A Novel Approach for Confidence Estimation using Support Vector Machines for more Accurate Value Prediction

Snigdha M. Mohapatra
Dept. of Computer Science
Centurion University of Technology and Management

Pradipta K. Mishra
Dept. of Computer Science
Centurion University of Technology and Management

ABSTRACT

Data dependencies create hurdles in exploiting ILP among instructions. To overcome them, data value predictors are used which guess instructions' result before it is actually executed. Thus, future instructions which depend on the outcome of that instruction executes sooner. But, since Value Prediction accuracy is very crucial in determining the amount of parallelism that can be exploited, Confidence estimation is used along with it to lessen the value prediction misprediction penalty by guessing whether or not to use a value prediction result. Previous confidence estimators were based on perceptrons which had the limitation of learning only linearly separable functions,[2, 24]. But sometimes linear inseparability may arise when a correct prediction on a past instruction causes the current instruction to predict incorrectly [25]. As Support Vector Machines belong to a family of generalized linear classifier and can be interpreted as extension of perceptron, they are both linear and non linear classifiers and are computationally more efficient than perceptrons. Thus, we propose a confidence estimator using SVM's in which the prediction accuracy of previous instructions is used to estimate the confidence of current prediction and decide based on its results whether or not the prediction is likely to be correct. The classification algorithm of SVM is implemented using MATLAB platform, and its novel learning methods have been applied on different data sets having two classes.

Keywords

Value Prediction, Confidence Estimation, SVM.

1. INTRODUCTION

Extensive research has been done in the area of data value prediction for overcoming these data dependencies [7, 8, 9, 10, 11,17,18,20 and 21]. The goal of data value prediction is to guess the outcome of an instruction before it is actually executed, allowing future instructions that depend on its output to be executed earlier. Data dependency is a normal situation in which the data that the instructions use depend upon the data created by other instructions, or the data is stored in locations which are used by other instructions. This problem is describes below. Let us consider two instructions:

1. Add r1, r2, r3 // r1=r2+ r3
2. Sub r3, r1, r2

Instruction 1 produces a result r1, which is the sum of r2 and r3. Instruction 2 uses the result of instruction1 , i.e. r1 and calculates the difference r3. Thus here, Instruction 2is said to be data dependent on Instruction 2, because it uses Instruction1's result. Since r1 is not known until Instruction 1 executes, they can't be executed in parallel. Data value predictors look for patterns among data produced in different iterations of static instructions. Accurate prediction can be attained when the repeated outcomes of a particular instruction follow easily perceptible patterns.

Accuracy is a major problem with data value prediction. Even in most advanced data predictors, 30% to 60% of the predictions are incorrect [17]. If an instruction is wrongly predicted, and that incorrect prediction is used for next data dependent instructions, then all of those instructions must be executed again. This causes high misprediction penalty. Thus in these cases, it is better not to predict than to mispredict. This is the reason why most data value predictors use a confidence estimator which determines whether or not a prediction for a particular instruction is likely to be correct or not[13]. If the estimator has high confidence in a prediction, then the predicted value is used by the data dependent instructions. Else, the prediction is discarded and the dependent instructions wait for the current instruction to be actually executed.

A typical confidence estimator tries to determine the accuracy of a prediction instructions predictability) by looking at whether the last several predictions for that instruction were correct or not. If they were all correct then naturally the next predictions should also be correct. But if the instruction was recently mispredicted, then the new prediction is also not trusted. But this is a localized approach; i.e. it doesn't consider the effect of other surrounding instructions on the current instruction which is to be predicted. So, there exist correlations between predictability of different instructions, especially if those instructions are data dependent [17]. Hence, an instructions prediction outcome may be correct only if a certain prior instructions prediction outcome was correct.

However, in order to make use of other instructions prediction accuracies, we must determine which surrounding instructions affect the current instruction. It is found that these predictability correlations tend to follow linearly separable as well as linearly inseparable patterns [25].

As Support Vector Machines belong to a family of generalized linear classifier and can be interpreted as extension of perceptron. They are both linear and non linear classifiers and are computationally more efficient than perceptrons. Thus, we propose a confidence estimator using SVM's in which the prediction accuracy of previous instructions is used to estimate the confidence of current prediction and decide based on its results whether the prediction is likely to be correct or not.

We have compared perceptron and SVM and the results show better accuracy in prediction. In this paper, we introduce some basic concepts of SVM, kernel function selection and model selection (parameter selection) of SVM. In section 7 we detail the experimental results. Finally, we have some conclusions in section 8.

