature Selection 1), QSFS2 (Query Sensitive Feature Selection 2) to draw the query-sensitive feature subsets.

1. QSFS1

QSFS1 is proposed by taking the view that if the query's value on a feature is singular from the normal value of this feature, the feature may describe a significant character of the current problem situation, so this feature should be added into the final query-sensitive feature subset.

The process defined in QSFS1 is that among the numeric features specified in the query but missing in the traditionally selected feature subset, the features on which the query's values depart more than one standard deviation from the means of the corresponding features would be added to final selected feature subset. QSFS1's pseudo code is showed in Table 1.

2. QSFS2

QSFS2 is similar to QSFS1 except that instead of using the fixed standard deviation on each feature as the threshold to select this feature it defines a threshold learning process. The thresholds take features' standard deviations as their initial values and are updated according to a learning algorithm defined in QSFS2. The pseudo code of QSFS2 is described in Table 2.

The learning process intends to increase the value of threshold for the possibly irrelevant features and decrease the thresholds for the relevant ones. After a query is classified correctly, the feature on which there is a large difference between the values of the query and nearest instance is probably not relevant very much, so its threshold should be increased, while the thresholds of other features will be decreased. And if a query is classified incorrectly, the features on which there are very small differences between the values of the query and nearest instance are probably not relevant very much, so their thresholds should be increased, while the thresholds of other features will be decreased. The thresholds can be learned from the training data. Later in the experiments described in the next section we use the leave-one-out method to control the learning process. The η here is a positive number determining how much to modify the thresholds at a time.

The idea behind this updating process is to measure the relevance of features in the neighborhoods around the target instances. The learning method itself is not query-sensitive. The similar idea and updating process has been adopted in other researches, such as Each [12], Relief [8].

Table 1. The pseudo-code of QSFS1.

Input: Q is a query
FS is the traditionally selected feature subset
M is the vector of the means of all numeric features
SD is the vector of the standard deviations of all numeric features
Note: QFS is query-sensitive selected feature subset

Procedure $QSFS1(Q, FS, M, SD)$

$QFS = FS$
for each numeric feature f in Q and not in FS
if $|Q(f) - M(f)| \geq SD(f)$
then
$QFS = QFS \cup f$
Return QFS

4 Experiment

The experiment is designed with the objective to test the effectiveness of query-sensitive feature selection algorithms reported in this paper. As we discussed in Section 1, conventional feature selection methods can be divided into two categories, 'Wrapper' and 'Filter'. Here, we select the FSS (Forward Sequential Selection) method as the representative for the 'Wrapper' methods, and Relief method for the 'Filter' methods. The lazy learner in the model is defined in Subsection 4.1. We use Weka system [16] as the test-bed in

Table 2. The pseudo-code of QSFS2.

Input: Q is a query
FS is the traditionally selected feature subset
M is the vector of the means of all numeric features
SD is the vector of the standard deviations of all numeric features
TD is the training database
Note: QFS is query-sensitive selected feature subset
T is the vector of the learned thresholds of all numeric features

Procedure $QSFS2(Q, FS, M, SD, TD)$
$T = LearnT(TD, SD)$
$QFS = FS$
for each numeric feature f in q and not in FS
if $|Q(f) - M(f)| \geq T(f)$
then
$QFS = QFS \cup f$
Return QFS

Procedure $LearnT(TD, SD)$
for each $q \in TD$,do
for each $c_i \in TD - q$
$Dis[c_i] \leftarrow distance(q, c_i)$
select c_k, where $Dis[c_k]$ is minimal of all $Dis[c_i]$
for each numeric feature f in q or in c_k
if $class(q) = class(c_k)$
then
$T(f) = T(f)(1 + \eta(|q(f) - c_k(f)| - SD(f)))$
else
$T(f) = T(f)(1 - \eta(|q(f) - c_k(f)| - SD(f)))$
Return T

the experiment and the ten-fold cross validation strategy is adopted to calculate the classification accuracy.

4.1 Experiment environment

We test the effectiveness of query-sensitive feature selection in the context of the instance-based classification task (particularly the nearest neighbor algorithm) where instances are represented as flat attribute-value vectors. Let's assume that there are n instances in a dataset, and each instance has m features. Assume also an additional feature called 'class' denoting instances' category. We can then represent the system as a n$*$(m+1) matrix.

$$\begin{array}{c} \\ instance_1 \\ instance_2 \\ instance_3 \\ \vdots \\ instance_n \end{array} \begin{array}{c} \begin{array}{cccc} f_1 & \ldots & f_m & class \end{array} \\ \begin{pmatrix} x_{11} & \ldots & x_{1m} & s_1 \\ x_{21} & \ldots & x_{2m} & s_2 \\ x_{31} & \ldots & x_{3m} & s_3 \\ \vdots & \vdots & \vdots & \vdots \\ x_{n1} & \ldots & x_{nm} & s_n \end{pmatrix} \end{array}$$

where x_{ij} is the value of the i_{th} instance on the j_{th} feature and s_i is the value of the i_{th} instance's class.

