How to Get a Free Lunch: A Simple Cost Model for Machine Learning Applications

Pedro Domingos
Artificial Intelligence Group
Instituto Superior Técnico
Lisbon 1096, Portugal
pedrod@gia.ist.utl.pt

Abstract

This paper proposes a simple cost model for machine learning applications based on the notion of net present value. The model extends and unifies the models used in (Pazzani *et al.*, 1994) and (Masand & Piatetsky-Shapiro, 1996). It attempts to answer the question "Should a given machine learning system now in the prototype stage be fielded?" The model's inputs are the system's confusion matrix, the cash flow matrix for the application, the cost per decision, the one-time cost of deploying the system, and the rate of return on investment. Like Provost and Fawcett's (1997) ROC convex hull method, the present model can be used for decision-making even when its input variables are not known exactly. Despite its simplicity, it has a number of non-trivial consequences. For example, under it the "no free lunch" theorems of learning theory no longer apply.

Introduction

A key point in the life cycle of a machine learning application is the decision to go from prototype system to fielded application. Many projects do not make it past this stage. How should this decision be made? What factors should be taken into account? How should they be combined? Clearly, knowing the predictive accuracy of the system is not sufficient to make an informed and intelligent decision, although this is all most authors report. Many other criteria for evaluating the output of learning systems have been proposed and used (e.g.,(Piatetsky-Shapiro 1991; Silberschatz & Tuzhilin 1995; Nakhaeizadeh & Schnabl 1997)), but they often involve a substantial degree of subjectivity, and do not directly address the deployment decision. A number of initial proposals and studies have been made that take the cost of decisions into account (Breiman *et al.* 1984; Pazzani *et al.* 1994; Turney 1995; Masand & Piatetsky-Shapiro 1996; Matheus, Piatetsky-Shapiro, & McNeill 1996; Provost & Fawcett 1997; Turney 1997). Simultaneously, a large applicable literature exists in the fields of management and decision theory (Brealey & Myers 1996; Keeney & Raiffa 1976; Berger 1985; Henrion, Breese, & Horvitz 1992), but it has so far made little contact with machine learning. The goal of this paper is to begin closing this gap.

This paper will focus on standard classification applications, but its conclusions are qualitatively applicable to many other types. We will use a banking problem as a running example: Bank X has applied machine learning to its database(s) of past loan applications and their results, and produced a prototype system for automatically making loan decisions (i.e., for classifying loan applications as "good credit risk" or "bad credit risk"). It now needs to decide whether to deploy this system in the field.

A Cost Model

Deploying a machine learning system is an investment decision like any other, and it can be made in the standard way: deploy the system if its net present value (NPV) is positive (Brealey & Myers 1996). The NPV of an investment is the sum of the cash flows it generates, discounted by the rate of return. If C_t is the cash flow during period t (typically a year), and r is the rate of return demanded by the investor for each period (assumed constant):

$$NPV = C_0 + \sum_{t=1}^{\infty} \frac{C_t}{(1+r)^t} \qquad (1)$$

C_0 is the initial cash flow, and is usually negative, representing the initial investment. In the case of a machine learning system, it is the one-time cost of deploying the system. This can include the cost of new equipment, software installation, personnel training, reorganization of procedures, etc. The cash flow at time $t > 1$ is the result of the decisions made by the system.[1] The cash flow associated with a single decision has two components: the cost of making the decision, and the cash flow resulting from the decision. The first component is an "internal" one: it includes the cost of keeping the system running, gathering the information needed to make the decision, periodically retraining the system

[1] If the investment has a limited time horizon t_0, $C_t = 0$ for $t > t_0$.

