Particle Swarm Optimization

Applications in Parameterization of Classifiers

James Blondin

jblondin@gmail.com

Armstrong Atlantic State University

Outline

- Introduction
- Canonical PSO Algorithm
- PSO Algorithm Example
- Classifier Optimization
- Conclusion

Particle Swarm Optimization

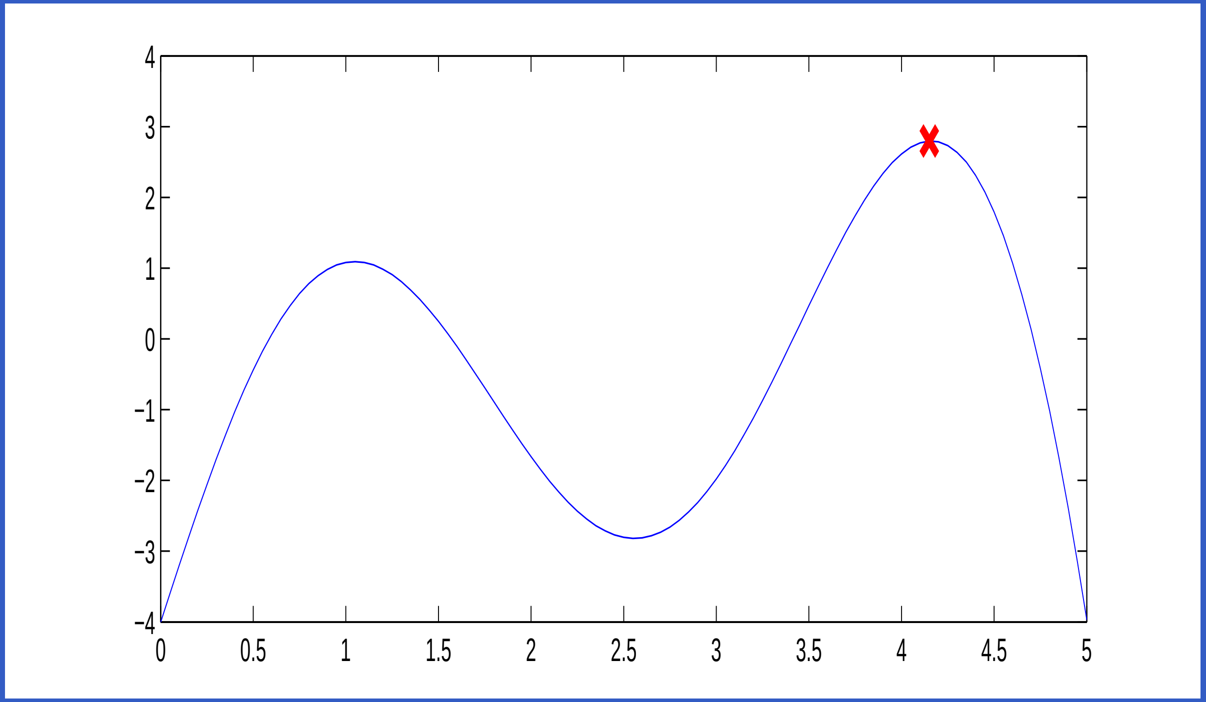
Particle Swarm Optimization (PSO) is a

- swarm-intelligence-based
- approximate
- nondeterministic

optimization technique.

Optimization Techniques

Optimization techniques find the parameters that provide the maximum (or minimum) value of a target function.



Uses of Optimization

In the field of machine learning, optimization techniques can be used to find the parameters for classification algorithms such as:

- Artificial Neural Networks
- Support Vector Machines

These classification algorithms often require the user to supply certain coefficients, which often have to be found by trial and error or exhaustive search.

Canonical PSO Algorithm

- Introduction
- **Canonical PSO Algorithm**
- PSO Algorithm Example
- Classifier Optimization
- Conclusion

Origins of PSO

PSO was first described by James Kennedy and Russell Eberhart in 1995.

