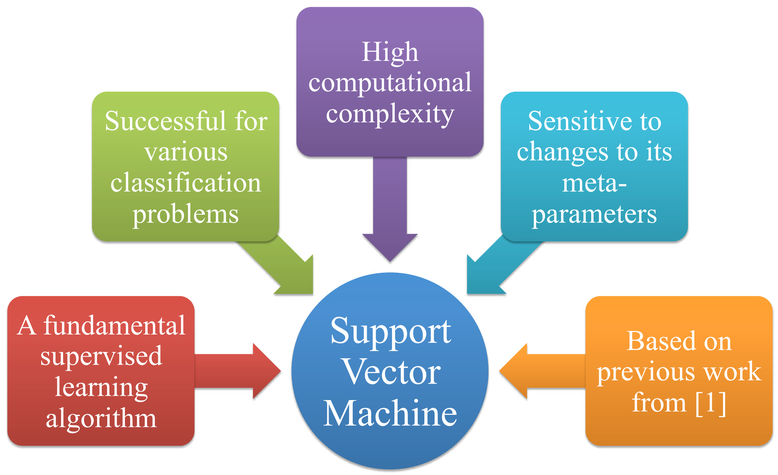
Handwritten Character Classification using GPUs and OpenACC

Valeriu Codreanu (SURFsara, The Netherlands)
Olarik Surinta, Marco A. Wiering (University of Groningen, The Netherlands)

Introduction



OpenACC Directives for Accelerators

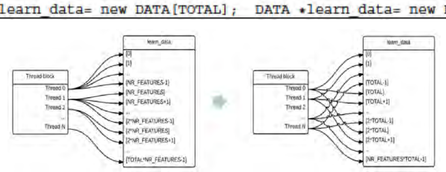
OpenACC-SVM

- Improve classification accuracy through extensive parameter search
- Maintain scalability and code readability by using OpenACC on multi-GPU
- Show the benefits of automatic parallelization for GPU computing
- Explore various feature types and problem sizes

- Started from a single-thread implementation of the SVM.
- The SVM is written in C++, augmented with OpenACC and OpenMP pragmas.
- 2 main code optimizations were performed:
  - **(i)** Arrays of structures are transformed to structures of arrays.
  - **(ii)** Minor data rearrangement to allow for memory coalescence.
- Particle Swarm Optimization (PSO) is implemented in Python:
  - PSO computes the next sets of parameters and launches SVM processes when GPU devices become available.

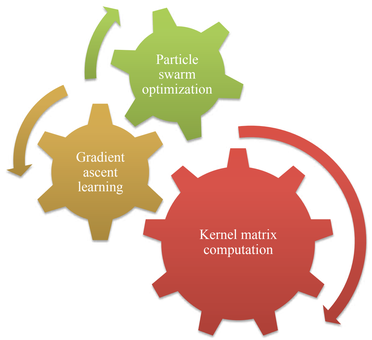
```
struct DATA{                          struct DATA{
  int class;                            int class[TOTAL];
  float input[NR_FEATURES];             float input[TOTAL*NR_FEATURES];
}                                     }

DATA *learn_data= new DATA[TOTAL];   DATA *learn_data= new DATA;
```

(i)



(ii)

Methods




```
#pragma acc data
pcopyin(learn_data[0:leer_tot*feature_num])
pcopy(kernel_train_pgi[0:leer_tot*leer_tot])
{
#pragma acc parallel loop collapse(2)
for (episode = 0; episode < leer_tot; ++episode) {
  for (ad = episode; ad < leer_tot; ++ad) {
    float diff;
    float sqdiff = 0.0f;
    float output;
    #pragma acc loop                                  (iii)
    for (feat = 0; feat < feature_num; ++feat) {
      diff = learn_data[feat*leer_tot + episode] –
             learn_data[feat*leer_tot + ad];
      sqdiff += diff*diff;
    }
    output = expf(-sqdiff / sigma);
    kernel_train_pgi[ad*leer_tot+episode] = output;
    kernel_train_pgi[episode*leer_tot+ad] = output;
  }
 }
}
```

Three functions were augmented with OpenACC and OpenMP directives:

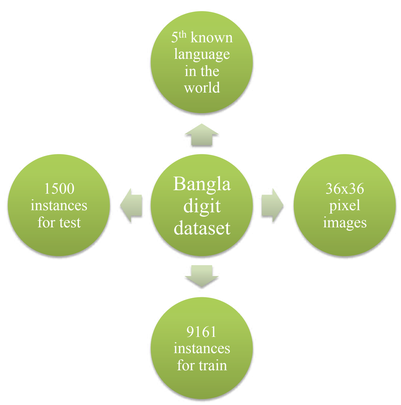
- Kernel matrix computation (KMC). **(iii)**
- Gradient-ascent learning (GAL).
- Bias computation (BC).