if necessary,[2] etc. For convenience, this cost will be viewed as having a fixed value per decision made. The second component is external: it will depend on the effects of the decision. In the case of Bank X, the cash flow will (presumably) be positive if the bank decides to loan and the loan is repaid, negative if the loan is not repaid, and zero if no loan is made.[3] In general, given m classes, an $m \times m$ *cash flow matrix* Q is required, with component $Q(i,j)$ representing the cash flow that results if the true class is j and class i is predicted. This still ignores that a given (i,j) may result in different cash flows in different situations (for example, depending on the dollar value of the loan), but in this paper we will simply take Q to be a pre-compiled matrix of the relevant averages. To compute the expected cash flow given Q, we also need to know the system's *confusion matrix* P, where component $P(i,j) = P(i \wedge j)$ is the probability that the true class is i and class j is predicted. The $P(i,j)$ estimates can be obtained the same way accuracy estimates usually are: by dividing the database randomly into training and test cases, learning on the training cases, and counting the number of test cases for which i is the true class and j is predicted. The expected cash flow C associated with a single decision is then:

$$C = D + \sum_{i=1}^{m} \sum_{j=1}^{m} P(i,j)\, Q(i,j) \qquad (2)$$

where $D < 0$ is the decision-making cash flow (i.e., $|D|$ is the cost of making the decision). If n decisions are made on average during each time period, $C_t = nC$, and:

$$NPV = C_0 + nC \sum_{t=1}^{\infty} \frac{1}{(1+r)^t} = C_0 + \frac{nC}{r} \qquad (3)$$

Strictly speaking, the latter identity applies only if the system is used in perpetuity; more pragmatically, it is valid if the system stays in operation long enough for its exact date of retirement to have little effect on the NPV. We will use it because of its simplicity, and because all conclusions will be similar in the two cases.

Equation 3 represents the NPV of deploying the system if nothing is currently being done in its place (i.e., if Bank X starts operating today from scratch). More frequently, the machine learning system will modify (or replace) an existing procedure (for example, manual loan evaluation, a hand-built expert system, or a combination). In this case, the "old" cash flow associated with each decision should be subtracted from the "new" one. A more transparent way of doing this is to compute the NPV of continuing with the current procedure, and deploy the machine learning system if its NPV is greater. An important difference between the two NPVs is that the current solution has no C_0 term, since it involves no new investment. Using M to index terms associated with the machine learning system, and L to index terms associated with maintaining the current solution:

$$NPV = NPV_M - NPV_L = \left(C_0 + \frac{nC_M}{r}\right) - \frac{nC_L}{r} \qquad (4)$$

Notice that in general D_M and D_L will be different, as will $P_M(i,j)$ and $P_L(i,j)$, but $Q(i,j)$ is the same for the two alternatives.

Some Consequences of the Model

By Equation 4, the machine learning system should be deployed iff:

$$\left(C_0 + \frac{nC_M}{r}\right) - \frac{nC_L}{r} > 0 \qquad (5)$$

Let $C_M - C_L = \Delta C$. Given that C_0 is negative, this inequality can be rearranged into a more expressive form:

$$\Delta C > \frac{r}{n}|C_0| \qquad (6)$$

In other words: for deployment to be justified, the gain in cash flow per decision from using the system has to be at least a fraction r/n of the cost of deployment. The frontier between deployment and non-deployment is a straight line in the $(\Delta C, |C_0|)$ (half-)plane with slope n/r. This is illustrated graphically in Figure 1.

The region of deployment increases with the number of decisions per year n and decreases with the rate of return r. Given these two factors, it is possible to determine if a system should be deployed even if the cash flow and confusion matrices, and the initial and per-decision costs, are not known exactly. It is also possible to perform sensitivity analysis. If the region in the $(\Delta C, |C_0|)$ plane representing the plausible range of those quantities falls entirely on the right side of the dividing line, the decision to deploy is robust, and conversely if it falls entirely on the left. If the dividing line intersects the plausible region, the decision is sensitive to the estimated values of of ΔC and C_0, and it may be advisable to attempt to determine these values more precisely, or to take into account the increased risk.

