Evolutionary Computation for Non-Convex Machine Learning

Ke Tang
Southern University of Science and Technology

Yang Yu
Nanjing University

Outlines

- **The need of machine learning**
- **Evolutionary computation for supervised Learning – New case studies.**
- **Evolutionary Reinforcement Learning**

The Need of Machine Learning

- Three key components of machine learning:
  - Data/model representation
  - Evaluation
  - Training algorithm

- Most modern machine learning problems are essentially searching for the model that is optimal with respect to some objective function (e.g., generalization).

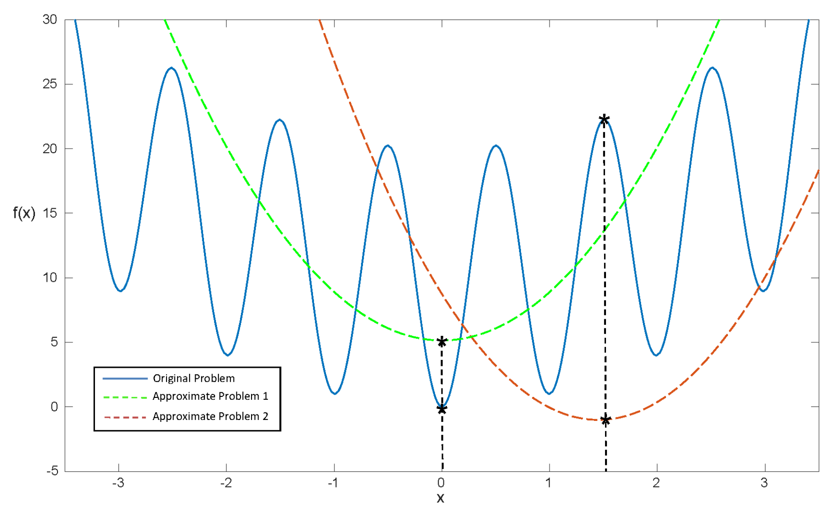
- Optimization algorithms thus play a crucial role in machine learning.

The Need of Machine Learning

- Many machine learning tasks, when formulated as an optimization problem, cannot be well solved by traditional (e.g., convex) optimization techniques.

Plan A: Approximate the original (non-convex) optimization problem with a convex one.

Plan B: Seek an approximate solution to the original problem with heuristic methods, e.g., Evolutionary algorithms.



The Need of Machine Learning

EAs have been applied to a large variety of learning problems in the past decades.

- Data representation
  - Feature selection
  - Feature extraction
  - Dimensionality Reduction
- Model training
  - Decision tree
  - Neural networks
  - Rule-based systems
  - Clustering
- Hyper-parameter tuning

The Need of Machine Learning

Machine learning has its own characteristics that calls for specialized EAs.

- Huge problem size (i.e., the search space).
- Noisy fitness evaluation (since the generalization cannot be precisely measured)
- Expensive fitness evaluation
- Theoretical guarantee is more preferred than in other areas.

Case Study (1)

- Subset selection: select a subset of size k from a total set of n variables for optimizing some criterion.

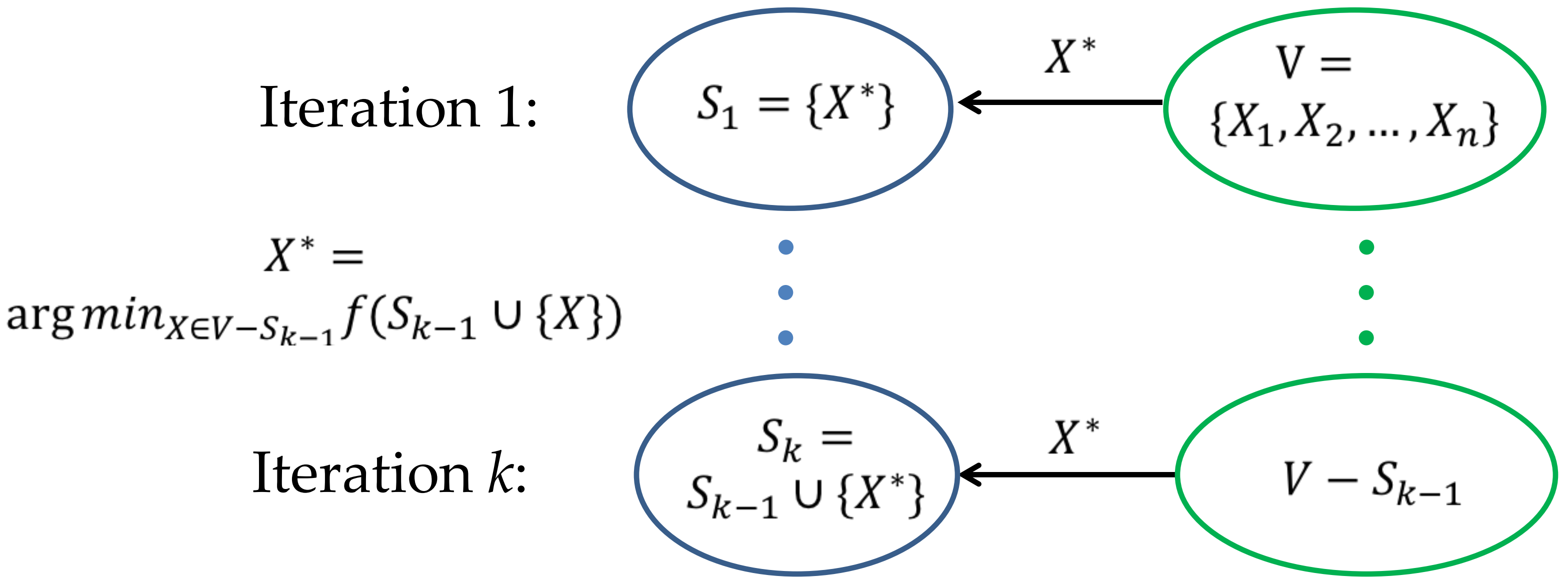
Formally stated: given all variables $V = \{X_1, \ldots, X_n\}$, a criterion f and a positive integer k,

$$\arg\ \min_{S \subseteq V}\ f(S) \quad s.t. \quad |S| \leq k.$$

- NP-hard in general [Natarajan,1995; Davis et al., 1997] and arises in many learning problems:
  - Feature Selection
  - Sparse Learning
  - Compressed Sensing

Case Study (1)

- Greedy algorithms [Gilbert et al., SODA'03; Tropp, TIT'04]
  - Process: iteratively select or abandon one variable that makes the criterion currently optimized
  - Weakness: get stuck in local optima due to the greedy behavior



Case Study (1)

- Convex relaxation methods [Tibshirani, JRSSB'96; Zou & Hastie, JRSSB'05]
  - Process: replace the set size constraint with convex constraints, then find the optimal solutions to the relaxed problem.
  - Weakness: the optimal solution of the relaxed problem may be distant to the true optimum.

$$\arg min_{S\subseteq V} \quad f(S) \quad s.t. \quad |S| \le k$$

$$\Updownarrow$$

$$\arg min_{w\in R^n} \quad g(w) \quad s.t. \quad |w|_0 \le k$$ non-convex

$$\arg min_{w\in R^n} \quad g(w) \quad s.t. \quad |w|_1 \le k$$ convex

Case Study (1)

- There have been numerous EAs for subset selection, while rigorous theoretical guarantee is few.

- Subset Selection as a bi-objective optimization problem

POSS (Pareto Optimization for Subset Selection)

The basic idea:

$$min_{S \subseteq V} \ f(S) \quad s.t. \quad |S| \leq k$$ constrained

$$\Downarrow$$

$$min_{S \subseteq V} \ (f(S), |S|)$$ bi-objective

Case Study (1)

Algorithm 1 POSS

Input: all variables $V = \{X_1, \ldots, X_n\}$, a given objective f and an integer parameter $k \in [1, n]$
Parameter: the number of iterations T
Output: a subset of V with at most k variables
Process:

Let $s = \{0\}^n$ and $P = \{s\}$.
Let $t = 0$.
while $t < T$ **do**
    Select s from P uniformly at random.
    Generate s' by flipping each bit of s with prob. $\frac{1}{n}$.
    Evaluate $f_1(s')$ and $f_2(s')$.
    if $\nexists z \in P$ such that $z \prec s'$ **then**
        $Q = \{z \in P \mid s' \preceq z\}$.
        $P = (P \setminus Q) \cup \{s'\}$.
    end if
    $t = t + 1$.
end while
return $\arg\min_{s \in P, |s| \leq k} f_1(s)$

Initialization: randomly generate a solution, put it into the archive P

Reproduction: pick a solution randomly from P, and randomly change it to make a new one

Evaluation & Selection: if the new solution is not dominated, put it into P and weed out bad solutions

Output: select the best feasible solution

- Chaoqian paper

Case Study (1)

- Sparse regression is to find a sparse approximation solution to the regression problem.

Formally stated: given all observation variables $V = \{X_1, \dots, X_n\}$, a predictor variable Z and a positive integer k, define the mean squared error of a subset $S \subseteq V$ as

$$MSE_{Z,S} = min_{\alpha \in R^{|S|}} E[(Z - \sum_{i \in S} \alpha_i X_i)^2]$$

Sparse regression is

$$\arg\ min_{S \subseteq V}\ \ MSE_{Z,S}\ \ s.t.\ \ |S| \leq k.$$

Chao Qian, Yang Yu, and Zhi-Hua Zhou. Subset Selection by Pareto Optimization.In: Advances in Neural Information Processing Systems 28 (NIPS'15), Montreal, Canada, 2015, pp.1765-1773.

Case Study (1)

Previous theoretical bounds:

u: the coherence, γ: the submodular ratio

- [Gilbert et al., 2003]: $(1+\Theta(uk^2))\cdot\ OPT$ on $MSE_{Z,S}$ for $u \in O(1/k)$ by a two-phased approach
- [Tropp et al., 2003; Tropp, 2004]: improve the above bound
- [Das & Kempe, 2008]: $(1-\Theta(uk))\cdot\ OPT$ on $R^2_{Z,S}$ for $u \in O(1/k)$ by the forward regression algorithm
- [Das & Kempe, 2011]: $(1-e^{-\gamma})\cdot\ OPT$ on $R^2_{Z,S}$ by the forward regression algorithm
- [Shalev-Shwartz et al., 2010; Yuan & Yan, 2013]: lower bounds on $|S|$ for achieving $OPT+\varepsilon$

strongest

Case Study (1)

Theorem 1. *For sparse regression, POSS with* $E[T] \leq 2ek^2n$ *and* $I(\cdot) = 0$ *(i.e., a constant function) finds a set* S *of variables with* $|S| \leq k$ *and* $R^2_{Z,S} \geq (1 - e^{-\gamma_{\emptyset,k}}) \cdot OPT$.

the best previous theoretical guarantee

POSS can do at least as well as previous methods.

Theorem 2. *For the Exponential Decay subclass of sparse regression, POSS with* $E[T] \in O(k^2n^2 \log n)$ *and* $I(\boldsymbol{s} \in \{0,1\}^n) = \min\{i \mid s_i = 1\}$ *can find the optimal solution.*

Proposition 1. *For Example 1 with* $n = 3$, $r_2 = 0.03$, $r_3 = 0.5$, $Cov(Y_1, Z) = Cov(Y_2, Z) = \delta$ *and* $Cov(Y_3, Z) = 0.505\delta$, *FR cannot find the optimal solution for* $k = 2$.

POSS can do strictly better than previous methods.

