Chess and "Natural Laws"

How might human decision regularities appear in other domains?

Kenneth W. Regan[1]
University at Buffalo (SUNY)

UB CSE 501, 11/13/2018

[1] Joint work with Tamal Tanu Biswas and with grateful acknowledgment to UB's Center for Computational Research (CCR)

Chess and “Natural Laws”

How might human decision regularities appear in other domains?

Kenneth W. Regan[1]
University at Buffalo (SUNY)

UB CSE 501, 11/13/2018 Password: Vicki Hanson

[1] Joint work with Tamal Tanu Biswas and with grateful acknowledgment to UB's Center for Computational Research (CCR)

[1] Joint work with Tamal Tanu Biswas and with grateful acknowledgment to UB's Center for Computational Research (CCR)

Chess and Natural Laws

- Logistic Laws. https://rjlipton.wordpress.com/2012/03/30/when-is-a-law-natural/

Chess and Natural Laws

- Logistic Laws. https://rjlipton.wordpress.com/2012/03/30/when-is-a-law-natural/
- The Win-Expectation Curve: https://rjlipton.wordpress.com/2016/12/08/magnus-and-the-turkey-grinder/

Chess and Natural Laws

- Logistic Laws.
  https://rjlipton.wordpress.com/2012/03/30/when-is-a-law-natural/
- The Win-Expectation Curve:
  https://rjlipton.wordpress.com/2016/12/08/magnus-and-the-turkey-grinder/
- Relative Perception of Value:
  https://rjlipton.wordpress.com/2016/11/30/when-data-serves-turkey/

Chess and Natural Laws

- Logistic Laws.
  https://rjlipton.wordpress.com/2012/03/30/when-is-a-law-natural/
- The Win-Expectation Curve:
  https://rjlipton.wordpress.com/2016/12/08/magnus-and-the-turkey-grinder/
- Relative Perception of Value:
  https://rjlipton.wordpress.com/2016/11/30/when-data-serves-turkey/
- Predictive Analytics: Inferring the probabilities p_j of various events j:
  - Risk or damage events.
  - Voter j choosing candidate i.
  - Student i choosing answer j.
  - Player choosing move m_j at chess.

Chess and Tests

The ____ of drug-resistant strains of bacteria and viruses has ____ researchers’ hopes that permanent victories against many diseases have been achieved.

(a) vigor . . corroborated

(b) feebleness . . dashed

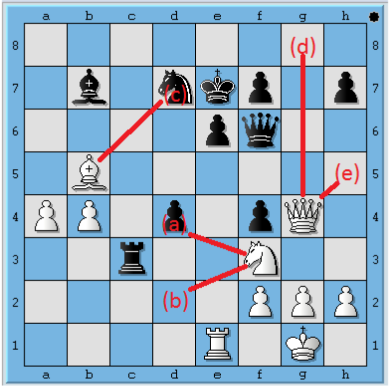
(c) proliferation . . blighted

(d) destruction . . disputed

(e) disappearance . . frustrated

(source: itunes.apple.com)

=



Multinomial Logit Model

Given options $m_1, \ldots, m_J$ and information $X = X_1, \ldots, X_J$ about all of them, and characteristics S of a person choosing among them, we want to project the probabilities p_j of m_j being chosen.

Multinomial Logit Model

Given options $m_1, \ldots, m_J$ and information $X = X_1, \ldots, X_J$ about all of them, and characteristics S of a person choosing among them, we want to project the probabilities p_j of m_j being chosen. First define numbers $u_j = g(X, S)_j$ often thought of as "utilities."

Multinomial Logit Model

Given options $m_1, \ldots, m_J$ and information $X = X_1, \ldots, X_J$ about all of them, and characteristics S of a person choosing among them, we want to project the probabilities p_j of m_j being chosen. First define numbers $u_j = g(X, S)_j$ often thought of as "utilities." Then the *multinomial logit* (MNL) model represents the probabilities via

$$\log(p_j) = \alpha + \beta u_j.$$

Multinomial Logit Model

Given options $m_1, \dots, m_J$ and information $X = X_1, \dots, X_J$ about all of them, and characteristics S of a person choosing among them, we want to project the probabilities p_j of m_j being chosen. First define numbers $u_j = g(X, S)_j$ often thought of as "utilities." Then the *multinomial logit* (MNL) model represents the probabilities via

$$\log(p_j) = \alpha + \beta u_j.$$

The quantities

$$L_j = e^{\alpha + \beta u_j}$$

are called *likelihoods*.