A particularly simple case occurs if the learning system emulates the previous decision process (almost) perfectly. In this case the cash flow matrix Q becomes irrelevant, and deployment depends solely on whether the compounded savings per decision $\Delta D = D_M - D_L$ exceed the cost of deployment. The savings per decision may result from going from a manual to an automated decision process, or from one automated process to another requiring lower maintenance.

[2] In general this will change the system's confusion matrix (see below), but here we will ignore this effect. Taking it into account is clearly non-trivial.

[3] More precisely, the decision to lend at time t will itself generate a sequence of cash flows (loan, interest, repayment of principal) whose net present value at that time can be computed by the usual method. For simplicity, this NPV will be taken as an "instant" cash flow at time t.

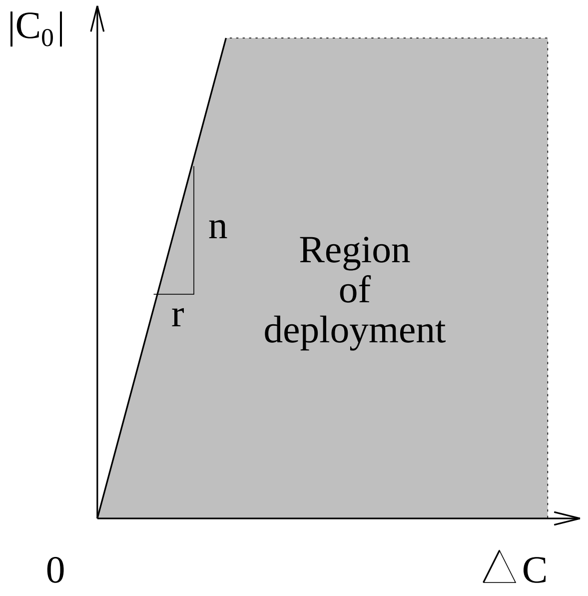

Figure 1: Region of deployment for a machine learning system.

These savings may also justify deploying a system that makes worse decisions on average, if ΔD sufficiently exceeds $\sum_{i=1}^{m}\sum_{j=1}^{m}[P_L(i,j) - P_M(i,j)]\,Q(i,j)$ (Equations 2 and 3). Conversely, a system that makes better decisions on average may not be worth fielding, for example if it has a very high cost of deployment $|C_0|$. (A negative deployment decision does not necessarily mean abandoning the system; it may simply mean that it needs to be further developed before it is ready for fielding.)

So far we have assumed that r is constant, but this need not be the case. Given a region of plausible values on the $(\Delta C, |C_0|)$ plane, the deployment decision's sensitivity to r can be observed by varying the slope of the decision frontier. We have also assumed that r is the same for the current solution and for the learning system, but in general it may differ, if the two have different risk characteristics. A common measure of risk is the variance of returns (Brealey & Myers 1996). Lower variance will lead to a lower return being demanded. Thus, NPV_M may be greater than NPV_L, even if the learning system makes on average worse decisions than the current procedure, if it reduces the variance in cash flows. This variance is proportional to:

$$\frac{1}{\sqrt{n}}\sum_{i=1}^{m}\sum_{j=1}^{m} P(i,j)\,[Q(i,j) - C + D]^2 \qquad (7)$$

Thus, risk is reduced if the learner makes fewer of the decisions whose cash flow differs most from the average, and more of the decisions whose cash flow is close the average. For example, if bad loans on average generate a larger (negative) cash flow than good ones, rejecting more loans may reduce risk and result in a higher NPV, even if it means rejecting more good loans and reducing the average cash flow per decision. (This will also be the case if, for example, Bank X now makes more average-sized performing loans and fewer very large and very small ones, and similarly for non-performing loans. However, this aspect is not covered by the current model, because the values of $Q(i,j)$ for each (i,j) are already pre-computed averages.)

