Random Kernel Perceptron on ATTiny2313 Microcontroller

Nemanja Djuric, Slobodan Vucetic

Department of Computer and Information Sciences
Temple University

SensorKDD, July 25th, 2010, Washington DC

Kernel Perceptron

- Predictor

$$f(\mathbf{x}) = sign(\sum_{i=1}^{N} \alpha_i K(\mathbf{x}, \mathbf{x}_i))$$

- Costs
  - $O(N)$ space
  - $O(N)$ update time
  - $O(N^2)$ total training time

Inputs : *data sequence* $((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N))$
Output : *trained Kernel Perceptron* $f(\mathbf{x})$

$f(\mathbf{x}) = 0$ *at time* $t = 0$
while (*training set not empty*)
{
if $(y_i \cdot f(\mathbf{x}_i) > 0)$
$\alpha_i = 0$
else
$\alpha_i = y_i$

$f(\mathbf{x}) \leftarrow f(\mathbf{x}) + \alpha_i \cdot K(\mathbf{x}_i, \mathbf{x})$
}

Random Budget Kernel Perceptron

- Idea
  - assign support vector budget *T*
  - when budget is exceeded, remove a random SV
  - resulting predictor

$$f(\mathbf{x}) = sign(\sum_{i=1}^{T} \alpha_i K(\mathbf{x}, \mathbf{x}_i))$$

- Costs
  - O(1) space
  - *O*(1) update time
  - *O*(*N*) training time

Inputs : *data sequence* $((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N))$, *budget* T
Output : *support vector set* $SV = \{SV_i,\ i = 1 \ldots I\}$

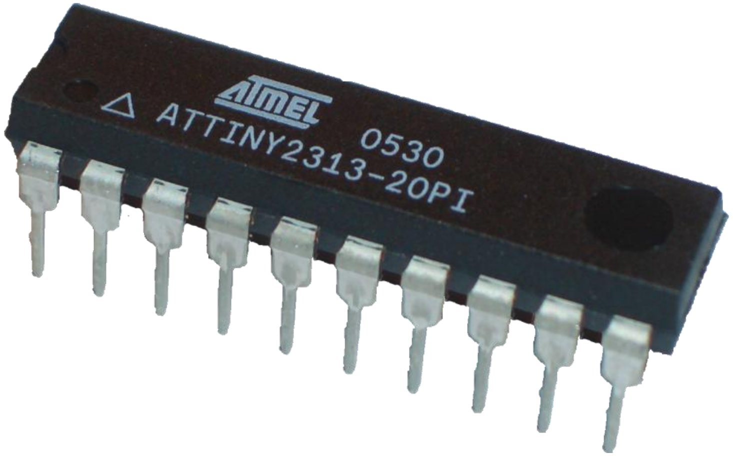
$I \leftarrow 0;\ i \leftarrow 1$
$SV = \emptyset$
for $i = 1 : N$
{
    if $((y_i \cdot \sum_{j=1}^{I} y_j \cdot K(\mathbf{x}_i, \mathbf{x}_j)) \le 0)$
    {
        if $(I == T)$
            $new = random(I)$
        else
        {
            $I \leftarrow I + 1$
            $new \leftarrow I$
        }
        $SV_{new} = (x_i, y_i)$
    }
}

Motivation

- Random Kernel Perceptron
  - online algorithm
  - low cost
  - easy to implement
  - can solve nonlinear problems
  - accurate
- It still **CANNOT** be implemented on the simplest computers
  - it uses floating-point operations
  - model size easily exceeds available memory
- Goal: Implement Kernel Perceptron on microcontrollers
- Applications
  - sensor networks
  - low-cost online data mining
  - resource-constrained environments

Microcontroller

- ATTiny2313
  - one of the most primitive processors
  - very cheap (< $1)
- Characteristics
  - 128 bytes to store:
    - Kernel Perceptron
    - working variables
  - 2 Kbytes to store program
  - 4 MHz processor speed
  - fixed-point arithmetic (**integers**)



