Comparison-Based Optimizers Need Comparison-Based Surrogates

Ilya Loshchilov[1,2], Marc Schoenauer[1,2], Michèle Sebag[2,1]

[1] TAO Project-team, INRIA Saclay - Île-de-France

[2] and Laboratoire de Recherche en Informatique (UMR CNRS 8623)
Université Paris-Sud, 91128 Orsay Cedex, France

Content

Why Comparison-Based Surrogates ?

Surrogate Model (Meta-Model) assisted optimization

- Construct the approximation model $M(x)$ of $f(x)$.
- Optimize the model $M(x)$ *in lieu* of $f(x)$ to **reduce** the number of **costly evaluations** of the function $f(x)$.

Example

- $f(x) = x_1^2 + (x_1 + x_2)^2$.
- An efficient Evolutionary Algorithm (EA) **with surrogate models** may be 4.3 faster on $f(x)$.
- But the same EA is only 2.4 faster on $f'(x) = f(x)^{1/4}$! [a]

[a] CMA-ES with quadratic meta-model (lmm-CMA-ES) on fSchwefel 2-D

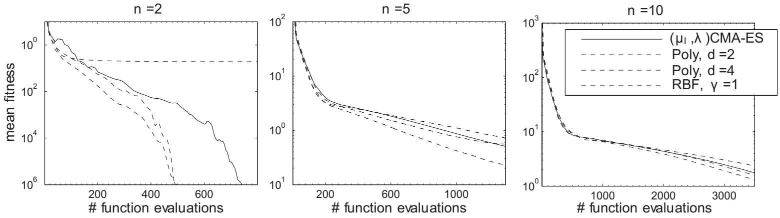
Ordinal Regression in Evolutionary Computation

- Goal: Find the function $F(x)$ which preserves the ordering of the training points x_i (x_i has rank i):

$$x_i \succ x_j \Leftrightarrow F(x_i) > F(x_j)$$

- $F(x)$ is invariant to any **rank-preserving** transformation.

CMA-ES with Rank Support Vector Machine on Rosenbrock: [1]



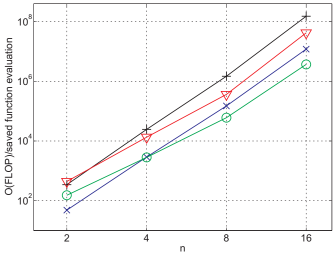
[1] T. Runarsson (2006). "Ordinal Regression in Evolutionary Computation"

Exploit the local topography of the function

- CMA-ES **adapts** the covariance matrix C which describes the **local structure** of the function.
- **Mahalanobis** (fully weighted Euclidean) distance:

$$d(x_i, x_j) = \sqrt{(x_i - x_j)^T C^{-1} (x_i - x_j)}$$

Results of CMA-ES with quadratic meta-models: [2]



lmm-CMA-ES

- Speed-up: a factor of 2-4 for $n \geq 4$
- Complexity: from $O(n^4)$ to $O(n^6)$
- Rank-preserving invariance: NO
- becomes intractable for n>16

[2] S. Kern et al. (2006). "Local Meta-Models for Optimization Using Evolution Strategies"

Tractable or Efficient ?

Answer: Tractable and Efficient and Invariant.

Ingredients: CMA-ES (Adaptive Encoding) and Rank SVM.

Covariance Matrix Adaptation Evolution Strategy

Decompose to understand

- While CMA-ES by definition is CMA and ES, only recently the **algorithmic decomposition** has been presented. [3]