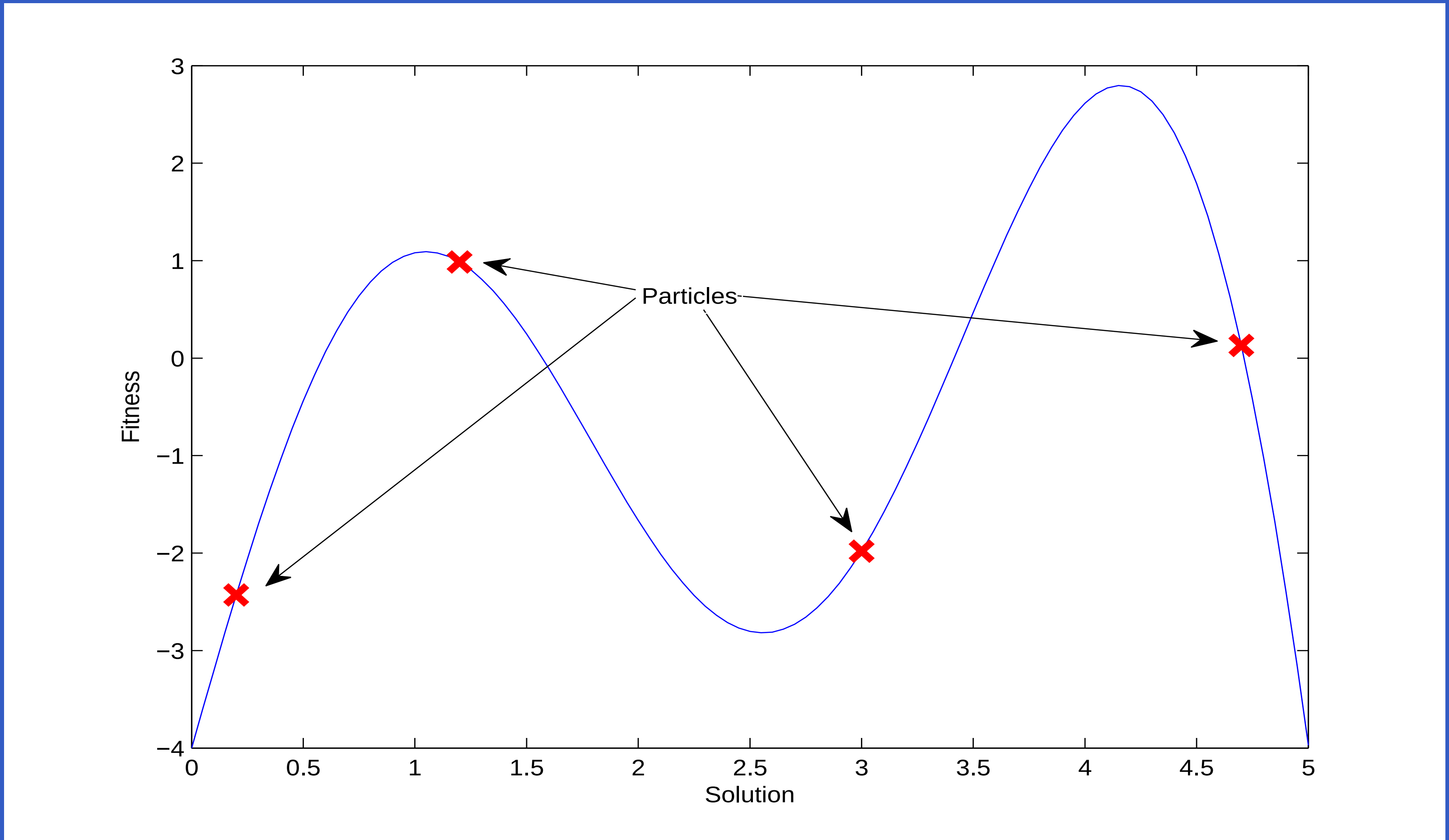
Derived from two concepts:

- The observation of swarming habits of animals such as birds or fish
- The field of evolutionary computation (such as genetic algorithms)

PSO Concepts

- The PSO algorithm maintains multiple potential solutions at one time
- During each iteration of the algorithm, each solution is evaluated by an objective function to determine its fitness
- Each solution is represented by a particle in the fitness landscape (search space)
- The particles “fly” or “swarm” through the search space to find the maximum value returned by the objective function

Fitness Landscape



Maintained Information

Each particle maintains:

- Position in the search space (solution and fitness)
- Velocity
- Individual best position

In addition, the swarm maintains its global best position.

Canonical PSO Algorithm

The PSO algorithm consists of just three steps:

1. Evaluate fitness of each particle
2. Update individual and global bests
3. Update velocity and position of each particle

These steps are repeated until some stopping condition is met.

Velocity Update

Each particle's velocity is updated using this equation:

$$v_i(t+1) = wv_i(t)+c_1r_1[\hat{x}_i(t)-x_i(t)]+c_2r_2[g(t)-x_i(t)]$$

- i is the particle index
- w is the inertial coefficient Hệ số quán tính để lao về phía trước
- c_1, c_2 are acceleration coefficients, $0 \leq c_1, c_2 \leq 2$ Các hệ số gia tốc
- r_1, r_2 are random values ($0 \leq r_1, r_2 \leq 1$) regenerated every velocity update

Velocity Update

Each particle's velocity is updated using this equation:

$$v_i(t+1) = wv_i(t)+c_1r_1[\hat{x}_i(t)-x_i(t)]+c_2r_2[g(t)-x_i(t)]$$

- $v_i(t)$ is the particle's velocity at time t
- $x_i(t)$ is the particle's position at time t
- $\hat{x}_i(t)$ is the particle's individual best solution as of time t
- $g(t)$ is the swarm's best solution as of time t

Velocity Update – Inertia Component

$$v_i(t+1) = \mathbf{wv_i(t)} + c_1 r_1[\hat{x}_i(t) - x_i(t)] + c_2 r_2[g(t) - x_i(t)]$$

- Keeps the particle moving in the same direction it was originally heading
- Inertia coefficient w usually between 0.8 and 1.2
- Lower values speed up convergence, higher values encourage exploring the search space

Velocity Update – Cognitive Component

$$v_i(t+1) = wv_i(t) + \mathbf{c_1 r_1}[\mathbf{\hat{x}_i(t)} - \mathbf{x_i(t)}] + c_2 r_2[g(t) - x_i(t)]$$

- Acts as the particle's memory, causing it to return to its individual best regions of the search space
- Cognitive coefficient c_1 usually close to 2
- Coefficient limits the size of the step the particle takes toward its individual best $\hat{x}_i$

Velocity Update – Social Component

$$v_i(t+1) = wv_i(t)+c_1r_1[\hat{x}_i(t)-x_i(t)]+\mathbf{c_2r_2[g(t) - x_i(t)]}$$

- Causes the particle to move to the best regions the swarm has found so far
- Social coefficient c_2 usually close to 2
- Coefficient limits the size of the step the particle takes toward the global best g