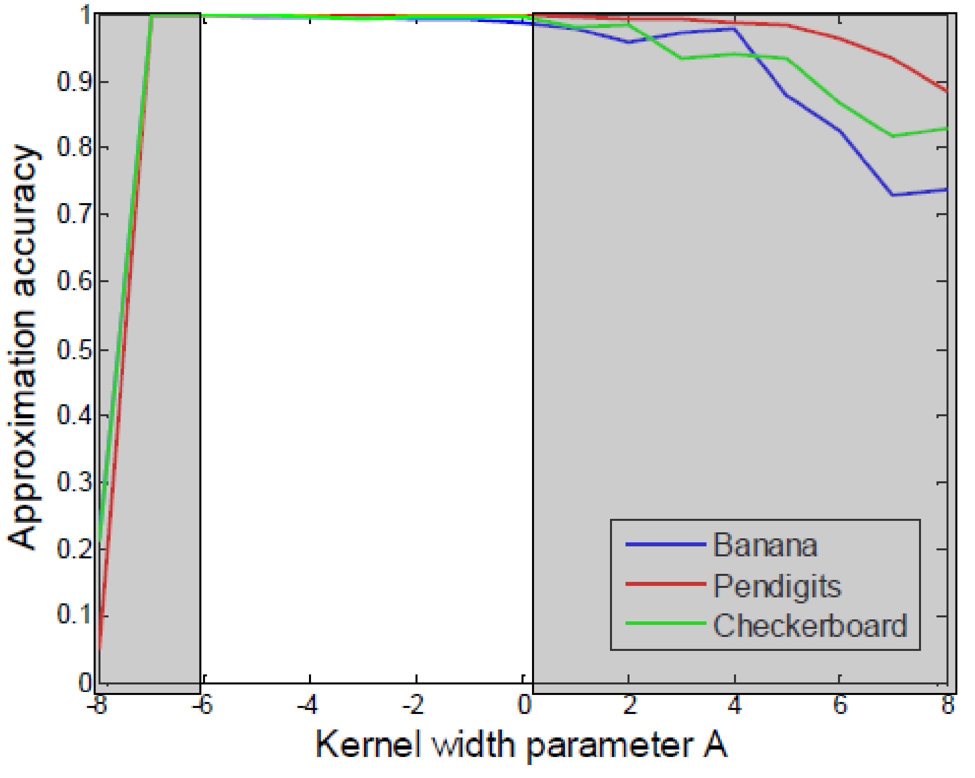
Some details

- Use Gaussian kernel: $K(\mathbf{x},\mathbf{x}_i) = \exp(\frac{\|\mathbf{x}-\mathbf{x}_i\|^2}{2^A})$

- Resource-saving strategies
  - Quantization of attributes using *b* bits
    - trade-off between #SV and #bits
    - quantization loss
  - Approximation of kernel function using only integers and integer calculations
    - we devised an iterative procedure that uses look-up table
    - approximation loss