Case Study (1)

- POSS for Sparse Regression: Summary

the number of iterations

Theory

Theorem 1. For sparse regression, POSS with $\boxed{\mathbb{E}[T] \leq 2ek^2n}$ finds a set S of variables with $|S| \leq k$ and $\boxed{R^2_{Z,S} \geq (1 - e^{-\gamma_{\emptyset,k}}) \cdot OPT.}$

the best known polynomial-time approximation bound [Das & Kempe, ICML'11]

Experiment

Data set	OPT	POSS	FR	FoBa	OMP	RFE	MCP
housing	.7437±.0297	.7437±.0297	.7429±.0300•	.7423±.0301•	.7415±.0300•	.7388±.0304•	.7354±.0297•
eunite2001	.8484±.0132	.8482±.0132	.8348±.0143•	.8442±.0144•	.8349±.0150•	.8424±.0153•	.8320±.0150•
svmguide3	.2705±.0255	.2701±.0257	.2615±.0260•	.2601±.0279•	.2557±.0270•	.2136±.0325•	.2397±.0237•
ionosphere	.5995±.0326	.5990±.0329	.5920±.0352•	.5929±.0346•	.5921±.0353•	.5832±.0415•	.5740±.0348•
sonar	–	.5365±.0410	.5171±.0440•	.5138±.0432•	.5112±.0425•	.4321±.0636•	.4496±.0482•
triazines	–	.4301±.0603	.4150±.0592•	.4107±.0600•	.4073±.0591•	.3615±.0712•	.3793±.0584•
coil2000	–	.0627±.0076	.0624±.0076•	.0619±.0075•	.0619±.0075•	.0363±.0141•	.0570±.0075•
mushrooms	–	.9912±.0020	.9909±.0021•	.9909±.0022•	.9909±.0022•	.6813±.1294•	.8652±.0474•
clean1	–	.4368±.0300	.4169±.0299•	.4145±.0309•	.4132±.0315•	.1596±.0562•	.3563±.0364•
w5a	–	.3376±.0267	.3319±.0247•	.3341±.0258•	.3313±.0246•	.3342±.0276•	.2694±.0385•
gisette	–	.7265±.0098	.7001±.0116•	.6747±.0145•	.6731±.0134•	.5360±.0318•	.5709±.0123•
farm-ads	–	.4217±.0100	.4196±.0101•	.4170±.0113•	.4170±.0113•	–	.3771±.0110•
POSS: win/tie/loss		–	12/0/0	12/0/0	12/0/0	11/0/0	12/0/0

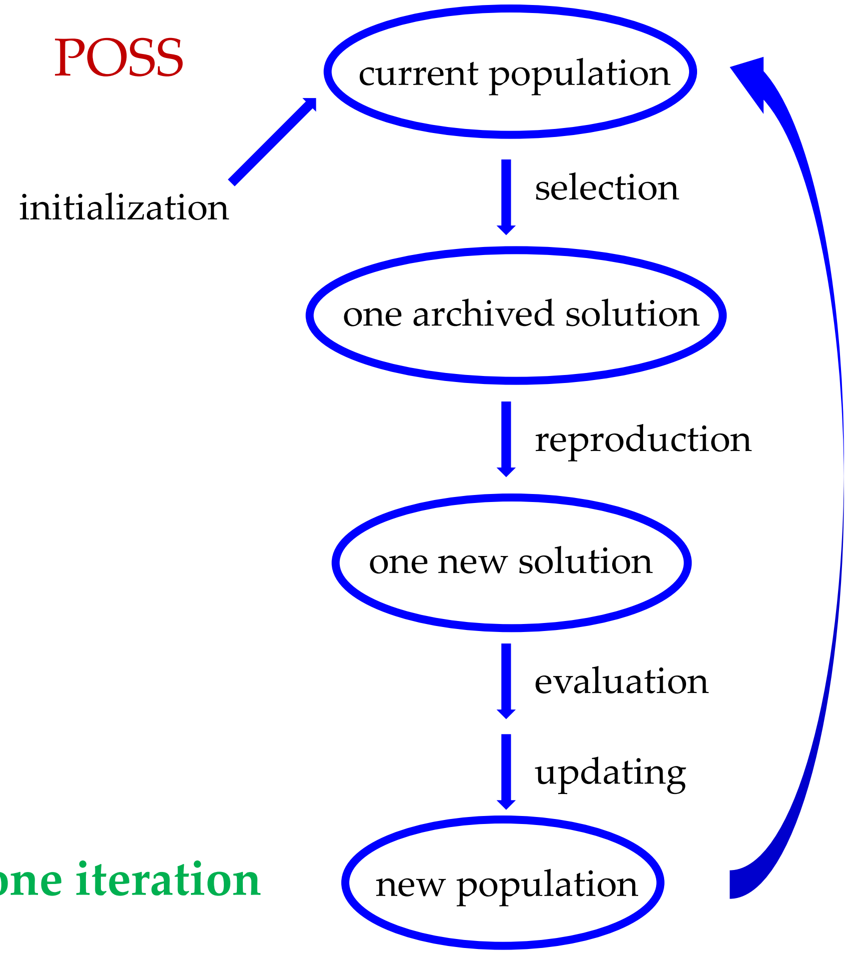
significantly better than all the compared methods on all data sets

Case Study (2)

Algorithm 1 POSS

Input: all variables $V = \{X_1, \ldots, X_n\}$, a given objective f and an integer parameter $k \in [1, n]$
Parameter: the number of iterations T
Output: a subset of V with at most k variables
Process:

Let $s = \{0\}^n$ and $P = \{s\}$.
Let $t = 0$.
while $t < T$ **do**
  Select s from P uniformly at random.
  Generate s' by flipping each bit of s with prob. $\frac{1}{n}$.
  Evaluate $f_1(s')$ and $f_2(s')$.
  if $\nexists z \in P$ such that $z \prec s'$ **then**
    $Q = \{z \in P \mid s' \preceq z\}$.
    $P = (P \setminus Q) \cup \{s'\}$.
  end if
  $t = t + 1$.
end while
return $\arg\min_{s \in P, |s| \leq k} f_1(s)$



generate only one new solution in one iteration

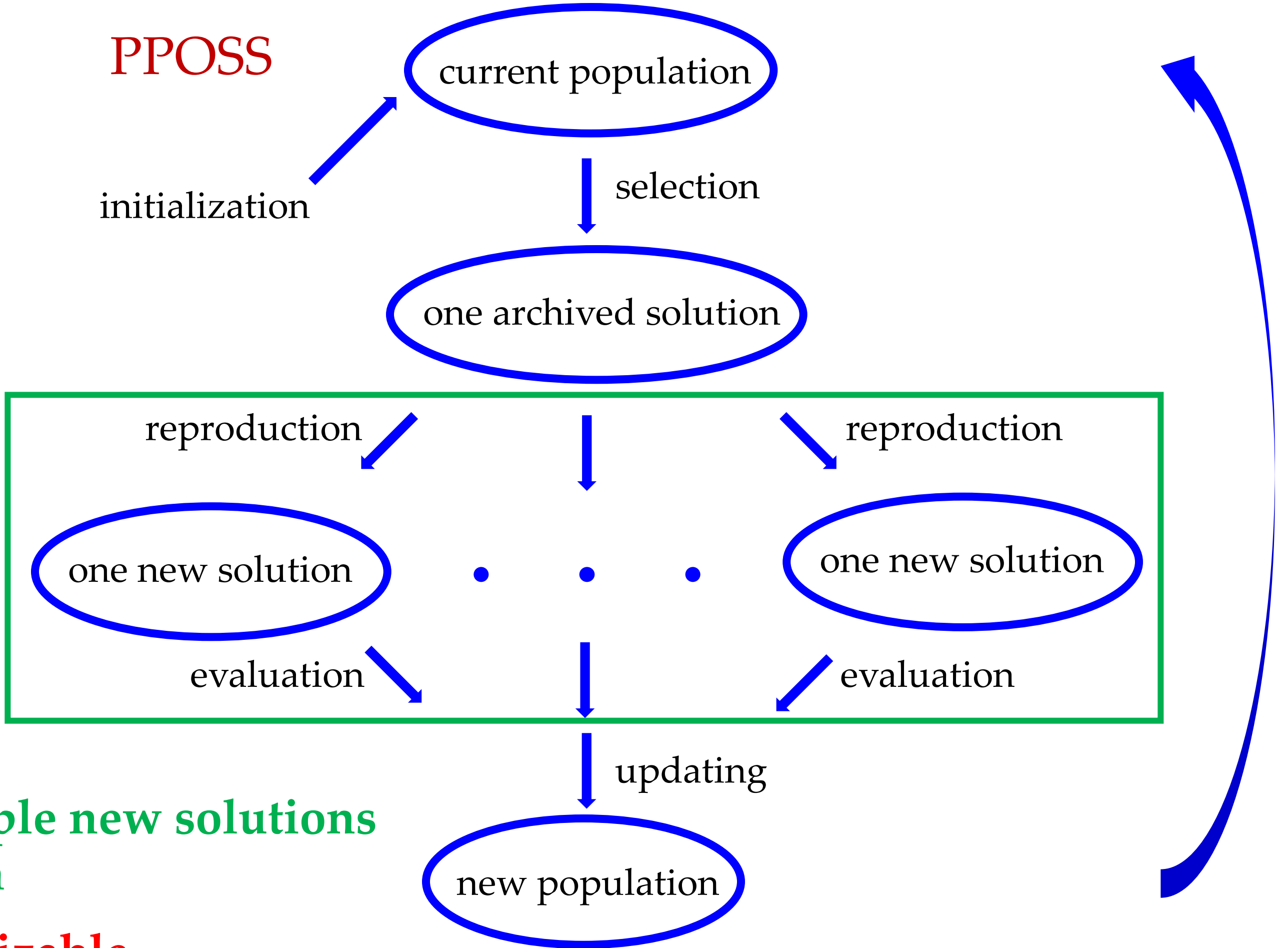
sequential

A sequential algorithm that cannot be readily parallelized

restrict the application to large-scale real-world problems

Case Study (2)

Theoretical guarantee still holds?



generate multiple new solutions in one iteration

parallelizable

Chao Qian, Jing-Cheng Shi, Yang Yu, Ke Tang, and Zhi-Hua Zhou. Parallel Pareto Optimization for Subset Selection. In: Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI'16), New York, NY, 2016, pp.1939-1945.

Case Study (2)

- Theoretical results

$f(S_2) \geq f(S_1)$ for any $S_1 \subseteq S_2$

Theorem 1. For maximizing a $\boxed{\text{monotone}}$ function under the set size constraint, the expected number of iterations until PPOSS finds a solution s with $|s| \leq k$ and $\boxed{f(s) \geq (1 - e^{-\gamma_{\min}}) \cdot OPT,}$ where $\gamma_{\min} = \min_{s:|s|=k-1} \gamma_{s,k}$, is

(1) if $N = o(n)$, then $\mathbb{E}[T] \leq 2ek^2 n/N$;

(2) if $N = \Omega(n^i)$ for $1 \leq i \leq k$, then $\mathbb{E}[T] = O(k^2/i)$;

(3) if $N = \Omega(n^{\min\{3k-1,n\}})$, then $\mathbb{E}[T] = O(1)$.

the same approximation bound

- When the number of processors is less than the number of variables, the number of iterations can be reduced linearly w.r.t. the number of processors
- With increasing number of processors, the number of iterations can be continuously reduced, eventually to a constant

The best previous known bound

- submodular [Nemhauser & Wolsey, MOR'78]
- sparse regression (non-submodular) [Das & Kempe, ICML'11]

Case Study (2)

Sparse regression

$$\arg\min_{S\subseteq V}\ MSE_{Z,S}\quad s.t.\quad |S|\le k$$

$$R^2_{Z,S} = (Var(Z) - MSE_{Z,S})/Var(Z)$$

the larger the better

data set

data set	#inst	#feat	data set	#inst	#feat	data set	#inst	#feat
housing	506	13	*sonar*	208	60	*clean1*	476	166
eunite2001	367	16	*triazines*	186	60	*w5a*	9888	300
svmguide3	1284	21	*coil2000*	9000	86	*gisette*	7000	5000
ionosphere	351	34	*mushrooms*	8124	112	*farm-ads*	4143	54877

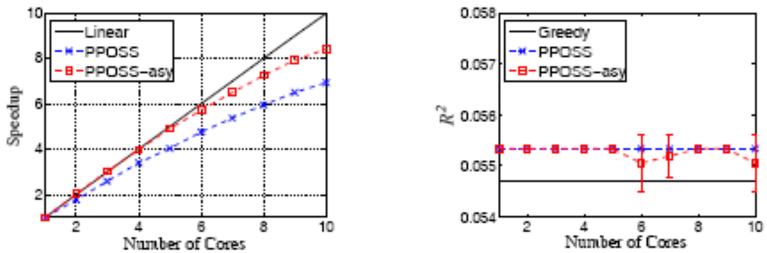
the sparsity $k = 8$

the number of cores N = 1 → 10

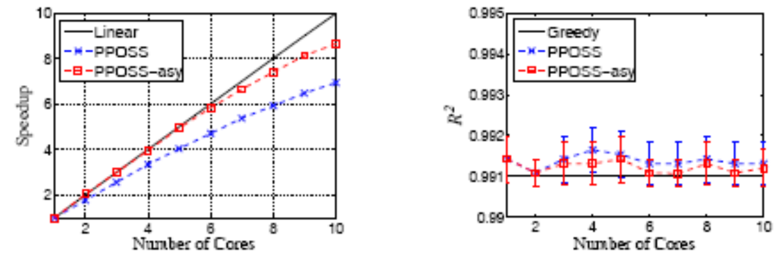
For PPOSS with each N value on each data set, the run is repeated for 10 runs independently, and the average results are reported

Case Study (2)

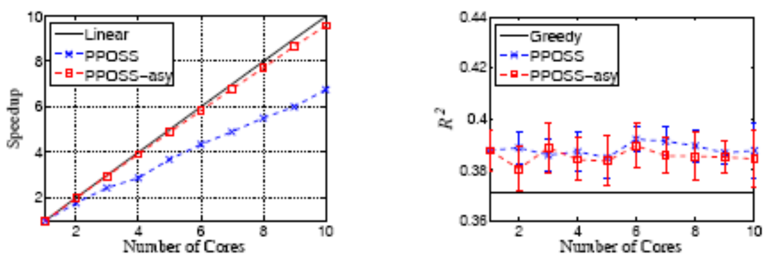
Compare the speedup as well as the solution quality measured by R^2 values with different number of cores



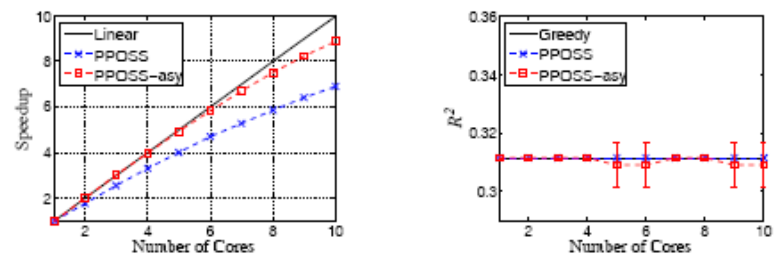
(a) on *coil2000* (9000 instances, 86 features)