All values of each feature would be normalized before computing distances in Equation 1.

A new problem to be solved is called a query here, represented as $(q_1, q_2, \ldots, q_m)$.

Similarity is measured through the notion of distance. That is, the most similar instance is the one which is of least distance from the query. The distance function between a query and an instance is defined using Equation 1.

$$distance(q, x_i) = \sqrt{\sum_{j \in J} \delta^2(q_j, x_{ij})} \tag{1}$$

where J is the set of selected features and $\delta(q_j, x_{ij})$ is the difference between the query and i_{th} instance on the j_{th} feature and has the value:

$$\delta(q_j, x_{ij}) = \begin{cases} q_j - x_{ij} & f_j \text{ is numerical} \\ 0 & f_j \text{ is nominal and } q_j = x_{ij} \\ 1 & f_j \text{ is nominal and } q_j \neq x_{ij} \end{cases} \tag{2}$$

The classification process in the system goes like this: given a query, find the instance with the least distance from the query and then take the category of this instance as this query's category.

4.2 Selected datasets

There are 18 datasets obtained to execute the experiment. All of them come from the UCI repository [3]. All numeric features in these datasets are normalized using corresponding 'Filter' provided in Weka3.4.3 according to the requirements of the lazy learner. The detailed information about the selected datasets is illustrated in Table 3. The columns denote respectively: the name of the dataset (Dataset), the number of the instances in each dataset (Instances), the number of the features excluding the feature 'class' (Total Features), the number of the numeric features (Numeric Features), the number of the nominal features (Nominal Features), the percentage of the missing data (Missing Data) calculated by 'the number of the missing items/(Instances$*$ Total Features)', and the number of categories (Classes).

Table 3. Datasets description

Dataset	Instances	Total Features	Numeric Features	Nominal Features	Missing Data	Classes
Anneal	898	38	6	32	64.98%	5
Anneal Original	898	38	9	29	73.13%	6
Autos	205	25	15	10	1.15%	7
Balance scale	625	4	4	0	0%	3
Breast Cancer	286	9	0	9	0.35%	2
Breast-W	699	9	9	0	0.25%	2
Credit Approval	690	15	6	9	0.65%	2
Diabetes	768	8	8	0	0%	2
Glass	214	9	9	0	0%	7
Heart-C	303	13	6	7	0.18%	5
Hepatitis	155	19	6	13	2.43%	2
Horse colic	368	22	7	15	23.80%	2
Ionosphere	351	34	34	0	0%	2
Iris	150	4	4	0	0%	3
Labor	57	16	8	8	35.75%	2
Segment	2310	19	19	0	0%	7
Sonar	208	60	60	0	0%	2
Vowel	990	13	10	3	0%	11

4.3 Testing the query-sensitive feature selection with FSS

FSS (Forward Sequential Selection) [1][5] is one of the most common algorithms used in 'Wrapper' feature selection methods. FSS starts with an empty feature subset and evaluates the performance of the lazy learner selecting a different feature each time. It then adds to this subset the feature that yields the best performance and takes the new subset as the starting subset for the next iteration. This cycle repeats until no improvement is obtained by extending the current subset.

In our experiment, FSS is selected as the benchmark in the 'Wrapper' feature selection category, and based on it, two other query-sensitive feature selection methods, FSS+QSFS1, and FSS+QSFS2 are designed:

- *FSS+QSFS1* In addition to FSS, the feature selection algorithm QSFS1 is used to determinate the final query-sensitive feature subset by using the fixed threshold (the standard deviation) on each feature in the query to decide whether the feature should be selected or not.

- *FSS+QSFS2* In addition to FSS, the feature selection algorithm QSFS2 is used to determinate the final query-sensitive feature subset by using a learned threshold on each feature in the query to decide whether the feature should be selected or not.

The experiment results for the FSS related feature selection methods are illustrated in Table 4. The columns FSS, FSS+QSFS1, and FSS+QSFS2, represent the ten-fold cross validation classification accuracy of the lazy learner with the corresponding feature selection algorithm FSS, FSS+QSFS1, and FSS+QSFS2. To show the comparisons more clearly, we add a particular row 'Average' in the bottom of the table to show the average value for all the numbers appearing in the corresponding columns. From the values in the 'Average' row, we can see that the classification accuracy of the learner based on FSS+QSFS1 algorithm is higher than that of the learner based on FSS algorithm ($83.41 > 81.81$), and further, the classification accuracy of the learner based on FSS+QSFS2 algorithm is higher than that of the learner based on FSS+QSFS1 algorithm ($84.46 > 83.41$).