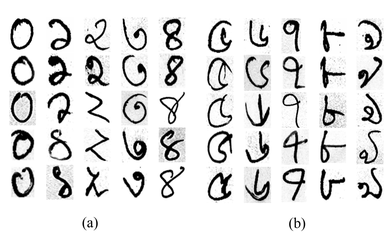
2-stage parallelization scheme:

- SVM instance is parallelized for a single GPU.
- PSO distributes SVM instances to multiple GPUs
  - SVM instances are completely independent.

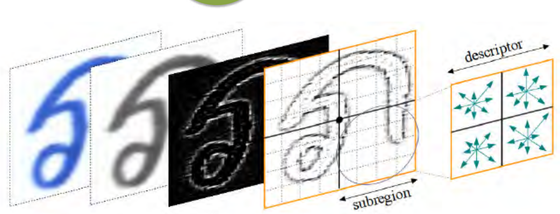
Dataset and feature description

We choose the handwritten character classification problem for Bangla (Bengali) digits to evaluate our approach

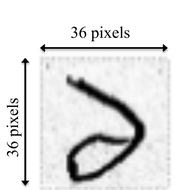




Handwritten Bangla digit samples (a) from 0 to 4 and (b) from 5 to 9



128-dimensional SIFT-based feature (Bangla-SIFT)



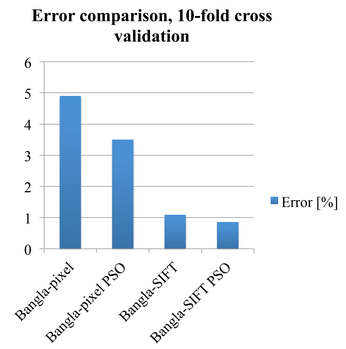
1296-dimensional pixel-based feature (Bangla-pixel)

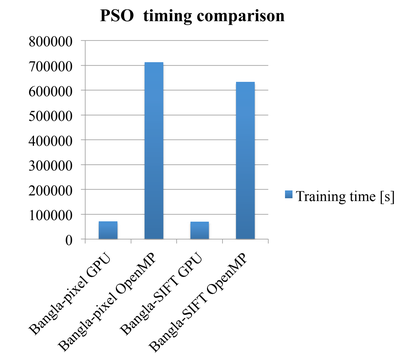
Experimental results

Test setup

CPU: 2x E5-2695 v2 @ 2.40GHz (24 cores)	GPU: 2x Tesla K40m

We perform PSO runs with 10,000 particles to find the best meta-parameters for both the pixel and SIFT-based feature types. Both GPU devices are used independently.





Feature size	Nr. examples (train/test)	KMC time [s]	GAL time [s]
128	9161/1500	1.01	7.03
1296	9161/1500	2.29	8.15
1296	5497/1500	0.82	3.26
12800	9161/1500	17.18	10.23

→ Good performance scaling both in terms of feature size and number of examples

The difference in accuracy is highly statistically significant, with more than 20% classification error reduction in both cases.

Conclusion

- Extensive parameter tuning allows for **significantly** better SVM classification accuracy results.
- Performs **significantly** faster than the OpenMP version, and is also comparable to GPU-based SVM libraries (see [1]).
- The 2-stage parallelization scheme provides good scalability, both in terms of problem size and number of GPU devices.
- Allowed for the best Bangla result so far, 99.15% accuracy!
- It takes 19 hours to tune the parameters for the Bangla-pixel dataset on the dual-K40s and 8.1 days on the 24-core Xeon!

- [1] **V. Codreanu**, B. Droge, D. Williams, B. Yasar, P. Yang, B.Q. Liu, F. Dong, J. Roerdink, M. Wiering, *"Evaluating automatically parallelized versions of the Support Vector Machine"*, Concurrency and Computation: Practice & Experience, 2014. doi: 10.1002/cpe.3413