Multinomial Logit Model

Given options $m_1, \dots, m_J$ and information $X = X_1, \dots, X_J$ about all of them, and characteristics S of a person choosing among them, we want to project the probabilities p_j of m_j being chosen. First define numbers $u_j = g(X, S)_j$ often thought of as "utilities." Then the *multinomial logit* (MNL) model represents the probabilities via

$$\log(p_j) = \alpha + \beta u_j.$$

The quantities

$$L_j = e^{\alpha + \beta u_j}$$

are called *likelihoods*. Then the probabilities are obtained simply by normalizing them:

$$p_j = \frac{L_j}{\sum_{j'=1}^{J} L_{j'}} =_{def} \mathit{softmax}(\beta u_1, \dots, \beta u_J).$$

Multinomial Logit Model

Given options $m_1, \dots, m_J$ and information $X = X_1, \dots, X_J$ about all of them, and characteristics S of a person choosing among them, we want to project the probabilities p_j of m_j being chosen. First define numbers $u_j = g(X, S)_j$ often thought of as "utilities." Then the *multinomial logit* (MNL) model represents the probabilities via

$$\log(p_j) = \alpha + \beta u_j.$$

The quantities

$$L_j = e^{\alpha + \beta u_j}$$

are called *likelihoods*. Then the probabilities are obtained simply by normalizing them:

$$p_j = \frac{L_j}{\sum_{j'=1}^{J} L_{j'}} =_{def} \mathit{softmax}(\beta u_1, \dots, \beta u_J).$$

Finally obtain β by fitting; e^α becomes a constant of proportionality so that the p_j sum to 1.

Chess Decision Setting

Chess Decision Setting

- One player P with characteristics S.

Chess Decision Setting

- One player P with characteristics S.
- Multiple *game turns* t, each has possible moves $m_{t,j}$.

Chess Decision Setting

- One player P with characteristics S.
- Multiple *game turns* t, each has possible moves $m_{t,j}$.
- For a given turn (i.e., chess position) t, legal moves are $m_1, \ldots, m_j, \ldots, m_J$ (index t understood).

Chess Decision Setting

- One player P with characteristics S.
- Multiple *game turns* t, each has possible moves $m_{t,j}$.
- For a given turn (i.e., chess position) t, legal moves are $m_1, \ldots, m_j, \ldots, m_J$ (index t understood).
- Moves indexed by values $v_1, \ldots, v_J$ in nonincreasing order.

Chess Decision Setting

- One player P with characteristics S.
- Multiple *game turns* t, each has possible moves $m_{t,j}$.
- For a given turn (i.e., chess position) t, legal moves are $m_1, \dots, m_j, \dots, m_J$ (index t understood).
- Moves indexed by values $v_1, \dots, v_J$ in nonincreasing order.
- Values determined by strong chess programs. Not apprehended fully by P (*bounded rationality*, *fallible agents*).

Chess Decision Setting

- One player P with characteristics S.
- Multiple *game turns* t, each has possible moves $m_{t,j}$.
- For a given turn (i.e., chess position) t, legal moves are $m_1, \dots, m_j, \dots, m_J$ (index t understood).
- Moves indexed by values $v_1, \dots, v_J$ in nonincreasing order.
- Values determined by strong chess programs. Not apprehended fully by P (*bounded rationality*, *fallible agents*).
- Raw utilities $u_j = \delta(v_1, v_j)$ by some difference-in-value function δ in either "pawn units" or "chance of winning" units.

Chess Decision Setting

- One player P with characteristics S.
- Multiple *game turns* t, each has possible moves $m_{t,j}$.
- For a given turn (i.e., chess position) t, legal moves are $m_1, \dots, m_j, \dots, m_J$ (index t understood).
- Moves indexed by values $v_1, \dots, v_J$ in nonincreasing order.
- Values determined by strong chess programs. Not apprehended fully by P (*bounded rationality*, *fallible agents*).
- Raw utilities $u_j = \delta(v_1, v_j)$ by some difference-in-value function δ in either "pawn units" or "chance of winning" units.
- Parameter β treated as a divisor s of those units, i.e., $\beta = \frac{1}{s}$.