The model can also be used to compare two alternative machine learning systems M_1 and M_2. Let $C_{0,1}$ be M_1's initial cash flow, C_1 its expected cash flow per decision, and similarly for M_2. M_2 is preferable to M_1 iff:

$$C_2 - C_1 > \frac{r}{n}(C_{0,1} - C_{0,2}) \qquad (8)$$

A useful notion is that of *effective value* EV_M of a machine learning system M. This is the NPV that M actually results in: $NPV_M - NPV_L$ if the system is deployed, and 0 if it is not. Since in our model deployment occurs iff $NPV_M - NPV_L > 0$, $EV_M = \max\{NPV_M - NPV_L, 0\}$. In other words, if M does not increase revenue it is effectively worthless; but it is never worth less than zero, because we will not deploy it if it would decrease revenue.

The proposed model can also be used to gauge which applications are *a priori* more likely to yield a system worth deploying. Some factors that favor a machine learning solution are: a low $|C_0|$ (for example if the necessary computing and organizational structure is already in place); a high D_L and low D_M (for example, if an expert, time-consuming manual procedure is replaced by a purely computational one); a high n and low r, because this will reduce the weight of C_0 in the total cost (see Equation 6). In short, the most promising areas to apply machine learning are those where risk is relatively low, a large number of decisions per year is made, the current procedure is expensive, and the computational and organizational infra-structure needed for the machine learning solution is (mostly) already in place. (However, a small n may also favor the use of a machine learning system, if the latter reduces variance; by Equation 7 a small n will make this effect more pronounced.)

How to Get a Free Lunch

Perhaps the most remarkable consequence of the proposed model is that in it the famous "no free lunch theorems" of learning theory (Schaffer 1994; Wolpert 1996) are no longer valid. Roughly speaking, these theorems say that no classifier can outperform another on average over all possible classification problems, and implicitly question the utility of much learning research. As Rao *et al.* (1995) have shown, these theorems do not necessarily apply if not all classification problems are equally likely. However, under the current model, a classifier can globally outperform another even without such a caveat.

Schaffer's (1994) is the simplest treatment, so we will follow it here. Schaffer uses as a measure of performance the *generalization accuracy* of the classifier. In terms of the current model, this is equivalent to: setting $Q(i,j) = 1$ when $i = j$ and $Q(i,j) = 0$ otherwise; excluding all training cases from the (exact) computation of the confusion matrix P; and ignoring the existence of initial and decision-making costs, $|C_0|$ and $|D|$, and of a current procedure ($NPV_L = 0$). Let M_1 and M_2 be two classifiers, and let S be the set of all possible domains. A *domain* is a set of class assignments to all instances in the instance space. Schaffer's result can be stated as follows.

Theorem 1 *If the confusion matrices P_1 and P_2 are computed using only examples not in the training set, $Q(i,j) = 1$ when $i = j$ and $Q(i,j) = 0$ otherwise, $C_{0,1} = C_{0,2} = 0$, $D_1 = D_2 = 0$, and $NPV_L = 0$ then:*

$$\forall M_1, M_2 \ \sum_S (EV_2 - EV_1) = 0$$

In other words, no classifier is more profitable than another, on average, over all domains. This result becomes trivially false if $C_{0,1} \neq C_{0,2}$ or $D_1 \neq D_2$. The more interesting case occurs when costs are equal for the two systems, which is what we will assume here. Our result can be formally stated thus.

Theorem 2 *If the confusion matrices P_1 and P_2 are computed using only examples not in the training set, $C_{0,1} = C_{0,2} = C_0$, $D_1 = D_2 = D$, and $NPV_L = 0$, then:*