To show how significant the evaluation results support the above findings, we further carry out the formal hypothesis tests. We identify the following two hypotheses based on the above findings:

H1: the lazy learner based on FSS+QSFS1 algorithm is more effective than that based on FSS algorithm, that is, the classification accuracy of FSS+QSFS1 based learner is higher than that of FSS based learner (the results of subtracting the values in column 'FSS' from those in column 'FSS+QSFS1' for each dataset are selected as the significance test parameter).

H2: the lazy learner based on FSS+QSFS2 algorithm

Table 4. Experiment results with FSS algorithm

Dataset	FSS	FSS+QSFS1	FSS+QSFS2
Anneal	88.98	93.76	97.10
Anneal Original	78.29	84.86	92.43
Autos	77.56	78.05	77.56
Balance Scale	79.04	79.04	78.56
Breast Cancer	66.43	59.44	63.64
Breast-W	94.71	95.85	95.28
Credit Approval	78.41	81.01	82.90
Diabetes	66.67	65.23	69.66
Glass	74.30	75.70	74.30
Heart-C	69.97	73.60	71.95
Hepatitis	80.65	80.00	81.29
Horse Colic	83.15	83.70	84.51
Ionosphere	88.60	88.89	88.89
Iris	94.67	96.00	96.00
Labor	78.95	82.46	80.70
Segment	96.88	95.93	97.14
Sonar	76.92	88.94	88.94
Vowel	98.38	98.99	99.49
Average	81.81	83.41	84.46

is more effective than that based on FSS+QSFS1 algorithm (the results of subtracting the values in column 'FSS+QSFS1' from those in column 'FSS+QSFS2' for each dataset are selected as the significance test parameter).

In our research, we select the one-tailed t test to complete the significance evaluation. With the degree of freedom of 17 and the significance level of 0.05, we find out the critical value as 1.740. For all these two hypotheses listed above, we get the calculated t values as 1.760 and 1.818 respectively. Since all the calculated t values are larger than the critical value, we refuse all the non-hypotheses and accept the two original hypotheses.

4.4 Testing the query-sensitive feature selection with Relief

Relief is one type of the 'Filter' feature selection method. Relief is a feature weight based algorithm. Given a training dataset D, Relief starts with selecting m samples from it. For each sample, a closest instance with the same class value (Near-hit) and a closest instance with different class value (Near-miss) are identified, and the weights for each features will increase by a value correlated with the deviation of the value of this feature in the sample instance from that in the Near-hit instance and decrease by a value correlated with that of the sample value on one feature from that in the Near-miss instance. The final selected feature set is decided by whether or not the weights of each feature (normalized by m) are larger that a specified threshold [7].

In this experiment, Relief is selected as the benchmark in the 'Filter' feature selection category, and based on it, two other query-sensitive feature selection methods, Relief+QSFS1, and Relief+QSFS2 are designed:

- *Relief+QSFS1* Based on the feature set selected by Relief algorithm, QSFS1 algorithm is further used to determinate whether the rest features in the query should be added into the selected feature set through comparing the deviation of each feature from its average value and a fixed threshold (the standard deviation).

- *Relief+QSFS2* Based on the feature set selected by Relief algorithm, QSFS2 decides whether the rest features in the query should be added into the selected feature set through comparing the deviation of each feature from its average value and a learned threshold.

Table 5 has the similar structure as Table 4, but its content illustrates the evaluation results concerning the lazy learners related to Relief algorithms. Concretely, the columns, Relief, Relief+QSFS1, and Relief+QSFS2 present the ten-fold cross validation classification accuracy of the lazy learner based on the Relief algorithm, Relief+QSFS1 and Relief+QSFS2 respectively. From the values in the 'Average' row, we can see that the classification accuracy of the learner based on Relief+QSFS1 algorithm is higher than that of the learner based on Relief algorithm ($80.73 > 76.19$), and further, the classification accuracy of the learner based on Relief+QSFS2 algorithm is higher than that of the learner based on Relief+QSFS1 algorithm ($84.28 > 80.73$).

To show how significant the evaluation results support the above findings, we further carry out the formal hypothesis tests. We further identify the following two hypotheses based on the above findings:

H3: the lazy learner based on Relief+QSFS1 algorithm is more effective than that based on Relief algorithm (the results of subtracting the values in column 'Relief' from those in column 'Relief+QSFS1' for each dataset are selected as the significance test parameter).