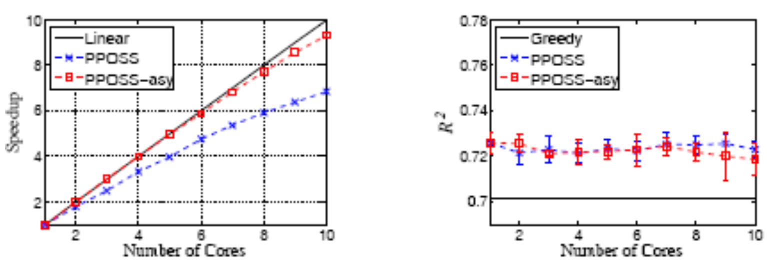
(b) on *mushrooms* (8124 instances, 112 features)



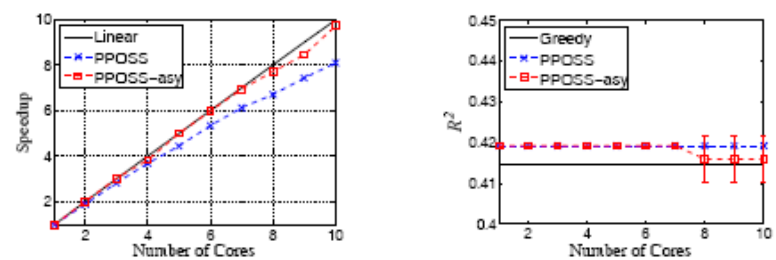
(c) on *clean1* (476 instances, 166 features)



(d) on *w5a* (9888 instances, 300 features)



(e) on *gisette* (7000 instances, 5000 features)

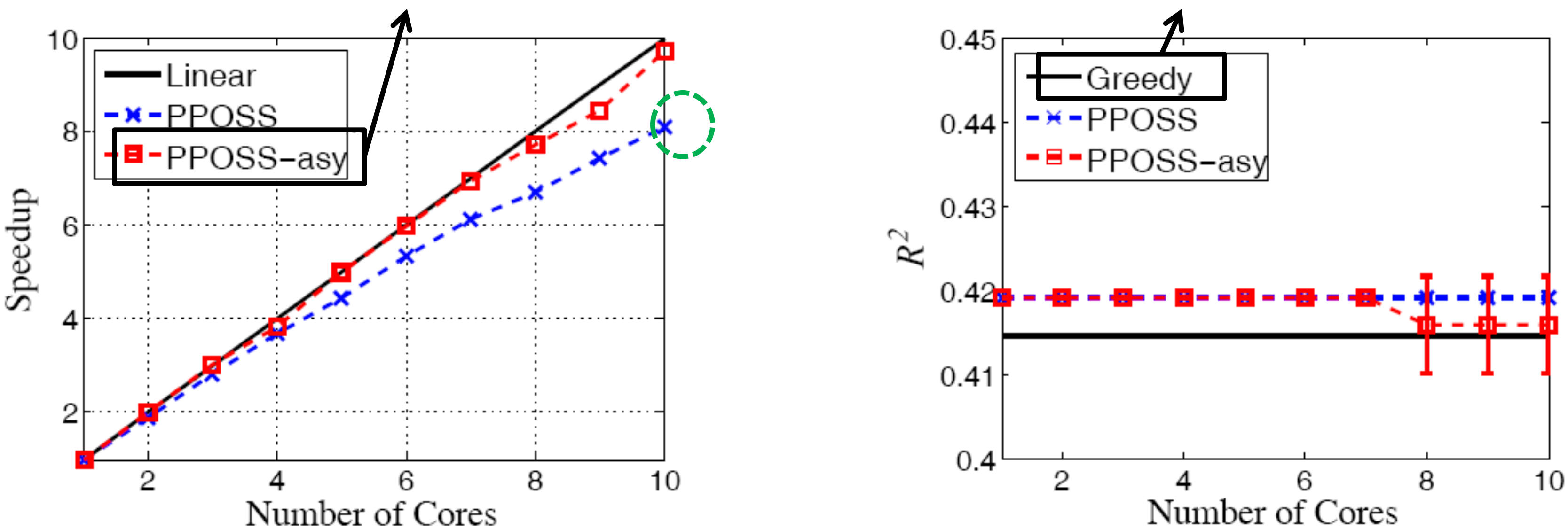


(f) on *farm-ads* (4143 instances, 54877 features)

Case Study (2)

the asynchronous version of PPOSS

the best previous algorithm [Das & Kempe, ICML'11]



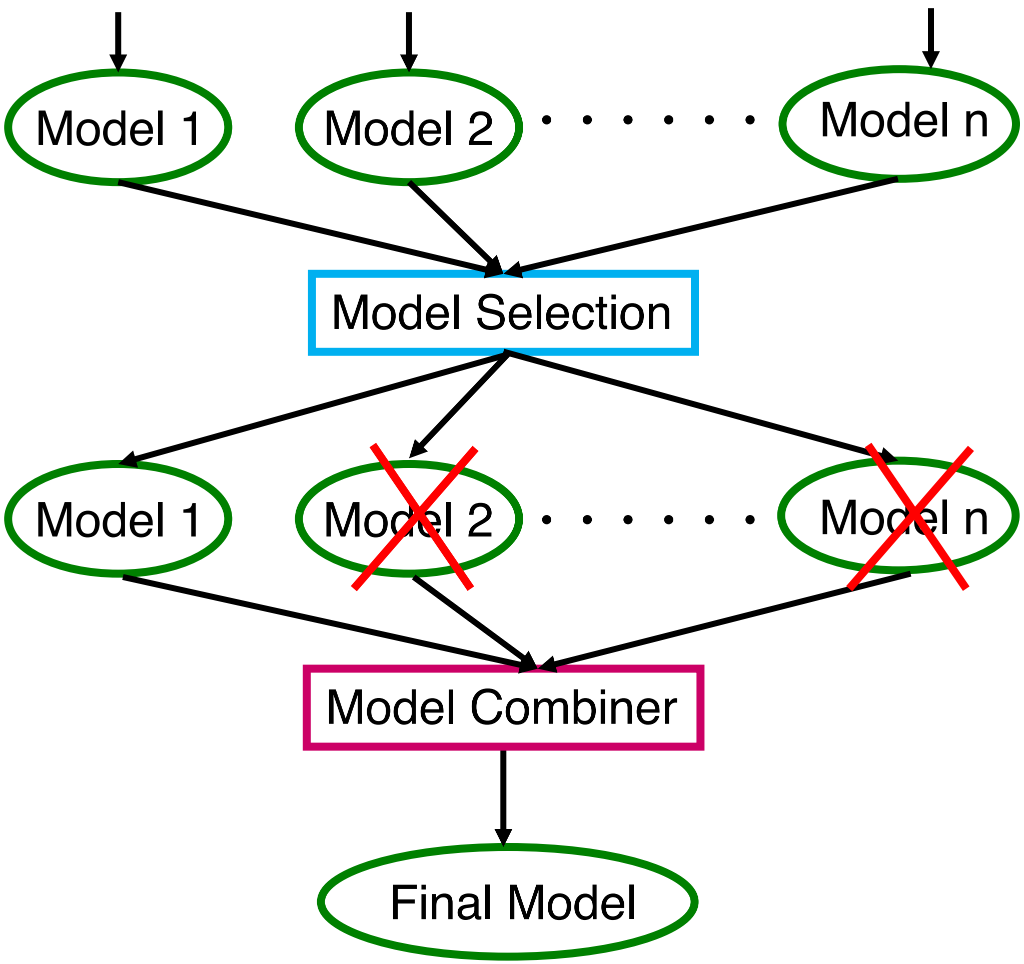
(f) on *farm-ads* (4143 instances, 54877 features)

PPOSS (blue line): achieve speedup around 8 when the number of cores is 10; the R^2 values are stable

PPOSS-asy (red line): achieve better speedup (avoid the synchronous cost); **the R^2 values are slightly worse** (the noise from asynchronization)

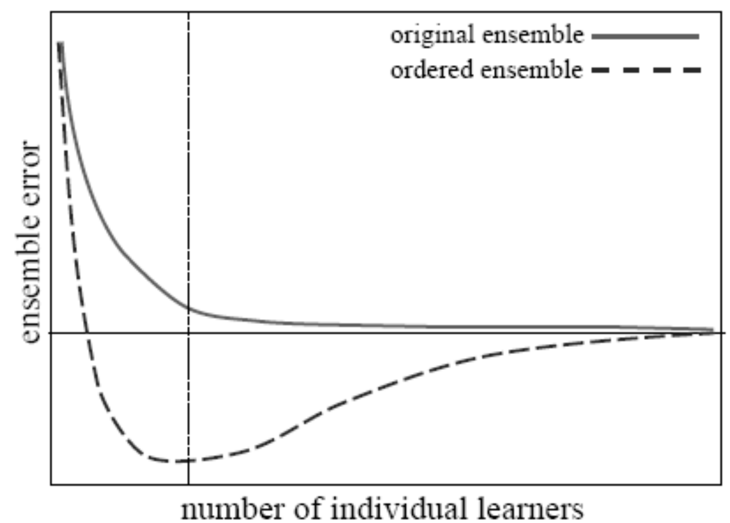
Case Study (3)

- Ensemble Pruning is also a subset selection problem.



Ensemble pruning: select a subset of learners to combine

- reduce storage & improve efficiency
- better performance than the complete ensemble



[Martinez-Munoz and Suarez, ICML'06]

Case Study (3)

- PEP: Pareto Ensemble Pruning

	Test Error								
Data set	PEP	Bagging	BI	RE	Kappa	CP	MD	DREP	EA
australian	.144±.020	**.143±.017**	.152±.023•	.144±.020	**.143±.021**	.145±.022	.148±.022	.144±.019	**.143±.020**
breast-cancer	**.275±.041**	.279±.037	.298±.044•	.277±.031	.287±.037	.282±.043	.295±.044•	**.275±.036**	**.275±.032**
disorders	**.304±.039**	.327±.047•	.365±.047•	.320±.044•	.326±.042•	.306±.039	.337±.035•	.316±.045	.317±.046•
heart-statlog	.197±.037	.195±.038	.235±.049•	**.187±.044**	.201±.038	.199±.044	.226±.048•	.194±.044	.196±.032
house-votes	.045±.019	**.041±.013**	.047±.016	.043±.018	.044±.017	.045±.017	.048±.018•	.045±.017	**.041±.012**
ionosphere	.088±.021	.092±.025	.117±.022•	.086±.021	**.084±.020**	.089±.021	.100±.026•	.085±.021	.093±.026
kr-vs-kp	**.010±.003**	.015±.007•	.011±.004	**.010±.004**	**.010±.003**	.011±.003	.011±.005	.011±.003	.012±.004
letter-ah	.013±.005	.021±.006•	.023±.008•	.015±.006•	**.012±.006**	.015±.006	.017±.007•	.014±.005	.017±.006•
letter-br	**.046±.008**	.059±.013•	.078±.012•	.048±.012	.048±.014	.048±.012	.057±.014•	.048±.009	.053±.011•
letter-oq	.043±.009	.049±.012•	.078±.017•	.046±.011	.042±.011	.042±.010	.046±.011	**.041±.010**	.044±.011
optdigits	**.035±.006**	.038±.007•	.095±.008•	.036±.006	**.035±.005**	.036±.005	.037±.006•	**.035±.006**	**.035±.006**
satimage-12v57	**.028±.004**	.029±.004	.052±.006•	.029±.004	**.028±.004**	.029±.004	.029±.004	.029±.004	.029±.004
satimage-2v5	**.021±.007**	.023±.009	.033±.010•	.023±.007	.022±.007	**.021±.008**	.026±.010•	.022±.008	**.021±.008**
sick	**.015±.003**	.018±.004•	.018±.004•	.016±.003	.017±.003•	.016±.003•	.017±.003•	.016±.003	.017±.004•
sonar	**.248±.056**	.266±.052	.310±.051•	.267±.053•	.249±.059	.250±.048	.268±.055•	.257±.056	.251±.041
spambase	**.065±.006**	.068±.007•	.093±.008•	.066±.006	.066±.006	.066±.006	.068±.007•	**.065±.006**	.066±.006
tic-tac-toe	.131±.027	.164±.028•	.212±.028•	.135±.026	.132±.023	.132±.026	.145±.022•	**.129±.026**	.138±.020
vehicle-bo-vs	**.224±.023**	.228±.026	.257±.025•	.226±.022	.233±.024•	.234±.024•	.244±.024•	.234±.026•	.230±.024
vehicle-b-v	**.018±.011**	.027±.014•	.024±.013•	.020±.011	.019±.012	.020±.011	.021±.011•	.019±.013	.026±.013•
vote	.044±.018	.047±.018	.046±.016	.044±.017	**.041±.016**	.043±.016	.045±.014	.043±.019	.045±.015
count of the best	12	2	0	2	7	1	0	5	5
PEP: count of direct win		**17**	**20**	**15.5**	12.5	**17**	**20**	12.5	**15.5**

Better than any other method on more than 60% (12.5/20) data sets, and never significantly worse

Case Study (3)