Algorithm 1 CMA-ES = Adaptive Encoding + ES

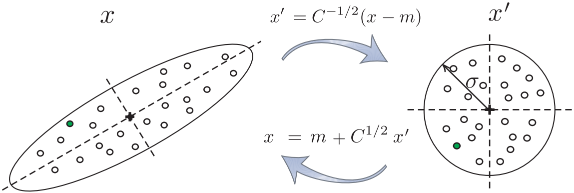
1: $x_i \leftarrow m + \sigma \mathcal{N}_i(0, I)$, for $i = 1 \ldots \lambda$
2: $f_i \leftarrow f(Bx_i)$, for $i = 1 \ldots \lambda$
3: **if** Evolution Strategy (ES) **then**
4: $\quad \sigma \leftarrow \sigma exp^{\propto(\frac{\texttt{success rate}}{\texttt{expected success rate}} - 1)}$
5: **if** Cumulative Step-Size Adaptation ES (CSA-ES) **then**
6: $\quad \sigma \leftarrow \sigma exp^{\propto(\frac{\|\texttt{evolution path}\|}{\|\texttt{expected evolution path}\|} - 1)}$
7: $B \leftarrow$AE$_{\text{CMA}}$-Update$(Bx_1, \ldots, Bx_\mu)$

[3] N. Hansen (2008). "Adaptive Encoding: How to Render Search Coordinate System Invariant"

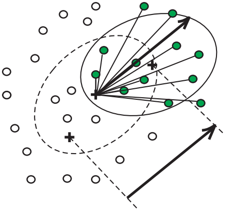
Adaptive Encoding

Inspired by Principal Component Analysis (PCA)

Principal Component Analysis

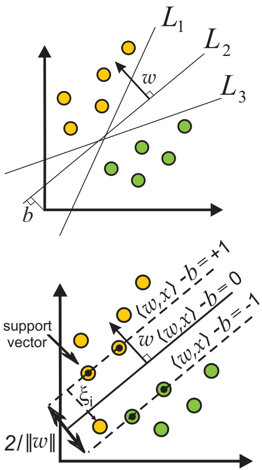


Adaptive Encoding Update



Support Vector Machine for Classification

Linear Classifier



Main Idea

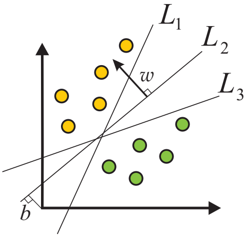
Training Data:
$D = \{(x_i, y_i)|x_i \in \mathbb{R}^p, y_i \in \{-1, +1\}\}_{i=1}^n$
$\langle w, x_i \rangle \geq b + \epsilon \Rightarrow y_i = +1;$
$\langle w, x_i \rangle \leq b - \epsilon \Rightarrow y_i = -1;$
Dividing by $\epsilon > 0$:
$\langle w, x_i \rangle - b \geq +1 \Rightarrow y_i = +1;$
$\langle w, x_i \rangle - b \leq -1 \Rightarrow y_i = -1;$

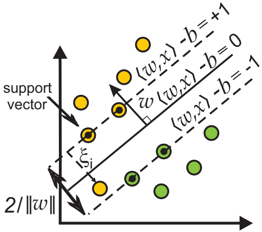
Optimization Problem: Primal Form

Minimize$_{\{w,\ \xi\}}$ $\frac{1}{2}||w||^2 + C\sum_{i=1}^n \xi_i$
subject to: $y_i(\langle w, x_i \rangle - b) \geq 1 - \xi_i,\ \xi_i \geq 0$

Support Vector Machine for Classification

Linear Classifier





Optimization Problem: Dual Form

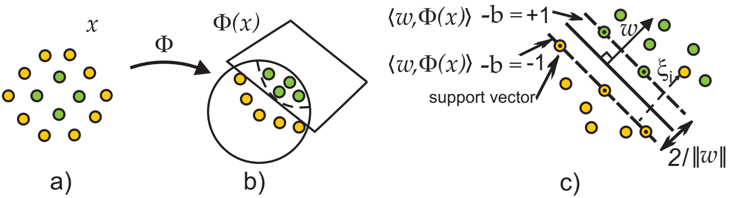
From Lagrange Theorem, instead of minimize F:
Minimize$_{\{\alpha,G\}} F - \sum_i \alpha_i G_i$
subject to: $\alpha_i \geq 0$, $G_i \geq 0$
Leaving the details, **Dual form**:
Maximize$_{\{\alpha\}} \sum_i^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle$
subject to: $0 \leq \alpha_i \leq C$, $\sum_i^n \alpha_i y_i = 0$

Properties

Decision Function:
$F(x) = sign(\sum_i^n \alpha_i y_i \langle x_i, x \rangle - b)$
The Dual form may be solved using standard quadratic programming solver.