$$\exists M_1, M_2 \ \sum_S (EV_2 - EV_1) > 0$$

Proof. Like Schaffer, we will assume an example distribution and training set are given, and for simplicity not index to them, noting that the results are valid for any such distribution and training set. We will also consider only two-class problems, again without loss of generality. It is not hard to see that simply going from generalization accuracy to an arbitrary cost/cash-flow matrix does not alter Schaffer's results. Thus, to keep matters simple, we will continue with the Q values used in Theorem 1. The accuracy of a classifier M_k is then $A_k = \sum_{i=1}^m \sum_{j=1}^m P_k(i,j)\, Q(i,j)$ (Equation 2). Let A_0 be the minimum accuracy required to justify deployment (i.e., for the classifier to pay for its deployment and decision costs): $A_0 = -D - rC_0/n$, by Equation 3. By the definition of effective value, $EV_k = C_0 + rA_k/n$ if $A_k > A_0$, and $EV_k = 0$ otherwise ($k \in \{1,2\}$). Let S_N be the set of domains where $A_1 \leq A_0$ (i.e., where A_1 would not be deployed), and let S_Y be the set of domains where $A_1 > A_0$ (i.e., where A_1 would be deployed). Suppose $A_2 > A_1$ in some subset S_{YY} of S_Y, $A_2 < A_1$ in some subset S_{NN} of S_N, and $A_2 = A_1$ in all other domains. Since in the the conditions of Theorem 1 $EV_k = A_k$, this is allowed by that theorem as long as $\sum_{S_{YY}} (A_2 - A_1) = -\sum_{S_{NN}} (A_2 - A_1)$. In S_{NN} $EV_1 = EV_2 = 0$, since $A_2 < A_1 < A_0$. In S_{YY} $EV_2 > EV_1$, since $A_2 > A_1 > A_0$ and thus $EV_k = C_0 + rA_k/n$. In all other domains $EV_1 = EV_2$. Therefore in some domains $EV_2 - EV_1 > 0$ and in all others $EV_2 - EV_1 = 0$, and $\sum_S (EV_2 - EV_2) > 0$. □

Theorem 2 generalizes easily to the case where M_1 and M_2 are being considered as alternative replacements for an existing procedure L, instead of deployed from scratch. Theorem 2 can be informally stated thus:

> *A classifier M_2 will have a globally higher effective value than another classifier M_1 if its generalization accuracy (or cash flow per decision) is higher in domains where M_2 is accurate enough to be deployed, and lower in domains where M_1 is not accurate enough for deployment.*

Thus, the way to produce a classifier M_2 that dominates another M_1 is to make M_2 less accurate in domains where M_1 would not be deployed anyway, and more accurate in those where it would. This is illustrated in Figure 2. M_1 and M_2 respect the "no free lunch" theorems, since they both have an average accuracy of 50% over all S. However, M_2 has a higher average effective value than M_1, since only the area above A_0 counts for purposes of computing EV. In short, a good agenda for research is to keep improving the current classifiers in the domains where they do well, regardless of the fact that this makes them worse where they already do poorly. Not surprisingly, this is largely what is done in practice. Another frequent approach that the "no free lunch" theorems cast doubt on but the present results validate is multistrategy learning: attempting to combine the best features of two or more classifiers in a single one (Michalski & Tecuci 1994; Chan, Stolfo, & Wolpert 1996). Figure 3 shows how this can be done: classifier M_3 is as good as M_1 in M_1's region of expertise and as good as M_2 in M_2's region of expertise, by being worse than either in regions where neither would be deployed. Thus M_1 and M_2 are effectively superseded by M_3: having the latter, there is nowhere in S the former are worth retaining.

Theorem 2 implies that machine learning researchers can be more optimistic about their work than previously assumed. Contrasted with Theorem 1, it also illustrates how abstract results can be misleading.