2. SUPPORT VECTOR MACHINES

SVMs are set of related supervised learning methods used for classification and regression [31]. They belong to a family of generalized linear classification. A special property of SVM is SVM simultaneously minimize the empirical classification error and maximize the geometric margin. So SVM called Maximum Margin Classifiers. SVM is based on the Structural risk Minimization (SRM). SVM map input vector to a higher dimensional space where a maximal separating hyperplane is constructed. Two parallel hyperplanes are constructed on each side of the hyperplane that separate the data. The separating hyperplane is the hyperplane that maximize the distance between the two parallel hyperplanes. An assumption is made that the larger the margin or distance between these parallel hyperplanes the better the generalization error of the classifier will be [31].

We consider data points of the form

$\{(x_1,y_1),(x_2,y_2),(x_3,y_3),(x_4,y_4)\ldots\ldots\ldots,(x_n, y_n)\}$. Where $y_n=1$ / -1 , a constant denoting the class to which that point x_n belongs. n = number of sample. Each x_n is p-dimensional real vector. The scaling is important to guard against variable (attributes) with larger variance. We can view this Training data, by means of the dividing (or seperating) hyperplane, which takes the form of,

$w . x + b = 0$ ……... (1)

Where b is scalar and w is p-dimensional Vector.

The vector w points perpendicular to the separating hyper plane. Adding the offset parameter b allows us to increase the margin. Absent of b, the hyperplane is forced to pass through the origin, restricting the solution. As we are interesting in the maximum margin, we are interested SVM and the parallel hyperplanes. Parallel hyperplanes can be described by equations,

$w.x + b = 1$

$w.x + b = -1$

If the training data are linearly separable, we can select these hyperplanes so that there are no points between them and then try to maximize their distance. By geometry, we find the distance between the hyperplane is 2 / $||w||$, So we want to minimize $||w||$. To excite data points, we need to ensure that for all i either

$w. x_i – b \geq 1$ or

$w. x_i – b \leq -1$

This can be written as

$y_i (w. x_i – b) \geq 1, 1 \leq i \leq n$ ……….. (2)

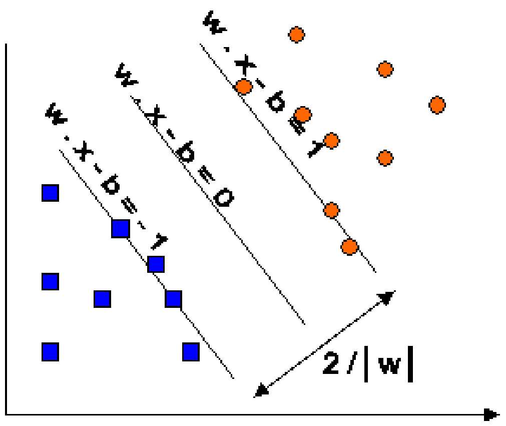


Figure.1 Maximum margin hyperplanes for a SVM trained with samples from two classes[33].

Samples along the hyperplanes are called Support Vectors (SVs). A separating hyperplane with the largest margin defined by M = 2 / $||w||$, that specifies support vectors means training data points closest to it. But which satisfy?

$y_j [w^T . x_j + b]= 1$, $i =1$ ………. (3)

Optimal Canonical Hyperplane (OCH) is a canonical Hyperplane having a maximum margin. For all the data, OCH should satisfy the following constraints.

$y_j [w^T . x_j + b]= 1$; $i =1,2\ldots l$ ……….. (4)

Where l is Number of Training data point. In order to find the optimal separating hyperplane having a maximul margin, A learning machine should minimize $||w||^2$ subject to the inequality constraints

$y_j [w^T . x_j + b] \geq 1$; $i =1,2\ldots\ldots l$

This optimization problem solved by the saddle points of the Lagrange's Function

$$L_P = L_{(w,b,\alpha)} = 1/2||w|| - \sum_{i=1}^{l} \alpha_i (y_i (w^T x_i + b) - 1) \ldots (5)$$

$$= 1/2 \; w^T \; w - \sum_{i=1}^{l} \alpha_i (y_i (w^T x_i + b) - 1)$$

Where α_i is a Lagrange's multiplier .The search for an optimal saddle points (w_0, b_0, α_0) is necessary because Lagrange's must be minimized with respect to w and b and has to be maximized with respect to nonnegative α_i ($\alpha_i \geq 0$). This problem can be solved either in primal form (which is the form of w & b) or in a dual form (which is the form of α_i).Equation 4, 5 are convex and KKT conditions, which are necessary and sufficient conditions for a maximum of equation 4. Partially differentiating equation 5 with respect to saddle points (w_0, b_0, α_0), we get

$\partial L / \partial w_0 = 0$

i .e $$w_0 = \sum_{i=1}^{l} \alpha_i y_i x_i \qquad \ldots\ldots\ldots\ldots(6)$$

And $\partial L / \partial b_0 = 0$

i .e $$\sum_{i=1}^{l} \alpha_i y_i x = 0 \qquad \ldots\ldots\ldots\ldots (7)$$

Substituting equation (6) and (7) in equation (5). We change the primal form into dual form.