Position Update

Each particle's position is updated using this equation:

$$x_i(t+1) = x_i(t) + v_i(t+1)$$

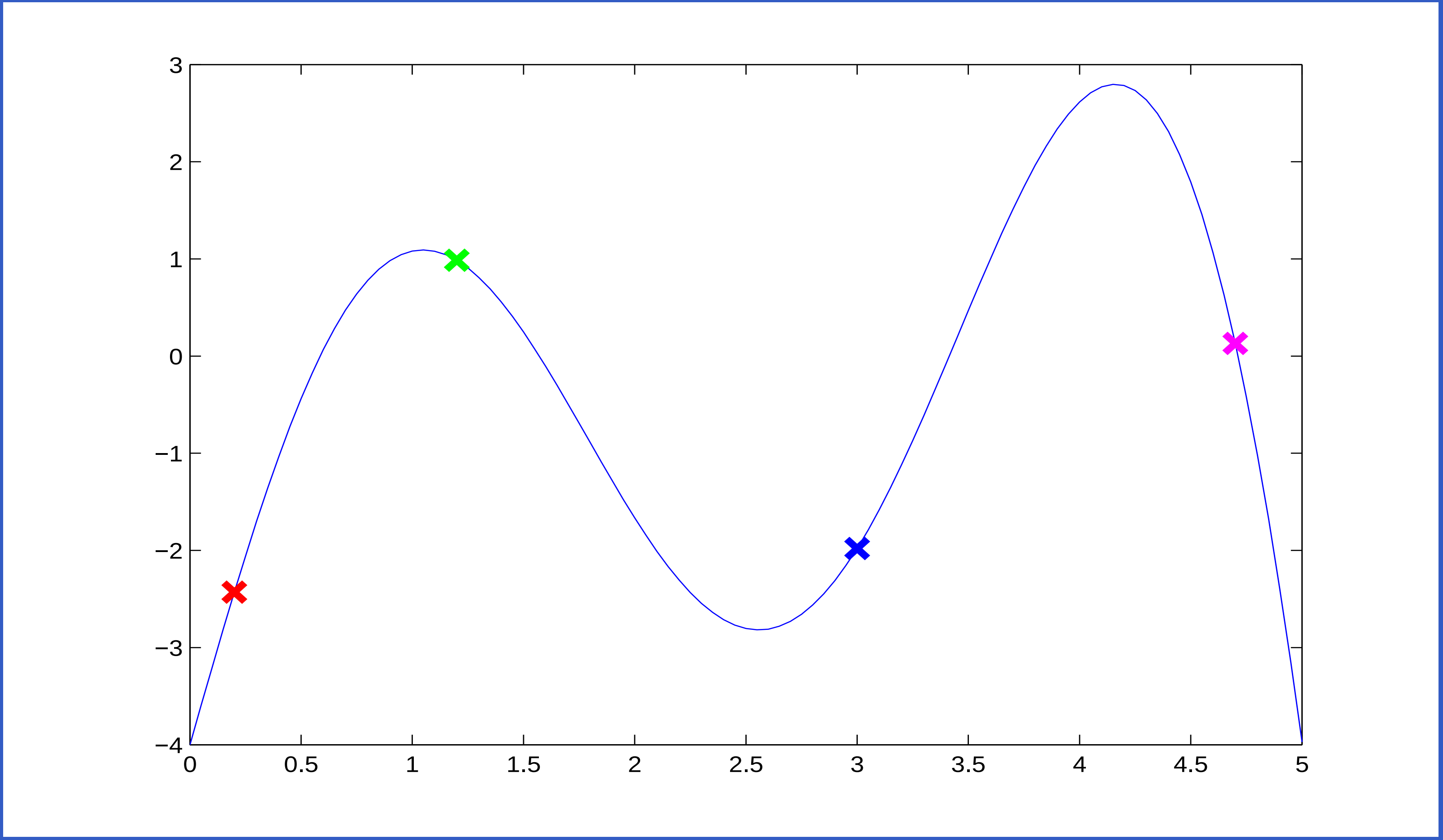
PSO Algorithm Example

PSO Algorithm Redux

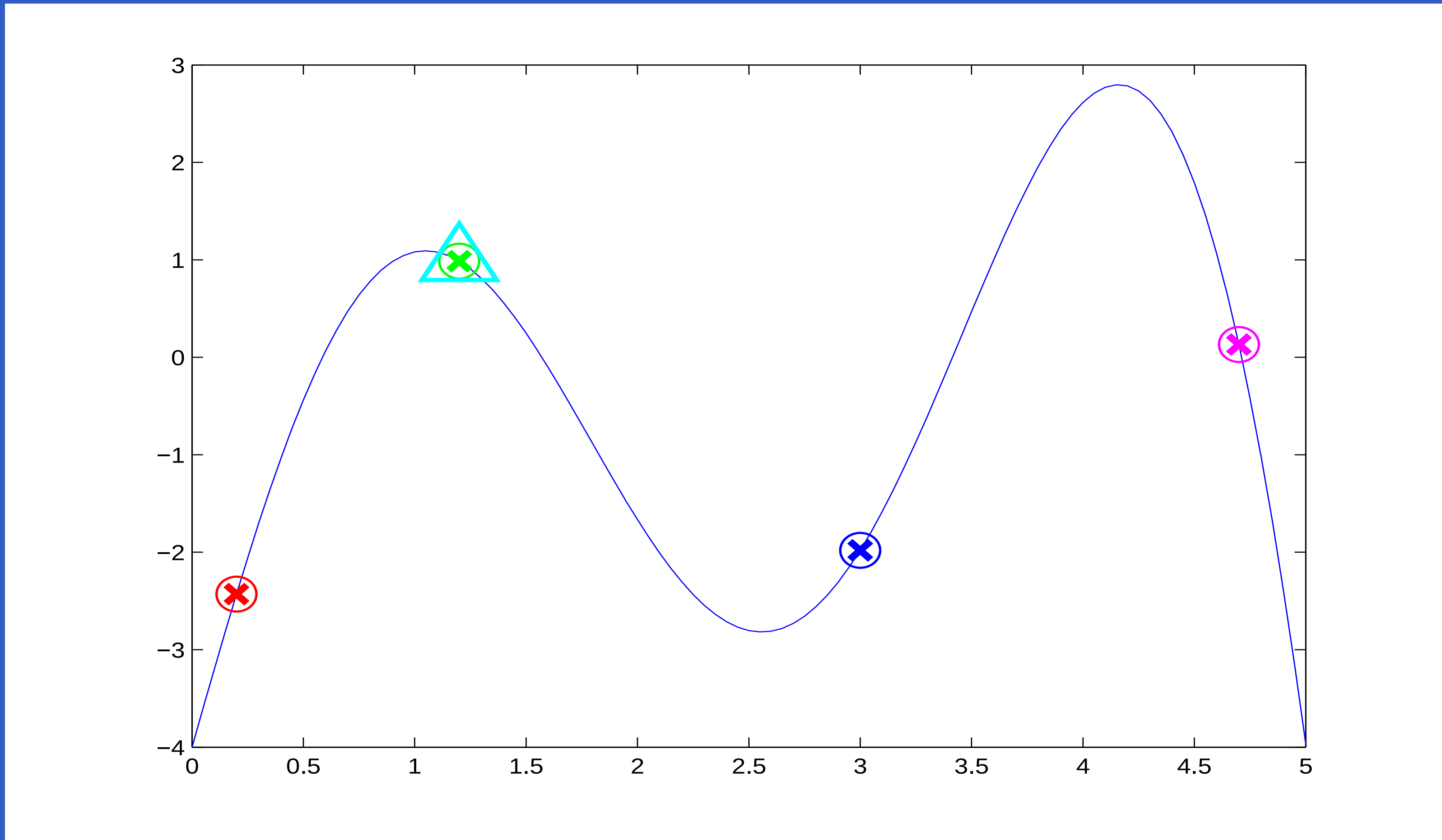
Repeat until stopping condition is met:

1. Evaluate fitness of each particle
2. Update individual and global bests
3. Update velocity and position of each particle

Fitness Evaluation (t=1)

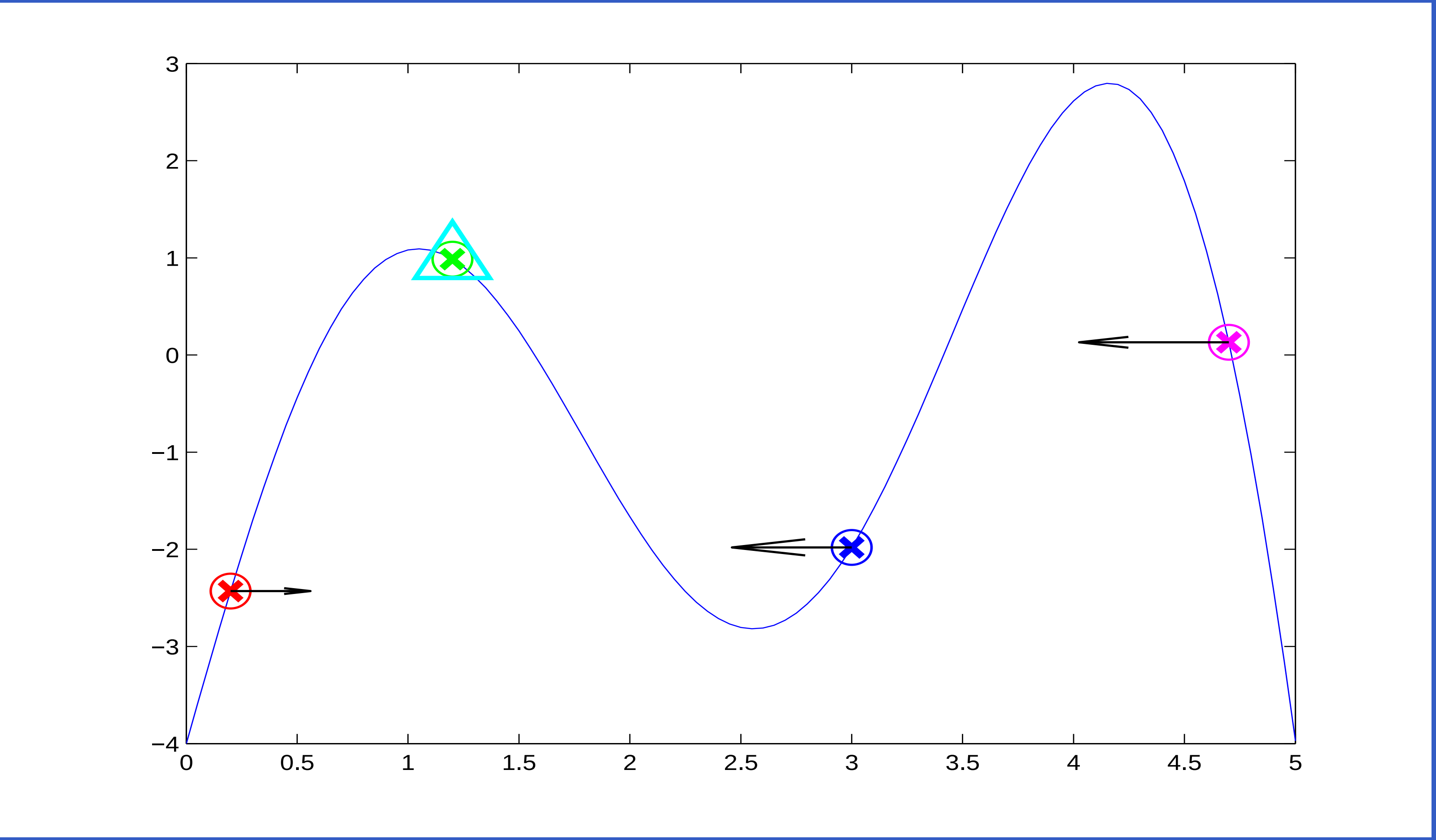


Update Individual / Global Bests (t=1)