Related Work

CART (Breiman *et al.* 1984) was one of the first learning systems to allow explicitly incorporating cost considerations. However, its method of variable misclassification costs only works for two classes, and Pazzani *et al.* (1994) found that in practice CART's altered priors method was not very effective in reducing misclassification costs. The cost model proposed here is a generalization of that used by Pazzani *et al.*, who always consider the cost to be zero when the correct decision is made, and ignore the deployment and decision-making costs. Our model is formulated in terms of cash

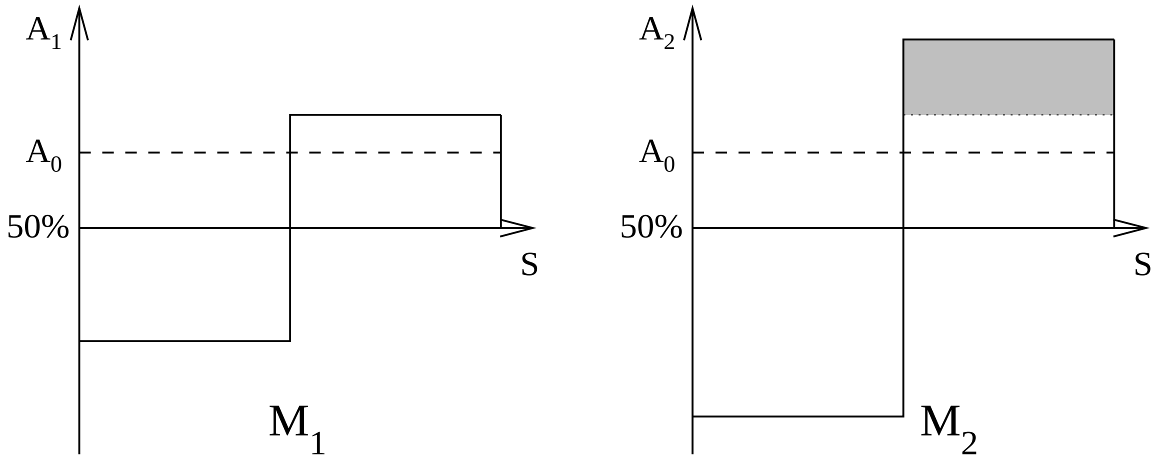

Figure 2: Improving the global effective value of a classifier. The shaded area represents the accuracy gained at no cost.

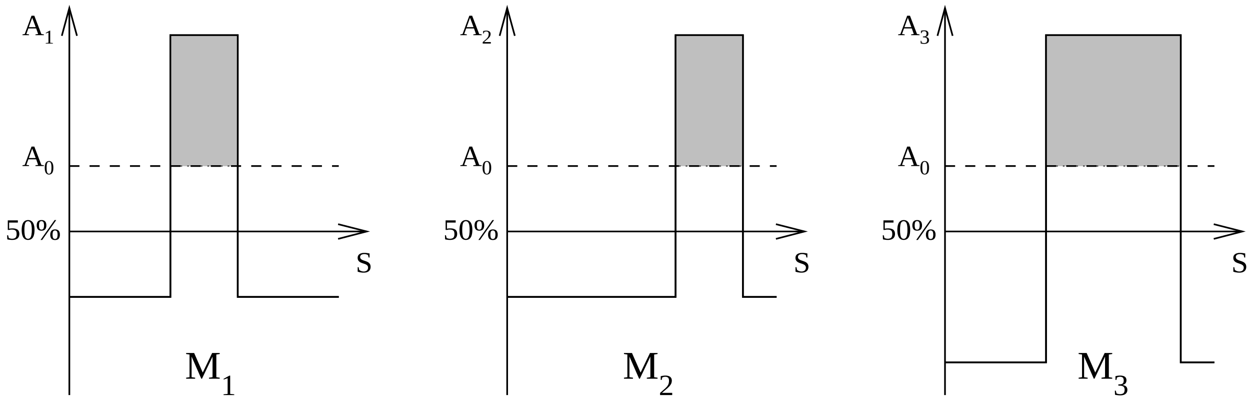

Figure 3: How multistrategy learning works. The shaded area represents the global effective value of each classifier.

flows instead of costs, because this is more compatible with the net present value model, and because treating revenues as negative costs is awkward. Masand and Piatetsky-Shapiro (1996) focus on a database marketing domain, and consider a payoff per decision of $rp-c$, where c is the cost of making an offer to a customer, r is the expected revenue if the offer is accepted, and p is the probability of acceptance. Our proposal also has this model as a special case. Provost and Fawcett (Provost & Fawcett 1997) propose a method for evaluating classifiers in two-class domains in terms of the true positive and false positive rates. Despite its intuitive appeal, this method may be hard to generalize to arbitrary cost matrices, and does not take the deployment and decision-making costs into account. The method we propose shares some of the same goals of simple visualization and robustness to imprecise information.