Ensemble Size							
Data set	PEP	RE	Kappa	CP	MD	DREP	EA
australian	10.6±4.2	12.5±6.0	14.7±12.6	11.0±9.7	**8.5±14.8**	11.7±4.7	41.9±6.7•
breast-cancer	8.4±3.5	8.7±3.6	26.1±21.7•	8.8±12.3	**7.8±15.2**	9.2±3.7	44.6±6.6•
disorders	14.7±4.2	**13.9±4.2**	24.7±16.3•	15.3±10.6	17.7±20.0	**13.9±5.9**	42.0±6.2•
heart-statlog	**9.3±2.3**	11.4±5.0•	17.9±11.1•	13.2±8.2•	13.6±21.1	11.3±2.7•	44.2±5.1•
house-votes	**2.9±1.7**	3.9±4.0	5.5±3.3•	4.7±4.4•	5.9±14.1	4.1±2.7•	46.5±6.1•
ionosphere	**5.2±2.2**	7.9±5.7•	10.5±6.9•	8.5±6.3•	10.7±14.6•	8.4±4.3•	48.8±5.1•
kr-vs-kp	**4.2±1.8**	5.8±4.5	10.6±9.1•	9.6±8.6•	7.2±15.2	7.1±3.9•	45.9±5.8•
letter-ah	**5.0±1.9**	7.3±4.4•	7.1±3.8•	8.7±4.7•	11.0±10.9•	7.8±3.6•	42.5±6.5•
letter-br	**10.9±2.6**	15.1±7.3•	13.8±6.7•	12.9±6.8	23.2±17.6•	11.3±3.5	38.3±7.8•
letter-oq	**12.0±3.7**	13.6±5.8	13.9±6.0	12.3±4.9	23.0±15.6•	13.7±4.9	39.3±8.2•
optdigits	22.7±3.1	25.0±9.3	25.2±8.1	**21.4±7.5**	46.8±23.9•	25.0±8.0	41.4±7.6•
satimage-12v57	**17.1±5.0**	20.8±9.2•	22.1±10.3•	21.2±10.0•	37.6±24.3•	18.1±4.9	42.7±5.2•
satimage-2v5	**5.7±1.7**	6.8±3.2	7.6±4.2•	10.9±7.0•	26.2±28.1•	7.7±3.5•	44.1±4.8•
sick	**6.9±2.8**	7.5±3.9	10.9±6.0•	11.5±10.0•	8.3±13.6	11.6±6.7•	44.7±8.2•
sonar	11.4±4.2	**11.0±4.1**	20.6±9.3•	13.9±7.1	20.6±20.7•	14.4±5.9•	43.1±6.4•
spambase	17.5±4.5	18.5±5.0	20.0±8.1	19.0±9.9	28.8±17.0•	**16.7±4.6**	39.7±6.4•
tic-tac-toe	14.5±3.8	16.1±5.4	17.4±6.5	15.4±6.3	28.0±22.6•	**13.6±3.4**	39.8±8.2•
vehicle-bo-vs	16.5±4.5	15.7±5.7	16.5±8.2	$\boxed{\mathbf{11.2\pm5.7}\circ}$	21.6±20.4	$\boxed{13.2\pm5.0\circ}$	41.9±5.6•
vehicle-b-v	**2.8±1.1**	3.4±2.1	4.5±1.6•	5.3±7.4	**2.8±3.8**	4.0±3.9	48.0±5.6•
vote	**2.7±1.1**	3.2±2.7	5.1±2.6•	5.4±5.2•	6.0±9.8	3.9±2.5•	47.8±6.1•
count of the best	12	2	0	2	3	3	0
PEP: count of direct win		17	19.5	18	17.5	16	20

Better than any other method on more than 80% (16/20) data sets; never significantly worse, except two losses

Chao Qian, Yang Yu, and Zhi-Hua Zhou. Pareto Ensemble Pruning. In: Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI'15), Austin, TX, 2015, pp.2935-2941.

Case Study (4)

- Many real-world classification problems are cost-sensitive.

- The optimal classifier for a binary classification problem

$$h_{opt} = \underset{h \in \Omega}{\operatorname{argmin}}\ C_{10} \cdot FNR(h) + C_{01} \cdot FPR(h) \qquad (1)$$

Cost matrix

	Predicted +	Predicted -
+	0	C_{10}
-	C_{01}	0

Confusion matrix

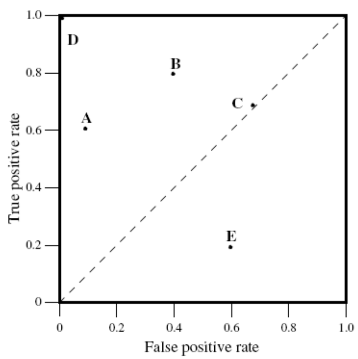
	Predicted +	Predicted -
+	TPR	FNR
-	FPR	TNR

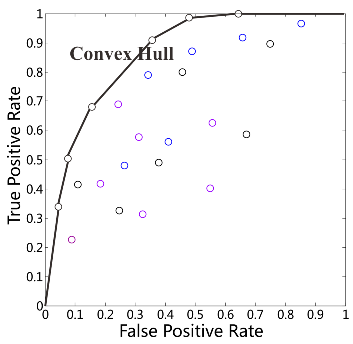
- A good classifier could be used via *cost-sensitive learning*.

Case Study (4)

- However, costs are often subject to great uncertainty.
  - Very difficult to specify the exact cost values *before training*.
  - The costs may change over time.

- Alternative: seeking a group of classifiers that maximize the the Receiver Operating Characteristic (ROC) convex hull.





"The optimal classifier for any cost values must be a vertex or on the edge of the convex hull of all (achievable) classifiers in the ROC space." [Provost and Fawcett, 2001]

Case Study (4)

- New Learning Target: To obtain a set of classifiers such that their ROCCH is maximized.

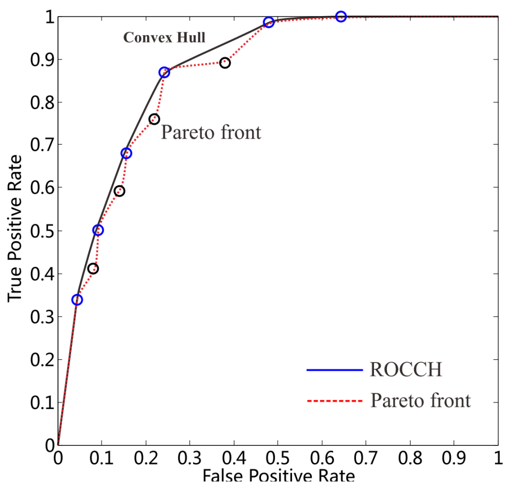
- This is a set-oriented optimization problem can could hardly be solved with existing approaches.

- Evolutionary Algorithms provides a natural way to search for a set (population) of classifiers.

Case Study (4)

- Multi-objective evolutionary algorithms (MOEAs) are off-the-shelf tools for this problem
  - Maximize TPR
  - Minimize FPR

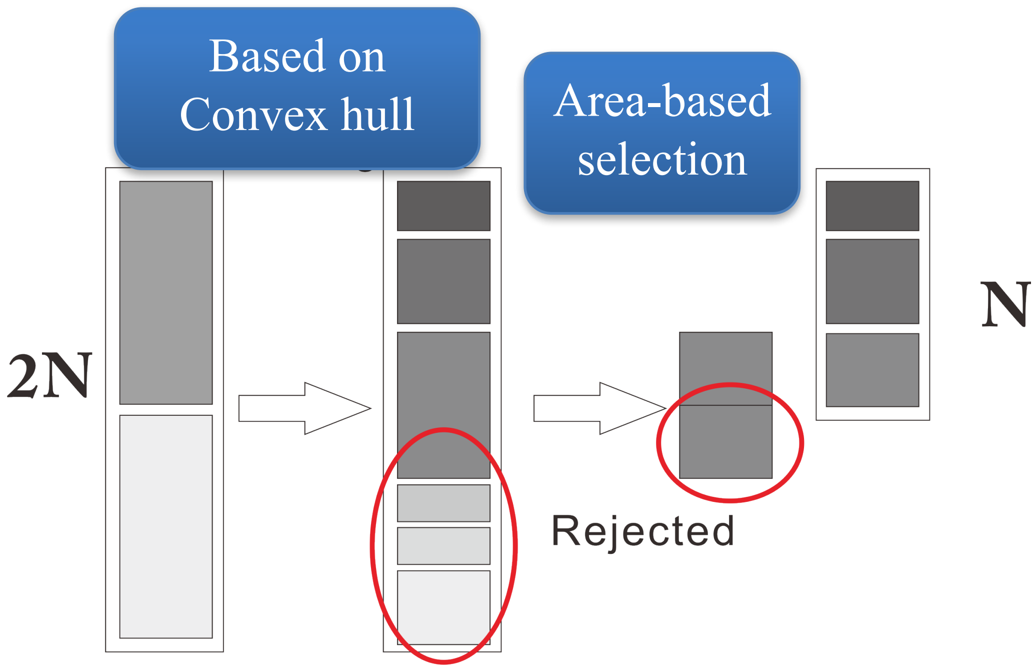
- Direct application of an MOEA is OK, while not ideal.
  - A Pareto optimal solution is not necessarily a vertex on the convex hull.
  - Many-to-one mapping between the hypothesis and ROC spaces.



Case Study (4)

- Approach: Convex Hull-based MOEA (CH-MOEA)

- Features of CH-MOEA:
  - Convex hull-based sorting
  - Redundancy elimination



Case Study (4)

- Convex Hull-based Sorting

Algorithm 2 *DeltaArea* (T)

Require: $T \neq \emptyset$

T is a solution set

Ensure: DeltaArea

$Q = T$

$m = sizeof(Q)$

E is performance of Q

$\mathbf{DeltaH}_1, ..., \mathbf{DeltaH}_m \leftarrow 0$

if $m < 3$ **then**

  Set $\mathbf{DeltaH}_1, ..., \mathbf{DeltaH}_m \leftarrow \infty$

else

  Set $\mathbf{DeltaH}_1, \mathbf{DeltaH}_m \leftarrow \infty$

  for $2 \leq i \leq sizeof(Q) - 1$ **do**

    $\mathbf{DeltaH}_i = 0.5 \times \det((\mathbf{E}_i\text{-}\mathbf{E}_{i-1}) \circ (\mathbf{E}_{i+1}\text{-}\mathbf{E}_{i-1}))$

  end for

  while $sizeof(Q) > 2$ **do**

    $r \leftarrow argmin\{\mathbf{DeltaH}\}$

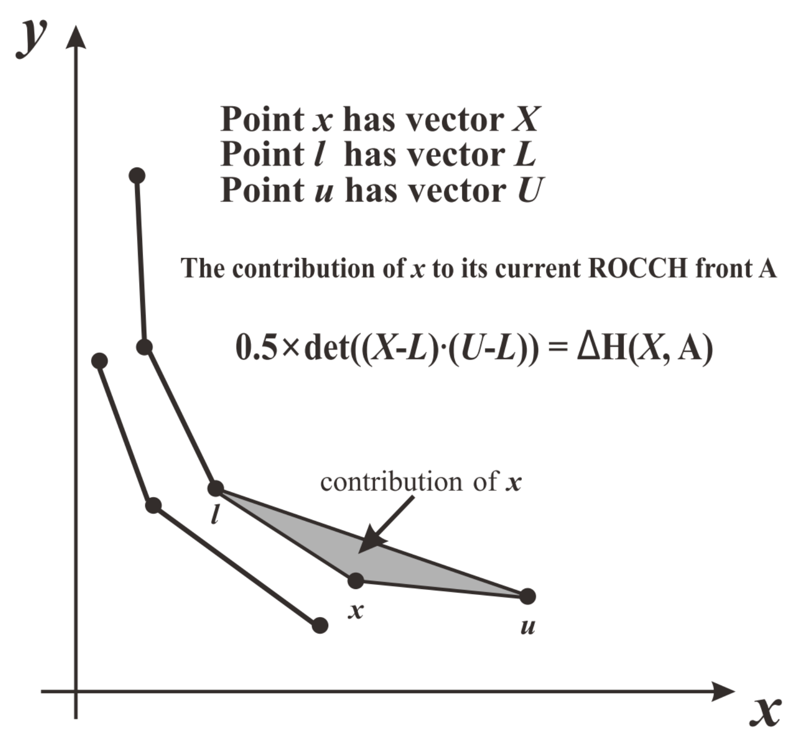
    $Q \leftarrow Q \setminus \{Q_r\}$

    Update($\mathbf{DeltaH}_{r-1}$,$\mathbf{DeltaH}_{r+1}$)