$$L_d(\alpha) = \sum \alpha_i \text{ -1/2} \sum_{i=1}^{l} \alpha_i \alpha_j y_i y_j x_i^T x_j \qquad \ldots\ldots (8)$$

In order to find the optimal hyperplane, a dual lagrangian (L_d) has to be maximized with respect to nonnegative α_i (i.e. α_i must be in the nonnegative quadrant) and with respect to the equality constraints as follow

$\alpha_i \geq 0$, $i = 1,2\ldots..l$

$$\sum \alpha_i y_i = 0$$

Note that the dual Lagrangian $L_d(\alpha)$ is expressed in terms of training data and depends only on the scalar products of input patterns (x_i^T x_j).More detailed information on SVM can be found in References, [30]&[31].

2.1 Kernel Selection of SVM

Training vectors x_i are mapped into a higher (may be infinite) dimensional space by the function Φ. Then SVM finds a linear separating hyperplane with the maximal margin in this higher dimension space . C > 0 is the penality parameter of the error term.

Furthermore, $K(x_i , x_j) \equiv \Phi(x_i)^T \Phi(x_j)$ is called the kernel function[31]. There are many kernel functions in SVM, so, how select a good kernel function according to the problem is also a research topic. However, there are some popular kernel functions [31] & [32], for general purposes.

- Linear kernel: $K(x_i , x_j) = x_i^T x_j$.
- Polynomial kernel:

  $K(x_i , x_j) = (\gamma x_i^T x_j.+ r)^d$, $\gamma > 0$
- RBF kernel :

  $K(x_i , x_j) = \exp(-\gamma \|x_i - x_j\|^2)$, $\gamma > 0$
- Sigmoid kernel:

  $K(x_i , x_j) = \tanh(\gamma x_i^T x_j + r)$

Here, γ, r and d are kernel parameters. RBF is most effective and important, because of the following reasons[30]:

1.The RBF kernel nonlinearly maps samples into a higher dimensional space unlike to linear kernel.

2.The RBF kernel has less hyperparameters than the polynomial kernel.

3.The RBF kernel has less numerical difficulties.

2.2 Model/ Parameter Selection of SVM

Model selection is also an important concern in SVM. Recently, SVM have shown good performance in data classification. Its success depends on the tuning of several parameters which affect the generalization error. We often call this parameter tuning procedure as the model selection. If you use the linear SVM, you only need to tune the cost parameter C. Unfortunately, linear SVM are often applied to linearly separable problem. Many problems are non-linearly separable. For example, Satellite data and Shuttle data are not linearly separable [30], and so as value prediction data. Therefore, we often apply nonlinear kernel to solve classification problems, so we need to select the cost parameter (C) and kernel parameters (γ, d).

Grid-search method is used generally in cross validation to select the best parameter set, but we have set it manually. We apply this parameter set to the training dataset to classify the testing dataset to obtain the generalization accuracy.

3. PROBLEM STATEMENT

Sometimes linear inseparability may arise when a correct prediction on a past instruction causes the current instruction to predict incorrectly [25]. Small improvements in accuracy can have a large impact on performance; decreasing the misprediction rate from, say, 5% to 4% can decrease the execution time of a typical program by as much as 14% [28]. Here, we propose a novel global confidence estimation scheme using SVM which can achieve a better accuracy.

