Demand-Driven Associative Classification

Adriano Veloso , Wagner Meira Jr.[1]

[1]Departamento de Ciência da Computação – Universidade Federal de Minas Gerais
{adrianov, meira}@dcc.ufmg.br

Abstract. *The ultimate goal of machines is to help humans to solve problems. Such problems range between two extremes: structured problems for which the solution is totally defined (and thus are easily programmed by humans), and random problems for which the solution is completely undefined (and thus cannot be programmed). Problems in the vast middle ground have solutions that cannot be well defined and are, thus, inherently hard to program. Machine Learning is the way to handle this vast middle ground, so that many tedious and difficult hand-coding tasks would be replaced by automatic learning methods. There are several machine learning tasks, and this work is focused on a major one, which is known as classification. Some classification problems are hard to solve, but we show that they can be decomposed into much simpler sub-problems. We also show that independently solving these sub-problems by taking into account their particular demands, often leads to improved classification performance. This is shown empirically, by solving real-world problems using the computationally efficient algorithms that we present in this work. Significant improvements in classification performance are reported for all these problems, under a comparative study involving a broad repertoire of representative algorithms. Further, theoretical evidence supporting our algorithms is also provided.*

1. Introduction

Learning is a fundamental ability of many living organisms. It leads to the development of new skills, values, and preferences. Improved learning capabilities catalyze the evolution, and may distinguish entire species with respect to the activities they are able to perform. The importance of learning is, thus, beyond question. Learning covers a broad range of tasks. Some tasks are particularly interesting because they can be mathematically modeled. This makes natural to wonder whether machines might be programmed to learn, leveraging one of the fastest growing research areas today: machine learning.

A prominent approach to machine learning is to repeatedly demonstrate how the problem is solved, and let the machine learn by example, so that it generalizes some rules about the solution and turn these into a program. This process is known as supervised learning. Specifically, the machine takes matched values of inputs (instantiations of the problem to be solved) and outputs (the solution), and absorb whatever information their relation contains in order to emulate the true mapping of inputs to outputs. When outputs are drawn from a fixed and finite set of possibilities, the process is known as classification.

The relationship between inputs and outputs may be expressed as a mapping function, which takes an input and provides the corresponding output. Since this function is unknown, the classification problem can be essentially stated as a function approximation problem: given as examples some inputs for which the outputs (i.e., the classes) are known, the goal is to extrapolate the outputs associated with other inputs as accurately as possible. Several classification algorithms follow this function approximation paradigm,

as discussed in [Poggio and Girosi 1998]. The limiting factor of these algorithms is the accuracy of the mapping functions they can provide in a reasonable time. Dramatic gains cannot be achieved through minor algorithmic modifications, but require the introduction of new strategies and approaches. The key approach we exploited in this work is to decompose a hard classification problem into possibly easier sub-problems, where each sub-problem is defined by inputs that are similar somehow. Then, a specific mapping function for each sub-problem is built independently from each other, on a demand-driven basis, according to particularities of each sub-problem. This approach leads to a finer-grained approximation, in which multiple mapping functions are built. Specifically, each function provides an optimized approximation of the target function for a specific input. Although this strategy is very intuitive, some key questions must be answered:

- How sub-problems are defined? Which particularities of a sub-problem can be used to improve function approximation?
- Is there a suitable way to search for mapping functions, such that the space for candidate functions is constrained?
- Does indenpendently solving sub-problems lead to better approximations than directly solving the entire problem?
- What is the computational cost associated with our algorithms? Are there polynomial time, efficient algorithms? Are they more effective than existing algorithms?