  end while

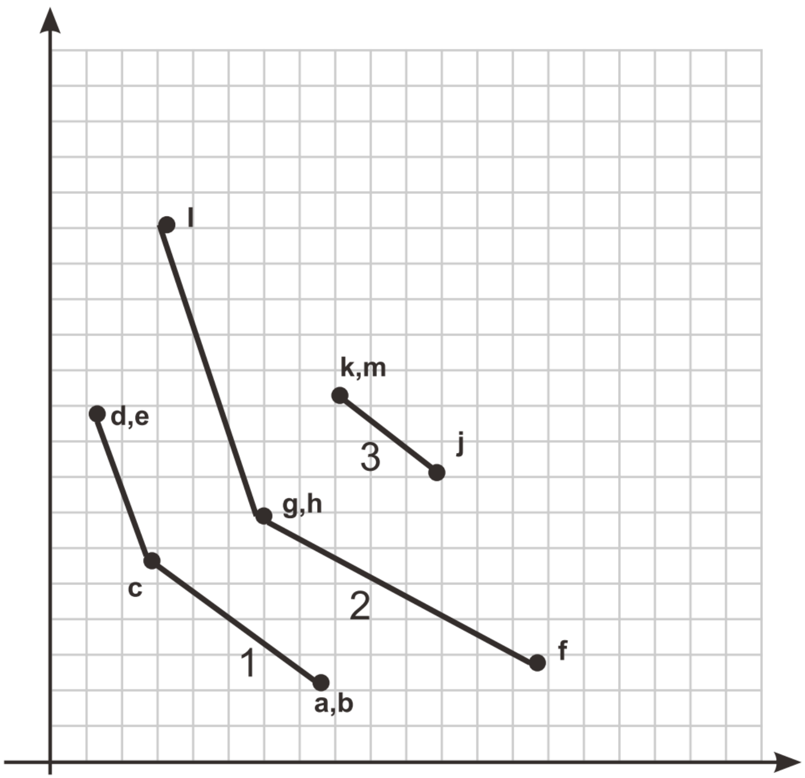
end if

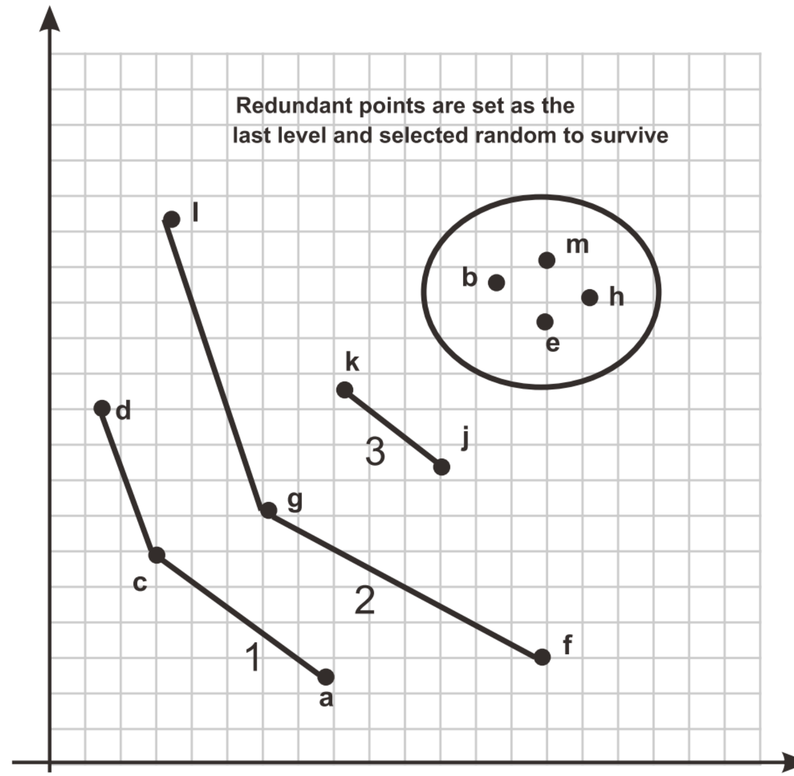
Return (**DeltaH**)



Case Study (4)

- Redundancy Elimination





Case Study (4)

- CH-MOEA can be combined with any base learners to build either homogeneous and heterogeneous ensembles
  - Neural Network
  - Decision Tree
  - SVM
  - …
- Different types of base learners need different search operators.
- We implemented CH-MOEA with Genetic Programming (CH-MOGP).

P. Wang, M. Emmerich, R. Li, K. Tang, T. Baeck and X. Yao, "Convex Hull-Based Multi-objective Genetic Programming for Maximizing Receiver Operating Characteristic Performance," IEEE Transactions on Evolutionary Computation, 19(2): 188-200, April 2015.

Case Study (4)

- Empirical studies
  - Which MOEA framework performs the best for our problem?
  - Is CH-MOGP competitive in comparison to non-evolutionary methods?

- Compared methods
  - NSGA-II
  - MOEA/D
  - SMS-EMOA (an indicator based MOEA)
  - C4.5
  - PRIE ([Fawcett, 2008], a state-of-the-art heuristic approach for ROCCH maximization)
  - Naïve Bayes

All evolutionary approaches adopt the same base learner and reproduction operator

Case Study (4)

- CH-MOGP outperformed state-of-the-art MOEAs

Performance on Ninteen UCI Datasets

Dataset	CH-MOGP	SMS-EMOA	NSGA-II	MOEA/D	Dataset	CH-MOGP	SMS-EMOA	NSGA-II	MOEA/D
australian	91.49 ± 2.72	91.67 ± 2.48	91.16 ± 2.41	90.29 ± 2.75	*bands*	77.00 ± 4.05	76.38 ± 4.09	75.54 ± 3.56	71.85 ± 3.82
bcw	97.94 ± 1.20	97.73 ± 1.56	97.84 ± 1.41	97.48 ± 1.48	*crx*	91.30 ± 2.45	91.16 ± 2.33	91.14 ± 2.36	89.88 ± 2.51
german	73.10 ± 3.24	73.32 ± 3.33	72.39 ± 3.07	71.45 ± 2.85	*house-votes*	97.94 ± 1.56	97.69 ± 1.59	97.74 ± 1.71	97.15 ± 1.75
ionosphere	91.07 ± 4.95	90.51 ± 4.52	90.45 ± 4.53	89.89 ± 4.83	*kr-vs-kp*	98.40 ± 0.89	98.63 ± 0.75	98.39 ± 0.79	96.67 ± 1.43
mammographic	89.75 ± 2.01	89.48 ± 1.94	89.41 ± 1.87	87.50 ± 2.23	*monks-1*	99.70 ± 1.68	97.62 ± 3.71	99.62 ± 1.35	96.51 ± 5.69
monks-2	91.05 ± 8.00	89.28 ± 5.58	90.53 ± 5.19	73.26 ± 9.14	*monks-3*	99.81 ± 0.43	99.74 ± 0.45	99.45 ± 2.87	99.07 ± 0.88
parkinsons	86.79 ± 6.86	85.11 ± 6.68	84.90 ± 7.54	83.94 ± 6.72	*pima*	80.08 ± 3.38	79.85 ± 3.38	79.29 ± 3.70	76.93 ± 3.10
sonar	79.42 ± 5.87	78.04 ± 5.91	77.79 ± 7.34	75.75 ± 5.66	*spect*	77.38 ± 7.36	76.27 ± 7.14	76.91 ± 8.46	74.88 ± 6.43
tic-tac-toe	83.40 ± 10.4	79.56 ± 11.1	79.07 ± 13.4	70.85 ± 10.4	*transfusion*	71.62 ± 4.62	71.48 ± 4.47	71.49 ± 4.84	68.77 ± 4.63
wdbc	96.78 ± 1.92	96.49 ± 2.25	96.70 ± 2.11	95.90 ± 2.19					

Performance on Three Large-scaled UCI Datasets

Dataset	CH-MOGP	SMS-EMOA	NSGA-II	MOEA/D	Dataset	CH-MOGP	SMS-EMOA	NSGA-II	MOEA/D
adult	84.58 ± 1.40	82.53 ± 2.15	84.01 ± 1.38	77.04 ± 2.54	*magic04*	83.02 ± 1.04	81.76 ± 1.57	82.01 ± 1.19	76.39 ± 3.07
skin	97.10 ± 1.11	95.46 ± 1.85	96.57 ± 1.25	93.20 ± 2.37					

Results for 19 UCI Dataset

Ratio	$\frac{1}{15}$	$\frac{1}{10}$	$\frac{1}{4}$	$\frac{1}{3}$	$\frac{1}{2}$	$\frac{2}{3}$	1
NSGA-II	4-15-0	4-15-0	2-17-0	4-15-0	5-14-0	5-14-0	4-15-0
SMS-EMOA	11-8-0	11-8-0	6-13-0	5-14-0	4-15-0	4-15-0	5-14-0
MOEA/D	19-0-0	19-0-0	19-0-0	19-0-0	19-0-0	19-0-0	19-0-0

Results for 3 Large-Scaled UCI Dataset

Ratio	$\frac{1}{15}$	$\frac{1}{10}$	$\frac{1}{4}$	$\frac{1}{3}$	$\frac{1}{2}$	$\frac{2}{3}$	1
NSGA-II	0-3-0	1-2-0	1-2-0	1-2-0	2-1-0	2-1-0	2-1-0
SMS-EMOA	3-0-0	3-0-0	3-0-0	3-0-0	3-0-0	3-0-0	3-0-0
MOEA/D	3-0-0	3-0-0	3-0-0	3-0-0	3-0-0	3-0-0	3-0-0

Case Study (4)

Dataset	CH-MOGP	C4.5	NB	PRIE
australian	91.97 ± 2.53	85.52 ± 4.05	89.47 ± 2.78	91.75 ± 2.36
bands	78.50 ± 3.56	74.56 ± 4.59	73.91 ± 4.68	76.07 ± 4.81
bcw	98.17 ± 1.06	95.05 ± 2.55	98.92 ± 0.62	98.16 ± 1.09
crx	91.82 ± 2.27	85.51 ± 3.94	87.88 ± 3.16	90.65 ± 2.77
german	74.27 ± 2.79	65.36 ± 4.74	78.42 ± 2.94	75.95 ± 3.25
house-votes	98.23 ± 1.26	96.35 ± 2.04	98.05 ± 1.04	97.80 ± 1.49
ionosphere	92.42 ± 3.66	88.20 ± 5.65	93.57 ± 3.18	93.68 ± 4.23
kr-vs-kp	99.40 ± 0.26	99.71 ± 0.23	93.21 ± 1.00	98.26 ± 0.44
mammographic	90.20 ± 1.76	87.66 ± 2.21	89.77 ± 1.96	89.70 ± 2.02
monks-1	100.0 ± 0.00	77.13 ± 6.90	73.18 ± 4.58	70.93 ± 5.59
monks-2	95.68 ± 4.61	94.17 ± 5.93	52.38 ± 7.04	51.25 ± 6.16

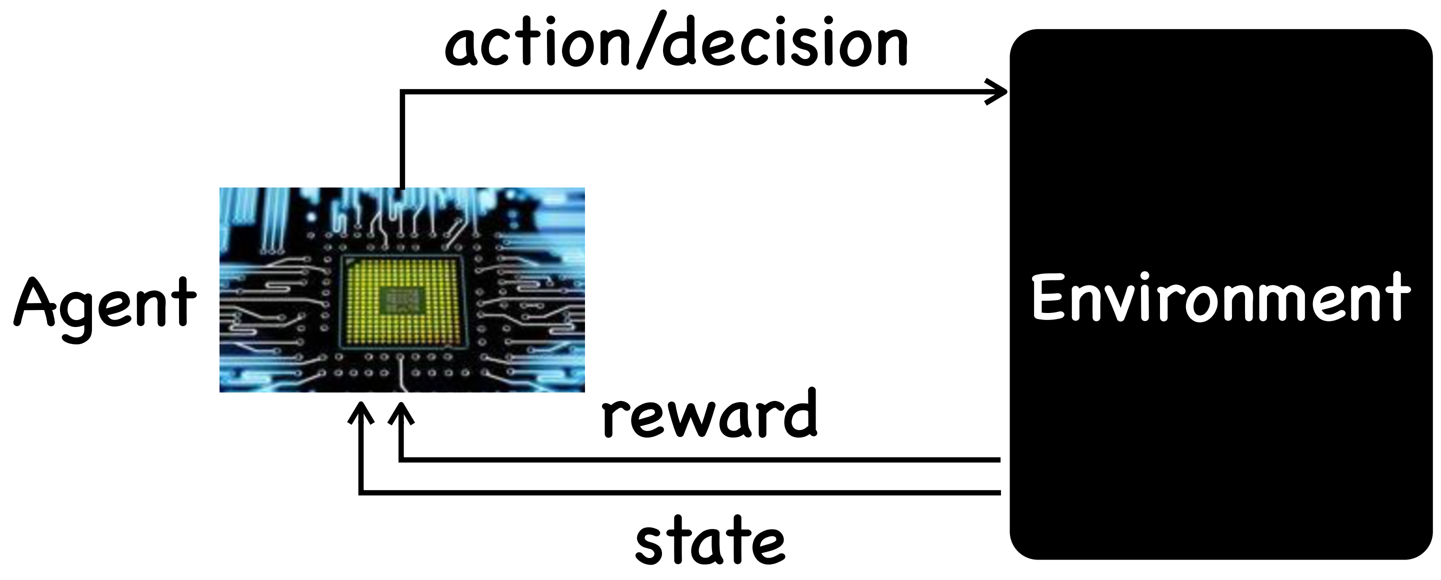
Dataset	CH-MOGP	C4.5	NB	PRIE
monks-3	100.0 ± 0.00	100.0 ± 0.00	95.94 ± 2.17	99.60 ± 0.27
parkinsons	86.10 ± 6.66	78.91 ± 9.76	85.91 ± 6.11	88.24 ± 5.83
pima	80.74 ± 3.12	75.23 ± 4.93	81.40 ± 3.01	79.58 ± 2.92
sonar	81.44 ± 5.15	73.85 ± 7.84	80.12 ± 7.03	69.92 ± 8.64
spect	78.56 ± 7.44	76.88 ± 8.91	84.09 ± 6.03	83.51 ± 7.01
tic-tac-toe	90.07 ± 8.88	84.91 ± 13.9	61.50 ± 14.7	70.41 ± 12.5
transfusion	72.19 ± 4.89	71.08 ± 5.08	70.93 ± 4.94	70.87 ± 5.39
wdbc	97.32 ± 1.40	92.74 ± 3.16	98.14 ± 1.33	96.58 ± 1.94
adult	88.97 ± 0.37	88.89 ± 0.53	85.27 ± 0.37	90.37 ± 0.25
magic04	87.16 ± 0.74	86.76 ± 0.83	75.70 ± 0.74	85.37 ± 0.76
skin	99.49 ± 0.11	99.93 ± 0.02	94.17 ± 0.07	98.15 ± 0.08

