Replicator equations versus the Hedge algorithm
Lecture 4, part 2, CMPS 272, W12

Manfred K. Warmuth

University of California - Santa Cruz

January 28, 2012
Outline

1. Discrete time replicator

2. Continuous time replicator

3. Hedge algorithm - the paradigmic Machine Learning problem

4. An Application: Disk Spindown Problem
Outline

1. Discrete time replicator
2. Continuous time replicator
3. Hedge algorithm - the paradigmic Machine Learning problem
4. An Application: Disk Spindown Problem
Discrete time: \( n \) dimensional

\[
s_i(t) = \frac{s_i(t-1) e^{w_i}}{\sum_{j=1}^{n} s_j(t-1) e^{w_j}}
\]

\[
W_i = e^{w_i} \frac{s_i(t-1) W_j}{\sum_{j=1}^{n} s_j(t-1) W_j}
\]

Equation (1.6) of book

\( W_i \) is fitness (survival factor) of species \( i \)

\( w_i = \ln(W_i) \) is fitness rate
Discrete time replicator

Discrete replicator over time

$W_1 = 0.9$

$W_2 = 0.75$

$W_3 = 0.8$

Concentrations

Initial $s(0) = (0.01, 0.39, 0.6)$

$W = (0.9, 0.75, 0.8)$
Discrete time replicator

\[ \text{max; } W_i \text{ eventually wins} \]

- Solution: \( s_i(t) = \frac{s_i(0) W_i^t}{\text{normalization}} \), where \( t \) is speed / time parameter
- A \( t_1 \) step from time \( t \): \( s_i(t + t_1) = \frac{s_i(t) W_i^{t_1}}{\text{normalization}} \)
- \( \frac{1}{2} \) step + \( \frac{1}{2} \) step = 1 step  and  +1 step + -1 step = 0 step

Largest \( W_i \) always increasing
Smallest \( W_i \) always decreasing
Negative range of $t$ reveals sigmoids

- Smallest: Reverse Sigmoid
- Largest: Sigmoid

.3 takes over .49
The best are too good

- Multiplicative update
  \[ s_i(t) \sim s_i(t - 1)^{W_i} \geq 0 \]

- Blessing:
  Best get amplified exponentially fast

- Curse:
  The best wipe out the others
  Loss of variety

- Later lesson on how Nature and Machine Learners prevent the curse
Motivation of the discrete update: \( n \) dimensional

\[
s(t) = \max_{\sum_{i=1}^{n} s_i = 1} \sum_{i=1}^{n} s_i w_i - \sum_{i=1}^{n} s_i \ln \frac{s_i}{s_i(t-1)}
\]

\[
\text{rel.entr.betw.new&old}
\]

Why does Nature use relative entropies as inertia terms
Deriving discrete update from the motivation

Define Lagrangian

\[ L(s) = \sum_{i=1}^{n} s_i w_i - \sum_{i=1}^{n} s_i \ln \frac{s_i}{s_i(t-1)} + \lambda \left( \sum_{i=1}^{n} s_i - 1 \right) \]

where \( \lambda \) Lagrange coeff.

\[ \frac{\partial L(s)}{\partial s_i} = w_i - \ln \frac{s_i}{s_i(t-1)} - 1 + \lambda = 0 \]

\[ w_i + \lambda = \ln \frac{s_i(t)}{s_i(t-1)} + 1 \]

\[ s_i(t) = s_i(t-1) e^{w_i} e^{\lambda-1} \]

Enforce normalization constraint by setting \( e^{\lambda-1} \) to \( \frac{1}{\sum_{j=1}^{n} s_j(t-1) e^{w_j}} \)

\[ s_i(t) = \frac{s_i(t-1) e^{w_i}}{\sum_{j=1}^{n} s_j(t-1) e^{w_i}} \]
Unravelling the discrete time update

\[
s_i(t) = \frac{s_i(t-1) e^{w_i}}{\sum_{j=1}^{n} s_j(t-1) e^{w_j}} = \frac{s_i(t-2) e^{2w_i}}{\sum_{j=1}^{n} s_j(t-2) e^{2w_j}} = \ldots = \frac{s_i(0) e^{tw_i}}{\sum_{j=1}^{n} s_j(0) e^{tw_j}}
\]

Motivation

\[
s(t) = \max_{\sum_i s_i = 1} \sum_i s_i w_i - \frac{\sum_i s_i \ln \frac{s_i}{s_i(0)}}{t}
\]