This work is mainly devoted to answer these questions. Specific contributions include:

- We introduce an approach that constrains the space for candidate mapping functions to only those functions that are likely to be accurate [Veloso et al. 2006b], leading to algorithms that need few examples to build accurate functions.
- We show that different sub-problems demand different mapping functions, and we propose polynomial-time algorithms for demand-driven classification.
- We present extensions to demand-driven associative classification, and we show that most of our conclusions in the classification scenario also hold for these extensions, such as multi-label classification [Veloso et al. 2007a], multi-metric classification [Veloso et al. 2009a], calibrated classification [Veloso et al. 2008b], self-training [Veloso et al. 2009d], and ranking [Veloso et al. 2008a].
- We demonstrate the effectiveness of the proposed algorithms through an extensive set of experiments. In addition to typical benchmarks, we also investigated various real-world application scenarios, including document categorization [Veloso et al. 2006a], gene functional analysis [Veloso and Meira 2005], Email/Web spam detection [Veloso et al. 2009b], revenue maximization [Veloso et al. 2009c], name disambiguation [Veloso et al. 2009d], and ranking [Veloso et al. 2007b, Veloso and Meira 2007].

2. Related Work

Many prior efforts have been devoted to the development of classification algorithms. Particularly noteworthy algorithms include Perceptrons [Rosenblatt 1958], Nearest Neighbors (kNNs [Cover and Hart 1967]), decision trees (DTs [Breiman 1984]), and Support Vector Machines (SVMs [Boser et al. 1992]). For several years or even decades, many modifications and improvements have been proposed to these algorithms, leading to algorithms such as [Joachims 2006, Quinlan 1986], which are among the baselines we used for comparison against the algorithms to be presented in this work. A much more detailed discussion about related work can be found in [Veloso 2009].

3. The Classification Problem

In this section we present the definitions necessary to understand the algorithms to be introduced in this work. These definitions are presented in more details in [Veloso 2009].

Training Data and Test Set – In a classification problem, there is a set of input-output pairs of the form z_i=(x_i, y_i) (i.e., examples). Each input x_i is a fixed-length record of the form $< a_1, \ldots, a_l >$, where each a_k is an attribute-value. Each output y_i draws its value from a discrete and finite set of possibilities $y = \{c_1, \ldots, c_p\}$, and indicates the class to which z_i belongs. Cases where $y_i =?$ indicate that the correct class of z_i is unknown. There is a fixed but unknown conditional probability distribution P$(y|x)$ (the relationship between inputs and outputs is fixed but unknown). The set of pairs is explicitly divided into two partitions, the training data (denoted as $\mathcal{S}$) and the test set (denoted as $\mathcal{T}$):

$$\mathcal{S} = \{s_1 = (x_1, y_1), \ldots, s_n = (x_n, y_n)\} \qquad \mathcal{T} = \{t_1 = (x_{n+1}, ?), \ldots, t_m = (x_{n+m}, ?)\}$$

Further, it is assumed that pairs in $\mathcal{T}$ are in some sense related to pairs in $\mathcal{S}$, and that $\{s_1, \ldots, s_n\}$ and $\{t_1, \ldots, t_m\}$ are sampled from the same distribution P$(y|x)$.

Classification Algorithm – A classification algorithm takes $\mathcal{S}$ and $\mathcal{T}$ as input, and returns a mapping function $f_{\mathcal{S}} : x \rightarrow y$ that represents the relation between inputs and outputs in $\mathcal{S}$, that is, the mapping function $f_{\mathcal{S}}$ is a discrete approximation of P$(y|x)$. Many possible functions can be derived from $\mathcal{S}$. The hypothesis space $\mathcal{H}$ is the space of functions explored by the algorithm in order to select $f_{\mathcal{S}}$. The selected function $f_{\mathcal{S}}$ is finally used to estimate outputs y given inputs x, for each $x_i \in \mathcal{T}$. The classification task is that of selecting, from all functions in $\mathcal{H}$, those that best approximate P$(y|x)$.