CH-MOEA outperformed other state-of-the-art methods in terms of solution quality

Evolutionary Reinforcement Learning

What is reinforcement learning

learning a strategy to interact with the environment for maximizing the long-term reward



Agent: Policy: $\pi : S \times A \to \mathbb{R}, \quad \sum_{a \in A} \pi(a|s) = 1$

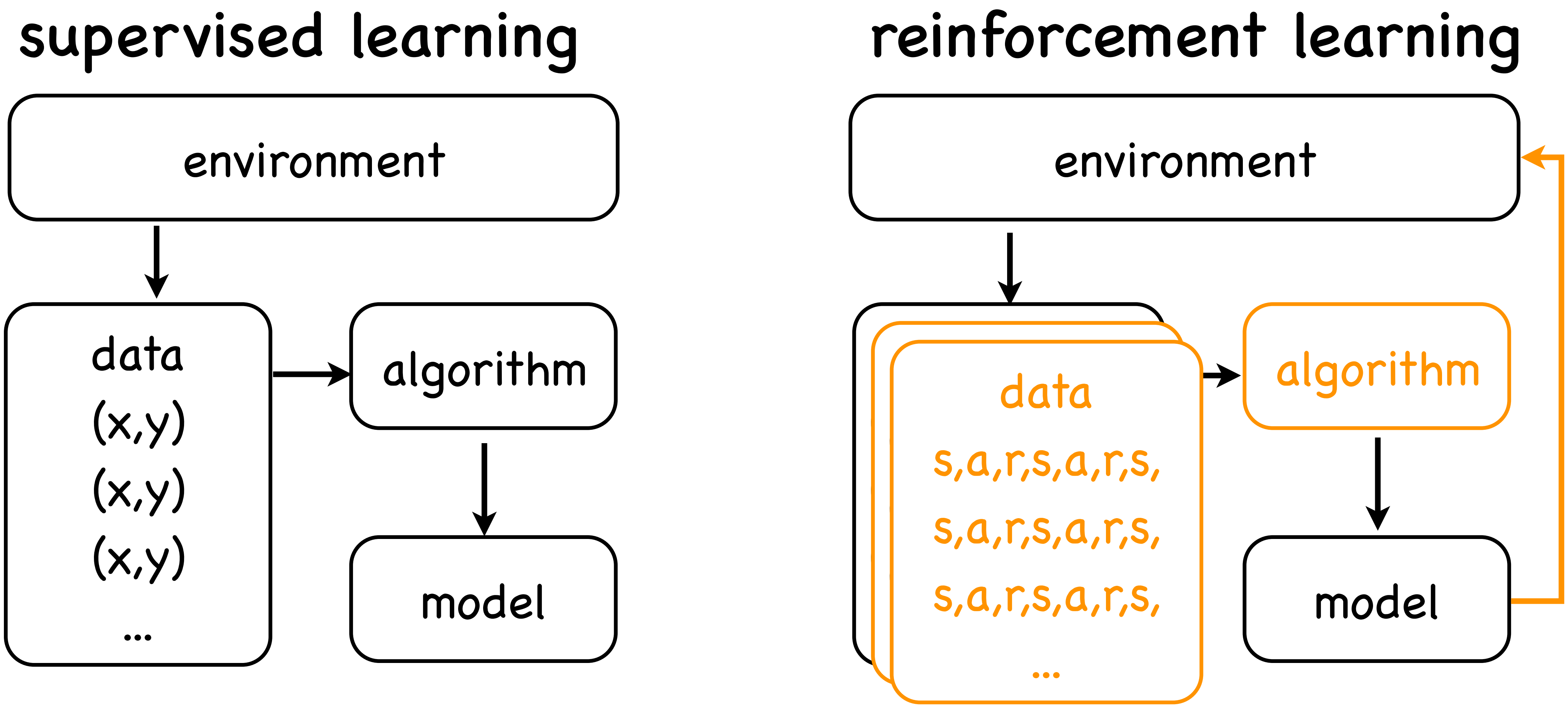
Policy (deterministic): $\pi : S \to A$

Agent's goal: learn a policy to maximize long-term total reward

T-step: $\sum_{t=1}^{T} r_t$

discounted: $\sum_{t=1}^{\infty} \gamma^t r_t$

Compare RL with SL

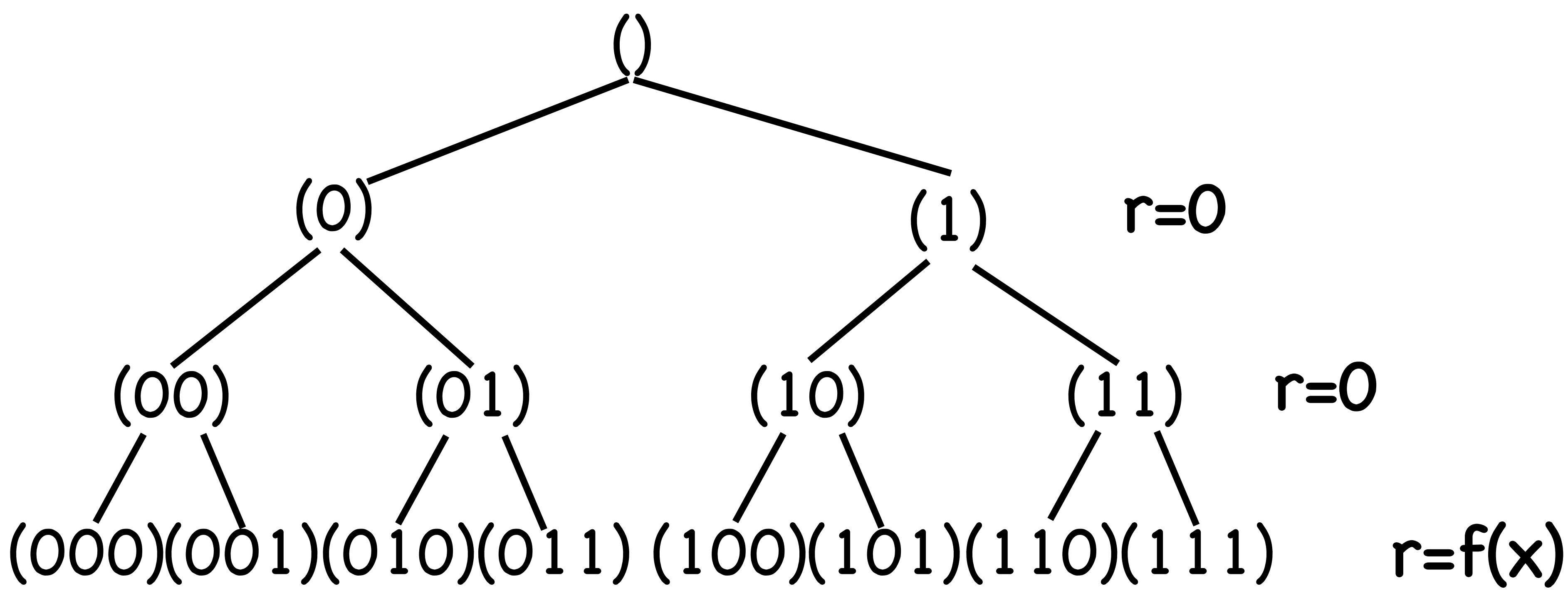


SL searches for a model

RL searches for the right output and a model

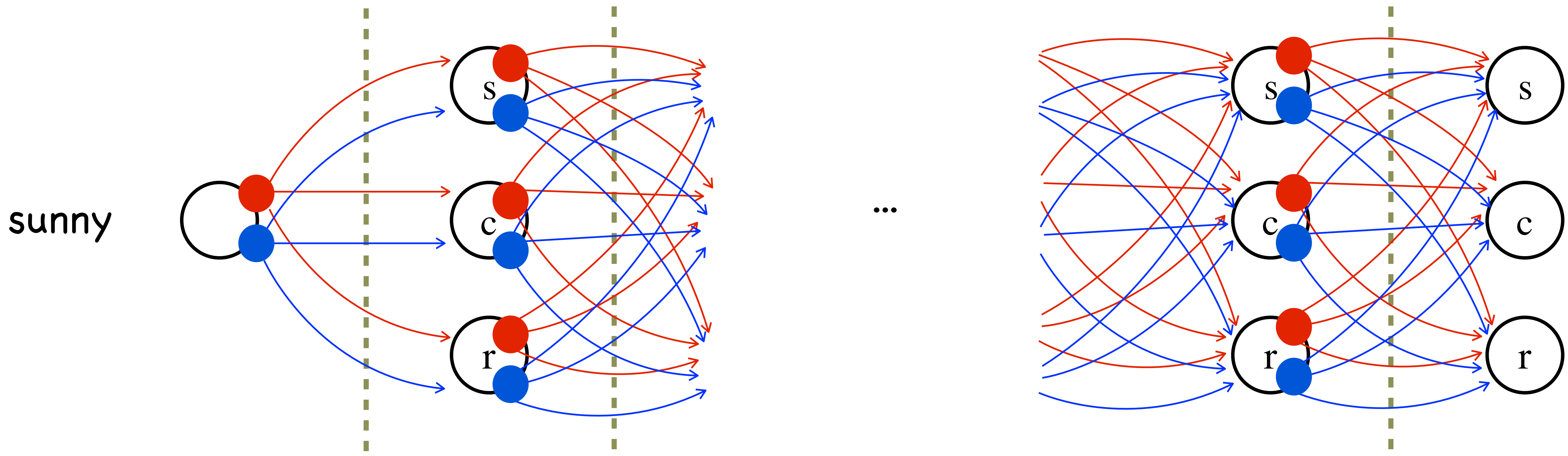
Hardness of RL

general binary space problem $\max_{x \in \{0,1\}^n} f(x)$



solving the optimal policy is NP-hard!

Value-based methods



dynamic programming

$$V^\pi(s) = \sum_a \pi(a|s) \sum_{s'} P(s'|s,a)\big(R(s,a,s') + V^\pi(s')\big)$$

$$Q^\pi(s,a) = \sum_{s'} P(s'|s,a)\big(R(s,a,s') + V^\pi(s')\big)$$

Value-based methods

overall idea:

how is the current policy — policy evaluation

improve the current policy — policy improvement

policy iteration:

policy evaluation: backward calculation

$$V^\pi(s) = \sum_a \pi(a|s) \sum_{s'} P(s'|s,a)\big(R(s,a,s') + \gamma V^\pi(s')\big)$$

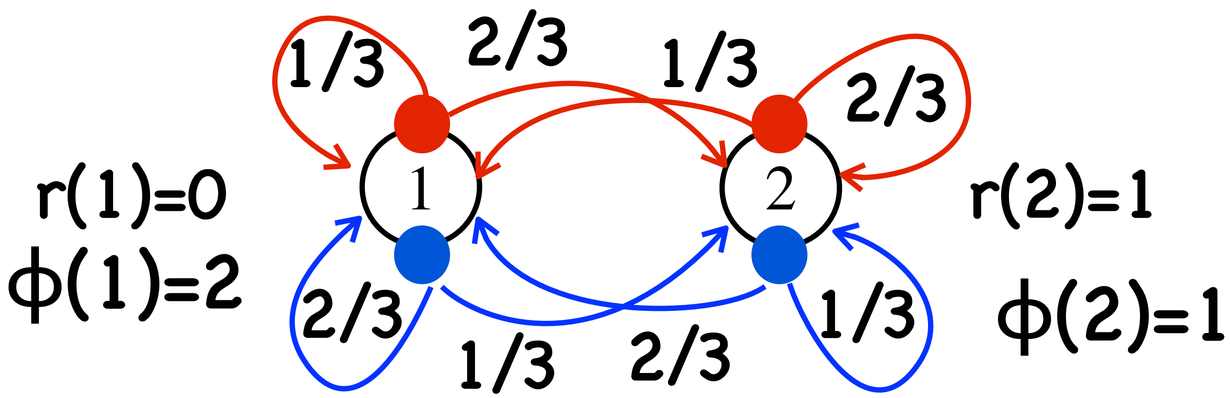
policy improvement: from the Bellman optimality equation

$$V(s) \leftarrow \max_a Q^\pi(s,a)$$

Value-based methods

policy degradation in value-based methods

[Bartlett. An Introduction to Reinforcement Learning Theory: Value Function Methods. Advanced Lectures on Machine Learning, LNAI 2600]



optimal policy: red

$V^*(2) > V^*(1) > 0$

let $\hat{V}(s) = w\phi(s)$, to ensure $\hat{V}(2) > \hat{V}(1)$, $w < 0$

as value function based method minimizes $\|\hat{V} - V^*\|$
results in $w > 0$

sub-optimal policy, better value $\neq$ better policy

Policy search

parameterized policy

$$\pi(a|s) = P(a|s, \theta)$$

Gibbs policy (logistic regression)

$$\pi_\theta(i|s) = \frac{\exp(\theta_i^\top \phi(s))}{\sum_j \exp(\theta_j^\top \phi(s))}$$

