Using Lanchester Attrition Laws for Combat Prediction in StarCraft

Authors: Marius Stanescu, et. al.
Presenter: Fenix Chen
Background

- Real-time Strategy (RTS) Game
  real-time; simultaneous control; usu. combat focused
- StarCraft
  RTS game by Blizzard Entertainment in 1998;
  Gameplay: resource gathering, production, combat

Images from StarCraft Wiki
Starcraft Battle Example

Image from StarCraft Compendium
Introduction

• Significance
  (1) RTS game represent well-defined complex adversarial environment for AI study;
  (2) Hard problem, harder than Go

• Limitations in existing works
  (1) Linear model only;
  (2) Only 4 unit types;
  (3) Only predict win/lose;
  (4) Partial health not modeled;
  (5) Experiments done only on simulated data;
  (6) Correlation between unit types not modeled

• Objective
  Introduce a model to address the first five limitations above.
Lanchester Models

- Initially proposed by Lanchester in 1916
- Model for the change of army sizes during a combat
- Assumptions
  1. Identical units within each side;
  2. No reinforcements;
  3. Most factors (other than army sizes) either ignored or abstracted;
  4. Ignore discretization;
  5. Army sizes do not go negative (termination)
Lanchester Models

• Notations
  \( t \): time, start at zero and increases;
  \( A, B \): the army sizes of the two sides at time \( t \);
  \( A_0, B_0 \): the initial (i.e. at time zero) army sizes of the two sides;
  \( \alpha, \beta \): the relative strengths (combat effectivenesses) of each unit in the two sides;
  \( n \): attrition order, usu. in [1, 2]

• Linear Law (ancient warfare)
  \[ \alpha (A - A_0) = \beta (B - B_0) \]

• Square Law (modern warfare)
  \[ \alpha (A^2 - A_0^2) = \beta (B^2 - B_0^2) \]

• Generalized Form
  \[ \alpha (A^n - A_0^n) = \beta (B^n - B_0^n) \]
Deriving Lanchester Models
- First Attempt

• “Brute-force Math” (e.g. for Square Law)

[Time-sink Warning]

\[ \frac{dA}{dt} = -\beta B \quad \text{and} \quad \frac{dB}{dt} = -\alpha A \]

……

(omitting several pages)

……

\[ A = \frac{\sqrt{\alpha} A_0 + \sqrt{\beta} B_0}{2\sqrt{\alpha}} e^{-\sqrt{\alpha} \beta t} + \frac{\sqrt{\alpha} A_0 - \sqrt{\beta} B_0}{2\sqrt{\alpha}} e^{\sqrt{\alpha} \beta t} \]

\[ B = \frac{\sqrt{\alpha} A_0 + \sqrt{\beta} B_0}{2\sqrt{\beta}} e^{-\sqrt{\alpha} \beta t} - \frac{\sqrt{\alpha} A_0 - \sqrt{\beta} B_0}{2\sqrt{\beta}} e^{\sqrt{\alpha} \beta t} \]
Deriving Lanchester Models - Redo

• Square Law

\[
\frac{dA}{dt} = -\beta B \quad \& \quad \frac{dB}{dt} = -\alpha A
\]

Do some little transformation

\[
\alpha A \frac{dA}{dt} = \beta B \frac{dB}{dt}
\]

then we have

\[
\alpha \frac{dA^2}{dt} = \beta \frac{dB^2}{dt}
\]

Integrate both sides over \(t\)

\[
\alpha (A^2 - A_0^2) = \beta (B^2 - B_0^2)
\]
Deriving Lanchester Models - Redo

- Linear Law

On the paper:

\[ \frac{dA}{dt} = -\beta AB \quad \text{&} \quad \frac{dB}{dt} = -\alpha BA \]

or my understanding (according to the melee interpretation)

\[ \frac{dA}{dt} = -\beta \min(A, B) \quad \text{&} \quad \frac{dB}{dt} = -\alpha \min(A, B) \]

Do some little transformation

\[ \alpha \frac{dA}{dt} = \beta \frac{dB}{dt} \]

Integrate both sides over \( t \)

\[ \alpha (A - A_0) = \beta (B - B_0) \]
• Generalized Form

\[
\frac{dA}{dt} = -\beta A^{2-n} B \quad \& \quad \frac{dB}{dt} = -\alpha B^{2-n} A
\]

Do some little transformation

\[
\alpha A^{n-1} \frac{dA}{dt} = \beta B^{n-1} \frac{dB}{dt}
\]

then we have

\[
\alpha \frac{dA^n}{dt} = \beta \frac{dB^n}{dt}
\]