Expected and Empirical Errors – The expected error of a function $f_{\mathcal{S}}$ is defined as:

$$\mathcal{I}_{\mathcal{T}}[f_{\mathcal{S}}] = \int_{t=(x,y)} \ell(f_{\mathcal{S}}, t)\mathrm{dP}(y|x) \text{ , where } \ell(f_{\mathcal{S}}, z_i) = \begin{cases} 0 & \text{if } f_{\mathcal{S}}(x_i) = y_i \\ 1 & \text{otherwise (prediction is wrong)} \end{cases}$$

Ideally, the selected function $f_{\mathcal{S}}$ should provide the lowest expected error $\mathcal{I}_{\mathcal{T}}[f_{\mathcal{S}}]$. However, $\mathcal{I}_{\mathcal{T}}[f_{\mathcal{S}}]$ cannot be computed because P$(y|x)$ is unknown, and thus such an optimal function cannot be directly selected. On the other hand, the empirical error of $f_{\mathcal{S}}$ can be easily computed using $\mathcal{S}$:

$$\mathcal{I}_{\mathcal{S}}[f_{\mathcal{S}}] = \frac{1}{n}\sum_{i=1}^{n} \ell(f_{\mathcal{S}}, s_i)$$

Efficient Algorithms – The empirical error can be considered an approximation of the expected error. It can be shown [Cucker and Smale 2001] that the empirical error converges uniformly to the expected error when $|\mathcal{S}| \rightarrow \infty$. An efficient classification algorithm ensures that this convergence occurs with high rate. Formally, given two constants ϵ and δ (with values between 0 and 0.5), a classification algorithm is efficient if it selects, in polynomial time and with a polynomial number of examples, with probability $(1 - \delta)$, a function $f_{\mathcal{S}} \in \mathcal{H}$ for which $\mathcal{I}_{\mathcal{S}}[f_{\mathcal{S}}] < \epsilon$, and $\mathcal{I}_{\mathcal{S}}[f_{\mathcal{S}}] \approx \mathcal{I}_{\mathcal{T}}[f_{\mathcal{S}}]$. Obviously, the more accuracy (lower ϵ values) and the more certainty (lower δ values) one wants, the more examples (i.e., n) the classification algorithm needs. As shown in [Valiant 1984], the expected error can be bounded by:

$$\epsilon \geq \frac{1}{n}\big(\ln|\mathcal{H}| + \ln(\frac{1}{\delta})\big) \tag{1}$$

4. Demand-Driven Associative Classification

We start this section with a simple, eager algorithm, which produces a single mapping function $f_{\mathcal{S}}$. Then, we present a more sophisticated, demand-driven algorithm, which produces multiple functions. Finally, we employ powerful function approximation techniques in order to boost the effectiveness of the demand-driven algorithms.

4.1. Classification using Association Rules

The hypothesis space, $\mathcal{H}$, may contain a huge number of functions. Randomly producing functions, in the hope of finding one that approximates well the target function $\mathrm{P}(y|x)$, is unlikely to be an efficient strategy. A better alternative is to directly exploit associations between inputs and outputs. Such associations are usually hidden in the examples, and, when uncovered, they may reveal important aspects concerning the underlying phenomenon that generated these examples. These aspects can be exploited for the sake of producing only functions that provide potentially good approximations of $\mathrm{P}(y|x)$. This strategy has led to algorithms which are referred to as associative algorithms, since they produce functions that are composed of rules $\mathcal{X} \rightarrow c_j$, indicating an association between attribute-values in $\mathcal{X} \subseteq x$, and a class $c_j \in y$. Next, we discuss properties of these rules.

Support and Confidence – The support of a rule $\mathcal{X} \rightarrow c_j$, denoted as $\sigma(\mathcal{X} \rightarrow c_j)$, indicates the number of examples in $\mathcal{S}$ in which $\mathcal{X}$ and c_j co-occur. The confidence of a rule $\mathcal{X} \rightarrow c_j$, denoted as $\theta(\mathcal{X} \rightarrow c_j)$, is an indication of how strongly $\mathcal{X}$ and c_j are associated. Support and confidence are defined in Equations 2 and 3:

$$\sigma(\mathcal{X} \rightarrow c_j) = |(x_i, y_i) \in \mathcal{S}| \text{ such that } \mathcal{X} \subseteq x_i \text{ and } y_i = c_j \tag{2}$$

$$\theta(\mathcal{X} \rightarrow c_j) = \frac{\sigma(\mathcal{X} \rightarrow c_j)}{|(x_i, y_i) \in \mathcal{S}| \text{ such that } \mathcal{X} \subseteq x_i} \tag{3}$$

Complexity – Often, the length of an explanation can be an indication of its complexity [Kolmogorov 1965]. A rule $\mathcal{X} \rightarrow c_j$ can be viewed as an explanation that leads to class c_j. In this case, the cardinality of $\mathcal{X} \rightarrow c_j$ (i.e., $|\mathcal{X}|$) is regarded as its complexity.

