

A vision for
reinforcement learning
and
predictive maintenance

Charles Elkan
University of California, San Diego

August 21, 2011

What is the goal of maintenance?

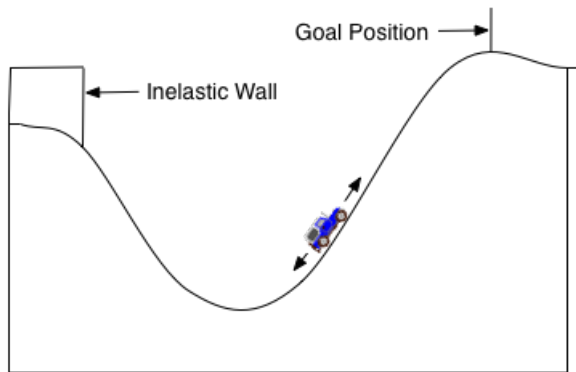
- Preventive maintenance is a small cost now intended to avoid a large cost later.
- Goal: Minimize **total** short-term + long-term cost.
 - ▶ Intrinsically probabilistic: Reduce **expected** later cost.
- From reactive maintenance to proactive maintenance?
- From scheduled maintenance to adaptive maintenance?
- From adaptive maintenance to predictive maintenance?

More concretely

- Every time step:
 - ▶ you have many measurements of a machine;
 - ▶ you can adjust settings and replace parts;
 - ▶ you get a short-term reward.
- The goal is to learn a strategy that maximizes the total long-term reward.
- Cost-sensitive learning maximizes short-term reward.

Which abstract research areas are relevant?

- From machine learning: reinforcement learning (RL).
- Operations research: approximate dynamic programming (ADP).



- Most important issue: Trade off cost now for reward later.
- Second issue: Exploration versus exploitation.

What are the goals of business?

- The goals of maintenance are the same.
- Maximize revenue.
- Minimize expense.
- Minimize risk.
 - ▶ Reduce probability of lost revenue, e.g. canceled flight.
 - ▶ Reduce probability of high expense, e.g. plane crash.

Lessons from data mining

- Predictions are useful **only** if they lead to better decisions.
- Costs **and** benefits **and** probabilities are important.
- Must take into account **follow-on** effects.
- Must take into account ability to **influence**.

Reinforcement learning (RL) framework

- An agent acts in an environment.
- Goal: Learn which action a is best in each state s .
- $Q(s, a)$ is the **total** reward achieved by
 - ▶ starting in state s ,
 - ▶ performing action a , then
 - ▶ performing the optimal action in each later state.

The bilinear idea

- States s and actions a are real-valued vectors.
- Recommended action is

$$a^* = \operatorname{argmax}_a Q(s, a)$$

- Write $Q(s, a) = s^T W a$ where W is a matrix.
- Recommended action is

$$a^* = \operatorname{argmax}_a (s^T W) \cdot a = \operatorname{argmax}_a x \cdot a$$

- Action selection **is** maximization **is** linear programming.
 - ▶ Linear constraints may be bounds for sub-actions of a .
 - ▶ Or, budget limits on costs of sub-actions.

Why is the bilinear approach useful?

- Given $a^* = \operatorname{argmax}_a (s^T W)a$
the chosen action a^* depends on the present state s only.
- But, the matrix W can be **learned**.
- So, W can emphasize aspects of a and s that are predictive of **long-term** reward.
- W has a weight for each component of a interacting with each component of s .

Alternative representations

- Tabular: separate Q value for each discrete s, a pair.
- Basis functions: $Q(s, a) = w \cdot [\phi_1(s, a), \dots, \phi_p(s, a)]$.
- Neural networks

- Shared difficulty: How to do the maximization efficiently?

$$a^* = \operatorname{argmax}_a Q(s, a)$$

Batch RL

- Also called RL from historical data.
- Extension of cost-sensitive supervised learning.
- A single training example is a tuple $\langle s, a, r, s' \rangle$ where
 - ▶ s is a state,
 - ▶ a is the action taken in that state,
 - ▶ r is the observed immediate reward,
 - ▶ s' is the state observed to happen next.
- An episode is a sequence of linked training examples $s'_t = s_{t+1}$.
- The learner does not control or know the policy π used to select actions $a = \pi(s)$.