Later we will let the time be positive or negative
Discrete time replicator

Discrete time: $n - 1$ dimensional

$$s_i(t) = \frac{s_i(t - 1) e^{w_i}}{\sum_{j=1}^{n-1} s_j(t - 1) e^{w_j} + (1 - \sum_{j=1}^{n-1} s_j(t - 1)) e^{w_n}}$$

(2)

$$= \frac{s_i(0) e^{t(w_i - w_n)}}{\sum_{j=1}^{n-1} s_j(0) e^{t(w_j - w_n)} + 1 - \sum_{j=1}^{n-1} s_j(0)}$$

(3)

Motivation - no constraint any more

$$s(t) = \max_{s_j} n-1 \sum_{i=1}^{n-1} s_i w_i + (1 - \sum_{i=1}^{n-1} s_j) w_n$$

$$- \sum_{i=1}^{n-1} s_i \ln \frac{s_i}{s_i(t - 1)} - (1 - \sum_{j=1}^{n-1} s_j) \ln \frac{1 - \sum_{j=1}^{n-1} s_j}{1 - \sum_{i=1}^{n-1} s_j(t - 1)}$$
Outline

1. Discrete time replicator

2. Continuous time replicator

3. Hedge algorithm - the paradigmic Machine Learning problem

4. An Application: Disk Spindown Problem
Continuous time: \( n - 1 \) dimensional

\[
\left( \ln \frac{s_i(t)}{1 - \sum_{j=1}^{n-1} s_j(t)} \right) = w_i - w_n \quad \text{for } 1 \leq i \leq n - 1
\]  

(4)

Plugging in solution (3)

\[
\frac{s_i(t)}{1 - \sum_{j=1}^{n-1} s_j(t)} = \frac{s_i(0) e^{t(w_i - w_n)}/Z}{(1 - \sum_{i=1}^{n-1} s_i(0))/Z}
\]

and therefore

\[
\left( \ln \frac{s_i(t)}{1 - \sum_{j=1}^{n-1} s_j(t)} \right) = \left( \ln \frac{s_i(0)}{1 - \sum_{j=1}^{n-1} s_j(0)} + t(w_i - w_n) \right)
\]

\[
= w_i - w_n
\]
Solving (4) using integration

\[
\int_{z=0}^{t} \left( \ln \frac{s_i(z)}{1 - \sum_{j=1}^{n-1} s_j(z)} \right) \, dz = \int_{z=0}^{t} (w_i - w_n) \, dz
\]

\[
\left[ \ln \frac{s_i(z)}{1 - \sum_{j=1}^{n-1} s_j(z)} \right]_{z=0}^{t} = t(w_i - w_n)
\]

\[
\ln \frac{s_i(t)}{1 - \sum_{j=1}^{n-1} s_j(t)} - \ln \frac{s_i(0)}{1 - \sum_{j=1}^{n-1} s_j(0)} = t(w_i - w_n)
\]

\[
s_i(t) = \frac{s_i(0) e^{t(w_i - w_n)}}{\sum_{j=1}^{n-1} s_j(0) e^{t(w_j - w_n)} + 1 - \sum_{j=1}^{n-1} s_j(0)} = (3)
\]
Solving \( \ln \frac{s_i(t)}{1 - \sum_{j=1}^{n-1} s_j(t)} - \ln \frac{s_i(t-1)}{1 - \sum_{j=1}^{n-1} s_j(t-1)} = w_i - w_n \) for \( s_i(t) \)

Rewrite to

\[
\frac{s_i(t)}{s_i(t-1)} e^{-w_i} = \frac{1 - \sum_{j=1}^{n-1} s_j(t)}{1 - \sum_{j=1}^{n-1} s_j(t-1)} e^{-w_n} := c
\]

Therefore

\[
s_i(t) = s_i(t-1) e^{w_i} c \quad \text{and} \quad 1 - \sum_{j=1}^{n-1} s_j(t) = (1 - \sum_{j=1}^{n-1} s_j(t-1)) e^{w_n} c
\]

Thus \( c \) is 1 over the normalization and we get update (2)
Continuous time: $n$ dimensional

\[
\dot{(\ln s_i(t))} = w_i - \sum_{j=1}^{n} w_j s_j(t), \quad \text{for } 1 \leq i \leq n
\]  

\( (5) \)