Usefulness – A rule $\mathcal{X} \rightarrow c_j$ matches an input x_i iff $\mathcal{X} \subseteq x_i$. Since the prediction provided by a rule is based on all the attribute-values that are in $\mathcal{X}$, only rules matching input x_i are used to estimate the corresponding output y_i. We say that a rule is useful if it matches at least one input in $\mathcal{T}$, otherwise the rule is said to be useless.

Hereafter, it will be denoted as $\mathcal{R}$ a set of rules extracted from $\mathcal{S}$. Also, it will be denoted as $\mathcal{R}^{x_i}$ the subset of $\mathcal{R}$ which contains only rules matching an input x_i. Finally, it will be denoted as $\mathcal{R}^{x_i}_{c_j}$ the subset of $\mathcal{R}^{x_i}$ which contains only rules predicting class c_j.

4.2. Eager Algorithms

The simplest algorithm to be presented, referred to as EAC-SR[1], performs the typical eager rule extraction strategy, and produces a function $f_{\mathcal{S}}$ using frequent rules in $\mathcal{S}$. In this case, $\mathcal{R}$ contains all rules r extracted from $\mathcal{S}$ such that $\sigma(r) \geq \sigma_{min}$, where σ_{min} is a user-specified minimum support threshold. Then, given an arbitrary input x_i, $f_{\mathcal{S}}(x_i)$ gives the predicted output for x_i, which is the class associated with the strongest rule in $\mathcal{R}^{x_i}$:

$$f_{\mathcal{S}}(x_i) = c_j \text{ such that } \theta(\mathcal{X} \rightarrow c_j) \text{ is } \mathrm{argmax}(\theta(r)) \; \forall \; r \in \mathcal{R}^{x_i} \tag{4}$$

[1]EAC-SR stands for *Eager Associative Classification using a Single Rule.*

4.3. Demand-Driven Algorithms

Although simple, EAC-SR suffers from two problems: (i) an infrequent, but useful rule may be not extracted from $\mathcal{S}$ (possibly hurting classification effectiveness), and (ii) a frequent, but useless rules may be extracted from $\mathcal{S}$ (incurring unnecessary overhead). Figure 1 (left) illustrates these problems. It shows the search space for rules, where black balls represent useful rules, and white balls represent useless rules. σ_{min} induces a border which separates frequent from infrequent rules. If σ_{min} is set too high, few useless rules are extracted, but several useful rules are missed. If σ_{min} is set too low, many useful rules are extracted, but a prohibitive, exponential number of useless rules is also extracted. As an alternative, we propose LAC-SR[2], which extracts rules on a demand-driven basis. Specifically, whenever an input $x_i \in \mathcal{T}$ is being considered, that input is used as a filter to remove from $\mathcal{S}$ all the attribute-values that are not in x_i. This process generates a *projected training data* (denoted as $\mathcal{S}^{x_i}$), which contains only attribute-values that are in x_i. LAC-SR directly produces $\mathcal{R}^{x_i}$ by extracting rules from $\mathcal{S}^{x_i}$ (instead of extracting rules from $\mathcal{S}$ and then filtering those matching x_i). LAC-SR produces a function $f_{\mathcal{S}}$ which, in fact, can be interpreted as a combination of multiple functions $f_{\mathcal{S}}^{x_i}$, where each $f_{\mathcal{S}}^{x_i}$ is used to predict only the output for input x_i. In the following we discuss the efficiency of LAC-SR.