Fitted Q iteration

- Algorithm:

Define $Q_0(s, a) = 0$ for all s and a .

For horizon $h = 0, 1, 2, \dots$

For each example $\langle s, a, r, s' \rangle$

let label $v = r + \gamma \max_b Q_h(s', b)$

Train Q_{h+1} with labeled tuples $\langle s, a, v \rangle$

- h is the horizon, γ is the discount factor.
- Proposed in parallel by [Murphy, 2005] and [Ernst et al., 2005].

Comparison with Q-learning

- The update rule of standard Q learning is

$$Q(s, a) := (1 - \alpha)Q(s, a) + \alpha[r + \gamma \max_b Q(s', b)].$$

- Drawbacks:
 - ▶ Each example $\langle s, a, r, s' \rangle$ is discarded after being used once.
 - ▶ How to choose the learning rate α ?
 - ▶ When $Q(s, a)$ is updated for one pair $\langle s, a \rangle$ then $Q(s, a)$ for different s, a changes unpredictably.
- Q iteration fits all $Q(s, a)$ values simultaneously, so the regression algorithm can minimize error on all values.

Sample selection bias

- The probability distribution of training examples $\langle s, a \rangle$ is not the distribution of optimal examples $\langle s, \pi^*(s) \rangle$.
- But Q iteration is discriminative, not generative.
 - ▶ It does not model the distribution of $\langle s, a \rangle$.
 - ▶ It only models how the Q value depends on $\langle s, a \rangle$.

Optimism bias

- Remember that the training label is

$$v = r + \gamma \max_b Q_h(s', b)$$

- The maximization tends to choose an action b that has upward error in $Q_h(s', b)$.
- Future work: use the double Q learning idea [[van Hasselt, 2010](#)] to reduce the optimism bias.

Learning the matrix W

- Consider a training set of examples $\langle s, a, v \rangle$.
- Suppose $W \in \mathbb{R}^{m \times n}$. Then

$$s^T W a = \sum_{i=1}^m \sum_{j=1}^n (W \circ sa^T)_{ij} = \text{vec}(W) \cdot \text{vec}(sa^T).$$

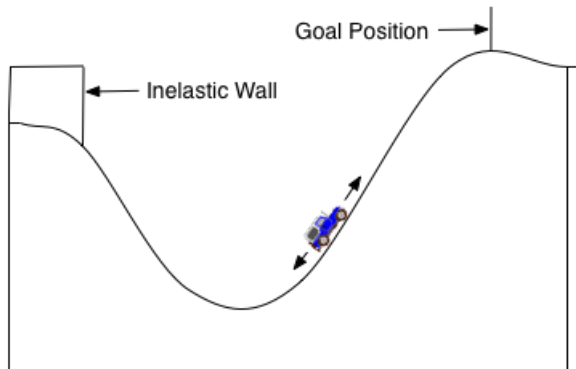
- Algorithm:
 - ▶ Convert each training triple $\langle s, a, v \rangle$ into $\langle \text{vec}(sa^T), v \rangle$.
 - ▶ Learn $\text{vec}(W)$ by standard linear regression.
- Each entry of the matrix sa^T is the interaction of a state feature and an action feature.

Practical variations

- Leave out interaction terms of sa^T known to be unproductive.
- Include bias (intercept) terms for states and actions.
- Can use stochastic gradient descent (SGD) instead of an exact solution.
- Use regularization to reduce overfitting.
- If s and a are short, then W may not be expressive enough.
 - ▶ Expand the s or a vectors with nonlinear transformations.

Mountain car domain

- Perhaps the best-known test case for reinforcement learning [Sutton and Barto, 1998].