Different from both (1.12) and (1.16) of Chapter 1

Satisfied by solution $s_i(t) = \frac{s_i(0) e^{tw_i}}{\sum_{j=1}^{n} s_j(0) e^{tw_j}}$:

\[
(\ln s_i(t)) = \left( \ln(s_i(0)) + tw_i - \ln\left( \sum_{j=1}^{n} s_j(0) e^{tw_j} \right) \right) = w_i - \sum_{j=1}^{n} \frac{s_j(0) e^{tw_j}}{\sum_{k} s_k(0) e^{tw_k}} w_j
\]
Solution for (5) stays in $\sum_{i=1}^{n} s_i(t)$ plane

Implied by a thm given in Weibull
Also we already showed that the standard solution solves (5)
Uniqueness of this solution is implied by Picard-Lindelöf
Solving (5) using integration

\[
\int_{z=0}^{t} (\ln s_i(z)) \, dz = \int_{z=0}^{t} (w_i - \mathbf{w} \cdot \mathbf{s}(t)) \, dz
\]

\[
\ln s_i(t) - \ln s_i(0) = tw_i - \int_{z=0}^{t} (\mathbf{w} \cdot \mathbf{s}(t)) \, dz
\]

\[
s_i(t) = s_i(0) \, e^{tw_i} \, e^{-\int_{z=0}^{t} (\mathbf{w} \cdot \mathbf{s}(t)) \, dz}
\]

- By the constraint, lhs sums to 1
- Therefore factor \( e^{-\int_{z=0}^{t} (\mathbf{w} \cdot \mathbf{s}(t)) \, dz} \) must be 1 over the normalization
- Resulting in update (1)
Discretization

\[
\left( \ln \frac{s_i(t)}{1 - \sum_{j=1}^{n-1} s_j(t)} \right) = w_i - w_n \quad \text{for } 1 \leq i \leq n - 1
\]

Discrete solution \( s_i(t) = \frac{s_i(0) e^{t(w_i - w_n)}}{\sum_{j=1}^{n-1} s_j(0) e^{t(w_j - w_n)} + 1 - \sum_{j=1}^{n-1} s_j(0)} \) satisfies

\[
\ln \frac{s_i(t)}{1 - \sum_{j=1}^{n-1} s_j(t)} - \ln \frac{s_i(t-1)}{1 - \sum_{j=1}^{n-1} s_j(t-1)} = w_i - w_n
\]

\[
(\ln s_i(t)) = w_i - \sum_{j=1}^{n} w_j s_j(t), \quad \text{for } 1 \leq i \leq n
\]

Discrete solution \( s_i(t) = \frac{s_i(t-1) e^{w_i}}{\sum_{j=1}^{n} s_j(t-1) e^{w_j}} \) does not satisfy

\[
\ln s_i(t) - \ln s_i(t-1) = w_i - \sum_{j=1}^{n} w_j s_j(t)
\]
Continuous time replicator

What is needed?

- Find continuous time relative entropy and thus explicit minimization problem that produces differential equation

\[
\max \quad \sum_{i=1}^{n} s_i \sum_{i=1}^{n} s_i w_i - ??? \text{divergence}???
\]

Easy to generalize from

- Find \( n \)-dimensional differential equation that discretizes to the right solution
Continuous time replicator

Time dependent rates $w_i(t)$

\[
\left( \ln \frac{s_i(t)}{1 - \sum_{j=1}^{n-1} s_j(t)} \right) = w_i(t) - w_n(t) \quad \text{for} \ 1 \leq i \leq n - 1
\]

Solution by integrating

\[
\ln \frac{s_i(t)}{1 - \sum_{j=1}^{n-1} s_j(t)} - \ln \frac{s_i(0)}{1 - \sum_{j=1}^{n-1} s_j(0)} = \int_{z=0}^{t} (w_i(z) - w_n(z)) \, dz
\]

\[
s_i(t) = \frac{s_i(0) \ e^{\int_{z=0}^{t} (w_i(z) - w_n(z)) \, dz}}{\sum_{j=1}^{n-1} s_j(0) \ e^{\int_{z=0}^{t} (w_j(z) - w_n(z)) \, dz} + 1 - \sum_{j=1}^{n-1} s_j(0)}
\]
Outline

1. Discrete time replicator
2. Continuous time replicator
3. Hedge algorithm - the paradigmic Machine Learning problem
4. An Application: Disk Spindown Problem
Expert algorithms

- Master algorithm shepards $n$ experts.
- Model as population with $s_i(t)$ share of species / expert $i$.
- Master ”hedges” as to which expert to trust. Share vector represents uncertainty of Master about which expert is good.
- Update shares multiplicatively based on the losses incurred by the experts.
Learning on-line