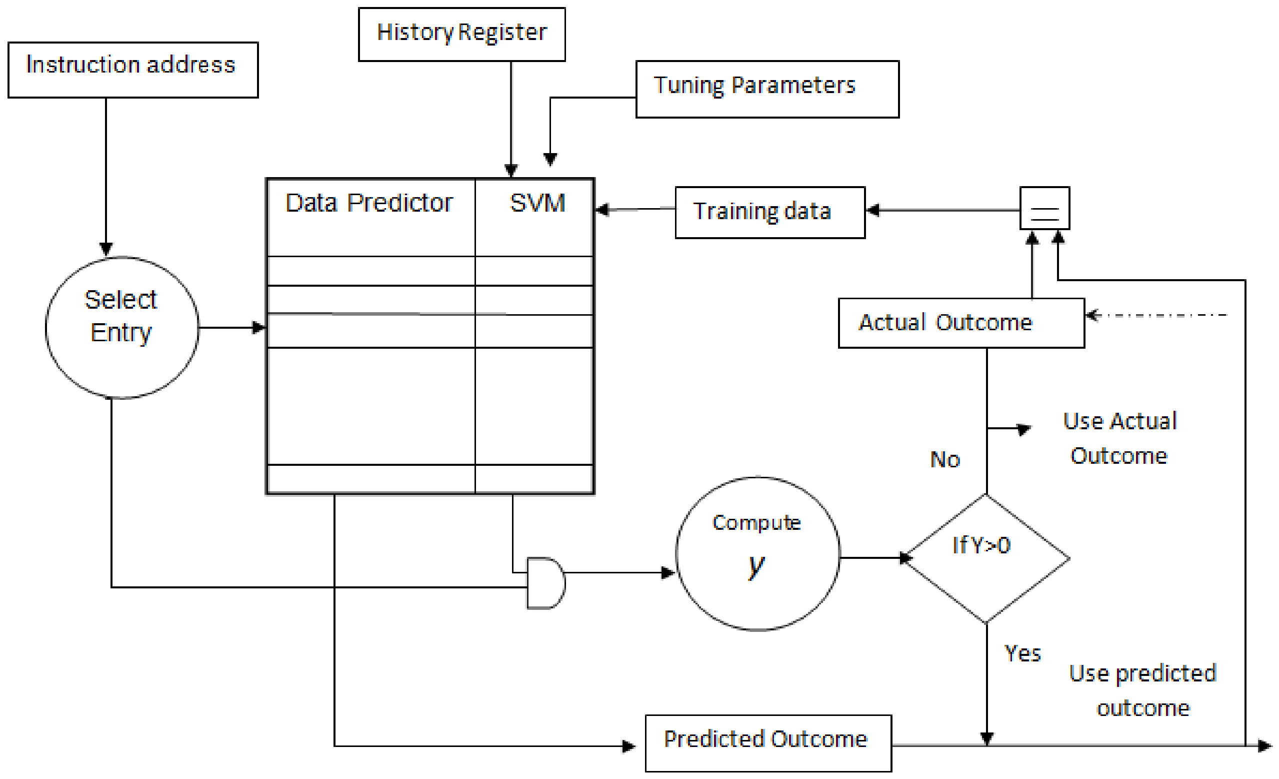


Figure 6: Block Diagram of the Prediction Architecture with our SVM Confidence estimator

4. EARLY WORK ON CONFIDENCE ESTIMATION

Lipasti, Wilkerson, and Shen introduced the earliest confidence estimator used in data value prediction in [4]. It is comprised of a 2-bit saturating up-down counter that chooses between three prediction states: 0 or 1 = "don't predict", 2 = "predict" and 3= "constant" (highly predictable). If a given instruction makes a correct prediction, the counter is incremented; otherwise, it is decremented. Regardless of whether the instruction predicts correctly or incorrectly, the counter is not allowed to exceed 3 or go under 0. This approach is used in many other data value predictors [8, 12, and 13].

The use of the perceptron as a predictor was first suggested by Vintan et al [14]. The perceptron is one of the simplest models of a neuron and was developed by Rosenblatt [**1**] to help study brain function. The simplest perceptron is a neuron that connects several weighted inputs to a single output. Classically, the output y of the perceptron is the dot product of the weights $w = (w_1, \cdots, w_n)$ and the inputs $x = (x_1, \cdots, x_n)$, with the bias input *b*, which can be thought of as a weight w_o with constant input $x_0 = 1$.

$$y = b + \langle x, w \rangle = w_0 + \sum_{i=1}^{n} x_i w_i \qquad \ldots\ldots\ldots\ (9)$$

The output y is used to classify a new pattern x. The perceptrons performance in classifying is improved by incrementally adjusting its weights during training using the perceptron learning algorithm (Figure 2a).

$if \quad t_y \le 0\, or \quad \lvert y \rvert \le \phi \quad the$ $for\, all \quad w_i$ $w_i \leftarrow w_i + t x_i$	$if \quad t_y \le 1\, or \quad \lvert y \rvert \le \theta \quad then$ $for\, all \quad x_i \in SV$ $update \quad \alpha_i \quad to$ $\min imize(w, w) \quad with \quad t_y \ge 1$
(a)	(b)

Figure 2: The Update Algorithm for the Perceptron (a) and SVM (b). θ is learning threshold parameter. *t* ϵ {±1} is the classification of the vector *x*. The sign of *y* is the classification hypothesis for *x*, given by Equations 9 and 4 for the perceptron and SVM respectively.

Michael Black and Manoj Franklin, in their paper named "Perceptron-based Confidence Estimation for Value Prediction" [24, 26], presented a perceptron-based confidence estimator for data value prediction that makes use of correlations between the predictability of different instructions.

Their confidence estimator uses the predictability information to raise the accuracy of data value prediction. Simulation results show that the perceptron confidence estimator generally offers significant improvement over the conventional up-down counter confidence estimator.

To uncover predictability dependencies, he used a perceptron based confidence estimator. A perceptron is a simple neural network consisting of an adder, a threshold function, and a set of weights implemented by saturating signed integer counters. The perceptron uses these components to guess an output based on a series of inputs.