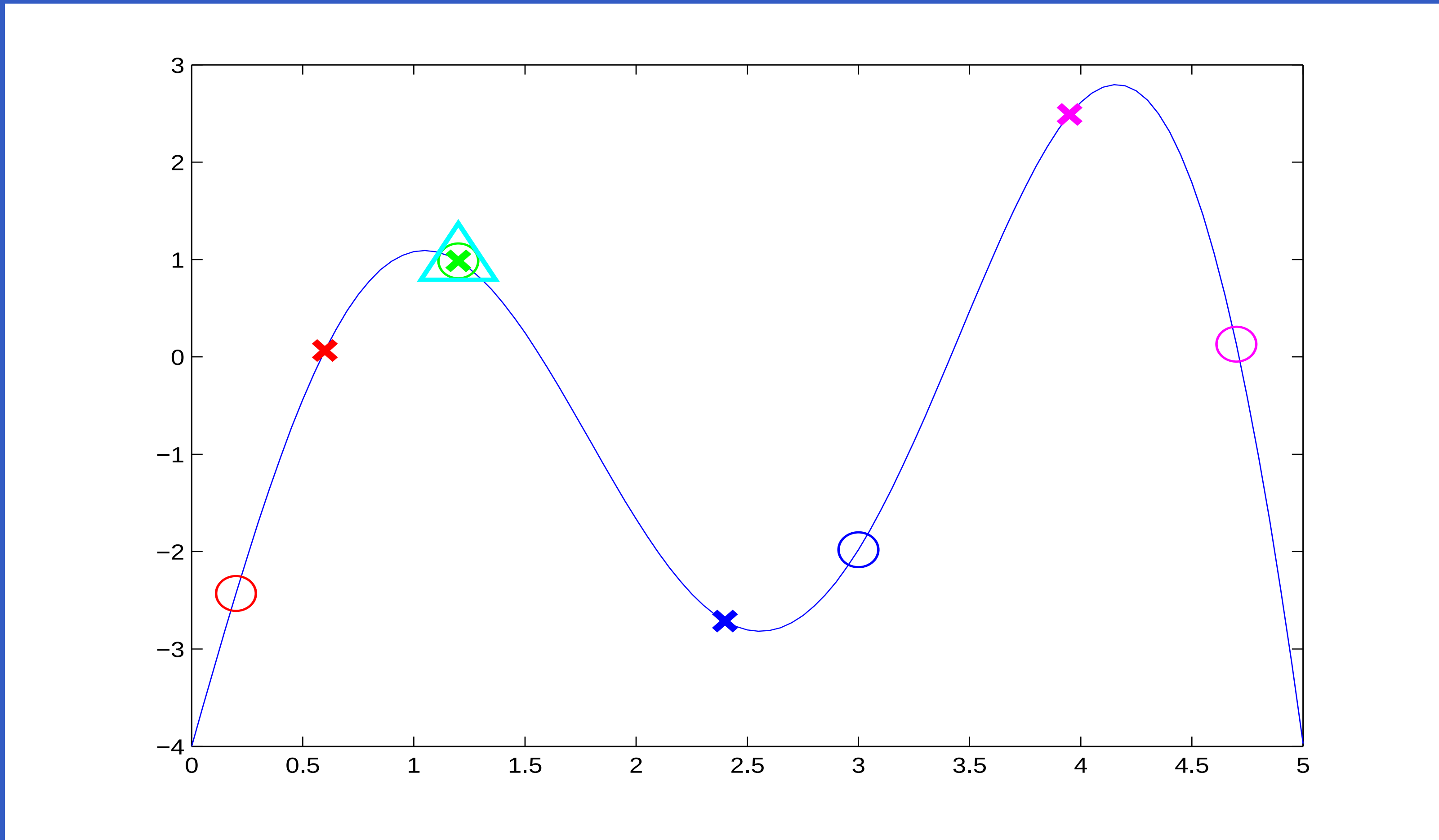


Update Velocity and Position (t=1)