Integrate both sides over \( t \)

\[
\alpha(A^n - A_0^n) = \beta(B^n - B_0^n)
\]
Apply Lanchester Model to RTS

- $\alpha A^n - \beta B^n = \alpha A_0^n - \beta B_0^n$
- Experiments suggest that $n \approx 1.56$ works best for StarCraft
- If $\alpha A_0^n > \beta B_0^n$, predict player A wins with remainder $A_f$ s.t. $\alpha A_f^n = \alpha A_0^n - \beta B_0^n$
- If $\alpha A_0^n < \beta B_0^n$, predict player B wins with remainder $B_f$ s.t. $\beta B_f^n = \beta B_0^n - \alpha A_0^n$
Further Considerations

• Default value for unit strength (α’s, β’s)
  \[ \alpha_i = \text{dmg}(i)\text{HP}(i) \]

• Partial health: use current HP instead of full HP (Lanchester’s model deals with the rest properly)
  \[ \alpha_i = \text{dmg}(i)\text{currentHP}(i) \]

• Heterogeneous army compositions: take average over all individuals
  \[ \alpha_{avg} = \frac{\sum_{j=1}^{A} \alpha_j}{A} \]
Learning Unit Strengths

• Finding maximum likelihood for unit strengths using gradient ascent method (starting from default values)

• Gaussian distribution for modeling combat result (i.e. the remaining total army strength)

• To avoid overfitting, add a regularization term to the error function (the function to minimize). The regularization prevents the learning goes too far from default value.
Experiments - Training

- Based on Simulator Generated data
- Simulator: SparCraft by David Churchill, UAlberta
- 7 unit types; up to population size of 100
- Attack-closest script
- Up to 500 battles for training
- Another 500 for validation (accuracy of prediction)

<table>
<thead>
<tr>
<th>Model</th>
<th>10</th>
<th>20</th>
<th>50</th>
<th>100</th>
<th>200</th>
<th>500</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lanchester</td>
<td>89.8</td>
<td>91.0</td>
<td>91.7</td>
<td>92.2</td>
<td>93.0</td>
<td>93.2</td>
</tr>
<tr>
<td>Bayesian</td>
<td>83.0</td>
<td>86.9</td>
<td>88.5</td>
<td>89.4</td>
<td>90.3</td>
<td>91.2</td>
</tr>
</tbody>
</table>
Experiments - Tournament Testing

- AIIDE StarCraft AI competitions
- Incorporate into UAlbertaBot (zealot only)
- Against 6 AI opponents, including original UAlbertaBot

I have and I know or think my opponent has

Do I attack or not?
Experiments - Tournament Testing

- Win rates

<table>
<thead>
<tr>
<th></th>
<th>UAB</th>
<th>Xelnaga</th>
<th>Aiur</th>
<th>MooseBot</th>
<th>IceBot</th>
<th>LetaBot</th>
<th>Avg.</th>
</tr>
</thead>
<tbody>
<tr>
<td>UAB</td>
<td>50.0</td>
<td>80.5</td>
<td>27.0</td>
<td>53.5</td>
<td>7.5</td>
<td>31.5</td>
<td>41.6</td>
</tr>
<tr>
<td>Sim.</td>
<td>60.0</td>
<td>79.0</td>
<td>84.0</td>
<td>65.5</td>
<td>19.5</td>
<td>57.0</td>
<td>60.8</td>
</tr>
<tr>
<td>Def.</td>
<td>64.5</td>
<td>81.0</td>
<td>80.5</td>
<td>69.0</td>
<td>22.0</td>
<td>66.5</td>
<td>63.9</td>
</tr>
<tr>
<td>Train</td>
<td>69.5</td>
<td>78.0</td>
<td>86.0</td>
<td>93.0</td>
<td>23.5</td>
<td>68.0</td>
<td>69.7</td>
</tr>
</tbody>
</table>

Sim.: use simulation to predict
Def.: use Lanchester model with default unit strengths to predict
Train.: use Lanchester model with unit strengths trained in previous set of tournament to predicts
Limitations / Things Left Undone

• Correlations between unit types are not modeled
• Army formation (i.e. position of the units, or spatial information) is not modeled
• The cost for losing certain unit is not modeled
• The effect of terrain etc. may affect the attrition order (e.g. choke points)
• Different units may suits different attrition orders; sometimes even asymmetrical model
• Correlation between unit type, terrain, formation etc.