Lemma 1 *All rules extracted from $\mathcal{S}^{x_i}$ (i.e., $\mathcal{R}^{x_i}$) are useful to input $x_i \in \mathcal{T}$.*

Proof: Since all inputs in $\mathcal{S}^{x_i}$ contain only attribute-values that are present in x_i, the existence of a rule $\{\mathcal{X} \rightarrow c_j\} \in \mathcal{R}^{x_i}$, such that $\mathcal{X} \not\subseteq x_i$, is impossible. ■

Theorem 1 *LAC-SR is efficient.*

Proof: Let q be the number of distinct attribute-values in $\mathcal{S}$. In this case, an upper bound for the hypothesis space $\mathcal{H}$, is the set of all possible rules in $\mathcal{S}$, which is clearly $|\mathcal{H}|=p\times 2^q$ (where p is the number of classes). According to Equation 1, $n \geq \frac{1}{\epsilon}\left(\ln(p \times 2^q) + \ln(\frac{1}{\delta})\right)$, and so, $n \geq \frac{1}{\epsilon}\left(\ln(p) + 0.69q + \ln(\frac{1}{\delta})\right)$. Thus, the number of required examples, n, increases only polynomially with q. It is also necessary to show that a function $f_{\mathcal{S}}$, for which $\mathcal{I}_{\mathcal{S}}[f_{\mathcal{S}}]=0$ is found in time polynomial in q (please, refer to Lemma 3.2 on page 33 in [Veloso 2009] for a proof that such $f_{\mathcal{S}}$ exists). Obviously, the number of all possible functions is exponential in q (i.e., $O(2^q)$ rules). However, since an arbitrary input $x_i \in \mathcal{T}$ contains at most l attribute-values (with $l \ll q$), then any rule matching x_i can have at most l attribute-values in its antecedent. Therefore, the number of possible rules matching x_i is $p \times (l + \binom{l}{2} + \ldots + \binom{l}{l}) = O(2^l) \ll O(q^l)$, and thus, the number of useful rules increases polynomialy in q. Since, according to Lemma 1, LAC-SR extracts only useful rules, then the complexity of LAC-SR also increases polynomialy in q. ■

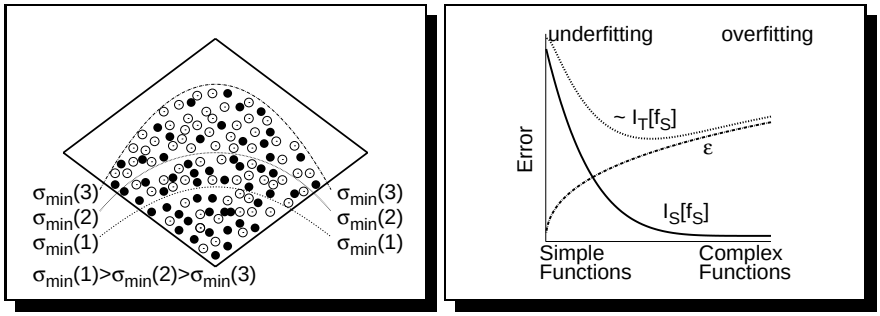

Figure 1. (left) The pruning dillema. (right) Structural risk minimization.

[2]LAC-SR stands for *Lazy Associative Classification using a Single Rule.*

Theorem 1 states that LAC-SR is efficient, since the number of useful rules grows polynomially in q (no matter the value of σ_{min}, which, in fact, can be arbitrarily low). Although efficient, improvements to LAC-SR are still possible. Particularly, LAC-SR uses the same prediction strategy of EAC-SR $-$ picks only the strongest rule to predict the output for a given input (i.e., Eq. 4). Such a simple pick may provide extremely biased predictions, and a safer strategy would be to combine the predictions of multiple rules, so that a collective prediction can be performed. Therefore we propose LAC-MR[3], which interprets $\mathcal{R}^{x_i}$ as a poll, where each rule $\{\mathcal{X}{\rightarrow}c_j\} \in \mathcal{R}^{x_i}$ is a vote given by $\mathcal{X}$ for output c_j. The weight of a vote $\mathcal{X}{\rightarrow}c_j$ depends on $\theta(\mathcal{X}{\rightarrow}c_j)$. Eq. 5 gives the score associated with output c_j for input x_i, and Eq. 6 gives the likelihood of c_j being the output of x_i. Finally, LAC-MR produces a function $f_{\mathcal{S}}^{x_i}$ for each x_i, which returns the output which receives the highest likelihood in a weighted voting process, as shown in Eq. 7.