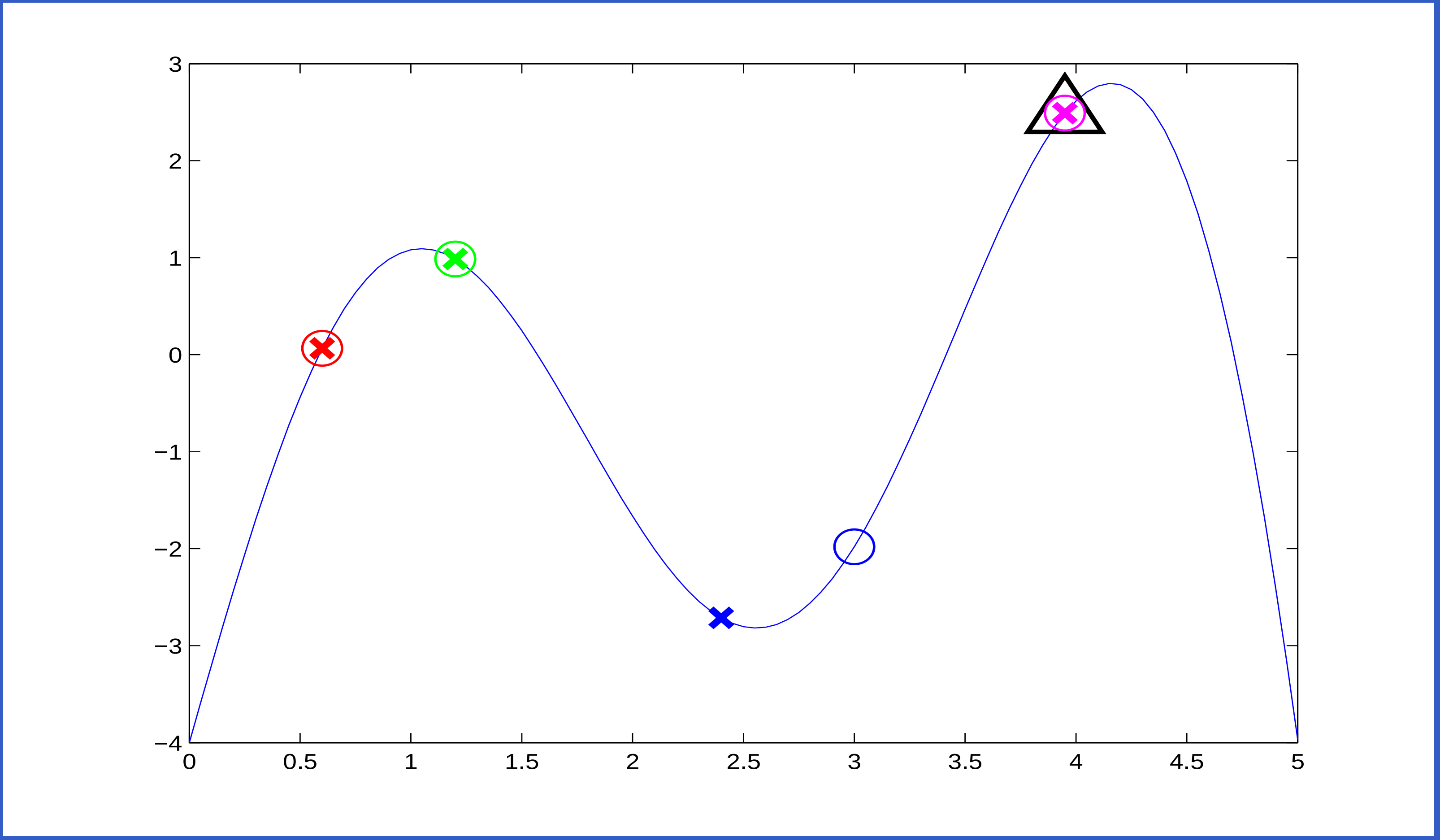
$$v_i(t+1) = wv_i(t)+c_1r_1[\hat{x}_i(t)-x_i(t)]+c_2r_2[g(t)-x_i(t)]$$



Fitness Evaluation (t=2)

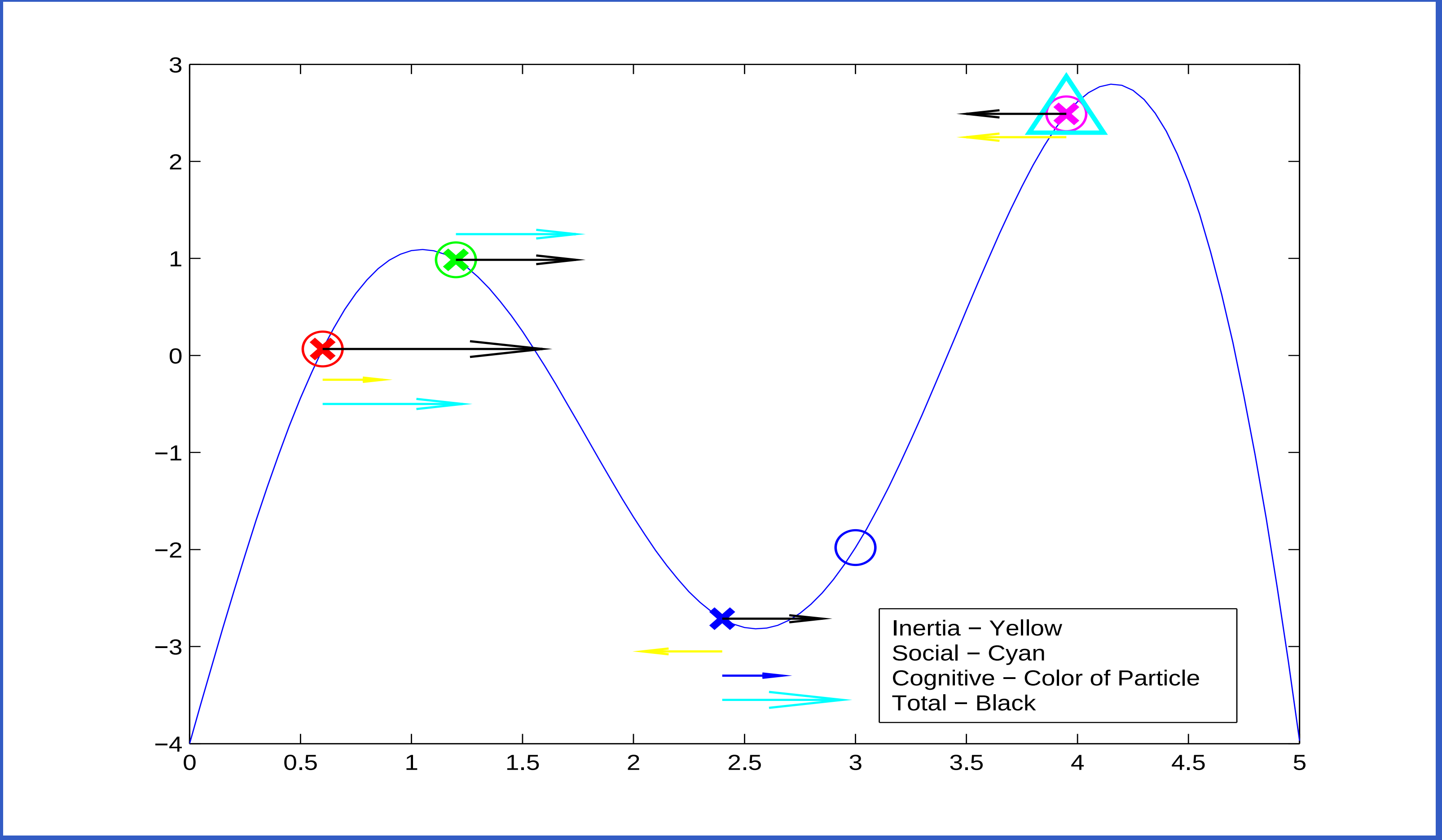


Update Individual / Global Bests (t=2)