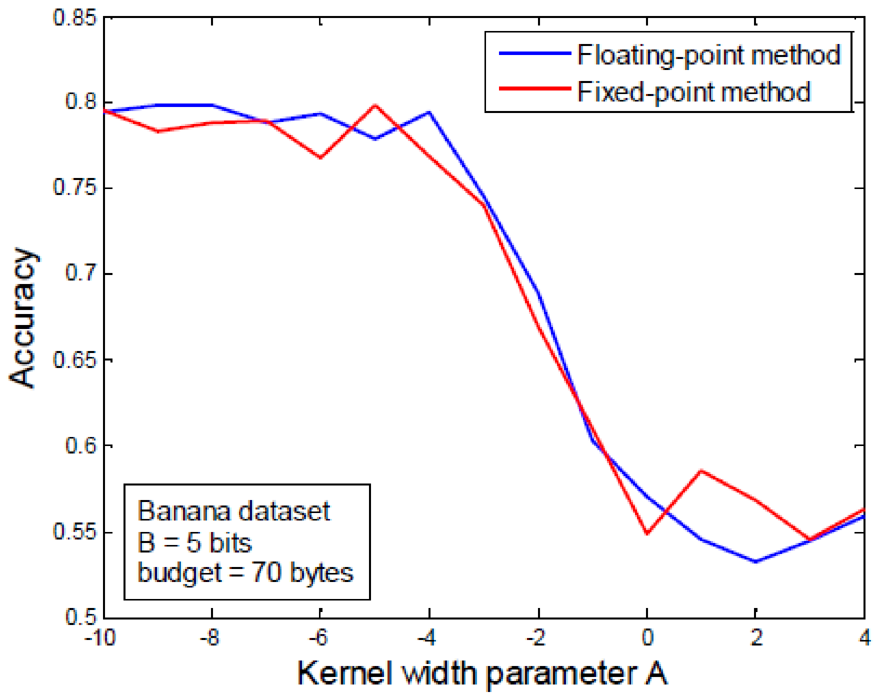
Results

- Fixed-point vs. floating-point method
- Approximation accuracy (kernel width = 2^A)

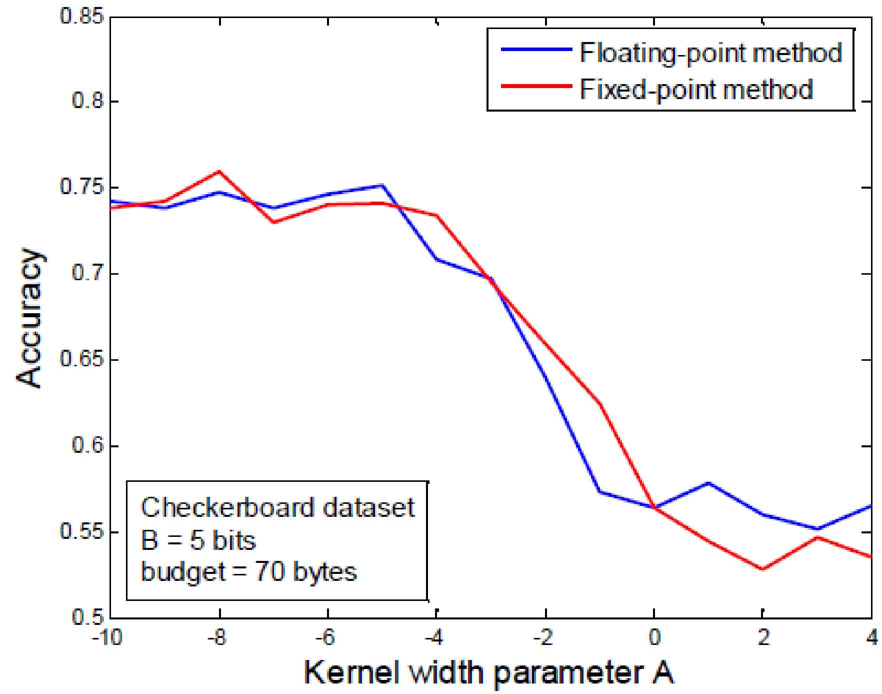


Results

- Accuracy on benchmark datasets



Banana dataset



Checkerboard dataset

Results

- Implementation on microcontroller
- Double-precision Kernel Perceptron: 89.2% accuracy
- Much less memory, faster execution time

Banana dataset	***4 bits***		***6 bits***	
Method	***Fixed***	***Float***	***Fixed***	***Float***
Data [B] (max 128B)	128	379	128	379
Program [B] (max 2048B)	1720	6012	1720	6012
***Time* [*ms*]**	1985	7505	1883	7610
***Accuracy* [%]**	**81.00**	**81.08**	**79.36**	**79.60**
# of SVs	62		43	

Data memory	***After quantization***		***Before (Double-precision)***	
	Model	***Working variables***	***Model***	***Working variables***
Memory size	***70B***	***58B***	***> 1KB***	***> 500B***

Conclusions

- Implemented Kernel Perceptron on ATTiny2313 microcontroller
- Fixed-point calculations of prediction
  - key for implementation
  - low data and program memory
  - speeds up calculations
  - only slightly decreases accuracy
- Our results
  - useful in establishing lower bounds on necessary computational resources for online learning
  - open doors for novel application of data mining, such as data mining from sensor data

Thank you!

- More details in the paper
- Questions? E-mail to nemanja.djuric@temple.edu