Gaussian policy (continuous !)

$$\pi_\theta(a|s) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(\theta^\top s - a)^2}{\sigma^2}\right)$$

Policy search

direct objective functions

episodic environments: over all trajectories

$$J(\theta) = \int_{Tra} p_\theta(\tau) R(\tau) \ \mathrm{d}\tau$$

where $p_\theta(\tau) = p(s_0) \prod_{i=1}^{T} p(s_i | a_i, s_{i-1}) \pi_\theta(a_i | s_{i-1})$

is the probability of generating the trajectory

continuing environments: over stationary distribution

$$J(\theta) = \int_S d^{\pi_\theta}(s) \int_A \pi_\theta(a|s) R(s,a) \ \mathrm{d}s \ \mathrm{d}a$$

d^{π_θ} is the stationary distribution of the process

Policy search by gradient: policy gradient

$$J(\theta) = \int_{Tra} p_\theta(\tau) R(\tau)\ \mathrm{d}\tau$$

logarithm trick $\nabla_\theta p_\theta = p_\theta \nabla_\theta \log p_\theta$

as $p_\theta(\tau) = p(s_0) \prod_{i=1}^{T} p(s_i | a_i, s_{i-1}) \pi_\theta(a_i | s_{i-1})$

$$\nabla_\theta \log p_\theta(\tau) = \sum_{i=1}^{T} \nabla_\theta \log \pi_\theta(a_i | s_{i-1}) + \mathrm{const}$$

gradient: $\nabla_\theta J(\theta) = \int_{Tra} p_\theta(\tau) \nabla_\theta \log p_\theta(\tau) R(\tau)\ \mathrm{d}\tau$

$$= E[\sum_{i=1}^{T} \nabla_\theta \log \pi_\theta(a_i | s_i) R(s_i, a_i)]$$

use samples to estimate the gradient (unbiased estimation)

Policy search v.s. value-based methods

Policy search advantages:

- effective in high-dimensional and continuous action space
- learn stochastic policies directly
- avoid policy degradation

disadvantages:

- converge only to a local optimum
- high variance

Policy gradient: variance control

actor-critic

$$\nabla_\theta J(\theta) \approx E[\nabla_\theta \log \pi_\theta(a|s) Q_w(s,a)]$$

$$\text{if } w \text{ is a minimizer of } E[(Q^{\pi_\theta}(s,a) - Q_w(s,a))^2]$$

Learn policy (actor) and Q-value (critic) simultaneously

baseline

$$\nabla_\theta J(\theta) = E[\nabla_\theta \log \pi_\theta(a|s)(Q^\pi(s,a) - b(s))]$$

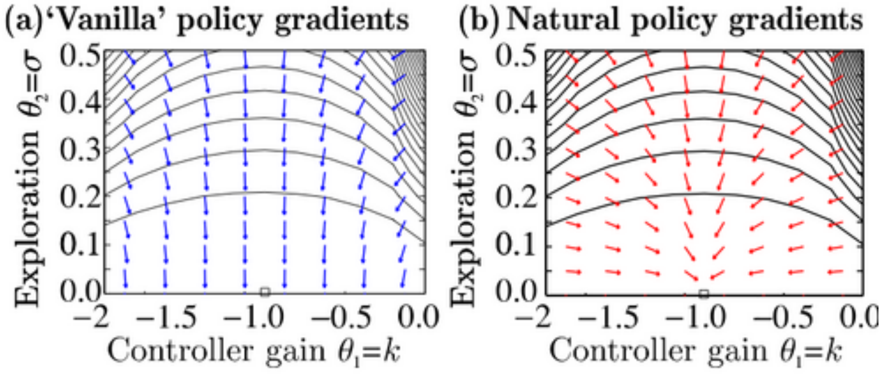
advantage function: $A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s)$

$$\nabla_\theta J(\theta) = E[\nabla_\theta \log \pi_\theta(a|s) A^\pi(s,a)]$$

learn policy, Q and V simultaneously

Policy gradient: other gradients

nature policy gradient



[Kakade. A Natural Policy Gradient. NIPS'01]

functional policy gradient

$$\pi_\Psi(a|\boldsymbol{s}) = \frac{\exp(\Psi(\boldsymbol{s},a))}{\sum_{a'}\exp(\Psi(\boldsymbol{s},a'))}$$

$$\Psi_t = \sum_{i=1}^{t} h_t$$

[Yu et al. Boosting nonparametric policies. AAMAS'16]

parameter-level exploration

$$\theta \sim \mathcal{N}$$

[Sehnke et al. Parameter-exploring policy gradients. Neural Networks'10]

asynchronous gradient update

[Mnih et al. Asynchronous Methods for Deep Reinforcement Learning . ICML'16]

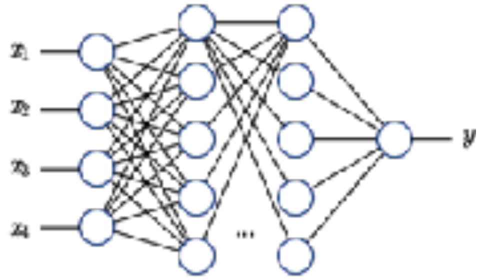
Optimization difficulty

the non-convexity

$$J(\theta) = \int_{Tra} p_\theta(\tau) R(\tau) \, \mathrm{d}\tau$$

where $p_\theta(\tau) = p(s_0) \prod_{i=1}^{T} p(s_i | a_i, s_{i-1}) \pi_\theta(a_i | s_{i-1})$

$$\pi_\theta(i|s) = \frac{\exp(f_\theta(i; \phi(s)))}{\sum_j \exp(f_\theta(j; \phi(s)))}$$

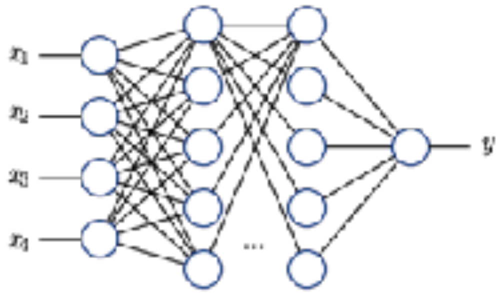
$f_\theta =$ 

too many local minima

EARL - EA for RL

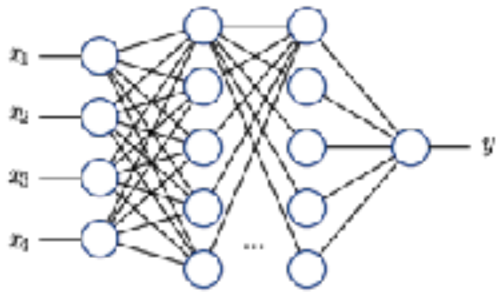
value-function representation

$$Q(s,a) =$$

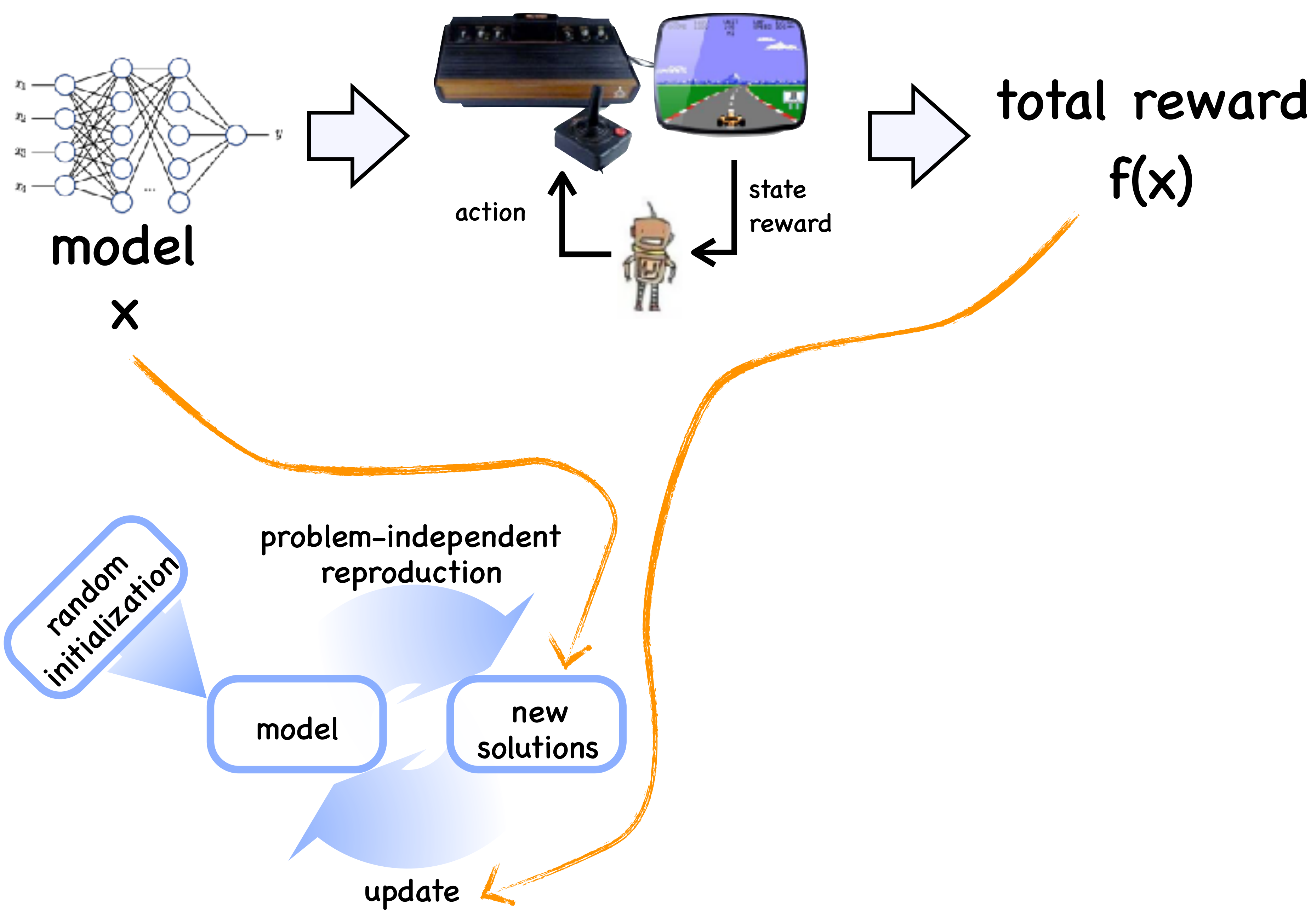


policy representation

$$\pi(s) =$$



Parameter search by EAs



Issue: low-efficiency

Does not utilize problem structure

1. NN parameters -> NN structure
2. Parameters -> dynamic programming

NEAT+Q

generations

evaluation with TD

reproduction

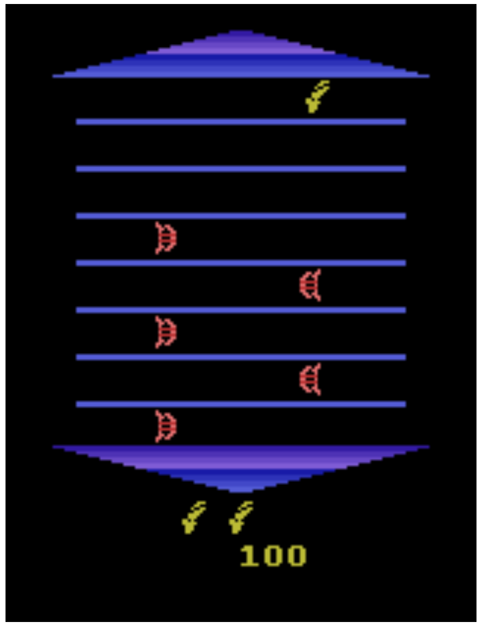
```
// S: set of all states, A: set of all actions, c: output scale, p: population size
// m_n: node mutation rate, m_l: link mutation rate, g: number of generations
// e: number of episodes per generation, α: learning rate, γ: discount factor
// λ: eligibility decay rate, ε_td: exploration rate

P[] ← INIT-POPULATION(S, A, p)     // create new population P with random networks
for i ← 1 to g do
  for j ← 1 to e do
    N, s, s' ← RANDOM(P[]), null, INIT-STATE(S)          // select a network randomly
    repeat
      Q[] ← c× EVAL-NET(N, s')          // compute value estimates for current state

      with-prob(ε_td) a' ← RANDOM(A)          // select random exploratory action
      else a' ← argmax_k Q[k]                          // or select greedy action
      if s ≠ null then
        BACKPROP(N, s, a, (r + γ max_k Q[k])/c, α, γ, λ) // adjust weights toward target

      s, a ← s', a'
      r, s' ← TAKE-ACTION(a')                  // take action and transition to new state
      N.fitness ← N.fitness + r                        // update total reward accrued by N
    until TERMINAL-STATE?(s)
    N.episodes ← N.episodes + 1                  // update total number of episodes for N
  P'[] ← new array of size p                      // new array will store next generation
  for j ← 1 to p do
    P'[j] ← BREED-NET(P[])           // make a new network based on fit parents in P
    with-probability m_n: ADD-NODE-MUTATION(P'[j]) // add a node to new network
    with-probability m_l: ADD-LINK-MUTATION(P'[j])  // add a link to new network
  P[] ← P'[]
```