$$s(x_i, c_j) = \frac{\sum \theta(r)}{\mid \mathcal{R}_{c_j}^{x_i} \mid} \qquad (5) \qquad\qquad \hat{p}(c_j|x_i) = \frac{s(x_i, c_j)}{\sum s(x_i, c_k)} \qquad (6)$$

$$f_{\mathcal{S}}^{x_i}(x_i) = c_j \text{ such that } \hat{p}(c_j|x_i) \text{ is argmax } (\hat{p}(c_k|x_i)), \text{ where } 1 \leq k \leq p \qquad (7)$$

Structural Risk Minimization – The algorithms discussed so far do not take into account the complexity of the functions they produce. The idea is that simple (or difficult) sub-problems (i.e., x_i) may demand simple (or complex) functions (i.e., $f_{\mathcal{S}}^{x_i}$). A general measure of the complexity of a function $f_{\mathcal{S}}^{x_i}$ is its VC-dimension [Guyon et al. 1992], denoted as $d_{f_{\mathcal{S}}^{x_i}}$ (please, refer to [Veloso 2009], page 16, for more details on how to compute VC-dimension [Guyon et al. 1992]). In order to compute $d_{f_{\mathcal{S}}^{x_i}}$, functions are first organized in a nested structure $\mathcal{F}$, such that $\mathcal{F}_1{\subseteq}\mathcal{F}_2{\subseteq}\ldots\mathcal{F}_l$, where functions in class $\mathcal{F}_k$ are simpler then functions in class $\{\mathcal{F}_{k+1} - \mathcal{F}_k\}$[4]. A way of controlling the complexity of $f_{\mathcal{S}}^{x_i}$ is to establish a relationship between $d_{f_{\mathcal{S}}^{x_i}}$ and how well $f_{\mathcal{S}}^{x_i}$ fits $\mathcal{S}$ (i.e., $\mathcal{I}_{\mathcal{S}}[f_{\mathcal{S}}^{x_i}]$). Structural Risk Minimization (SRM) provides this trade-off [Guyon et al. 1992], as shown in Eq. 8. The shape of this bound, which is shown in Figure 1 (right), can be exploited to select a function $f_{\mathcal{S}}^{x_i}$ with the most appropriate complexity for input x_i. The next algorithm to be proposed, LAC-MR-SRM[5], uses Eq. 7 to produce functions in increasing order of complexity (i.e., functions in $\mathcal{F}_k$ is considered before than functions in $\mathcal{F}_{k+1}$), and it uses Eq. 8 to find the simplest function $f_{\mathcal{S}}^{x_i}$ which provides the lowest empirical error.

$$\mathcal{I}_{\mathcal{T}}[f_{\mathcal{S}}^{x_i}] \leq \mathcal{I}_{\mathcal{S}}[f_{\mathcal{S}}^{x_i}] + \sqrt{\frac{d_{f_{\mathcal{S}}^{x_i}}\left(\ln \frac{2n}{d_{f_{\mathcal{S}}^{x_i}}} + 1\right) - \ln \frac{\delta}{4}}{n}} \qquad (8)$$

4.4. Empirical Results

In all experiments we used the standard k-fold cross-validation[6]. All experiments were performed on a Linux-based PC with Intel Pentium III 1.0 GHz and 1 GB RAM. We will present only a summary of the main results. The detailed experimental setup, including the description of baselines and parameters, can be found in [Veloso 2009].

[3]LAC-MR stands for *Lazy Associative Classication using Multiple Rules.*

[4]Functions in $\mathcal{F}_1$ are built using rules of cardinality 1, $\mathcal{F}_2$ using rules of cardinality at most 2, and so on.

[5]LAC-MR-SRM stands for *LAC-MR with Structural Risk Minimization.*

[6]The dataset (i.e., input/output pairs) is randomly partitioned into k sub-samples. A single sub-sample is used as test set, and the remaining k-1 sub-samples are used as training. The process is repeated k times, and each of the k sub-samples are used exactly once as test set. The final result is the average of the k runs.