Given a set of input bits, it computes the dot product of the inputs and the weights, and compares the result to a threshold value, typically 0 (an extra weight is hardwired to an input of 1 to provide a bias). If the result is greater than 0, the perceptron returns "True"; otherwise it returns "False."

The perceptron determines the values of its weights by learning. When a correct value is found, the perceptron is "trained." i.e., an error value is computed by the difference between the training value and the perceptron output. This error value is multiplied by each input bit and is added to the corresponding weight. In this manner, each weight is adjusted so that the desired output is realized from the particular input combination. When applied to confidence estimation, each weight value determines the relationship between a particular past instruction and the current instruction. If a weight value is positive and large, the past instruction's predictability tends to have a direct effect on the current instruction's predictability [17]. i.e. to say, the current instruction's data value predictor tends to predict correctly only when the past instruction's data value predictor predicted correctly. If the weight value is negative and large, the past instruction's predictability effect is inverse; the current instruction's data predictor tends to predict correctly only when the past instruction mispredicted. If the weight value's magnitude is small, the past instruction has been found to have little effect on the current instruction.

5. WHY SUPPORT VECTOR MACHINES?

Today, support vector machines and along with other learning based-kernel algorithms show better results than artificial neural networks and other intelligent or statistical models, on the most popular benchmark problems [23].
A. Zanaty [29], introduced a new kernel function called Gaussian Radial Basis Polynomials Function (GRPF) that combines both Gaussian Radial Basis Function (RBF) and Polynomial (POLY) kernels for improving the accuracy of the Support Vector Machines (SVMs) classification for both linear and non-linear data sets.
Osowski, Siwek, and Markiewic [30] solved the two spiral problem using both: MLP network trained by using Levenberg-Marquardt algorithm and SVM with radial basis function trained by applying Platt method. The training time of MLP was approximately 10 times longer than SVM. According to them, SVM is unbeatable in classification mode, while in regression MLP possesses better generalization ability.As confidence estimation is also a classification problem; here we have used SVM for the purpose.

5.1 Linear separability and inseparability

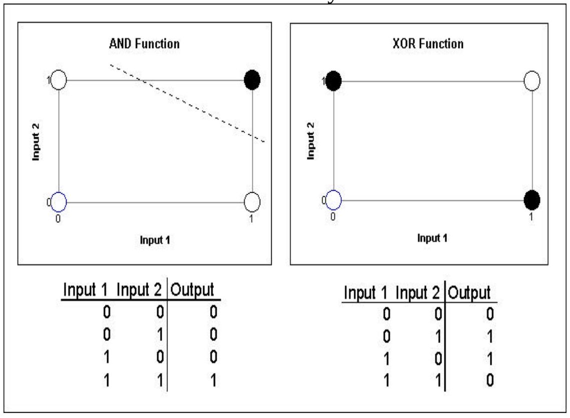
A limitation of perceptrons is that they are only capable of learning *linearly separable* functions. Minsky and Papert **[2]** show that perceptrons cannot learn *linearly-inseparable* functions, like XOR (fig 3), with 100% accuracy. Minsky's work originally claimed that this was the case for all neural networks, but it was later discovered that linearly inseparable functions can be learned in larger neural networks using hidden layers and more advanced training mechanisms. However, this is still a handicap for the simple single layered perceptron. Linear separability is classically pictured geometrically in an *n*-dimensional space, where *n* is the number of inputs. All the possible outputs are placed in the space. If the space can be divided by a plane so that all positive outputs are on one side of the plane and all negative outputs are on the other side, the function is linearly separable [3]. If no plane can be drawn, the function cannot be learned by a perceptron.

Imagine the set of all possible inputs to a perceptron as an n - dimensional space. The solution to the Equation

$$w_0 + \sum_{1=1}^{n} x_i\, w_i = 0$$

is a hyperplane (e.g. a line, if n=2) dividing the space into the set of inputs for which the perceptron will respond *false* and the set for which the perceptron will respond *true* [8]. A Boolean function over variables $x_{1\ldots n}$ is *linearly separable* if and only if there exist values for $w_{0..n}$ such that all of the *true* instances can be separated from all of the *false* instances by that hyperplane. Since the output of a perceptron is decided by the above equation, only linearly separable functions can be learned perfectly by perceptrons. For instance, a perceptron can learn the logical AND of two inputs, but not the exclusive-OR, since there is no line separating *true* instances of the exclusive-OR function from *false* ones on the Boolean plane.

Linear inseparability arises if a correct prediction on a past instruction causes the current instruction to predict correctly sometimes and incorrectly at other times.