Update Velocity and Position (t=2)

$$v_i(t+1) = wv_i(t)+c_1r_1[\hat{x}_i(t)-x_i(t)]+c_2r_2[g(t)-x_i(t)]$$



Classifier Optimization

- Introduction
- Canonical PSO Algorithm
- PSO Algorithm Example
- **Classifier Optimization**
- Conclusion

Support Vector Machines

Support Vector Machines (SVMs) are a group of machine learning techniques used to classify data.

- Effective at classifying even non-linear datasets
- Slow to train
- When being trained, they require the specification of parameters which can greatly enhance or impede the SVM's effectiveness

Support Vector Machine Parameters

One specific type of SVM, a Cost-based Support Vector Classifier (C-SVC), requires two parameters:

- Cost parameter (C), which is typically anywhere between 2^{-5} and 2^{20}
- Gamma parameter (γ), which is typically anywhere between 2^{-20} and 2^{3}

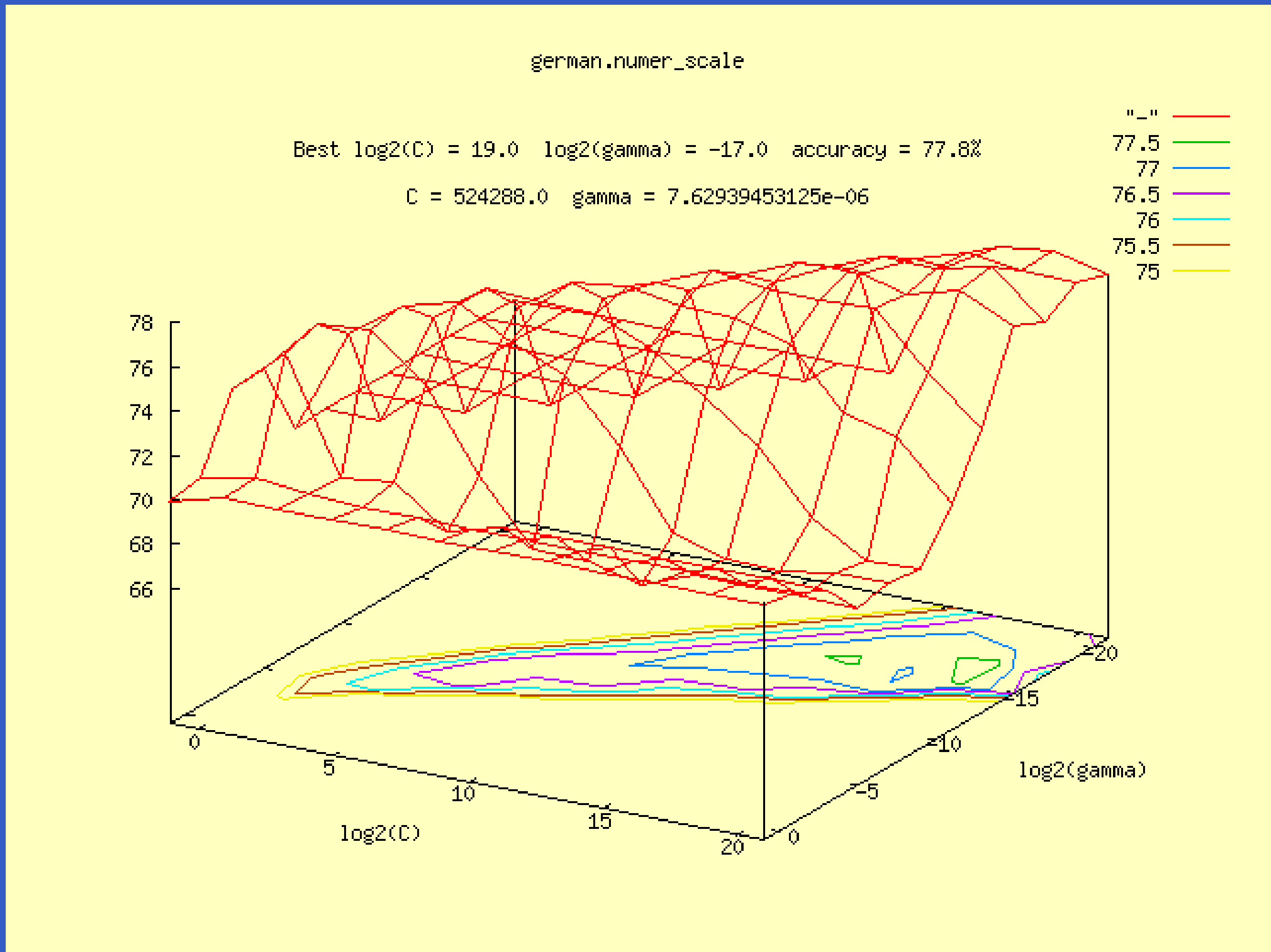
Supper Vector Machine Parameters

For different datasets, the optimal values for these parameters can be very different, even on the same type of C-SVC.