Shimon Whiteson and Peter Stone. Evolutionary Function Approximation for Reinforcement Learning. Journal of Machine Learning Research, 7:877-917, 2006

Some comparison

on Atari games

freeway



asterix

	Freeway	Asterix
Sarsa(λ)-BASS	0	402
Sarsa(λ)-DISCO	0	301
Sarsa(λ)-RAM	0	545
Random	0	156
HyperNEAT-GGP (Average)	27.4	870
HyperNEAT-GGP (Best)	29	1000

Matthew J. Hausknecht, Piyush Khandelwal, Risto Miikkulainen, Peter Stone:HyperNEAT-GGP: a hyperNEAT-based atari general game player. GECCO 2012: 217-224

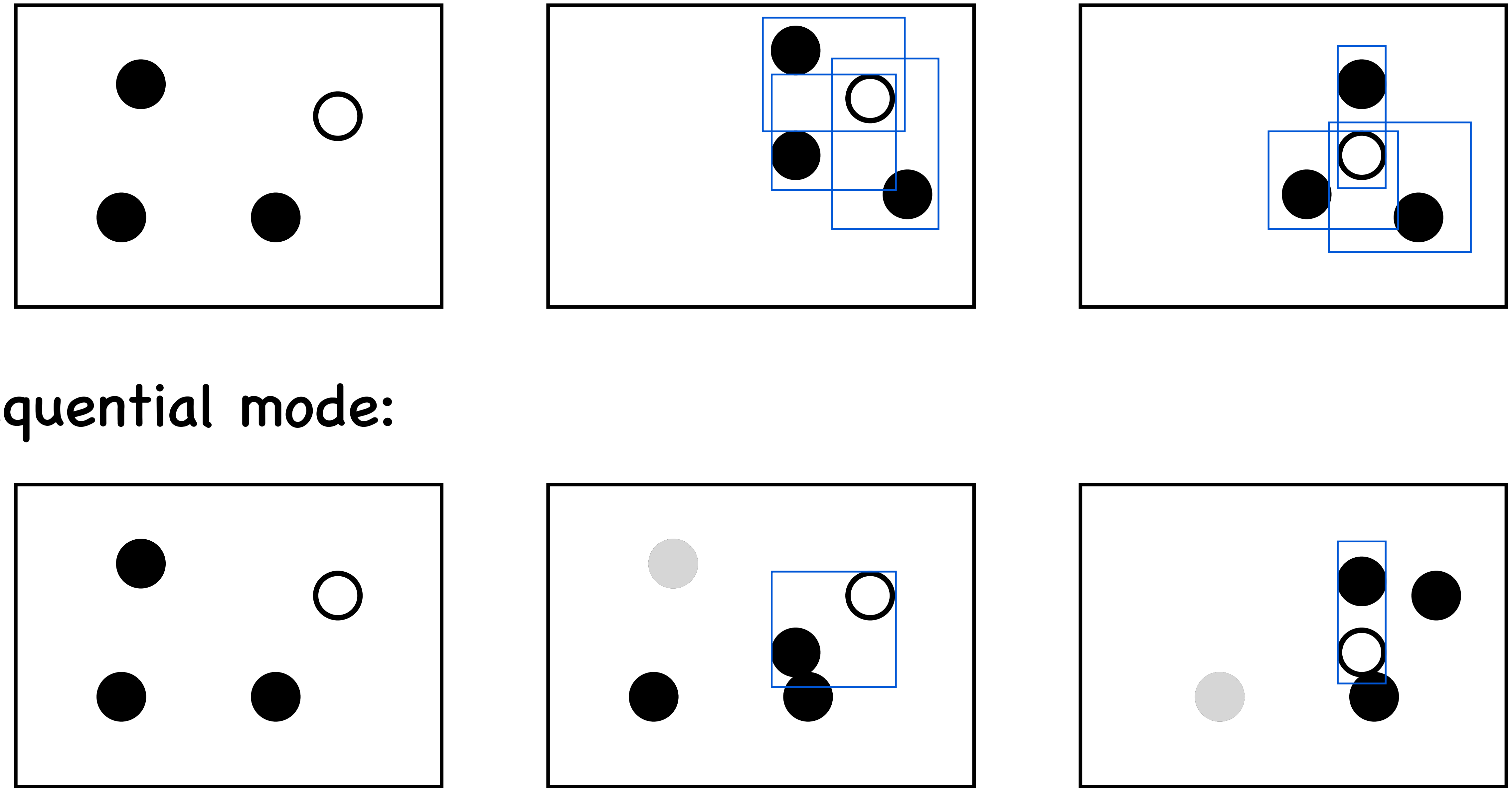
Issue: low-efficiency

Batch sampling

Online update

batch mode:

sequential mode:



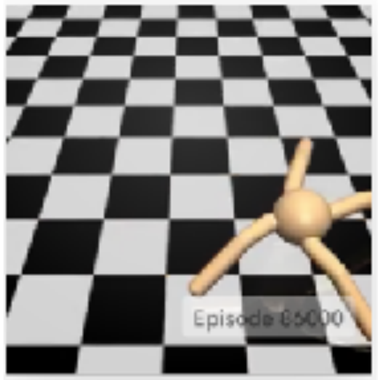
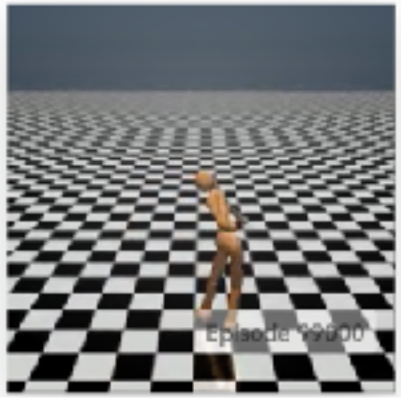
Yi-Qi Hu, Hong Qian, and Yang Yu. Sequential classification-based optimization for direct policy search. In: Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI'17), San Francisco, CA, 2017, pp.2029-2035.

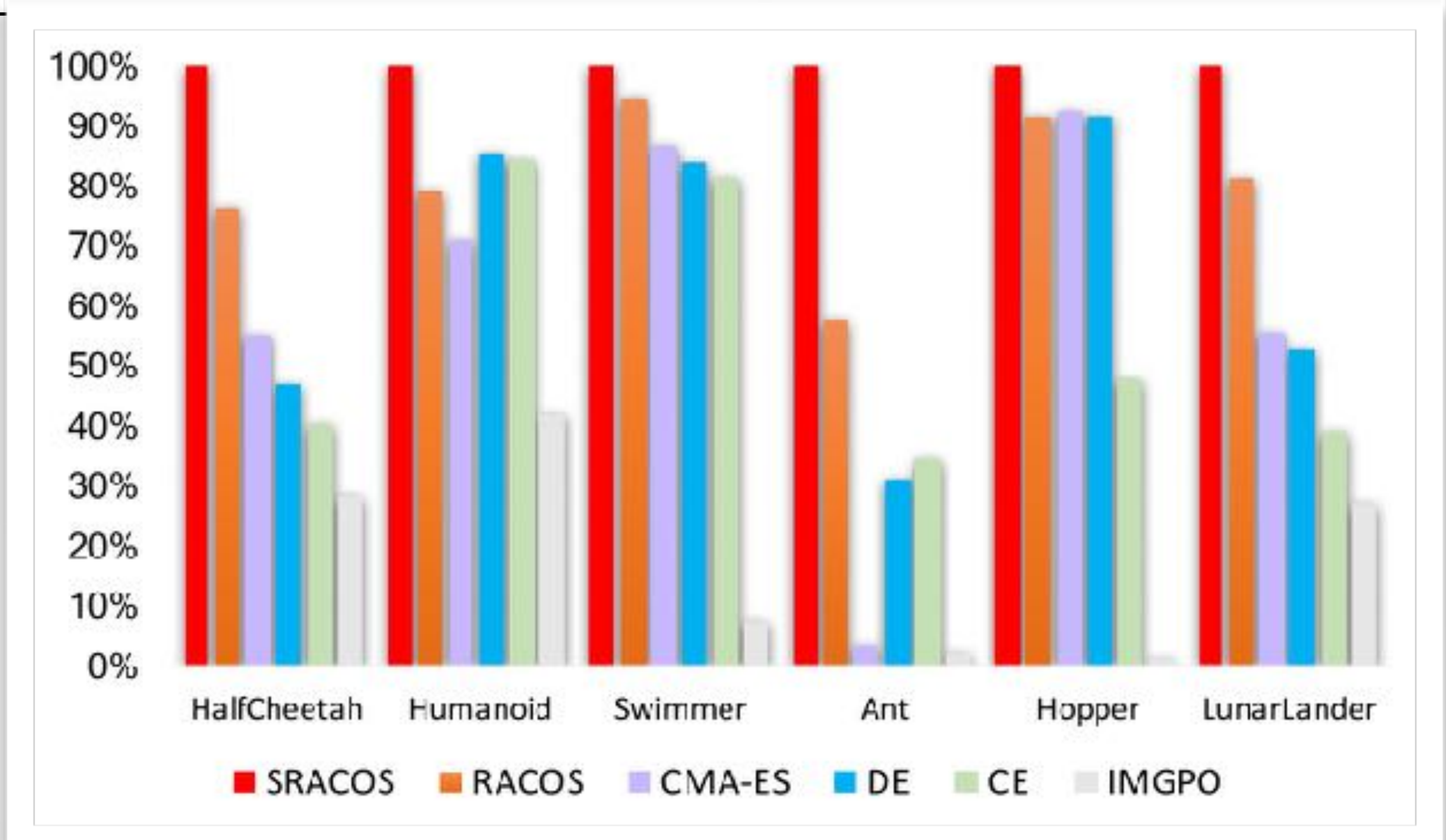
Issue: low-efficiency

Batch sampling   Online update

Task name	d_{State}	#Actions	NN nodes	#Weights	Horizon
Acrobot	6	1	5, 3	48	2,000
MountainCar	2	1	5	15	10,000
HalfCheetah	17	6	10	230	10,000
Humanoid	376	17	25	9825	50,000
Swimmer	8	2	5, 3	61	10,000
Ant	111	8	15	1785	10,000
Hopper	11	3	9, 5	159	10,000
LunarLander	8	1	5, 3	58	10,000





Yi-Qi Hu, Hong Qian, and Yang Yu. Sequential classification-based optimization for direct policy search. In: Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI'17), San Francisco, CA, 2017, pp.2029-2035.

Issue: low-efficiency

Parallel: evolutionary strategies

centralized

Input: Learning rate α, noise standard deviation σ, initial policy parameters θ_0
for $t = 0, 1, 2, \ldots$ **do**
    Sample $\epsilon_1, \ldots \epsilon_n \sim \mathcal{N}(0, I)$
    Compute returns $F_i = F(\theta_t + \sigma\epsilon_i)$ for $i = 1, \ldots, n$
    Set $\theta_{t+1} \leftarrow \theta_t + \alpha \frac{1}{n\sigma} \sum_{i=1}^{n} F_i \epsilon_i$
end for

parallel

Input: Learning rate α, noise standard deviation σ, initial policy parameters θ_0
Initialize: n workers with known random seeds, and initial parameters θ_0
for $t = 0, 1, 2, \ldots$ **do**
    for each worker $i = 1, \ldots, n$ **do**
        Sample $\epsilon_i \sim \mathcal{N}(0, I)$
        Compute returns $F_i = F(\theta_t + \sigma\epsilon_i)$
    end for
    Send all scalar returns F_i from each worker to every other worker
    for each worker $i = 1, \ldots, n$ **do**
        Reconstruct all perturbations ϵ_j for $j = 1, \ldots, n$
        Set $\theta_{t+1} \leftarrow \theta_t + \alpha \frac{1}{n\sigma} \sum_{j=1}^{n} F_j \epsilon_j$
    end for
end for

time steps of ES to the top performance of TRPO

Environment	25%	50%	75%	100%
HalfCheetah	0.15	0.49	0.42	0.58
Hopper	0.53	3.64	6.05	6.94
InvertedDoublePendulum	0.46	0.48	0.49	1.23
InvertedPendulum	0.28	0.52	0.78	0.88
Swimmer	0.56	0.47	0.53	0.30
Walker2d	0.41	5.69	8.02	7.88

Tim Salimans, Jonathan Ho, Xi Chen, Ilya Sutskever. Evolution Strategies as a Scalable Alternative to Reinforcement Learning. arXiv:1703.03864

More issues to be solved

Noisy evaluation: too many repetitions

Large model: too large search space

Long-term reward: too complex objective function

Thanks for your time!

Questions/comments?