- Pick expert $i$ based on share vector $s(t-1)$
- Receive loss vector $\lambda(t) \in [0, 1]^n$
- Incur loss $\lambda_i(t)$ and expected loss $s(t-1) \cdot \lambda(t)$
- Update $s(t-1)$ to $s(t)$ using discrete replicator
Example:

- \( s(0) = (1/3, 1/3, 1/3)^\top \) - distribution on experts
- Pick an expert according to \( s(0) \), say \( i = 2 \)
- Receive losses for all experts \( \lambda_1 = (1, 0, 1)^\top \)
- Incur loss \( \lambda_2(1) = 0 \), expected loss is \( s(0) \cdot \lambda(1) = 2/3 \)
- Update \( s(0) \) to \( s(1) \)
Hedge algorithm: Softmin with exponential weights

[\textbf{LW,FS}]

- Choose expert based on probability vector

\[ s_i(t) = \frac{s_i(0) e^{-\eta \lambda_i(<t)}}{Z_t} \quad s_i(t) = \frac{s_i(t - 1) e^{-\eta \lambda_i(t)}}{Z'_t} \]

Time dependent discrete replicator with \( w(t) = -\eta \lambda(t) \)

- Motivation

\[ s(t) = \arg \inf \left( \sum_{i} \omega_i = 1 \right) \left( \sum_i \omega_i \ln \left( \frac{s_i}{s_i(t - 1)} \right) + \eta s \cdot \lambda(t) \right) \]
Machine Learning goal

- \( \text{loss}_{\text{alg}} = \sum_t s(t - t) \cdot \lambda(t) \) not too much larger than loss of best in hindsight \( \text{loss}_{\text{best}} = \inf_i \sum_t \lambda_i(t) \)
- Prove a certain inequality that uses the relative entropy as a measure of progress
- Sum this inequality over trials and tune \( \eta \)
  \[
  \text{loss}_{\text{alg}} \leq \text{loss}_{\text{best}} + \sqrt{2\text{loss}_{\text{best}}} \log n + \log n
  \]
- Bound holds for any \( \lambda(t) \)!
- Relative entropy used in motivation of update and as measure of progress in analysis
Outline

1. Discrete time replicator
2. Continuous time replicator
3. Hedge algorithm - the paradigmic Machine Learning problem
4. An Application: Disk Spindown Problem
Spinning down the disk costs energy

If next idle period long then should spin down

Problem: We don’t know length of next idle period \( d(t) \)
  - If short then we should keep the disk spinning
  - Idle periods are bursty and chaotic

Idea: Model as a game against adversary
  - Algorithm picks a timeout
  - Adversary picks the next idle period
An Application: Disk Spindown Problem

Loss function for Disk Spindown Problem

\[
\text{Loss}(x, d) = \begin{cases} 
  d & \text{if } d \leq x \\
  x + c & \text{if } d \geq x 
\end{cases}
\]

For timeout 0, \( \text{Loss}(0, d) \) is always the spindown cost \( c \)

Only timeouts \( < c \) can ever have loss \( < c \)

If \( x < c \), then \( \text{Loss}(x, d) \in [0, 2c] \)
What experts?

Discretize interval [0, \text{spindowncost}]
Each point $x_i$ represents expert using timeout $x_i$
How to use the Hedge algorithm

- Maintain share vector $s(t)$, where $s_i(t)$ is share of timeout $x_i$ at time $t$
- Hedge update

$$s_i(t) = s_i(t-1) e^{-\eta \frac{\text{Loss}(x_i,d(t))}{2c}}$$

- Since $x_i \in [0, c)$, scaled loss $\in [0, 1)$
- So bounds apply to the total (scaled) loss (when $\eta$ tuned):

$$\text{loss}_{\text{alg}} \leq \text{loss}_{\text{best}} + \sqrt{2 \text{loss}_{\text{best}}} \log n + \log n$$
What if data changes over time

- Burst favors long timeout
- Coefficients of small timeouts wiped out

Fix:
- Do multiplicative updates but if weights get too low then reset to $\alpha^{1/n}$ for some small $\alpha$
- Prevents loss of variety - curse of the multiplicative update
Conclusion

- Nature uses multiplicative updates
- Motivated by relative entropies
- Tricky to go form continuous time to discrete time
- Continuous entropies needed

To come
- Where to entropies come from
- Framework for how to use them
- Implicit versus explicit updates for frequency dependant setting
- Tricks for how to prevent the curse