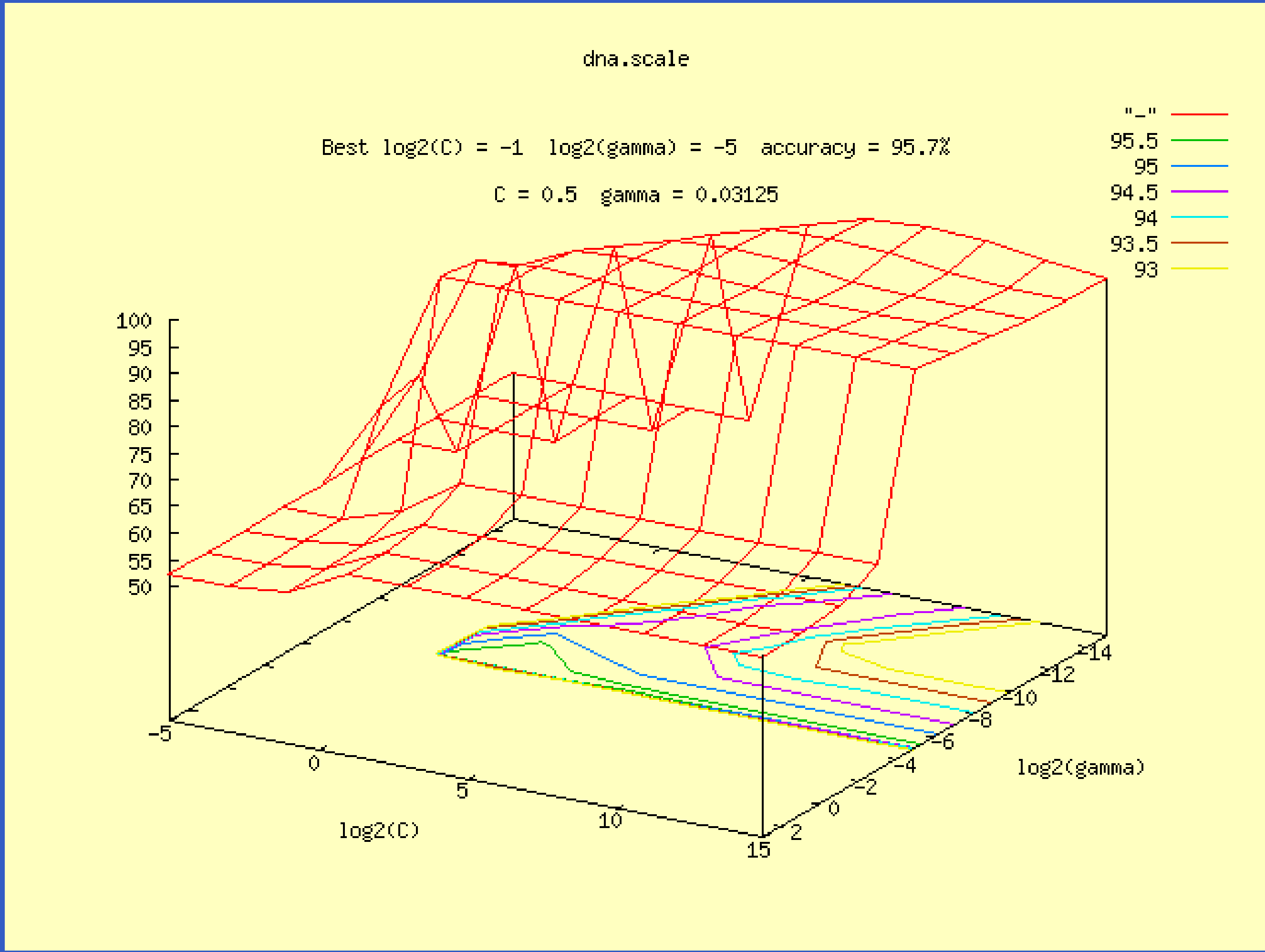
To find the optimal parameters, two approaches are often used:

- Random selection
- Grid search

Financial Data



DNA Splicing Data



Grid Search Problems

While effective, grid search has some problems:

- Computationally intensive
  - Financial Data – 144 SVM training runs, approximately 9 minutes
  - DNA Splicing Data - 110 SVM training runs, approximately 48 minutes
- Only as exact as the spacing of the grid (coarseness of search), although once a peak has been identified, it can be searched more closely

Applying PSO to SVM Parameters

Alternatively, PSO can be used to parameterize SVMs, using the SVM training run as the objective function.

Implementation considerations:

- Finding maximum among two dimensions (as opposed to just one, as in the example)
- Parameters less than zero are invalid, so position updates should not move parameter below zero

PSO Parameters

Parameters used for PSO algorithm:

- Number of particles: 8
- Inertia coefficient (w): .75
- Cognitive coefficient(c_1): 1.8
- Social coefficient(c_2): 2
- Number of iterations: 10 (or no improvement for 4 consecutive iterations)

Preliminary Results

	Dataset	
	DNA Splicing	Financial
Grid search		
Num. Training Runs	110	144
Max. Accuracy	95.7%	77.8%
PSO		
Num. Training Runs	56	72
Max. Accuracy	96.1%	77.7%

Analysis of Results

- Results are still preliminary, but encouraging
- Due to randomized aspects of PSO algorithm, the optimization process would need to be run several times to determine if results are consistent
- Alternative PSO parameters can be attempted, and their effectiveness measured

Conclusion

- Introduction
- Canonical PSO Algorithm
- PSO Algorithm Example
- Classifier Optimization
- **Conclusion**

Conclusions and Future Work

Conclusions:

- Significant speedup using PSO over exhaustive search
- Additional testing needed

Future Work:

- Other PSO variants can be tried
- Need to find optimal parameters for PSO itself

Questions

Questions?