Support Vector Machine for Classification

Non-Linear Classifier



Non-linear classification with the "Kernel trick"

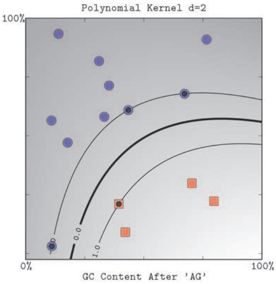
Maximize$_{\{\alpha\}} \sum_i^n \alpha_i - \frac{1}{2}\sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j K(x_i, x_j)$
subject to: $a_i \geq 0$, $\sum_i^n \alpha_i y_i = 0$,
where $K(x, x') =_{def} < \Phi(x), \Phi(x') >$ **is the Kernel function**
Decision Function: $F(x) = sign(\sum_i^n \alpha_i y_i K(x_i, x) - b)$

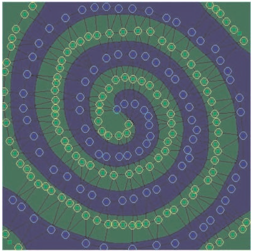
Support Vector Machine for Classification

Non-Linear Classifier: Kernels

- Polynomial: $k(x_i, x_j) = (\langle x_i, x_j \rangle + 1)^d$
- Gaussian or Radial Basis Function: $k(x_i, x_j) = exp(\frac{\|x_i - x_j\|^2}{2\sigma^2})$
- Hyperbolic tangent: $k(x_i, x_j) = tanh(k \langle x_i, x_j \rangle + c)$

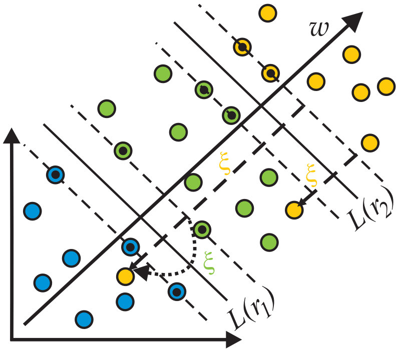
Examples for Polynomial (left) and Gaussian (right) Kernels:





Ranking Support Vector Machine

Find $F(x)$ which preserves the ordering of the training points.



Ranking Support Vector Machine

Primal problem

Minimize$_{\{w,\ \xi\}}$ $\frac{1}{2}||w||^2 + \sum_{i=1}^{N} C_i \xi_i$

subject to $\begin{cases} \langle\, w, \Phi(x_i) - \Phi(x_{i+1})\,\rangle \geq 1 - \xi_i \quad (i = 1 \ldots N-1) \\ \xi_i \geq 0 \quad (i = 1 \ldots N-1) \end{cases}$

Dual problem

Maximize$_{\{\alpha\}}$ $\sum_{i}^{N-1} \alpha_i - \sum_{i,j}^{N-1} \alpha_{ij} K(x_i - x_{i+1}, x_j - x_{j+1}))$

subject to $0 \leq \alpha_i \leq C_i \quad (i = 1 \ldots N-1)$

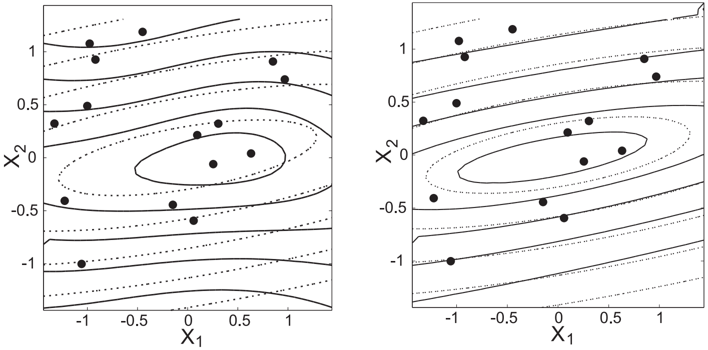
Rank Surrogate Function in the case 1 rank = 1 point