Typical Classification Problems – In these problems, each object belongs to exactly one class, and the goal is to predict the correct class for a given object. We will report results using datasets collected from applications as diverse as cancer diagnosis (benign/malignant), and classification of radar returns from the ionosphere. Baselines include decision trees and kNNs. LAC-MR-SRM was the best performer, providing, on average, an accuracy number of 87.9%, while the baseline accuracy is no greater than 82.1%.

Multi-Label Classification – Now, each object may belong to multiple classes simultaneously. Algorithmic modifications were proposed to LAC-MR in order to deal with such problems. Basically, $\hat{p}(c_j|x_i)$ (i.e., Equation 6) is used to select classes that are more likely to be associated with the object ([Veloso 2009], page 62). We will report results concerning the prediction of the function of certain genes of the *Yeast Saccharomyces cerevisiae* organism. Depending on the evaluation metric used, LAC-MR based algorithms provide gains of more than 15%, over SVMs and other algorithms that were used as baselines.

Self-Training – In certain problems, the (high) cost associated with labeling examples in order to create a training data, may render vast amounts of examples unfeasible. Algorithmic modifications were proposed to LAC-MR, so that it can perform well in scenarios in which only few training examples are available. Basically, $\hat{p}(c_j|x_i)$ is used to detect predictions that are likely to be correct. These "safe" predictions are regared as new examples and inserted into the training data, which is automatically augumented with this new information ([Veloso 2009], page 98). We will report results concerning the identification of entities/persons associated with ambiguous names. LAC-MR based algorithms provide gains of more than 18% when compared against SVMs and Naive Bayes.

Ranking – In certain problems we may want to sort objects according to their likelihood of membership. An example is ranking documents returned by search engines – documents likely to be relevant should appear first than documents likely to be irrelevant. Modifications were proposed to LAC-MR, so that it can sort objects ([Veloso 2009], page 108). Basically, the rank of an object x_i is given by the linear combination of the likelihoods associated with each class: $\sum \big(c_j \times \hat{p}(c_j|x_i)\big)$. LAC-MR based algorithms provide gains ranging from 6.6% to 42% when compared against SVMs and other algorithms.

5. Conclusions and Final Remarks

In this paper we summarized some of the research contributions of [Veloso 2009]. Specificaly, we posed classification as a function approximation problem, which we showed that can be decomposed into simpler sub-problems. We also showed that approximating the target function on a demand-driven basis, taking into account demands of each subproblem, leads to better mapping functions. We presented efficient algorithms for classification and related extensions, and we provided theoretical evidence supporting them. We demonstrated the superiority of our algorithms using actual scenarios. We are currently extending our algorithms to deal with privacy-sensitive data [Veloso 2002, Veloso 2003]. Application scenarios such as sentiment analysis (`http://observatorio.inweb.org.br`) and Web polygraph (i.e., detecting lies in posts/statements on the Web) are also our current focus. Apart from our results, we have implemented a vast amount of software, currently available at `http://www.dcc.ufmg.br/~adrianov/software`.

Due to lack of space, we were not able to include: (1) an entire branch of new algorithms based on Empirical Risk Minimization ([Veloso 2009], pages 14, 36, and 53), (2) discussions about Calibrated and Multi-Metric Classification, and algorithms for these problems ([Veloso 2009], pages 69 and 84), (3) several experiments, and (4) some references to papers published during the development of [Veloso 2009].

References

Boser, B., Guyon, I., and Vapnik, V. (1992). A training algorithm for optimal margin classifiers. In *Conf. on Computational Learning Theory*, 144–152. Springer.

Breiman, L. (1984). Classification and regression trees. *Wadsworth Intl.*

Cover, T. and Hart, P. (1967). NN pattern classification.*Trans.on Inf.Theory*,13(1):21-27.

Cucker, F. and Smale, S. (2001). On the mathematical foundations of learning. *Bulletin of the American Mathematical Society*, 39(1):1–49.