Another learning system that allows explicit consideration of costs is described in (Turney 1995). There is a relatively small but growing literature on cost-sensitive learning; see (Turney 1997) for an online bibliography. Some of the systems referenced in this bibliography take into account that different attributes have different costs of evaluation (e.g., some medical tests are more expensive than others); in our model, this corresponds to having a variable D term (Equation 2).

If the utility of a decision is equated with the cash flow it generates, Equation 2 minus the D term corresponds to computing the expected utility of the system's decision (Keeney & Raiffa 1976). If the learner is being considered as an additional source of information to the current procedure (i.e., M includes L), $NPV_M - NPV_L$ corresponds to the net expected value of the information (NEVI) induced by the learner,[4] generalized to take time and the rate of return into account. Information value theory was originally introduced by Howard (1966) , and developed in the context of resource-limited reasoning by Horvitz (1990) . Here we propose applying it to the machine learning process

[4]With the maximization of utility implicit in M's/L's choice of class.

(i.e., considering the cost and value of learned information).

Future Work

The main limitation of the current model is that it assumes a constant cash flow matrix Q, independent of the specific decision being made at each point. In general, Q may depend on attributes of the case in question (e.g., size of the loan) and external factors (e.g., Bank X's credit policy at the time). In order to preserve generality, we have not coupled the cost model with any particular type of induced model (e.g., decision trees or neural networks). However, if the induced model is an explicit representation of a probability distribution (e.g, a Bayesian network (Henrion, Breese, & Horvitz 1992)), the inclusion of attribute-dependent cost and confusion information is conceptually straightforward. On the other hand, it may be computationally hard, requiring approximations (e.g., (Horvitz, Suermondt, & Cooper 1989)) and/or careful management of computation (e.g., (Horvitz 1997)). Many classification models also have implicitly associated probability models (e.g., the leaf probabilities in decision trees), and using them may be computationally easier.

In the future we would like to apply the proposed model to specific areas (e.g., banking, database marketing, health care). This will allow us to determine how useful it is, identify where the main difficulties in applying it lie, and develop extensions.

Conclusion

This paper proposed a simple cost model for machine learning applications that generalizes and unifies several previous ones. This model uses as inputs the cash-flow matrix for the application, the confusion matrices for the alternatives being considered, their decision-making and deployment costs, and the rate of return. The model helps to answer the question "Should a given machine learning system now in the prototype stage be fielded?" It also allows us to identify *a priori* promising application areas, where machine learning is most likely to succeed. An interesting consequence of the model is that it shows the "no free lunch" theorems of learning theory do not apply under more realistic cost considerations. This lends support to much recent machine learning, statistical and data mining research.

References

Berger, J. O. 1985. *Statistical Decision Theory and Bayesian Analysis.* New York, NY: Springer-Verlag.

Brealey, R. A., and Myers, S. C. 1996. *Principles of Corporate Finance.* New York, NY: McGraw-Hill, 5th edition.

Breiman, L.; Friedman, J. H.; Olshen, R. A.; and Stone, C. J. 1984. *Classification and Regression Trees.* Belmont, CA: Wadsworth.

Chan, P.; Stolfo, S.; and Wolpert, D., eds. 1996. *Proceedings of the AAAI-96 Workshop on Integrating Multiple Learned Models for Improving and Scaling Machine Learning Algorithms.* Portland, OR: AAAI Press.