- 2D state: position x and velocity v .
- 1D action: acceleration a .

Mountain car notes

- Use 6D expanded state vector $\langle x, v, x^2, xv, v^2, x^3 \rangle$ and 2D action vector $\langle 1, a \rangle$
 - ▶ The matrix W has 12 trainable parameters.
- Acceleration a is continuous between -0.001 and $+0.001$, but linear programming makes it always extremal.
- Training states and actions are chosen uniformly at random.
- Discount factor $\gamma = 0.9$.

Results

- Test episode starts at the bottom of the valley with velocity 0, and terminates at the goal or after 500 steps.
- Success as function of training set size:

# training tuples	mean test episode length
100	285.6
200	317.8
400	88.1
800	88.6
1600	87.4
3200	87.2
6400	87.3
12800	85.9

Discussion

- Q iteration with a bilinear Q function learns good control with just 400 training examples $\langle s, a, r, s' \rangle$.
 - ▶ Orders of magnitude faster than variants of Q learning [Smart and Kaelbling, 2000].
 - ▶ Faster than Q iteration with a neural network [Riedmiller, 2005].

Inventory management

- An agent has stocks of many products
- At each time step, it sees random demand and supply vectors.
- The agent must choose how to satisfy each demand, subject to rules about substitutability and perishability.
- Example: Managing the stocks of a blood bank.
 - ▶ 8 blood types, 3 ages, 27 allowed substitutions.
 - ▶ 24D state vector, 81D action vector.

Immediate reward functions

- Objectives to be maximized, short-term and long-term, are subject to debate.
- Alternative immediate benefits of different supply choices:

	[Yu, 2007]	new
give exact blood type	50	0
substitute O- blood	60	0
substitute other type	45	0
fail to meet demand	0	-60
discard blood	-20	0

Success of alternative learned policies

- Measure of long-term success: low frequency of severe (over 10%) failure to meet demand for one blood type.

	average unmet A+ demand	frequency of severe unmet A+ demand
greedy policy	7.3%	46%
policy of [Yu, 2007]	7.9%	30%
bilinear method (i)	18.4%	29%
bilinear method (ii)	7.55%	12%

What's next?

- For real applications, we need data!
- In maintenance: $\langle s, a, r, s' \rangle$ tuples where
 - ▶ s is the state of the locomotive
 - ▶ a is the set of maintenance actions taken, possibly empty
 - ▶ r is the short-term loss or gain
 - ▶ s' is the state of the locomotive the next week.
- A state is a vector of measurements,
e.g. mileage, temperature, oil pressure, oil age.

Possible issues

- High-cost outcomes are too rare in training data.
- There are too many different model types.
- Observed tuples cover too little of the state/action space.

- A commonsense or greedy policy is already close to optimal.
- Good policies are highly nonlinear.

References I

 Ernst, D., Geurts, P., and Wehenkel, L. (2005).

Tree-based batch mode reinforcement learning.

Journal of Machine Learning Research, 6(1):503–556.

 Murphy, S. A. (2005).

A generalization error for Q-learning.

Journal of Machine Learning Research, 6:1073–1097.

 Riedmiller, M. (2005).

Neural fitted Q iteration—first experiences with a data efficient neural reinforcement learning method.

In *Proceedings of the 16th European Conference on Machine Learning (ECML)*, pages 317–328.

References II

 Smart, W. D. and Kaelbling, L. P. (2000).

Practical reinforcement learning in continuous spaces.

In Proceedings of the 17th International Conference on Machine Learning (ICML), pages 903–910.

 Sutton, R. S. and Barto, A. G. (1998).

Reinforcement learning: An introduction.

MIT Press.

 van Hasselt, H. P. (2010).

Double Q-learning.

Advances in Neural Information Processing Systems (NIPS), 23.

References III



Yu, V. (2007).

Approximate dynamic programming for blood inventory management.

Honors thesis, Princeton University.