Input 1	Input 2	Output
0	0	0
0	1	0
1	0	0
1	1	1

Input 1	Input 2	Output
0	0	0
0	1	1
1	0	1
1	1	0

Fig 3: Inseparability and Separability of XOR and AND function

Because a correct prediction on a past instruction rarely causes the current instruction to predict incorrectly [24]. But sometimes this can happen[25].
In a perceptron, the effect of an input on the output is determined by its weight. As stated before, a positive weight means that the output varies directly with the input, while a negative weight causes the output to vary inversely with the input. Based on its weight, a 1 at a particular input can make the total output more positive or more negative. However, a 1 at a particular input cannot make the total output more positive sometimes and more negative at other times.

Functions tend not to be linearly separable if one input's effect on the output relies on another input's effect which can happen in value prediction. And as support vector machines belong to a family of generalized linear classifiers and can be interpreted as an extension of the perceptron, They are both linear and non-linear classifiers.

6. EXPERIMENTAL RESULTS

6.1 Experimentation Methodology

Our measurements were performed on the PISA instruction set architecture. The data value predictor considers every instruction that has a single destination register. Predictions are made after each instruction executes and the actual instruction output is immediately used to train the predictor. Our study is performed using three types of predictors: Last-Value, Stride, and Context. The Last-Value predictor simply returns the value that an instruction produced the last time it was executed. The Stride predictor computes the difference between the last two results of an instruction, and adds it to the most recent result to predict a value. The Context predictor uses the most recent four data values produced by an instruction to index a pattern table of up-down counters [8]. The counters choose one of the four data values to be the prediction. Each of the three predictors includes a table indexed by the instruction address. We use 16k table entries and a direct-map organization for the table. We have also used LIBSVM 3.17 in windows command prompt to find out the accuracy rate of prediction.

We have used our independent data sets for finding out the misprediction rate in support vector machine as well as perceptron. The results in figure-4, shows that SVM has performed better than Perceptron.

As no. of data points increases, SVM achieves upto 3% less misprediction than Perceptron. That means more the amount of instructions executed, higher the accuracy.
Our predictor uses *n* SVMs, The *i*th SVM, which we informally call SVM[i], is distributed between two tables in hardware:

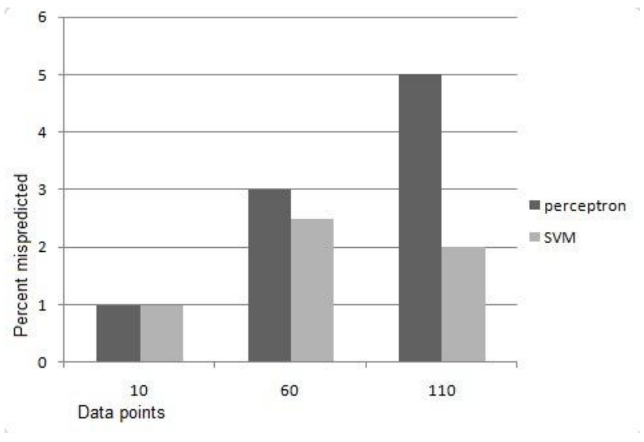


Fig 4: Percentage of misprediction in perceptron and SVM.

The weight table $(\mathrm{SVM})_{\alpha}$ and the table of support vectors$(\mathrm{SVM})_{SV}$ For each SVM, we set a strict maximum of *m* on the number of support vectors that can be collected during training.
n- The number of SVMs used
m- The maximum number of support vectors each SVM may accumulate
Hash (·) -The hash function used to map each instruction address to one of the *n* SVMs
Ker (u, v) -The kernel function
$\mathrm{Algo}_{\mathrm{SVM}}(\theta, \ldots) -$ The algorithm to train each SVM, a function of the learning error θ
GHT- Global History Table

6.2 Proposed Algorithm

The data for training are vectors x_i along with their categories y_i. For some dimension d, the $x_i \in R^d$, and the $y_i = \pm 1$. The equation of a hyperplane is $<w,x>+\alpha_0 = 0$
Assuming w =weights and x_i=address of instruction set.

A. Initialize tolerance ϵ for Support Vector Detection and parameters for kernel function.
B. Initialize and set up Hessian matrix H.
C. Initialize Parameters for the Optimization problem.
D. Set up the equality constraints.
E. Solve the Quadratic programming problem, i.e,

$$\min \frac{1}{2} \parallel w \parallel^2 st \;\; y_i(x_i.w + \alpha_0) - 1 \geq 0 \; for\ all\ i$$

F. Find the unbounded support vectors *m* and store in SVM_{SV}
When an instruction is encountered:
1. The instruction address is hashed to index *i*, to access SVM[*i*].
2. SVM[i] and its weights are put into a register of weights, $\alpha = (\alpha_{0.}\,.\,.\,\alpha_m)$. In parallel, SVM[$i$]'s *m* support vectors are fetched from SVM_{SV} and brought into vectors SV_i……. SV_m
3. The dot products k_i= Ker (GHT,SV_i) for $i\epsilon\{1\ .\ .\ .m\}$ are calculated.
4. $\alpha_i * k_i$ for $i\epsilon\{1\ .\ .\ .m\}$ is calculated.
5. The results of the multiplications and the bias, $y = a_0 + \sum_{i=1}^{m} a_i\, k_i$ are summed.
6. The prediction of SVM is the sign of y. The predictor stores the value *of i*, for later training of SVM[i].
7. When the actual outcome *t* of the instruction is known, shift the values in GHT, add *t*, and train SVM[i].