Henrion, M.; Breese, J.; and Horvitz, E. 1992. Decision analysis and expert systems. *AI Magazine* 12:64–91.

Horvitz, E.; Suermondt, H.; and Cooper, G. 1989. Bounded conditioning: Flexible inference for decisions under scarce resources. In *Proceedings of the Fifth Conference on Uncertainty in Artificial Intelligence,* 182–193. Windsor, Canada: Morgan Kaufmann.

Horvitz, E. 1990. *Computation and Action Under Bounded Resources.* Ph.D. Dissertation, Department of Computer Science, Stanford University, Stanford, CA.

Horvitz, E. 1997. Models of continual computation. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence,* 286–293. Providence, RI: AAAI Press.

Howard, R. A. 1966. Information value theory. *IEEE Transactions on Systems Science and Cybernetics* 2:22–26.

Keeney, R. L., and Raiffa, H. 1976. *Decisions with Multiple Objectives: Preferences and Value Trade-Offs.* New York, NY: Wiley.

Masand, B., and Piatetsky-Shapiro, G. 1996. A comparison of approaches for maximizing business payoff of prediction models. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining,* 195–201. Portland, OR: AAAI Press.

Matheus, C. J.; Piatetsky-Shapiro, G.; and McNeill, D. 1996. Selecting and reporting what is interesting: The KEFIR application to healthcare data. In Fayyad, U. M.; Piatetsky-Shapiro, G.; Smyth, P.; and Uthurusamy, R., eds., *Advances in Knowledge Discovery and Data Mining.* Menlo Park, CA: AAAI Press. 495–515.

Michalski, R. S., and Tecuci, G., eds. 1994. *Machine Learning: A Multistrategy Approach.* San Mateo, CA: Morgan Kaufmann.

Nakhaeizadeh, G., and Schnabl, A. 1997. Development of multi-criteria metrics for evaluation of data mining algorithms. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining,* 37–42. Newport Beach, CA: AAAI Press.

Pazzani, M.; Merz, C.; Murphy, P.; Ali, K.; Hume, T.; and Brunk, C. 1994. Reducing misclassification costs. In *Proceedings of the Eleventh International Conference on Machine Learning,* 217–225. New Brunswick, NJ: Morgan Kaufmann.

Piatetsky-Shapiro, G. 1991. Discovery, analysis, and presentation of strong rules. In Piatetsky-Shapiro, G., and Frawley, W. J., eds., *Knowledge Discovery in Databases*. Menlo Park, CA: AAAI Press. 229–248.

Provost, F., and Fawcett, T. 1997. Analysis and visualization of classifier performance: Comparison under imprecise class and cost distributions. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, 43–48. Newport Beach, CA: AAAI Press.

Rao, R. B.; Gordon, D.; and Spears, W. 1995. For every action, is there really an equal and opposite reaction? Analysis of the conservation law for generalization performance. In *Proceedings of the Twelfth International Conference on Machine Learning*, 471–479. Tahoe City, CA: Morgan Kaufmann.

Schaffer, C. 1994. A conservation law for generalization performance. In *Proceedings of the Eleventh International Conference on Machine Learning*, 259–265. New Brunswick, NJ: Morgan Kaufmann.

Silberschatz, A., and Tuzhilin, A. 1995. On subjective measures of interestingness in knowledge discovery. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining*, 275–281. Montréal, Canada: AAAI Press.

Turney, P. 1995. Cost-sensitive classification: Empirical evaluation of a hybrid genetic decision tree algorithm. *Journal of Artificial Intelligence Research* 2:369–409.

Turney, P. 1997. Cost-sensitive learning bibliography. Online bibliography, Institute for Information Technology of the National Research Council of Canada, Ottawa, Canada. http://ai.iit.nrc.ca/bibliographies/-cost-sensitive.html.

Wolpert, D. 1996. The lack of a priori distinctions between learning algorithms. *Neural Computation* 8:1341–1390.