$$\mathcal{F}(x) = \sum_{i=1}^{N-1} \alpha_i (K(x_i, x) - K(x_{i+1}, x))$$

Model Learning

Non-separable Ellipsoid problem

$$K(x_i, x_j) = e^{-\frac{(x_i - x_j)^T (x_i - x_j)}{2\sigma^2}}; \quad K_C(x_i, x_j) = e^{-\frac{(x_i - x_j)^T C^{-1} (x_i - x_j)}{2\sigma^2}}$$



Optimization or filtering?

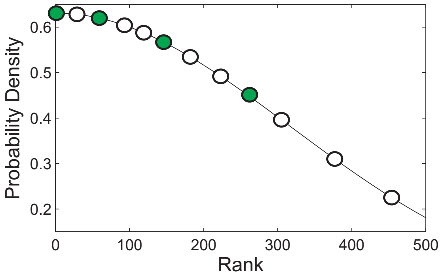
Don't be too greedy

- Optimization: Significant Potential Speed-Up if the surrogate model is global and accurate enough
- Filtering: "Guaranteed" Speed-Up with the local surrogate model

Prescreen (λ ○) → Retain ○ with rank $a < N_{test}$, $a \sim N_{test}\mathcal{N}(0, \sigma^2_{sel0})$

Evaluate (λ′ ●) → Retain ● with rank $a < \lambda$, $a \sim \lambda\mathcal{N}(0, \sigma^2_{sel1})$

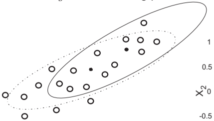
$$N_{test} = 500$$
$$\sigma^2_{sel0} = 0.4$$
$$\sigma^2_{sel1} = 0.8$$
$$\lambda = 12$$
$$\lambda' = 4$$



ACM-ES Optimization Loop

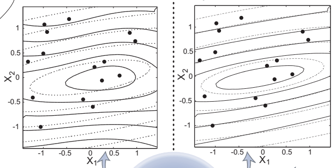
A. Select training points

1. Select best $N_{training} = k\sqrt{d}$ training points.

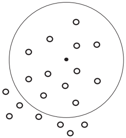


2. The change of coordinates, defined from the current covariance matrix C and the current mean value m, reads [4]:

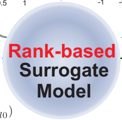
$$x'_j = C^{-1/2}(x_j - m)$$



B. Build a surrogate model

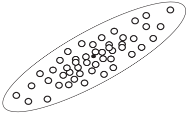


3. Build a surrogate model using Rank SVM.



C. Generate pre-children

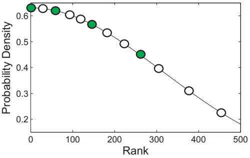
4. Generate $N_{test} = 500$ pre-children and rank them according to surrogate fitness function.



D. Select most promising children

5. Prescreen (λ ○) → Retain ○ with rank $a < N_{test}$, $a \sim N_{test}\mathcal{N}(0, \sigma^2_{sel0})$
6. Evaluate (λ' ●) → Retain ● with rank $a < \lambda$, $a \sim \lambda\mathcal{N}(0, \sigma^2_{sel1})$

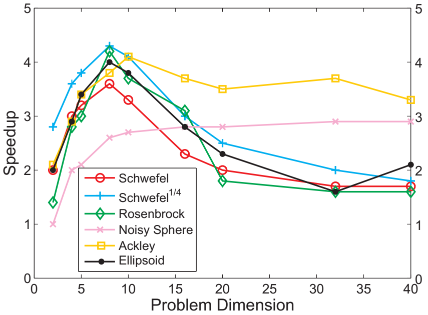
$N_{test} = 500$
$\sigma^2_{sel0} = 0.4$
$\sigma^2_{sel1} = 0.8$
$\lambda = 12$
$\lambda' = 4$