6.3 Confidence estimator organization

The prediction system proposed by us works as follows: The instruction address is used to select a table entry. This table entry consists of a value predictor, which predicts a value and the SVM takes the GHR as its input and uses its weights to determine whether its output is "predict" or "don't predict". If the output is "predict", the value predictor outcome is used as a prediction; otherwise the prediction is not used. Regardless of the SVMs outcome, when the actual result of the instruction is known, it is compared against the prediction of data predictor.
If they match, a 1 (predicted correctly) is shifted into the GHR at the instruction's completion stage. Otherwise, a 0 (predicted incorrectly) is shifted into the GPH. The difference between the actual result and the prediction is then used to train the SVM.
Figure 6 shows our block diagram and figure 5 shows that SVM classifies the two data sets with a very low misclassification rate. Thus, in case of value prediction, the data points position are the predictability of the instructions and it is SVMs job to find out which instruction should lie in which class. If the current instruction lies in class -1 that means some of the last predictions were incorrect.

So, the prediction should not be used. SVM gradually learns to classify better. However, linear inseparability is rare in these cases, but still it happens. So a very small improvement in accuracy leads to a better performance.

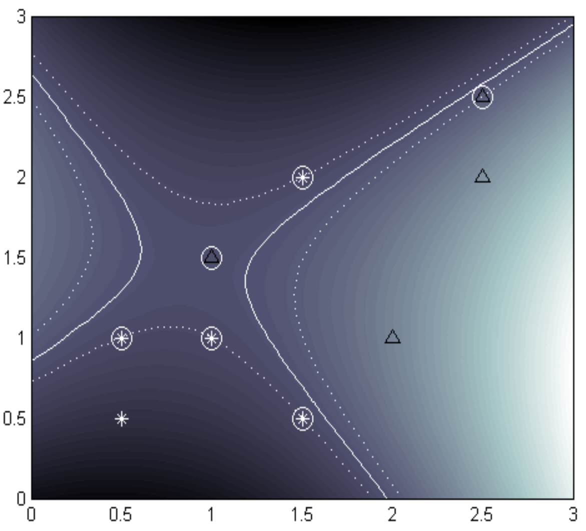


Fig 5: Classification by SVM

7. CONCLUSION

Based on the previous work and our simulation results it will be safe to assume that SVM's work better than perceptrons. In case of branch prediction and confidence estimation, both produce same two classes, i.e. "taken", "not taken" [27, 28] or "predict", "Don't predict". So, we propose a SVM based confidence estimator that estimates the confidence value for prediction, where prediction will be done by data value predictors. Perceptron and SVM are compared and the prediction error rates are calculated. Support Vector Machines are used to identify which past instructions affect the accuracy of a prediction and to decide based on their results whether the prediction is likely to be correct or not . The confidence estimator raises the accuracy of value prediction.

In future we would like to employ the algorithms more efficiently, because there are many issues like appropriate parameter and kernel selection. Using appropriate data set is also necessary to get SVMs optimum results. Thus we would be more specific in that when it comes to reducing the misprediction rate. As perceptrons and SVMs can also be used for value prediction, we would like to use them as well and also employ a confidence estimator using SVM's along with it to gain more accuracy.

8. REFERENCES

[1] F. Rosenblatt, Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms, Spartan, 1962.

[2] M.L. Minsky and S.A. Papert, Perceptrons, MIT Press, 1969.

[3] S. Russell and P. Norvig. "Artificial Intelligence: A Modern Approach." Prentice-Hall, Inc., Upper Saddle River, NJ, 1995, pp. 563-593.

[4] M. H. Lipasti, C. B. Wilderson, and J. P. Shen, "Value locality and load value prediction," in Proceedings of the 7th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOSVII), October 1996.

[5] M. H. Lipasti and J. P. Shen. "Exceeding the Dataflow Limit via Value Prediction." Proceedings of the 29th Annual ACM/IEEE International Symposium on Microarchitecture, Dec., 1996

[6] K. Wang and M. Franklin. "Highly Accurate Data Value Prediction using Hybrid Predictors." Proc 30th Intl Symp on Microarchitecture, Dec. 1997.