Guyon, I., Boser, B., and Vapnik, V.(1992). Automatic capacity tuning of very large VC-dimension classifiers. In *Conf.on Neural Inf. Proc. Systems (NIPS)*, 147–155. MIT.

Joachims, T. (2006). Training linear SVMs in linear time. In *Conf. on Knowledge Discovery and Data Mining (SIGKDD)*, 217–226. ACM.

Kolmogorov, A. (1965). Three approaches to the quantitative definition of information. *Problems of Information Transmission*, 1:4–7.

Poggio, T. and Girosi, F. (1998). A sparse representation for function approximation. *Neural Computation*, 10(6):1445–1454.

Quinlan, J. (1986). Induction of decision trees. *Machine Learning*, 1:81–106.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408.

Valiant, L. (1984). A theory of the learnable. *Commun. ACM*, 27(11):1134–1142.

Veloso, A. (2009). *Demand-Driven Associative Classification.* PhD Thesis, UFMG.

Veloso, A., Zaki,M., Meira,W., and Gonçalves, M. (2009a). Competence-conscious classification (**best paper runner up**).In *Data Mining Conf. (SDM)*,918-929.SIAM.

Veloso, A., Zaki, M., Meira, W., Gonçalves, M., and Mossri, H. (2009b). Competence-conscious associative classification.*Stat.Analysis and Data Mining*,2(5-6):361–377.

Veloso, A., Meira, W., Zaki, M., Gonçalves, M., and Mossri, H. (2009c). Calibrated lazy associative classification. (accepted, to appear). *Information Sciences*.

Veloso, A., Ferreira, A., Gonçalves, M., and Laender, A. (2009d). Cost-effective on-demand associative name disambiguation. *Inf.Proc. and Management. (submitted)*.

Veloso, A., Gonçalves, M., and Meira, W. (2008a). Learning to rank at query-time using association rules. In *Conf. on Res. and Dev. in Inf. Ret. (SIGIR)*, 267–274. ACM.

Veloso, A., Meira, W., and Zaki, M. (2008b). Calibrated lazy associative classification (**best paper runner up**). In *Braz. Symp. on Databases (SBBD)*, 135–149. SBC.

Veloso, A., Meira, W., Gonçalves, M., and Zaki, M. (2007a). Multi-label lazy associative classification. In *Euro. Conf. on Data Mining and Knowl. Disc.(PKDD)*, 605–612.

Veloso, A., Meira, W. (2007b). Automatic moderation of comments in a large on-line journalistic environment. In *Int.Conf. on Weblogs and Social Media (ICWSM)*.AAAI.

Veloso, A. and Meira, W. (2007). Efficient on-demand opinion mining. In *Braz. Symp. on Databases (SBBD)*, 332–346. SBC.

Veloso, A., Meira, W., Gonçalves, M., and Zaki,M.(2006a). Multi-evidence, lazy associative classification. In *Conf.on Inf.and Know.Managem.(CIKM)*,218–227.ACM.

Veloso, A., Meira, W., and Zaki, M. J. (2006b). Lazy associative classification. In *Int. Conf. on Data Mining (ICDM)*, 645–654. IEEE.

Veloso, A. and Meira, W.(2005). Rule generation and rule selection techniques for cost-sensitive classification. In *Braz. Symp. on Databases (SBBD)*, 295–309. SBC.

Veloso, A., and Meira, W.(2004). Efficient Data Mining for Frequent Itemsets in Evolving and Distributed Data (**best Master Thesis**). In *Braz. Comp. Soc. Conf. (CTD)*. SBC.

Veloso, A., Meira, W., and Parthasarathy, S.(2003). Efficient, Accurate and Privacy-Preserving Data Mining for Frequent Itemsets in Distributed Databases (**best paper runner up**). In *Braz. Symp. on Databases (SBBD)*, 281–292. SBC.

Veloso, A., Meira, W., and Bunte, M.(2002). Mining Reliable Models of Associations in Dynamic Databases(**best paper**). In *Braz. Symp. on Databases (SBBD)*,263–277.SBC.