7. Add new λ' ● training points and update parameters of CMA-ES.

Results

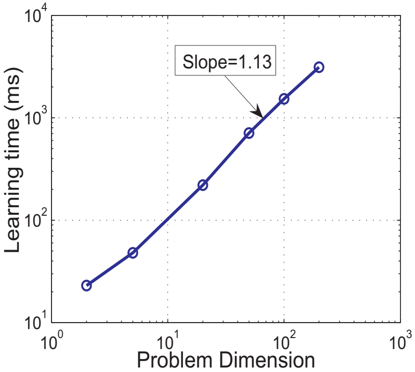
Speed-up



Function	n	λ	λ′	e	ACM-ES	spu	CMA-ES
Schwefel	10	10	3		801 ± 36	3.3	2667 ± 87
	20	12	4		3531 ± 179	2.0	7042 ± 172
	40	15	5		13440 ± 281	1.7	22400 ± 289
Schwefel$^{1/4}$	10	10	3		1774 ± 37	4.1	7220 ± 206
	20	12	4		6138 ± 82	2.5	15600 ± 294
	40	15	5		22658 ± 390	1.8	41534 ± 466
Rosenbrock	10	10	3		2059 ± 143 (0.95)	3.7	7669 ± 691 (0.90)
	20	12	4		11793 ± 574 (0.75)	1.8	21794 ± 1529
	40	15	5		49750 ± 2412 (0.9)	1.6	82043 ± 3991
NoisySphere	10	10	3	0.15	766 ± 90 (0.95)	2.7	2058 ± 148
	20	12	4	0.11	1361 ± 212	2.8	3777 ± 127
	40	15	5	0.08	2409 ± 120	2.9	7023 ± 173
Ackley	10	10	3		892 ± 28	4.1	3641 ± 154
	20	12	4		1884 ± 50	3.5	6641 ± 108
	40	15	5		3690 ± 80	3.3	12084 ± 247
Ellipsoid	10	10	3		1628 ± 95	3.8	6211 ± 264
	20	12	4		8250 ± 393	2.3	19060 ± 501
	40	15	5		33602 ± 548	2.1	69642 ± 644
Rastrigin	5	140	70		23293 ± 1374 (0.3)	0.5	12310 ± 1098 (0.75)

Results
Learning Time

Cost of model learning/testing increases quasi-linearly with *d* on Sphere function:



Summary

ACM-ES

- ACM-ES is from 2 to 4 times faster on **Uni-Modal Problems**.
- Invariant to rank-preserving transformation: Yes
- The computation complexity (the cost of speed-up) is $O(n)$ comparing to $O(n^6)$
- The source code is available online: `http://www.lri.fr/~ilya/publications/ACMESppsn2010.zip`

Open Questions

- Extention to multi-modal optimization
- Adaptation of selection pressure and surrogate model complexity

Summary

Thank you for your attention!

Questions?

Parameters

SVM Learning:

- Number of training points: $N_{training} = 30\sqrt{d}$ for all problems, except Rosenbrock and Rastrigin, where $N_{training} = 70\sqrt{(d)}$
- Number of iterations: $N_{iter} = 50000\sqrt{d}$
- Kernel function: RBF function with σ equal to the average distance of the training points
- The cost of constraint violation: $C_i = 10^6(N_{training} - i)^{2.0}$

Offspring Selection Procedure:

- Number of test points: $N_{test} = 500$
- Number of evaluated offsprings: $\lambda' = \frac{\lambda}{3}$
- Offspring selection pressure parameters: $\sigma^2_{sel0} = 2\sigma^2_{sel1} = 0.8$