[7] F. Gabbay and A. Mendelson. "Can Program Profiling support Value Prediction?" Proc 30th Intl Symp on Microarchitecture, Dec. 1997.

[8] K. Wang and M. Franklin. "Highly Accurate Data Value Prediction using Hybrid Predictors." Proc 30th Intl Symp on Microarchitecture, Dec. 1997

[9] Y. Sazeides, J. E. Smith. "Implementations of Context Based Value Predictors." Technical Report ECE-97-8, University of Wisconsin-Madison, Dec. 1997.

[10] Y. Sazeides, J. E. Smith. "Implementations of Context Based Value Predictors." Technical Report ECE-97-8, University of Wisconsin-Madison, Dec. 1997.

[11] B. Calder, G. Reinman, and D. Tullsen. "Selective value prediction." Technical Report UCSD-CS98- 597, University of California, San Diego, Sep. 1998.

[12] M. Burtscher, B. G. Zorn. "Profile-Supported Confidence Estimation for Load-Value Prediction." Technical Report CU-CS-872-98, University of Colorado at Boulder, Oct. 1998.

[13] M. Burtscher and B. G. Zorn. "Prediction Outcome History-based Confidence Estimation for Load Value Prediction." Journal of Instruction Level Parallelism, May 1999.

[14] Lucian N. Vintan and Mihaela Iridon, "Towards a high performance neural branch predictor," in Proceedings of the 1999 International Joint Conference on Neural Networks. July 1999, vol. 2, pp. 868–873, IEEE Computer Society

[15] Bernhard Sch¨olkopf, Chris Burges, and Alex Smola, Eds., Advances in Kernel Methods - Support Vector Learning, MIT Press, 1999.

[16] N. Cristianini and J. Shawe-Taylor, An Introduction to Support Vector Machines, Cambridge University Press, 2000

[17] R. Thomas and M. Franklin, "Characterization of Data Value Unpredictability to Improve Predictability." Proc Intl Conf on High Performance Computing, 2001.

[18] R. Thomas and M. Franklin. "Using Dataflow Based Context for Accurate Value Prediction," Proc Intl Conf on Parallel Architectures and Compilation Techniques, 2001

[19] Bernhard Sch¨olkopf, "Svm and kernel methods," December2001,http://www.kernelmachines.org/papers/tutorial-nips.ps.gz.

[20] D. Jimenez and C. Lin. "Composite Confidence Estimators for Enhanced Speculation Control." Technical Report TR2002-14, Dept. of Computer Sciences, University of Texas at Austin, 2002.

[21] H. Zhou, J. Flanagan, and T. Conte. "Detecting Global Stride Locality in Value Streams." Proc Intl Symp on Computer Architecture, 2003.

[22] M. Black. "Perceptron-based Global Confidence Estimation for Value Prediction." M.S. Thesis, Department of Electrical and Computer Engineering, University of Maryland, June 2003.

[23] Meyer D, Leisch F, Hornik K. The support vector machinesunder test. Neurocomputing 2003 ;55:169–86.

[24] M. Black and M. Franklin. "Perceptron-based Confidence Estimation for Value Prediction." International Conference on Intelligent Sensors and Information Processing, Jan. 2004.

[25] M. Black and M. Franklin. "Applying Perceptrons to Computer Architecture." Proceedings of the Second International Conference on Intelligent Sensors and Information Processing, Jan. 2005.

[26] M. Black and M. Franklin. "Neural Confidence Estimation for More Accurate Value Prediction." International Conference on High Performance Computing, 2005.

[27] culpepper and gondree, "SVMs for Improved Branch Prediction" ECS201A Computer Architecture[2008].

[28] Asis Kumar Tripathy et al, International Journal of Advanced Research in Computer Science, 2 (1), Jan. – Feb, 2011,310-313

[29] E.A. Zanaty "Support Vector Machines (SVMs) versus Multilayer Perception (MLP) in data classification" Egyptian Informatics Journal (September 2012) 13, 177–183)

[30] Boser, B. E., I. Guyon, and V. Vapnik (1992). A training algorithm for optimal margin classifiers . In Proceedings of the Fifth Annual Workshop on Computational Learning Theory, pages. 144 -152. ACM Press 1992.

[31] V. Vapnik. The Nature of Statistical Learning Theory. NY: Springer-Verlag. 1995.

[32] Chih-Wei Hsu, Chih-Chung Chang, and Chih- Jen Lin. "A Practical Guide to Support VectorClassification" . Deptt of Computer Sci. National Taiwan Uni, Taipei, 106, Taiwan http://www.csie.ntu.edu.tw/~cjlin 2007.

[33] Durgesh Srivastava and Lekha Bhambhu, "Data classification using support vector machine" Journal of Theoretical and Applied Information Technology, Department of CSE/IT, BRCM CET, Bhiwani, Haryana, India 2005