H4: the lazy learner based on Relief+QSFS2 algorithm is more effective than that based on Relief+QSFS1 algorithm (the results of subtracting the values in column 'Relief+QSFS1' from those in column 'Relief+QSFS2' for each dataset are selected as the significance test parameter).

The same hypothesis test process as described in above subsection is further used to test the significances of hypothesis H3 and H4. With significant level of 0.05, the calculated t values for these two hypotheses are 1.872 and 1.961 respectively. Since all these two t values are larger than the critical value (1.740), we refuse the non-hypotheses and accept the two original hypotheses.

Table 5. Experiment results with Relief algorithm

Dataset	Relief	Relief+QSFS1	Relief+QSFS2
Anneal	98.44	98.22	95.77
Anneal Original	95.99	96.10	92.43
Autos	75.61	76.59	75.61
Balance Scale	61.12	74.24	79.04
Breast Cancer	56.99	56.99	59.44
Breast-W	95.28	95.28	95.57
Credit Approval	80.72	80.72	82.46
Diabetes	61.98	66.28	68.36
Glass	20.56	45.33	72.90
Heart-C	80.53	78.22	78.88
Hepatitis	78.71	80.65	81.29
Horse Colic	77.99	77.99	82.34
Ionosphere	89.46	86.89	87.46
Iris	95.33	96.00	96.00
Labor	80.70	80.70	84.21
Segment	97.14	96.49	97.10
Sonar	81.25	86.54	88.94
Vowel	43.54	80.00	99.29
Average	76.19	80.73	84.28

5 Conclusion

Noticing the relevance of feature is sensitive to different tasks, our research describes an attempt to devise means by which this kind of change of feature relevance can be detected. In this paper, we introduce the notion of query-sensitive feature selection which takes query information into account during the feature selection. A general query-sensitive feature selection model is proposed, providing a framework under which different query-sensitive selection algorithms could be developed. Furthermore, we propose two query-sensitive feature selection algorithms QSFS1 and QSFS2. Based on the feature set selected by the conventional feature selection algorithms, QSFS1 extends the selected feature set by adding the numeric features on which the query's value is more than one standard deviation from the average feature value. QSFS2 adds to the selected featrue set with the numeric features on which the query's value is more than a learned threshold of this feature from the average feature value. Experimental results has shown the effectiveness of query-sensitive feature selection.

References

[1] D. W. Aha and R. Bankert. Feature selection for case-based classification of cloud types: An empirical comparison. *In D. W. Aha (Ed.) Case-Based Reasoning: Papers from the 1994 Workshop (Technical Report WS-94-01). Menlo Park, CA: AAAI Press*, 1994.

[2] H. Almauallium and T. Dietterich. Learning with many irrelevant features. In *Ninth National Conference on Artificial Intelligence*, volume 2, pages 547–552, 1991.

[3] C. Blake and C. Merz. Uci repository of machine learning databases [http://www.ics.uci.edu/ mlearn/mlrepository.html], 1998.

[4] P. A. Devijver and J. Kittler. Pattern recognition: A statistical approach. *Egnlewood Cliffs*, 1982.

[5] P. Domingos. Context-sensitive feature selection for lazy learners. *Artificial Intelligence Review*, 11(1-5):227–253, 1997.

[6] K. Fukunaga. *Introduction to Statistical Pattern Recognition*. CA: Academic Press, San Diego, 1990.

[7] K. Kira and L. Rendell. The feature selection problem traditional methods and a new algorithm. In *Tenth National Conference on Artificial Intelligence*, 1992.

[8] K. Kira and L. Rendell. A practical approach to feature selection. In D. Sleeman and P. Edwards, editors, *the Ninth International Conference on Machine Learning*, pages 249 – 256, San Mateo, California, 1992.

[9] R. Kohavi and G. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2):273–324, 1997.

[10] H. Liu and H. Motoda. *Feature Selection for Knowledge Discovery and Data Mining*. Kluwer Academic Publishers, MA, 1998.

[11] R. M. Nosofsky. Attention, similarity, and the identification-categorization relationship. *Journal of Experimental Psychology: General*, 115:39–57, 1986.

[12] S. Salzberg. A nearest hyper-rectangle learning method. *Machine Learning*, 6:251–276, 1991.

[13] L. Schamber, M. B. Eisenberg, and M. S. Nilan. A re-examination of relevance: toward a dynamic, situational definition. *Information Processing and Management*, 26(6):755, 1990.

[14] A. Tombros and C. v. Rijsbergen. Query-sensitive similarity measures for information retrieval (invited paper). *Knowledge and Information Systems*, 6(5):617–642, 2004.

[15] A. Tversky. Features of similarity. *Psychological Review*, 84(4):327–352, 1977.

[16] I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools with Java implementations*. Morgan Kaufmann, San Francisco, 2000.