Chess Decision Setting

- One player P with characteristics S.
- Multiple *game turns* t, each has possible moves $m_{t,j}$.
- For a given turn (i.e., chess position) t, legal moves are $m_1, \dots, m_j, \dots, m_J$ (index t understood).
- Moves indexed by values $v_1, \dots, v_J$ in nonincreasing order.
- Values determined by strong chess programs. Not apprehended fully by P (*bounded rationality*, *fallible agents*).
- Raw utilities $u_j = \delta(v_1, v_j)$ by some difference-in-value function δ in either "pawn units" or "chance of winning" units.
- Parameter β treated as a divisor s of those units, i.e., $\beta = \frac{1}{s}$.
- Second parameter c allows nonlinearity: $\delta(v_1, v_i)^c$. (First $c = 1$.)

Chess Decision Setting

- One player P with characteristics S.
- Multiple *game turns* t, each has possible moves $m_{t,j}$.
- For a given turn (i.e., chess position) t, legal moves are $m_1, \ldots, m_j, \ldots, m_J$ (index t understood).
- Moves indexed by values $v_1, \ldots, v_J$ in nonincreasing order.
- Values determined by strong chess programs. Not apprehended fully by P (*bounded rationality*, *fallible agents*).
- Raw utilities $u_j = \delta(v_1, v_j)$ by some difference-in-value function δ in either "pawn units" or "chance of winning" units.
- Parameter β treated as a divisor s of those units, i.e., $\beta = \frac{1}{s}$.
- Second parameter c allows nonlinearity: $\delta(v_1, v_i)^c$. (First $c = 1$.)
- MNL model (called "Shares" by me) then equivalent to:

$$\log(p_j) = U_j = \left(\frac{\delta(v_1, v_j)}{s}\right)^c$$

and we go as before.

Chess Decision Setting

- One player P with characteristics S.
- Multiple *game turns* t, each has possible moves $m_{t,j}$.
- For a given turn (i.e., chess position) t, legal moves are $m_1, \ldots, m_j, \ldots, m_J$ (index t understood).
- Moves indexed by values $v_1, \ldots, v_J$ in nonincreasing order.
- Values determined by strong chess programs. Not apprehended fully by P (*bounded rationality*, *fallible agents*).
- Raw utilities $u_j = \delta(v_1, v_j)$ by some difference-in-value function δ in either "pawn units" or "chance of winning" units.
- Parameter β treated as a divisor s of those units, i.e., $\beta = \frac{1}{s}$.
- Second parameter c allows nonlinearity: $\delta(v_1, v_i)^c$. (First $c = 1$.)
- MNL model (called "Shares" by me) then equivalent to:

$$\log(p_j) = U_j = \left(\frac{\delta(v_1, v_j)}{s}\right)^c$$

and we go as before. Taking $\log(p_j) - \log(p_1)$ on LHS gives same model.

Alternative "Loglog-Linear" Model

Represent a difference in *double logs* of probabilities on left-hand side instead.

Alternative "Loglog-Linear" Model

Represent a difference in *double logs* of probabilities on left-hand side instead. Now nice to keep signs nonnegative by inverting probabilities.

$$\log\log(1/p_j) - \log\log(1/p_1) = \beta U_j$$

Alternative "Loglog-Linear" Model

Represent a difference in *double logs* of probabilities on left-hand side instead. Now nice to keep signs nonnegative by inverting probabilities.

$$\log\log(1/p_j) - \log\log(1/p_1) = \beta U_j$$

The β can be absorbed as $(\frac{1}{s})^c$ even when $c \neq 1$ so my nonlinearized utility still fits the setting.

Alternative "Loglog-Linear" Model

Represent a difference in *double logs* of probabilities on left-hand side instead. Now nice to keep signs nonnegative by inverting probabilities.

$$\log\log(1/p_j) - \log\log(1/p_1) = \beta\, U_j$$

The β can be absorbed as $(\frac{1}{s})^c$ even when $c \neq 1$ so my nonlinearized utility still fits the setting. Then abstractly:

$$\begin{aligned}
\frac{\log(1/p_j)}{\log(1/p_1)} &= \exp(\beta\, U_j) =_{def} L_j \\
\log(1/p_j) &= \log(1/p_1) L_j \\
\log(p_j) &= \log(p_1) L_j \\
p_j &= p_1^{L_j}.
\end{aligned}$$

Analogy to power decay, Zipf's Law... *Proceed to demo.*